Formal Verification Final Project

Submission Deadline: 12.05.24

Sokoban, Japanese for “warehouse keeper”, is a transport puzzle created by Hiroyuki Imabayashi in 1980. The goal of the game is simple, the warehouse keeper must push the boxes to designated locations in the warehouse.

The warehouse is depicted as a grid with walls creating a labyrinth. The following are rules that must be respected:

- Warehouse keeper can only move horizontally or vertically in the grid, one cell at a time.
- Boxes can only be pushed, not pulled, into an empty space.
- Warehouse keeper and boxes cannot enter “wall” cells.

For this project, you will be using the XSB format to describe the board. See Figure 1 for an example board in XSB format with a legend. Solutions should be given in LURD format (l - left, u - up, r - right, d - down).

```
----#####----------
----#---#----------
----#$--#----------
--###--$##---------
--#--$-$-#---------
###-#-##-#---######
#---#-##-#####--..#
#-$--$----------..#
#####-###-#@##--..#
----#-----#########
----#######--------
```

Symbol	Definition
@	warehouse keeper
+	warehouse keeper on goal
$	box
∗	box on goal
#	wall
.	goal
-	floor

Figure 1: Sokoban Board using XSB (left) and symbol definitions (right).

Part 1

1. Define an FDS for a general $n \times m$ Sokoban board. Use XSB format to describe the board.
2. Define a general temporal logic specification for a win of the Sokoban board.

Part 2

1. Using Python, or another coding language, and your answers to Part 1, automate definition of given input boards into SMV models. These models should contain both the model and the temporal logic formulae defining a win.
2. Run each of these models in nuXmv. Indicate the commands you used to run nuXmv, as well as screenshots of the nuXmv outputs.
3. For each board, indicate if it is winnable. If it is, indicate your winning solution in LURD format.

Part 3

1. Measure performance of nuXmv's BDD and SAT Solver engines on each of the models.
2. Compare the performance of the two engines. Is one engine more efficient than the other?

Part 4

1. Break the problem into sub-problems by solving the boards iteratively. For example, solve for one box at a time. Indicate the temporal logic formulae used for each iteration.
2. Indicate runtime for each iteration, as well as the total number of iterations needed for a given board.
3. Test your iterative solution on larger more complex Sokoban boards.

Submission

- Upload all your codes, xsb Sokoban boards, SMV files and outputs to a GitHub repo. Include a README explaining how to run your codes.
- Compile all your answers and results into a PDF report.
- Include a link to your GitHub repo in your PDF submission.

Sample Sokoban Boards

Here are some sample Sokoban boards for you to use to help setup your codes.