Assignment Part2 (Promela and Spin)

Software Engineering

Name: Dushyanth

Roll No: 18CS01009

Q1. Model the elevator's controller in the controller proctype. Explain your modelling choices (e.g., the used Promela statements, sequence of statements to be executed based on the user choice, etc.).

Modelling: The variables – **at** (denotes the current floor of the elevator) and **closed** (indicates if doors of the elevator are closed) are declared as global variables.

File name: elevator_Q1.pml

In the elevator controller, a 'do .. od' statement is used with 8 alternatives in it. The guards are the receiving operations on **elevator_buttons** (4 alternatives one each for go_0, go_1, go_2, go_3) and **floor_buttons** (4 alternatives one each for call_0, call_1, call_2, call_3) with pattern matching.

Consider the below alternative that represents the selection of 1st floor button in the elevator.

```
61          :: elevator_buttons ? go_1 ->
62              if
63                  :: at==1 ->
64                      commands ! open;
65                      closed = false;
66                      commands ! close;
67                      closed = true;
68                  :: at==0 ->
69                      commands ! up;
70                      at++;
71                      commands ! open;
72                      closed = false;
73                      commands ! close;
74                      closed = true;
75                  :: else ->
76                      do
77                          :: at>1 ->
78                              commands ! down;
79                              at--;
80                          :: else -> break;
81                      od
82                      commands ! open;
83                      closed = false;
84                      commands ! close;
85                      closed = true;
86              fi
```

The elevator should move to 1st floor. Since **at** denotes the current floor of the elevator, 'if .. if' statement is used to direct the elevator based on the current location of the elevator.

Case-1: If the elevator is in the 1st floor then the doors need to be opened (implemented by sending 'open' message over the 'commands' channel) and then the doors need to be closed (implemented by sending 'close' message over the 'commands' channel)

Case-2: If the elevator is at 0th floor it needs to be moved up in-order to reach 1st floor (implemented by sending 'up' message over 'commands' channel). The door is opened at the first floor and then closed.

Case-3: If the elevator is at floors higher than 1 then it needs to be moved down until it reaches floor 1 (implemented by using 'do .. od' statement and sending the 'down' commands over the channel 'commands').

In this way the remaining code of the elevator controller is designed and implemented.

Q2. Add assertions to the Promela model that trigger if the elevator receives the up message while at floor 3 or if the elevator receives the down message while at floor 0. (Please include the full Promela model in the submitted solution.) Check with Spin whether the assertions are triggered. Explain the obtained result. If the assertions are triggered, provide a corrected model. (Please include a Spin run log with your explanations.)

File Name: elevator_Q2.pml

Modelling: The same model as that of the question-1 is used with the below modification in the **elevator process**:

```
13  active proctype elevator() {
14      do
15      :: commands ? open -> printf("Elevator: opened doors.\n");
16      :: commands ? close -> printf("Elevator: closed doors.\n");
17      :: commands ? up -> assert(at!=3);printf("Elevator: moved up one floor.\n");
18      :: commands ? down -> assert(at!=0);printf("Elevator: moved down one floor.\n");
19      od
20  }
```

at variable is globally declared.

When the elevator receives **up** command, the assert statement gets triggered if the elevator is at 3rd floor.

When the elevator receives **down** command, the assert statement gets triggered if the elevator is at 0th floor.

Steps for execution:

1. spin –a elevator_Q2.pml
2. gcc –o pan pan.c
3. ./pan

Execution steps and output:

```
dush123@ubuntu:~/Desktop/se/Q2$ spin -a elevator_Q2.pml
dush123@ubuntu:~/Desktop/se/Q2$ gcc -o pan pan.c
```

```
dush123@ubuntu:~/Desktop/se/Q2$ ./pan
pan:1: assertion violated (at!=0) (at depth 70)
pan: wrote elevator_Q2.pml.trail

(Spin Version 6.4.9 -- 17 December 2018)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid end states      +

State-vector 68 byte, depth reached 71, errors: 1
       60 states, stored
        5 states, matched
       65 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.005       equivalent memory usage for states (stored*(State-vector + overhead))
    0.270       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage



pan: elapsed time 0 seconds
```

We can see that there are errors detected by the verifier.

Now, we can view the failing run using the below command:

1. Spin –t –p –l –g elevator_Q2.pml > error.txt

The failing run is redirected to **error.txt** file and attached in the submission.

Explanation: Consider the code segment starting from the line 50 in 'elevator_Q2.pml' file of the **controller** process

```
50                  do
51                  :: at>0 ->
52                      commands ! down;
53                      at--;
54                  :: else -> break;
55                  od
```

At 52 line **down** message is sent over the **commands** channel and since it's a rendezvous channel, it blocks for the receiver to receive.

```
13  active proctype elevator() {
14      do
15      :: commands ? open -> printf("Elevator: opened doors.\n");
16      :: commands ? close -> printf("Elevator: closed doors.\n");
17      :: commands ? up -> assert(at!=3);printf("Elevator: moved up one floor.\n");
18      :: commands ? down -> assert(at!=0);printf("Elevator: moved down one floor.\n");
19      od
20  }
```

The **elevator** process then receives the **down** message (at line 18 in above) and before the execution of the assert statement, context switch happens and the **at- -** statement at 53 line of **controller** process gets executed. Now, again when context switches to assert statement of **elevator** process. Since the **at** is decremented to zero the assertion gets violated.

Corrected model: The model can be corrected by the below modifications in the elevator process (**corrected.pml** file)

```
active proctype elevator() {
    do
    :: atomic {commands ? open -> printf("Elevator: opened doors.\n");}
    :: atomic {commands ? close -> printf("Elevator: closed doors.\n");}
    :: atomic {commands ? up -> assert(at!=3);printf("Elevator: moved up one floor.\n");}
    :: atomic {commands ? down -> assert(at!=0);printf("Elevator: moved down one floor.\n");}
    od
}
```

To avoid the context switch the alternatives are placed in atomic blocks in the **elevator** process.

Steps for execution:

1. spin –a corrected.pml
2. gcc –o pan pan.c
3. ./pan

Output:

```
dush123@ubuntu:~/Desktop/se/Q2$ spin -a corrected.pml
dush123@ubuntu:~/Desktop/se/Q2$ gcc -o pan pan.c
dush123@ubuntu:~/Desktop/se/Q2$ ./pan

(Spin Version 6.4.9 -- 17 December 2018)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid end states      +

State-vector 68 byte, depth reached 1355, errors: 0
     4800 states, stored
     7741 states, matched
    12541 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.439       equivalent memory usage for states (stored*(State-vector + overhead))
    0.464       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.925       total actual memory usage


unreached in proctype elevator
        corrected.pml:20, state 18, "-end-"
        (1 of 18 states)
unreached in proctype floor_button_input
        corrected.pml:29, state 8, "-end-"
        (1 of 8 states)
unreached in proctype elevator_button_input
        corrected.pml:38, state 8, "-end-"
        (1 of 8 states)
unreached in proctype controller
        corrected.pml:230, state 200, "-end-"
        (1 of 200 states)

pan: elapsed time 0.01 seconds
```

We can see that there are errors generated by the model.

Q3. Add assertions to the Promela model that trigger if the elevator receives the up or down messages while its doors are open. (Please include the full Promela model in the submitted solution.) Check with Spin whether the assertions are triggered. Explain the obtained result. If the assertions are triggered, provide a corrected model. (Please include a Spin run log with your explanations.)

File name: elevator_Q3.pml

Model:

```
active proctype elevator() {
    do
    :: atomic {commands ? open -> printf("Elevator: opened doors.\n");}
    :: atomic {commands ? close -> printf("Elevator: closed doors.\n");}
    :: atomic {commands ? up -> assert(closed==true);printf("Elevator: moved up one floor.\n");}
    :: atomic {commands ? down -> assert(closed==true);printf("Elevator: moved down one floor.\n");}
    od
}
```

When the elevator receives **up** or **down** message, assertions are written to check if the doors are closed. If the doors are open then the error gets triggered.

Execution and output:

```
dush123@ubuntu:~/Desktop/se/Q3$ spin -a elevator_Q3.pml
dush123@ubuntu:~/Desktop/se/Q3$ gcc -o pan pan.c
dush123@ubuntu:~/Desktop/se/Q3$ ./pan

(Spin Version 6.4.9 -- 17 December 2018)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid end states      +

State-vector 68 byte, depth reached 1757, errors: 0
     9224 states, stored
    11820 states, matched
    21044 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         7 (resolved)

Stats on memory usage (in Megabytes):
    0.844       equivalent memory usage for states (stored*(State-vector + overhead))
    0.759       actual memory usage for states (compression: 89.82%)
                state-vector as stored = 58 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  129.218       total actual memory usage


unreached in proctype elevator
        elevator_Q3.pml:20, state 14, "-end-"
        (1 of 14 states)
unreached in proctype floor_button_input
        elevator_Q3.pml:29, state 8, "-end-"
        (1 of 8 states)
unreached in proctype elevator_button_input
        elevator_Q3.pml:38, state 8, "-end-"
        (1 of 8 states)
unreached in proctype controller
        elevator_Q3.pml:230, state 200, "-end-"
        (1 of 200 states)

pan: elapsed time 0.01 seconds
```

We can see that no errors were generated by the model. So the doors are closed when the elevator receives the **up** or **down** message.

Q4. Mention and explain some properties that should be satisfied by the system/controller that you have designed (that can be expressed using LTL). Check with Spin whether the LTL properties are satisfied by the model (explain the obtained result).

File name: elevator_Q4.pml

Steps for execution:

```
dush123@ubuntu:~/Desktop/se$ cd Q4
dush123@ubuntu:~/Desktop/se/Q4$ spin -a elevator_Q4.pml
ltl withinLimits: [] (((at>=0)) && ((at<=3)))
ltl openedDoorCloses: [] ((! (! (closed))) || (<> (closed)))
ltl outOfBound: <> ((at>=4))
  the model contains 3 never claims: outOfBound, openedDoorCloses, withinLimits
  only one claim is used in a verification run
  choose which one with ./pan -a -N name (defaults to -N withinLimits)
  or use e.g.: spin -search -ltl withinLimits elevator_Q4.pml
dush123@ubuntu:~/Desktop/se/Q4$ gcc -o pan pan.c
```

The following are some of the LTL properties considered:

Property-1: **withinLimits**

The **at** variable denotes the location of the elevator. Since floor can be either 0, 1, 2 or 3, the variable **at** should **always** lie between 0 and 3. This property is defined using LTL syntax in the promela model.

Output:

```
dush123@ubuntu:~/Desktop/se/Q4$ ./pan -a -N withinLimits
pan: ltl formula withinLimits

(Spin Version 6.4.9 -- 17 December 2018)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (withinLimits)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 76 byte, depth reached 3419, errors: 0
     7100 states, stored
     8931 states, matched
    16031 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         3 (resolved)

Stats on memory usage (in Megabytes):
    0.704       equivalent memory usage for states (stored*(State-vector + overhead))
    0.658       actual memory usage for states (compression: 93.41%)
                state-vector as stored = 69 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  129.120       total actual memory usage


unreached in proctype elevator
        elevator_Q4.pml:20, state 12, "-end-"
        (1 of 12 states)
unreached in proctype floor_button_input
        elevator_Q4.pml:29, state 8, "-end-"
        (1 of 8 states)
unreached in proctype elevator_button_input
        elevator_Q4.pml:38, state 8, "-end-"
        (1 of 8 states)
unreached in proctype controller
        elevator_Q4.pml:230, state 200, "-end-"
        (1 of 200 states)
unreached in claim withinLimits
        _spin_nvr.tmp:8, state 10, "-end-"
        (1 of 10 states)

pan: elapsed time 0.01 seconds
```

We can see that the property specified is statisfied by the model.

Property-2: **openedDoorCloses**

If the door opens then it should closed eventually. This is expressed by using always, implication and eventually in LTL.

Output:

```
dush123@ubuntu:~/Desktop/se/Q4$ ./pan -a -N openedDoorCloses
pan: ltl formula openedDoorCloses

(Spin Version 6.4.9 -- 17 December 2018)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (openedDoorCloses)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 76 byte, depth reached 3419, errors: 0
     8680 states, stored (10260 visited)
    16191 states, matched
    26451 transitions (= visited+matched)
        0 atomic steps
hash conflicts:        16 (resolved)

Stats on memory usage (in Megabytes):
    0.861       equivalent memory usage for states (stored*(State-vector + overhead))
    0.755       actual memory usage for states (compression: 87.75%)
                state-vector as stored = 63 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  129.218       total actual memory usage


unreached in proctype elevator
        elevator_Q4.pml:20, state 12, "-end-"
        (1 of 12 states)
unreached in proctype floor_button_input
        elevator_Q4.pml:29, state 8, "-end-"
        (1 of 8 states)
unreached in proctype elevator_button_input
        elevator_Q4.pml:38, state 8, "-end-"
        (1 of 8 states)
unreached in proctype controller
        elevator_Q4.pml:230, state 200, "-end-"
        (1 of 200 states)
unreached in claim openedDoorCloses
        _spin_nvr.tmp:19, state 13, "-end-"
        (1 of 13 states)

pan: elapsed time 0.02 seconds
pan: rate    513000 states/second
```

We can see that property is satisfied by the model.

Property-3: **outOfBound**

Since the **at** variable lies between 0 and 3, an invalid property **outOfBound** is defined as shown above. It says that at some point **at** will be greater than 4.

Output:

```
dush123@ubuntu:~/Desktop/se/Q4$ ./pan -a -N outOfBound
pan: ltl formula outOfBound
pan:1: acceptance cycle (at depth 0)
pan: wrote elevator_Q4.pml.trail

(Spin Version 6.4.9 -- 17 December 2018)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             + (outOfBound)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 76 byte, depth reached 19, errors: 1
        9 states, stored
        0 states, matched
        9 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.001       equivalent memory usage for states (stored*(State-vector + overhead))
    0.267       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage



pan: elapsed time 0 seconds
```

We can see that this property gets violated by the model which is as expected.

Note: Screen shots and output files are zipped and submitted.