CP468 ASSIGNMENT 2

Sudoku Puzzle Solver

Group 16:
Stuart Isley: 160757220
Erman Dinsel: 160364040
Jiana Lin: 161470860

CP468
Assignment 2
Group 16
Submission Date: November 11, 2019

Problem Formulation:

Problem Statement

Initial State: Can be any possible combination of numbers in the 9x9 grid whether there is a solution to the state or not.

Actions: From any state, any number can be placed in a box as long as it remains arc consistent with the three constraints that it applies too. The constraint that says in each row of the 9x9 matrix each number must be distinct, the constraint that says in each column of the 9x9 matrix each number must be distinct, and the constraint in each 3x3 box of the 9x9 matrix each number must be distinct. If all of these constraints hold then that number may be placed in that position. The number being chosen is based off, the dynamic domain of that particular spot.

If the sudoku is not solved and there are no possible moves for a given square, then backtracking must be performed in which the algorithm backtracks to a value that is causing a problem in the matrix and changes it to a different value for that position.

Transition State: For each of these actions, whether it is backtracking or simply just placing a new number in a square, the values of the domains of the remaining unselected squares must be updated to remove or insert the value that was just changed.

Goal State: A completed 9x9 matrix, where none of the arc constraints are violated.

Path Cost: 1 unit each time a number is placed in a position, and 1 unit every time a number must be removed from backtracking due to the problem being unsolvable in its current state.

CSP Representation

Variables: 81 variables in total for a 9x9 sudoku puzzle. Let X be the set containing the variables. Let {A,B,C,D,E,F,G,H,I} represent the rows in the sudoku puzzle and {1,2,3,4,5,6,7,8,9} represent the columns. Therefore, the set of variables is:
X = {A1, A2, A3… I8, I9}.

Domains: Let D be the set of domains for the sudoku puzzle. Each empty square's domain: {1,2,3,4,5,6,7,8,9}. Each filled square's domain: {the value itself}. Therefore, the set of domains is: D = {D_{A1}, D_{A2}, … D_{I9}}

Constraints: Let C be the set of constraints for the sudoku puzzle. The constraints are:

1. Each empty square in a row must contain a unique value.
2. Each empty square in a column must contain a unique value.

CP468
Assignment 2
Group 16
Submission Date: November 11, 2019

3. Each empty square in a distinct 3x3 square must contain a unique value.

For example, if A1 contains the value 5, then A2-A9 cannot contain the value 5 (rows). Also, B1-I1 cannot contain the value 5 (columns). Finally, A2-A3 & B1-B3 & C1-C3 cannot contain the value 5 (3x3 square). Therefore, there are 9 columns, 9 rows and 9 3x3 boxes that need to contain unique values. Example as a binary constraint for A1:
(A1 != A2) & (A1 != A3) & (A1 != A4) & (A1 != A5) & (A1 != A6) & (A1 != A7) & (A1 != A8) & (A1 != A9) & (A1 != B1) & (A1 != C1) & (A1 != D1) & (A1 != E1) & (A1 != F1) & (A1 != G1) & (A1 != H1) & (A1 != I1) & (A1 != B2) & (A1 != B3) & (A1 != C2) & (A1 != C3)

Inputted Sudoku Puzzle:

```
0 8 0 | 0 0 9 | 7 4 3
0 5 0 | 0 0 8 | 0 1 0
0 1 0 | 0 0 0 | 0 0 0
- - - - - - - - - - - - -
8 0 0 | 0 0 5 | 0 0 0
0 0 0 | 8 0 4 | 0 0 0
0 0 0 | 3 0 0 | 0 0 6
- - - - - - - - - - - - -
0 0 0 | 0 0 0 | 0 7 0
0 3 0 | 5 0 0 | 0 8 0
9 7 2 | 4 0 0 | 0 5 0
```

(The lines are added to help understand when viewing the solution and the input. The lines are not used in the input text file)

Console Output:

```
2 8 6 | 1 5 9 | 7 4 3
3 5 4 | 7 6 8 | 9 1 2
7 1 9 | 2 4 3 | 5 6 8
- - - - - - - - - - - - -
8 2 3 | 6 1 5 | 4 9 7
6 9 7 | 8 2 4 | 1 3 5
1 4 5 | 3 9 7 | 8 2 6
- - - - - - - - - - - - -
5 6 8 | 9 3 1 | 2 7 4
4 3 1 | 5 7 2 | 6 8 9
9 7 2 | 4 8 6 | 3 5 1
```

CP468
Assignment 2
Group 16
Submission Date: November 11, 2019

Python Code Solution for the Problem:

```
'''
Created on Nov. 9, 2019
Sudoku Solver
@author: Stuart Isley 160757220 Erman Dinsel 160364040 Jiana Lin 161470860
'''


def read_file(fileName):
    f = open(fileName,"r")
    matrix = []
    line = f.readline()
    while line != "":
        line = line.strip("\n")
        temp = line.split(" ")
        matrix.append(temp)
        line = f.readline()
    return matrix
#0 represents an empty space in the matrix
def solve(matrix):
    temp = findNext(matrix)
    row = temp[0]
    col = temp[1]
    if row == -1:
        return True
    for i in range(1,10):
        if checkRow(matrix,row,str(i)) and checkCol(matrix, col, str(i)) and checkBox(matrix, row,
col, str(i)):
            matrix[row][col] = str(i)
            if (solve(matrix)):
                return True
            matrix[row][col] = str(0)
    return False



#find the next index in the matrix that has a 0, if none is found return (-1,-1)
def findNext(matrix):
    for row in range(9):
        for col in range(9):
            if matrix[row][col] == '0':
                return (row,col)

    return (-1,-1)
```

CP468
Assignment 2
Group 16
Submission Date: November 11, 2019

```
def checkRow(matrix,row,num):
    for i in range(9):
        if matrix[row][i] == num:
            return False
    return True

def checkCol(matrix,col,num):
    for i in range(9):
        if matrix[i][col] == num:
            return False
    return True

def checkBox(matrix,row,col,num):
    if row<3:
        if col<3:
            for i in range(3):
                for j in range(3):
                    if matrix[i][j] == num:
                        return False
        elif 3<=col and col<=5:
            for i in range(3):
                for j in range(3,6):
                    if matrix[i][j] == num:
                        return False
        else:
            for i in range(3):
                for j in range(6,9):
                    if matrix[i][j] == num:
                        return False
    elif 3<=row and row<=5:
        if col<3:
            for i in range(3,6):
                for j in range(3):
                    if matrix[i][j] == num:
                        return False
        elif 3<=col and col<=5:
            for i in range(3,6):
                for j in range(3,6):
                    if matrix[i][j] == num:
                        return False
        else:
            for i in range(3,6):
                for j in range(6,9):
                    if matrix[i][j] == num:
                        return False
```

CP468
Assignment 2
Group 16
Submission Date: November 11, 2019

```
    else:
        if col<3:
            for i in range(6,9):
                for j in range(3):
                    if matrix[i][j] == num:
                        return False
        elif 3<=col and col<=5:
            for i in range(6,9):
                for j in range(3,6):
                    if matrix[i][j] == num:
                        return False
        else:
            for i in range(6,9):
                for j in range(6,9):
                    if matrix[i][j] == num:
                        return False
    return True

def print_board(board):
    for i in range(len(board)):
        if i % 3 == 0 and i != 0:
            print("- - - - - - - - - - - - ")

        for j in range(len(board[0])):
            if j % 3 == 0 and j != 0:
                print(" | ", end="")

            if j == 8:
                print(board[i][j])
            else:
                print(str(board[i][j]) + " ", end="")

matrix = read_file("test.txt")
check = solve(matrix)
if check:
    print_board(matrix)
else:
    print("No solution found for this puzzle")
```