Final Project - Formal Verification

Guy Varsano 326194123

Yuval Varsano 32614115

Part 1

1. Define an FDS for a general n × m Sokoban board. Use XSB format to describe the board.

Warehouse keeper is on the floor and next to him is a floor:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = -) \wedge (B'[i][j] = -) \wedge (B'[i-1][j] = @)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = -) \wedge (B'[i][j] = -) \wedge (B'[i+1][j] = @)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = -) \wedge (B'[i][j] = -) \wedge (B'[i][j-1] = @)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = -) \wedge (B'[i][j] = -) \wedge (B'[i][j+1] = @)$$

Warehouse keeper is on the floor and next to him is a wall:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = \#) \wedge (B'[i][j] = @) \wedge (B'[i-1][j] = \#)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = \#) \wedge (B'[i][j] = @) \wedge (B'[i+1][j] = \#)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = \#) \wedge (B'[i][j] = @) \wedge (B'[i][j-1] = \#)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = \#) \wedge (B'[i][j] = @) \wedge (B'[i][j+1] = \#)$$

Warehouse keeper is on the goal and next to him is a wall:

$$\rho_u = (B[i][j] = *) \wedge (B[i-1][j] = \#) \wedge (B'[i][j] = *) \wedge (B'[i-1][j] = \#)$$

$$\rho_d = (B[i][j] = *) \wedge (B[i+1][j] = \#) \wedge (B'[i][j] = *) \wedge (B'[i+1][j] = \#)$$

$$\rho_l = (B[i][j] = *) \wedge (B[i][j-1] = \#) \wedge (B'[i][j] = *) \wedge (B'[i][j-1] = \#)$$

$$\rho_r = (B[i][j] = *) \wedge (B[i][j+1] = \#) \wedge (B'[i][j] = *) \wedge (B'[i][j+1] = \#)$$

Warehouse keeper is on the floor and next to him is a goal:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = .) \wedge (B'[i][j] = -) \wedge (B'[i-1][j] = +)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = .) \wedge (B'[i][j] = -) \wedge (B'[i+1][j] = +)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = .) \wedge (B'[i][j] = -) \wedge (B'[i][j-1] = +)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = .) \wedge (B'[i][j] = -) \wedge (B'[i][j+1] = +)$$

Warehouse keeper is on the goal and next to him is a goal:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = .) \wedge (B'[i][j] = .) \wedge (B'[i-1][j] = +)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = .) \wedge (B'[i][j] = .) \wedge (B'[i+1][j] = +)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = .) \wedge (B'[i][j] = .) \wedge (B'[i][j-1] = +)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = .) \wedge (B'[i][j] = .) \wedge (B'[i][j+1] = +)$$

Warehouse keeper is on the goal and next to him is a floor:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = -) \wedge (B'[i][j] = .) \wedge (B'[i-1][j] = @)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = -) \wedge (B'[i][j] = .) \wedge (B'[i+1][j] = @)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = -) \wedge (B'[i][j] = .) \wedge (B'[i][j-1] = @)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = -) \wedge (B'[i][j] = .) \wedge (B'[i][j+1] = @)$$

Warehouse keeper is on the floor and next to him is a box next to a floor:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i-1][j] = @) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i+1][j] = @) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i][j-1] = @) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i][j+1] = @) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the goal and next to him is a box next to a floor:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i-1][j] = @) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i+1][j] = @) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j-1] = @) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j+1] = @) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the floor and next to him is a box next to a wall:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = \#) \wedge (B'[i][j] = @) \wedge (B'[i-1][j] = \$) \wedge (B'[i-2][j] = \#)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = \#) \wedge (B'[i][j] = @) \wedge (B'[i+1][j] = \$) \wedge (B'[i+2][j] = \#)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = \#) \wedge (B'[i][j] = @) \wedge (B'[i][j-1] = \$) \wedge (B'[i][j-2] = \#)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = \#) \wedge (B'[i][j] = @) \wedge (B'[i][j+1] = \$) \wedge (B'[i][j+2] = \#)$$

Warehouse keeper is on the goal and next to him is a box next to a wall:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = \#) \wedge (B'[i][j] = +) \wedge (B'[i-1][j] = \$) \wedge (B'[i-2][j] = \#)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = \#) \wedge (B'[i][j] = +) \wedge (B'[i+1][j] = \$) \wedge (B'[i+2][j] = \#)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = \#) \wedge (B'[i][j] = +) \wedge (B'[i][j-1] = \$) \wedge (B'[i][j-2] = \#)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = \#) \wedge (B'[i][j] = +) \wedge (B'[i][j+1] = \$) \wedge (B'[i][j+2] = \#)$$

Warehouse keeper is on the floor and next to him is a box next to a box:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = \$) \wedge (B'[i][j] = @) \wedge (B'[i-1][j] = \$) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = \$) \wedge (B'[i][j] = @) \wedge (B'[i+1][j] = \$) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = \$) \wedge (B'[i][j] = @) \wedge (B'[i][j-1] = \$) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = \$) \wedge (B'[i][j] = @) \wedge (B'[i][j+1] = \$) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the floor and next to him is a box next to a goal:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = .) \wedge (B'[i][j] = -) \\ \wedge (B'[i-1][j] = @) \wedge (B'[i-2][j] = *)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = .) \wedge (B'[i][j] = -) \\ \wedge (B'[i+1][j] = @) \wedge (B'[i+2][j] = *)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = .) \wedge (B'[i][j] = -) \\ \wedge (B'[i][j-1] = @) \wedge (B'[i][j-2] = *)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = .) \wedge (B'[i][j] = -) \\ \wedge (B'[i][j+1] = @) \wedge (B'[i][j+2] = *)$$

Warehouse keeper is on the goal and next to him is a box next to a goal:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = .) \wedge (B'[i][j] = .) \\ \wedge (B'[i-1][j] = @) \wedge (B'[i-2][j] = *)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = .) \wedge (B'[i][j] = .) \\ \wedge (B'[i+1][j] = @) \wedge (B'[i+2][j] = *)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = .) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j-1] = @) \wedge (B'[i][j-2] = *)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = .) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j+1] = @) \wedge (B'[i][j+2] = *)$$

Warehouse keeper is on the goal and next to him is a box next to a box:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] = \$) \wedge (B'[i][j] = .) \\ \wedge (B'[i-1][j] = \$) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] = \$) \wedge (B'[i][j] = .) \\ \wedge (B'[i+1][j] = \$) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] = \$) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j-1] = \$) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] = \$) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j+1] = \$) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the floor and next to him is a box on goal next to a floor:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] =*) \wedge (B[i-2][j] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i-1][j] = +) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] =*) \wedge (B[i+2][j] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i+1][j] = +) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] =*) \wedge (B[i][j-2] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i][j-1] = +) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] =*) \wedge (B[i][j+2] = -) \wedge (B'[i][j] = -) \\ \wedge (B'[i][j+1] = +) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the goal and next to him is a box on goal next to a floor:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] =*) \wedge (B[i-2][j] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i-1][j] = +) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] =*) \wedge (B[i+2][j] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i+1][j] = +) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] =*) \wedge (B[i][j-2] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j-1] = +) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] =*) \wedge (B[i][j+2] = -) \wedge (B'[i][j] = .) \\ \wedge (B'[i][j+1] = +) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the floor and next to him is a box on goal next to a wall:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] =*) \wedge (B[i-2][j] = \#) \wedge (B'[i][j] = @) \\ \wedge (B'[i-1][j] =*) \wedge (B'[i-2][j] = \#)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] =*) \wedge (B[i+2][j] = \#) \wedge (B'[i][j] = @) \\ \wedge (B'[i+1][j] =*) \wedge (B'[i+2][j] = \#)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] =*) \wedge (B[i][j-2] = \#) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j-1] =*) \wedge (B'[i][j-2] = \#)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] =*) \wedge (B[i][j+2] = \#) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j+1] =*) \wedge (B'[i][j+2] = \#)$$

Warehouse keeper is on the goal and next to him is a box on goal next to a wall:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = *) \wedge (B[i-2][j] = \#) \wedge (B'[i][j] = +) \\ \wedge (B'[i-1][j] = *) \wedge (B'[i-2][j] = \#)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = *) \wedge (B[i+2][j] = \#) \wedge (B'[i][j] = +) \\ \wedge (B'[i+1][j] = *) \wedge (B'[i+2][j] = \#)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = *) \wedge (B[i][j-2] = \#) \wedge (B'[i][j] = +) \\ \wedge (B'[i][j-1] = *) \wedge (B'[i][j-2] = \#)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = *) \wedge (B[i][j+2] = \#) \wedge (B'[i][j] = +) \\ \wedge (B'[i][j+1] = *) \wedge (B'[i][j+2] = \#)$$

Warehouse keeper is on the floor and next to him is a box on goal next to a box:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = *) \wedge (B[i-2][j] = \$) \wedge (B'[i][j] = @) \\ \wedge (B'[i-1][j] = *) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = *) \wedge (B[i+2][j] = \$) \wedge (B'[i][j] = @) \\ \wedge (B'[i+1][j] = *) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = *) \wedge (B[i][j-2] = \$) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j-1] = *) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = *) \wedge (B[i][j+2] = \$) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j+1] = *) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the goal and next to him is a box on goal next to a box:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] = *) \wedge (B[i-2][j] = \$) \wedge (B'[i][j] = +) \\ \wedge (B'[i-1][j] = *) \wedge (B'[i-2][j] = \$)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] = *) \wedge (B[i+2][j] = \$) \wedge (B'[i][j] = +) \\ \wedge (B'[i+1][j] = *) \wedge (B'[i+2][j] = \$)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] = *) \wedge (B[i][j-2] = \$) \wedge (B'[i][j] = +) \\ \wedge (B'[i][j-1] = *) \wedge (B'[i][j-2] = \$)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] = *) \wedge (B[i][j+2] = \$) \wedge (B'[i][j] = +) \\ \wedge (B'[i][j+1] = *) \wedge (B'[i][j+2] = \$)$$

Warehouse keeper is on the floor and next to him is a box on goal next to a box on goal:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] =*) \wedge (B[i-2][j] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i-1][j] =*) \wedge (B'[i-2][j] =*)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] =*) \wedge (B[i+2][j] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i+1][j] =*) \wedge (B'[i+2][j] =*)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] =*) \wedge (B[i][j-2] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j-1] =*) \wedge (B'[i][j-2] =*)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] =*) \wedge (B[i][j+2] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j+1] =*) \wedge (B'[i][j+2] =*)$$

Warehouse keeper is on the goal and next to him is a box on goal next to a box on goal:

$$\rho_u = (B[i][j] = +) \wedge (B[i-1][j] =*) \wedge (B[i-2][j] =*) \wedge (B'[i][j] = +) \\ \wedge (B'[i-1][j] =*) \wedge (B'[i-2][j] =*)$$

$$\rho_d = (B[i][j] = +) \wedge (B[i+1][j] =*) \wedge (B[i+2][j] =*) \wedge (B'[i][j] = +) \\ \wedge (B'[i+1][j] =*) \wedge (B'[i+2][j] =*)$$

$$\rho_l = (B[i][j] = +) \wedge (B[i][j-1] =*) \wedge (B[i][j-2] =*) \wedge (B'[i][j] = +) \\ \wedge (B'[i][j-1] =*) \wedge (B'[i][j-2] =*)$$

$$\rho_r = (B[i][j] = +) \wedge (B[i][j+1] =*) \wedge (B[i][j+2] =*) \wedge (B'[i][j] = +) \\ \wedge (B'[i][j+1] =*) \wedge (B'[i][j+2] =*)$$

Warehouse keeper is on the floor and next to him is a box next to a box on goal:

$$\rho_u = (B[i][j] = @) \wedge (B[i-1][j] = \$) \wedge (B[i-2][j] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i-1][j] = \$) \wedge (B'[i-2][j] =*)$$

$$\rho_d = (B[i][j] = @) \wedge (B[i+1][j] = \$) \wedge (B[i+2][j] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i+1][j] = \$) \wedge (B'[i+2][j] =*)$$

$$\rho_l = (B[i][j] = @) \wedge (B[i][j-1] = \$) \wedge (B[i][j-2] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j-1] = \$) \wedge (B'[i][j-2] =*)$$

$$\rho_r = (B[i][j] = @) \wedge (B[i][j+1] = \$) \wedge (B[i][j+2] =*) \wedge (B'[i][j] = @) \\ \wedge (B'[i][j+1] = \$) \wedge (B'[i][j+2] =*)$$

Warehouse keeper is on the goal and next to him is a box next to a box on goal:

$$\rho_u = (B[i][j] = +) \land (B[i-1][j] = \$) \land (B[i-2][j] =*) \land (B'[i][j] = +) \land (B'[i-1][j] = \$) \land (B'[i-2][j] =*)$$

$$\rho_d = (B[i][j] = +) \land (B[i+1][j] = \$) \land (B[i+2][j] =*) \land (B'[i][j] = +) \land (B'[i+1][j] = \$) \land (B'[i+2][j] =*)$$

$$\rho_l = (B[i][j] = +) \land (B[i][j-1] = \$) \land (B[i][j-2] =*) \land (B'[i][j] = +) \land (B'[i][j-1] = \$) \land (B'[i][j-2] =*)$$

$$\rho_r = (B[i][j] = +) \land (B[i][j+1] = \$) \land (B[i][j+2] =*) \land (B'[i][j] = +) \land (B'[i][j+1] = \$) \land (B'[i][j+2] =*)$$

$$u(B,i,j) = !\,((i=0) \lor ((i \geq 1) \land (B[i-1][j] = \#)) \lor ((i \geq 2) \land (B[i-1][j] \in \{*,\$\}) \land (B[i-2][j] \in \{*,\$,\#\}))\,)$$

$$d(B,i,j) = !\,((i=n-1) \lor ((i \leq n-2) \land (B[i+1][j] = \#)) \lor ((i \leq n-3) \land (B[i+1][j] \in \{*,\$\}) \land (B[i+2][j] \in \{*,\$,\#\}))\,)$$

$$l(B,i,j) = !\,((j=0) \lor ((j \geq 1) \land (B[i\,][j-1] = \#)) \lor ((j \geq 2) \land (B[i][j-1] \in \{*,\$\}) \land (B[i][j-2] \in \{*,\$,\#\}))\,)$$

$$r(B,i,j) = !\,((j=n-1) \lor ((j \leq n-2) \land (B[i][j+1] = \#)) \lor ((j \leq n-3) \land (B[i][j+1] \in \{*,\$\}) \land (B[i][j+2] \in \{*,\$,\#\}))\,)$$

u, d, l, r -A Boolean value that holds true if it is possible to move in that direction

$D = \{V, \theta, \rho, J, C\}$

$V = \{\mathrm{B}\,, i\,, j, u, d, l, r\,\}$

$\rho = \rho_u \lor \rho_d \lor \rho_l \lor \rho_r$

$$\theta = (\mathrm{B}\; = input) \land \big(i = \text{Warehouse keeper. row } (@ \lor +)\big) \land \big(j = \text{Warehouse keeper. col } (@ \lor +)\big) \land u(B,i,j) \land d(B,i,j) \land l(B,i,j) \land r(B,i,j)$$

$J =$ if there is an empty space available, the keeper will keep moving into that space infinitely. Similarly, If a box can be moved, it will continue to be moved infinitely until it reaches its target.

$C =$ if a box can be pushed to a goal it eventually must be pushed there.

B - a general n × m Sokoban board in XSB format.

i, j- position of the warehouse keeper on board $0 \leq i \leq n-1$, $0 \leq j \leq m-1$

2. Define a general temporal logic specification for a win of the Sokoban board.

Winning condition: a temporal logic expression that specifies when the game is won. This involves checking if all boxes are on goal cells.

winning_condition = (box1_on_goal && box2_on_goal && ... && boxN_on_goal) or in different words "there are no $ on the board" (there are no boxes on the board that are not on goal)

Express in Temporal Logic:

Winning = finally (\$ $\notin Board$)

# for running nuXmv we used the following commands:

```
nuXmv -int
```

```
read_model -i sokoban.smv (smv file name)
```

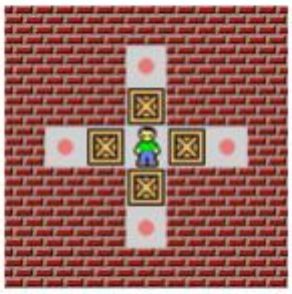
go

check_ltlspec

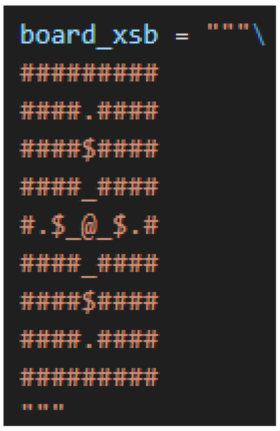
a link to github:

https://github.com/GuyVar/Formal-Verification/tree/main

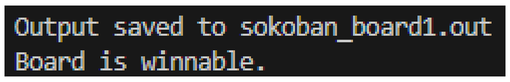
Board 1:



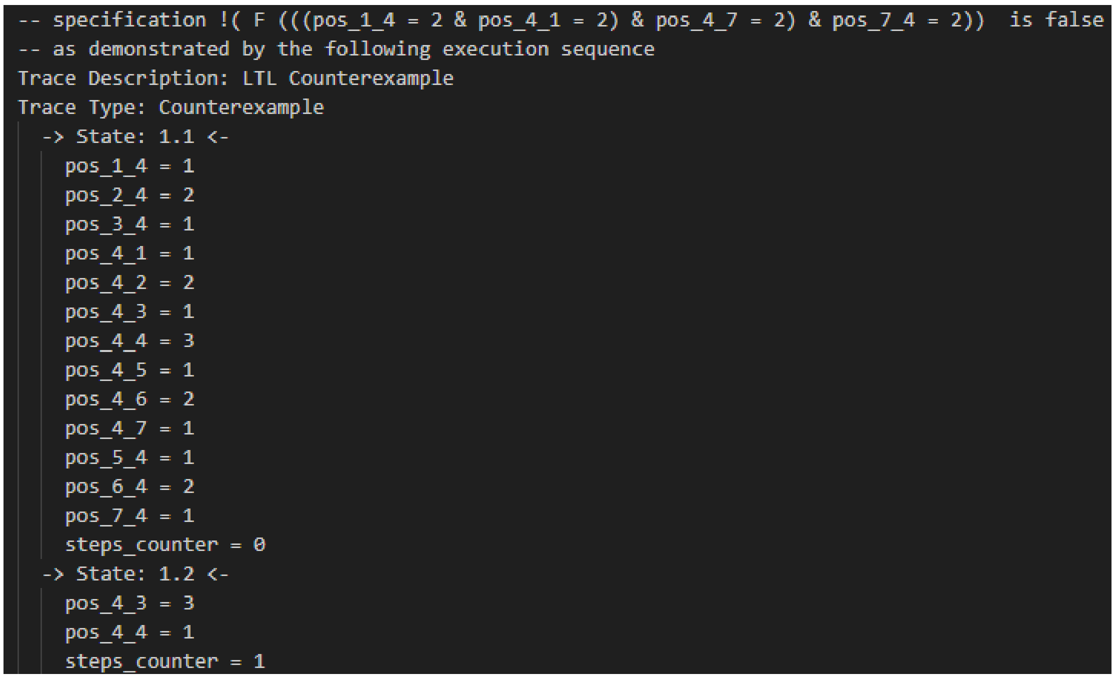
Part 2:



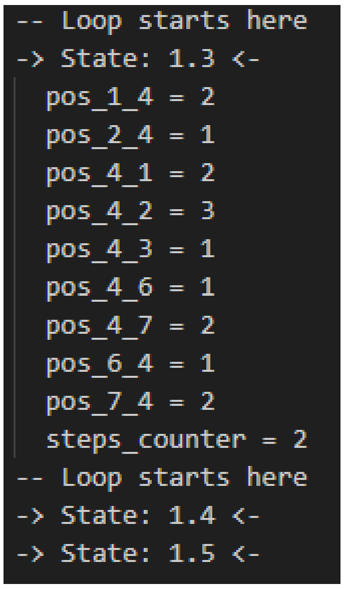
```
board_xsb = """\
#########
####.####
####$####
####_####
#.$_@_$.#
####_####
####$####
####.####
#########
"""
```



```
Output saved to sokoban_board1.out
Board is winnable.
```

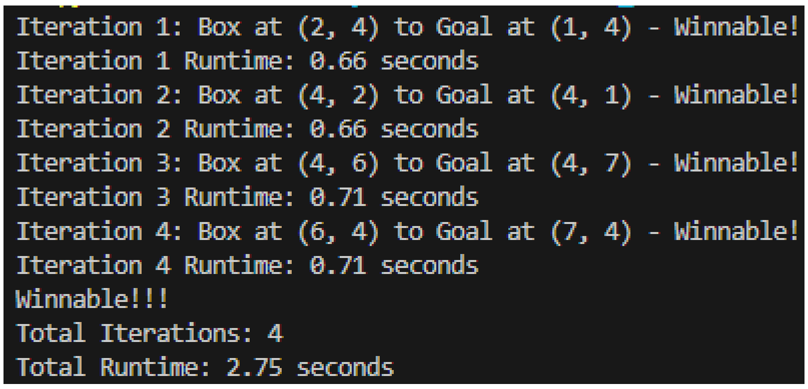


```
-- specification !( F (((pos_1_4 = 2 & pos_4_1 = 2) & pos_4_7 = 2) & pos_7_4 = 2))  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    pos_1_4 = 1
    pos_2_4 = 2
    pos_3_4 = 1
    pos_4_1 = 1
    pos_4_2 = 2
    pos_4_3 = 1
    pos_4_4 = 3
    pos_4_5 = 1
    pos_4_6 = 2
    pos_4_7 = 1
    pos_5_4 = 1
    pos_6_4 = 2
    pos_7_4 = 1
    steps_counter = 0
  -> State: 1.2 <-
    pos_4_3 = 3
    pos_4_4 = 1
    steps_counter = 1
```



```
-- Loop starts here
-> State: 1.3 <-
  pos_1_4 = 2
  pos_2_4 = 1
  pos_4_1 = 2
  pos_4_2 = 3
  pos_4_3 = 1
  pos_4_6 = 1
  pos_4_7 = 2
  pos_6_4 = 1
  pos_7_4 = 2
  steps_counter = 2
-- Loop starts here
-> State: 1.4 <-
-> State: 1.5 <-
```

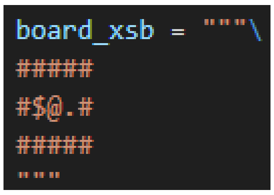
Part 4:



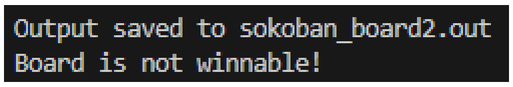
```
Iteration 1: Box at (2, 4) to Goal at (1, 4) - Winnable!
Iteration 1 Runtime: 0.66 seconds
Iteration 2: Box at (4, 2) to Goal at (4, 1) - Winnable!
Iteration 2 Runtime: 0.66 seconds
Iteration 3: Box at (4, 6) to Goal at (4, 7) - Winnable!
Iteration 3 Runtime: 0.71 seconds
Iteration 4: Box at (6, 4) to Goal at (7, 4) - Winnable!
Iteration 4 Runtime: 0.71 seconds
Winnable!!!
Total Iterations: 4
Total Runtime: 2.75 seconds
```

Board 2:

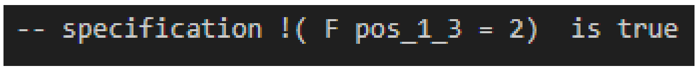
Part 2:



```
board_xsb = """\
#####
#$@.#
#####
"""
```

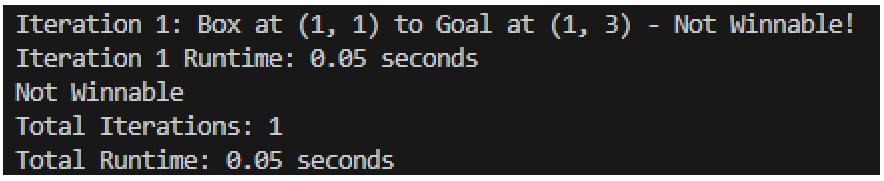


```
Output saved to sokoban_board2.out
Board is not winnable!
```



```
-- specification !( F pos_1_3 = 2)  is true
```

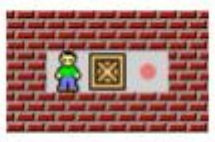
Part 4:

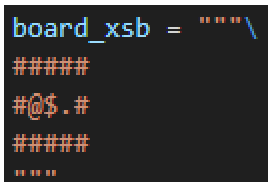


```
Iteration 1: Box at (1, 1) to Goal at (1, 3) - Not Winnable!
Iteration 1 Runtime: 0.05 seconds
Not Winnable
Total Iterations: 1
Total Runtime: 0.05 seconds
```

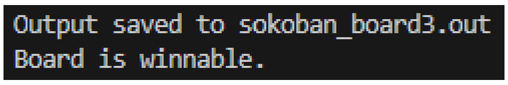
Board 3:

Part 2:

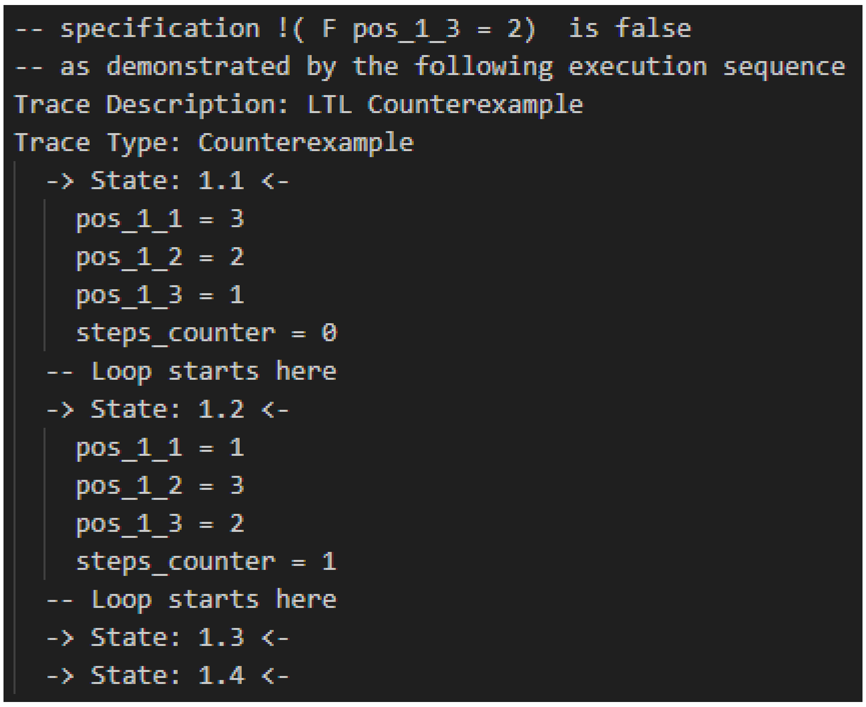




```
board_xsb = """\
#####
#@$.#
#####
"""
```

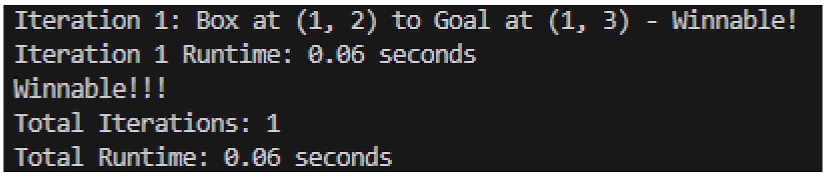


```
Output saved to sokoban_board3.out
Board is winnable.
```



```
-- specification !( F pos_1_3 = 2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    pos_1_1 = 3
    pos_1_2 = 2
    pos_1_3 = 1
    steps_counter = 0
  -- Loop starts here
  -> State: 1.2 <-
    pos_1_1 = 1
    pos_1_2 = 3
    pos_1_3 = 2
    steps_counter = 1
  -- Loop starts here
  -> State: 1.3 <-
  -> State: 1.4 <-
```

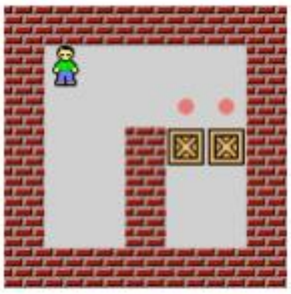
Part 4:

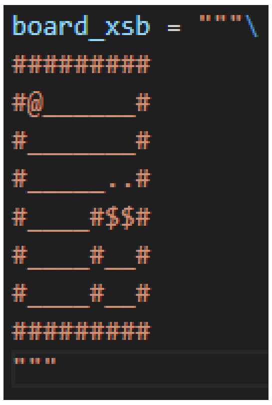


```
Iteration 1: Box at (1, 2) to Goal at (1, 3) - Winnable!
Iteration 1 Runtime: 0.06 seconds
Winnable!!!
Total Iterations: 1
Total Runtime: 0.06 seconds
```

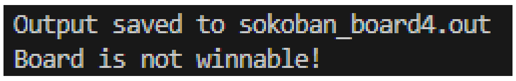
Board 4:

Part 2:

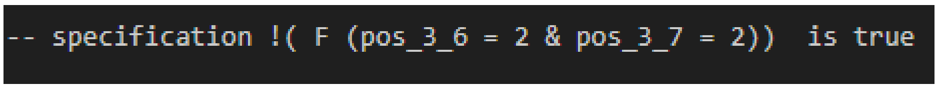




```
board_xsb = """\
#########
#@______#
#_______#
#_____..#
#____#$$#
#____#__#
#____#__#
#########
"""
```

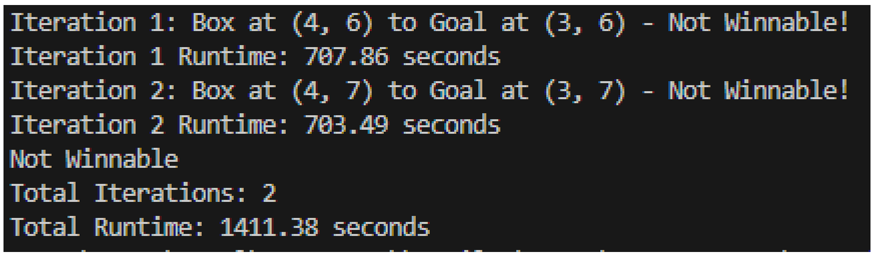


```
Output saved to sokoban_board4.out
Board is not winnable!
```



```
-- specification !( F (pos_3_6 = 2 & pos_3_7 = 2))  is true
```

Part 4:

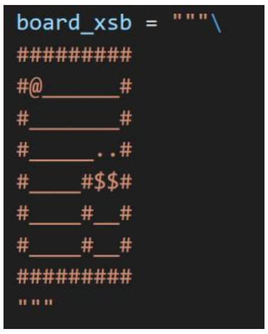


```
Iteration 1: Box at (4, 6) to Goal at (3, 6) - Not Winnable!
Iteration 1 Runtime: 707.86 seconds
Iteration 2: Box at (4, 7) to Goal at (3, 7) - Not Winnable!
Iteration 2 Runtime: 703.49 seconds
Not Winnable
Total Iterations: 2
Total Runtime: 1411.38 seconds
```

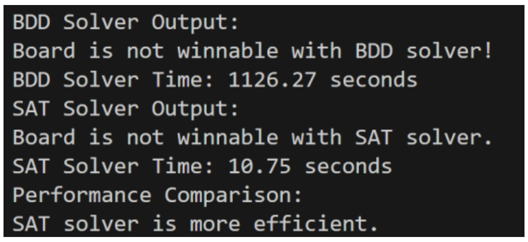
Part 3:

The choice between BDD and SAT-based model checking depends on the specific characteristics of the problem at hand. BDDs are advantageous for problems with a high degree of regularity and require symbolic manipulation, but they suffer from memory limitations. SAT solvers, on the other hand, excel in handling large, sparse, and complex state spaces efficiently, benefiting from modern advancements in solver technology. For many modern verification tasks, SAT-based approaches are preferred due to their scalability and performance.

Not winnable board, SAT is more efficient:

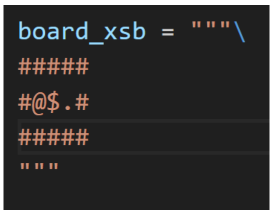


```
board_xsb = """\
#########
#@______#
#_______#
#_____..#
#____#$$#
#____#__#
#____#__#
#########
"""
```

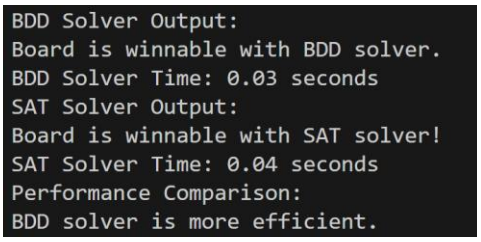


```
BDD Solver Output:
Board is not winnable with BDD solver!
BDD Solver Time: 1126.27 seconds
SAT Solver Output:
Board is not winnable with SAT solver.
SAT Solver Time: 10.75 seconds
Performance Comparison:
SAT solver is more efficient.
```

Winnable board, BDD is more efficient:

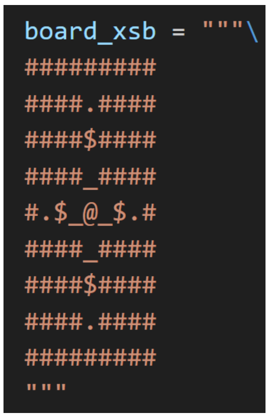


```
board_xsb = """\
#####
#@$.#
#####
"""
```

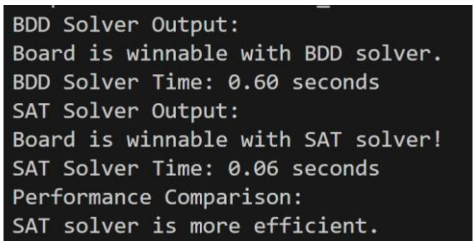


```
BDD Solver Output:
Board is winnable with BDD solver.
BDD Solver Time: 0.03 seconds
SAT Solver Output:
Board is winnable with SAT solver!
SAT Solver Time: 0.04 seconds
Performance Comparison:
BDD solver is more efficient.
```

Winnable board, SAT is more efficient:

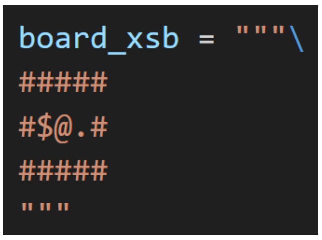


```
board_xsb = """\
#########
####.####
####$####
####_####
#.$_@_$.#
####_####
####$####
####.####
#########
"""
```

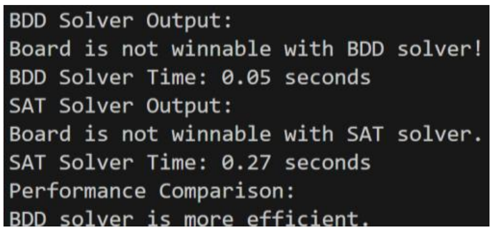


```
BDD Solver Output:
Board is winnable with BDD solver.
BDD Solver Time: 0.60 seconds
SAT Solver Output:
Board is winnable with SAT solver!
SAT Solver Time: 0.06 seconds
Performance Comparison:
SAT solver is more efficient.
```

Not winnable board, BDD is more efficient:



```
board_xsb = """\
#####
#$@.#
#####
"""
```



```
BDD Solver Output:
Board is not winnable with BDD solver!
BDD Solver Time: 0.05 seconds
SAT Solver Output:
Board is not winnable with SAT solver.
SAT Solver Time: 0.27 seconds
Performance Comparison:
BDD solver is more efficient.
```

As we can see, for structural boards the bdd engine is more efficient and for more complicated boards the sat engine is more efficient.