Róisín Ní Bhriain - 18326577

Research Paper Review

The paper I chose was “A program to solve Sudoku” by Richard Bird. The reason I chose it was that I have some experience with solving sudoku (albeit not using code) so it was interesting to think of a solution before reading the paper and then investigating the way the author solved it. This paper is about using a functional language to approach solving sudokus. The author presents his method for finding a solution. This solution is developed from a general method moving to a more efficient method at the end. The paper overall was quite convincing although not everything about the way he constructed his solution was perfect.

His assumption is that the initially given board is valid and does not contain duplicates. This is a reasonable assumption although it would not be difficult to implement a check to ensure this validity. He presents a wholemeal type programming style by attempting to solve a suduoku by finding every single solution for the given board and working backwards to more efficient programming and valid solutions for the given board. This seems an easy way to work through the program but I would be concerned about efficiency as it would be almost impossible to use the first solution he proposes in the paper. He says himself that it would have 9^40 boards to test for validity in the end and no programmer would be willing to wait that long nor store that kind of data. I did think this was an interesting way to approach a problem however as majority of the time people would have a focus on efficiency from the beginning which may not enable them to come up with the best solution.

After having found all these solutions the author then attempts to “prune” the choices via a prune function. In this section, he gives a matrix of “choices” for each cell that is not already fixed i.e. one choice for that cell. He gives a series of calculations needed using some laws he came up with for filtering these boards. I think that these proofs were a very interesting part of this paper as it shows his thought process and made it easier to follow along with how the solution was progressing and more specifically why the solution worked. After explaining this prune function he also has an inexecutable piece of code with around 3^40 boards to check and this is if half of the entries are fixed. It seems reasonable that a reader at this point would think that the obvious solution is to use the prune function more than once. So the question is: why add another section when the solution seems so obvious?

The final section of this solution is to use this pruning function in a more useful way. Instead of using the matrix cartesian product when there are no more choices available. Instead, use cells that have at least two choices and generate an expanded matrix of its fixed choices. Then apply prune to these matrices and discard any blocked ones until all we have are fixed choices. Blocked matrices were a key idea in this paper and I felt as though the author disregarded this until the end of the paper. It seems an easy idea that could have been introduced at the start. It is any matrix that contains a cell with zero choices or the same fixed choice appears in more than one position - a duplicate. The second idea of this section is to expand on the cell with the smallest number of choices. This makes sense in terms of optimising the program. The author claims that this final solution is quite fast - taking a second or two to solve a sudoku.

In general, I think this is quite an elegant solution to the sudoku problem. The author has assumed that the reader has some knowledge of functional programming which is a reasonable assumption. Although any programmer would understand the general logic of the solution he provides. The author's logic is quite clear and he solves the problem that he presents at the start of the paper. I think the best idea in this paper was the idea of starting from the start of the problem with a general solution. Representing the sudoku as a matrix makes sense and is similar to how a sudoku has been modelled in other solutions. Checking validity of a sudoku also makes sense as a function. The process of improving the solution as time progresses is something that can be used in non-functional languages as well (provided you aren't testing the original solution). However, this is one of the reasons why the solution may not be perfect. If there is a mistake made early on in the development of the solution then it may not be caught until much later. This also means lots of time is wasted spent on general solutions which certainly won't be the end solution.

I don't think that this is necessarily the best solution to a sudoku solver but the pruning technique he uses is something that could almost certainly be applied to other similar games such as Kakuro, Chess or any game that has multiple options which impact the final solution.. Providing an option to "prune" a solution after a choice was made is very useful for optimising algorithms over using a brute force technique and evaluating all options.

Another strength of the paper was his inclusion of the number of boards the solution would generate. The first being 9^40 boards which is obviously a number far too large to test. Having these clear numbers was a good measure of the development of the program as it became more efficient. Something that could have been added is a step by step solving of a sudoku using the algorithm. Particularly while the solver uses the prune function. It would

have been nice to see the matrix of choices being whittled down and the first solution being found.

The author has approached this solver as a teaching exercise. In doing so, he has come up with a clear solution. He says himself that he presented this solver to first year undergraduates omitting the calculations. This indicates that the logic of the solver is not too complicated to explain to people with only a small amount of understanding of programming.

Overall the author describes a fairly convincing sudoku solver. I thought his method of optimising a general solution towards a more efficient solution was an excellent aspect of his argument. I also thought the pruning system was one of the better features of the approach and certainly usable in other domains. His logic was easy to follow although occasionally he overexplained concepts such as in the recurring use of the prune function.