Available at the COMP2111 website.

Assignment 3 COMP2111 13s1 Sudoku

Kai Engelhardt

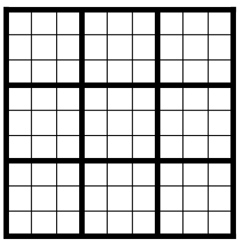
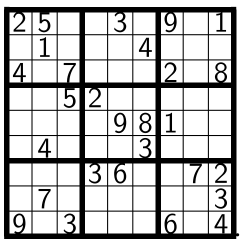
Revision: 1.3 of Date: 2013/05/17 00:25:18

This assignment is worth 20 marks and due **Sunday May 26th, 23:59:59** local time Sydney.

Tasks

Develop a Sudoku solving assistant in Event-B. If you don't know Sudoku yet, consult wikipedia.

One required event is `setup`, which takes a partial function from Sudoku coordinates to digits as argument.[1] Its purpose is to set up the initially empty grid with the seed values typically found in a sudoku challenge.

Example: before `setup`:  and after: .

At the abstract level, the current state of the Sudoku grid is represented as a partial function (say, g for *grid*) similar to the input for `setup`. Another required event is `fill` with arguments for the Sudoku cell and digit to indicate which Sudoku grid cell should be filled with which digit. Illegal moves must be prevented, idiotic moves not. Someone using the abstract level machine should be able to make poor choices. At this level, it should already be possible to play Sudoku by animating the machine.

Refine the abstract level by adding a helpful representation (say, p for *possibilities*) of the grid, namely one that keeps track of the possible values of each cell. So before *setup*, every cell can hold every digit `1..9`. After setting up a single cell (i, j) to hold digit k the grid should update to reflect the reduction in the possible values for affected coordinates. So $p(i, j)$ is reduced to the singleton $\{k\}$, and k needs to be removed from the possible values of all cells in the same 3×3 square, the same row, and the same column. Strong invariants should link the two representations.

Introduce an argument-less event `autofill` that fills a cell in g by exploiting the presence of a singleton set in p at an unfilled coordinate in g.

[1] This event is present so we don't have to hard-code a particular Sudoku instance. For actual play testing it may be advisable to have some test case arguments stored in constants.

Introduce an event `hint` that reveals Sudoku coordinates (i, j) and $p(i, j)$ such that $|p(i, j)|$ is minimal among the unfilled cells in g.

Introduce an event `guess` that becomes enabled after a `hint` and fills the hinted at cell with one of the possible values.

Introduce an argument-less event `alarm` that indicates the presence of a cell with no remaining possible digits, i.e., $p(i, j) = \emptyset$.

With these events in place, it should be easier to finish off some Sudoku instances by using `autofill` whenever it's enabled, and targeting cells with few remaining possibilities with `guess` and `fill`. And when `alarm` is enabled it's time to start over.

Having to start over is of course frustrating, whence, in a final refinement, introduce an `undo` event. This requires another data refinement because it is not allowed to undo a set-up grid cell. The previous representations did not distinguish between set-up cells and those filled later.

Feel free to introduce other events as you see fit. The ultimate goal is to have a helpful Sudoku assistant.

As usual, strive to discharge all POs in Rodin. Document your development with an emphasis on conciseness and legibility in a report. Discuss POs you couldn't discharge within Rodin in the report.

Deliverables

`Sudoku.zip` is your Event-B project exported from `Rodin`.

`Sudoku.tex` is a LaTeX document with your name or student number mentioned in the `\author` command. It contains your report.

Submission Instructions

The `give` command to be run is:

```
% 2111
% give cs2111 ass3 Sudoku.zip Sudoku.tex
```

The command above submits the bare minimum. Should you feel the need to include more files, e.g., for vector diagrams, just list them as well.