Undergraduate AI and its Non-imperative Prerequisite*

Deepak Kumar
Department of Math & Computer Science
Bryn Mawr College
Bryn Mawr, PA 19010
(610) 526-7485
dkumar@cc.brynmawr.edu

Richard Wyatt
Department of Math & Computer Science
West Chester University
West Chester, PA 19383
(610) 436-3230
wyatt@checkov.wcupa.edu

Introduction

This position paper presents, we believe, a strong case for including in an undergraduate Computer Science (CS) program a required course in non-imperative programming. Such a course is motivated by the need to provide a uniform and adequate background for a course in Artificial Intelligence (AI). We make no claim about graduate AI courses. We survey the recommendations of the ACM-IEEE Joint Curriculum Task Force for an undergraduate course in AI and argue that some of the "dilemmas" of teaching introductory AI courses can be resolved by changes in the core CS curriculum. The argument for such a course is the focus of the paper; we also offer some ancillary remarks on other matters relating to teaching AI at the introductory level.

West Chester University and Bryn Mawr College are in many respects rather different schools (see Appendix). Even so, the problems encountered in developing a successful AI program, and the resulting programs themselves, are remarkably similar. We think that our separate experiences have benefited each other. The aims of our AI courses are:

- To expose students to the general domain of AI.
- To set their horizons fairly high so that they have a grasp of "what is out there".
- To provide some exposure to "real" AI which students, it is hoped, can make use of in the "real world".

AI in the CS Curriculum

It has been argued that an undergraduate course in AI could play a fundamental role in the CS curriculum (Aiken 1991). The recent ACM-IEEE Joint Curriculum Task Force Report (Tucker 1991) recommends that a core undergraduate curriculum include nine lecture hours on AI, consisting of history of AI, applications, state spaces and search strategies, and offers sample curricula which includes an advanced or supplemental AI course described thus:

> *Topic Summary:* A selective survey of key concepts and applications of artificial intelligence, and an in-depth experience with a language commonly used for building AI systems (e.g., Lisp or Prolog).
>
> Subtopics include knowledge representation, state space searching, heuristic search, expert systems, expert system shells, natural language processing, propositional logic and cognitive models, and vision.
>
> *Suggested Laboratories:* Students will implement, modify, or enhance several AI systems using an AI language and associated tools (e.g., expert system shells, knowledge acquisition tools).
>
> *Prerequisites:* AL1–AL3, AI1, AI2, PL11, SE1, SE2, Discrete Mathematics. (Tucker 1991, page 71)

The only Knowledge Unit (KU) in the prerequisites which bears on AI is PL11, a Programming Paradigms KU described thus:

> Introduction to alternative programming paradigms (e.g. functional, logic, and object-oriented) and languages (e.g. Lisp, Prolog, and SmallTalk, respectively). Program construction using at least two of these paradigms. Advantages and disadvantages *vs.* the procedural paradigm. (Tucker 1991, page 63)

The suggested laboratory work for PL11 includes developing short programs in languages exemplifying the

*From ACM SIGART Bulletin Special Issue on AI Education, Kumar &Hearst (editors), Forthcoming in Spring/Summer 1995

three paradigms. In the sample curricula, this component is usually included in the course on Programming Language Concepts and consists of approximately 10 hours of lectures; ten hours of lectures for functional, logic programming *and* object-oriented programming.

The Language Barrier

The following problems, in our view, seriously affect how an AI course is to be designed in relation to the background programming language.

1. Some schools ignore the non-imperative languages altogether and usually end up offering an AI course which might best be described as pseudo-AI. Other schools transfer the responsibility of learning Lisp or Prolog to the AI course itself. This confuses AI with the means of implementation. While Lisp and Prolog (and, for that matter, object-oriented programming) have their roots in AI, they are no longer considered "AI Languages". Lisp is *an* example of a functional programming language, at least in its pure form, and Prolog of logic programming.

2. The ACM-IEEE guidelines, which recommends ten hours of lectures, do not suffice to provide the needed background in alternative programming paradigms. For example, in a Lisp based AI course which relied on such a background course, the students could be expected to have had a mere four or five hours of prior language instruction. This is entirely inadequate for all but the most trivial of purposes. Moreover, devoting time to functional and logic-based languages in a Programming Language Concepts course takes time away from the important task, which is the focus of such a course, of examining in detail the run-time structure of programming languages. Most of this examination traditionally involves imperative (procedural) languages. Rather than additionally learning the run-time structure of functional, logic, and object-oriented languages, one is forced, if one follows the ACM-IEEE guidelines, to learn —or merely be exposed to —just their syntax.

3. In most first courses in AI, some students start out with an elementary knowledge of Lisp or Prolog. But many do not; many have never seen a single non-imperative language. If the course is pitched to the former, it tends to exclude the latter, but if pitched to the latter, the former are invariably "under" taught. Though such issues arise with all classes, it is especially acute in AI because adequate instruction in non-imperative languages is still not mainstream. In fact, many CS programs graduate students without any knowledge of functional or logic programming paradigms at all.

4. When a serious background course more or less devoted to non-imperative languages is offered, it is usually an upper-level elective course. Thus, by the time AI is offered, many students who are otherwise interested in taking the course discover that they are unable to do so because they have not taken the prerequisite. Again, though this issue arises with most courses, it is especially acute in AI. Students, many of them, have a narrow, even naive, view of the discipline. They tend to prefer courses such as imperative programming, networking, and operating systems, rather than those such as theory of computation, non-imperative programming, and AI. Part of the reason, we suspect, is that many students focus too much on the perceived extent to which their courses will afford immediate and clear skills of interest to industry, rather than the longer term benefits of a thorough understanding of the discipline. In any case, students often will not choose as an elective a course in non-imperative programming and, if AI were to require such a course as a prerequisite, would thus find themselves, perhaps inadvertently, excluded from AI.

A Solution

Though an undergraduate introduction to AI should concentrate mainly on the concepts relevant to AI, it should also provide a considerable exposure to implementation matters. The needed background for the implementation component is to be provided by a course in non-imperative programming. Such a course should be offered after the introductory imperative (or imperative augmented with OOP) course(s), should preferably be offered at the freshman or sophomore level, and should be required of all CS majors.

Such a course would maintain a sharp distinction between non-imperative languages and AI, permit a conventional programming language course to focus on the run-time structure of imperative and non-imperative languages, and provide all students with an adequate, and by and large equal, background from which they may proceed, if desired, to AI.

Such a course is also entirely justified on independent grounds. Exposure to different programming paradigms, functional and logical, broadens ones outlook in much the way that exposure to foreign languages broadens one's understanding of English. By seeing how different languages tackle and solve similar problems, a far grater understanding of the issues, both theoretical and practical, is developed. Such a course also provides intensive exposure to recursion, which is very often a weakness in beginning students.

A First Course in AI

The Language Prerequisite

Elementary knowledge of the programming languages used for the laboratory component of the course should be a prerequisite. This requirement should be fulfilled by a course on non-imperative programming paradigms. The course should, preferably, be offered at the freshman or sophomore level (post CS1) and should be required of all majors.

The Core Concepts

In an early draft of a recently published introductory AI text, the author claimed that despite the apparent diversification in the field of AI there is only a small set of core ideas, which can be successfully presented in less than 300 pages. By the time the book was published, however, it was close to 450 pages; and much was still omitted.

There is, of course, no clear core to AI. There is a considerable range of topics, most selections from which, would yield an interesting and worthwhile course. There is one caveat: most undergraduate students do not go on to complete a Ph.D. or even a Master's degree. Students have a legitimate interest in acquiring skills that make them employable. To this end, some exposure to Expert Systems, Neural Networks, NLU Systems, and so on, is usually in order as they become, and to the extent that they are, commercially viable.

The Lab Component

The laboratory and homework assignments should include:

- At least one programming assignment in which students design and implement a non-trivial AI algorithm or program from scratch.
- At least one assignment in which students use a "real" AI system.
- At least one assignment in which students use an AI tool to improve or modify an AI system.
- Several demonstrations of existing AI programs or systems.

Ancillary Remarks

Breadth vs Depth: There are too many topics to hope to cover them all adequately in a single course. A mix of depth and breadth seems to work well, there being considerable flexibility as to which topics should be covered in depth. No topic, no doubt, *must* be covered in depth.

AI Background: Because AI involves knowledge of such things as philosophy, cognitive science, linguistics, psychology, logic, etc, undergraduate students rarely have enough of a background to tackle AI problems without at least rehearsing such material. But if one is not careful, one can spend a great deal of time teaching the background knowledge to AI rather than AI itself. But some background is unavoidable. One must balance the additional time the background to a topic takes against the merits of the topic itself.

Philosophical Matters: Is it really intelligence, or just a simulation? Is it still AI if we know how to do it? Is our approach motivated by cognition? We should not be concerned with these matters in introductory AI. Such matters are best dispatched in short order; there is little enough time for the meat of AI. We favour a view of AI which is neutral on such issues.

Formalism: Formalism is an essential aspect of AI, and indeed, of many advanced learning domains. AI students should therefore be exposed to it and have its importance, at the advanced level, impressed upon them. But most undergraduate CS students are not gifted mathematicians nor formalists, and so the use of formalism should be limited.

History: Students should be exposed to a brief history of AI, though this is no doubt best intermixed with the course, rather than treating it as a separate topic. Some coverage of classical AI programs should be included.

References

Aiken, R. M. 1991. The New Hurrah: Creating a Fundamental Role for Artificial Intelligence in the Computing Curriculum. *Education & Computing* 7:119–124.

Tucker, A., ed. 1991. *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force.* ACM Inc.

Appendix

The following outlines the nature of our schools and CS programs, which, we believe, has influenced our views.

Bryn Mawr College (BMC) is a small women's liberal arts college located 11 miles west of Philadelphia. The college has about 1200 undergraduates and 500 graduate students in sixteen degree programs. The Computer Science program at BMC is currently being developed in cooperation with Haverford College. The bi-college computer science program currently consists of three CS faculty and several affiliates from other departments. The current enrollment in computer science is about 25 students some of which are pursuing interdisciplinary degrees in cognitive science.

West Chester University (WCU) is the second largest of the fourteen state institutions in the Pennsylvania State System of Higher Education. It is located about thirty miles west of Philadelphia. The university offers the bachelor's and master's degree in a wide variety of subjects. The student population, which consists of about 10,000 undergraduate students represents a wide range of demographic groups (There are also about 2,000 graduate students.). The majority of the students are Pennsylvania residents, but most other states and several foreign countries are also represented. The Computer Science group, which is about to form a new department, consists of eight faculty. Approximately 140 students are currently enrolled as undergraduate majors in computer science (There are also about 70 students in the graduate program.). The undergraduate program is currently being upgraded to conform, more or less, to the Computing Curricula '91(Tucker 1991).