Teaching AI and Philosophy at School?

Aaron Sloman
`http://www.cs.bham.ac.uk/~axs/`
School of Computer Science
The University of Birmingham

Contents

1 Introduction: We Need Something Different

This paper proposes a way of teaching computing, not as a branch of engineering, but as a way of learning to do philosophy, cognitive science, psychology, linguistics, and biology, among other things. It could be the core of a new kind of liberal education. But what I am proposing is not new and untried – what is proposed is close to the spirit and philosophy of teaching programming and AI to complete beginners, which some of us developed at Sussex University from the mid 1970s onwards. A revival of that approach might address a serious current malaise. The vision presented here overlaps with that in Jeannette Wing's (2006), but has a different emphasis.

In the UK there has been much discontent in recent years about the teaching of computing in schools, not least because many bright learners form the impression that the study of computing is simply a matter of learning to use tools that everyone needs to learn to use, but without intellectual content of a type that could make it a subject worthy of study at university level. A similar view of chemistry might be produced if chemistry were taught mainly by teaching cooking. That is what has happened in schools by switching from the early experiments in teaching children to design, test, debug, and describe computer programs to teaching them only how to use word processors, email systems, web browsers, and possibly databases, spread-sheets and other tools – like attempting to teach physics by teaching pupils to drive cars and buses. This switch completely defeated the vision I wrote about in 1978:

> *"Another book on how computers are going to change our lives? Yes, but this is more about computing than about computers, and it is more about how our thoughts may be changed than about how housework and factory chores will be taken over by a new breed of slaves.*
>
> *Thoughts can be changed in many ways. The invention of painting and drawing permitted new thoughts in the processes of creating and interpreting pictures. The invention of speaking and writing also permitted profound extensions of our abilities to think and communicate. Computing is a bit like the invention of paper (a new medium of expression) and the invention of writing (new symbolisms to be embedded in the medium) combined. But the writing is more important than the paper. And computing is more important than computers: programming languages, computational theories and concepts – these are what computing is about, not transistors, logic gates or flashing lights. Computers are pieces of machinery which permit the development of computing as pencil and paper permit the development of writing. In both cases the physical form of the medium used is not very important, provided that it can perform the required functions.*
>
> *Computing can change our ways of thinking about many things, mathematics, biology, engineering, administrative procedures, and many more. But my main concern is that it can change our thinking about ourselves: giving us new models, metaphors, and other thinking tools to aid our efforts to fathom the mysteries of the human mind and heart. The new discipline of Artificial Intelligence is the branch of computing most directly concerned with this revolution. By giving us new, deeper, insights into some of our inner processes, it changes our thinking about ourselves. It therefore changes some of our inner processes, and so changes what we are, like all social, technological and intellectual revolutions."*
> (From the preface.)

In pursuit of that dream, for many years (starting in 1974, in a team led by the late Max Clowes (Sloman, 1984b)), I was involved with teaching AI to novice university students; initially only students in humanities and social science disciplines, though later also students majoring in computing or AI. (NB: In those days hardly any first year students had even used a typewriter, let alone a computer.)

The idea was not that AI had produced theories about how minds worked, but that it provided a new way of thinking about what form good theories might take – e.g. not theories about necessary and/or sufficient conditions for some mental state to exist, as favoured by most philosophers, nor theories about correlations between stimuli and behaviour, as favoured by most psychologists, nor theories about the advantages that might have led to the evolution of particular competences and traits, as favoured by some evolutionary theorists, but theories about what minds could *do* and *how* they might do it, sought by designers and a few philosophers, e.g. Immanuel Kant.

Students learnt, from first hand experience of simple examples, about how explanatory theories can be developed, tested, debugged, extended, and in some cases used to generate new empirical research leading to better theories. My assumption was then (as explained in chapter 2 of the 1978 book) and remains now, that the alleged distinction between philosophy as a purely conceptual study and science as empirical was mistaken: when done well, philosophy and science overlap substantially. A well-known example is Einstein's thinking leading up to the special theory of relativity, using ideas from Hume and Mach (Sloman, 1978 Chap3; Norton, 2005). Computers provided powerful new tools to extend that overlap.

The approach to teaching, inspired by that vision, was developed with colleagues in the early years at Sussex University, from the mid 1970s and continued after I moved to Birmingham in 1991. We did not teach

AI primarily as a branch of engineering, but as a way of trying to understand human (or more generally animal) competences, though with potential applications to engineering. No claim was made (by us) that the problems were close to being solved, or that human-like robots would soon be available (Sloman, 1978, Sec. 9.13, Chap. 10).

For some learners, their first ever programming exercise used the "riverworld" library with pre-built commands, such as 'putin(X)', 'takeout(X)', 'getin()', 'crossriver()', 'getout()' to instruct the computer to get a farmer, fox, chicken and grain across a river, using a boat that could contain only two things, while avoiding ever leaving the chicken with the grain, or the fox with the chicken.

```
: start();
** Here is the state of the river-world:
** [chicken fox grain man ---\ \_ _/ ________________ /---]
: database ==>
** [[boat isat left]
    [chicken isat left]
    [fox isat left]
    [grain isat left]
    [man isat left]]
: putin(grain);
** Here is the state of the river-world:
** [chicken fox man ---\ \_ grain _/ ________________ /---]
: database ==>
** [[grain isat boat]
    [boat isat left]
    [chicken isat left]
    [fox isat left]
    [man isat left]]
: getin();
** Here is the state of the river-world:
** [chicken fox ---\ \_ man grain _/ ________________ /---]

;;; MISHAP - DISASTER
;;; INVOLVING:  fox has eaten chicken TOO BAD
;;; DOING    :  river_mishap eat checkeat getin pop_setpop_compiler
** Here is the state of the river-world:
** [fox ---\ \_ man grain _/ ________________ /---]

Setpop
: database ==>
** [[man isat boat]
    [grain isat boat]
    [boat isat left]
    [fox isat left]]
```

Figure 1. *An example interaction with the Pop-11 "Riverworld" program.*

In Figure 1, colons precede user commands and asterisks precede program output. It illustrates that, although we had no graphical terminals in those days, the contents of a simple world could be displayed either in the form of a changing list of propositions stored in the program (the "database") or pictorially, using a pseudo-graphical display. The example also illustrates the friendly form run-time error messages ("Mishap messages") could take.[1] After an error, the program does not abort: interaction can continue, including the option of re-setting the world.

In another exercise, by exploring different ways of writing conversational programs, and seeing their limitations, students could begin deep new learning about the structure of their own language. Unlike many others teaching AI, we did not simply provide a programming language and expect students to build upwards from its primitive constructs. Instead we produced a variety of library packages that we had written (taking full advantage of the support for advanced AI programming in Pop-11), and then allowed the students first of all to play with and use the packages then to extend them and then later to devise their own alternatives.

[1]For more details see: `http://www.cs.bham.ac.uk/research/projects/poplog/teach/river`

A student could try to design a grammar and lexicon for railway station announcements, and test it using the sentence generator provided in a library. Seeing some unexpected sentences, like "Platform 5 departed from the London train at 4pm" might lead the student to ask: How can I enrich my language specification so as to rule out sentences like that?[2] Our ideas were partly based on experience with the Logo programming language[3] and turtle graphics[4] using Logo, but after investigating it closely we decided a richer and more flexible language was needed (to which we later added an enriched turtle graphics subsystem). If we had not had a powerful programming language usable by both teachers and students we could not have developed such varied, mind-stretching, teaching examples so easily, as explained in Sloman (1984a).

2 Some philosophical issues

We also hoped our students would come to understand the differences between programs that could be described as blindly doing exactly what they were originally programmed to do, and programs that were able to modify themselves in the light of their "experience" so that what they did satisfied their own "preferences" rather than the preferences of their programmers. Likewise they could change their preferences instead of being stuck with those provided by the programmer. This could give students new arguments to use in philosophical debates about whether machines could have free will and whether they could understand what they were doing and why, as opposed to being mere syntax manipulators. (See also Franklin (1995, Ch. 2).)

Another philosophical topic that emerges from thinking about how programs work, involves the idea of *a running virtual machine*, as opposed to a virtual machine that is a mathematical abstraction whose instances are running virtual machines, e.g. the Linux virtual machine. In principle, this enables thoughtful students to have far more sophisticated discussions of problems about emergence, supervenience and the mind-brain relationship than is common in philosophy, since most philosophers completely ignore the profound developments in computer science and software engineering that allow many virtual machines to run on their computers. Compare (Sloman & Chrisley, 2003; Pollock, 2008; Sloman, 2008, 2009).

Other important questions with philosophical implications that can fruitfully be discussed in the context of playing with, designing, or extending simple AI programs include questions about different forms in which information can be represented (like the pictorial and propositional representations of the river world), and questions about the implications of different information-processing architectures combining different sorts of interacting subsystems. Immanuel Kant, among others, might have benefited immensely from such experience.

Alas, neither learning about systems with human-like competences, nor learning new ways to think about old philosophical problems went on either in most university computer science or philosophy courses, or in most schools teaching computing.

3 Is it possible to start again?

As more and more computer power became available in schools, it seems there was more and more pressure to use computers simply as tools, supporting all sorts of tasks except the one thing that could have been of greatest educational value (for at least a subset of children): namely, learning to design, implement, test, debug, analyse,

[2]See `http://www.cs.bham.ac.uk/research/projects/poplog/teach/grammar` for examples of the use of the "grammar" library with support for parsing and sentence generation using a student-supplied recursive context-free grammar.

[3]`http://el.media.mit.edu/Logo-foundation/logo/programming.htm`

[4]`http://en.wikipedia.org/wiki/Turtle_graphics`

and explain working systems of increasing sophistication. The educational uses of computers in schools seem, for most learners, to have left out a key topic that is central to what makes computers possible, namely the study of interactions between structures and processes, especially structures and processes in information-processing systems, such as minds and operating systems. Paradoxically, the problem many employers are complaining most about, namely the lack of new graduate employees with programming skills, is a direct consequence of misguided decisions to teach children to use computers for the tasks that most employers, politicians, parents and teachers thought computers were needed for, namely *using* packages, not *developing* them.

A few years ago, when there was much agitation about how to get students to consider the study of computing as a worthy activity, I looked at some proposals for (re-)introducing programming ideas into the school syllabus, and felt that whereas they were well intentioned and would work well for some students, they would not attract some high fliers interested in the humanities, psychology, philosophy, etc.

A typical Computer Science syllabus takes a “bottom up” approach in which an understanding of computing is based on understanding of some of the important “low level” features of computers, and showing progressively how more complex capabilities can be built up. Such courses are intended to attract and teach students who will develop abilities needed to create computing systems that meet important current and future practical needs. However, it is wrong to assume that that is the only way, or even the best way, to inspire such learners. In any case, there are some very bright students whose ambitions are of a different sort and who would not be attracted by such CS courses, but nevertheless have the potential to acquire and use a very deep understanding of many varieties of computation. These students might go into other disciplines that require a deep understanding of varieties of information processing mechanisms.

So I proposed an alternative syllabus offering a type of education that would be attractive to students who were interested in the study of philosophy, psychology, language, social science, biology, or mathematics. The proposal sketched four modules to be studied in the last two years of school, two modules in each year, which could be studied alongside work in other disciplines.[5] It adopted a more “top down” approach, by introducing such students to “higher level” ideas such as the notion that there are various kinds of information processing, some of which occur in nature, e.g. in all animals, including microbes and humans, whereas others occur only in man-made machines. This contrast leads to the challenge of using man-made machines to produce systems that exhibit capabilities that are characteristic of humans and other animals. Such a syllabus should introduce:

- The high level goals of AI and Computational Cognitive Science, and some of its history;[6]
- Some of the techniques and programming languages, and different forms of representation, that have been developed in pursuit of those goals;
- A brief introduction to some of the practical achievements of AI (including its growing importance in computer games and other entertainments);[7]
- Various kinds of challenges to AI, such as philosophical and empirical arguments claiming to prove that the goals are unattainable, and attempts at rebuttals;
- Ethical arguments related to whether the goals should be pursued and what the social and ethical consequences would be if they were achieved.

[5]Details are here `http://www.cs.bham.ac.uk/~axs/courses/alevel-ai.html`

[6]Recently comprehensively surveyed in a two volume book by Margaret Boden (2006).

[7]A vast amount of information about that, and other things, is available at this web site `http://www.aaai.org/aitopics` produced by the Association for the Advancement of Artificial Intelligence (AAAI).

Whereas a conventional CS syllabus might start its programming component with work on logical and arithmetical operations, and demonstrate what could be built out of those, the proposed AI syllabus would begin with programs using a high level AI language along with a variety of libraries illustrating AI techniques, including list-processing techniques using pattern-based list construction and analysis.[8] Students could play with those libraries, initially using them, then combining them, then possibly extending or modifying them, and later developing new systems of their own, including simple kinds of 'natural language' processing, such as generating and later parsing sentences, and then making plans, solving puzzles, and perhaps doing some learning, by keeping records, or modifying generalisations. Some schools might wish to introduce AI teaching based on programming physical or simulated robots, though that should be an option rather than a requirement. A toolkit supporting the development of simulated animals or robots (e.g. "Braitenberg vehicles" (Braitenberg, 1984)) with different internal architectures of varying complexity was added to Pop-11 in the decade after 1994.[9]

4 What is Artificial Intelligence (AI)?

AI is a (badly named) field of enquiry with two closely interrelated strands: science and engineering.

- The scientific strand of AI attempts to provide understanding of the requirements for, and mechanisms enabling, intelligence of various kinds in humans, other animals and information processing machines and robots.
- The engineering strand of AI attempts to apply such knowledge in designing useful new kinds of machines and helping us to deal more effectively with natural intelligence, e.g. in education and therapy.

AI is inherently highly interdisciplinary because all kinds of intelligence, whether natural or artificial, are concerned with subject matters that are studied in other disciplines, and the explanatory models of natural intelligence have to take account of and be evaluated in the disciplines that study the natural forms.[10]

Like Turing (1950) I regard attempting to define "intelligence" as a waste of time. Instead, we can collect many different examples of competences displayed by humans or other animals, and examples of challenging biologically-inspired behaviours required in future machines, and we can investigate requirements for modelling or replicating them without needing to draw any definite line between those that are and those that are not intelligent. We may find it useful to subdivide the cases in terms of either their capabilities, or the mechanisms required, or the kinds of information they use, or their potential usefulness in various contexts. Those divisions will be much more interesting and useful than any binary division based on a pre-theoretical concept like "intelligence". Similar comments can be made about binary divisions between entities with and without consciousness (Sloman & Chrisley, 2003), or with and without emotions (Sloman, 2001).

[8]For some examples see `http://www.cs.bham.ac.uk/research/projects/poplog/teach/matches`

[9]`http://www.cs.bham.ac.uk/research/projects/poplog/packages/simagent.html` A few examples of its use by university students are demonstrated in `http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent`. For use by younger learners simplified packages using the same tools could be provided.

[10]Further information about the scope of AI is provided in `http://www.cs.bham.ac.uk/~axs/courses/ai-overview.html`

5 Why would students choose to study AI?

The collection of course descriptions below is aimed at students who are interested in finding out how important ideas associated with the development of computer-based systems are relevant to the broad study of naturally occurring information-processing systems, and to the development of new machines with human-like or animal-like capabilities.

Students taking such courses, sampling a variety of AI approaches and techniques, will not only start learning how to design, test, analyse, describe, and compare working computer models of diverse kinds, but will be better equipped than most to think about their broader significance in helping us understand such phenomena as human use of language, learning, development, visual and other forms of perception, problem solving, motive formation, and creativity (Boden, 1990). They may also learn new ways to think about evolutionary processes that produced such capabilities in humans and other animals. Students with an engineering bent can focus on some of the practical applications of these techniques, e.g. in medical diagnosis, in plant control systems, intelligent tutoring systems, computer games and in new forms of entertainment. All students should be drawn into philosophical and ethical debates related to these ideas.

The course proposals below are not specifically aimed at students who wish to go on to higher education courses in computing or employment as computer developers or advanced computer users, though some of those students will certainly benefit from this unusual kind of computing education. It would also stretch the minds of students who wish to study other university subjects, such as psychology, biology, linguistics, philosophy, engineering, or management.

6 High level overview of a possible syllabus

Unlike the web site from which this proposal is derived,[11] this paper does not present a specific syllabus. The ideas presented here could be incorporated into very many different types of syllabus suited to learners of different ages with different backgrounds and career objectives. Merely in order to illustrate some of the possibilities I present a high level overview of a syllabus made up of four units, the first two of which might be taught during one year, and optionally followed by the second pair in the next year. It may be better to split some of the units into smaller, separately assessed components spread over a longer time. How much of a student's time the units would take would depend on other educational requirements: in the UK system it might be possible for students to spend between a quarter and about a third of their time on this work in their last two years. In other systems, requiring a wider spread of studies, the proportion of time would have to be reduced. Many modified versions of this outline might be offered to younger learners.

- **Unit 1:** Introduction to AI programming: building blocks
  Using simple, idealised models and games, students could learn to represent aspects of the world and rules of behaviour in such a world. Factual and other information could be stored in list structures, using pattern matching where appropriate, with various kinds of procedures devised for constructing, comparing, analysing and interpreting different kinds of symbolic structures. Example demonstrations could include programs that analyse or generate sentences, hold simple conversations (initially Eliza-like, then more knowledge-based), draw, describe and reason about pictures, explore simulated locations made of rooms, doors and corridors, make simple plans, solve problems, and play games, where appropriate

[11] `http://www.cs.bham.ac.uk/~axs/courses/alevel-ai.html`

using pre-built libraries to provide some of the building blocks. Several of our students have enjoyed working on scenarios using simulated sheep and a sheepdog, with or without complex obstacles making the dog's task hard.[12] Others have investigated (simplified) models of motivation and emotion.[13]

The programming techniques can make use of standard programming building blocks enhanced with AI mechanisms (e.g. use of local and global variables, conditionals, loops, recursion, case-constructs, along with pattern matching and rule-based programming). Simple AI toolkits may be used to implement concurrency, e.g. in simple adventure games or simulated robots or animals. Trainable neural net mechanisms might be available as libraries. For some students a basic introduction to logic programming could be included, e.g. using Prolog to manage and interrogate a simple database.

Assessment could take many different forms, including, for example, use of interactive computer-based tests of understanding of programming constructs used in the course, individual mini-projects, and group mini-projects. Students should learn how to describe and compare working systems and should be able to write an essay on limitations and possible ways of extending something they have developed, or read about.

- **Unit 2:** History, philosophy, ethics and social implications of AI
  Many essay and discussion topics could be related to the techniques encountered in Unit 1, including, for example, similarities and differences between symbolic AI, connectionist AI, and evolutionary computation. Students could learn enough to write about these without, at this stage, having learnt how to build all the varieties of AI programs. They should be able to answer questions about the importance of representations, algorithms and architectures in AI systems, and the problems of choosing between alternatives. They should be able to explain the role of a running virtual machine as a platform on which competences can be built, and say a little about requirements for virtual machines to exist.

  They should be able to explain and illustrate the recurring problem of combinatorial explosions (in time and/or space requirements) and the differences between major complexity classes. They should know about some of the ambitious and controversial things being attempted in AI (e.g. attempts to give machines emotions), and understand some of the conceptual problems in defining such goals and evaluating progress, as well as some of the ethical problems.

  Regarding philosophical and ethical issues, there is a very varied range of possibilities here including reading and discussing philosophical and ethical books and papers, or notes prepared by a teacher, with questions about whether computers could in principle replicate animal or human behaviours, whether machines could have experiences, motives, emotions, or values, whether development of such systems should be allowed, what we can learn about human nature from these investigations, what the implications are for evolutionary theories (e.g. evolution of intelligence, or unselfishness), what the long term social, economic implications are, the ethics and practicalities of military applications of AI, and so on. Some students may wish to learn about possibilities and uses of realistic models of social interaction or socio-economic systems. Metaphysical problems to be discussed include the status of virtual machines, their components and their causal relationships (Sloman, 2008, 2009).

  There are many different forms of assessment possible, including essays, presentations, participation in debates, and writing short answers to some of the more technical questions.

[12]See examples 3, 4, 5 and 8 here: `http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent`
[13]See examples 6, 9, 10, and 11 here: `http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent`

- **Unit 3:** Advanced AI programming: designing integrated systems
  Depending on how much has been achieved in Unit 1, students could learn additional programming techniques, such as (depending on their interests or what their teachers can offer) creation of planners, reasoners, proof checkers, theorem provers, language understanding systems, conversation managers, neural nets, evolutionary computations, constraint nets, image analysis techniques, image interpretation, and tools for combining such components in an integrated working system, possibly a robot or simulated robot. Obviously, many of these will be possible only after several years of programming experience. However it may be possible to learn by playing with and modifying or extending systems already developed by teachers or others. Everyone should have some experience of enabling disparate components to work together in an integrated architecture, possibly performing tasks concurrently, e.g. control of movement, perception, planning, communication and generation and evaluation of new motives. For younger, less experienced students this could mainly involve assembling pre-built units, and using existing integration toolkits (especially ones using object-oriented design with multiple inheritance, so that different functionalities can be composed).[14] Older students would include some components they have built themselves, as well as their own integration tools. Group projects would be very valuable in extending their communication and collaborative skills. There should be considerable emphasis on preliminary documentation of high level requirements as well as designs (e.g. using architecture diagrams), and on analytical or empirical comparisons of alternative solutions, as well as documentation of weaknesses and limitations of initial solutions to problems. In some cases, students can write about the philosophical, psychological, biological, social or ethical implications of their work, or its possible extensions, though that would figure in Unit 4 if it is taught in parallel.

- **Unit 4:** AI Project
  The culmination of the learning in the various units could take the form of a group project, bringing together threads from the other units. Depending on local needs, resources and timescales students might be able to devise their own project, or select from a list of possibilities provided by a teacher (with more or less specification detail provided in advance, and more or less of the required infrastructure provided in advance).

  It would be desirable to allow such projects to include use of physical robots, but that is not essential, and in many cases the use of carefully designed simulation environments can provide the most important kinds of learning – bypassing the important, but irrelevant problems connected with unreliable electrical or mechanical components. Other students, including those interested in intelligent fault analysis or design of robust systems could use physical robots.

These hypothetical course units are merely illustrative of what is possible. Teachers should be allowed to use their independence and creativity so that they can tailor their teaching to their own expertise and interests, while meeting the needs of students and the broader community. Putting too much uniformity into a national syllabus can seriously deplete the pool of talents and ideas in the next generation as well as restricting opportunities for children who require unusual learning trajectories.

[14]Illustrated here: `http://www.cs.bham.ac.uk/research/projects/poplog/teach/objectclass_example`

7 Prerequisites

Depending on the level and speed of presentation, students studying the earlier units will not require any prior knowledge of programming. A great deal of the work will involve typing text into a computer and reading textual output, and some students with visual or other disabilities may need special equipment or special help. Some will object that this is too difficult and learners should be given some of the recently developed graphical tools which allow children to develop working systems by assembling components using a computer display and pointing device. Different tools are suited to different learners, but the view that everything should be made easy for learners is based on a seriously flawed understanding of the variety of cognitive transformations required in human learning. Only trivial things can be taught without generating confusion, so it is a mistake to try to avoid confusing students. However, teachers need to be able to help learners work through those confusions to deeper understanding – like learning to find your way around a town with an irregular street pattern caused in part by natural features like rivers and hills.

The proposed introductory units do not require specific mathematical skills, though understanding elementary arithmetic and logic will help. High logical and mathematical *potential* will be very useful, and the programming exercises should help to develop both, as well as providing opportunities to use such capabilities later on. For example, learning AI will inevitably involve learning some formal logic and set theory (both of which can be learnt as sub-tasks), and study of complexity issues can be used as a basis for teaching students about combinations and permutations. Requirements for programs with graphical interactions can be used to teach students about coordinate systems and some linear algebra. The most important prerequisite is a liking for solving problems with an intricate structure, such as crossword-puzzles, sudoku, or Rubik cubes, and a strong desire to understanding how complex things work.

8 Resources needed

It may be surprising to some people to learn how much can be achieved with relatively primitive and old fashioned computing resources. When we started teaching at Sussex in 1974, we had to share a university mainframe computer that could not be used interactively, but from about 1975 we acquired our own PDP/11-40 computer with a small number of terminals (paper teletypes printing at 10 characters per second!). As there was then no AI software available for that machine, Steve Hardy, appointed as a lecturer in AI, produced a reduced implementation of the Edinburgh AI language Pop-2,[15] which he called Pop-11.[16] That later grew into a multi-language system Poplog,[17] with incremental compilers for Pop-11, Prolog,[18] Common Lisp[19] and ML,[20] first running on a VAX under VMS, then later ported to a variety of other machines, and sold commercially, mainly for AI teaching, research and development (including software validation, plant control, expert systems, data-mining and other applications). In 1998, ISL, the company then selling it for Sussex University, was bought by SPSS in order to take over the Clementine Data-Mining system, the most successful Poplog/Pop-11 product. After that Poplog became available as a free, open source system.[21] Despite its power and support for four

[15]http://en.wikipedia.org/wiki/POP-2

[16]http://en.wikipedia.org/wiki/POP-11

[17]http://en.wikipedia.org/wiki/Poplog

[18]http://en.wikipedia.org/wiki/Prolog

[19]http://en.wikipedia.org/wiki/Common_Lisp

[20]http://en.wikipedia.org/wiki/http://en.wikipedia.org/wiki/ML_(programming_language)

[21]http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html

major programming languages the download package for version v15.63 on linux requires under 17Mbytes, and the run-time system is very compact, enabling it to support a whole class of students sharing a single compute server.

This paper is not about Pop-11 or about Poplog, but the ideas proposed here were originally developed while using Pop-11 for teaching and research and have been used successfully, e.g. at Sussex University and at Birmingham University, some of it successfully using plain text visual displays, long before there were widely available graphical terminals. So these proposals are based on real experience of teaching courses somewhat like the ones being proposed. Although they were not taught to school children, as proposed here, some of them were taught to first year arts and social studies students who had never previously used a computer. Moreover at least one school teacher, Marcus Gray, demonstrated that the ideas could work in a British School (Gray, 1984).

9 Some practical challenges and possible (initial) solutions

There are many practical problems that will have to be addressed if a proposal like this is to be implemented on a large scale. Problems include: too few teachers, suitable programming tools not available in schools, lack of appropriate computing support staff, finding time in the syllabus (though I would argue that the educational value of this sort of activity could be more profound than several other things now taught in schools), funding, and time required to enable such a proposal to be developed and made available in a range of schools, with course materials suited to different ages and backgrounds. A potentially insuperable problem may turn out to be the declining interest in science and intellectual challenges in young learners, though courses of the kind proposed here might help to reverse that trend as well as showing young learners that computers are not merely (boring) tools that help you do “non-computational” things, like sending messages or downloading and playing music.[22]

References

Boden, M. A. (1990). *The creative mind: Myths and mechanisms*. London: Weidenfeld & Nicolson.

Boden, M. A. (2006). *Mind As Machine: A history of Cognitive Science (Vols 1–2)*. Oxford: Oxford University Pres.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: The MIT Press.

Franklin, S. (1995). *Artificial minds*. Cambridge, MA: Bradford Books, MIT Press.

Gray, M. (1984). POP-11 for everyone. In M. Yazdani (Ed.), *New horizons in educational computing* (pp. 252 – 271). Chichester: Ellis Horwood Series In Artificial Intelligence.

Norton, J. D. (2005). How Hume and Mach Helped Einstein Find Special Relativity. In M. Dickson & M. Domski (Eds.), *Synthesis and the Growth of Knowledge: Essays at the Intersection of History, Philosophy, Science, and Mathematics*. Open Court (Forthcoming). (http://www.pitt.edu/ jdnorton/papers/HumeMach.pdf)

[22]Further analysis of the problems and some partial solutions are discussed in these two online documents:

```
http://www.cs.bham.ac.uk/~axs/courses/alevel-ai.html
http://www.cs.bham.ac.uk/research/projects/cogaff/misc/compedu.html
```

Pollock, J. L. (2008). What Am I? Virtual machines and the mind/body problem. *Philosophy and Phenomenological Research.*, *76*(2), 237–309. (http://philsci-archive.pitt.edu/archive/00003341)

Sloman, A. (1978). *The computer revolution in philosophy*. Hassocks, Sussex: Harvester Press (and Humanities Press). (http://www.cs.bham.ac.uk/research/cogaff/crp)

Sloman, A. (1984a). Beginners need powerful systems. In M. Yazdani (Ed.), *New horizons in educational computing* (pp. 220–234). Chichester: Ellis Horwood Series In Artificial Intelligence. (http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#45)

Sloman, A. (1984b). Experiencing Computation: A Tribute to Max Clowes. In M. Yazdani (Ed.), *New horizons in educational computing* (pp. 207 – 219). Chichester: Ellis Horwood Series In Artificial Intelligence. (http://www.cs.bham.ac.uk/research/projects/cogaff/00-02.html#71)

Sloman, A. (2001). Beyond shallow models of emotion. *Cognitive Processing: International Quarterly of Cognitive Science*, *2*(1), 177-198.

Sloman, A. (2008). Virtual Machines in Philosophy, Engineering & Biology [Extended abstract]. In N. McCarthy & D. Goldberg (Eds.), *Proceedings Workshop on Philosophy & Engineering WPE-2008.* Royal Academy of Engineering, London. (http://www.cs.bham.ac.uk/research/projects/cogaff/08.html#803)

Sloman, A. (2009). What Cognitive Scientists Need to Know about Virtual Machines. In N. A. Taatgen & H. van Rijn (Eds.), *Proceedings of the 31st Annual Conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society. (http://www.cs.bham.ac.uk/research/projects/cogaff/09.html#901)

Sloman, A., & Chrisley, R. (2003). Virtual machines and consciousness. *Journal of Consciousness Studies*, *10*(4-5), 113–172.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460. ((reprinted in E.A. Feigenbaum and J. Feldman (eds) *Computers and Thought* McGraw-Hill, New York, 1963, 11–35))

Wing, J. M. (2006). Computational Thinking. *CACM*, *49*(3), 33–35. (http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf)