CSCI 1710 Capstone: Generating Sudoku Boards With Unique Solutions Using Z3

Emily Hinds
Brown University
Faculty Sponsor: Tim Nelson

Background

I worked on this project with Christopher Pellinger and Elizabeth Wu for Tim Nelson's course CSCI 1710: Logic for Systems. Our goal in this project was to determine properties of incomplete sudoku boards with unique solutions. In order to determine this, we had to model a sudoku board, find a way to generate valid full boards (and determine validity), remove values from those boards to end up with a solvable partial board, and finally ensure that the partial board only had one possible solution. We began by working in Froglet and then Temporal Forge, but we eventually switched to Python and Z3 for our final model.

Model

We changed part way through the process of this project from Temporal Forge to Python and Z3 after facing difficulties with ensuring unique solutions in Forge. We were working in a bit width of 4, so we were only able to generate a maximum of 7 boards to run our code with, which made it difficult to determine if a board had a unique solution. We also faced performance issues, so we made the move to Z3.

Our Z3 model generates a complete sudoku board and then removes one number at a time, ensuring that with that number removed there is still a unique solution. The number of values to remove is a parameter that we take in. This is the result of running the model with 55 numbers to remove:

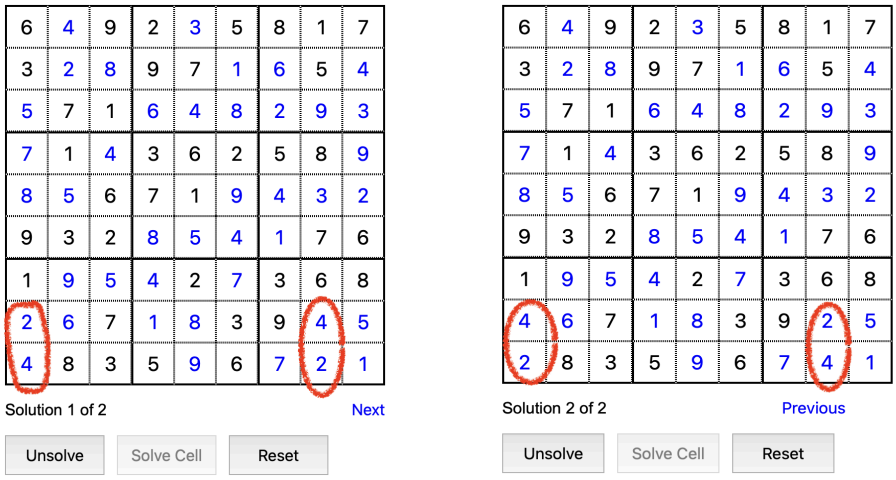
```
Incomplete Sudoku Board:
. . . 8 . . 7 6 .
. . 5 . 6 9 . 3 2
. . 6 . . 3 5 . .
. . 9 . . . . . .
. 8 . . 7 . 6 4 1
. 2 . . . 8 . 7 .
. 4 . . . . . . .
. . . 1 . . . . .
3 . . 2 4 . . 1 .

Fully Solved Sudoku Board:
2 3 4 8 5 1 7 6 9
8 7 5 4 6 9 1 3 2
9 1 6 7 2 3 5 8 4
7 6 9 5 1 4 3 2 8
5 8 3 9 7 2 6 4 1
4 2 1 6 3 8 9 7 5
1 4 8 3 9 6 2 5 7
6 5 2 1 8 7 4 9 3
3 9 7 2 4 5 8 1 6
```

Findings

We used the Hypothesis testing library to test our model. We wanted to find properties of boards with unique solutions that always held true. Some properties that we were able to find were that an incomplete sudoku board must only have a maximum of one value fully absent from the board. We also found incomplete sudoku boards must have a maximum of three fully empty 3x3 grids in order to ensure a unique solution.

We noticed in the process of creating our model that some boards with multiple solutions had a pattern we called “pair diagonals” in which there are two values in adjacent rows or columns that could be arranged in two separate orders and still be a valid solution. We hypothesized that if this pattern is present in a full sudoku board, then at least one of the four values must be included in the partial board in order to ensure a unique solution. Here is an example of pair diagonals in which flipping the 4 and 2 generates multiple solutions to the same incomplete sudoku board, created using sudokuspoiler.com:



Upon writing and running the tests for pair diagonals, we found that this pattern did not hold true in every case. It turns out that it is not a necessary property of incomplete sudoku boards.

When testing the number of values to remove, we found that for any full sudoku board, there is a way to remove anywhere between 1-55 values and ensure a unique solution. We found that it is not possible to remove between 66-81 values and ensure a unique solution, which leaves a volatile range when removing 56-65 numbers. We ran our model with 1000 different boards each time, but our tests would sometimes pass and sometimes not in this range. If we were to expand upon this project, we would like to find a way to further optimize our model so that we are able to run it on more boards and perhaps close in on the boundaries of this volatile range.