Logic Programming as an Introductory Programming Paradigm

Juris Reinfelds and Gopal Gupta

Department of Computer Science
New Mexico State University
Las Cruces, NM 88003-0001
USA

Abstract

This report contains contributions from the speakers who spoke at the birds-of-a-feather session organized by us to discuss the suitability of *Logic Programming as an Introductory Programming Paradigm*. The session was held on Tuesday, Nov 10th, during the Joint International Conference and Symposium on Logic Programming 1992 in Washington D.C. The contributions of individual speakers (in the form of copies of transparencies they used in their talks) are preceded by a brief summary of the session proceedings. This summary also appears in the February issue of the Logic Programming Newsletter **6**(1) pp. 5-7. We would like to acknowledge the support of our colleagues at NMSU who are working with us on the NSF curriculum development project.

Report on Birds-of-a-feather Session on

Logic Programming as an Introductory Programming Paradigm

Juris Reinfelds and Gopal Gupta

Department of Computer Science
New Mexico State University
Las Cruces, NM 88003-0001
USA

A birds-of-a-feather session on using Logic Programming as an Introductory Programming Paradigm for Computer Science Majors was held at the Joint International Conference and Symposium on Logic Programming (JICSLP) 1992 in Washington D.C. A room with about 40 chairs was provided by the conference organizers. The room became full soon after the start of the session with five to ten people 'hovering' around the door.

The session started with opening remarks from Juris Reinfelds of New Mexico State University who introduced the topic and then went on to explain the importance of declarative thinking in teaching programming and the reasons why he thinks Logic Programming is a more suitable declarative candidate for a first language. The primary reason he gave for suitability of LP as a first language were that it provides a better sequencing of Computer Science concepts, so that students can start writing meaningful programs right after day 1 of class after having seen the famous family database program. Its computation model needs no reference to a model of computer hardware. The mastery of small set of concepts permits solving of interesting problems, and, it provides a gradual transition from concrete to abstract concepts in problem solving. Declarative (Logic) programming permits a clear separation of the concern for a precise specification of the solution of a problem from the concern for an efficient implementation of that solution. Problem solving becomes a genuine three step process: from problem description to specification to program.

Juris's opening remarks were followed by a short presentation by Gopal Gupta of New Mexico State University who presented the issues that needed to be addressed in the session, namely: Is Logic Programming a good candidate for teaching introductory programming, or would Functional Programming or some other programming paradigm serve just as well or better? Do students need more mathematical maturity for Logic Programming than for Functional or Imperative Programming? How can we judge the success of an enterprise that attempts to use LP as a first language? And most important, what flavor of LP should be adopted for teaching to beginners (since Prolog, the most widely available LP language, is not truly declarative and logical)? Gopal then went on to explain that Logic Programming is directly and indirectly related to many diverse fields in Computer Science, such as Compiling and Parsing, Databases, Symbolic Computing, AI and Expert Systems, Specification Languages in Software Engineering, etc. and therefore is an ideal paradigm for teaching introductory programming since it will make teaching of advanced courses later much easier (It was pointed out later by a participant from Uppsala University, Sweden, that at Uppsala they were indeed using Logic Programming for teaching many of the advanced courses in CS).

The first speaker was John Lloyd of University of Bristol, who argued that to use LP as an introductory paradigm we need a language that is "better" than Prolog. This ideal language must

be declarative, must support Types and Modules, must have autonomous control of search as much as possible, and should support declarative debugging. He went on to present his new language for teaching and research called Goedel, which possesses all these properties, in addition to many others. (A prototype implementation of Goedel can be obtained by anonymous ftp from Bristol.)

The next speaker was Professor Bharat Jayaraman of SUNY-Buffalo. He took the stand that there is no need for using LP in teaching introductory programming and that Functional Programming will do the job better (By the way, Bharat is a believer in both Functional as well as Logic Programming. He was playing the Devil's advocate at the request of the organizers; not many Functional Programming researchers attend LP conferences). His reasons for favoring FP were mainly that the use of Logic Programming will require mathematically unsophisticated students to master the concept of relations, which he thinks is more complex than the notion of functions. Also the students are exposed to functions more than they are to relations in their previous training. Bharat also said that Logic Programming provides a very high level of abstraction, and too much of an abstraction is not always a very good thing. Bharat compared Logic Programming to a fine wine which can only be enjoyed if the patron is old enough. Bharat pointed out that we should not look to predicate Logic Programming only, there are other forms of programming logics as well—constructive logic, equational logic, subset logic, linear logic, etc. Bharat feels that Functional Programming is more natural at an introductory level, and that it encourages good programming habits. Relational programming, on the other hand, is less natural, since search, constraints, etc. are too sophisticated for beginners; and implicational reasoning is harder to understand than equational reasoning.

Bharat's talk generated some interesting discussion. Juris argued that Logic Programming was a good vehicle too, probably better, because it provides better sequencing and separation of concepts of Computer Science. Students can start writing programs immediately and start putting into use what they have learned immediately.

The next speaker was Juris Reinfelds who told the audience about the experiment he and his colleagues in Computer Science and Mathematics are conducting at New Mexico State University, where a co-ordinated pair of courses is being taught jointly by the Maths and Computer Science Department under an NSF Curriculum Development Grant. The Computer Science department is teaching the introductory programming course to freshmen using Logic Programming, and the Maths Department is teaching a course in Discrete Mathematics to these same students that stresses on mathematical concepts needed in the Computer Science course. The two courses co-ordinate with each other very closely, so that when the CS class is introduced to *facts* in LP, the Math class concurrently teaches Sets, Relations and Functions; when the CS class is covering *rules* in LP, the maths class teaches syllogisms, when the CS class teaches recursion, the Math class covers induction, etc. Juris told about his experience in teaching the Computer Science course to a group of honors students at NMSU during last spring. He felt that the results both from the students' as well as his point of view were very positive. Juris also said that what people at NMSU had in mind was to teach a set of two Computer Science Courses, the second being the sequel to the introductory one. The sequel course should teach more advanced concepts in Logic Programming, Functional Programming and enough Imperative Programming so that when students do more advanced courses, that assume knowledge of Imperative Programming, then they will not be at a

loss. This is based on the belief that the students should be taught Imperative Programming at some point (but not right in the beginning).

Juris's talk was followed by J. Mack Adams of New Mexico State University. J. Mack told the audience about the course he was currently teaching to regular Computer Science students as part of the NSF Curriculum Development project at NMSU. Most of these computer science students were also doing the sister course in Discrete Mathematics. He discussed the contents of his course, saying that in the first two weeks he discussed the structure of the electronic computer and introduced a little bit of Imperative Programming. For the bulk of the semester he covered Logic Programming, trying to avoid Prolog specific things as much as possible. J. Mack complained about the lack of tools for visualizing execution of Logic Programs. His students had used tools available from the Open University of England, but J. Mack felt these tools were not adequate enough. Towards the end of the course he covered a large example using Logic Programming based on writing a compiler for a simple imperative language. He also taught some Modula-2 in the beginning and end of the course, not only for the benefit of those who were going to take other advanced courses in Computer Science that required knowledge of Modula-2, but also for the reasons mentioned by Juris. J. Mack's experience of using Logic Programming in the course was very positive too, and his feeling was that the using Logic Programming was a viable option.

The second last speaker was Owen Astrachan of Duke University. Owen had a very provocative title for his talk "Logic Programming considered harmful?" He expressed doubts about whether indeed Logic Programming was a good candidate for an introductory paradigm. He expressed his reservations for using Prolog (lack of encapsulation facility, typelessness, extra-logical constructs, etc.) as an introductory language. He argued that most college/university students are exposed to programming in High School, and invariably the language used is imperative. Therefore, we should stay with what's comfortable for the students, and rather focus our attention on breaking bad habits that students may have picked up during initial exposure. He argued that students find it difficult to appreciate elegance (reinforcing Bharat's point that Logic Programming is like a fine wine ...). Owen stated that Logic Programming should only be used as an introductory paradigm in institutions where adherents and experts of LP reside, since there is a possibility that a non-expert may not teach it so well because he/she is not a believer in LP. He felt that if such an enterprise is to succeed, then our first task should be to convert our colleagues to the benefits of Logic Programming.

The last speaker was Graem Ringwood of Queen Mary and Westfield University of London. Graem primarily tried to addressed the issue of judging the success of using one paradigm versus another for teaching introductory programming. He spoke about an experiment he and his colleagues were doing where they were observing the progress of two sets of students who had been taught introductory courses using different paradigms. Their intent is to see if the difference in paradigm has any effect on subsequent performance.

As the protocol of a birds-of-a-feather session requires, members of the audience then introduced themselves and contributed their views or experience on teaching Logic Programming to beginners. Many interesting views and ideas emerged. Overall one can perhaps summarize the general feeling that although most participants were aware of the swing towards declarative programming in the form of Functional Programming, few had given serious thought to the building

of a first course around the database like computational model of Logic Programming. Most were intrigued by the idea, however. Many expressed doubts about the extensive extra-logical baggage of Prolog. A good teaching language that is 'more' declarative than Prolog would probably convert many, and we can only hope that Goedel will provide such an alternative. As the meeting concluded many participants went away wondering how Logic Programming could be best applied to shape the thinking of future computer scientists at an early stage.

A technical report containing this report and copies of overhead transparencies used by speakers is available from NMSU. We think that it will be worthwhile to further discuss the issues raised in the session, perhaps on the usenet and/or through another such session at a future Logic Programming Conference, perhaps ICLP'93.