CS-3MI3 Homework 8

James Zhang

November 27th, 2019

Due Wednesday November 27th by end of day.

Task 1 – A 9×9 Sudoku solver

Using the 4×4 sudoku solver above as a starting point, create a 9×9 sudoku solver.

Note this code makes use of a built-in relation on lists, `permutation`, which checks that two lists are permutations of each other.

The code below is exactly like the example given. The only difference is that we are know solving the 9x9 sudoku. Which means that we need to define as 9x9 Puzzle, and the permutaion needs to be modyfied to calculate for number 1 to 9 instead of 1 to 4.

We also add the printBoard and printRow functions that print a nice game board that we can test the code on.

```
nineSudoku(Puzzle, Solution) :-

    % The solution must unify with the puzzle
    Solution = Puzzle,

    % The Puzzle must be a list of 16 elements
    Puzzle = [C11, C12, C13, C14, C15, C16, C17, C18, C19,
     C21, C22, C23, C24, C25, C26, C27, C28, C29,
     C31, C32, C33, C34, C35, C36, C37, C38, C39,
     C41, C42, C43, C44, C45, C46, C47, C48, C49,
     C51, C52, C53, C54, C55, C56, C57, C58, C59,
     C61, C62, C63, C64, C65, C66, C67, C68, C69,
     C71, C72, C73, C74, C75, C76, C77, C78, C79,
     C81, C82, C83, C84, C85, C86, C87, C88, C89,
     C91, C92, C93, C94, C95, C96, C97, C98, C99
```

```
 ],

% Define the columns, rows and squares via unification
% with the values in Puzzle.
Row1 = [C11, C12, C13, C14, C15, C16, C17, C18, C19],
Row2 = [C21, C22, C23, C24, C25, C26, C27, C28, C29],
Row3 = [C31, C32, C33, C34, C35, C36, C37, C38, C39],
Row4 = [C41, C42, C43, C44, C45, C46, C47, C48, C49],
Row5 = [C51, C52, C53, C54, C55, C56, C57, C58, C59],
Row6 = [C61, C62, C63, C64, C65, C66, C67, C68, C69],
Row7 = [C71, C72, C73, C74, C75, C76, C77, C78, C79],
Row8 = [C81, C82, C83, C84, C85, C86, C87, C88, C89],
Row9 = [C91, C92, C93, C94, C95, C96, C97, C98, C99],

Column1 = [C11, C21, C31, C41, C51, C61, C71, C81, C91],
Column2 = [C12, C22, C32, C42, C52, C62, C72, C82, C92],
Column3 = [C13, C23, C33, C43, C53, C63, C73, C83, C93],
Column4 = [C14, C24, C34, C44, C54, C64, C74, C84, C94],
Column5 = [C15, C25, C35, C45, C55, C65, C75, C85, C95],
Column6 = [C16, C26, C36, C46, C56, C66, C76, C86, C96],
Column7 = [C17, C27, C37, C47, C57, C67, C77, C87, C97],
Column8 = [C18, C28, C38, C48, C58, C68, C78, C88, C98],
Column9 = [C19, C29, C39, C49, C59, C69, C79, C89, C99],


Square1 = [C11, C12, C13, C21, C22, C23, C31, C32, C33],
Square2 = [C14, C15, C16, C24, C25, C26, C34, C35, C36],
Square3 = [C17, C18, C19, C27, C28, C29, C37, C38, C39],
Square4 = [C41, C42, C43, C51, C52, C53, C61, C62, C63],
Square5 = [C44, C45, C46, C54, C55, C56, C64, C65, C66],
Square6 = [C47, C48, C49, C57, C58, C59, C67, C68, C69],

Square7 = [C71, C72, C73, C81, C82, C83, C91, C92, C93],
Square8 = [C74, C75, C76, C84, C85, C86, C94, C95, C96],
Square9 = [C77, C78, C79, C87, C88, C89, C97, C98, C99],

% Check that all elements in each of the above are permutations
% of the list [1, 2, 3, 4, 5, 6, 7, 8, 9]. That will mean, each row, column and squ
% contains each digit from 1 to 9 once and only once.
Values = [1, 2, 3, 4, 5, 6, 7, 8, 9],
```

```
    permutation(Row1, Values),
    permutation(Row2, Values),
    permutation(Row3, Values),
    permutation(Row4, Values),
    permutation(Row5, Values),
    permutation(Row6, Values),
    permutation(Row7, Values),
    permutation(Row8, Values),
    permutation(Row9, Values),
    permutation(Column1, Values),
    permutation(Column2, Values),
    permutation(Column3, Values),
    permutation(Column4, Values),
    permutation(Column5, Values),
    permutation(Column6, Values),
    permutation(Column7, Values),
    permutation(Column8, Values),
    permutation(Column9, Values),
    permutation(Square1, Values),
    permutation(Square2, Values),
    permutation(Square3, Values),
    permutation(Square4, Values),
    permutation(Square5, Values),
    permutation(Square6, Values),
    permutation(Square7, Values),
    permutation(Square8, Values),
    permutation(Square9, Values).

printRow(Row) :- format('~w ~w ~w  ~w ~w ~w  ~w ~w ~w\n', Row).

printRows([]).
printRows(Rows) :-
    % Create a list of length 9.
    length(First,9),

    % Split Rows into First and Rest.
    append(First,Rest,Rows),

    % Print the first row, then move on to the rest.
    printRow(First),
```

```
    printRows(Rest).

printBoard([]).
printBoard(Board) :-
    % Create a list of length 27 (length of 3 rows)
    length(FirstRows,27),

    % Split Board into Rows and Rest.
    append(FirstRows,Rest,Board),

    % Print the first set of rows, then add space before the rest.
    printRows(FirstRows),
    format('\n'),
    printBoard(Rest).

test :-
    % Set the board
    Board = [1, 4, 7,   5, 6, 2,   3, 8, 9,
 5, 9, 2,   3, 8, _,   1, 7, 6,
 8, 3, 6,   7, _, 9,   5, 2, 4,

 3, _, 1,   _, _, _,   7, 5, 8,
 6, 8, _,   _, 5, _,   _, 1, 3,
 7, 5, 4,   _, _, _,   9, _, 2,

 9, 6, 8,   1, _, 3,   2, 4, 5,
 2, 7, 3,   _, 4, 5,   8, 9, 1,
 4, 1, 5,   9, 2, 8,   6, 3, 7],
    % Unify Solution and the Board using the nineSudoku relation
    nineSudoku(Board, Solution),
    % Print out the solution
    printBoard(Solution).

% After loading this file, write ‘‘test.’’
% and hit enter to see the printed solution.
```

Task 2 – A 9×9 *Diagonal* Sudoku solver

Create one more solver, this time for *diagonal* 9×9 puzzles. A diagonal Sudoku adds the requirement that the two diagonal lines from top left to bottom right and from top right to bottom left contain the numbers 1 through 9 exactly once.

Note this code makes use of a built-in relation on lists, `permutation`, which checks that two lists are permutations of each other.

This code is exactly same as the code from task1. The only difference is that here we also want to check the sudoku's diagonal. So here we have *Diagonal1* and *Diagonal2*, then use permutation to check if the diagonals in suduku contains the number 1 to 9.

```
diagonalSudoku(Puzzle, Solution) :-

    % The solution must unify with the puzzle
    Solution = Puzzle,

    % The Puzzle must be a list of 16 elements
    Puzzle = [C11, C12, C13, C14, C15, C16, C17, C18, C19,
     C21, C22, C23, C24, C25, C26, C27, C28, C29,
     C31, C32, C33, C34, C35, C36, C37, C38, C39,
     C41, C42, C43, C44, C45, C46, C47, C48, C49,
     C51, C52, C53, C54, C55, C56, C57, C58, C59,
     C61, C62, C63, C64, C65, C66, C67, C68, C69,
     C71, C72, C73, C74, C75, C76, C77, C78, C79,
     C81, C82, C83, C84, C85, C86, C87, C88, C89,
     C91, C92, C93, C94, C95, C96, C97, C98, C99
     ],

    % Define the columns, rows and squares via unification
    % with the values in Puzzle.
    Row1 = [C11, C12, C13, C14, C15, C16, C17, C18, C19],
    Row2 = [C21, C22, C23, C24, C25, C26, C27, C28, C29],
    Row3 = [C31, C32, C33, C34, C35, C36, C37, C38, C39],
    Row4 = [C41, C42, C43, C44, C45, C46, C47, C48, C49],
    Row5 = [C51, C52, C53, C54, C55, C56, C57, C58, C59],
    Row6 = [C61, C62, C63, C64, C65, C66, C67, C68, C69],
    Row7 = [C71, C72, C73, C74, C75, C76, C77, C78, C79],
    Row8 = [C81, C82, C83, C84, C85, C86, C87, C88, C89],
```

```
Row9 = [C91, C92, C93, C94, C95, C96, C97, C98, C99],

Column1 = [C11, C21, C31, C41, C51, C61, C71, C81, C91],
Column2 = [C12, C22, C32, C42, C52, C62, C72, C82, C92],
Column3 = [C13, C23, C33, C43, C53, C63, C73, C83, C93],
Column4 = [C14, C24, C34, C44, C54, C64, C74, C84, C94],
Column5 = [C15, C25, C35, C45, C55, C65, C75, C85, C95],
Column6 = [C16, C26, C36, C46, C56, C66, C76, C86, C96],
Column7 = [C17, C27, C37, C47, C57, C67, C77, C87, C97],
Column8 = [C18, C28, C38, C48, C58, C68, C78, C88, C98],
Column9 = [C19, C29, C39, C49, C59, C69, C79, C89, C99],


Square1 = [C11, C12, C13, C21, C22, C23, C31, C32, C33],
Square2 = [C14, C15, C16, C24, C25, C26, C34, C35, C36],
Square3 = [C17, C18, C19, C27, C28, C29, C37, C38, C39],
Square4 = [C41, C42, C43, C51, C52, C53, C61, C62, C63],
Square5 = [C44, C45, C46, C54, C55, C56, C64, C65, C66],
Square6 = [C47, C48, C49, C57, C58, C59, C67, C68, C69],
Square7 = [C71, C72, C73, C81, C82, C83, C91, C92, C93],
Square8 = [C74, C75, C76, C84, C85, C86, C94, C95, C96],
Square9 = [C77, C78, C79, C87, C88, C89, C97, C98, C99],

Diagonal1 = [C11, C22, C33, C44, C55, C66, C77, C88, C99],
Diagonal2 = [C91, C82, C73, C64, C55, C46, C37, C28, C19],

% Check that all elements in each of the above are permutations
% of the list [1, 2, 3, 4, 5, 6, 7, 8, 9]. That will mean, each row, diagonal, colu
% contains each digit from 1 to 9 once and only once.
Values = [1, 2, 3, 4, 5, 6, 7, 8, 9],
permutation(Row1, Values),
permutation(Row2, Values),
permutation(Row3, Values),
permutation(Row4, Values),
permutation(Row5, Values),
permutation(Row6, Values),
permutation(Row7, Values),
permutation(Row8, Values),
permutation(Row9, Values),
permutation(Column1, Values),
```

```
permutation(Column2, Values),
permutation(Column3, Values),
permutation(Column4, Values),
permutation(Column5, Values),
permutation(Column6, Values),
permutation(Column7, Values),
permutation(Column8, Values),
permutation(Column9, Values),
permutation(Square1, Values),
permutation(Square2, Values),
permutation(Square3, Values),
permutation(Square4, Values),
permutation(Square5, Values),
permutation(Square6, Values),
permutation(Square7, Values),
permutation(Square8, Values),
permutation(Square9, Values)
permutation(Diagobal1, Values),
permutation(Diagonal2, Values).
```