The puzzle solving with EM - Sudoku

Abstract

Sudoku is a logic-based number placement puzzle. The objective is to fill a 9×9 grid so that each column, each row, and each of the nine 3×3 boxes (also called blocks or regions) contains the digits from 1 to 9, only one time each (that is, exclusively). The puzzle setter provides a partially completed grid.
Many computational methods to solve the puzzle like Sudoku had been developed in many ways. In this paper I am trying to create a Sudoku solving method by using the Empirical modelling Method based on the knowledge from the tuition and the pioneers.

1.Introduction

1.1More about Sudoku

Completed Sudoku puzzles are a type of Latin square, with an additional constraint on the contents of individual regions. The modern puzzle was invented by an American architect, Howard Garns, in 1979 and published by Dell Magazines under the name "Number Place". It became popular in Japan in 1986, after it was published by Nikoli and given the name Sudoku, meaning single number.(Brian Hayes, 2006) It became an international hit in 2005.

1.2Why EM, What is EM

In developing the next generation of IT applications it will be vital to rethink computer programming. Specifically, programming has two complementary ingredients: the identification of patterns of reliable agency and state-change to embody the interaction with ‘the computing machine’, and the prescription of recipes to meet specific functional goals. The term ‘identification’ is used here in a broad sense to encompass the idea that a context for computation is not typically merely found, but engineered. The term ‘prescription’ entails specifying what agents act and how this action is mediated by stimuli to which they can respond. It must be emphasized that ‘identification’ and ‘prescription’ are intended to designate activities that lead to the discovery of actual environments, artifacts and mechanisms that can be directly constructed, observed and experienced, and are terms intended to be used only with such concrete products and recipes in mind. This level of concrete engagement with experience is what should be read into the expressions “reliable agency and state-change to embody interaction” and “specifying what agents act and how this action is mediated by stimuli”. In understanding what is meant by ‘rethinking programming’, the major conceptual difficulty is that of appreciating that, whatever the level of sophistication of programing languages, there is always an implicit underlying account of how a program operates with reference to physical devices and explicit actions.(W.M.Beynon, R.C.Boyatt, S.B.Russ)
In rethinking programming, it is necessary to support both in terms of requirements,

specification, and implementation, and in terms of identification and prescription, and find a coherent perspective from which they can be understood in their relation to each other. This has been one of the principal motivations for research into Empirical modelling.

EM is centrally concerned with activities relating to identification. These are associated with investigating the observables, dependency and agency that characterize the application domain.

2. The strategies of Sudoku

The strategy for solving a puzzle may be regarded as comprising a combination of three processes: scanning, marking up, and analyzing. The approach to analysis may vary according to the concepts and the representations on which it is based.

2.1 Scanning Strategy

As figure 1 bellow, source from Tim Stellmach (2006). The top right region must contain a 5. By hatching across and up from 5s elsewhere, the solver can eliminate all empty cells in the region which cannot contain a 5. This leaves only one possibility.

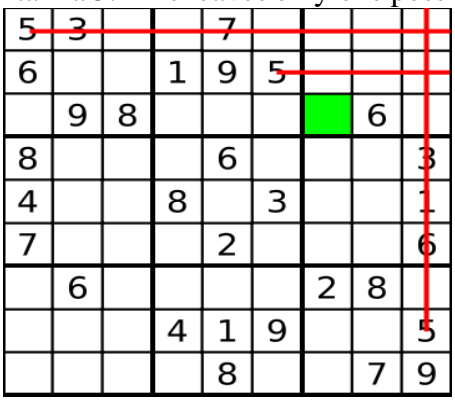

Figure 1

2.2 Marking up Strategy

Scanning stops when no further numerals can be discovered, making it necessary to engage in logical analysis. One method to guide the analysis is to mark candidate numerals in the blank cells. As figure 2, source from Pierre Dumuid (2007). The bottom middle sub-square needs a 3, 5, and 6 in the top row. This creates a contingency which, although unresolved, reveals that the green square must be a 4.

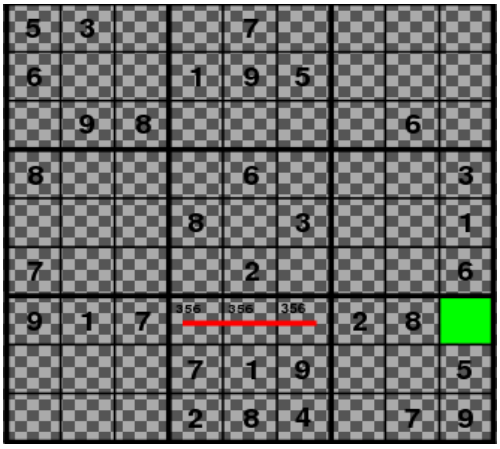

Figure 2

2.3 Analyzing Strategy

The two main approaches to analysis are "candidate elimination"(Goals of Sudoku-Grok. 2005) and "what-if"(Play Sudoku. 2006).

Candidate elimination

In "candidate elimination", progress is made by successively eliminating candidate numerals to leave one choice for a given cell. After each answer is found, another scan may be performed—usually checking to see the effect on contingencies. In general, if entering a numeral prevents completion of other empty cells, then the numeral can be eliminated as a candidate.

One method of candidate elimination works by identifying "matched cell groups". For instance, if precisely two cells within a scope (a particular row, column, or region) contain the same two candidate numerals (*p*,*q*), or if precisely three cells within a scope contain the same three candidate numerals (*p*,*q*,*r*), these cells are said to be matched. The placement of those candidate numerals anywhere else within the same scope would make a solution impossible, allowing the numbers to be eliminated as candidates from those other cells.

What-if

In the "what-if" approach (also called "guess-anhttp://search.warwick.ac.uk/website?indexSection=sitebuilder&q=boss&x=0&y=0d-check", "bifurcation", "backtracking" and "Ariadne's thread"), a cell with two candidate numerals is selected, and a guess is made. The results are followed until a duplication is found or a cell is left without a candidate, in which case the alternative must have been the solution. For each cell's candidate, the question is posed: 'will entering a particular numeral prevent completion of the other placements of that numeral?' If 'yes', then that candidate can be eliminated. If the "what-if" exercises show that either candidate is possible, then another pair should be tried. Alternatively, if the "what-if" exercises for both candidates imply an identical result, then that result is known. The what-if approach requires a pencil and eraser or a good layout memory.

There are three kind of conflicts, which can appear during puzzle solving:

1. basic conflicts - there are only N-1 different candidates in N cell in the area
2. fish conflicts - when eliminating number from N rows/columns, it will disappear also from N+1 columns/rows.
3. unique conflicts - this pattern means multiple solutions, all numbers in the pattern exist exactly two times in every area, row and column. If there is only one candidate in the cell, any virtual candidate can be added.

Encountering any of those would indicate that the puzzle is not uniquely solvable. Encountering any of them as a consequence of "what-if" indicates that an untried alternative is correct.

3. Computer Solutions

There are three general approaches taken in the creation of serious Sudoku-solving programs: human solving methods, rapid-style methods, and pure brute-force algorithms. Human-style solvers will typically operate by maintaining a mark-up matrix, and search for contingencies, matched cells, and other elements that a human solver can utilize in order to determine and exclude cell values.
Many rapid-style solvers employ backtracking（Gurari Eitan 1999）, searches, with various pruning techniques also being used in order to help reduce the size of the search tree. The term rapid-style may be misleading: Most human-style solvers run considerably faster than a rapid-style solver, although the latter takes less time to write and is more easily adapted to larger grids. A purely brute-force algorithm is very simple and finds a solution to a puzzle essentially by "counting" upward until a string of eighty-one digits is constructed which satisfies the row, column, and box constraints of the puzzle.
Rapid solvers are preferred for trial-and-error puzzle-creation algorithms, which allow for testing large numbers of partial problems for validity in a short time; human-style solvers can be employed by hand-crafting puzzlesmiths for their ability to rate the difficulty of a created puzzle and show the actual solving process their target audience can be expected to follow.
Although typical Sudoku puzzles (with 9×9 grid and 3×3 regions) can be solved quickly by computer, the generalization to larger grids is known to be NP-complete (S.A.Cook 1971). Various optimization methods have been proposed for large grids.

3.1 The EM Sudoku Model
We could be able to create an EM Sudoku Model by introducing to the EDEN, SCOUT and DoNaLD.

Observables
Grid – 9 by 9 grid.
digits – from 1 to 9.

functions – add, remove, hint, answer, new game.

Dependency
The Sudoku strategies described above.

Agency
The changing-state. once changing happened will active and conform the Dependency.

4. Evaluation of EM

Flexibility - The EM Sudoku Model enable the user change the definitions and rules of games where applicable, which is much more flexibility then other computer Sudoku Models.

Customisability - The user could be able to choose the level of difficulty suitably depending on the ability of their own.

5. Conclusion

This paper is weighted as 70% of the assignment. I started with the concepts, rules and the strategies of Sudoku, thus even you never heard about the game, you should get a brief idea in you mind about it now.

And then I introduced the EM model, which as I described above, it is a new foundation for computation.

The Sudoku is just a member of puzzles, by introducing to the EM model, we could be able to construct even more difficult puzzle models in the early future. Besides the puzzle solving, there are many many other areas also applicable for EM, and which could be adapted much better than the conventional computation models.

Acknowledgements

I would like to thank Meurig Beynon and Steve Russ for their prominent tuition in Module of Empirical modelling, and would like to thank many other pioneers whose sources I quoted in this paper.

Reference

Brian Hayes (2006), Unwed Numbers, vol. 94, American Scientist, pp. pp. 12-15

Gurari, Eitan (1999). Backtracking algorithms CIS 680: DATA STRUCTURES:
Chapter 19: Backtracking Algorithms

S. A. Cook (1971). "The complexity of theorem proving procedures". Proceedings,
Third Annual ACM Symposium on the Theory of Computing, ACM, New York:

151-158.

W.M.Beynon, R.C. Boyatt, S.B.Russ. Rethinking Programming. Source from Computer Science Dept intranet, University of Warwick.

Tim Stellmach. http://en.wikipedia.org/wiki/Image:Cross-hatching.svg. 2006

Pierre Dumuid. http://en.wikipedia.org/wiki/Image:Sudoko-Row_contingency.svg. 2007

Goals of Sudoku-Grok. http://www.sudoku-grok.com/help.jsp. 2005

Play Sudoku. Online Learning Haven. Retrieved on Oct 1, 2006.