PROJECT 3: Sudoku

Introduction to Artificial Intelligence Spring 2022

Eyþór Mikael Eyþórsson
Jakob Nikolaus Mummert
Line Mei Tong Mao Dejgaard
{eyey, jamu, ldej}@itu.dk

April 8, 2022

1 Introduction

For the implementation of the Sudoku Solver, we've utilized Forward Checking (see section 3) and the methods already provided in the zip-file. We've implemented things in one java class SudokuSolver.

2 Initializing Domains

In the setup InitializeD() is called. This method is making a list of possible values (from 1 to length of the puzzle), and adds that to the domain for each variable. This means that when the method is finish, every domain contains values from 1 to the length of the puzzle.

3 The Forward Checking (method: FC())

The forward checking algorithm is a recursive algorithm which ends when all variables (positions) in the sudoku is assigned a value.

In the algorithm X is assigned to the first index in the assignment (see section 4) where no value is assigned, and the goal in here is to get X assigned to a value that still satisfies the constraints in sudoku:

- Each row must contain the numbers from 1 to the size of the puzzle, without repetitions
- Each column must contain the numbers from 1 to the size of the puzzle, without repetitions
- Each block must contain the numbers from 1 to the size of the puzzle, without repetitions

In the implementation we save a copy of D, since we manipulate it while looping though the values in the domain for the variable X.

In the loop we check if the value V can be assigned to the variable X without breaking arc consistency. If V breaks the arc consistency, D is set back to the copy of D we made in the beginning, because checking for arc consistency is changing D. Then we try the next value in the domain for X if any. If there is no more values, we return null corresponding to a failure. This means that we have end up in a dead end, and we have to go back.

If V is not breaking arc consistency, we can try V, and go further down the tree. This is done by calling FC again with the assignment, where V is assigned to the variable with the index X. If this call it not returning a failure, we have a solution, since the previous FC call returned an assignment due to the fact that there were no more variables unassigned. This solution is returned. If the FC call is returning a failure, the variable with index X is set back to 0, and D is also changed back to the copy made before checking for arc consistency. And then we try another value if any, else we return null (a failure).

4 Assignment

The assignment is implemented as a list of integers representing the sudoku board. This was aready done in the given code.

5 Solve

When solve is called, the puzzle is translated to an assignment (see section 4), and the values in the domains is deleted, such that the variables already assigned a value only have that value in their domain.

After this, we initialize fc meaning that we enforces consistency between unassigned variables and all initially assigned values (method given in zip-file). If arc consistent is not possible from the beginning, the sudoku can not be solved, and therefore false is returned. Else we call FC, which returns the complete assignment (the solution) or null (a failure). If it is a complete assignment, the puzzle is set to the solution, and true is returned. Else false is returned.

6 Domains: Copy methods

The method getCopy() takes in an arraylist of arraylists of integers. The method has an inner and an outer loop. The outer loop makes an arraylist of integers, and the inner loop adds all the integers from each arraylist of integers given in the parameter to the temp arraylists. These temp arraylists is in the outer loop added to the result arraylist made in the begginning of the method. This arraylist is returned in the end.

The method getCopyOfD() just calls getCopy(D).

The method getCopyOfDX() takes in a index X, makes a new arraylist and add all the values from the X domain, and return this arraylist.

7 Testing

We've made a method called getPuzzle, which takes in an integer. There is 0-3+ corresponding to 4 different sudoku that can be used for testing purpose. If you want to type in a sudoku yourself comment out line 32. These sudokus are found on the internet.