An experimental evaluation of different proposals for teaching Prolog that is designed to be run over a network.

Mike Brayshaw[1], Helen Pain[2], Paul Brna[2], Andrew Bowles[2], and Dave Robertson[2]

[1] Human Cognition Research Laboratory, The Open University, Milton Keynes, England, MK7 6AA., Tel: [+44] (0)908 65 5015, FAX: [+44] (0)908 65 3169, email: m.c.brayshaw@open.ac.uk

[2] Department of Artificial Intelligence, University of Edinburgh, 80 Southbridge, Edinburgh, Scotland.

Extended Abstract for PPIG-6, Psychology of Programming Workshop, The Open University, 6th.-8th. January, 1994.

1. Introduction.

The paper will introduce an approach for designing and running experiments that address curricula issues for Prolog teaching. The primary focus of the paper will be a discussion of an experimental evaluation of four different approaches for teaching Prolog. As a secondary issue, we intend this experiment to be run remotely over computer networks and we will discuss some of the practical issues and design problems this raises.

The particular domain of interest is how we can improve the teaching of logic programming. It's been noted elsewhere (e.g. Taylor, 1984, 1987;Ormerod, 1987) that languages like Prolog seem to present a particular obstacle to novice programmers. The unification algorithm and backtracking in particular lend themselves to a series of misconceptions (Fung et al, 1981). This in turn has led to a number of proposals about how we can improve the learning experience for novices (e.g. Rajan (1986), Gregg-Harrison (1991), Looi (1988), Bundy et al (1986), and Eisenstadt and Brayshaw (1991)). Here we wish to complement the above work to see if there are particular parts of the teaching material that we present to novices that can be modified in a specific way to help novices learn and to alleviate misconceptions. Two proposals in particular will be investigated (a) teaching novices well known programming methods, and (b) attempting to improve the mapping between course text and exercise material.

2. Motivation

The subject of the evaluation are two proposals for teaching Prolog. Starting from a curricula which attempts to present a "typical" Prolog course, we will seek to evaluate the effects of these proposals. The first is the concept of a Prolog technique (Brna et al, 1991). By technique we mean a common way of writing things in Prolog. This corresponds to a surface cliché in the code. Please note that no claim is being made about the psychological depth of such a concept, and no link is being made to the "Yale" notion of plans. We are merely saying that these are common ways that Prolog programmers encode their problem solutions. If you like their "tricks of the trade". By extrapolation it is argued that if it's clear proficient programmers use them, why should we not teach them explicitly to novices and see if their performance improves.

The second area of concern for Prolog teaching is the arguments for a rigorous treatment of isomorphism between examples in the course text and set exercises. Robertson and Kahney (1993) have demonstrated that novices are frequently thrown by the lack of isomorphism between course text examples and subsequent exercises and assignments. As Keane et al (1989) argued, this lack of isomorphism can often be subtle and unintended by the courseware author, but nonetheless fatal to the analogical mapping process of the novice. Further Robertson and Kahney (ibid.) note novices often are able to identify some surface features and map then correctly in similar conditions, but lack the deep model of the relation between the example and exercise when presented with a more different variant. Thus in this experiment we were anxious to also see if strict and explicit mapping of information between example and exercise has any effect on the performance of learners.

3. The Experiment.

The experiment looks to use Hypercard™ as the basis for the experimental materials. There are two justifications for this, one educational, the second practical. Firstly, the growing number of educationally based Hypercard stacks already demonstrate the technology's potential, not least for distance education. Here we aim to combine this potential with the opportunity it affords us for empirical study of novices. Secondly, this allows us to mail out a complete experiment to an already large potential audience of Macintosh based users.

The experiment itself follows a Latin Square design. The control case is an introductory Prolog course that aims to be typical in its curricula. No attempt is made to emphasise structural isomorphism or Prolog techniques. Two variations feature respectively (a) The Techniques Condition: the introduction of the concept of techniques and how to use them and (b) The Explicit Mapping Condition: contains explicit structural mappings between exercises and problems. Finally in (c) The Combined Condition, we integrate these two proposals and present techniques alongside mapping information.

The core curricula introduces the itemised concepts in the following order.

facts.
facts with arguments
variables and unification
rules
search including backtracking
recursion
lists
lists and unification
listing search (list deconstruction)
list building (list construction)

Each concept is introduced and both procedural and declarative readings are discussed. The structure of the dialogue is

(a) exposition of concept;

(b) a worked example(s);

(c) an exercise.

If the exercise result is better than a predetermined figure, then the student moves on to the next card. If not they are offered an extra remedial card and it is suggested that they also go back and study further cards on this topic.

Subjects are first given a questionnaire about their background and experiences. This includes their age, number of years experience with computers, and details of what other languages they know and a rating of their proficiency in them. They are randomly assigned to one of the four experimental conditions. Once in a condition they are asked how much Prolog they already know. If they are a total novice, then they are assigned to the start of the tutorial. If not, they are given a series of graded exercises in order to determine their proficiency, and then placed into the tutorial at an appropriate spot. This is recorded so that we can correlate perceived proficiency, observed proficiency, and effects that the conditions had on particular subject populations. All movements between cards are monitored, as are responses or changes of answers to questions. The subjects are allowed to wander forwards and backwards through the cards at will.

In addition to raw performance on exercise tasks, at the end of the learning episode we present a series of short experimental tasks. These include specific problems to solve as well as fixing and explaining bugs. A comprehension task based on an adaptation of Kahney's (1983) coloured pens experiment is also included. This requires students to copy a program. They must study the program on one card, then move to another and write it down. Each move and changes to the transcribed code are recorded. This allows us to see how novices are chunking the information they transcribe. Of interest is if novices exposed to techniques or mapping role fillers use these concepts as the basis for how they copy the program, rather than doing it in a purely syntactic left-to-right manner.

4. Running the Experiment, Experiences, and Results.

With the advent of massive new interconnectivity via email and the Internet we have become interested in using these facilities as a means of interacting with experimental subjects. A major potential benefit of this approach is access to a large population base of subjects, an important issue in the psychology of programming where individual differences plays such an important role. However, this raises a series of issues for experimental content and structure, including: how it is to be distributed, how we persuade net users to take part, how we measure their performance, and not least, how we get the results back! The way we approached these problems was as follows:

The experiment itself was shipped late 1993. This included posting to the News Net, BIX, and CompuServe as well as individual email postings. The posting were of two types. The first was to offer software on an individual self improvement basis as a "Free introduction to Prolog". The deal is "We'll give you a taster (for free) of what Prolog is like and show you round a bit of it. However, we're also looking at how to improve tutorials like this so we'd like to know how you got on. You can do this by emailing the log file back to us. If you do so we'd be most grateful and you'd be helping others in future." The second offer was made to teachers. It said that if you had a Prolog class to teach in the next term here was a free tutorial that took them as far as lists and recursion. They were free to use it in a classroom setting, but would they please send us the results on how the students got on. Finally we sent a version of this second deal out to various colleagues who we'd forewarned and who had agreed to either be a subject themselves or use it on their own classes. Note was made of the method each subject got the tutorial and how they used it (alone or in class).

Due to timescale consideration, the paper will not report results as the experiment will still be running in January 1994. However, we should be able to report initial responses and have some feel for the take up. Further we will discuss what comments we have got back and doubtless note further practical hitches that we encountered.

5. Conclusions

The experiment looks to see what effects various proposals have on a specific method of teaching Prolog. In doing so we'd had to make assumptions. The actual pedagogy of the tutorial itself, the use of Hypercard™, and the use we have made of it, are all, to some extent, open issues. The experiment here looks to see if these proposals work with the types of tutorial techniques that we have used here. Depending on what results we get will allow us to speculate how this might alter if we changed our tutorial format. Indeed, this speculation may even turn into another experiment!

6. References

Brna, P.,Bundy, A., Dodd, A., Eisenstadt, M, Looi, C-K, Pain, H, Robertson, D., Smith, B, and Van Someren, M, Prolog programming techniques, Instructional Science, **20**(2/3), 1991.

Bundy, A., Pain, H., Brna, P., and Lynch, L. A proposed Prolog story. *DAI Research Paper 283*, Department of Artificial Intelligence, University of Edinburgh, 1986.

Eisenstadt, M. and Brayshaw, M. A Fine-Grained Account of Prolog Execution for Teaching and Debugging. *Instructional Science*, **16**, pp. 407-436, 1990. *ISSN 0020-4277.*

Fung, P., Brayshaw, M., DuBoulay, J.M.B., and Elsom-Cook, M.T., A Taxonomy of Novices misconceptions of the Prolog Interpreter. *Instructional Science,* **19**, pp. 311-336, 1990.

Gegg-Harrison, T. Learning Prolog in a schema-based environment, *Instructional Science*, **20** (2/3), 1991.

Kahney, J. An In-depth study of the Cognitive Behaviour of novice programmers, *Technical Report No. 5* (Ph.D. Thesis), Human Cognition Research Laboratory, The Open University, Milton Keynes, 1982

Keane, M., Kahney, H., and Brayshaw, M. Simulating Analogical Mapping Difficulties in Recursion Problems. In A.G. Cohen (Ed.), *Advances in Artificial Intelligence: Proceedings of AISB-89.* London: Pitman, 1989.

Looi, C.K., Analysing novices' programs in a Prolog intelligent tutoring system. *Proceeding European Conference on Artificial Intelligence ECAI-88*, Pitman (London), pp.314-319, 1988.

Omerod, T. Content and representation effects with reasoning task in Prolog form, *Behaviour and Information Technology*, **5**(2), pp. 157-168., 1987.

Rajan, T. APT: A Principled Design for an Animated View of Program Execution for Novice Programmers. *Technical Report No. 19 (Ph.D. Thesis)*, Human Cognition Research Laboratory, The Open University, Milton Keynes, 1986.

Robertson, I. and Kaheny, J.H. How do examples Help Solvers Solve Problems? An Interpretation Theory for Textual Analysis, Technical Report 96, Human Cognition Research Laboratory, The Open University, Milton Keynes, England, MK7 6AA,

Taylor, J. Why novices will find learning Prolog hard. Procedings of the European Conference on Artificial Intelligence (ECAI-84), Pisa, 1984.

Taylor, J. Programming in Prolog: an in-depth study of problems for beginners learning to program in Prolog. Unpublished Doctoral Dissertation, School of Cognitive Studies, The University of Sussex, Brighton, UK., 1987.