Documentation

We strive to fulfill all extra points in this lab. The file `ship3.1.smv` includes the basic implementation and assignment 3.1, `ship3.2.smv` assignment 3.2 and `ship3.3.svm` assignment 3.3. In addition to this, we created this file for the fairness analysis (3.2), error traces, the simulation, and the model analysis (3.4).

3.2 Fairness analysis: What happens without the new fairness property?

Removing the fairness condition breaks the following specifications:

```
-- specification AF outer_door.status = open  is false
-- specification AG (outer_buttons.pressed -> AF !outer_buttons.pressed)  is false
-- specification  G (!airlock.reset_outer U outer_door.status = open)  is false
-- specification AF inner_door.status = open  is false
-- specification AG (inner_buttons.pressed -> AF !inner_buttons.pressed)  is false
-- specification  G (!airlock.reset_inner U inner_door.status = open)  is false
```

We decided to analyze the error trace of the first specification, `AF outer_door.status = open`, that broke when removing the fairness conditions. In this scenario, at state 1.3, the `inner_door_status` is first set to open simultaneously as the `inner_door.sensor` is set to TRUE. From here a loop is started where either the `inner_door_sensor` is TRUE or the `airlock.inner_door_cmd` is set to nop. Even though `airlock.inner_door_cmd` is set to close in every second state, this only occurs when `inner_door_sensor` is TRUE, which means that the `inner_door` can never close. The `inner_door` being open forever implies that the `outer_door` can never open. The error trace could be seen below:

Fairness condition

```
-- specification AF outer_door.status = open  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    inner_door_sensor = FALSE
    outer_door_sensor = FALSE
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons.pressed = FALSE
```

```
    outer_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
  -> State: 1.2 <-
    inner_buttons.pressed = TRUE
    outer_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
  -> State: 1.3 <-
    inner_door_sensor = TRUE
    inner_door.status = open
    airlock.inner_door_cmd = close
    airlock.reset_inner = TRUE
  -- Loop starts here
  -> State: 1.4 <-
    inner_door_sensor = FALSE
    inner_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.last_open = inner
    airlock.reset_inner = FALSE
  -> State: 1.5 <-
    inner_door_sensor = TRUE
    inner_buttons.pressed = TRUE
    airlock.inner_door_cmd = close
    airlock.reset_inner = TRUE
  -> State: 1.6 <-
    inner_door_sensor = FALSE
    inner_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.reset_inner = FALSE
```

3.4.1 Model error trace (+0.25)

```
    inner_door_cmd :=
      case
        -- Specify when the door should open or close
-->     (inner_door != open) & inner_buttons & outer_buttons : open;
        outer_door = closed & inner_door = closed & inner_buttons : open;
        reset_inner : close;
        TRUE: nop;
      esac;
```

Without the added `inner_door != open` condition, we got the follwing errors:

The first and the second error

For the first and the second error, the properties `outer_door.status = open` and `AG (outer_buttons.pressed -> AF !outer_buttons.pressed)` fails. This occurs because of a loop where the `inner_door` is constantly open. `inner_door` being always open implies that `outer_door` never opens, which is why `outer_button` never reset and are pressed forever.

The third error

For the third error, the property the when both buttons are pressed fails. The issue raises because of the button invariant. The door opens normally, then the double press case runs. Which means that the door sent the open command twice.

The full error trace:

```
-- specification AF outer_door.status = open  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons.pressed = FALSE
    outer_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
  -> State: 2.2 <-
    inner_buttons.pressed = TRUE
    outer_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
  -> State: 2.3 <-
    inner_door.status = open
    airlock.reset_inner = TRUE
  -- Loop starts here
  -> State: 2.4 <-
    inner_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.last_open = inner
    airlock.reset_inner = FALSE
  -> State: 2.5 <-
    inner_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
    airlock.reset_inner = TRUE
  -> State: 2.6 <-
    inner_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.reset_inner = FALSE
-- specification AG (inner_buttons.pressed -> AF !inner_buttons.pressed)  is true
-- specification AG (outer_buttons.pressed -> AF !outer_buttons.pressed)  is false
```

```
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 3.1 <-
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons.pressed = FALSE
    outer_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
  -> State: 3.2 <-
    inner_buttons.pressed = TRUE
    outer_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
  -> State: 3.3 <-
    inner_door.status = open
    airlock.reset_inner = TRUE
  -- Loop starts here
  -> State: 3.4 <-
    inner_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.last_open = inner
    airlock.reset_inner = FALSE
  -> State: 3.5 <-
    inner_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
    airlock.reset_inner = TRUE
  -> State: 3.6 <-
    inner_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.reset_inner = FALSE
-- specification AG ((((inner_buttons.pressed & outer_buttons.pressed) & inner_door
-- invariant (door_cmd = open -> status = closed) IN inner_door is false
-- as demonstrated by the following execution sequence
Trace Description: AG alpha Counterexample
Trace Type: Counterexample
  -> State: 4.1 <-
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons.pressed = FALSE
    outer_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
  -> State: 4.2 <-
    inner_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
  -> State: 4.3 <-
    inner_door.status = open
    outer_buttons.pressed = TRUE
    airlock.reset_inner = TRUE
```

3.4.2 Property error trace (+0.25)

The specification `AG (inner_door.status = open -> AX airlock.reset_inner)` fails since `airlock.reset_inner` changes directly when `inner_door.status` is set to open. The correct specification is: `AG (inner_door.status = open -> airlock.reset_inner)`.

```
-- specification AG (inner_door.status = open -> AX airlock.reset_inner)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons.pressed = FALSE
    outer_buttons.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
  -> State: 1.2 <-
    inner_buttons.pressed = TRUE
    airlock.inner_door_cmd = open
  -> State: 1.3 <-
    inner_door.status = open
    airlock.inner_door_cmd = close
    airlock.reset_inner = TRUE
  -> State: 1.4 <-
    inner_door.status = closed
    inner_buttons.pressed = FALSE
    outer_buttons.pressed = TRUE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = open
    airlock.last_open = inner
    airlock.reset_inner = FALSE
```

3.4.3 Simulation

We used the commands below to perform the simulation. To start with, we used the commands `NuSMV -int {FILENAME}` and `go` to load our model into the NUSMV system.

```
NuSMV -int {FILENAME}
go
```

The command `pick_state -i` was then used to interactively choose the state to build the trace from. We decided to pick state 0 here.

```
NuSMV > pick_state -i
Chosen state is: 0
```

The command `simulate -r -k 6` was then used to build a six-step simulation trace. The random flag `-r` indicates that the next step is picked randomly at each step and the length flag `-k` together with `6` sets the length of the simulation to 6 steps. One can also perform the simulation without the flag `-r` to manually choose the next step in each state.

```
NuSMV > simulate -r -k 6
```

We then used the `show_traces` command to print the simulated trace. The flag `-t` prints the total number of currently stored traces and the flag `-v` verbosely prints the traces.

```
NuSMV > show_traces -t
NuSMV > show_traces -v
```

The full trace:

```
$ nusmv -int ship3.3.smv
*** This is NuSMV 2.6.0 (compiled on Sat Apr 17 17:24:37 2021)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson
```

```
NuSMV > go
NuSMV > simulate -r -k 6
No current state set. Use the "pick_state" command.
NuSMV > pick_state -i

***************  AVAILABLE STATES  *************

  ================= State =================
  0) -------------------------
  outer_buttons.pressed = FALSE
  inner_buttons.pressed = FALSE
  inner_door_sensor = TRUE
  outer_door_sensor = TRUE
  state = overheat
  inner_door.status = closed
  outer_door.status = closed
  inner_buttons_o.pressed = FALSE
  inner_buttons_i.pressed = FALSE
  outer_buttons_o.pressed = FALSE
  outer_buttons_i.pressed = FALSE
  airlock.outer_buttons_combine = FALSE
  airlock.inner_buttons_combine = FALSE
  airlock.inner_door_cmd = nop
  airlock.outer_door_cmd = nop
  airlock.last_open = none
  airlock.reset_inner = FALSE
  airlock.reset_outer = FALSE


  ================= State =================
  1) -------------------------
  inner_door_sensor = FALSE


  ================= State =================
  2) -------------------------
  inner_door_sensor = TRUE
  outer_door_sensor = FALSE


  ================= State =================
  3) -------------------------
  inner_door_sensor = FALSE


  ================= State =================
  4) -------------------------
  inner_door_sensor = TRUE
  outer_door_sensor = TRUE
  state = normal


  ================= State =================
  5) -------------------------
  inner_door_sensor = FALSE
```

```
  ================= State =================
  6) ------------------------
  inner_door_sensor = TRUE
  outer_door_sensor = FALSE


  ================= State =================
  7) ------------------------
  inner_door_sensor = FALSE


Choose a state from the above (0-7): 0

Chosen state is: 0
NuSMV > simulate -r -k 6
********  Simulation Starting From State 1.1   ********
NuSMV > show_traces -t
There is 1 trace currently available.
NuSMV > show_traces -v
    <!-- ################### Trace number: 1 ################### -->
Trace Description: Simulation Trace
Trace Type: Simulation
  -> State: 1.1 <-                    -- The initial state with all sensors TRUE ,al
    inner_door_sensor = TRUE
    outer_door_sensor = TRUE
    state = overheat
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons_o.pressed = FALSE
    inner_buttons_i.pressed = FALSE
    outer_buttons_o.pressed = FALSE
    outer_buttons_i.pressed = FALSE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
    outer_buttons.pressed = FALSE
    inner_buttons.pressed = FALSE
    airlock.outer_buttons_combine = FALSE
    airlock.inner_buttons_combine = FALSE
  -> State: 1.2 <-
    inner_door_sensor = FALSE
    outer_door_sensor = TRUE          -- Something in the way for the outer sensor
    state = overheat                  -- In overheat mode
    inner_door.status = closed
    outer_door.status = closed
    inner_buttons_o.pressed = FALSE
    inner_buttons_i.pressed = TRUE    -- The inside button of inner_door is pressed
    outer_buttons_o.pressed = FALSE
    outer_buttons_i.pressed = TRUE    -- The inside button of outer_door is pressed
    airlock.inner_door_cmd = open     -- tries to open inner_door
    airlock.outer_door_cmd = nop      -- no operation since inner door takes precede
    airlock.last_open = none
```

```
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
    outer_buttons.pressed = TRUE
    inner_buttons.pressed = TRUE
    airlock.outer_buttons_combine = TRUE
    airlock.inner_buttons_combine = TRUE
  -> State: 1.3 <-
    inner_door_sensor = FALSE
    outer_door_sensor = FALSE        -- both sensors become FALSE
    state = normal                   -- state becomes normal
    inner_door.status = open         -- the inner_door becomes open
    outer_door.status = closed
    inner_buttons_o.pressed = FALSE
    inner_buttons_i.pressed = TRUE
    outer_buttons_o.pressed = TRUE   -- the outside button of outer door is pressed
    outer_buttons_i.pressed = TRUE
    airlock.inner_door_cmd = close   -- trying to close the inner_door
    airlock.outer_door_cmd = nop
    airlock.last_open = none
    airlock.reset_inner = TRUE       -- trying to reset inner door's buttons
    airlock.reset_outer = FALSE
    outer_buttons.pressed = TRUE
    inner_buttons.pressed = TRUE
    airlock.outer_buttons_combine = TRUE
    airlock.inner_buttons_combine = TRUE
  -> State: 1.4 <-
    inner_door_sensor = FALSE
    outer_door_sensor = TRUE
    state = normal                   -- in the normal state
    inner_door.status = closed       -- the inne_door is now closed
    outer_door.status = closed
    inner_buttons_o.pressed = FALSE
    inner_buttons_i.pressed = FALSE  -- inner door's buttons are reset
    outer_buttons_o.pressed = TRUE
    outer_buttons_i.pressed = TRUE
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = open    -- trying to open the outer door
    airlock.last_open = inner        -- recording the last opened door is the inner
    airlock.reset_inner = FALSE
    airlock.reset_outer = FALSE
    outer_buttons.pressed = TRUE
    inner_buttons.pressed = FALSE
    airlock.outer_buttons_combine = TRUE
    airlock.inner_buttons_combine = FALSE
  -> State: 1.5 <-
    inner_door_sensor = TRUE
    outer_door_sensor = TRUE         -- both sensors are TRUE
    state = overheat                 -- overheat state again
    inner_door.status = closed
    outer_door.status = open         -- outer door is open
    inner_buttons_o.pressed = FALSE
    inner_buttons_i.pressed = FALSE
    outer_buttons_o.pressed = TRUE   -- the outside button of outer door is pressed
    outer_buttons_i.pressed = TRUE   -- the inside button of outer door is pressed
    airlock.inner_door_cmd = nop
    airlock.outer_door_cmd = close   -- trying to close the outer door
```

```
      airlock.last_open = inner
      airlock.reset_inner = FALSE
      airlock.reset_outer = TRUE       -- trying to reset the outer buttons
      outer_buttons.pressed = TRUE
      inner_buttons.pressed = FALSE
      airlock.outer_buttons_combine = TRUE
      airlock.inner_buttons_combine = FALSE
    -> State: 1.6 <-
      inner_door_sensor = FALSE
      outer_door_sensor = FALSE
      state = normal                   -- state back to normal again
      inner_door.status = closed
      outer_door.status = open         -- the outer door is opened and inner door is
      inner_buttons_o.pressed = FALSE
      inner_buttons_i.pressed = FALSE
      outer_buttons_o.pressed = FALSE
      outer_buttons_i.pressed = FALSE  -- no button is pressed
      airlock.inner_door_cmd = nop
      airlock.outer_door_cmd = nop     -- no command sent
      airlock.last_open = outer        -- recording last opened door as outer door
      airlock.reset_inner = FALSE
      airlock.reset_outer = FALSE
      outer_buttons.pressed = FALSE
      inner_buttons.pressed = FALSE
      airlock.outer_buttons_combine = FALSE
      airlock.inner_buttons_combine = FALSE
    -> State: 1.7 <-               -- The outer door doesn't close since there was so
      inner_door_sensor = FALSE
      outer_door_sensor = FALSE
      state = normal
      inner_door.status = closed
      outer_door.status = open
      inner_buttons_o.pressed = TRUE   -- the inner door button is pressed but the in
      inner_buttons_i.pressed = FALSE
      outer_buttons_o.pressed = FALSE
      outer_buttons_i.pressed = FALSE
      airlock.inner_door_cmd = nop
      airlock.outer_door_cmd = nop
      airlock.last_open = outer
      airlock.reset_inner = FALSE
      airlock.reset_outer = FALSE
      outer_buttons.pressed = FALSE
      inner_buttons.pressed = TRUE
      airlock.outer_buttons_combine = FALSE
      airlock.inner_buttons_combine = TRUE
```

3.4.4 Analysis

As seen by the example above. The specification does not handle closing process of a door that has been interrupted by one of the sensors. When a sensor goes from the state TRUE to

FALSE, we need to add a feature that actually closes the door after.