CSCE A405
Assignment 4 (Sudoku)
Team: Git Happens
Marshall Pratt
Chris Hill
Lydia Stark

Introduction:

In this assignment, we wrote a program that solves sudoku puzzles using constraint satisfaction. To accomplish this, we used simple Constraint Propagation using the AC3 algorithm, and as a second resort we applied backtracking should AC3 not be able to solve the puzzle.

The program will ask the user to enter a path to a .txt file in order to populate the Sudoku puzzle. The user will have the option of entering a web domain or a path to a local text file. Once the values are read into the program, the program automatically begins to solve the puzzle.

Algorithms Used:

We start by restricting the domains of our cells based on the original puzzle state. Once all domains of legal values have been updated, we enter the AC3 CSP algorithm. If this fails to render a solution, then we send the state of the puzzle to the backtracking algorithm.

1) AC3, Constraint Propagation: This algorithm assesses the domains of all cells in the puzzle. It searches for any domain of size 1, thereby inferring that the only possible solution is the one left in the domain. This value is assigned as the cell's value. Once this value is assigned, it can be removed from all neighboring cell domains. This can propagate to a large number of cells.
2) Backtracking and Constraint Propagation: This recursive algorithm looks at a cell with a domain larger than 1 value. It iterates through the cell's domain values, assigning them to the cell and performing a depth first recursive search. If this path results in failure, then it will try the next value from the domain. Each time a value is assigned, that value is constrained to all neighbors and a constraint propagation is assessed, thereby pruning a large number of possible puzzle states should it propagate into an invalid state. This greatly reduces the time complexity from the worst case O(b^n) where n is the 81 cells in the puzzle and the base b is 9 (1-9 possible values per square).

This is the extent of our algorithm. All tested puzzles can be solved using these approaches. Below is our table of attempts of varying difficulties. We used the web domain http://lipas.uwasa.fi/~timan/sudoku/ as our source for .txt based sudoku puzzles of varying difficulty.

Difficulty Level	Puzzle 1	Puzzle 2	Puzzle 3
Easy	AC3	AC3	AC3
Moderate	Backtracking	Backtracking	Backtracking
Hard	Backtracking	Backtracking	Backtracking

Conclusion:

This approach was very effective at solving all puzzles that we found to test it against. Easy puzzles were solved by the AC3 algorithm using simple constraint propagation. Moderate and difficult problems were easily solved by backtracking. Using this tiered approach (attempting constraint propagation and then falling back on backtracking) was an efficient way to solve the puzzles and also served to display which algorithm was most capable of solving difficult sudoku puzzles. We had applied a “Naked Triples” algorithm, but it was only capable of solving Moderate puzzles; as a result we replaced it entirely with the more effective Backtracking solution. We recommend using Backtracking for Constraint Satisfaction Problems to solve sudoku puzzles as the rules of the sudoku board, and the propagation of constraints, make for a fast, efficient, and effective algorithm for solving all puzzles that we tested it against. This algorithm proved capable of breaking past any situations where the AC3 and Naked Sets got stuck.