In this chapter we work through a number of techniques raised in previous chapters. Each of the examples is accompanied with a brief explanation, although the syntax within some listings have been abbreviated to simplify the description. We introduced both static and dynamic methods associated with deriving results. A number of algorithms are then explained before discussing the MVC pattern associated with the Sudoku solver application.

11.1 Sudoku Puzzle

We have encountered many websites promoting this strategy based puzzle, such as Sudoku Dragon, EzSudoku and Sudoku Essentials. Most provide detailed descriptions on strategies to solve puzzles using a variety of techniques.
11.2 Problem Solving

There are many ways to solve problems. Over the past century, humans have created progressively more complex toys and puzzles. Examples include simple peg board, jigsaws, card games, board games, computer games and now even electromechanical entertainment devices. Technology has played an important role in this evolution. Both AI and CI have featured prominently as problem solving tools. We loosely categorize these tools based on the technique used to solve or optimise each problem. These generally include problem solving using:

Search: Brute force thru A*;
First Order Logic: Agents, Knowledge Inference, Planning and Reasoning;
Uncertainty: Combinations, Permutations, Probabilities and Decision Making;
Learning: Case Based, Guided (Knowledge Based), Probabilistic and Reinforcement; and

There are many courses in universities that teach problem solving with computers. They begin by identifying skills and personality traits of humans before moving into heuristic methods. Most people initially approached problems using a Trial and Error approach. To provide a practical example, this text will follow the approach. A number of strategies that could be implemented are discussed below. The methodology that aligns to this approach relies on recursion for the guess and backtracking to correct any errors or resolve conflicts. Generally all problems are solved within a constrained environment based on a preset number of conditions, such as position, state or orientation. The biggest issues generally encountered the humans is the interface required to initiate or maintain those conditions. We have chosen to use a tradition window based presentation to dynamically display the puzzle as algorithms permeate towards the solution.

The human interface is more accepted when implemented as a point and click (or touch) display. In this case a Soduko matrix can be displayed using a GUI show as a matrix or grid of buttons. The buttons will display the current state, can react to human input and can be used to trigger the logic required to process the interaction. This code can also verify and validate the type, range and accuracy of the input (this may eliminate conflicts and reduce the complexity of the code by confining the inputs to known values).

---

1 This edition is being completely revised for release as the third edition.
2 Nobody reads the manual anymore, they simply push buttons until the desired goal is achieved.