Can't Program, Won't Program, *Will* Program!

Amanda Peart, Alex Bennett

School of Computing, University of Portsmouth
Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK
{ amanda.peart, alex.bennett} @ port.ac.uk

Abstract

The higher education system, and indeed industry as a whole, relies on students graduating from University with the ability to think logically, laterally and creatively, yet students increasingly seem to find this process alien and incomprehensible. Research has shown that this affects performance in sociological aspects of the computing discipline and impacts significantly on student performance in the final year extended project yet based on the work of Partridge (1996), it is clear that programming implicitly requires many of the same skills. This paper explores the use of logic puzzles based on numbers and symbols such as Sudoku as a means of promoting analytical and logical thought processes in a problem-based environment, which can be applied to any computing student.
The study involves identifying a sample group of students, testing their incoming logical and analytical skills and introducing them to logic puzzles. The hypothesis is that students who improve their logical and lateral thought processes will in turn perform more effectively in subjects with a high logical content, improving both retention and progression rates.

Keywords

Computer Programming, Programming practical, Software Engineering skills, Inductive Logic, Deductive Logic.

Background

The UK higher education (HE) system, and indeed industry as a whole, relies on students graduating from University with the ability to think logically, laterally and creatively, yet students increasingly seem to find this process alien and incomprehensible. Preliminary research for this study has shown that this affects performance in sociological aspects of the computing discipline and impacts significantly on student performance in the final year extended project yet based on the work of Partridge (1996), it is clear that programming implicitly requires many of the same skills.

Over recent years, numerous institutions have documented a decrease in students' grasp of the essentials of programming skills (Thomas et al, 2002), (Partridge 1996) among others. There have been a number of proposed causes and posited solutions however they have focused primarily on the first programming unit at level one. Problems include disengagement of the learner, problems in making the transition to HE and inability to problem solve. It is believed that these occur in situations where there is miscomprehension of the fundamental concepts by the students, compounded by trying to follow half-understood rules.

Partridge (1996) has shown that the logical skills required in software engineering, regardless of the programming language in use are inductive, whereas areas of the curricula attempt to teach a deductive model, personal experience from taught sessions and assessments has shown that many students find it challenging to recognise the similarities between situations that would allow inductive reasoning to be employed.

Aims

The main aim of the project was to prove the role of inductive logic in programming and produce a suite of resources to support students' acquisition of these skills.

Outcomes

Improvements in results for subjects emphasising analytical and logical thought, evidenced by programming units in the initial study. It was found within the study that there is a relationship between analytical and logical thought. Changes to practice have been identified, particularly identify those students that have a deficiency in this area and provide further support and focus teach to supplement the assumed difficulties.

Particular improvements in the results of international and WP students in these units. In general it wasn't specifically WP student or international students that had problems in this area. Although for those international students having difficulties had other issues that were identified as problematic and not exclusively analytical and logical thinking. Although this group of students was initially identified as being predominantly in the failing group, the programming issues were more global, crossing all social economic and cultural sectors.

Identification of possible correlations between programming and weakness in the particular models of logical and analytical thought processes evidenced in Sudoku. Suduku did provide an ideal tool to identify analytical and logical thinking, as it is a publicly available puzzle that does not rely on advanced numerical or linguistic skills. It also provides an opportunity to abstract the puzzle model into a programming exercise that provides the learner with a practical application that they can all relate to.

Potential to provide new additional resources to reduce failure rates in the identified units and assist referred students to pass at their second attempt. As stated above the early identification of those students that experience difficulties with analytical and logical thinking with the aid of a support programme catering for these particular issues, would help reduce failure rates in programming units and help the increase of achievement is subsequent programming units.

Embed specific training in logical and analytical thinking into the curriculum. It is hope that the identification of these issues will provide the evidence that embedding analytical and logical thinking into the curricula will aid student progression within computing.

Deliverables

The deliverable for this project was a planned support course to assist students in acquiring and improving logical and critical thinking skills to support them in their program of study, with particular emphasis on improving performance in programming units.

The course started with formally introducing a foursquare Sudoku puzzle to the student group. The students were taken through the puzzle interactively. Once the students had gained a comprehensive understanding of the puzzle, they were asked the explain how to complete the puzzle, to reinforce the learning.

This took the form of a written exercise were teaching staff was on hand to help and support those student that found this transition more difficult. We found that many students found this abstraction of the problem the hardest part to understand. From this point the students refined explanation this into pseudo code and the next stage incorporated translating this into Java code (the departments programming language of choice for the level one students). This was completed over two sessions following typical problem based learning skills, the key thing was to ensure that the entire student class understood how they were achieving the actual code.

This same process was employed with larger puzzles along with the complexities incurred introduced to the class at each stage of the process. Students found this difficult at first and slowly applied thought to the problems the increased scale of the puzzles posed.

These students were then better prepared to apply this learning to their studies and in all cases improved their achieve assessment grade predominantly in their coursework assignment.

Putting it into Practice

In achieving this aim we tried to understand more clearly what was happening we explored the fundamental skills that these students were firstly failing to acquire and then subsequently failing to apply with the intension that this information would be used to improve the way in which the material was structured and delivered. The initial aim of the project was to understand how individuals (the won't or can't programmers) are experiencing real difficulties with programming skills, could be supported. The intended outcomes were to improved results in the follow on programming unit, as well as a wider insight into a significant problem within the discipline.

The underlying hypothesis of this study was that while students could learn the rules of a programming language by rote their previous education had not equipped them with the logical skills needed to apply generic solutions in different circumstances.

Forty students were selected to participate in the initial study for this project, all of whom had shown significantly higher results in the exam component of their last programming unit than in the coursework. It is possible that some students took time to acclimatise to higher education and therefore did not put in the requisite amount of work until the exam period: a small number of students participating in the study suggest this is taking place but is not the most crucial factor. The evidence equally supports the hypothesis that students struggling to understand the inductive application of knowledge implicit in a coursework assignment were subsidising their marks by rote learning of facts and figures which allowed them to cope more effectively with the standardised questioning style of an exam. Levels one and two of Bloom's taxonomy (Bloom/ UVIC, 2003) at which most first year exams would be targeted require minimal re-evaluation or re-application of data and are therefore suited to this learning model in a way that exams at higher levels are not.

In order to gauge the current level of logical skill, the participating students were given a short skills assessment. The assessment was divided into three sections, the skills assessment explored the students' abilities with written logic, programming logic and an inductive logical model based on the Sudoku puzzle. Parts one and two of the test explored deductive reasoning skills to determine whether students suffered from generally poor logic, a poor grasp of programming logic or some other kind of problem with comprehending program code. The test was deliberately partitioned in this way to improve the validity of the resulting data: if the hypothesis was accurate, students would struggle with parts one and three, but perform reasonably well in part two where they could rely on the application of their rote learned rules.

The results of these first two parts of the test were helpful in confirming the pattern of behaviour suggested in the hypothesis. The majority of students did well in the programming related questions, showing that they had been able to *learn* the traits of a given programming language without necessarily understanding its fundamental underpinnings. The following chart (see figure 1) illustrates a sample of 15 students, typical of the whole study: all but two students (4 and 13) performed significantly better in section two of the skills assessment with a number achieving 100%.

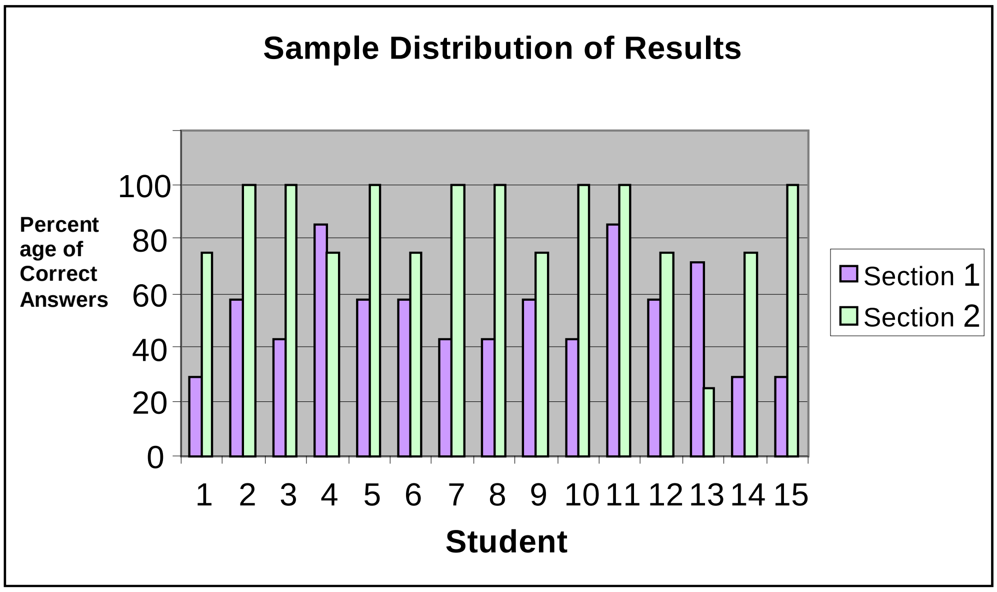

Figure 1: Sample distribution of results for skills assessment pilot

Two patterns emerged amongst these students: those who were equipped to cope with number based logic but found word based logic difficult and those who tackled the word based logic well but struggled with number based. This may be a consequence of the strong boundaries placed between subjects throughout compulsory education, preventing students from applying the skills learned in one area elsewhere. It may also be representative of the range of skills that lead students into a computing degree.

A small proportion of students, 12%, defied this trend, showing strong deductive skills across both parts, however the majority of these went on to do very badly in the final part of the skills assessment. This could suggest that their poor performance is related to disengagement but there may also be broader implications.

The final section of the skills assessment used the Japanese number placement puzzle Sudoku to test the students' inductive logic. This was chosen to reflect a different kind of logical thinking, free from mathematical or linguistic connotations: the puzzle requires no kind of calculation and the rules are simple to explain if not necessarily to apply. Based on the rules of Sudoku (Sudoku Dragon, 2006) it is also clear to see that this represents inductive rather than deductive logic and has strong comparisons with the workings of software engineering where a known range of parts need to be fitted together in a broadly understood way to achieve the intended result.

Calculating the logic being used by the students to complete these puzzles was particularly challenging, especially in a classroom environment where it was not feasible to observe the whole assessment for each student in the sample: trying to do so would have prejudiced the approaches taken by other students within hearing distance or by disturbing the environment that the assessment took place in if recording equipment was utilised. After exploring a number of different options, students were eventually asked to list the first two squares that they completed and indicate what strategy from a broad range of options they used when faced with difficulties. The intent was to try and identify correlations between the weak logic demonstrated in the first two sections of the assessment and approaches to the final part and to explore, at least in principle, how these students tackled adversity to see if it was possible to identify the early stages of a rote learning pattern.

The results of this were fascinating. Most students used one of two models to start the puzzle, with one group having a significantly higher chance of success. Of those who failed to complete the puzzle many either duplicated numbers by not checking both the column and the row, suggesting that they were 'reading' the puzzle like a piece of text; others started with squares where there was not enough information to be ascertain the contents. The conclusion must be that these represent pure guesses without any sufficient attribution of logic.

Some strong patterns did emerge in the distribution of numbers and answers across the grid (see figure 2). Few students 'chased' numbers: after identifying one number three in the grid they did not move on to see if they could use that additional knowledge to identify other three. At the same time there was no sense of a scattergun approach: 5% of the available squares on the grid accounted for 54% of the starting squares.

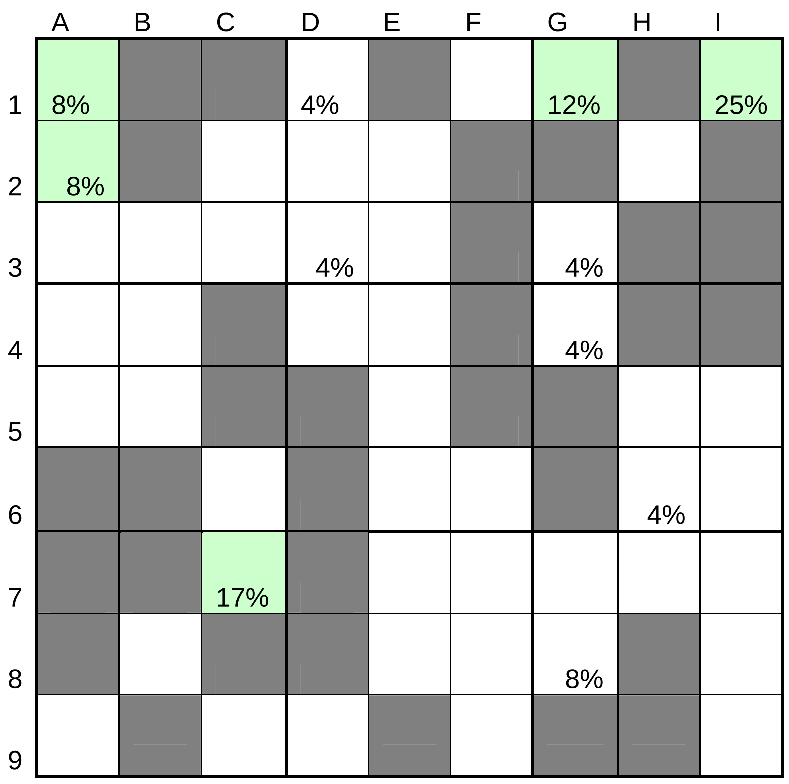

Figure 2: Distribution of starting squares on a Sudoku grid

This emphatically supports the hypothesis that, faced with adversity or a lack of understanding, these students were resorting, with some consistency, to identifying their own set of often faulty rules by which the puzzle was governed and then applying them uncritically, leading to errors when the rule failed.

As part of the skills assessment, students were given a sample Sudoku with several squares completed and explained to illustrate the logic; the puzzle used for the example was deliberately identical to the puzzle they were subsequently asked to complete and it is clear from the information captured that very few students recognised this. Of those who failed to complete the matching puzzle not a single one took advantage of the extra answers. This further suggests that the students in this sample had difficulty spotting complex patterns. One student in the sample successfully completed the sample grid but was unable to replicate that, immediately afterwards, in the assessed grid.

The inferences drawn from this skills assessment could conclude that students are accommodating a lack of fundamental understanding by ‘learning’ rules; this method functions adequately in a known or tightly enclosed environment but is far less suitable for providing generalisations and therefore cannot handle the needs of the inductive logic that underpins software engineering principle and especially actual programming skills. This has a significant impact on students programming languages: if they have misconception of the rules learned in the programming language and believe them to be universal truths without recognising that those rules have been built upon a substructure of logic then there will only be limited programming success.

Issues and Debates

Learning issues

Q. How do I apply what I learn in the lectures to the practical session?

A. Programming is just like learning to playing a musical instrument. You need to do it to really learn it. Everyone plays a few bad notes before the skill is perfected and the sound is clear. Try and keep thing simple and build on what you do understand.

Q. I just don’t get programming?

A. Do you like doing logic puzzles like Sudoku or crossword puzzles?
Do you find them straightforward?
If not practices logical puzzles may help you enhance you analytical and logical thinking skills. Some play station games that incorporate missions where you have to deduce the correct actions to complete tasks can also aid your analytical and logical thought processes.

Q. I find it hard to abstract the code from the given problem?

A. Apply logic! Breakdown the problem into the small parts. Ensure each part has only one aim. Write out how you would accomplish each aim, in an instructional format (in a similar way to pseudo code), this should then help you to translate these instructions to programming code. Each part will then become a Class, method or function of the whole program.

Teaching issues

Q. My programming unit has a large failure rate, what can I do?

A. Test your students for their analytical and logical thinking skills. This can be formal or as simple as asking who can do logic puzzles. If you find that a large number of your student find these difficult you need to embed this skill within your teaching.

Q. My students found it hard to abstract the code form the problem. How can I help them?

A. It maybe an idea to return to basics and provide the opportunity for hand holding your student through a problem. In our support session we used sudoku puzzles, any logic puzzles will do. These puzzles also provide the opportunity to look at the complexity of scalability. This enabled us to have good interactive sessions that equipped students with the skills to think about the problems posed clearly.

Resources

A set of practical programming exercises that utilizes scalable Sudoku puzzles to enhance student understanding of programming concepts in an engaging way. (Shortly available at http://www.tech.port.ac.uk/staffweb/pearta/)

Bibliography

BLOOM, B / UVIC (2003) Summary of Bloom's Taxonomy of Educational Objectives, *University of Victoria*

FERRIS, T & AZIZ, S (2005) A Psychomotor Skills Extension to Bloom's Taxonomy of Education Objective for Engineering Education *Exploring Innovation in Education and Research Conference Proceedings*

PARTRIDGE, D (1996) The Case for an Inductive Computer Science *University of Exeter*

SUDOKU DRAGON (2006) Sudoku Rules (http://www.sudokudragon.com/sudokutheory.htm accessed 04.04.06)

THOMAS L, RATCLIFE M, WOODBURY J & JARMAN E (2002) Learning Styles and Performance in the Introductory Programming Sequence, University of Wales, Aberystwyth.