F20RS Coursework 2 Report

A SPIN Design Modelling Exercise

Heriot-Watt University

November 29th 2021

By Drew Tomkins (H00294338)

Table of Contents

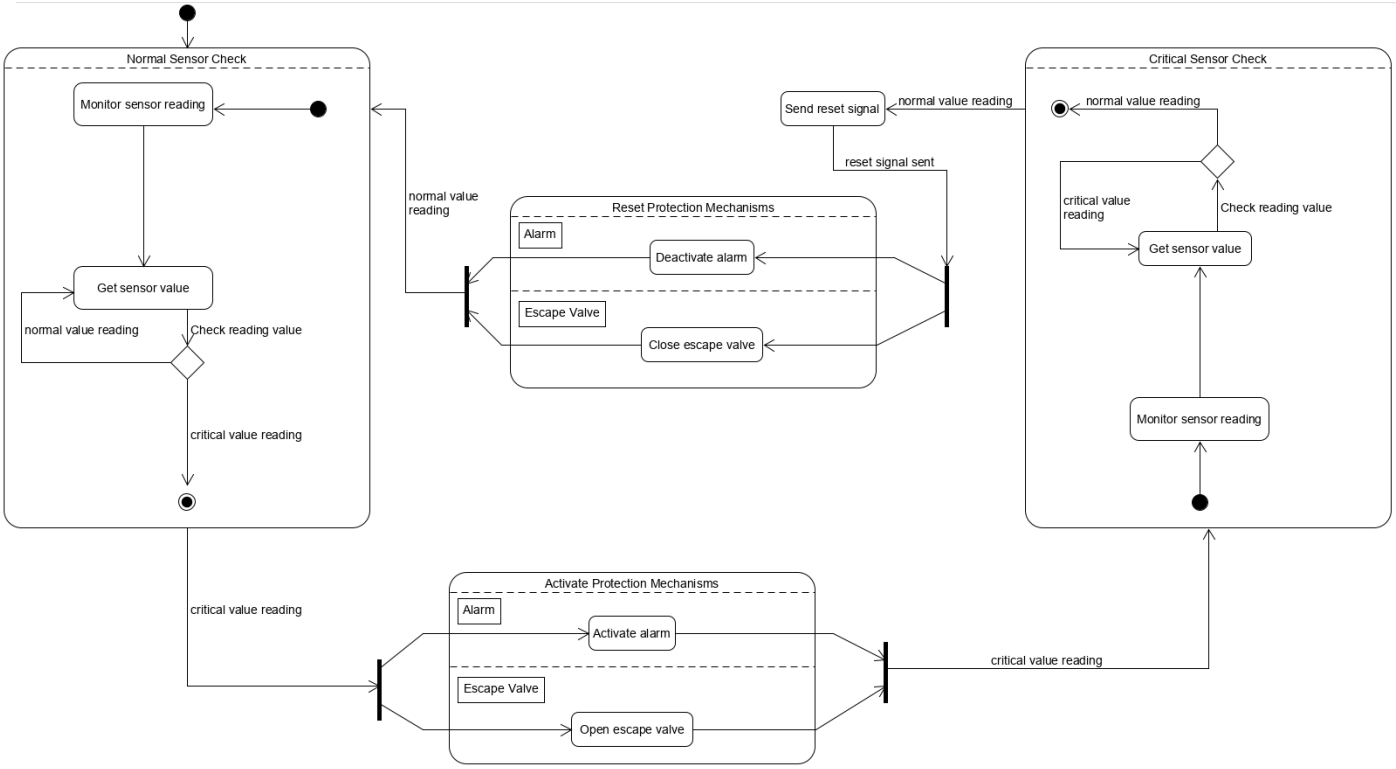
1. Introduction

This project involved creating an Automatic Vessel Protection (AVP) system using the Promela programming language that prevents explosions on vessels by handling high-pressure hazards in the process. This report contains information regarding the project, such as assumptions made before development, a state machine diagram of the AVP system, listings of the source file used to create the system and the verifications of this source file in Spin/iSpin.

2. Assumptions Made Before Development

- The monitored variables I would be including for controlling the environment are the sensor values themselves and the type of reading based off of the sensor values (normal, critical) so the controller can read the values in to alter aspects of the system.
- The controlled variables I would be including for controlling the environment would be the alarm state and the escape valve state, both of which would depend on the monitored variables values to change their status.
- The values used for the normal and critical readings in the environment will be based off of the values provided in coursework 1, with 0-149 consisting of normal values and 150-199 consisting of critical values - these will be used to control the opening and closing of the escape valve alongside triggering the alarm.

3. State Machine Diagram of System

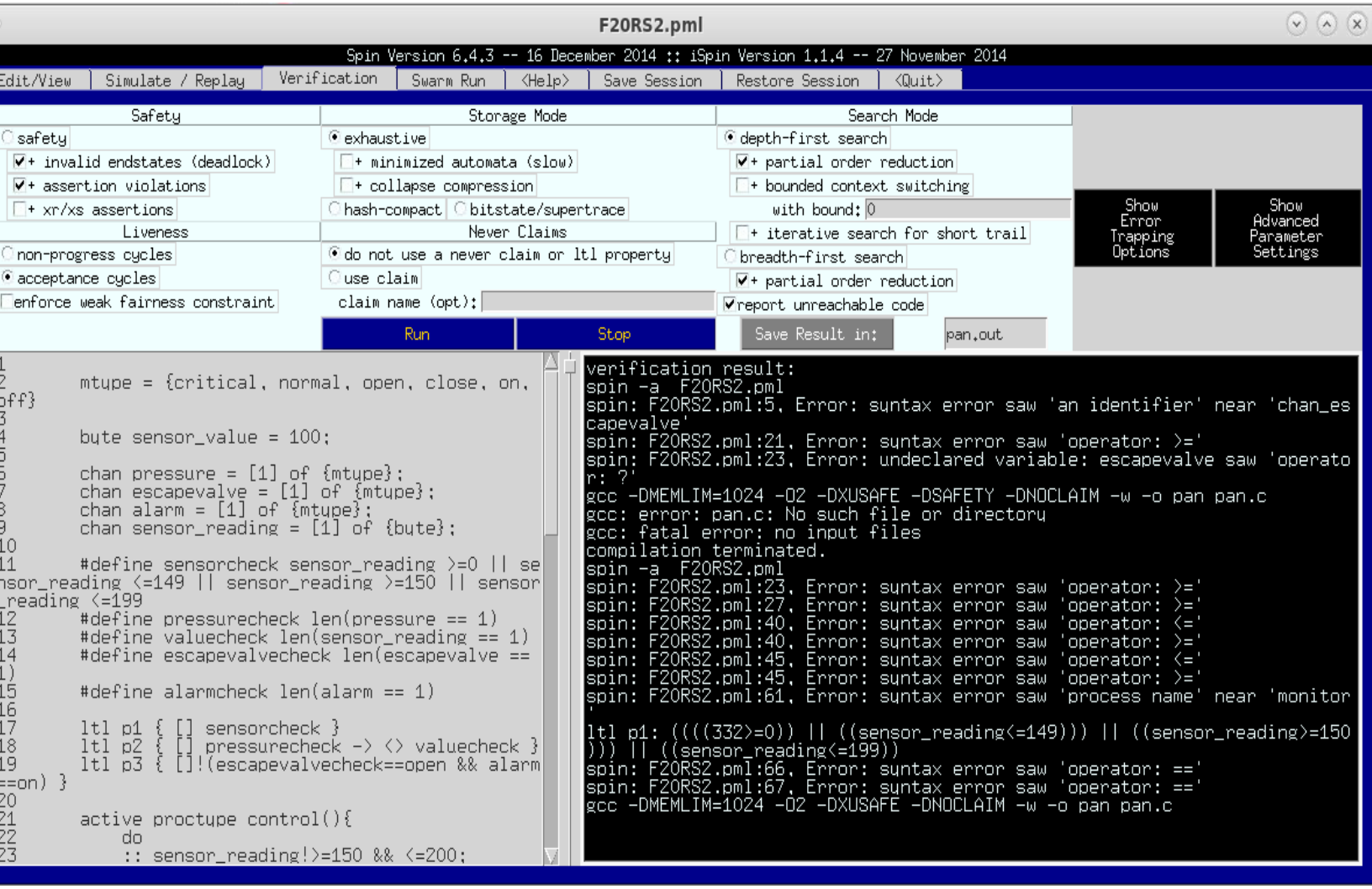


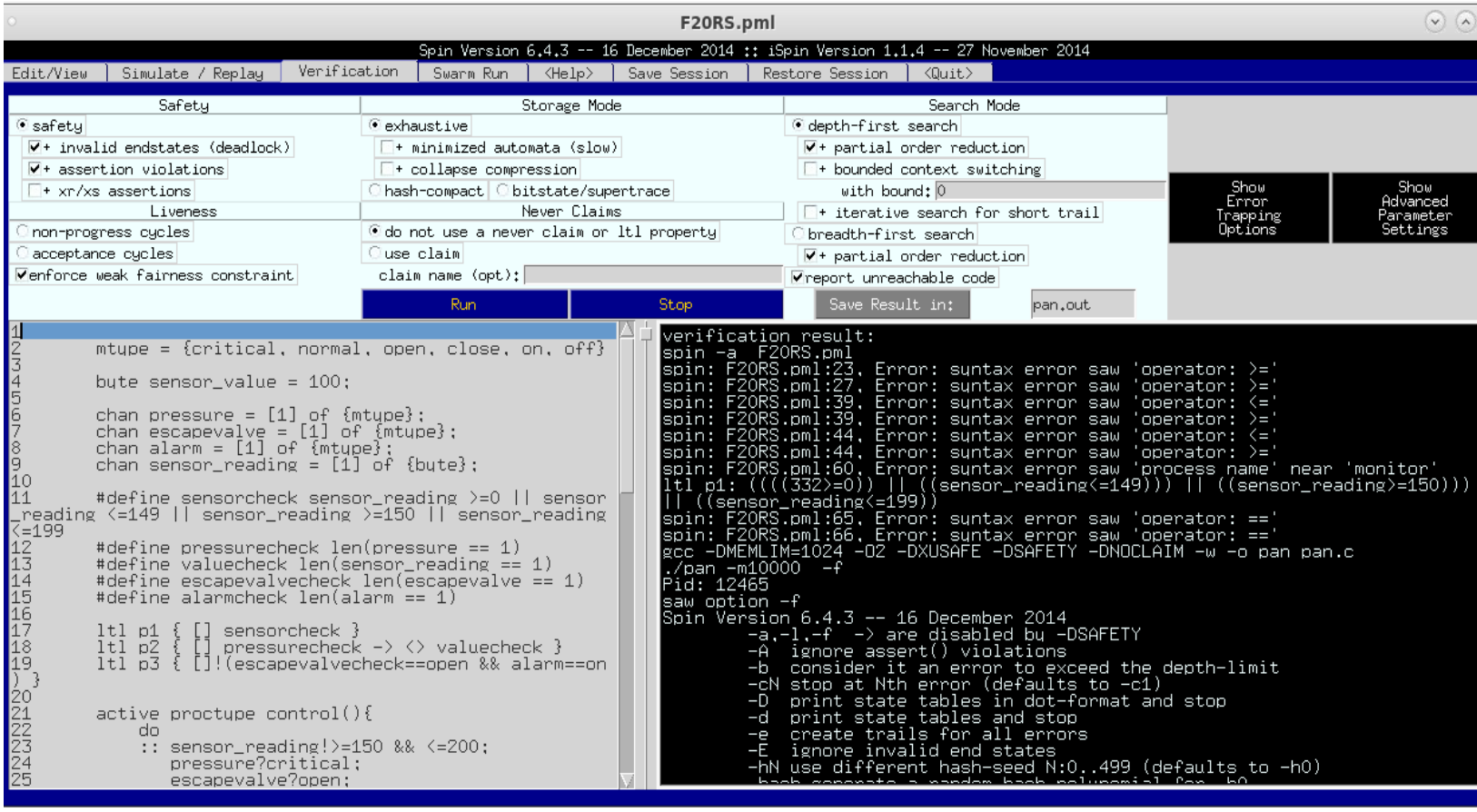
4. Listing of Source File

```
mtype = {critical, normal, open, close, on, off}

byte sensor_value = 100;

chan pressure = [1] of {mtype};
chan escapevalve = [1] of {mtype};
chan alarm = [1] of {mtype};
chan sensor_reading = [1] of {byte};

#define sensorcheck sensor_reading >=0 || sensor_reading <=149 || sensor_reading >=150 || sensor_reading <=199
#define pressurecheck len(pressure == 1)
#define valuecheck len(sensor_reading == 1)
#define escapevalvecheck len(escapevalve == 1)
#define alarmcheck len(alarm == 1)

ltl p1 { [] sensorcheck }
ltl p2 { [] pressurecheck -> <> valuecheck }
ltl p3 { []!(escapevalvecheck==open && alarm==on) }

active proctype control(){
    do
    :: sensor_reading!>=150 && <=200;
       pressure?critical;
       escapevalve?open;
       alarm?on;
    :: sensor_reading!>=0 && <=149;
       pressure?normal;;
       escapevalve?close;
       alarm?off;

    od
}


active proctype env(){

    do
    :: (sensor_value >=150 && <=199) -> sensor_reading?>=150 && <=200;
                                        pressure!critical;
                                        escapevalve!open;
                                        alarm!on;
                                        sensor_value = sensor_value-1;
```

```
    :: (sensor_value >=0 && <=149) -> sensor_reading?>=0 && <=149;
                                      pressure!normal;
                                      escapevalve!close;
                                      alarm!off;
                                      sensor_value = sensor_value+1;
    od
}


 active proctype monitor(){
    do :: assert sensor_value >= 0; sensor_value <= 149 ||
                  sensor_value >= 150; sensor_value <= 199;
    od
}

init{
       ryn monitor();
       run control();
       run env();
}
```

5. Verification Runs and Parameters




F2ORS.pml
Spin Version 6.4.3 -- 16 December 2014 :: iSpin Version 1.1.4 -- 27 November 2014
Edit/View
Simulate / Replay
Verification
Swarm Run
<Help>
Save Session
Restore Session
<Quit>
Safety
safety
+ invalid endstates (deadlock)
+ assertion violations
+ xr/xs assertions
Liveness
non-progress cycles
acceptance cycles
enforce weak fairness constraint
Storage Mode
exhaustive
+ minimized automata (slow)
+ collapse compression
hash-compact
bitstate/supertrace
Never Claims
do not use a never claim or ltl property
use claim
claim name (opt):
Search Mode
depth-first search
+ partial order reduction
+ bounded context switching
with bound: 0
+ iterative search for short trail
breadth-first search
+ partial order reduction
report unreachable code
Show Error Trapping Options
Show Advanced Parameter Settings
Run
Stop
Save Result in:
pan.out
1
2 mtype = {critical, normal, open, close, on, off}
3
4 byte sensor_value = 100;
5
6 chan pressure = [1] of {mtype};
7 chan escapevalve = [1] of {mtype};
8 chan alarm = [1] of {mtype};
9 chan sensor_reading = [1] of {byte};
10
11 #define sensorcheck sensor_reading >=0 || sensor
_reading <=149 || sensor_reading >=150 || sensor_reading
<=199
12 #define pressurecheck len(pressure == 1)
13 #define valuecheck len(sensor_reading == 1)
14 #define escapevalvecheck len(escapevalve == 1)
15 #define alarmcheck len(alarm == 1)
16
17 ltl p1 { [] sensorcheck }
18 ltl p2 { [] pressurecheck -> <> valuecheck }
19 ltl p3 { []!(escapevalvecheck==open && alarm==on
) }
20
21 active proctype control(){
22 do
23 :: sensor_reading!>=150 && <=200;
24 pressure?critical;
25 escapevalve?open;
verification result:
spin -a F2ORS.pml
spin: F2ORS.pml:23, Error: suntax error saw 'operator: >='
spin: F2ORS.pml:27, Error: suntax error saw 'operator: >='
spin: F2ORS.pml:39, Error: suntax error saw 'operator: <='
spin: F2ORS.pml:39, Error: suntax error saw 'operator: >='
spin: F2ORS.pml:44, Error: suntax error saw 'operator: <='
spin: F2ORS.pml:44, Error: suntax error saw 'operator: >='
spin: F2ORS.pml:60, Error: suntax error saw 'process name' near 'monitor'
ltl p1: ((((332>=0)) || ((sensor_reading<=149))) || ((sensor_reading>=150)))
|| ((sensor_reading<=199))
spin: F2ORS.pml:65, Error: suntax error saw 'operator: =='
spin: F2ORS.pml:66, Error: suntax error saw 'operator: =='
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000 -f
Pid: 12465
saw option -f
Spin Version 6.4.3 -- 16 December 2014
-a,-l,-f -> are disabled by -DSAFETY
-A ignore assert() violations
-b consider it an error to exceed the depth-limit
-cN stop at Nth error (defaults to -c1)
-D print state tables in dot-format and stop
-d print state tables and stop
-e create trails for all errors
-E ignore invalid end states
-hN use different hash-seed N:0..499 (defaults to -h0)