



## **Project Report Part-2:**

**Project Title: Restaurant Management System**

**Course Title: Software Quality Assurance**

**Course Code: CSE435**

**Semester: SUMMER'21**

**SUBMITTED TO,**

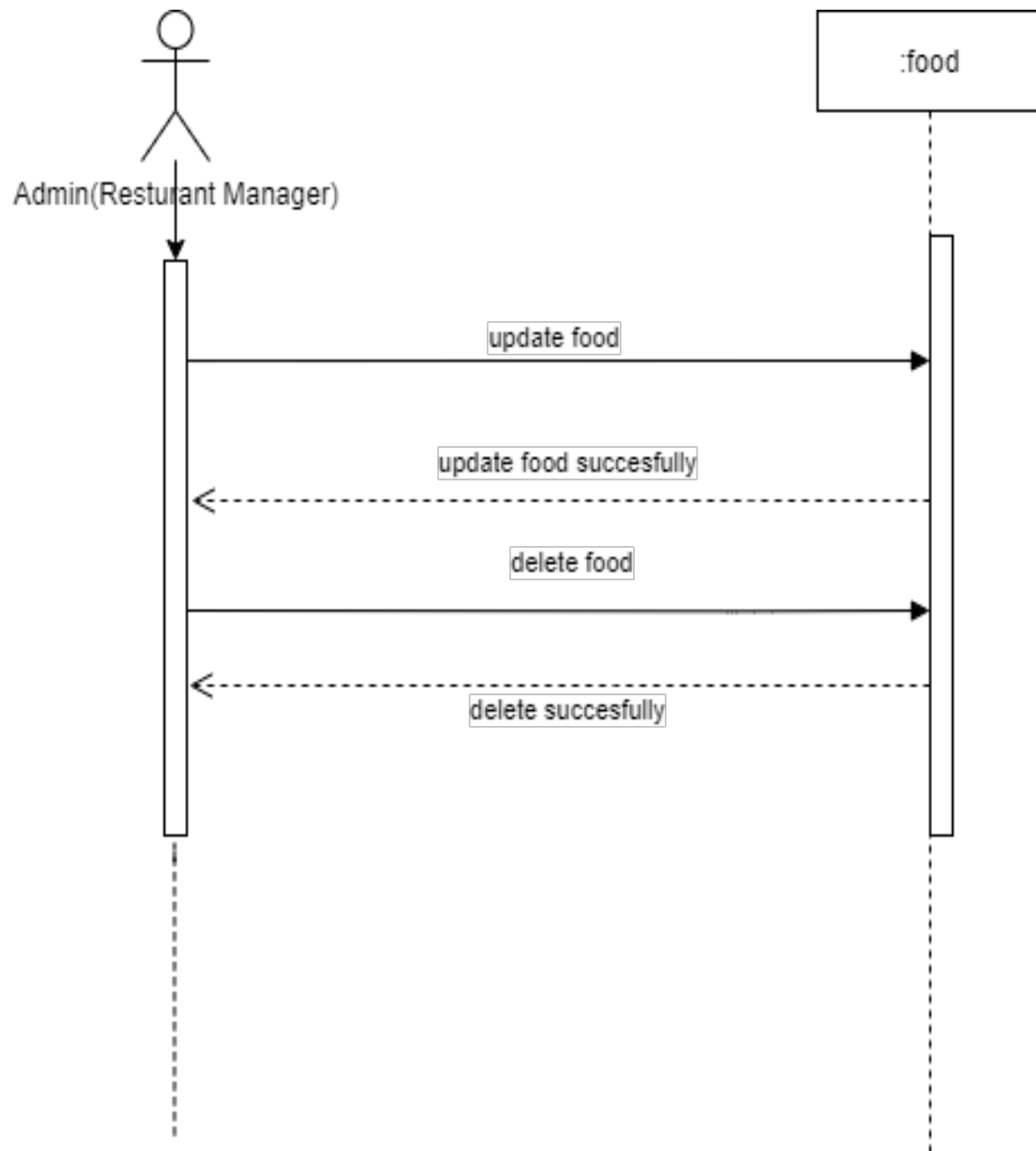
**Dr. Shamim H Ripon,  
Professor, Department of Computer Science & Engineering,  
East West University, Dhaka, Bangladesh.**

**SUBMITTED BY:**

- ❖ **Maria Mehjabin Shenjuti (2018-1-60-244)-Sec02**
- ❖ **Rafina Afreen (2018-1-60-119)-Sec03**

## For Update & Delete:

Sequence Diagram:

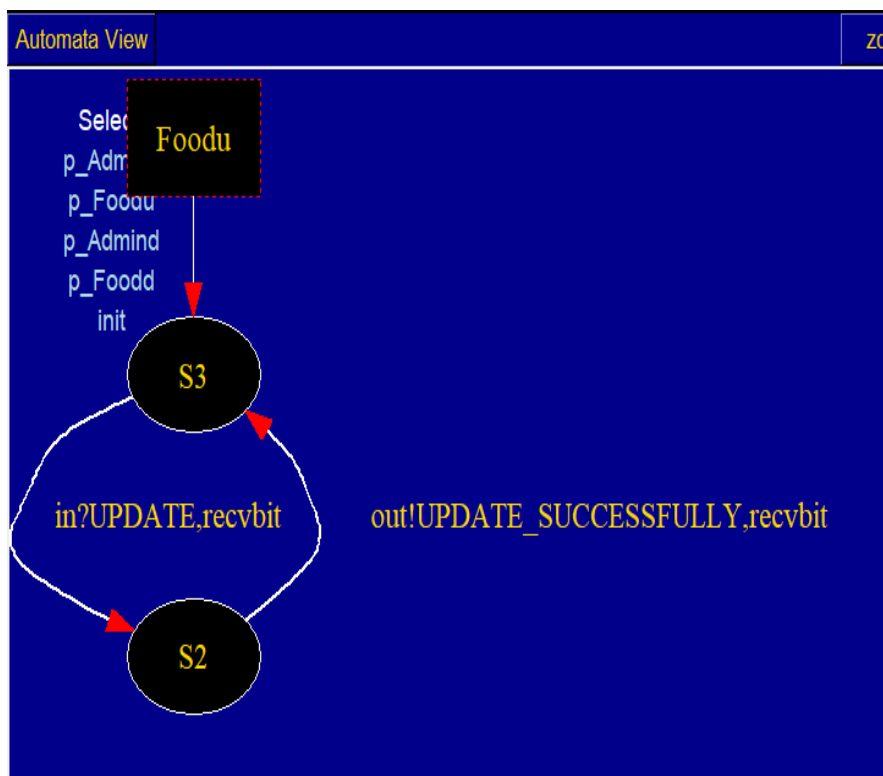
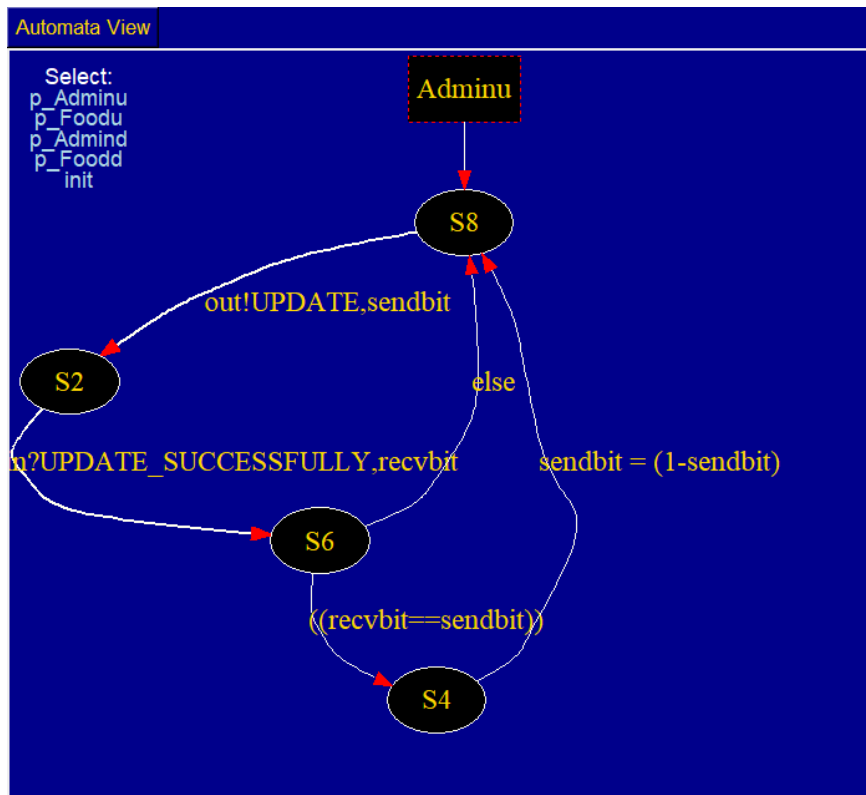


## Promela Code:

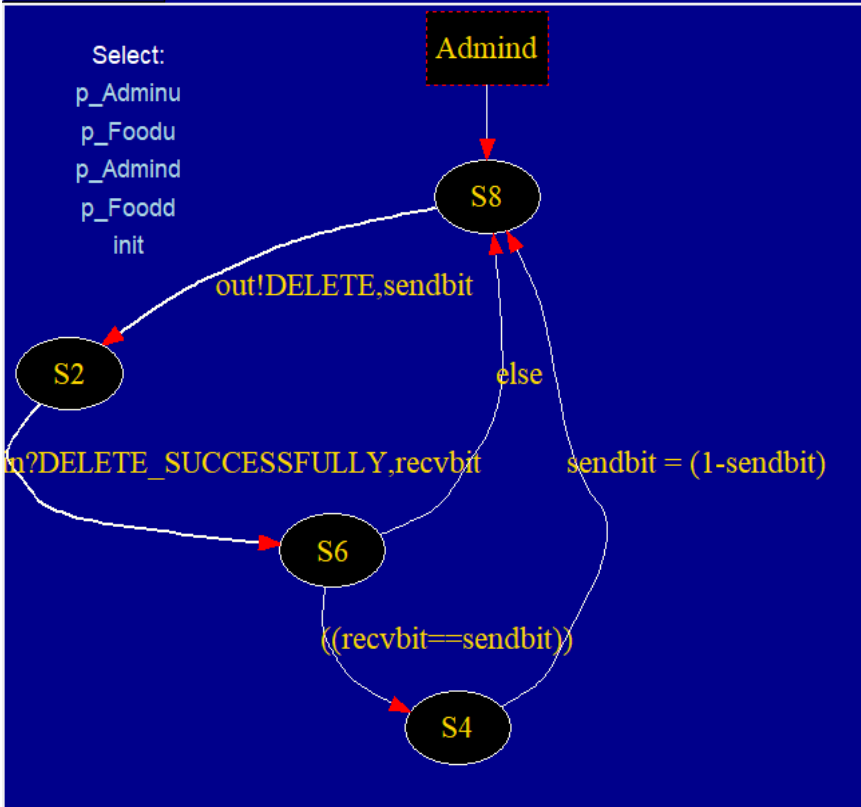
```
Edit/View  Simulate / Replay  Verification  Swarm Run  <Help>  Save Session  Restore Session  <Quit>
Open...  ReOpen  Save  Save As...  Syntax Check  Redundancy Check  Symbol Table  Find: 

1  mtype {UPDATE, UPDATE_SUCCESSFULLY}
2  mtype {DELETE, DELETE_SUCCESSFULLY}
3  chan toAu = [2] of {mtype,bit};
4  chan toFu = [2] of {mtype,bit};
5  chan toAd = [2] of {mtype,bit};
6  chan toFd = [2] of {mtype,bit};
7  proctype Adminu(chan in, out)
8  {
9      bit sendbit,recvbit;
10     do
11         :: out !UPDATE, sendbit ->
12             in ? UPDATE_SUCCESSFULLY,recvbit;
13             if
14                 :: recvbit == sendbit ->
15                     sendbit = 1-sendbit
16             :: else
17                 fi
18         od
19     }
20
21     proctype Foodu(chan in, out)
22     {
23         bit recvbit
24         do
25             :: in ? UPDATE(recvbit) ->
26                 out ! UPDATE_SUCCESSFULLY(recvbit);
27         od
28     }
29
30     proctype Admind(chan in, out)
31     {
32         bit sendbit,recvbit;
33         do
34             :: out !DELETE, sendbit ->
35                 in ? DELETE_SUCCESSFULLY,recvbit;
36                 if
37                     :: recvbit == sendbit ->
38                         sendbit = 1-sendbit
39                 :: else
40                     fi
41             od
42         }
43
44         proctype Foodd(chan in, out)
45         {
46             bit recvbit
47             do
48                 :: in ? DELETE(recvbit) ->
49                     out ! DELETE_SUCCESSFULLY(recvbit);
50             od
51         }
52         init
53         {
54             run Adminu(toAu, toFu);
55             run Foodu(toFu, toAu);
56             run Admind(toAd, toFd);
57             run Foodd(toFd, toAd);
58         }
59     }
```

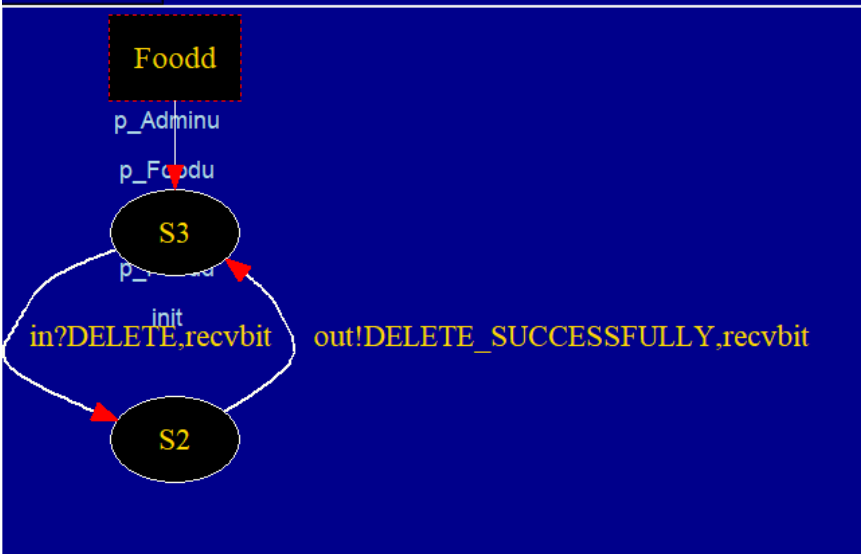
Automata of each process:



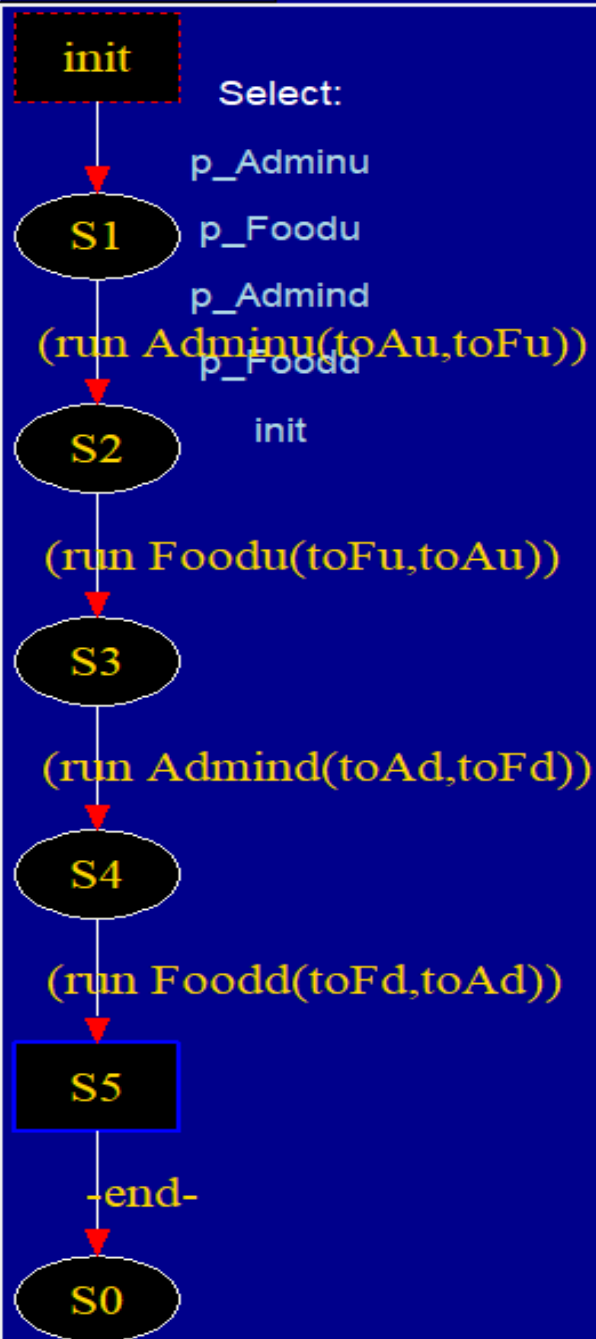
# Automata View



# Automata View



# Automata View



## Simulation run:

Spin Version 6.5.1 -- 31 July 2020 : iSpin Version 1.1.1 -- 23 February 2014

Edit/View Simulate / Replay Verification Swarm Run <Help> Save Session Restore Session <Quit>

Mode: Random, with seed: 123  
 Interactive (for resolution of all nondeterminism)  
 Guided, with trail: UpdateDelete.pml.trail  
 Initial steps skipped: 0  
 maximum number of steps: 10000  
 Track Data Values (this can be slow)

A Full Channel: blocks new messages  
 loses new messages  
 MSC+stmnt  
 MSC max text width: 20  
 MSC update delay: 25

Output Filtering (reg. exps.): process ids:  
 queue ids:  
 var names:  
 tracked variable:  
 track scalings:

(Re)Run  
 Stop  
 Rewind  
 Step Forward  
 Step Backward

Background command executed:  
 spin -p -s -r -X -v -n123 -l -g -u10000 UpdateDelete.pml

Save in: msc.ps

```

1  mtype (UPDATE, UPDATE_SUCCESSFULLY)
2  mtype (DELETE, DELETE_SUCCESSFULLY)
3  chan toAu = [2] of {mtype.bit};
4  chan toFu = [2] of {mtype.bit};
5  chan toAd = [2] of {mtype.bit};
6  chan toFd = [2] of {mtype.bit};
7  proctype Adminu(chan in, out)
8  {
9      bit sendbit,rcvbit;
10     do
11         :: out !UPDATE, sendbit ->
12             in ? UPDATE_SUCCESSFULLY,rcvbit;
13         if
14             :: rcvbit == sendbit ->
15                 sendbit = 1-sendbit
16         :: else

```

```

108  3!DELETE, SUCCESSFULLY...
109  3?DELETE, SUCCESSFULLY...
111  1!UPDATE, SUCCESSFULLY...
116  1?UPDATE, SUCCESSFULLY...
122  2!UPDATE, 0
123  2?UPDATE, 0
124  1!UPDATE, SUCCESSFULLY...
125  1?UPDATE, SUCCESSFULLY...
128  4!DELETE, 1
132  2!UPDATE, 1
133  2?UPDATE, 1

```

[variable values, step 127]

```

Adminu(3):rcvbit = 0
Adminu(3):sendbit = 1
Adminu(1):rcvbit = 0
Adminu(1):sendbit = 1
Foodu(4):rcvbit = 0
Foodu(2):rcvbit = 0

```

[queues, step 132]

```

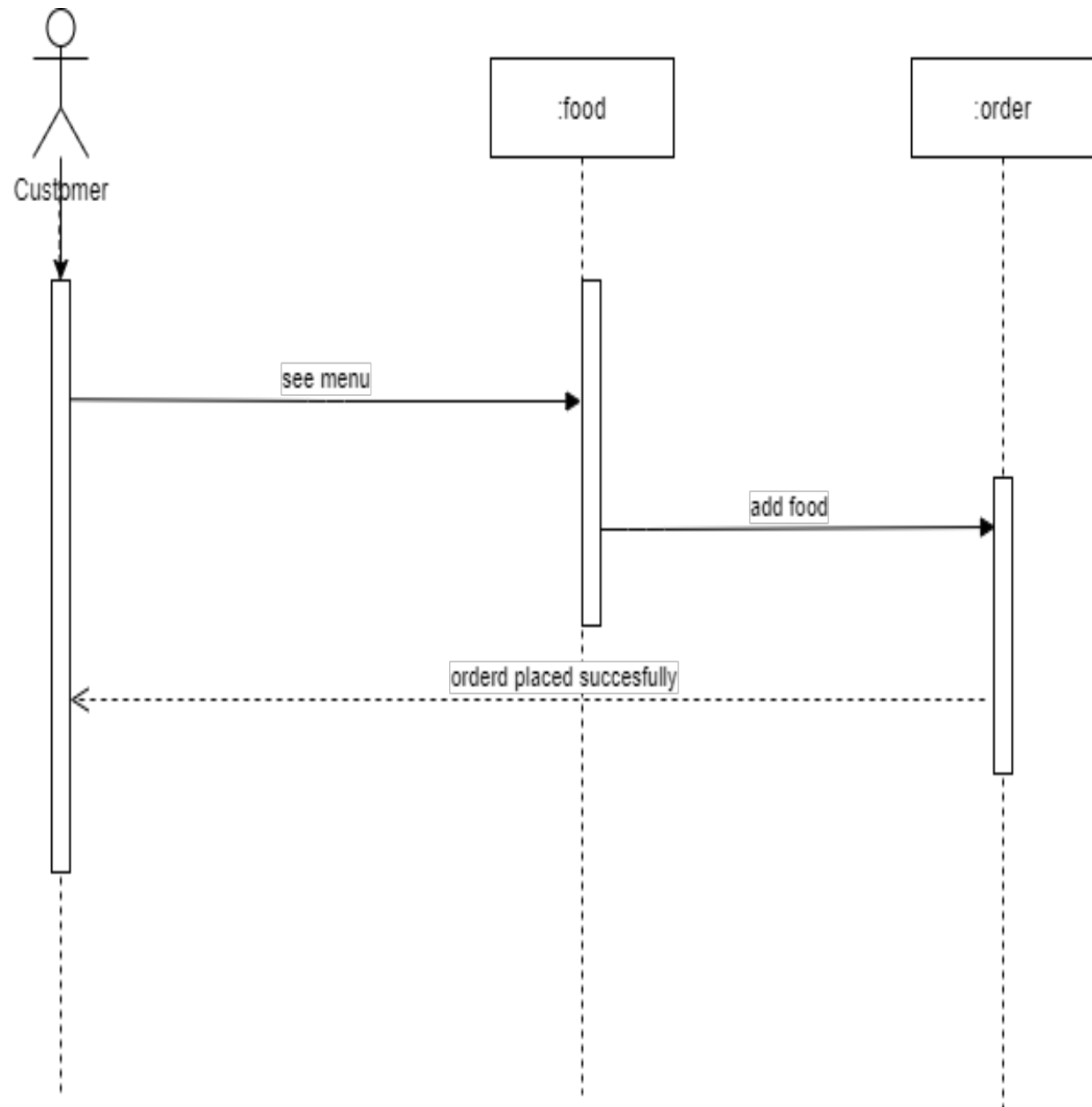
q 1 :: Adminu(1):in:
q 2 :: Adminu(1):out: [UPDATE,1]
q 3 :: toAd:
q 4 :: Adminu(3):out: [DELETE,1]

```

107: proc 2 (Foodu:1) UpdateDelete.pml:25 (state 1) [in?UPDATE,rcvbit]  
 108: proc 4 (Foodu:1) UpdateDelete.pml:48 (state 2) [out!DELETE\_SUCCESSFULLY,rcvbit]  
 109: proc 3 (Adminu:1) UpdateDelete.pml:34 (state 2) [in?DELETE\_SUCCESSFULLY,rcvbit]  
 110: proc 3 (Adminu:1) UpdateDelete.pml:36 (state 3) [!(rcvbit==sendbit)]  
 111: proc 2 (Foodu:1) UpdateDelete.pml:26 (state 2) [out!UPDATE\_SUCCESSFULLY,rcvbit]  
 112: proc 3 (Adminu:1) UpdateDelete.pml:37 (state 4) [sendbit = (1-sendbit)]  
 116: proc 1 (Adminu:1) UpdateDelete.pml:12 (state 2) [in?UPDATE\_SUCCESSFULLY,rcvbit]  
 117: proc 1 (Adminu:1) UpdateDelete.pml:14 (state 3) [!(rcvbit==sendbit)]  
 119: proc 1 (Adminu:1) UpdateDelete.pml:15 (state 4) [sendbit = (1-sendbit)]  
 122: proc 1 (Adminu:1) UpdateDelete.pml:11 (state 1) [out!UPDATE,sendbit]  
 123: proc 2 (Foodu:1) UpdateDelete.pml:25 (state 1) [in?UPDATE,rcvbit]  
 124: proc 2 (Foodu:1) UpdateDelete.pml:26 (state 2) [out!UPDATE\_SUCCESSFULLY,rcvbit]  
 125: proc 1 (Adminu:1) UpdateDelete.pml:12 (state 2) [in?UPDATE\_SUCCESSFULLY,rcvbit]  
 126: proc 1 (Adminu:1) UpdateDelete.pml:14 (state 3) [!(rcvbit==sendbit)]  
 127: proc 1 (Adminu:1) UpdateDelete.pml:15 (state 4) [sendbit = (1-sendbit)]  
 128: proc 3 (Adminu:1) UpdateDelete.pml:33 (state 1) [out!DELETE,sendbit]

## For Order Place:

### Sequence Diagram:



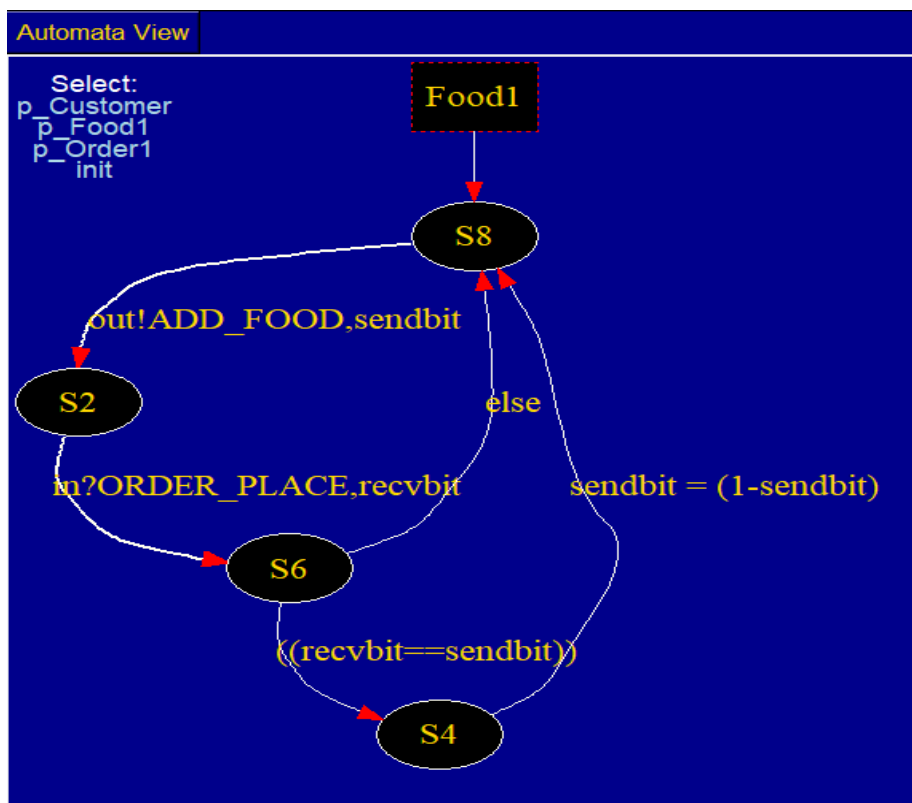
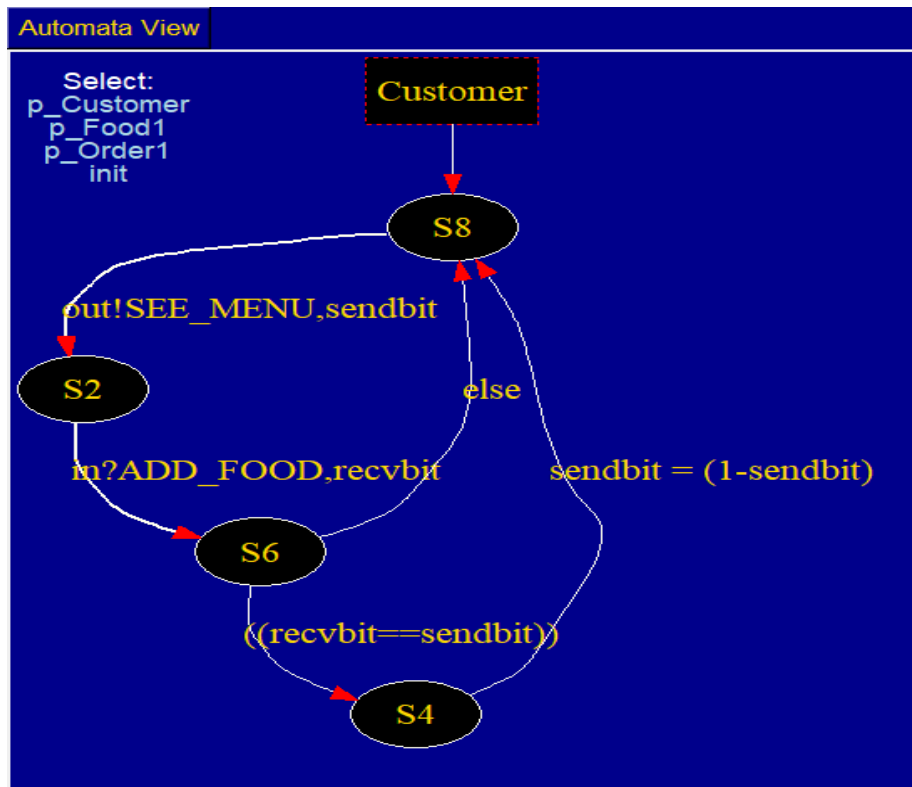


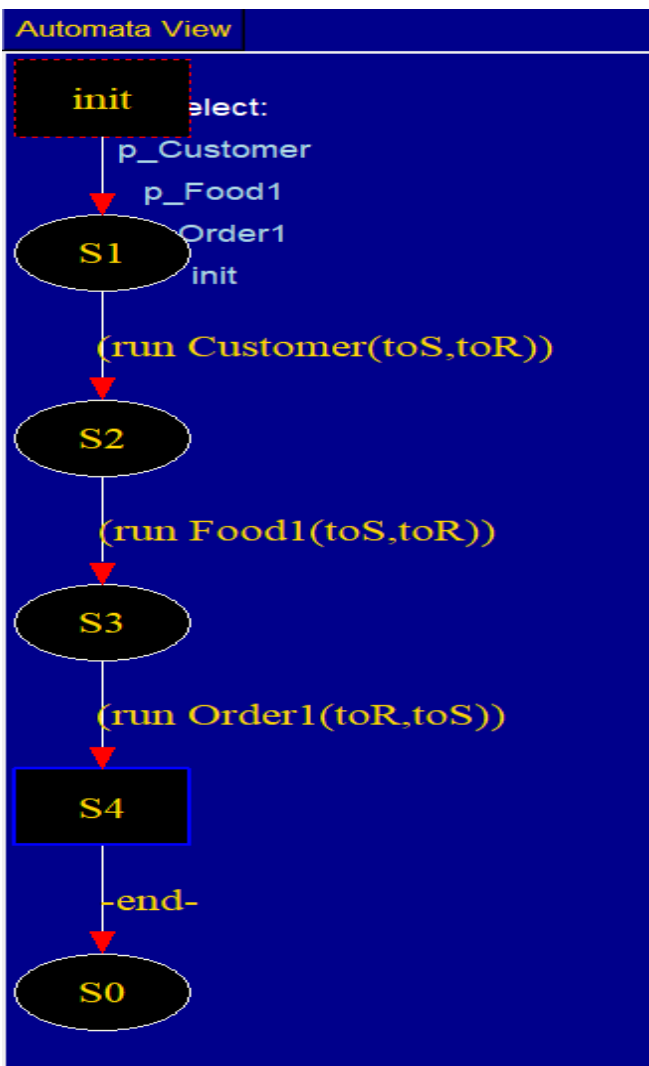
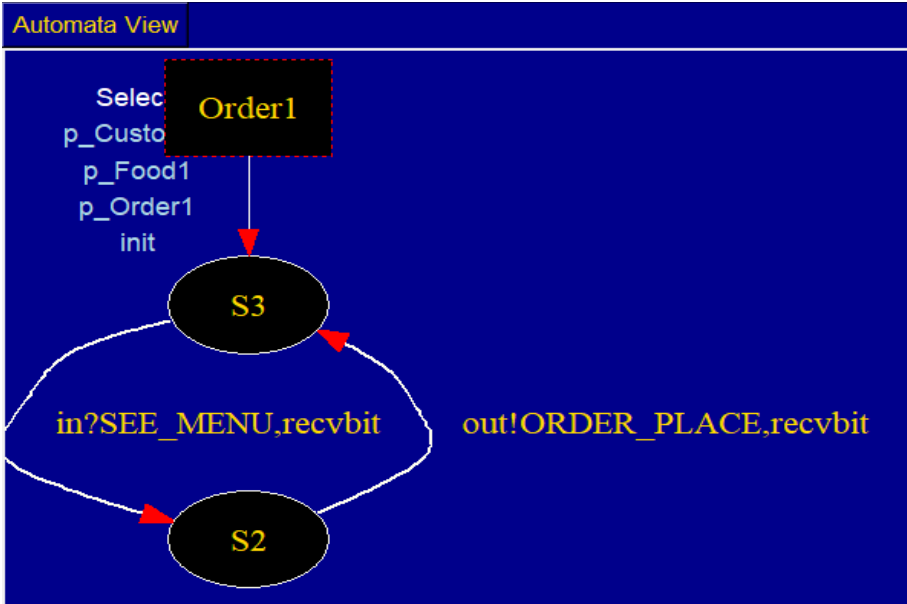
## Promela Code:

Edit/View	Simulate / Replay	Verification	Swarm Run	<Help>	Save Session	Restore Session	<Quit>
Open...	ReOpen	Save	Save As...	Syntax Check	Redundancy Check	Symbol Table	Find: <input type="text"/>

```
1  mtype {SEE_MENU,ADD_FOOD,ORDER_PLACE}
2
3
4  chan toS = [3] of {mtype,bit,bit};
5  chan toR = [3] of {mtype,bit,bit};
6
7  proctype Customer(chan in, out)
8  {
9      bit sendbit,recvbit;
10     do
11         :: out !SEE_MENU, sendbit ->
12             in ? ADD_FOOD,recvbit;
13             if
14                 :: recvbit == sendbit ->
15                     sendbit = 1-sendbit
16                 :: else
17                     fi
18             od
19     }
20
21     proctype Food1(chan in, out)
22     {
23         bit sendbit,recvbit;
24         do
25             :: out !ADD_FOOD, sendbit ->
26                 in ? ORDER_PLACE,recvbit;
27                 if
28                     :: recvbit == sendbit ->
29                         sendbit = 1-sendbit
30                     :: else
31                         fi
32                 od
33     }
34
35     proctype Order1(chan in, out)
36     {
37         bit recvbit
38         do
39             :: in ? SEE_MENU(recvbit) ->
40                 out ! ORDER_PLACE(recvbit);
41         od
42     }
43     init
44     {
45         run Customer(toS, toR);
46         run Food1(toS, toR);
47         run Order1(toR, toS);
48     }
```

Automata of each process:





## Simulation run:

Spin Version 6.5.1 31 July 2020 - iSpin Version 1.1.1 23 February 2014

Edit/View Simulate / Replay Verification Swarm Run <Help> Save Session Restore Session <Quit>

Mode A Full Channel Output Filtering (reg. exps.) (Re)Run

☒ Random, with seed: 123 ☒ blocks new messages process ids:   
☐ Interactive (for resolution of all nondeterminism) ☐ loses new messages queue ids:   
☐ Guided, with trail: OrderPlace2.pml.trail browse ☐ MSC+stmt var names:   
initial steps skipped: 0 MSC max text width: 20 Step Forward  
maximum number of steps: 10000 MSC update delay: 25 tracked variable:   
☒ Track Data Values (this can be slow) track scaling:  Step Backward

Background command executed:  
spin -p -s -r -X -v -n123 -l -g -u10000 OrderPlace.pml

Save in: msc.ps

```
1 mtype (SEE_MENU,ADD_FOOD,ORDER_PLACE)
2
3
4 chan toS = [3] of (mtype,bit,bit);
5 chan toR = [3] of (mtype,bit,bit);
6
7 proctype Customer(chan in, out)
8 {
9     bit sendbit,recvbit;
10    do
11        :: out!SEE_MENU,sendbit->
12            in?ADD_FOOD,recvbit;
13        if
14            :: recvbit == sendbit->
15                sendbit = 1-sendbit
16        :: else
```

[variable values, step 238]

Food1(1):recvbit = 1  
Food1(1):sendbit = 0  
Food2(3):recvbit = 1  
Food2(3):sendbit = 0  
Order1(2):recvbit = 0  
Order2(4):recvbit = 1

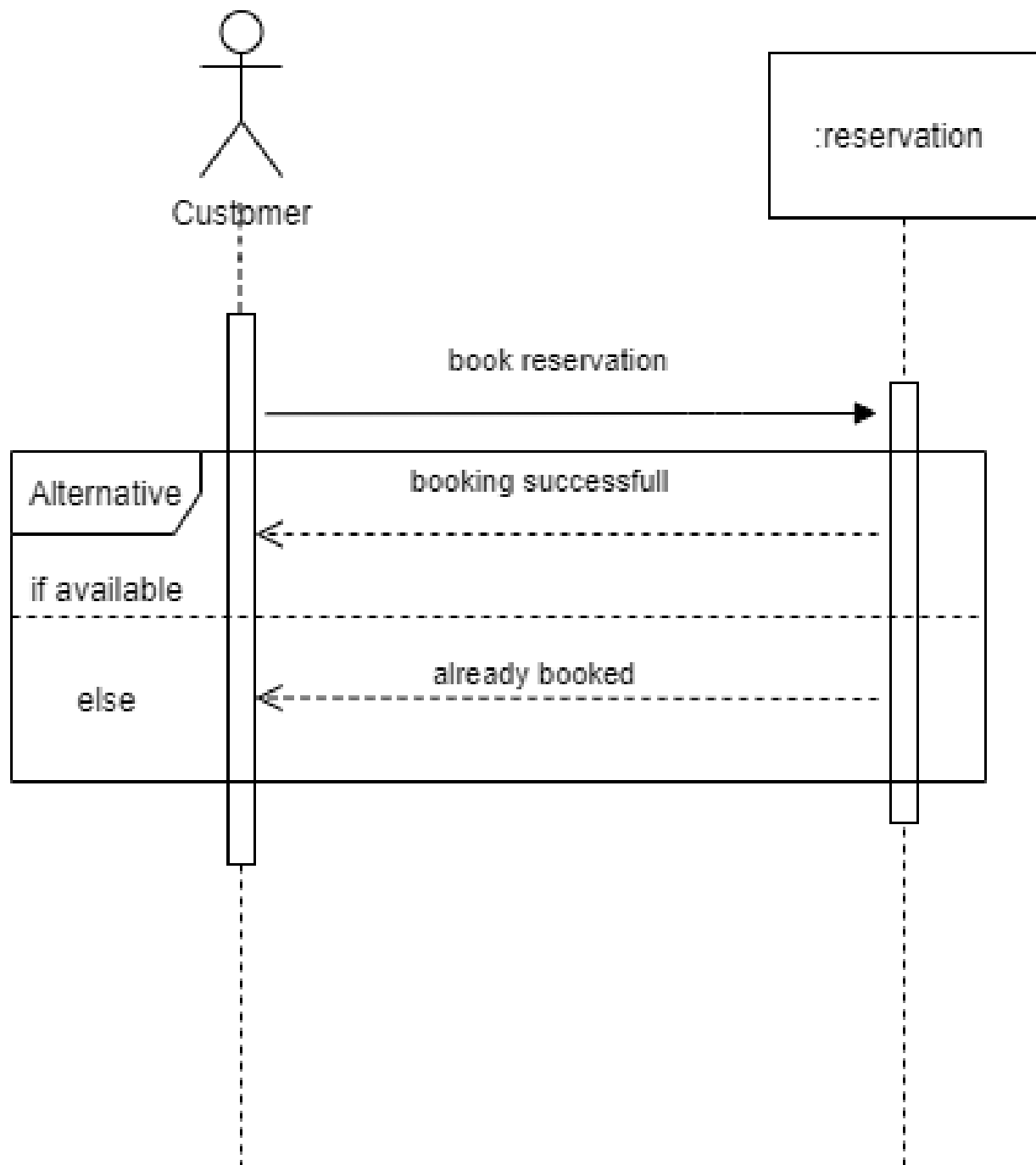
216: proc 1 (Food1:1) OrderPlace2.pml:14 (state 2) [in?ADD\_FOOD1,recvbit]  
217: proc 4 (Order2:1) OrderPlace2.pml:49 (state 1) [in?ADD\_FOOD2,recvbit]  
219: proc 1 (Food1:1) OrderPlace2.pml:16 (state 3) [(recvbit==sendbit)]  
220: proc 4 (Order2:1) OrderPlace2.pml:50 (state 2) [out!ORDER\_PLACE,recvbit]  
222: proc 3 (Food2:1) OrderPlace2.pml:36 (state 2) [in?ORDER\_PLACE,recvbit]  
223: proc 3 (Food2:1) OrderPlace2.pml:38 (state 3) [(recvbit==sendbit)]  
224: proc 1 (Food1:1) OrderPlace2.pml:17 (state 4) [sendbit = (1-sendbit)]  
227: proc 3 (Food2:1) OrderPlace2.pml:39 (state 4) [sendbit = (1-sendbit)]  
230: proc 3 (Food2:1) OrderPlace2.pml:35 (state 1) [out!ADD\_FOOD2,sendbit]  
231: proc 4 (Order2:1) OrderPlace2.pml:49 (state 1) [in?ADD\_FOOD2,recvbit]  
232: proc 4 (Order2:1) OrderPlace2.pml:50 (state 2) [out!ORDER\_PLACE,recvbit]  
234: proc 3 (Food2:1) OrderPlace2.pml:36 (state 2) [in?ORDER\_PLACE,recvbit]  
235: proc 3 (Food2:1) OrderPlace2.pml:38 (state 3) [(recvbit==sendbit)]  
236: proc 3 (Food2:1) OrderPlace2.pml:39 (state 4) [sendbit = (1-sendbit)]  
237: proc 1 (Food1:1) OrderPlace2.pml:13 (state 1) [out!SEE\_MENU,sendbit]  
238: proc 2 (Order1:1) OrderPlace2.pml:27 (state 1) [in?SEE\_MENU,recvbit]

[queues, step 238]

q 1 :: (Food1(1):in):  
q 2 :: (Order1(2):in):  
q 3 :: (Food2(3):in):  
q 4 :: (tn02):

## For Reservation:

### Sequence Diagram:

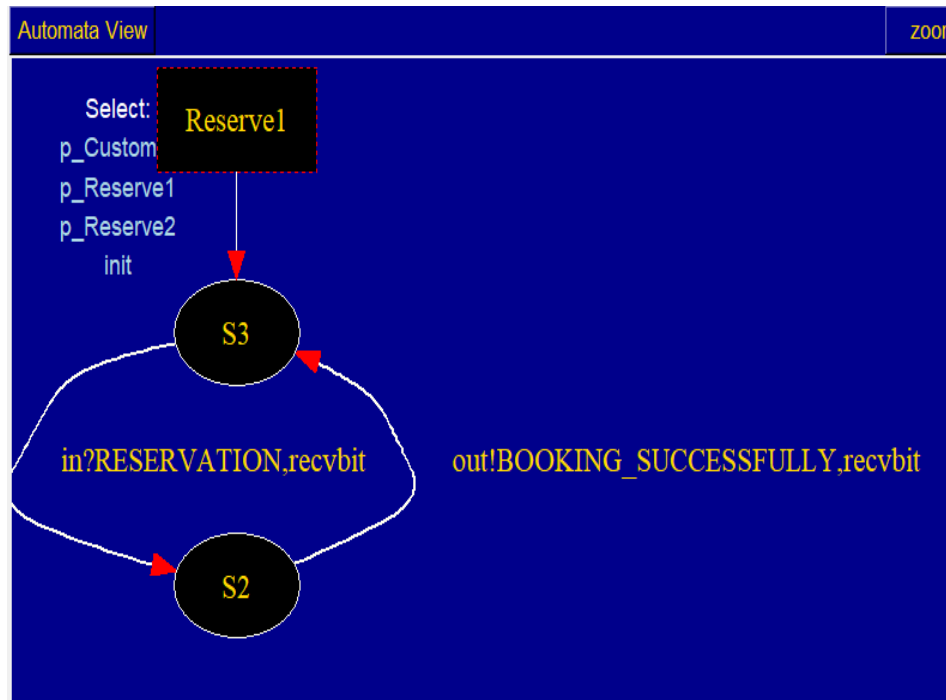
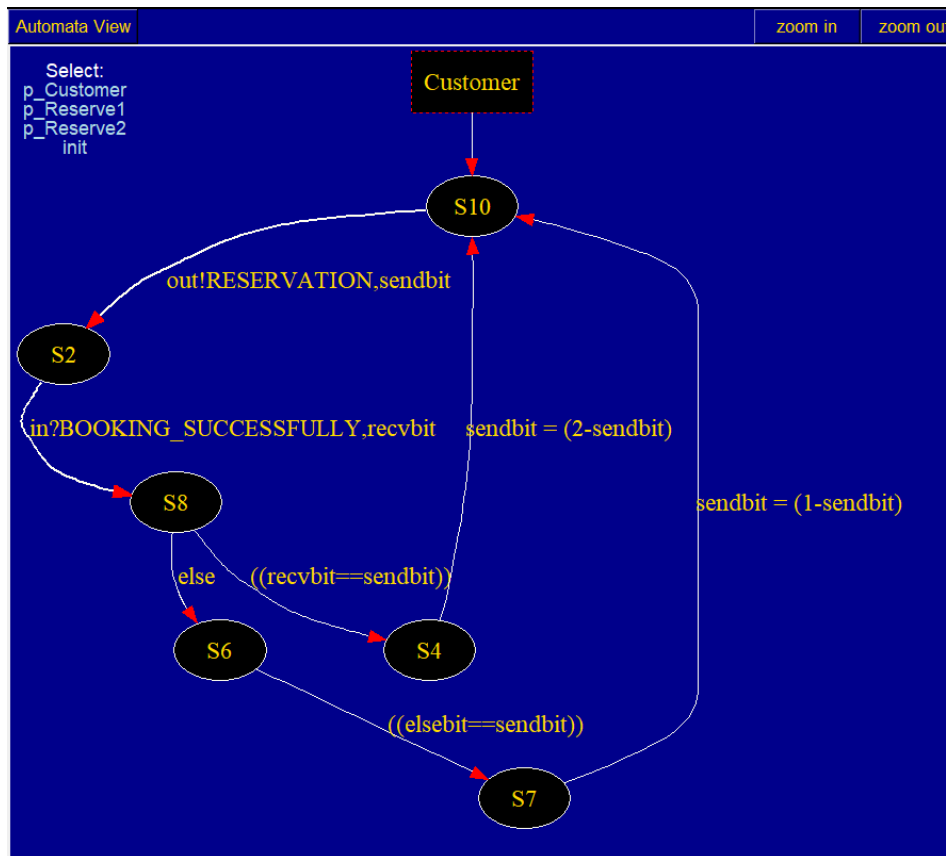


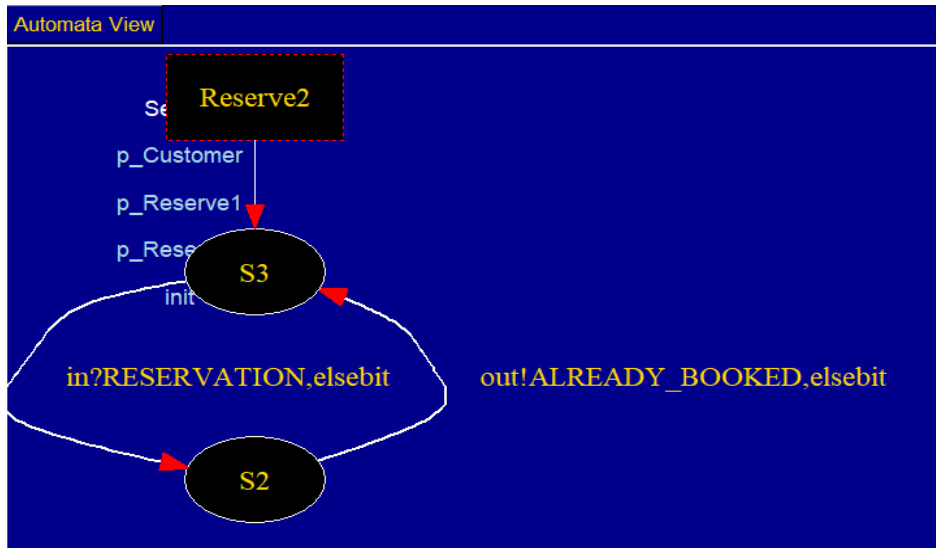
## Promela Code:

Edit/View	Simulate / Replay	Verification	Swarm Run	<Help>	Save Session	Restore Session	<Quit>
Open...	ReOpen	Save	Save As...	Syntax Check	Redundancy Check	Symbol Table	Find: <input type="text"/>

```
1  mtype {RESERVATION, BOOKING_SUCCESSFULLY, ALREADY_BOOKED}
2  chan toS = [3] of {mtype,bit,bit};
3  chan toR = [3] of {mtype,bit,bit};
4
5  proctype Customer(chan in, out)
6  {
7      bit sendbit,rcvbit,elsebit;
8      do
9          :: out !RESERVATION, sendbit ->
10             in ? BOOKING_SUCCESSFULLY,rcvbit;
11             if
12                 :: rcvbit == sendbit ->
13                     sendbit = 2-sendbit
14                 :: else -> elsebit == sendbit ->
15                     sendbit = 1-sendbit
16             fi
17         od
18     }
19     proctype Reserve1(chan in, out)
20     {
21         bit rcvbit
22         do
23             :: in ? RESERVATION(rcvbit) ->
24                 out ! BOOKING_SUCCESSFULLY(rcvbit);
25         od
26     }
27     proctype Reserve2(chan in, out)
28     {
29         bit elsebit
30         do
31             :: in ? RESERVATION(elsebit) ->
32                 out ! ALREADY_BOOKED(elsebit);
33         od
34     }
35     init
36     {
37         run Customer(toS, toR);
38         run Reserve1(toR, toS);
39         run Reserve2(toR, toS);
40     }
41
```

### Automata of each process:







## Simulation run:

Spin Version 6.5.1 -- 31 July 2020 : iSpin Version 1.1.1 -- 23 February 2014

Edit/View Simulate / Replay Verification Swarm Run <Help> Save Session Restore Session <Quit>

Mode: Random, with seed: 123  
Interactive (for resolution of all nondeterminism)  
Guided, with trail: Reservation.pml.trail  
Initial steps skipped: 0  
Maximum number of steps: 10000  
Track Data Values (this can be slow)

A Full Channel  
blocks new messages  
loses new messages  
MSC: stmt  
MSC max text width: 20  
MSC update delay: 25  
track scaling:

Output Filtering (reg. exps):  
process ids:  
queue ids:  
var names:  
tracked variable:  
track scaling:

(Re)Run  
Stop  
Rewind  
Step Forward  
Step Backward

Background command executed:  
spin -p -s -r -X -v -n 123 -l -g -u 10000 Reservation.pml

Save in: msc.ps

```
1 mtype (RESERVATION, BOOKING_SUCCESSFULLY, ALREADY_BOOKED)
2 chan toS = [3] of (mtype.bit.bit);
3 chan toR = [3] of (mtype.bit.bit);
4
5 proctype Customer(chan in, out)
6 {
7     bit sendbit,recvbit,elsebit;
8     do
9         :: out IRESERVATION, sendbit ->
10             In ? BOOKING_SUCCESSFULLY,recvbit;
11             if
12                 :: recvbit == sendbit ->
13                     sendbit = 2-sendbit
14                 :: else -> elsebit == sendbit ->
15                     sendbit = 1-sendbit
16             fi
17     od
18 }
```

2 IRESERVATION,0,0  
3 IRESERVATION,0,0  
4 IRESERVATION,0,0  
5 IRESERVATION,0,0  
6 IRESERVATION,0,0  
7 IRESERVATION,0,0  
8 IRESERVATION,0,0  
9 IRESERVATION,0,0  
10 IRESERVATION,0,0  
11 IRESERVATION,0,0  
12 IRESERVATION,0,0  
13 IRESERVATION,0,0  
14 IRESERVATION,0,0  
15 IRESERVATION,0,0  
16 IRESERVATION,0,0  
17 IRESERVATION,0,0  
18 IRESERVATION,0,0  
19 IRESERVATION,0,0  
20 IRESERVATION,0,0  
21 IRESERVATION,0,0  
22 IRESERVATION,0,0  
23 IRESERVATION,0,0  
24 IRESERVATION,0,0

[variable values, step 25]

Customer(1):elsebit = 0  
Customer(1):recvbit = 0  
Customer(1):sendbit = 0  
Reserve1(2):recvbit = 0  
Reserve2(3):elsebit = 0

16: proc 1 (Customer:1) Reservation.pml:12 (state 3) [!(recvbit==sendbit)]  
spin: Reservation.pml:13: Error: value (2->0 (1)) truncated in assignment  
19: proc 1 (Customer:1) Reservation.pml:13 (state 4) [sendbit = (2-sendbit)]  
22: warning: missing params in send  
22: proc 1 (Customer:1) Reservation.pml:9 (state 1) [out IRESERVATION, sendbit]  
23: warning: missing params in next recv  
23: proc 3 (Reserve2:1) Reservation.pml:31 (state 1) [in ? RESERVATION, elsebit]  
24: warning: missing params in send  
24: proc 3 (Reserve2:1) Reservation.pml:32 (state 2) [out !ALREADY\_BOOKED, elsebit]  
timeout  
#processes: 4  
25: proc 3 (Reserve2:1) Reservation.pml:30 (state 3)  
25: proc 2 (Reserve1:1) Reservation.pml:22 (state 3)  
25: proc 1 (Customer:1) Reservation.pml:10 (state 2)  
25: proc 0 (init:1) Reservation.pml:40 (state 4)  
4 processes created

[queues, step 25]

q 1 :: (Customer(1):in): [ATREADY\_BOOKED, 0, 0]  
q 2 :: (Customer(1):out):