Report

Problem 1:

1. The Promela code can be found in problem1.pml
2. The sequence logs for the specified trace are shown in 1B.txt and msc.ps and can be replicated by opening problem1.pml in ispin and clicking on (Re)Run in the Simulate/Replay tab with the random seed set to 123 (default). The description for this trace is as follows:
3. Steps 0-20: System is initializing. In this step, the conditions given in the problem statements with regards to the 4 shuttles and the 2 orders are input into the system.
4. Steps 21-29: The Shuttle Management System sends order 1 to Shuttle 1.
5. Steps 30-33: Shuttle 1 begins processing the offer for the newly received order 1
6. Steps 34-36: The Shuttle Management System sends order 1 to Shuttle 2.
7. Steps 37-50: The Shuttle Management System sends order 1 to Shuttle 3.
8. Steps 51-56: Shuttle 3 begins processing the offer for the newly received order 1
9. Steps 57-59: Shuttle 2 begins processing the offer for the newly received order 1
10. Steps 60-61: The Shuttle Management System sends order 1 to Shuttle 4.
11. Steps 62-70: Shuttle 1 calculates the start destination of order 1 to be 0 stations away from its current position
12. Steps 71-75: Shuttle 4 begins processing the offer for the newly received order 1
13. Steps 80-100: The Shuttle Management System waits for an offer for order 1 from Shuttle 1
14. Steps 101-107: Shuttle 2 calculates the start destination of order 1 to be 0 stations away from its current position
15. Steps 111-115: Shuttle 3 calculates the start destination of order 1 to be 1 station away from its current position
16. Steps 120-127: Shuttle 1 sends an offer to the management system with an acceptance
17. Steps 128-130: Shuttle 4 calculates the start destination of order 1 to be 2 stations away from its current position
18. Steps 131-132: Shuttle 2 sends an offer to the management system with a rejection
19. Steps 135-144: Shuttle 3 sends an offer to the management system with an acceptance
20. Steps 145-152: Shuttle 4 sends an offer to the management system with a rejection
21. Steps 153-156: The Shuttle Management System waits for an offer for order 1 from Shuttle 2
22. Steps 157-168: The Shuttle Management System waits for an offer for order 1 from Shuttle 3
23. Steps 169-178: The Shuttle Management System waits for an offer for order 1 from Shuttle 4
24. Steps 179-188: The Shuttle Management System accepts Shuttle 3’s offer and assigns order 1 to Shuttle 3
25. Steps 255-265: The Shuttle Management System sends order 2 to Shuttle 1.
26. Steps 266-271: The Shuttle Management System sends order 2 to Shuttle 2.
27. Steps 272-275: Shuttle 1 begins processing the offer for the newly received order 2.
28. Steps 276-284: Shuttle 2 begins processing the offer for the newly received order 2.
29. Steps 285-289: The Shuttle Management System sends order 2 to Shuttle 3.
30. Steps 290-293: Shuttle 2 requests access from the Railway Network to travel from station 2 to station 1
31. Steps 294-310: The Shuttle Management System sends order 2 to Shuttle 4.
32. Steps 311-320: Shuttle 4 begins processing the offer for the newly received order 2.
33. Steps 321-328: Shuttle 1 calculates the start destination of order 2 to be 1 station away from its current position
34. Steps 330-339: The Railway Network grants access for Shuttle 2 to travel from station 2 to station 1
35. Steps 340-349: The Shuttle Management System waits for an offer for order 2 from Shuttle 1
36. Steps 350-354: Shuttle 1 sends an offer to the management system with an acceptance
37. Steps 355-355: Shuttle 2 calculates the start destination of order 2 to be 1 station away from its current position
38. Steps 357-364: Shuttle 3 travels from Station 2 to Station 1
39. Steps 365-381: Shuttle 2 sends an offer to the management system with an acceptance
40. Steps 382-392: Shuttle 2 begins processing the offer for the newly received order 2.
41. Steps 393-393: Shuttle 3 loads 4 people at station 1
42. Steps 396-398: Shuttle 4 calculates the start destination of order 2 to be 1 station away from its current position
43. Steps 399-402: The Shuttle Management System waits for an offer for order 2 from Shuttle 2
44. Steps 403-419: Shuttle 4 sends an offer to the management system with an acceptance
45. Steps 420-423: Shuttle 3 begins processing the offer for the newly received order 2.
46. Steps 424-435: The Shuttle Management System waits for an offer for order 2 from Shuttle 3
47. Steps 436-444: Shuttle 3 calculates the start destination of order 2 to be 0 stations away from its current position
48. Steps 445-454: Shuttle 3 sends an offer to the management system with a rejection
49. Steps 455-460: The Shuttle Management System waits for an offer for order 2 from Shuttle 4
50. Steps 461-470: The Shuttle Management System accepts Shuttle 1’s offer and assigns order 2 to Shuttle 1
51. Steps 471-541: Shuttle 1 requests access from the Railway Network to travel from station 1 to station 2
52. Steps 542-565: Shuttle 3 requests access from the Railway Network to travel from station 1 to station 2
53. Steps 566-574: The Railway Network grants access for Shuttle 1 to travel from station 1 to station 2
54. Steps 575-584: Shuttle 1 travels from station 1 to station 2
55. Steps 585-598: The Railway Network rejects access for Shuttle 3 to travel from station 1 to station 2
56. Steps 599-601: Shuttle 1 loads to people at station 2
57. Steps 602-633: Shuttle 1 requests access from the Railway Network to travel from station 2 to station 3
58. Steps 634-640: The Railway Network grants access for Shuttle 3 to travel from station 1 to station 2
59. Steps 641-653: Shuttle 3 travels from station 1 to station 2
60. Steps 654-657: The Railway Network grants access for Shuttle 1 to travel from station 2 to station 3
61. Steps 658-667: Shuttle 1 travels from station 2 to station 3
62. Steps 668-689: Shuttle 1 unloads 2 people at station 3. Order 2 is completed
63. Steps 690-698: Shuttle 3 requests access from the Railway Network to travel from station 2 to station 3
64. Steps 699-707: The Railway Network grants access for Shuttle 3 to travel from station 2 to station 3
65. Steps 708-715: Shuttle 3 travels from station 2 to station 3
66. Steps 716-726: Shuttle 3 unloads 4 people at station 3. Order 1 is completed
67. Steps 726-730: No more pending orders. System terminates
68. The predicate travelling(S) is true if and only if shuttle S is currently on a track (i.e., the shuttle is currently stationed. The predicate noLoad(S) is true if and only if shuttle S has no load (i.e., not carrying any people).

We define the following LTL property: ∀S GFG(!travelling(S) ∧ noLoad(S)). !travelling(S) is represented in spin as predicate p (!travelling). noLoad(S) is represented in spin as predicate q (currentLoad == 0).

We run verification (with non-progress cycles) on the system in problem1-ltl.pml using the LTL property p1 {always eventually always (p && q)}, which results in an acceptance cycle found (at depth 718)). The output (with acceptance cycles) is shown below and can be found in pan1.out and problem1-ltl.pml.trail:

Pid: 13228

Depth= 410 States= 1e+006 Transitions= 1.39e+006 Memory= 928.699 t= 3.93 R= 3e+005

Hence, it is shown that the system always returns into a state that all shuttles are

at stations without load.

<<<<<START OF CYCLE>>>>>

MSC: ~G line 13

719: proc - (p1:1) \_spin\_nvr.tmp:13 (state 15) [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

Never claim moves to line 13 [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

720: proc 1 (Shuttle:1) problem1-ltl.pml:149 (state 88) [else]

MSC: ~G line 9

721: proc - (p1:1) \_spin\_nvr.tmp:9 (state 10) [(1)]

Never claim moves to line 9 [(1)]

722: proc 1 (Shuttle:1) problem1-ltl.pml:149 (state 89) [(1)]

MSC: ~G line 13

723: proc - (p1:1) \_spin\_nvr.tmp:13 (state 15) [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

Never claim moves to line 13 [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

724: proc 1 (Shuttle:1) problem1-ltl.pml:145 (state 84) [shuttleRequests!request.track,request.direction,request.id]

MSC: ~G line 9

725: proc - (p1:1) \_spin\_nvr.tmp:9 (state 10) [(1)]

Never claim moves to line 9 [(1)]

726: proc 6 (RailwayNetwork:1) problem1-ltl.pml:188 (state 1) [shuttleRequests?request.track,request.direction,request.id]

MSC: ~G line 13

727: proc - (p1:1) \_spin\_nvr.tmp:13 (state 15) [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

Never claim moves to line 13 [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

728: proc 6 (RailwayNetwork:1) problem1-ltl.pml:190 (state 2) []

MSC: ~G line 9

729: proc - (p1:1) \_spin\_nvr.tmp:9 (state 10) [(1)]

Never claim moves to line 9 [(1)]

730: proc 6 (RailwayNetwork:1) problem1-ltl.pml:191 (state 3) [((request.direction==1))]

MSC: ~G line 13

731: proc - (p1:1) \_spin\_nvr.tmp:13 (state 15) [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

Never claim moves to line 13 [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

732: proc 6 (RailwayNetwork:1) problem1-ltl.pml:195 (state 8) [else]

MSC: ~G line 9

733: proc - (p1:1) \_spin\_nvr.tmp:9 (state 10) [(1)]

Never claim moves to line 9 [(1)]

734: proc 6 (RailwayNetwork:1) problem1-ltl.pml:195 (state 9) [reply.granted = 0]

Railway Network: rejecting access to track from station 2 to station 3

734: proc 6 (RailwayNetwork:1) problem1-ltl.pml:196 (state 10) [printf('Railway Network: rejecting access to track from station %d to station %d\\n',request.track,((request.track+1)%4))]

MSC: ~G line 13

735: proc - (p1:1) \_spin\_nvr.tmp:13 (state 15) [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

Never claim moves to line 13 [(!(((((!(travelling[0])&&!(travelling[1]))&&!(travelling[2]))&&!(travelling[3]))&&((((currentLoad[0]==0)&&(currentLoad[1]==0))&&(currentLoad[2]==0))&&(currentLoad[3]==0)))))]

736: proc 6 (RailwayNetwork:1) problem1-ltl.pml:206 (state 25) [railwayReplies[request.id]!reply.granted]

MSC: ~G line 9

737: proc - (p1:1) \_spin\_nvr.tmp:9 (state 10) [(1)]

Never claim moves to line 9 [(1)]

738: proc 1 (Shuttle:1) problem1-ltl.pml:146 (state 85) [railwayReplies[id]?reply.granted]

spin: trail ends after 738 steps

1. Can you find any additional problems in the system, by verifying other properties? Please state what property you verified and give a clear explanation of any errors you found.

The system only checks the current load of the shuttle and not future load. Consider the case where an empty shuttle has a pending order (o1) that hasn’t yet been loaded, and another order (o2) to which the shuttle is required to reply with an offer to the shuttle management system. When the shuttle is processing the offer for o2, it will only consider the current empty load and not the future load with o1 included.

Consider the when the combined size of o1 and o2 were to exceed the shuttle’s capacity. If o2’s starting station is located before o1’s ending station, the shuttle would end up loading o1 first and then o2 before it unloads o1. The capacity of the shuttle would be exceeded and there would be an error in the system when o2 is being loaded.

We represent the above property in spin as predicate r (currentLoad <= capacity).

We can run verification on the system (with acceptance cycles) in problem1-additional.pml using the LTL property p2 { always r}, which results in a acceptance cycle found (at depth 718)). The output for this (with acceptance cycles) is unable to be shown in pan1-additional.out and problem1-additional.pml.trail.

Hence, it is shown that the capacity of the shuttle can be exceeded by the current load.

<<<<<START OF CYCLE>>>>>

720: proc 1 (Shuttle:1) problem1-additional.pml:149 (state 88) [else]

722: proc 1 (Shuttle:1) problem1-additional.pml:149 (state 89) [(1)]

724: proc 1 (Shuttle:1) problem1-additional.pml:145 (state 84) [shuttleRequests!request.track,request.direction,request.id]

726: proc 6 (RailwayNetwork:1) problem1-additional.pml:188 (state 1) [shuttleRequests?request.track,request.direction,request.id]

728: proc 6 (RailwayNetwork:1) problem1-additional.pml:190 (state 2) []

730: proc 6 (RailwayNetwork:1) problem1-additional.pml:191 (state 3) [((request.direction==1))]

732: proc 6 (RailwayNetwork:1) problem1-additional.pml:195 (state 8) [else]

734: proc 6 (RailwayNetwork:1) problem1-additional.pml:195 (state 9) [reply.granted = 0]

Railway Network: rejecting access to track from station 2 to station 3

734: proc 6 (RailwayNetwork:1) problem1-additional.pml:196 (state 10) [printf('Railway Network: rejecting access to track from station %d to station %d\\n',request.track,((request.track+1)%4))]

736: proc 6 (RailwayNetwork:1) problem1-additional.pml:206 (state 25) [railwayReplies[request.id]!reply.granted]

738: proc 1 (Shuttle:1) problem1-additional.pml:146 (state 85) [railwayReplies[id]?reply.granted]

spin: trail ends after 738 steps

Problem 2:

1. The Promela code can be found in problem2.pml
2. We run verification (with safety) on the system in problem2.pml, which results in an invalid end state (deadlock) at depth 263. The output for the deadlock can be found in pan2.out and problem2.pml.trail.

245: proc 4 (Client:1) problem2.pml:76 (state 55) [(((nempty(wcpRequestClient[id])&&(currStatus==8))&&connected))]

Client 4: manual update request received

246: proc 4 (Client:1) problem2.pml:77 (state 56) [printf('Client %d: manual update request received\\n',(id+1))]

247: proc 3 (Client:1) problem2.pml:109 (state 96) [cmStatus[id]?currStatus]

248: proc 3 (Client:1) problem2.pml:113 (state 103) [cmCommand[id]?currCommand]

249: proc 3 (Client:1) problem2.pml:117 (state 107) [else]

250: proc 3 (Client:1) problem2.pml:117 (state 108) [(1)]

251: proc 3 (Client:1) problem2.pml:76 (state 55) [(((nempty(wcpRequestClient[id])&&(currStatus==8))&&connected))]

Client 3: manual update request received

252: proc 3 (Client:1) problem2.pml:77 (state 56) [printf('Client %d: manual update request received\\n',(id+1))]

253: proc 2 (Client:1) problem2.pml:109 (state 96) [cmStatus[id]?currStatus]

254: proc 2 (Client:1) problem2.pml:113 (state 103) [cmCommand[id]?currCommand]

255: proc 2 (Client:1) problem2.pml:117 (state 107) [else]

256: proc 2 (Client:1) problem2.pml:117 (state 108) [(1)]

257: proc 2 (Client:1) problem2.pml:76 (state 55) [(((nempty(wcpRequestClient[id])&&(currStatus==8))&&connected))]

Client 2: manual update request received

258: proc 2 (Client:1) problem2.pml:77 (state 56) [printf('Client %d: manual update request received\\n',(id+1))]

259: proc 1 (Client:1) problem2.pml:109 (state 96) [cmStatus[id]?currStatus]

260: proc 1 (Client:1) problem2.pml:113 (state 103) [cmCommand[id]?currCommand]

261: proc 1 (Client:1) problem2.pml:117 (state 107) [else]

262: proc 1 (Client:1) problem2.pml:117 (state 108) [(1)]

263: proc 1 (Client:1) problem2.pml:76 (state 55) [(((nempty(wcpRequestClient[id])&&(currStatus==8))&&connected))]

Client 1: manual update request received

264: proc 1 (Client:1) problem2.pml:77 (state 56) [printf('Client %d: manual update request received\\n',(id+1))]

spin: trail ends after 264 steps

1. Can you find any additional problems in the protocol? Please state what property you verified and clear explanation of any errors you found.