Project description

The project assigned to me is to make an animation of a particular algorithm used to solve a Sudoku puzzle. A Sudoku puzzle is a 9x9 grid with digits that can range from 1 to 9 and blanks that need to be filled in by the player. A solution will contain all the digits from 1 to 9 in each row, cell and 3x3 sub-grids of the bigger grid. No repetition of a digit is allowed in any row, column of 3x3 sub-grid.

By the above definition it is easy to observe that inside each row, column or 3x3 sub-grid, the values should be "all-different". In fact, the algorithm that is going to be used in this project to solve Sudoku instances uses the "all-different constraint". This technique is used to solve problems in Constraint Programming, a branch of Artificial Intelligence. When solving constraint satisfaction problems we create a model that contains variables, domains of values associated to variables, and constraints in which variables take part.

In the case of a Sudoku puzzle, we have 81 variables for the 9x9 grid, each initially associated to a domain of values from 1 to 9. There will be 27 "all-different" constraints between the 9 variables of each row, column and 3x3 sub-grid.

A great exercise now is to imagine a Sudoku puzzle using the above model. In this view of the puzzle we start with the original 9x9 grid, but instead of having blanks for unknown digits, we fill in the blanks with the sequence of all digits from 1 to 9. This will represent available digits from which we can eliminate values until we have only a single value used in the solution. The Sudoku puzzle given to the player to solve will always be partially completed, therefore we should be able to keep our model in a consistent state by making sure that digits that have already been used in a row, column or 3x3 sub-grid are removed from the domain of available values in their respective group of 9 variables representing 9 cells. The algorithms for the all-different constraint does exactly this, it removes values from the domain of the variables that are no longer available as they are already used in a particular row, column or 3x3 sub-grid.

Initial Sudoku puzzles should always be partially completed in such a way there will be only one unique solution after the player completes the rest of the puzzle. It is then trivial to arrive to the solution by running the all-different algorithm on rows, columns and 3x3 sub-grids. By removing values from the domains in order to keep the puzzle consistent, we can now observe that no guessing is involved in arriving to a solution and every progress done counts towards finding the valid solution of the puzzle.

It is good to note that solving a Sudoku puzzle is only one out of the many use cases of the all-different algorithm, such as arriving to a solution for job-shop scheduling problems. Once the algorithm is coded up, it is very easy to apply the same technique to other real-world problems with only minor modifications.

Progress

Throughout the weekly iterations during the first semester I have gained a lot of insight about techniques used in the Constraint Programming course thought by Prof. Patrick Prosser and the particular way of thinking required to approach and model real-world problems, including the Sudoku puzzle.

The project is coded up in Java and works without the need of any third-party library. As of the time of writing this project report, most of the implementation for the Graphical User Interface and the logic required to solve a Sudoku puzzle is done. The user is able to open and load a text file that represents a Sudoku instance. The GUI will update accordingly showing the partially completed puzzle and instead of having blank in the 9x9 grid for unknown values, the blanks are filled in with the sequence of digits from 1 to 9. The user can then select a row, column of 3x3 sub-grid and run the all-different algorithm on that selection. After running the algorithm on one selection, the user will notice that the original digits that were used in the partially completed Sudoku will disappear from the domain of available values for each of the 9 variables in the selection. This makes the original selection of the row, column of 3x3 sub-grid consistent in terms of getting to a solution with "all-different" values.

The all-different algorithm is animated in the right side of the screen as a sequence of steps that modify a bipartite graph containing the selected 9 variables of a row, column or 3x3 sub-grid in one partition and the digits from 1 to 9 in the second partition. Edges between the partitions mean that a variable contains a particular value in its domain.

Users of the program are also able to run a demonstration that will prove that the implementation of the all-different algorithm is correct by showing the first 5 steps towards the solution. After running the all-different algorithm on a hard-coded sequence of 5 rows, columns and 3x3 sub-grids, the user can notice that the algorithm is "intelligent" and mimics the real-life thinking of a player.

Although there is still some work to be done, everything until now runs smooth with particular attention given to multi-threading and clean and clear debug information.

Plan

The time left until the deadline of the project will be mostly devoted to writing the individual project report that will be 30 pages long, worth 20 credits and not 40 credits as I am a joint-degree student. The report is being written in LaTeX for better formatting of text, code and figures. I have already started learning on how to write LaTeX code and have a draft showing the use of a title page, table of contents, chapters, figures, references and bibliography.

As I believe the first week of the second semester will see the finishing touches of program done, I am set to continue this fast pace in writing the project report throughout the weekly meetings with the supervisor, in order to have everything done long before the deadline, so that I can pay enough attention to rephrasing paragraphs I believe are hard to understand or do not convey the intended meaning to the reader.

Problems

Almost all the problems in terms of thinking, modeling the problem or coding were discussed and solved during the weekly meetings with my project supervisor.

A difficulty I foresee is making the Graphical User Interface of the program look the same, or almost the same on different platforms running different operating systems, but I will try to solve this by either supplying an open-source font with the project or in the worst-case scenario, I will hard-code in values for size and alignment of the text across the mostly used operating systems.

As with any software engineering project, bugs can always arise during unexpected times or difficulties finding that perfect word to convey the intended meaning in the project report, but I know I can always rely on the help I get from my project supervisor.

Gabriel I. Stratan
2024112s
4th year, School of Computing Science
Studying Computing Science & Business management BSc

December 18th, 2015