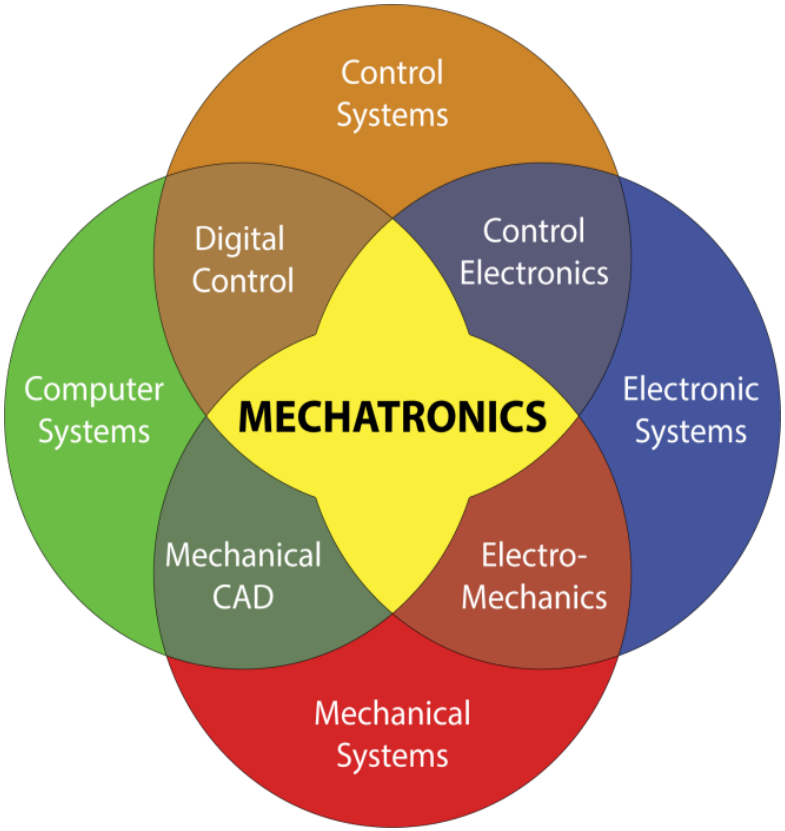
UNIT 4b: KEY CONCEPTS



Finite State Machine:

- Boolean Inputs
- Boolean Output
- Can only be in one state at a time

Two Formulations:

- Moore (which we use in this course)
- Mealy

Design Steps

- Create abstract representation
- Minimize states (optional)
- Perform state assignment (if using gate logic)
- Select flip-flop for implementation (use D flip-flop when possible)
- Generate next-state logic (if using gate logic)
- Simulate and implement the state machine

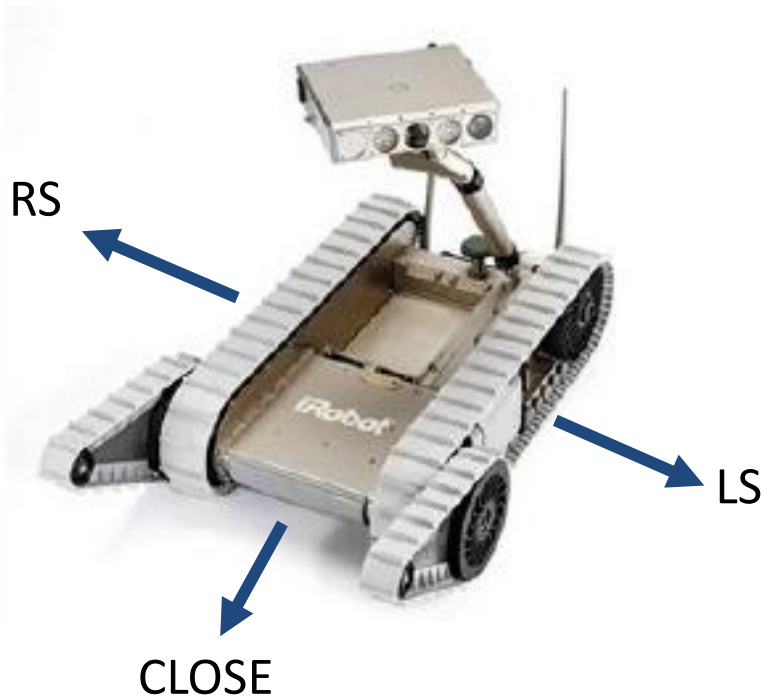
UNIT 4:
SEQUENTIAL LOGIC

PART C:
FINITE STATE MODELING

DEVELOP A CONTROL SYSTEM FOR THE iROBOT 510 PACKBOT

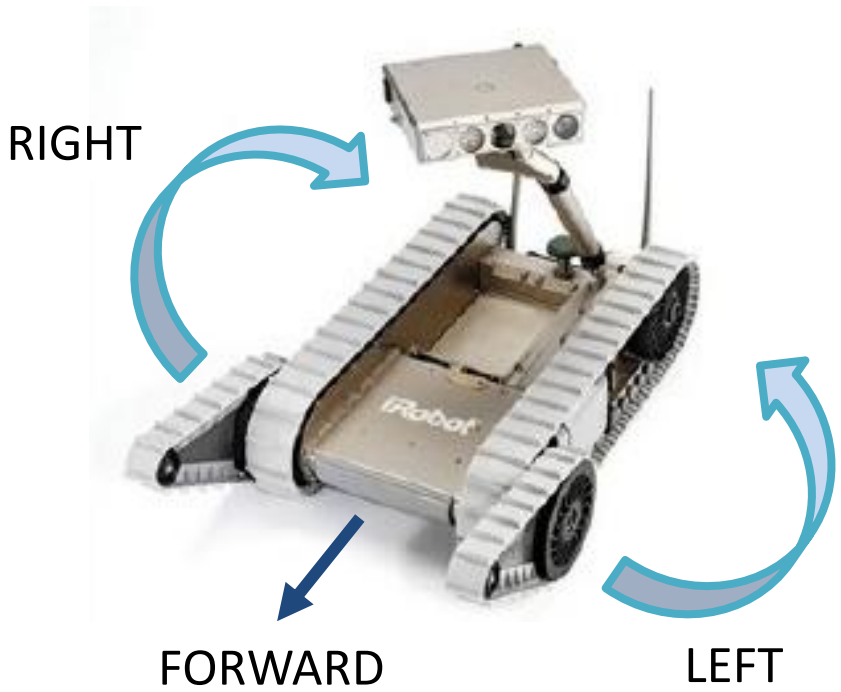
- The iRobot 510 Packbot is designed for bomb disposal, surveillance and reconnaissance, CBRN detection and HazMat handling operations
- Your focus is on navigating the Packbot to an object for interaction

THE PACKBOT BASE HAS THREE SENSORS FOR OBJECT DETECTION



- All three sensors are mounted to the robot base
- LS is active if an object is detected to the robot's left side
- RS is active if an object is detected to the robot's right side
- CLOSE is active if an object is detected within 2 inches of the robot's front side

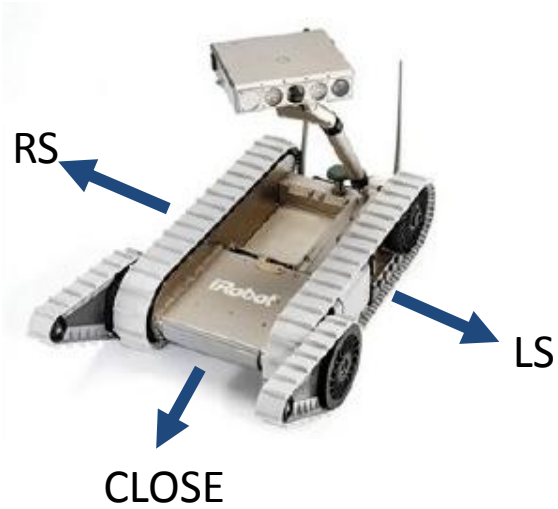
THE PACKBOT MOTOR CONTROLLER HAS THREE INPUTS

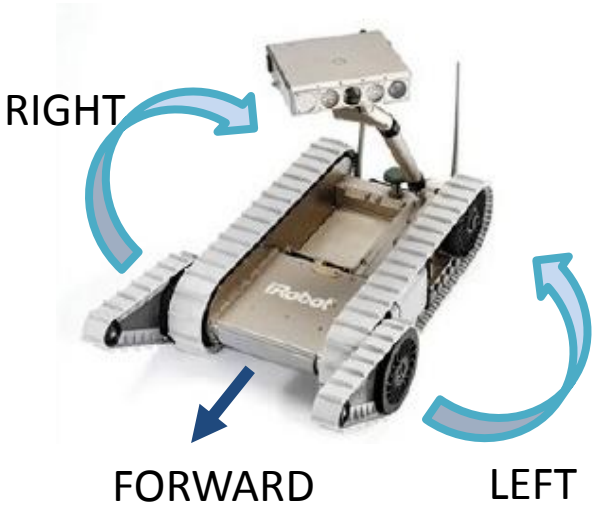


- FORWARD should be activated when the robot is to move forward and deactivated when the robot is to stop.
- LEFT should be activate to turn the robot to the left.
- RIGHT should be activated to turn the robot to the right.

CERTAIN OPERATING PRINCIPLES SHOULD BE FOLLOWED

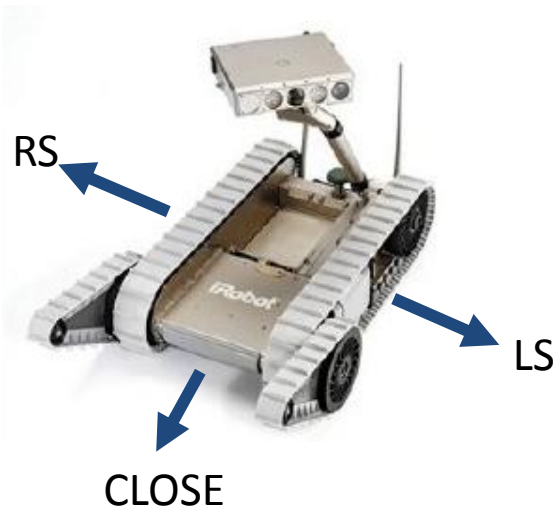
- When LS is active and RS is not, then the robot should turn left.
- When RS is active and LS is not, then the robot should turn right.
- When both LS and RS are active, then the robot should go straight.
- When CLOSE is active, the robot should stop.
- The robot should start from stop.
- When none of the LS, RS, and CLOSE inputs are active, the robot should go forward and turn toward right to circle around.

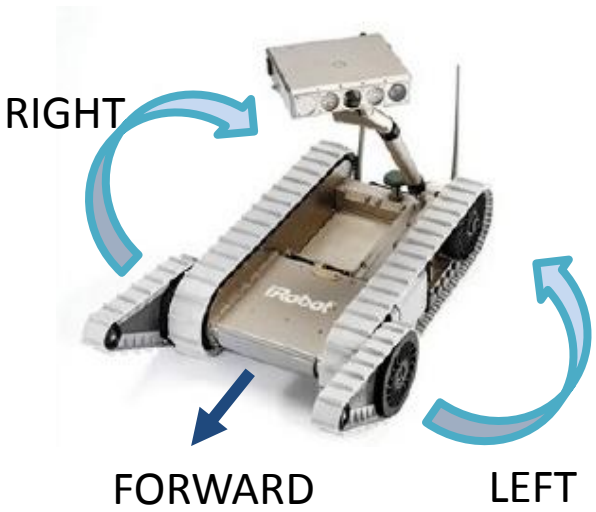




IN ADDITION, CERTAIN CONSTRAINTS ARE IN EFFECT

- LEFT and RIGHT cannot be activated unless FORWARD is also activated.
- To stop the robot, FORWARD, LEFT, and RIGHT should be deactivated.
- To move the robot forward in the current direction, both LEFT and RIGHT need to be deactivated, and FORWARD needs to be activated.





LABEL AND IDENTIFY MEANING FOR INPUTS AND OUTPUTS

- Inputs (LS, RS, CL)
  - LS = 0 LS deactivated (off) (nothing is sensed to the left)
  - LS = 1 LS activated (on) (something is sensed to the left)
  - RS = 0 RS deactivated (off) (nothing is sensed to the right)
  - RS = 1 RS activated (on) (something is sensed to the right)
  - CL = 0 CL deactivated (off) (nothing in front of robot)
  - CL = 1 RS activated (on) (object detected in front of robot)

- Outputs (L, R, F)
  - L = 0 L is low, vehicle does not turn left
  - L = 1 L is activated, vehicle turns left
  - R = 0 R is low, vehicle does not turn right
  - R = 1 R is activated, vehicle turns right
  - F = 0 F is low, vehicle stops
  - F = 1 F is activated, vehicle moves forward in current direction

AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

S0 (STOP)
0 0 0

S1 (LT)
1 0 1

S2 (RT)
0 1 1

S3 (FOR)
0 0 1

AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

By convention, the initial (starting) state is typically named S0. If this is not the case, you should indicate the initial state for your FSM.

S0 (STOP)
0 0 0

S1 (LT)
1 0 1

S2 (RT)
0 1 1

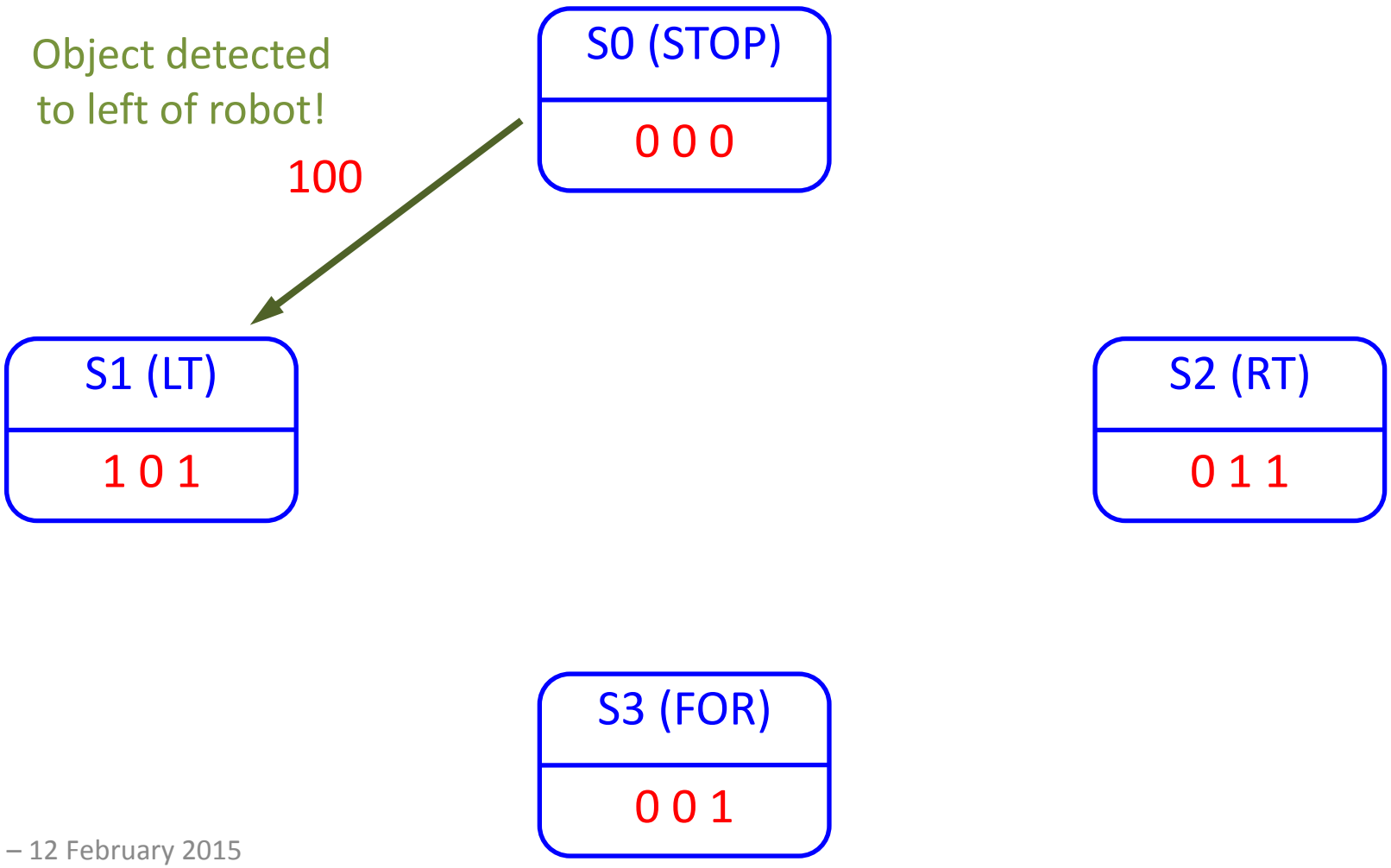
S3 (FOR)
0 0 1

AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

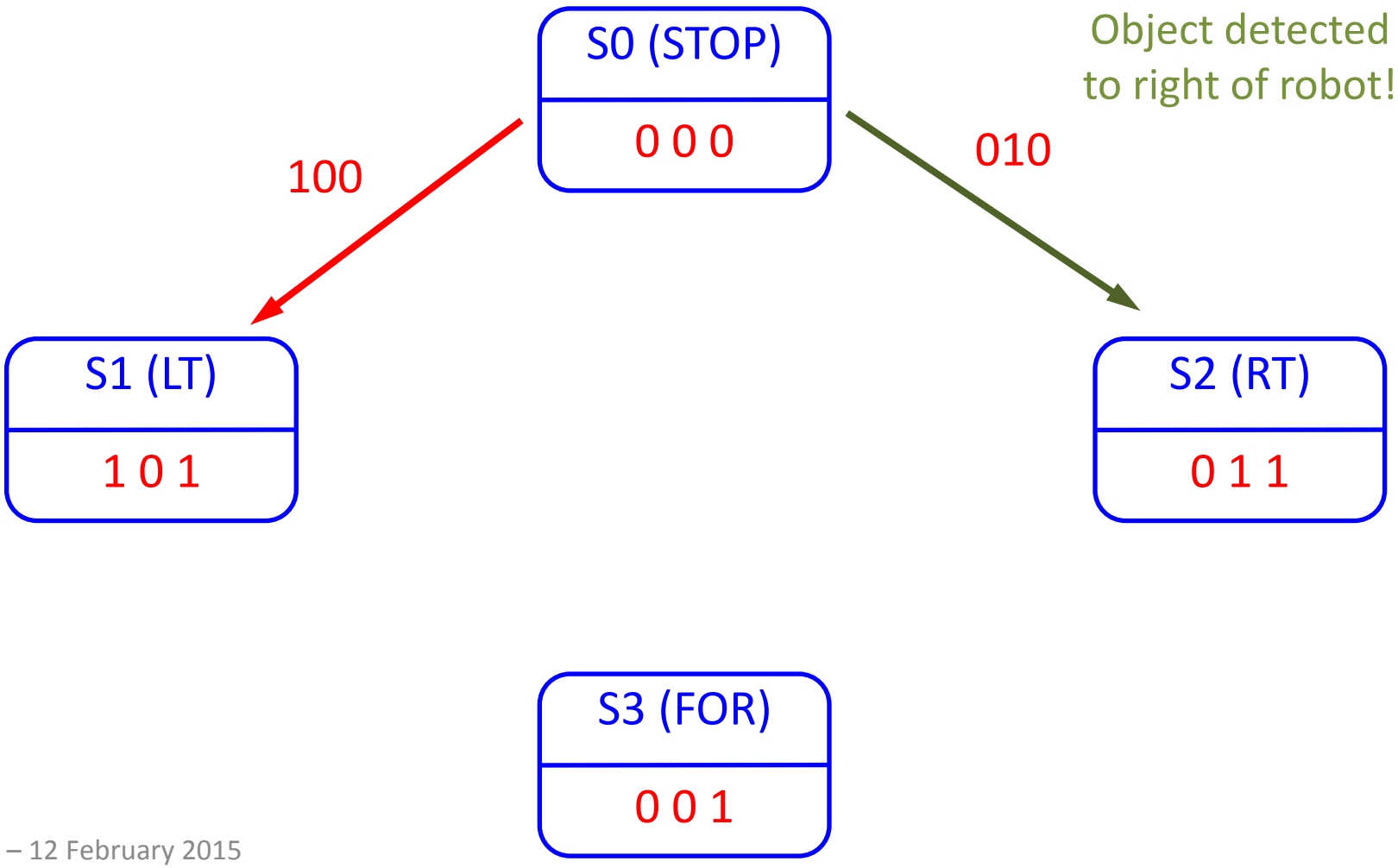


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

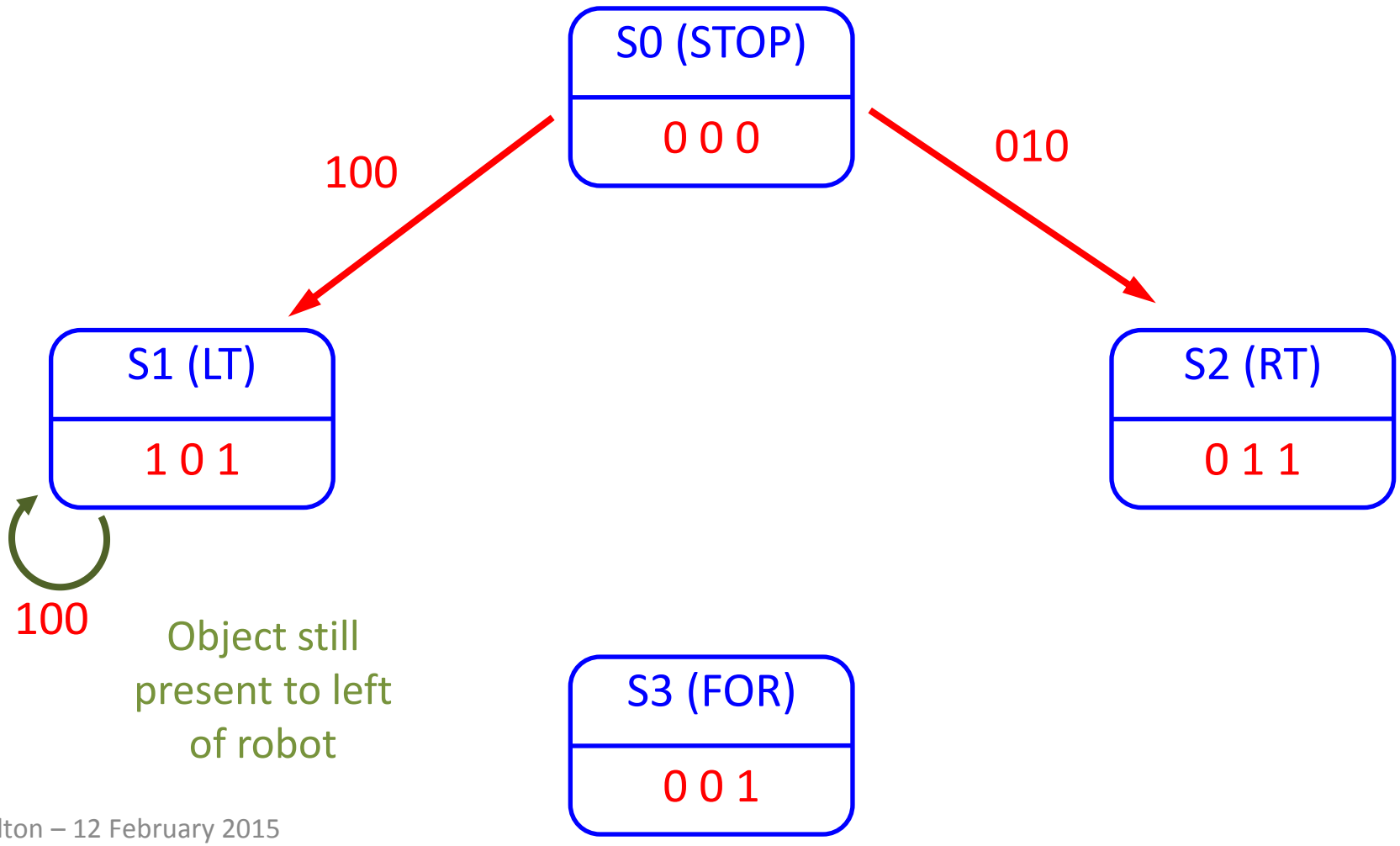


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

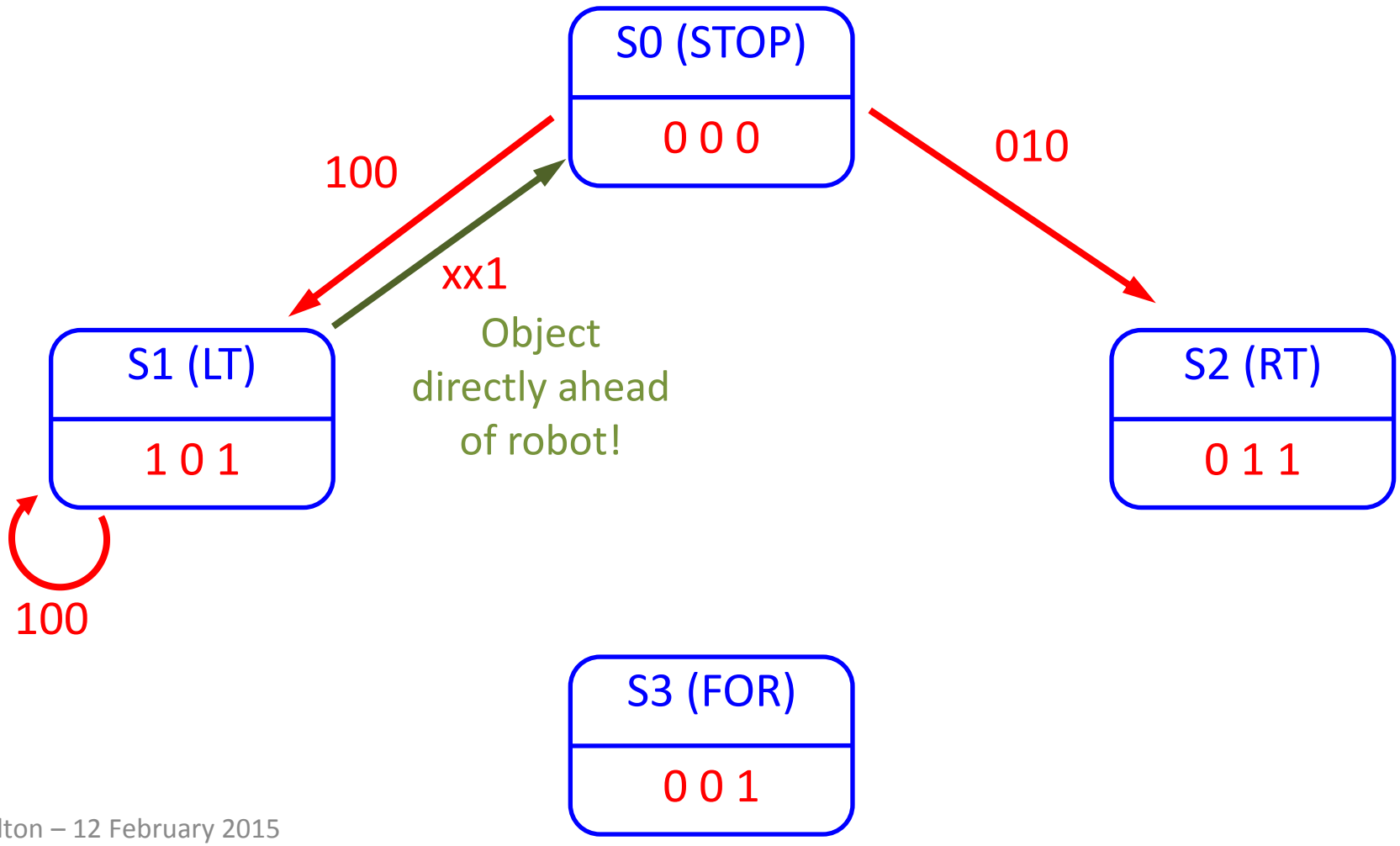


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

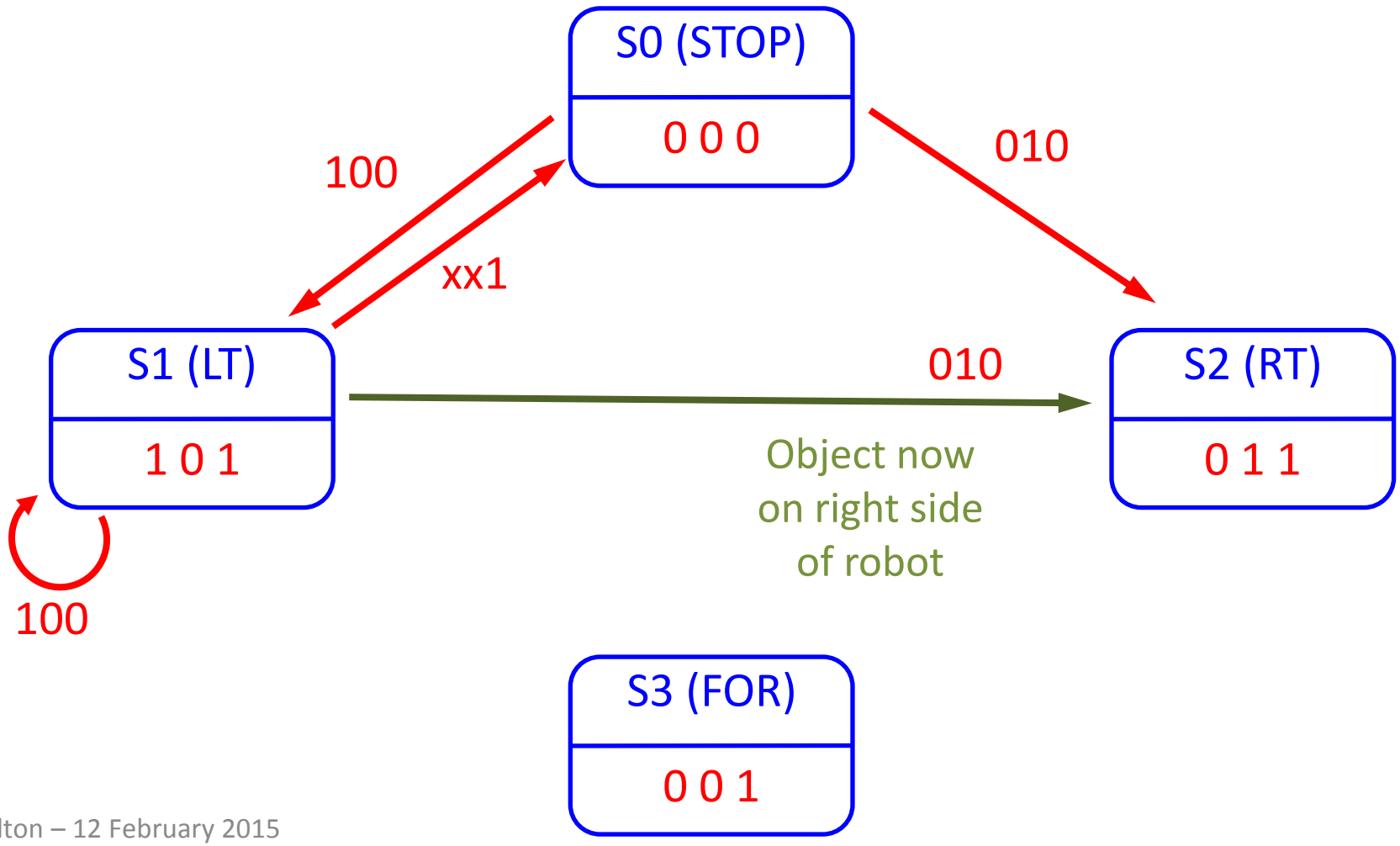


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

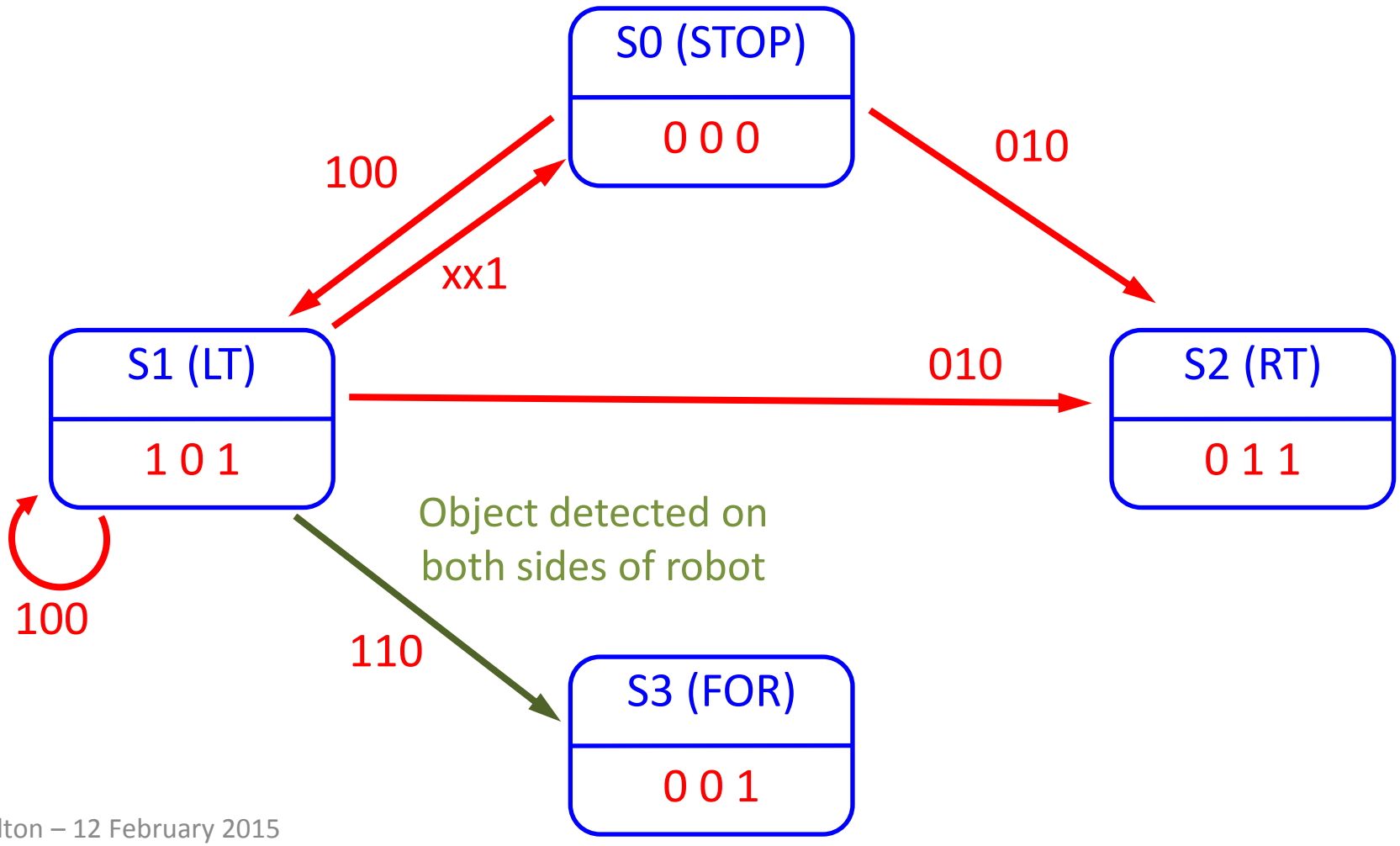


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

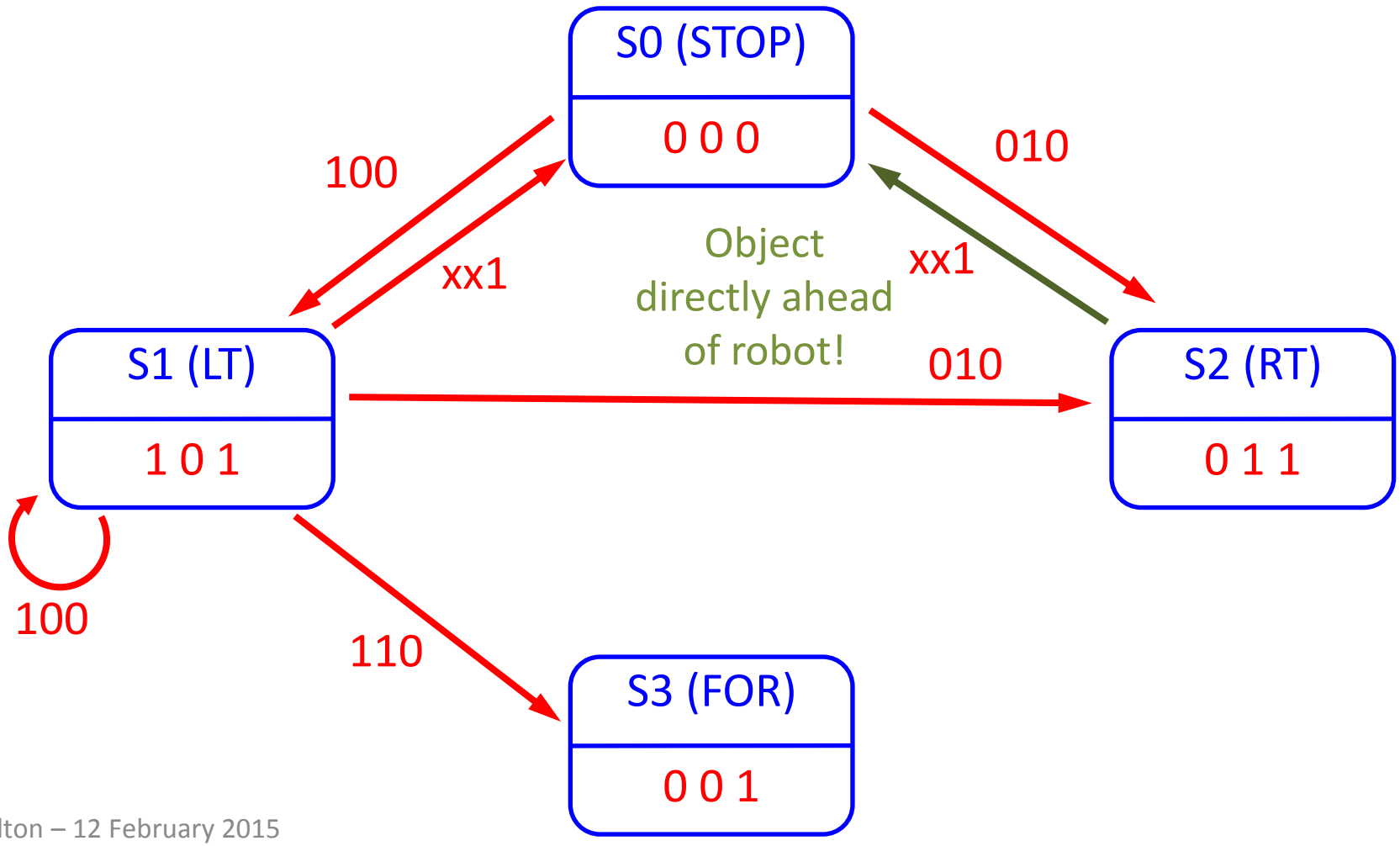


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

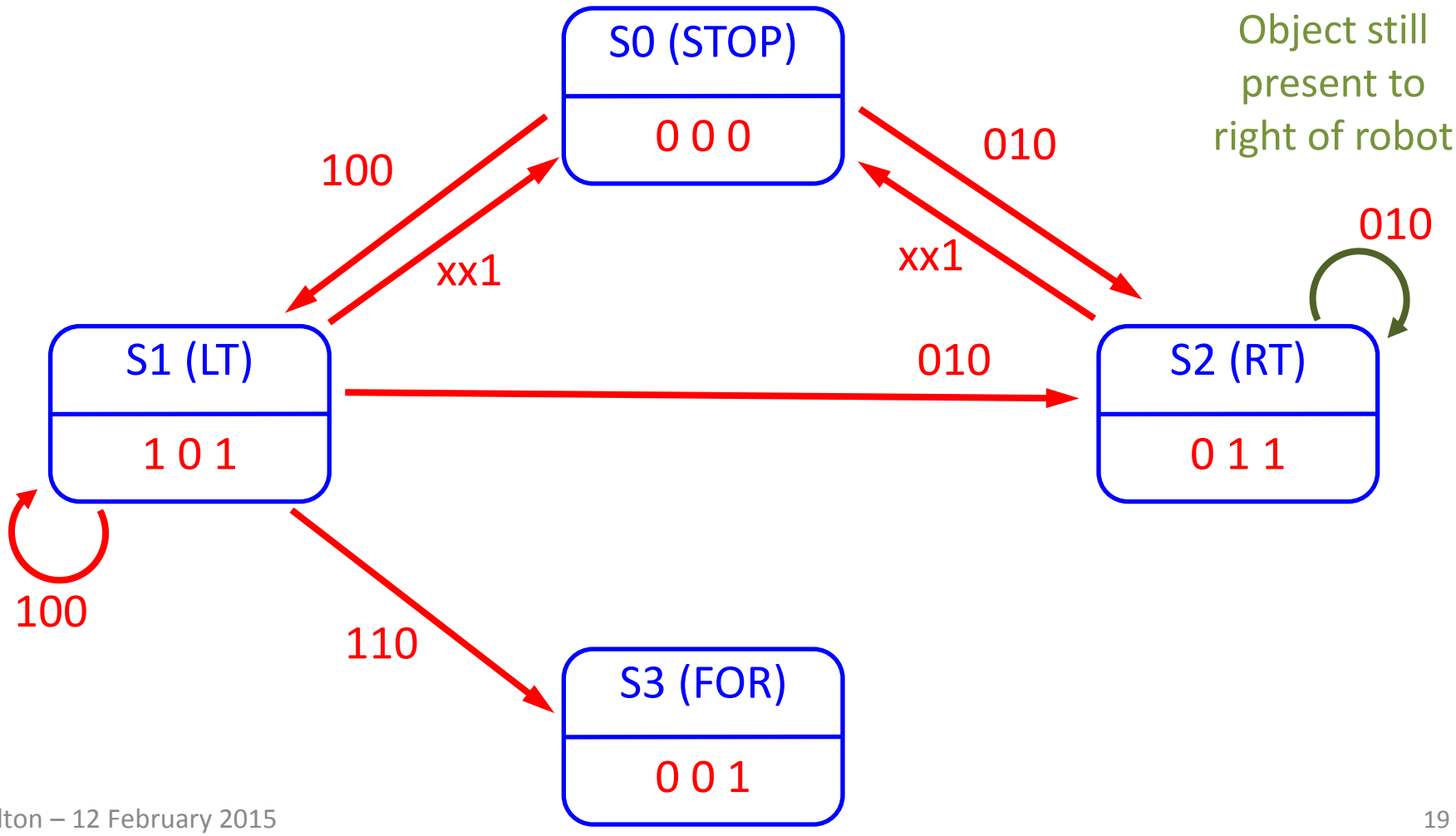


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

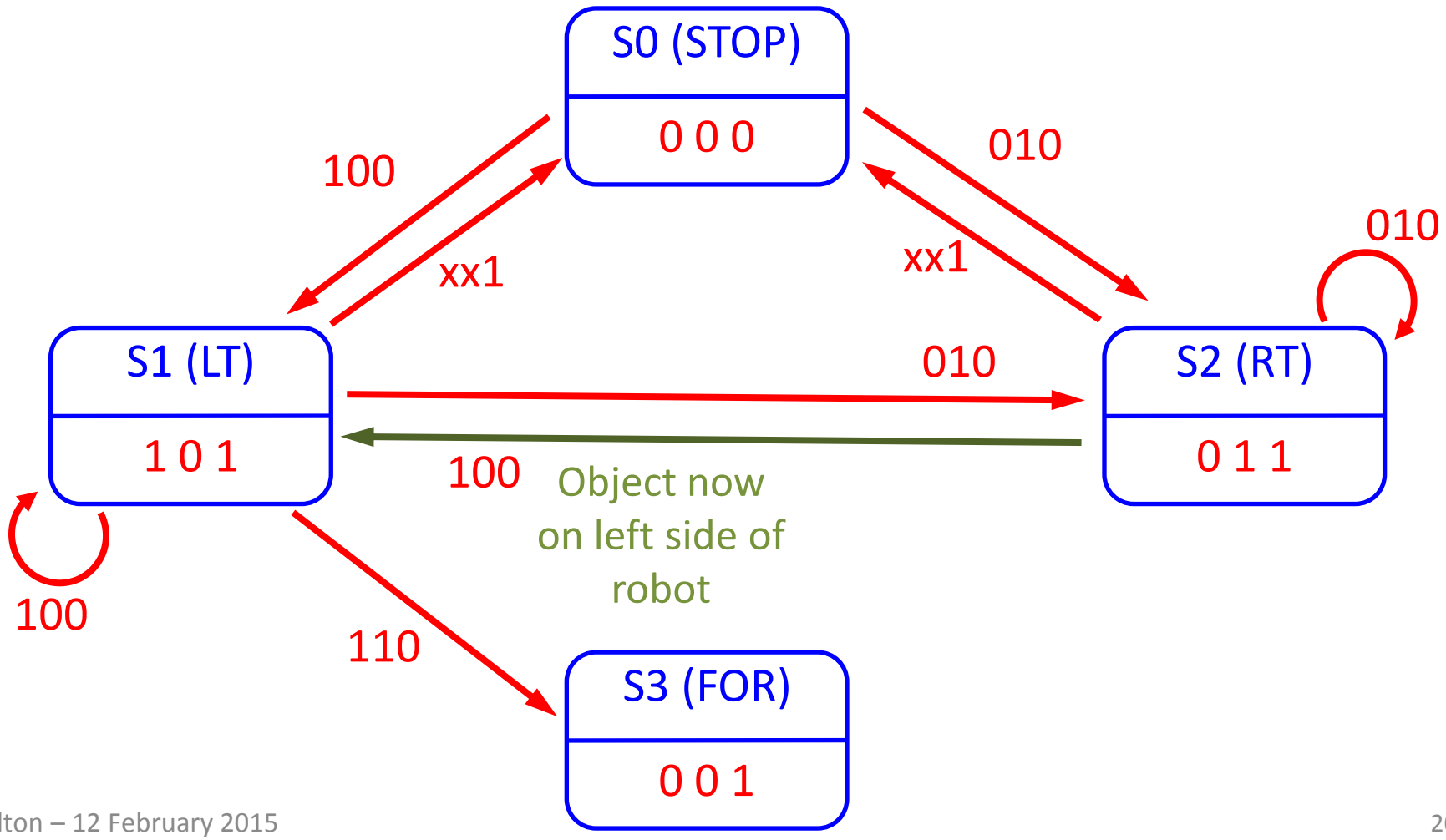


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

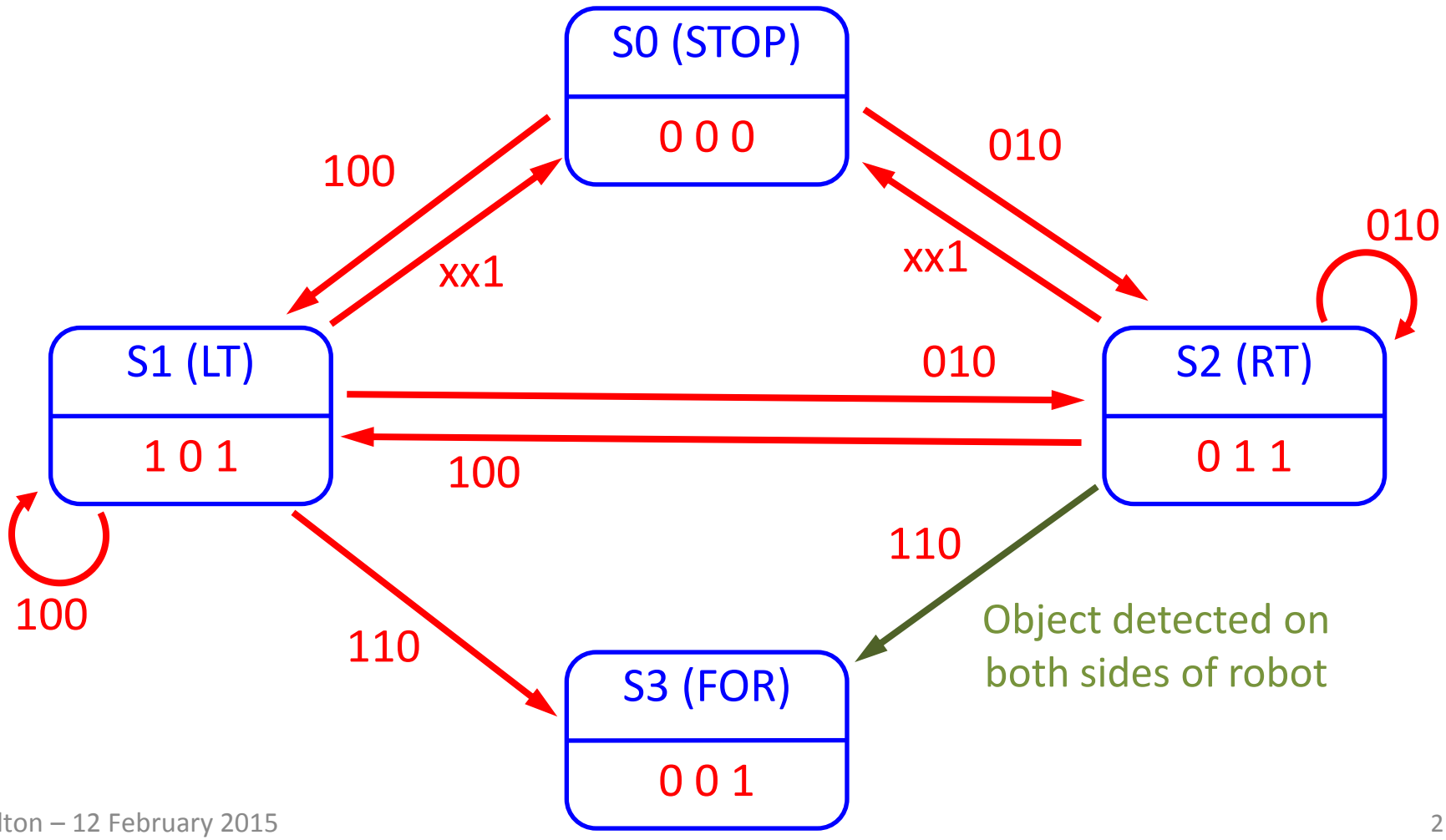


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

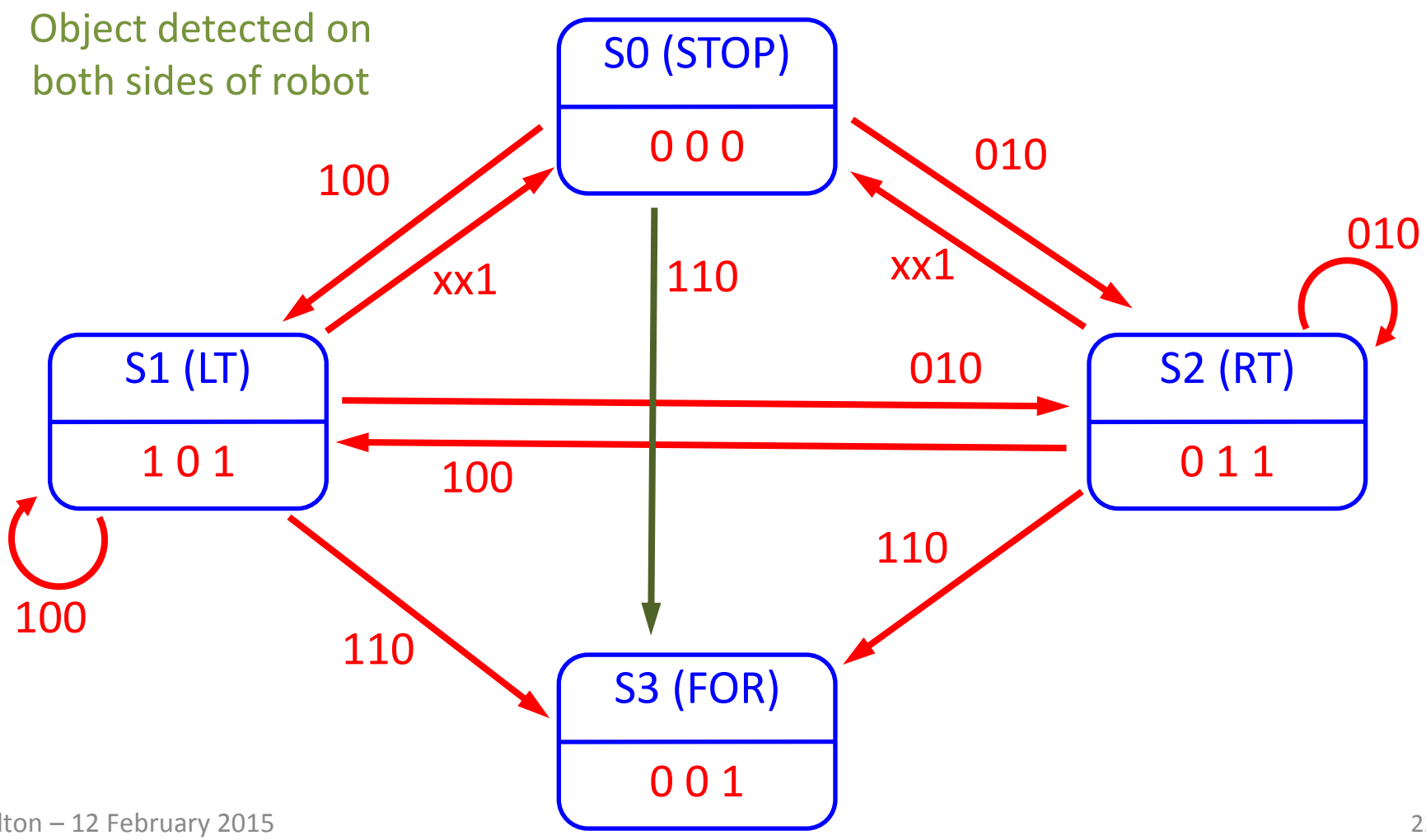


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

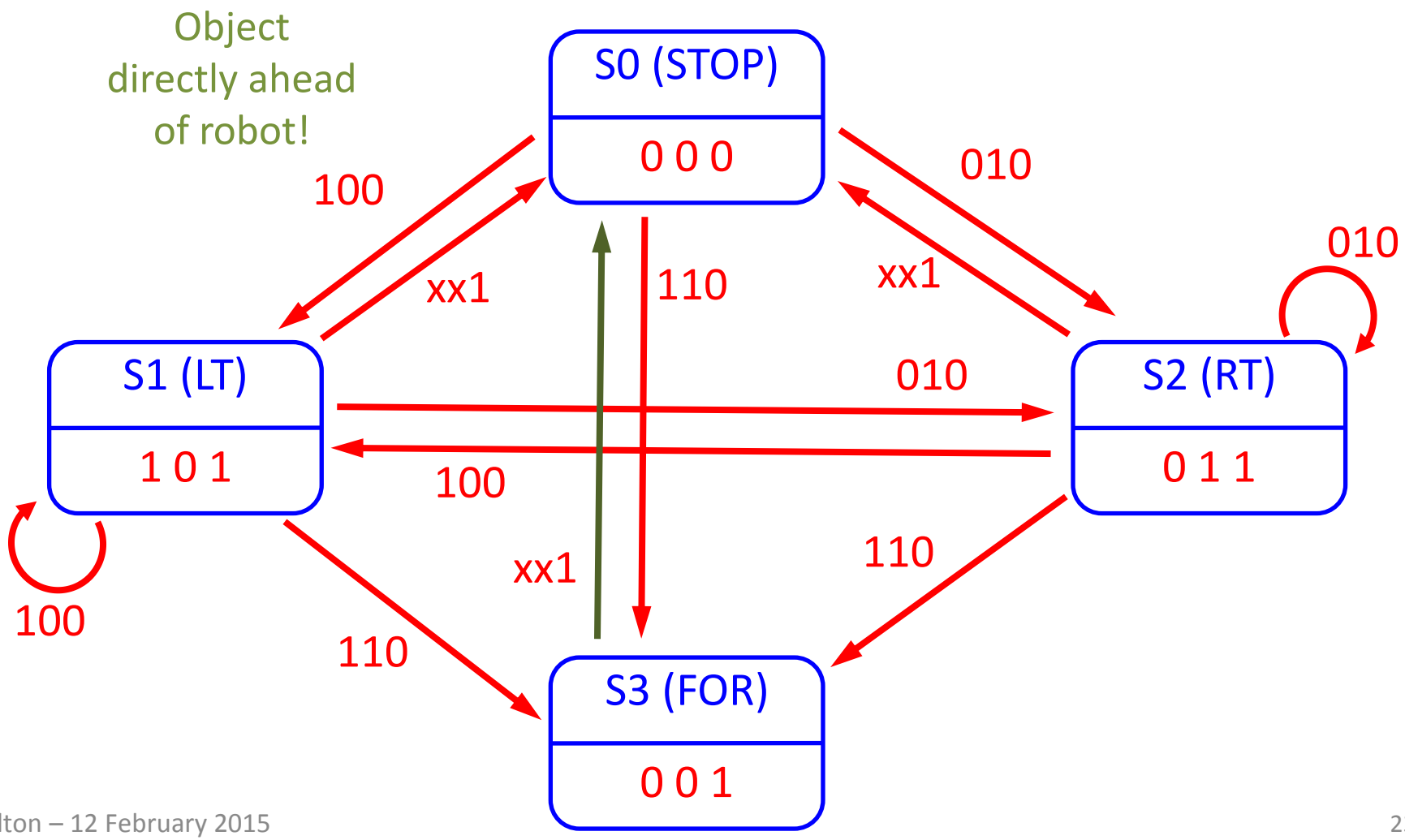


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

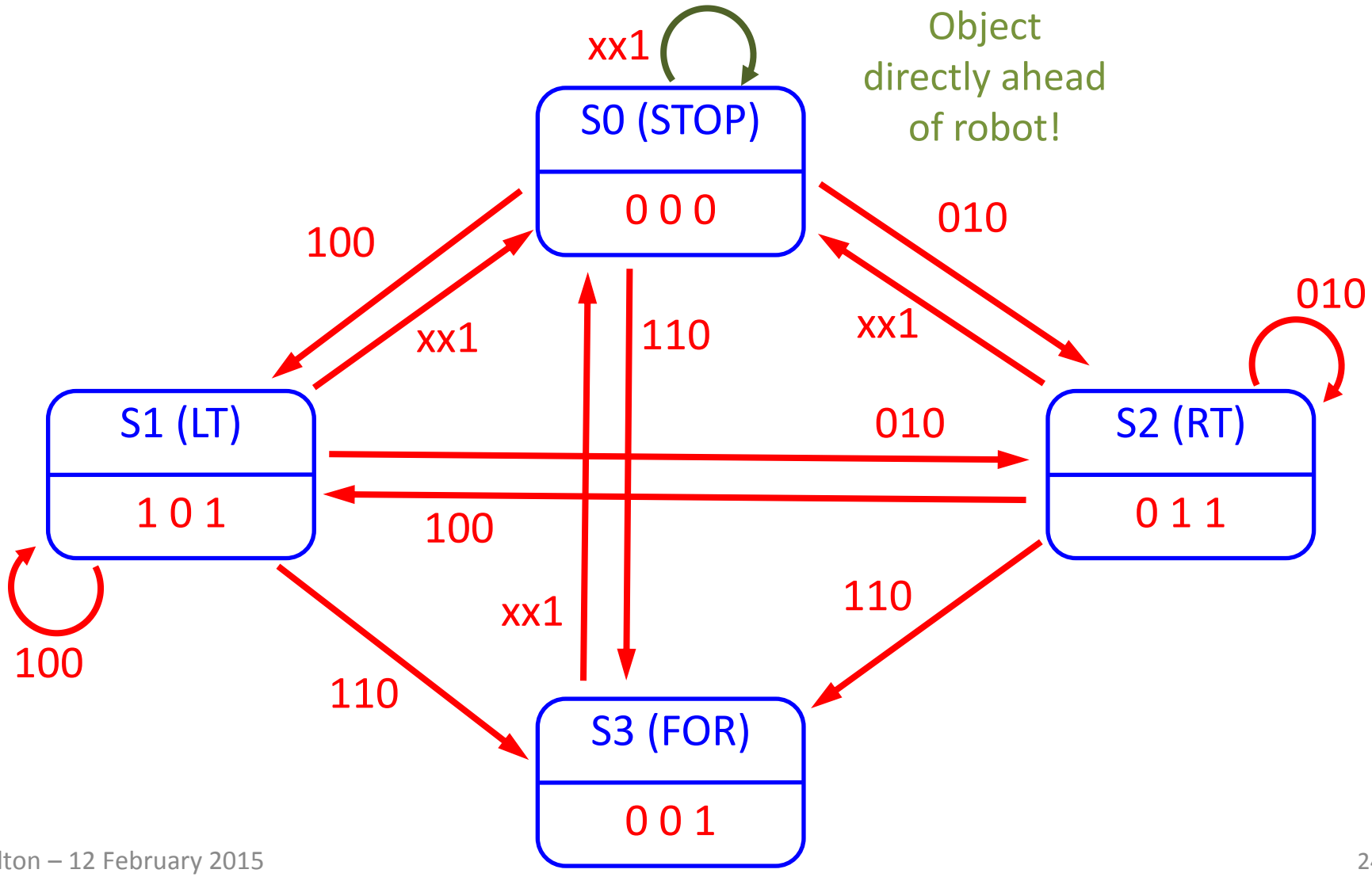


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

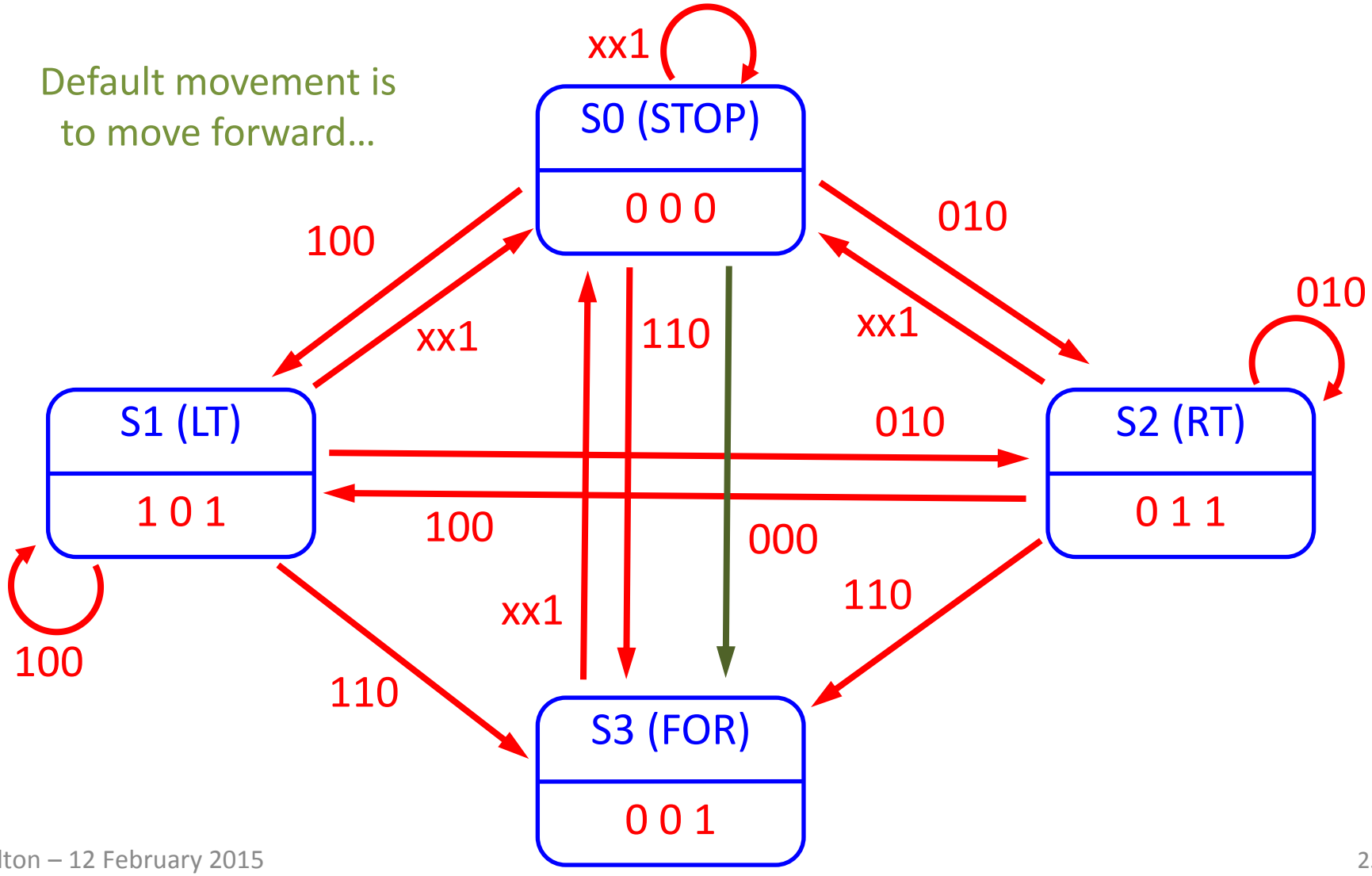


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

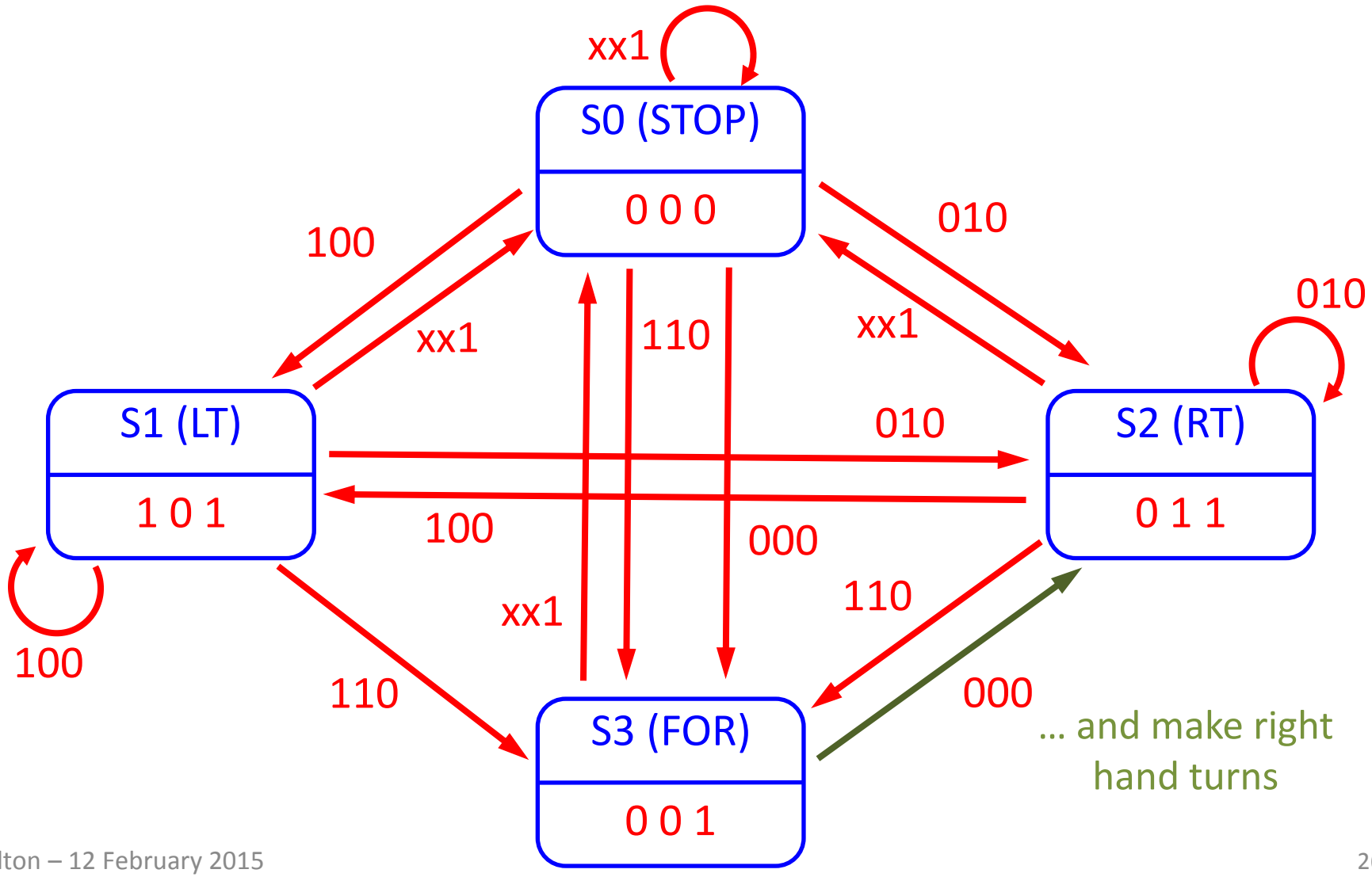


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

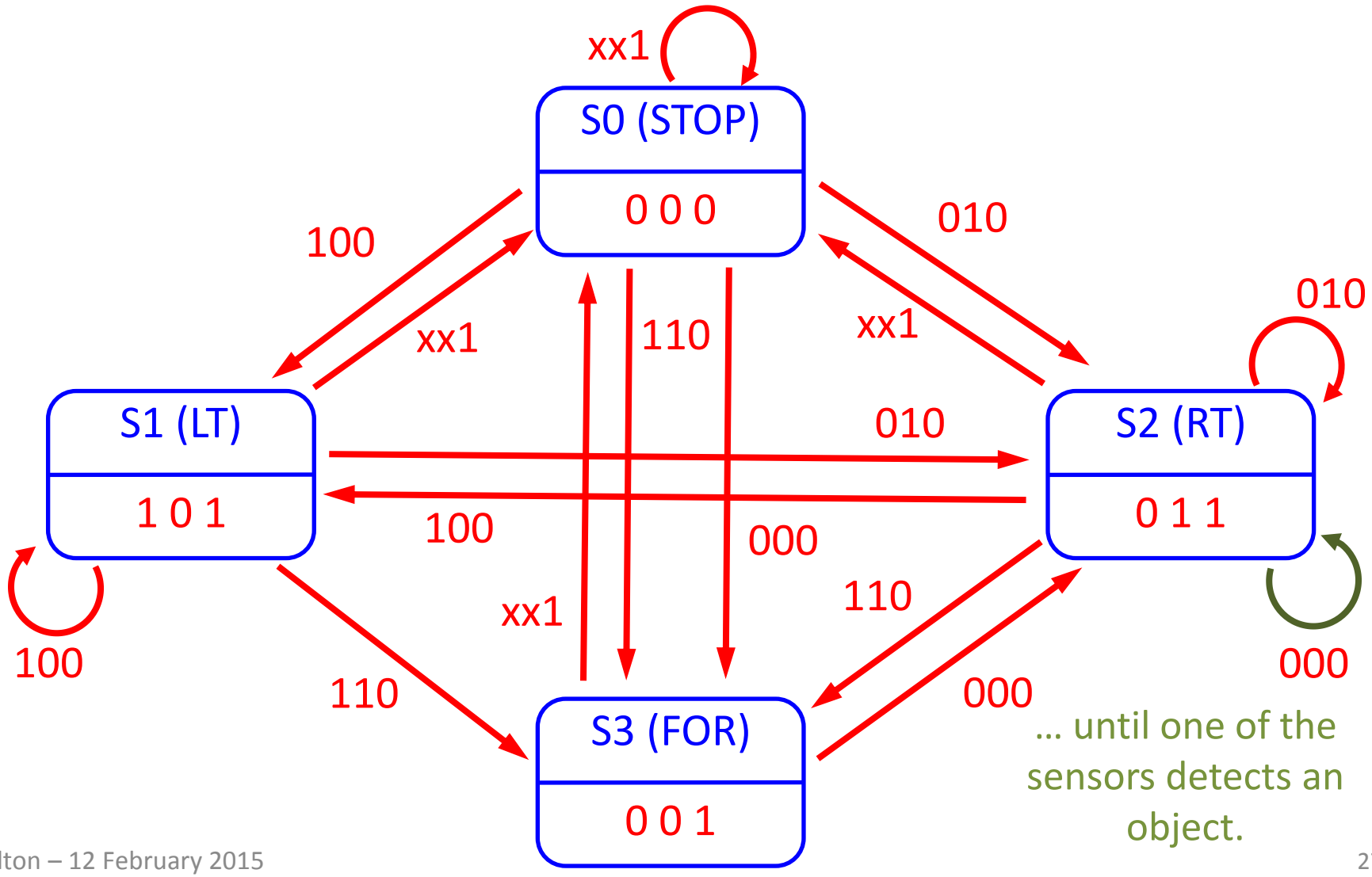


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.



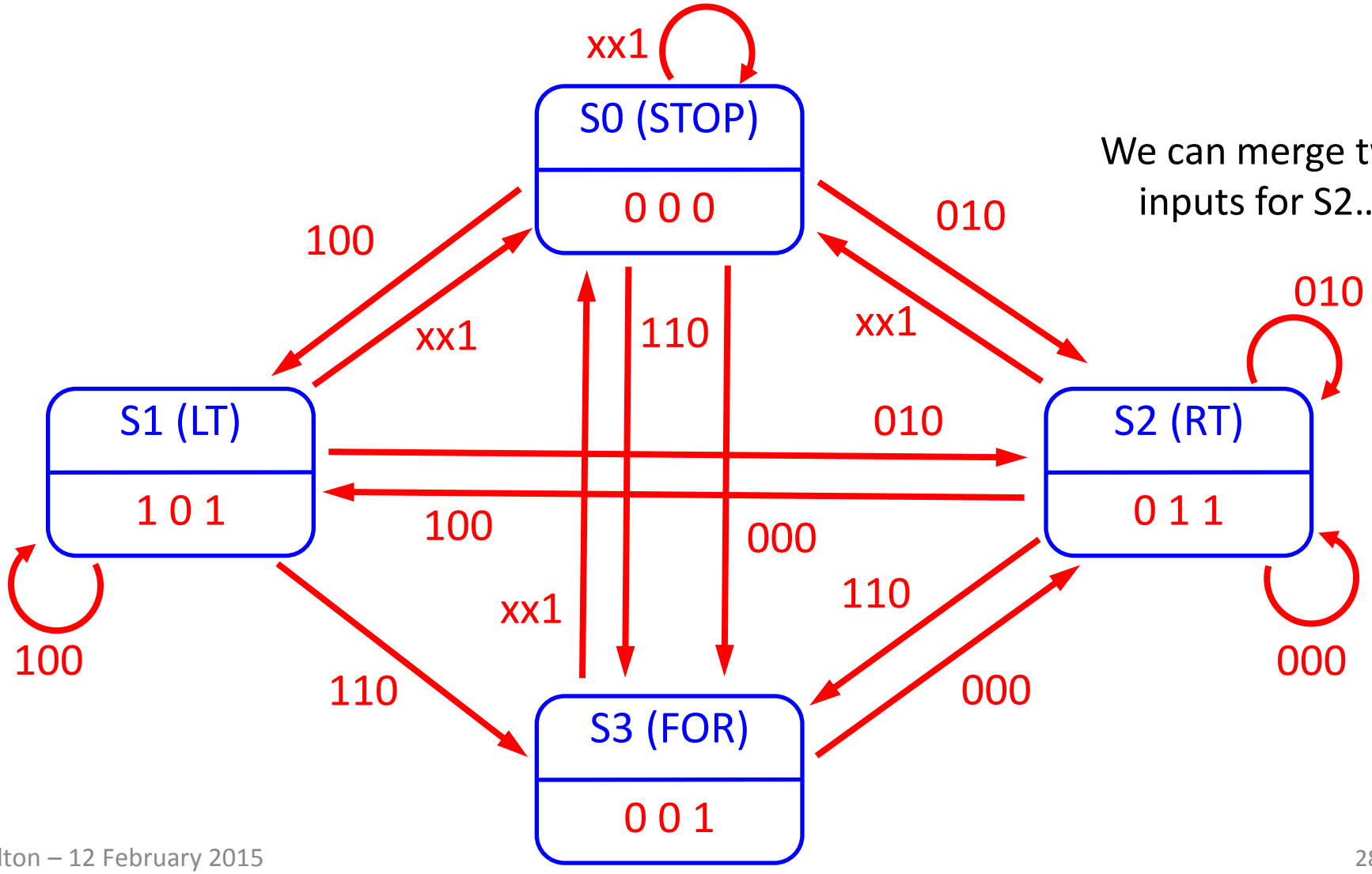
… until one of the sensors detects an object.

AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.



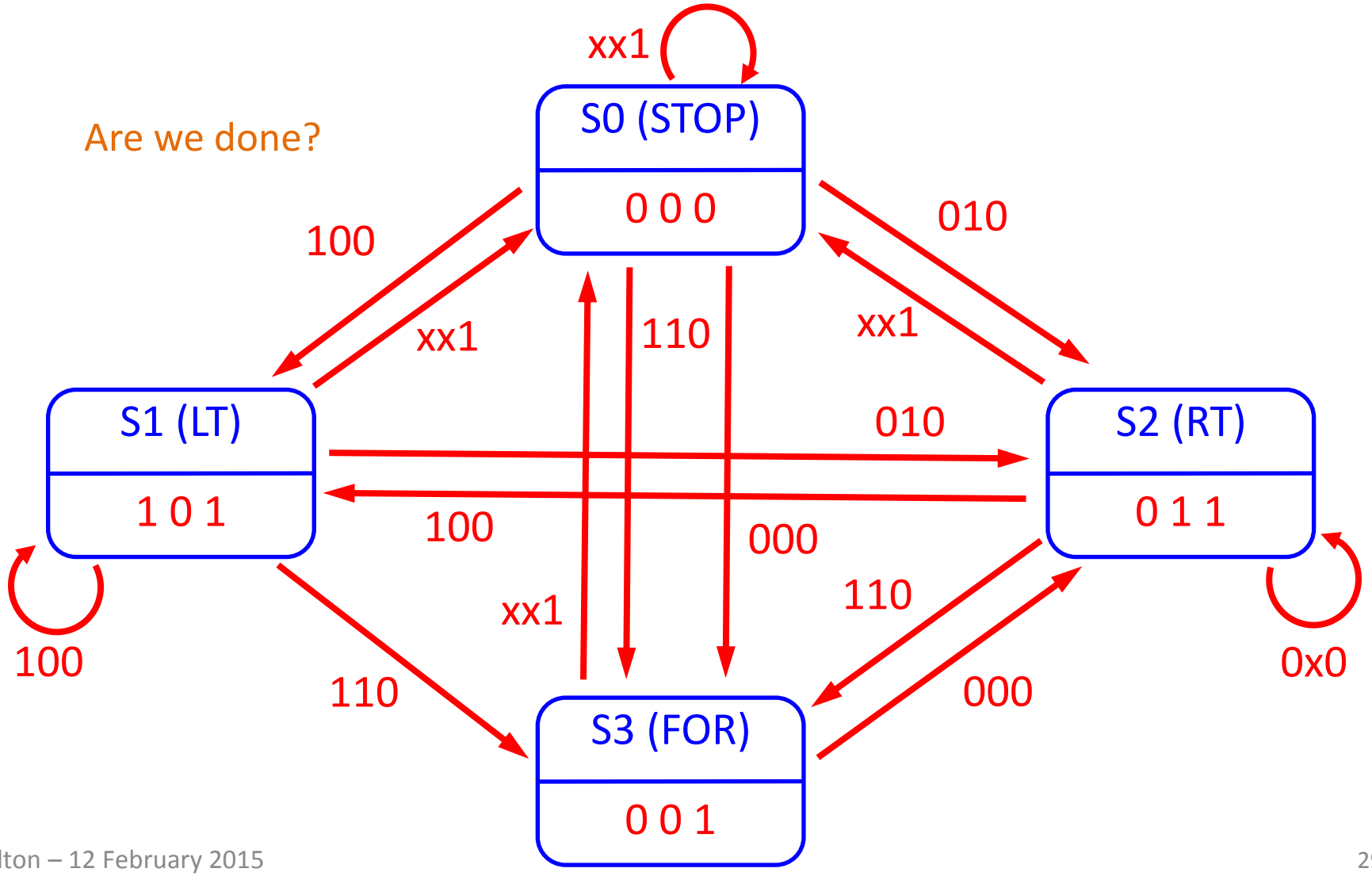
We can merge two inputs for S2…

AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.



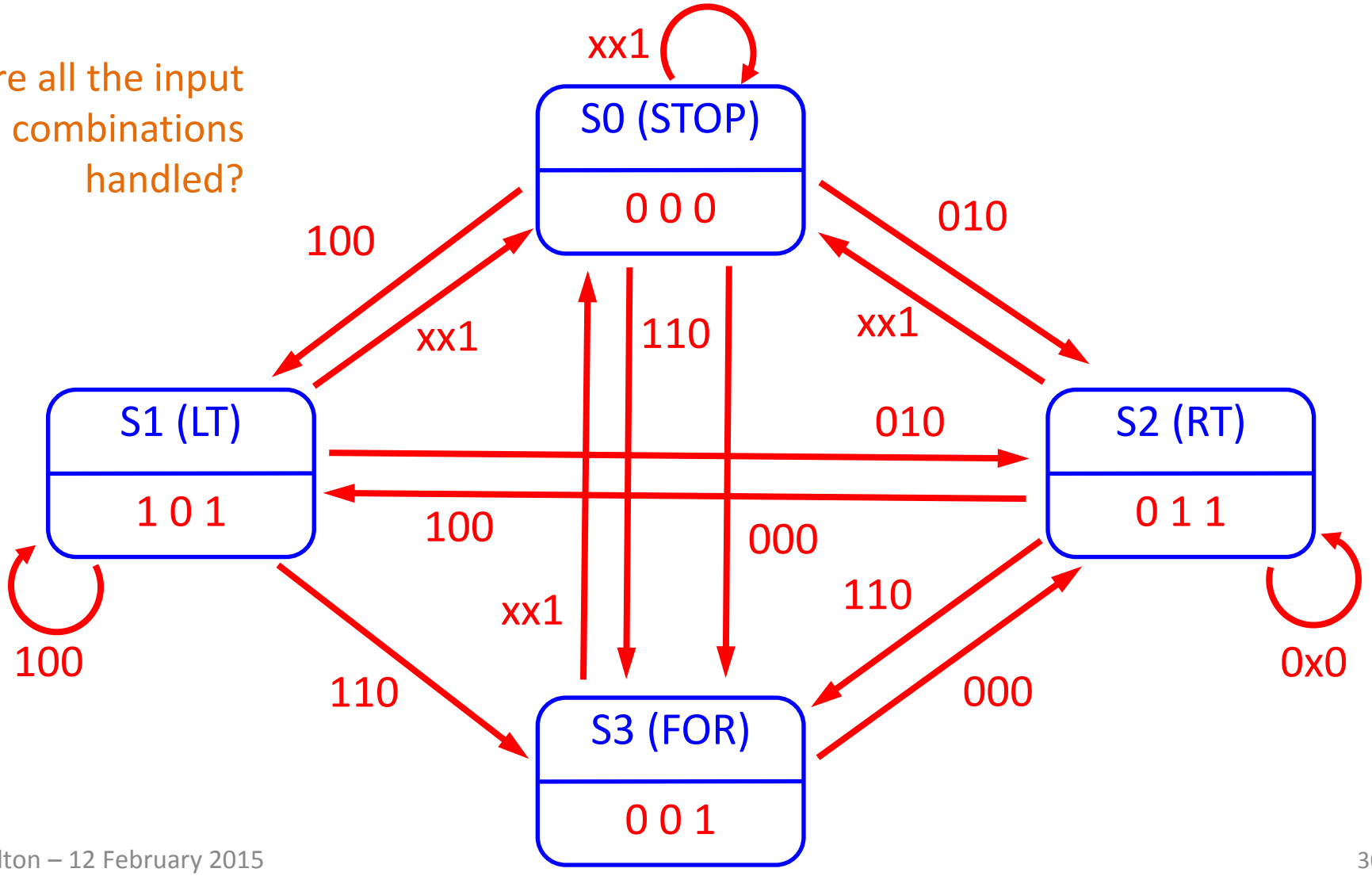
AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

Are all the input combinations handled?

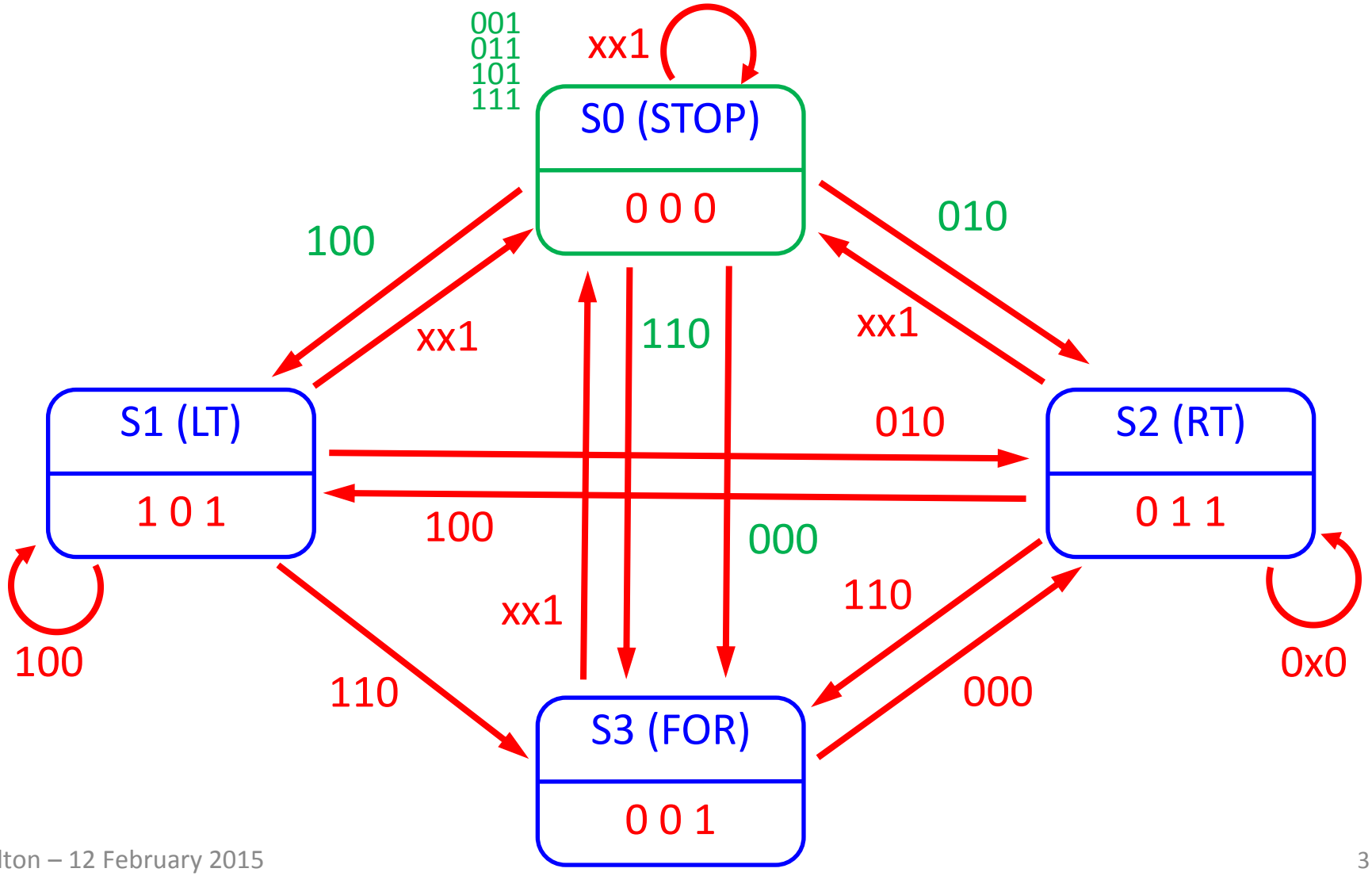


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

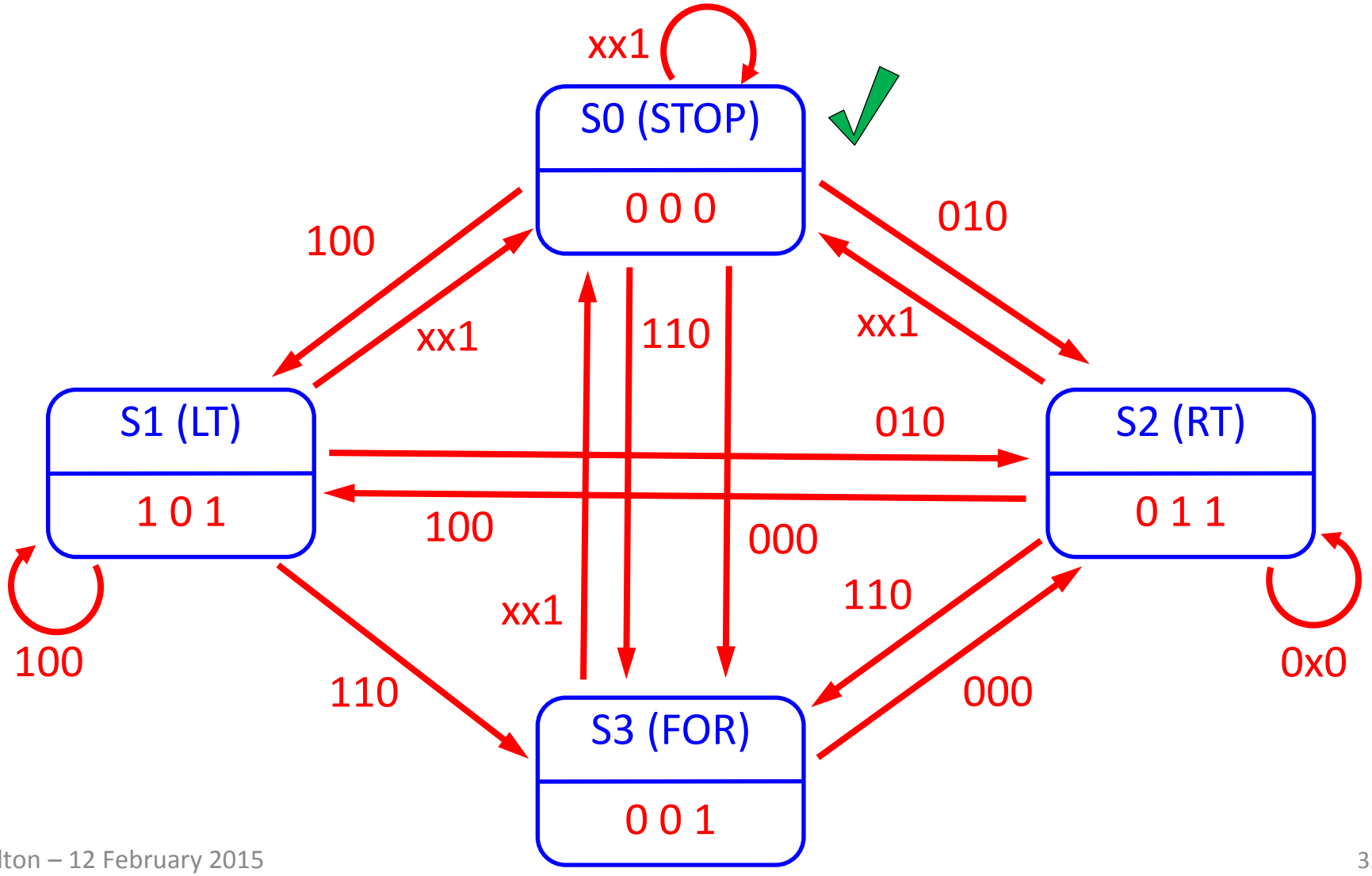


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

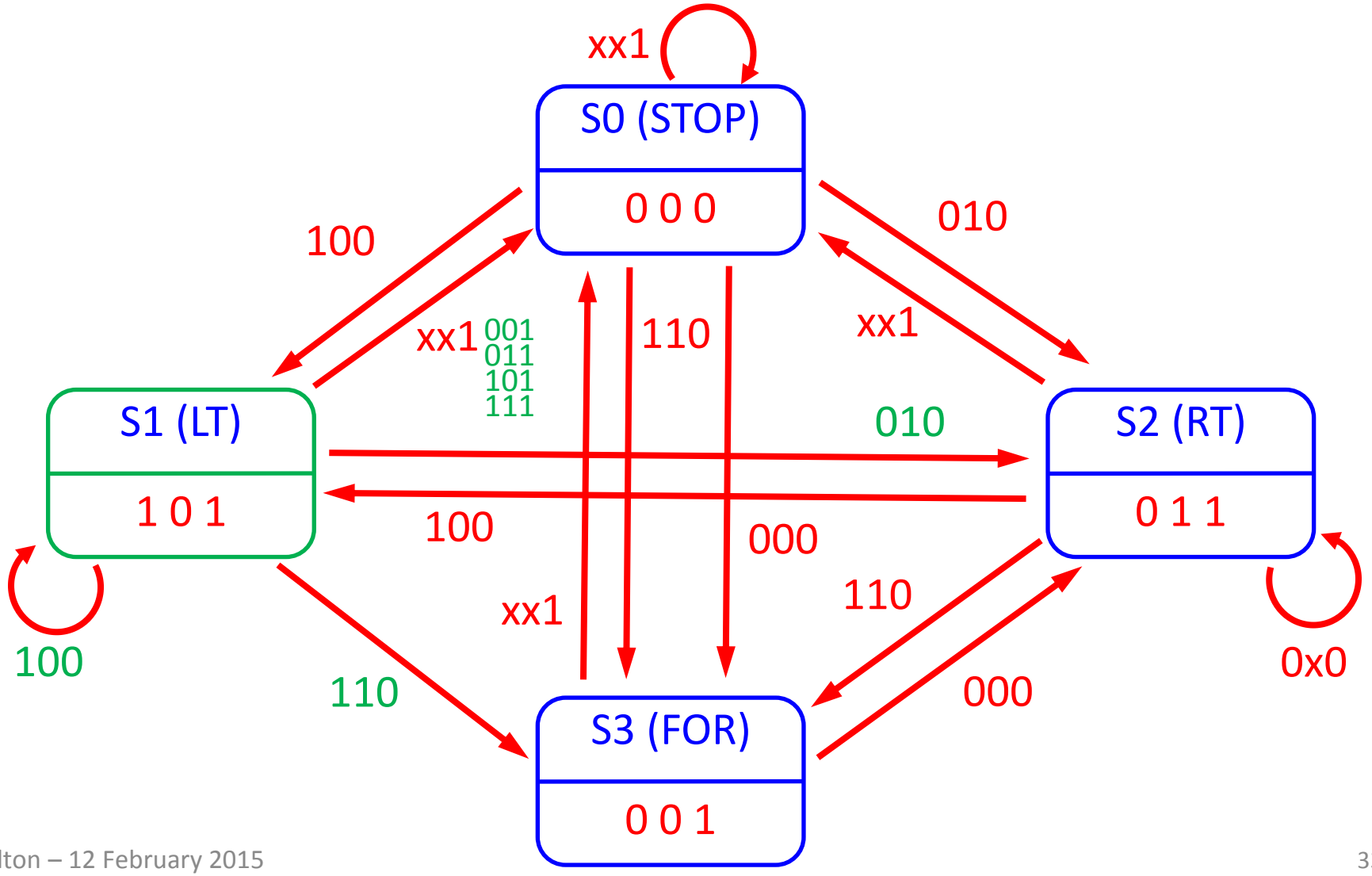


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

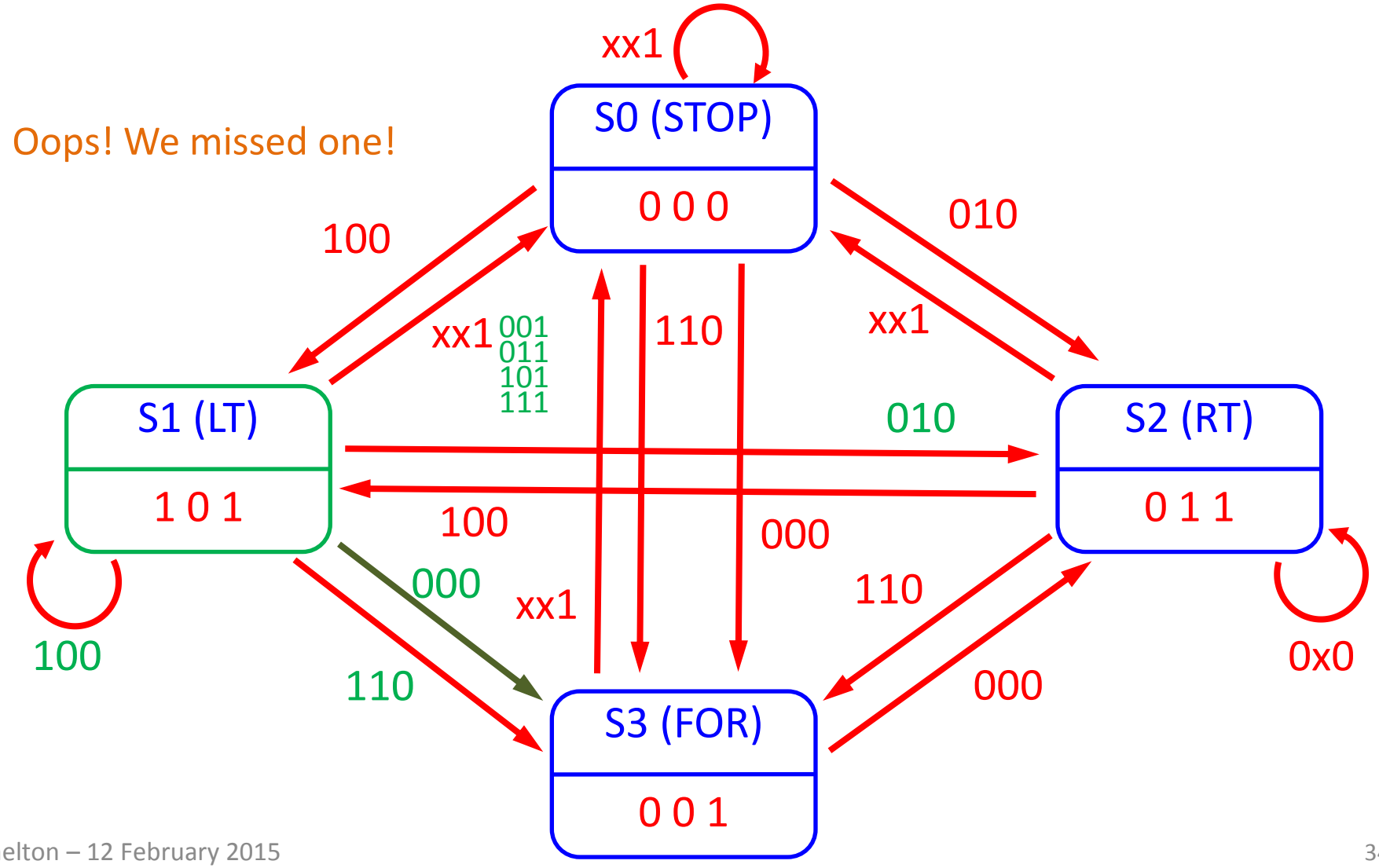


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

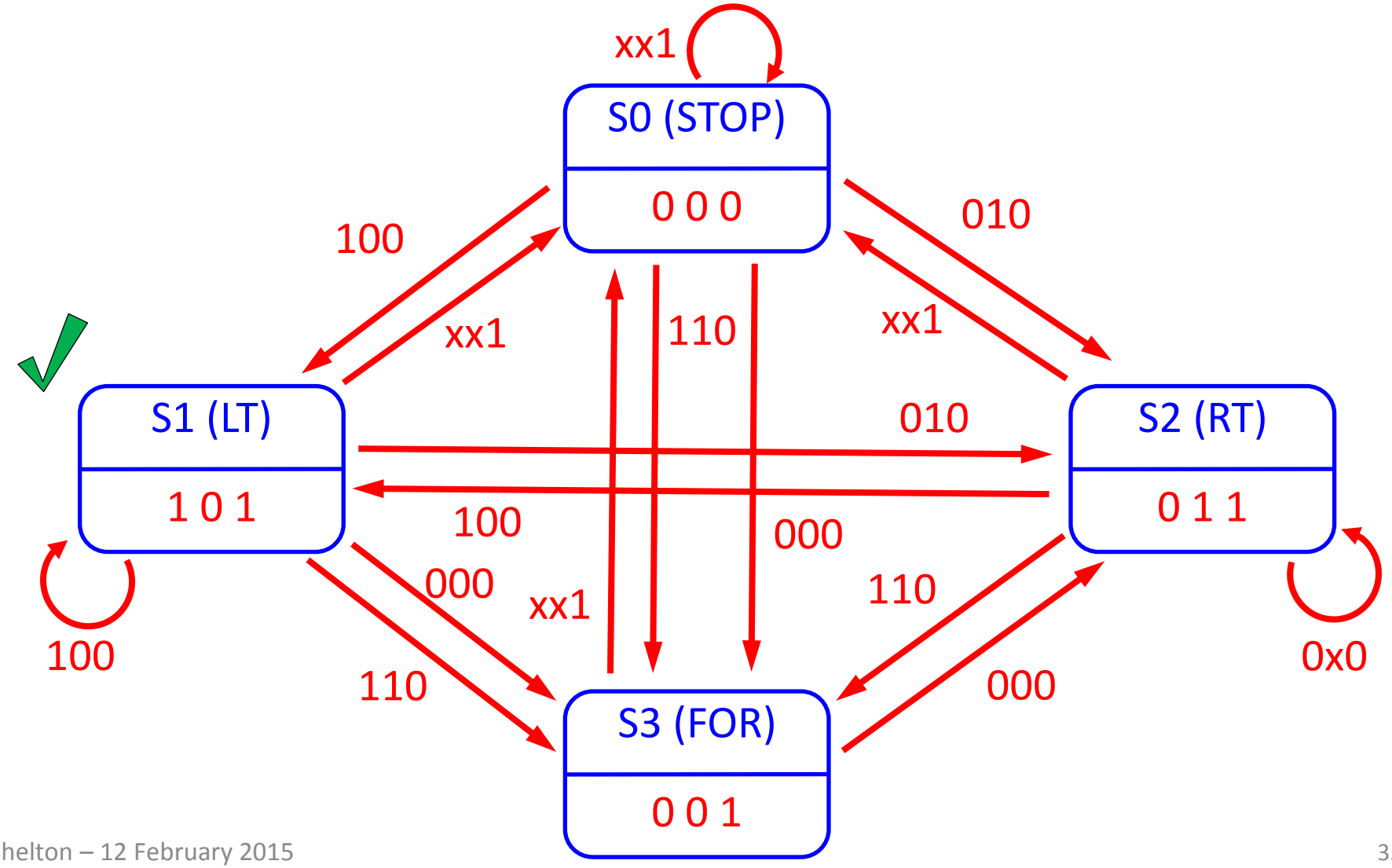


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

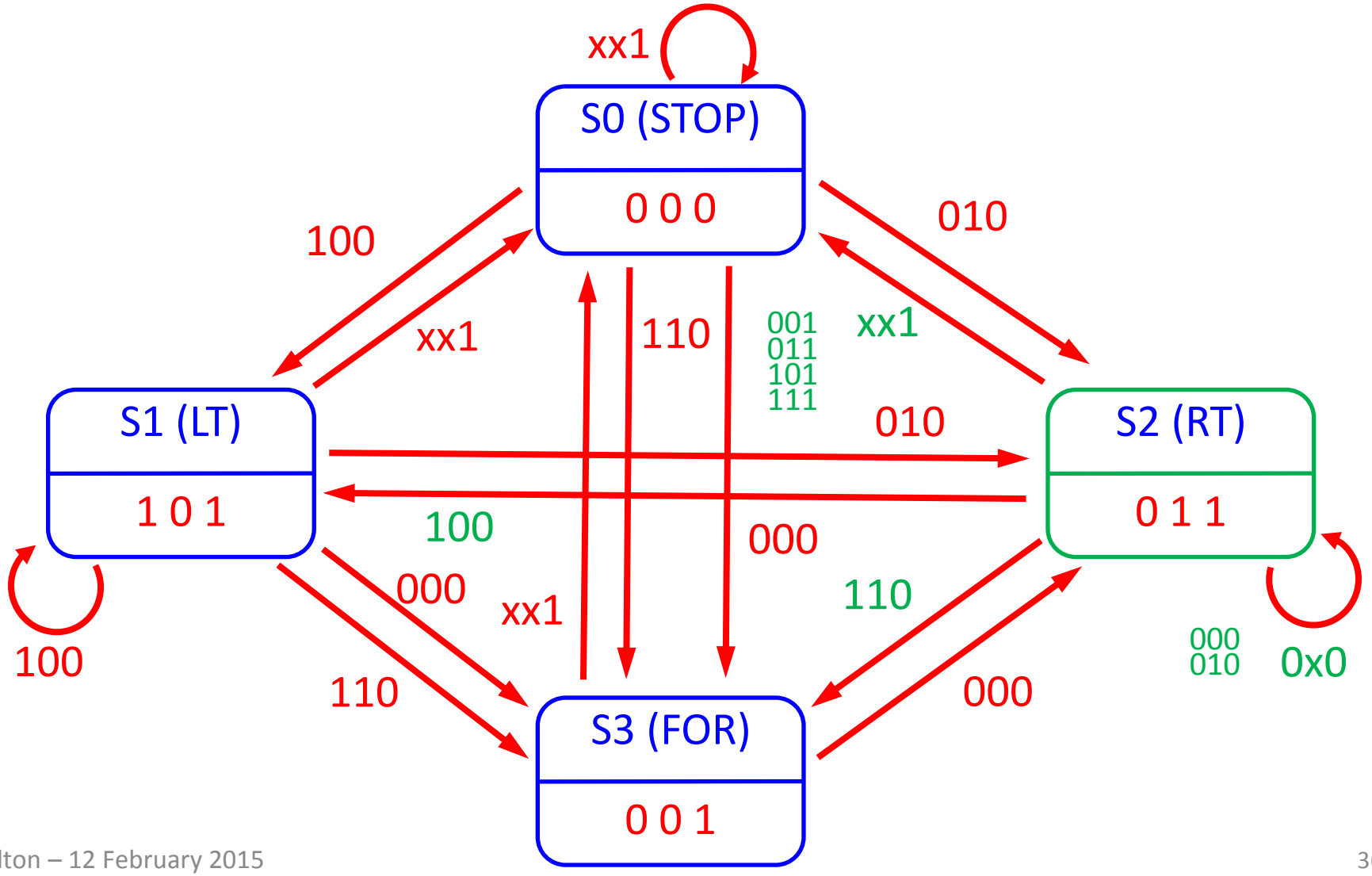


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

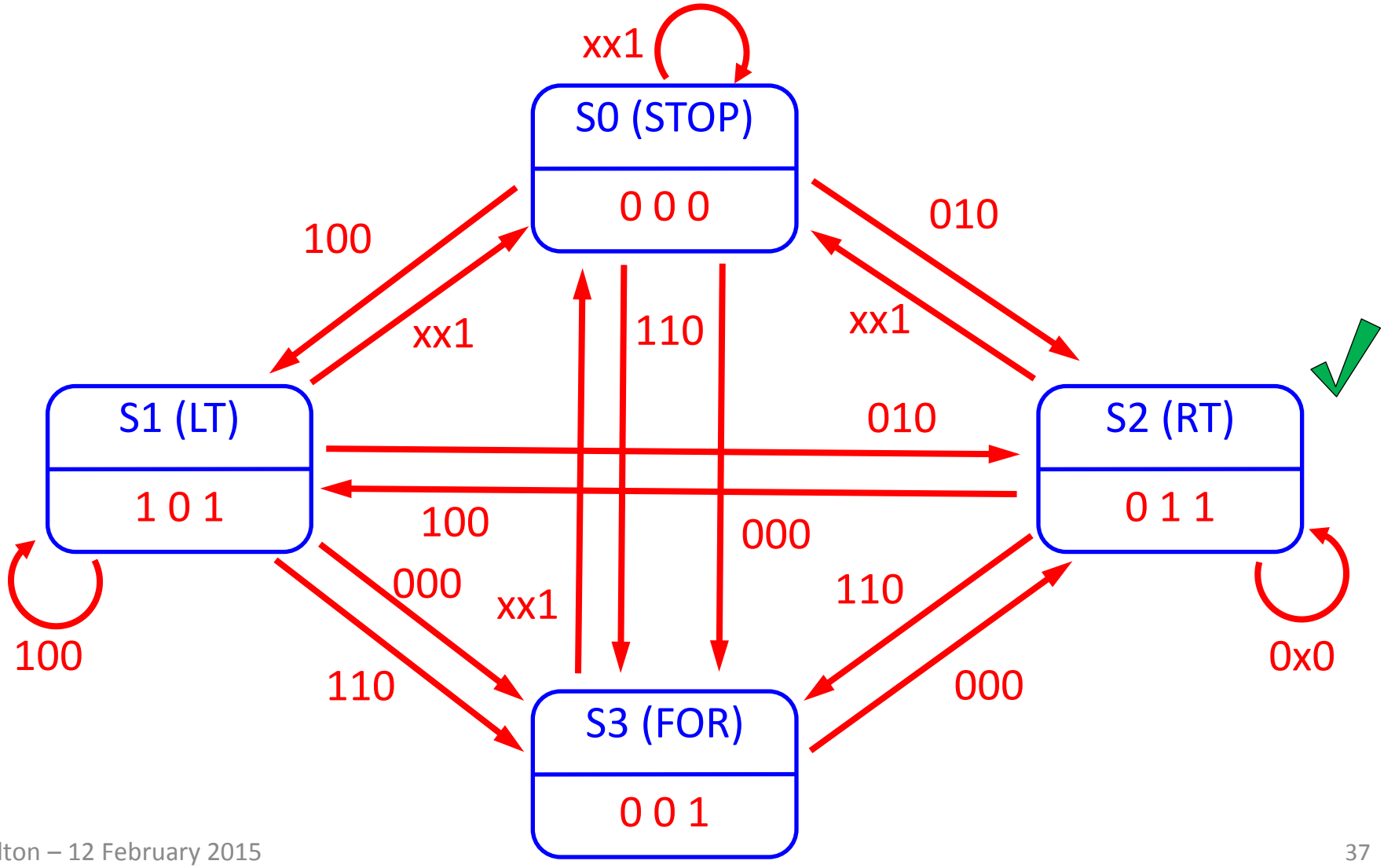


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

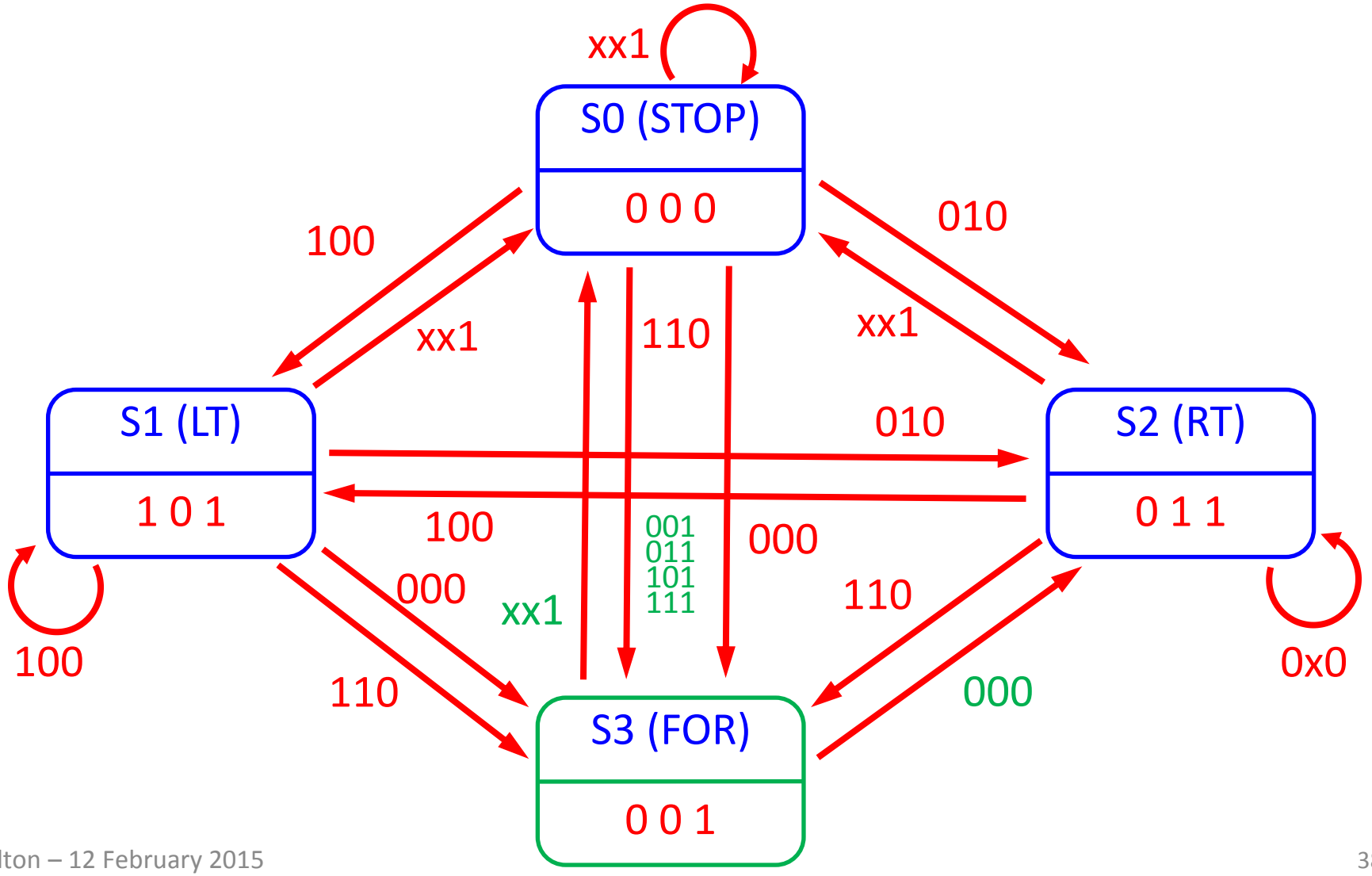


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

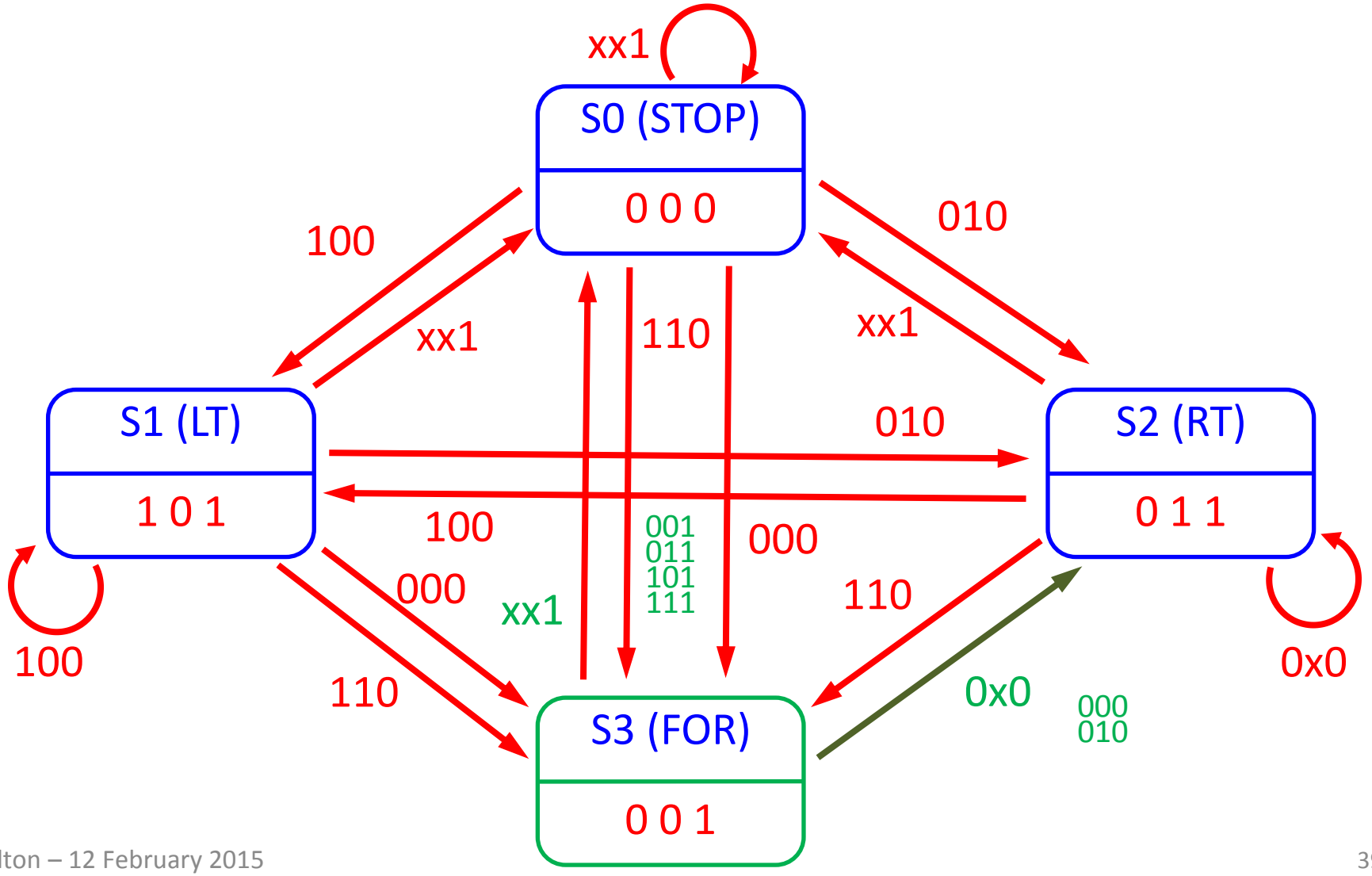


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

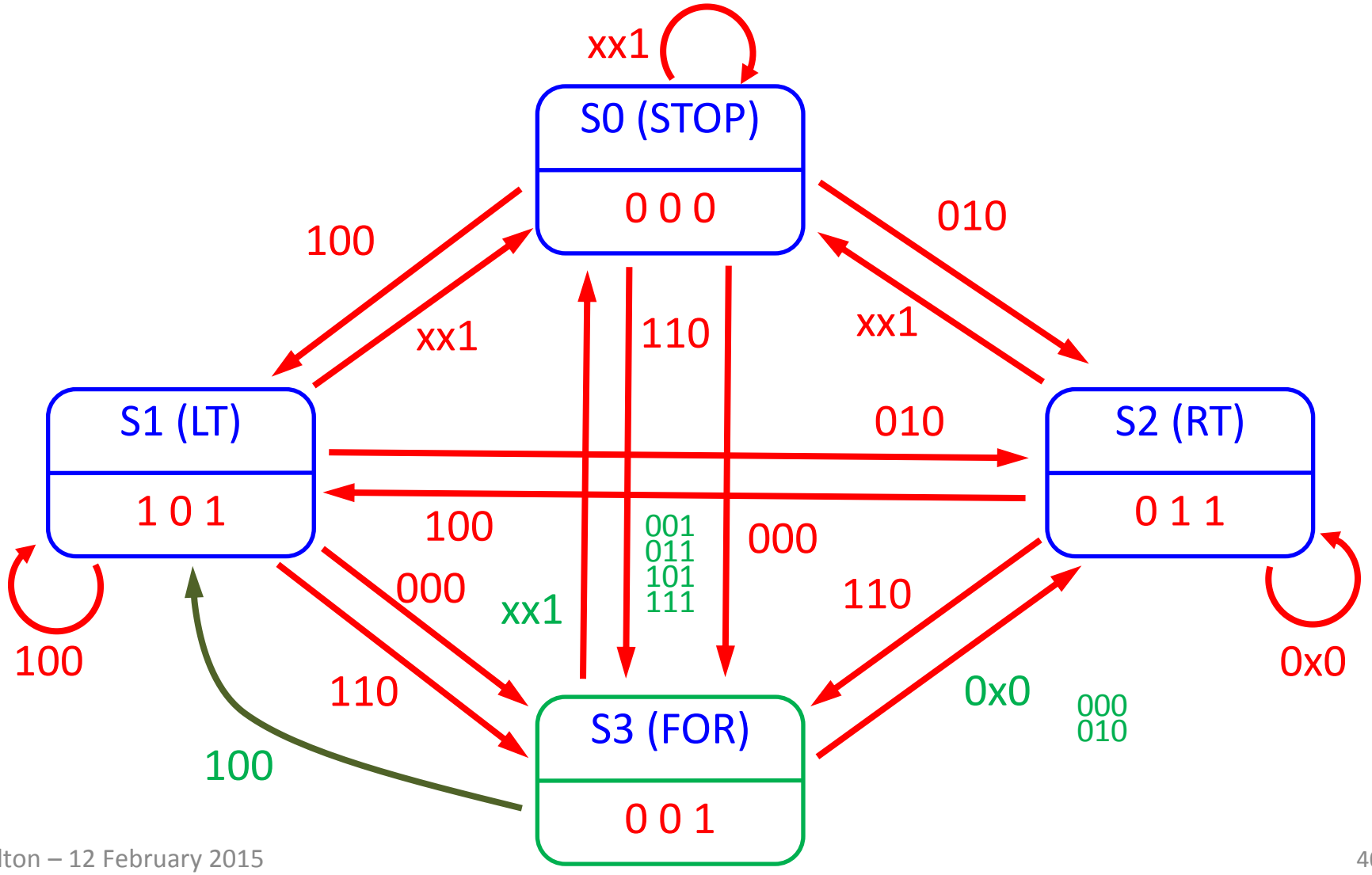


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

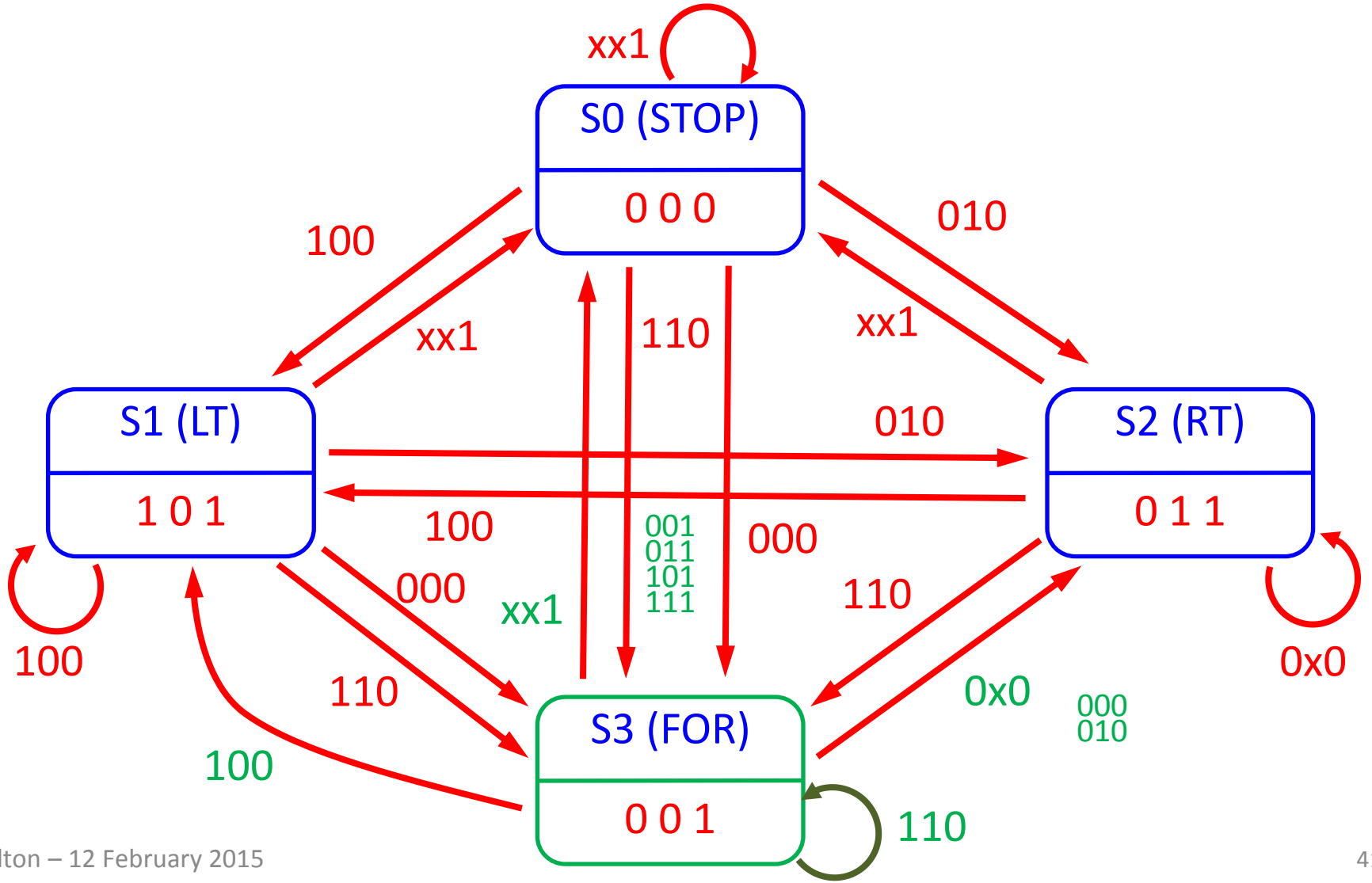


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

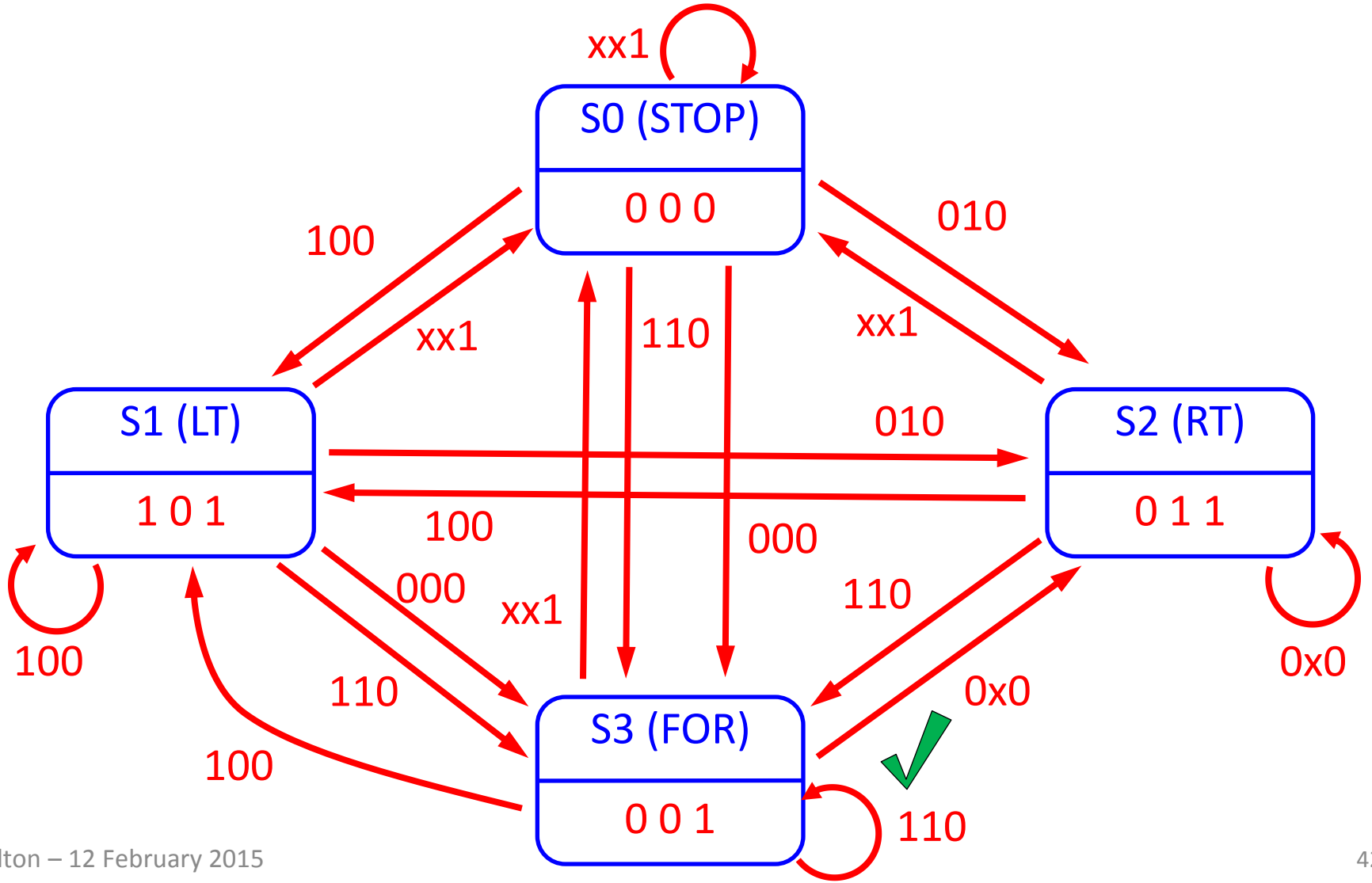


AUTONOMOUS OBJECT SEEKING

Inputs:
LS RS CL

Sx (xxx)
L R F

Sketch the state transition diagram for the FSM.

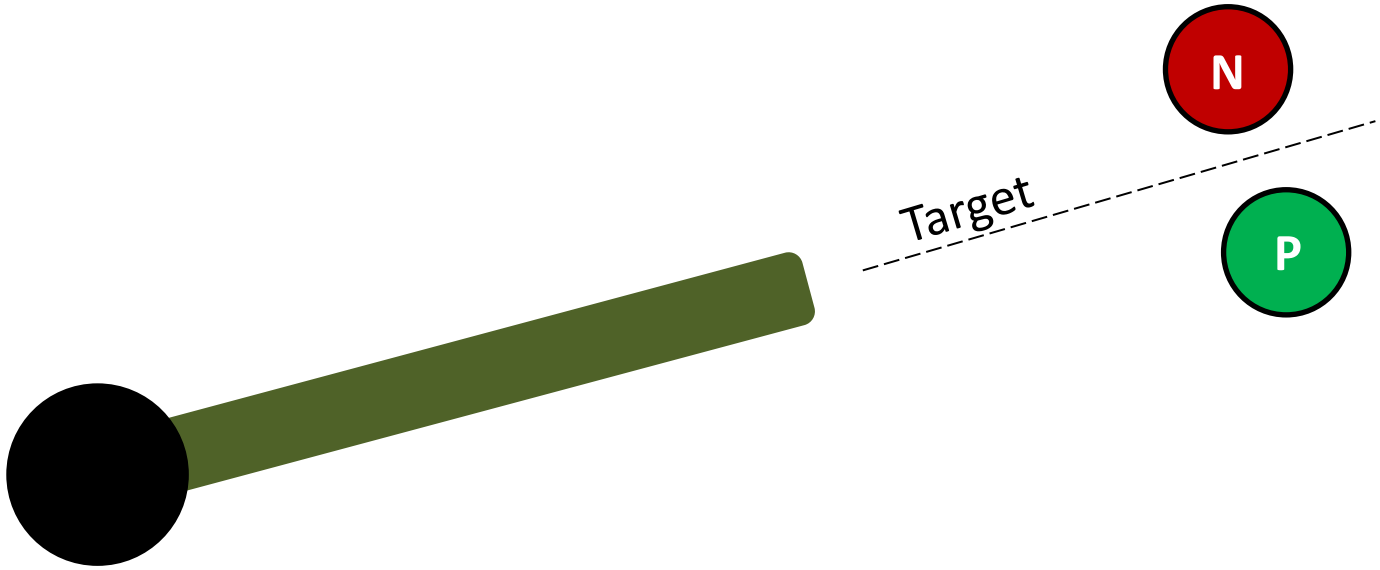


FINALLY, IMPLEMENT THE DESIGNED FSM ON ARDUINO

- Simulate operation using LEDs and switches
- Test operation with actual system

SIMPLE POSITION CONTROL

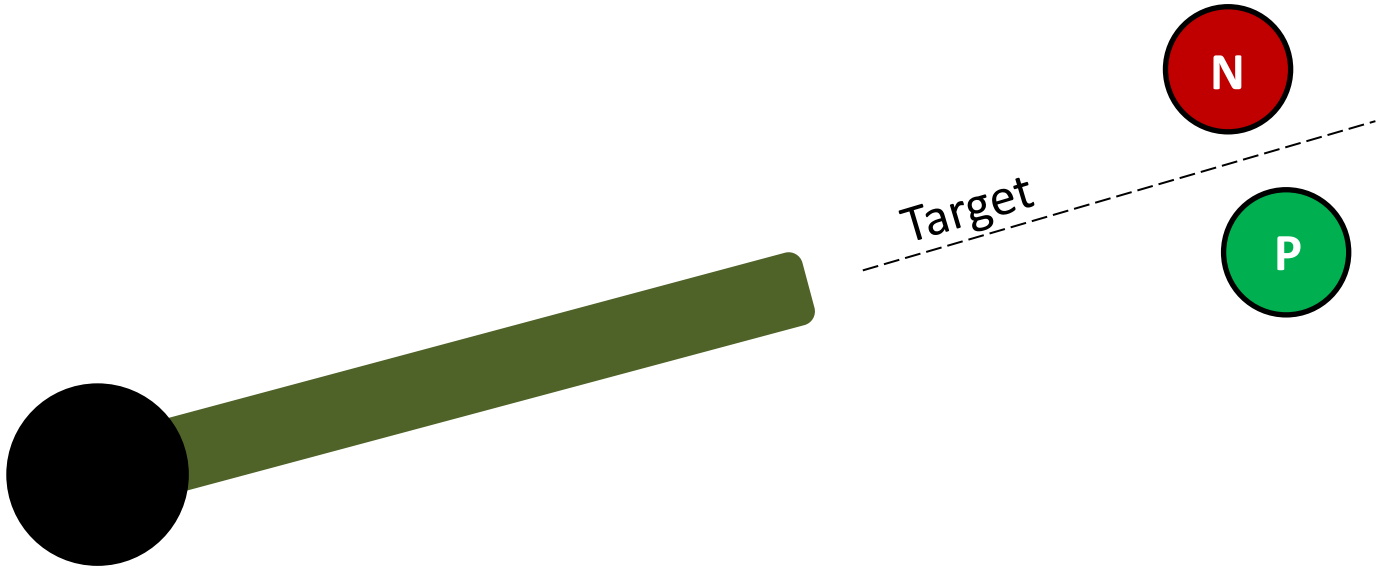
You are asked to design a controller (finite state machine) to control the motion of a robot arm. The machine has two inputs (P, N) and three outputs (Forward, Reverse, Slow).



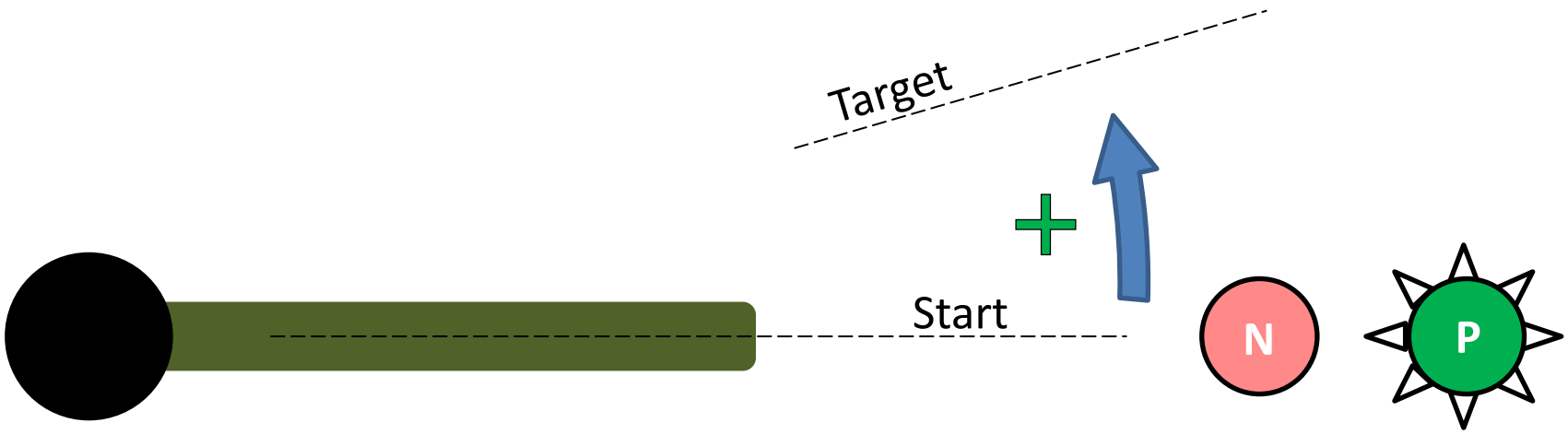
SIMPLE POSITION CONTROL

The inputs (P, N) are described as follows:

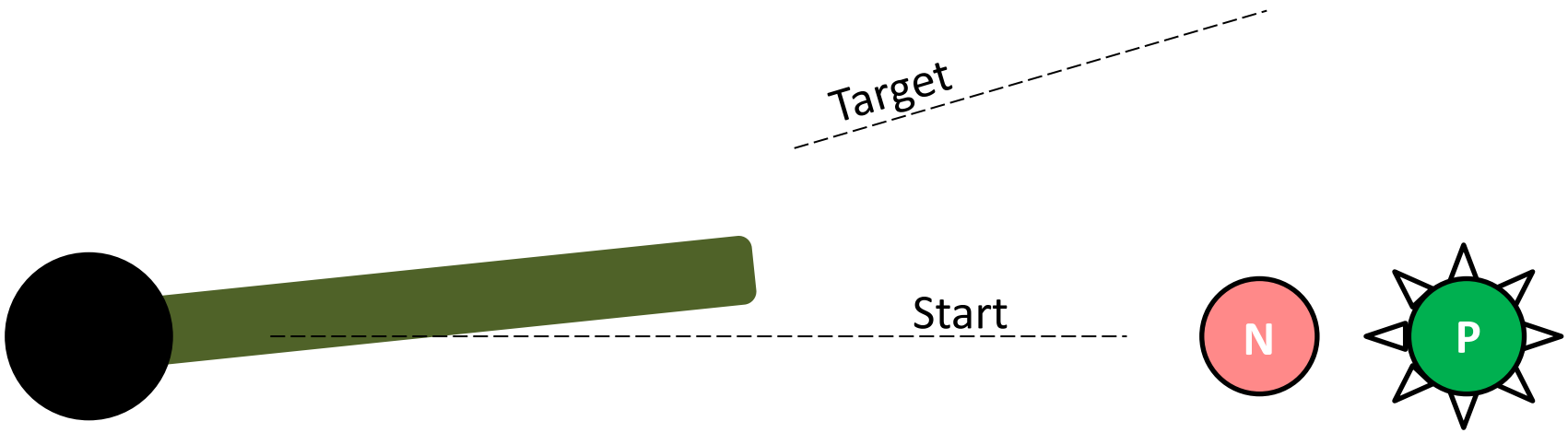
- When the robot position is less than the desired position; P=1, N=0.
- When the robot position is larger than the desired position: P=0, N=1.
- When the robot position is identical to the desired position: P=0,N=0.
- When the robot has moved at least 7/8 of the way to the desired position, moving either forward or backward: P=1,N=1.



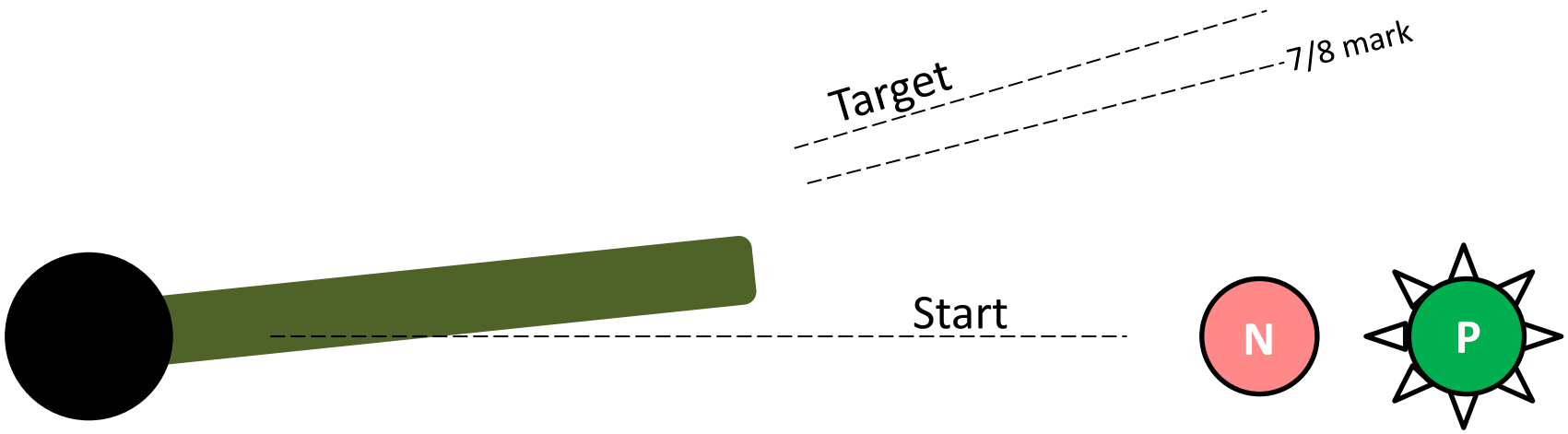
SIMPLE POSITION CONTROL



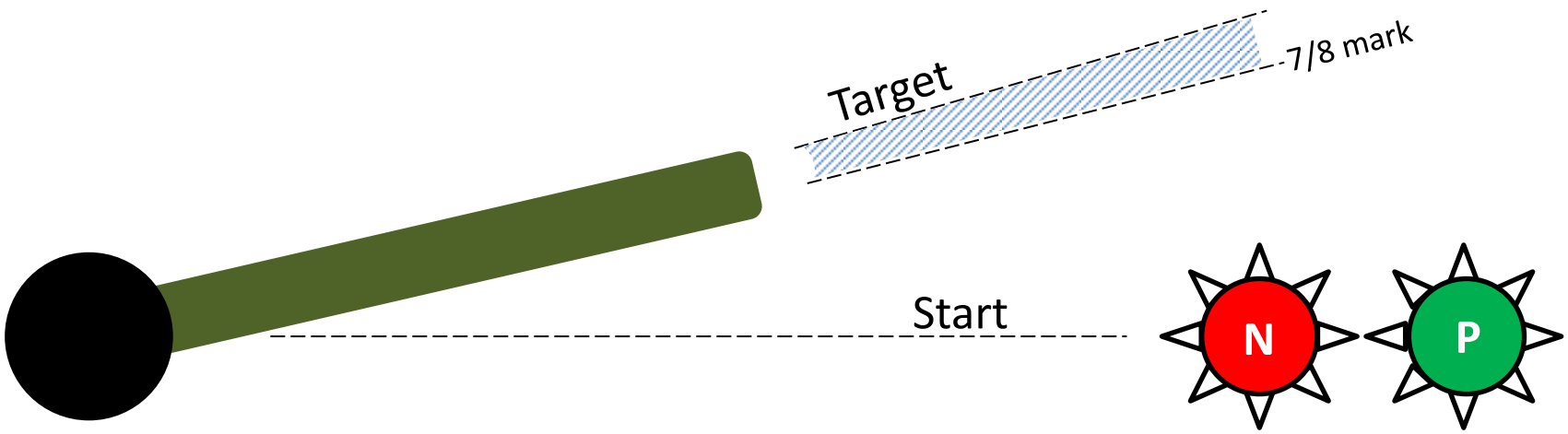
SIMPLE POSITION CONTROL



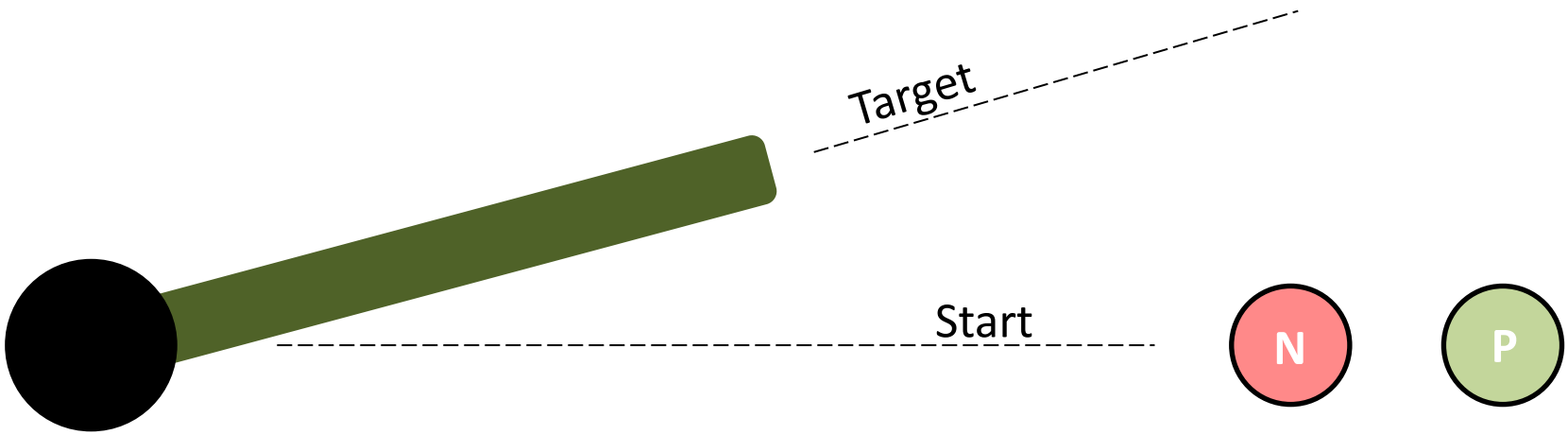
SIMPLE POSITION CONTROL



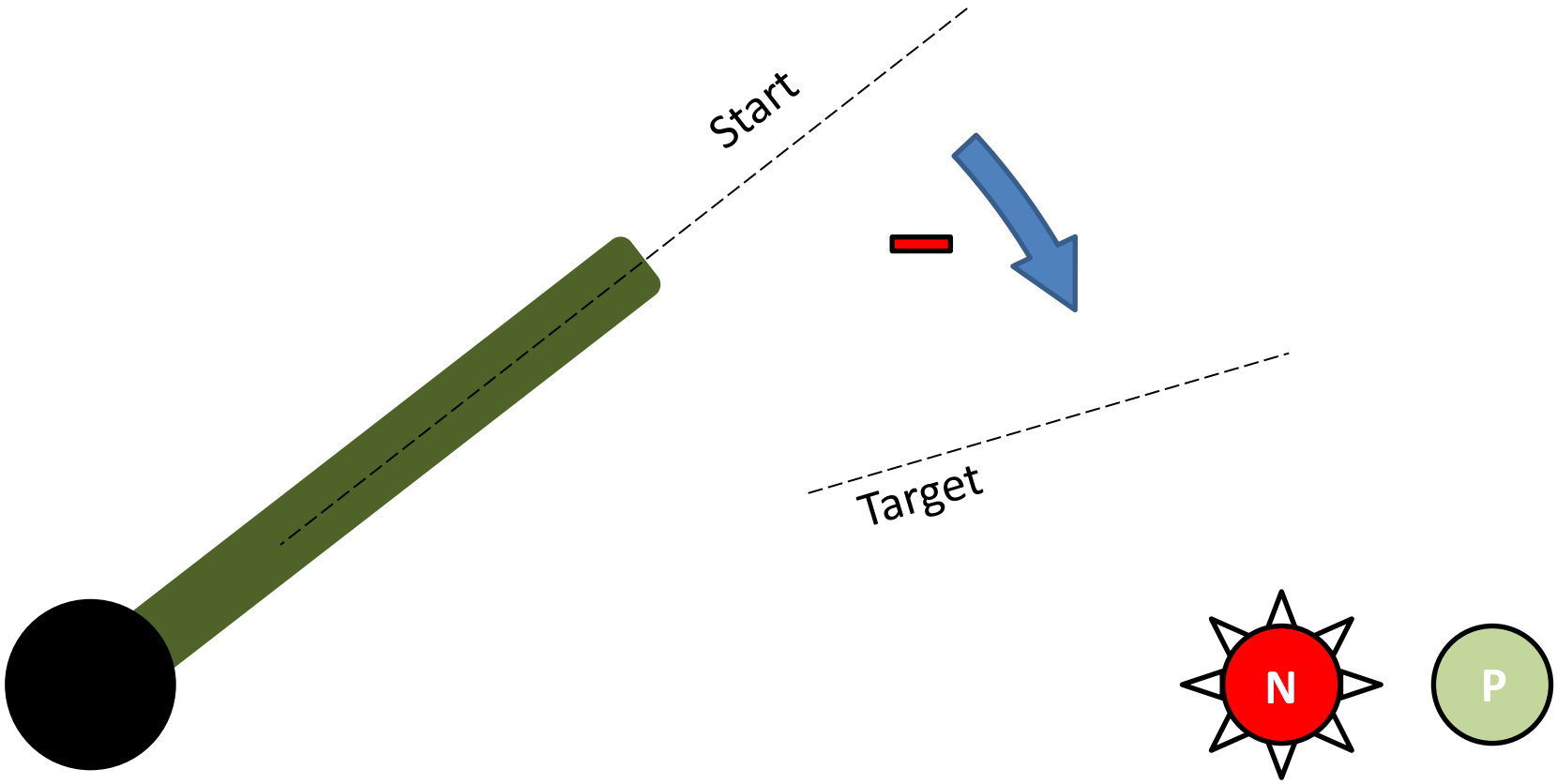
SIMPLE POSITION CONTROL



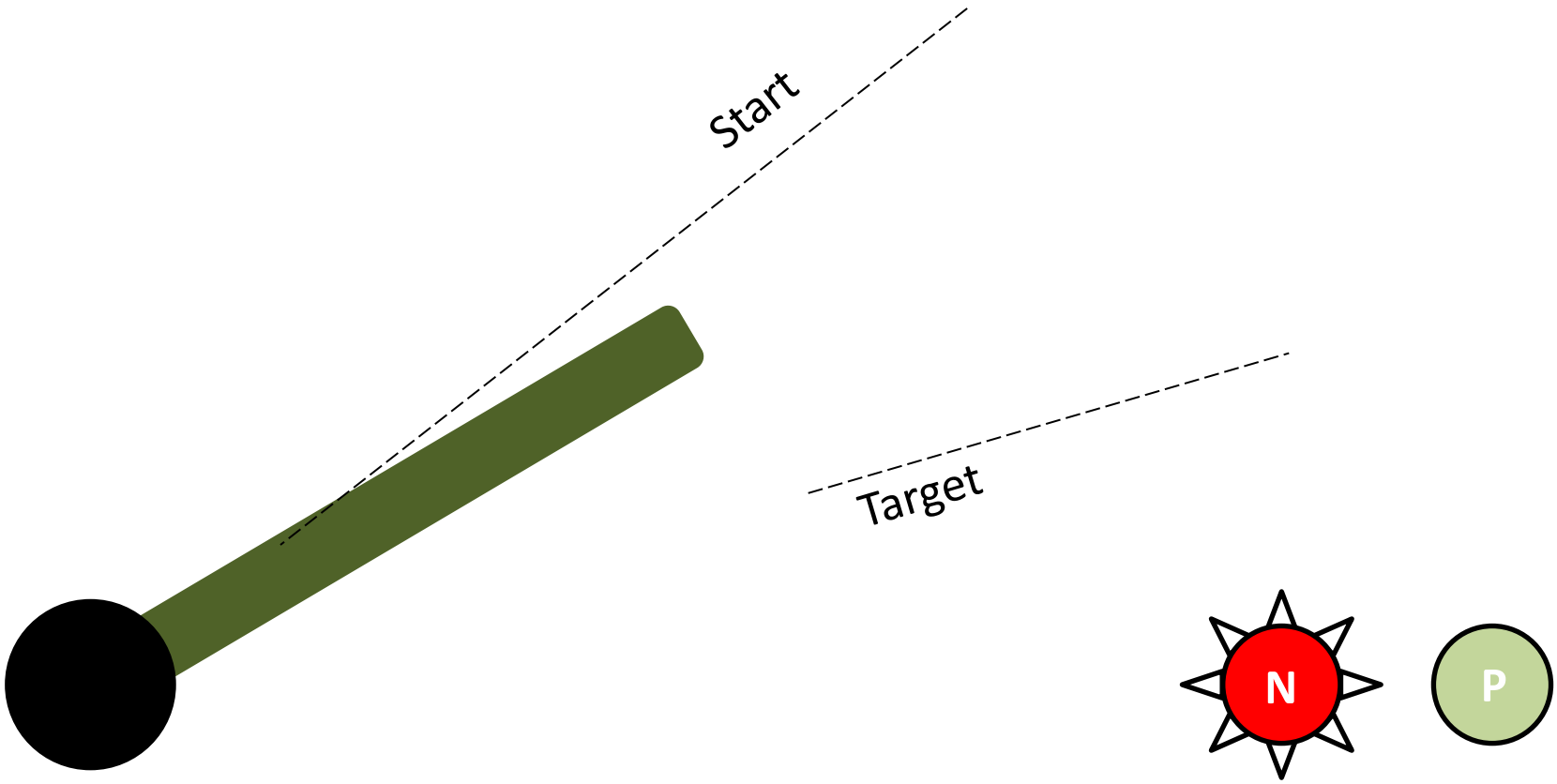
SIMPLE POSITION CONTROL



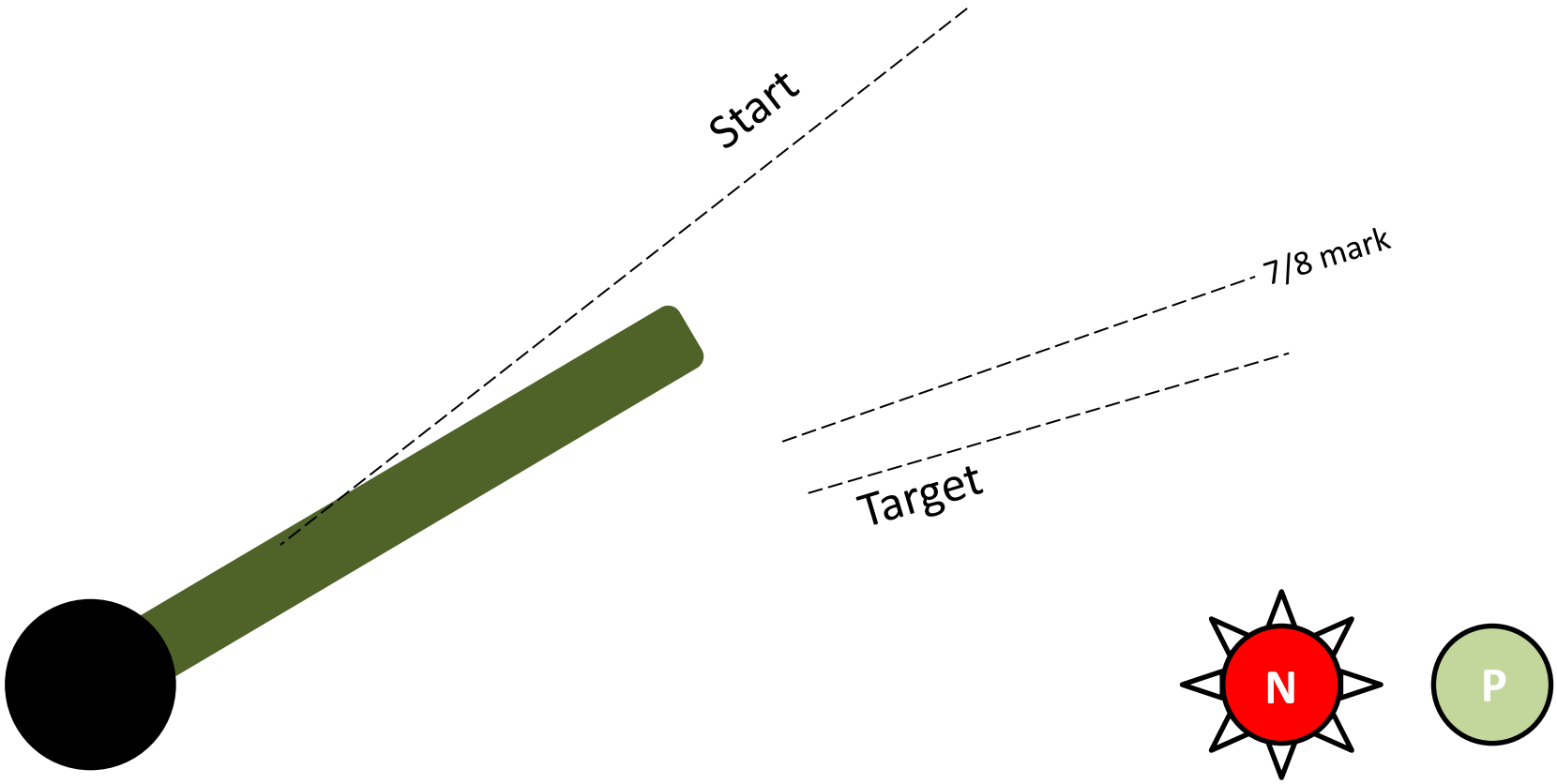
SIMPLE POSITION CONTROL



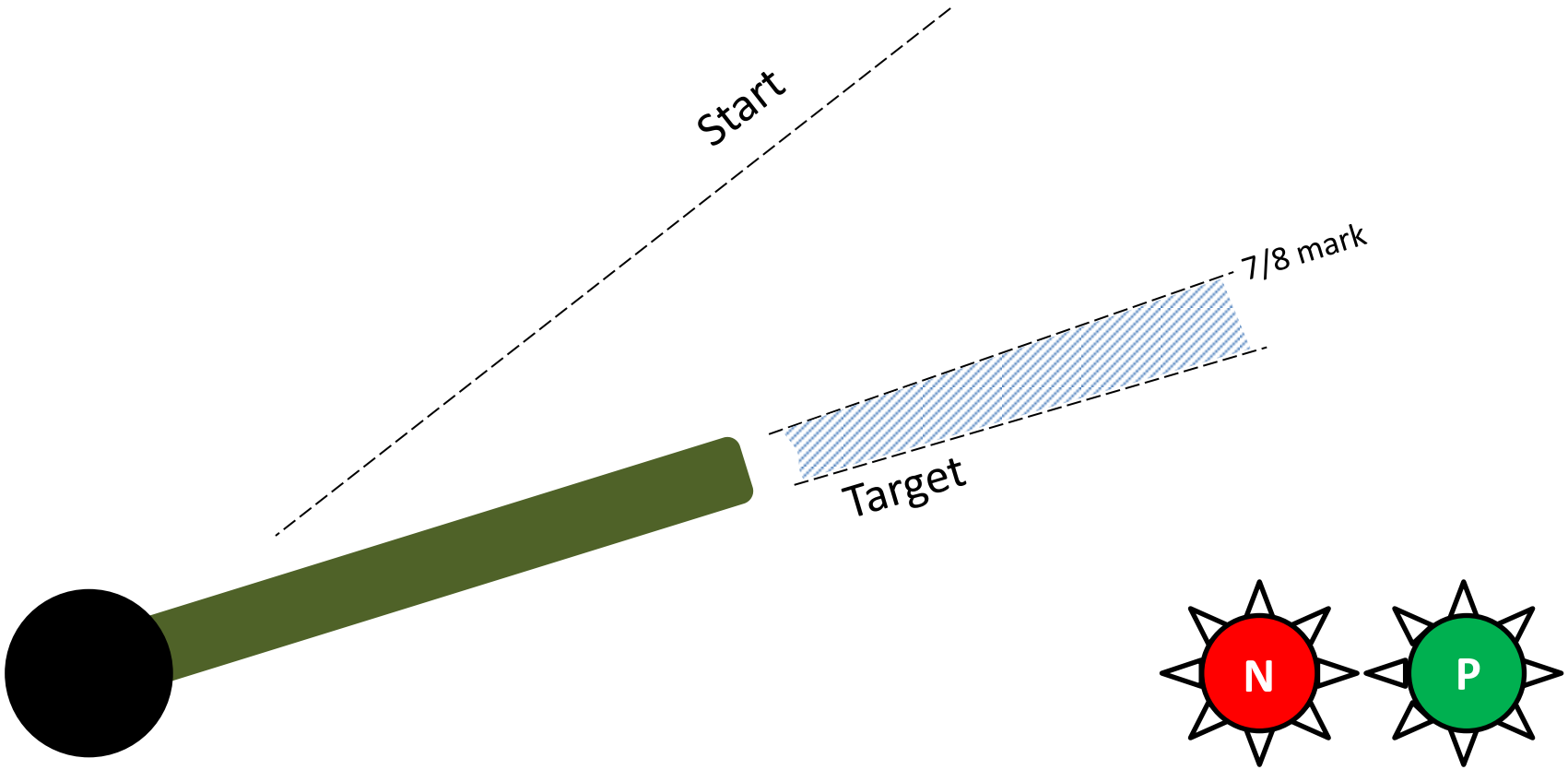
SIMPLE POSITION CONTROL



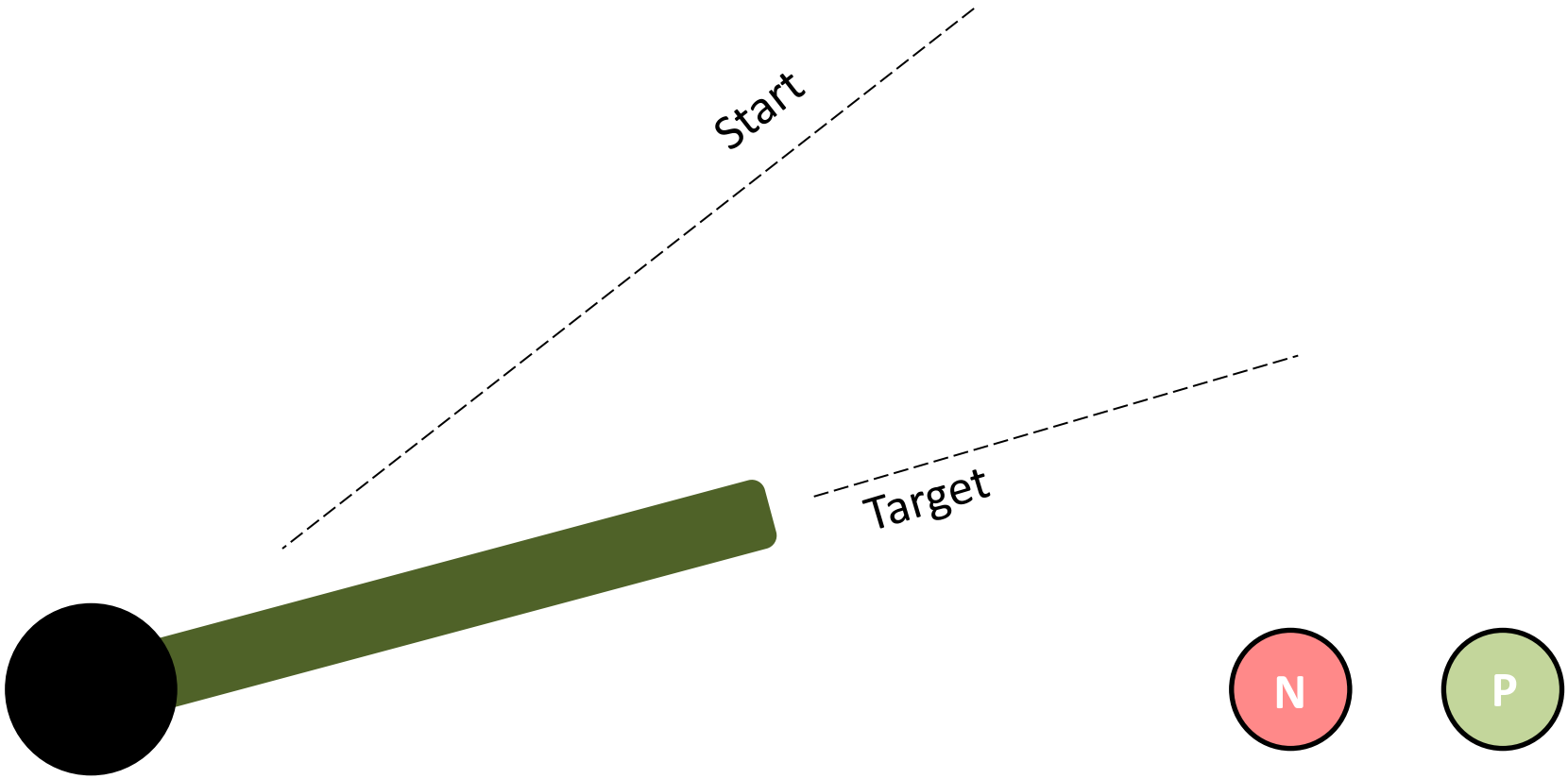
SIMPLE POSITION CONTROL



SIMPLE POSITION CONTROL



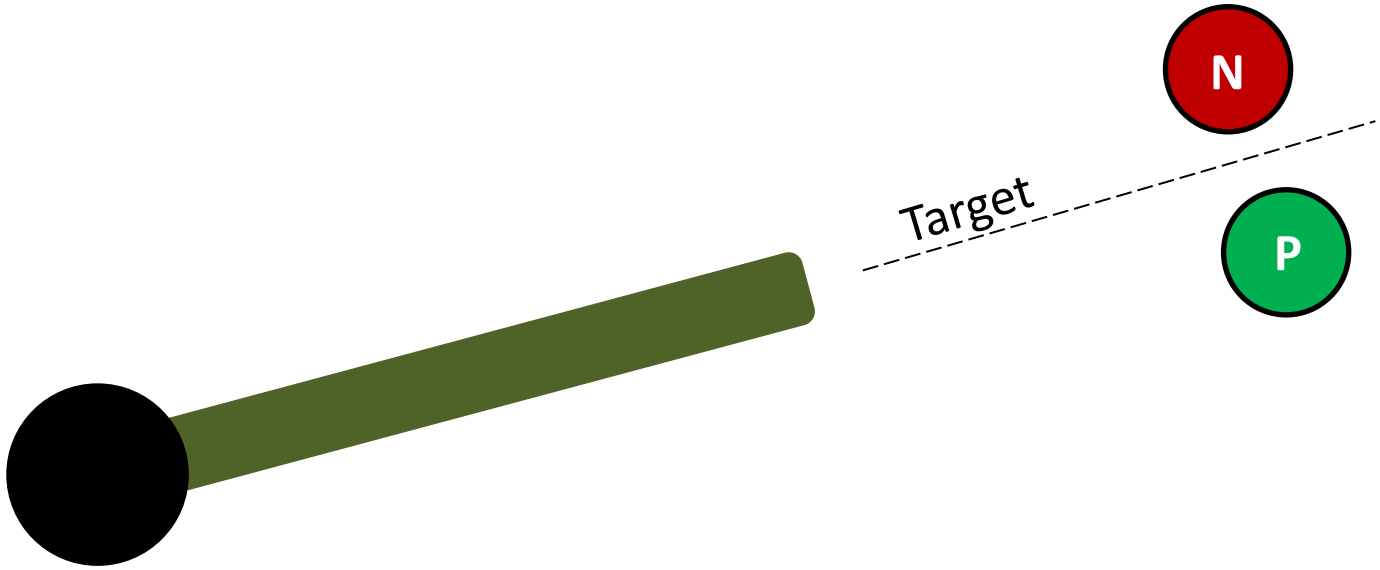
SIMPLE POSITION CONTROL



SIMPLE POSITION CONTROL

The outputs are:

- FORWARD (F): Robot moves forward when this output is asserted.
- REVERSE (R): Robot moves backward when this output is asserted.
- SLOW (S): Brake is applied to slow down the robot.



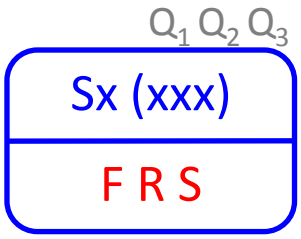
SIMPLE POSITION CONTROL

The operation of the machine is:

- The system starts at a stop state where all outputs are zero.
- If the current position of the robot is less than the desired position, FORWARD should be asserted (F = 1).
- If the current position of the robot is larger than the desired position, REVERSE should be asserted (R = 1).
- If the position is correct, stop the robot by setting all outputs to zero.
- If the robot has moved 7/8 the way to the target position, the SLOW output should be asserted (S =1) to brake the motion of the arm. FORWARD or REVERSE signals must remain high while SLOW is asserted for movement to continue.

SIMPLE POSITION CONTROL

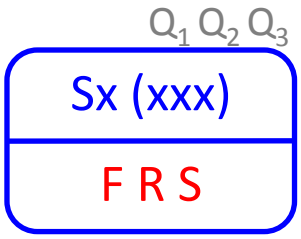
Inputs:
P N



Sketch the state transition diagram for the FSM.

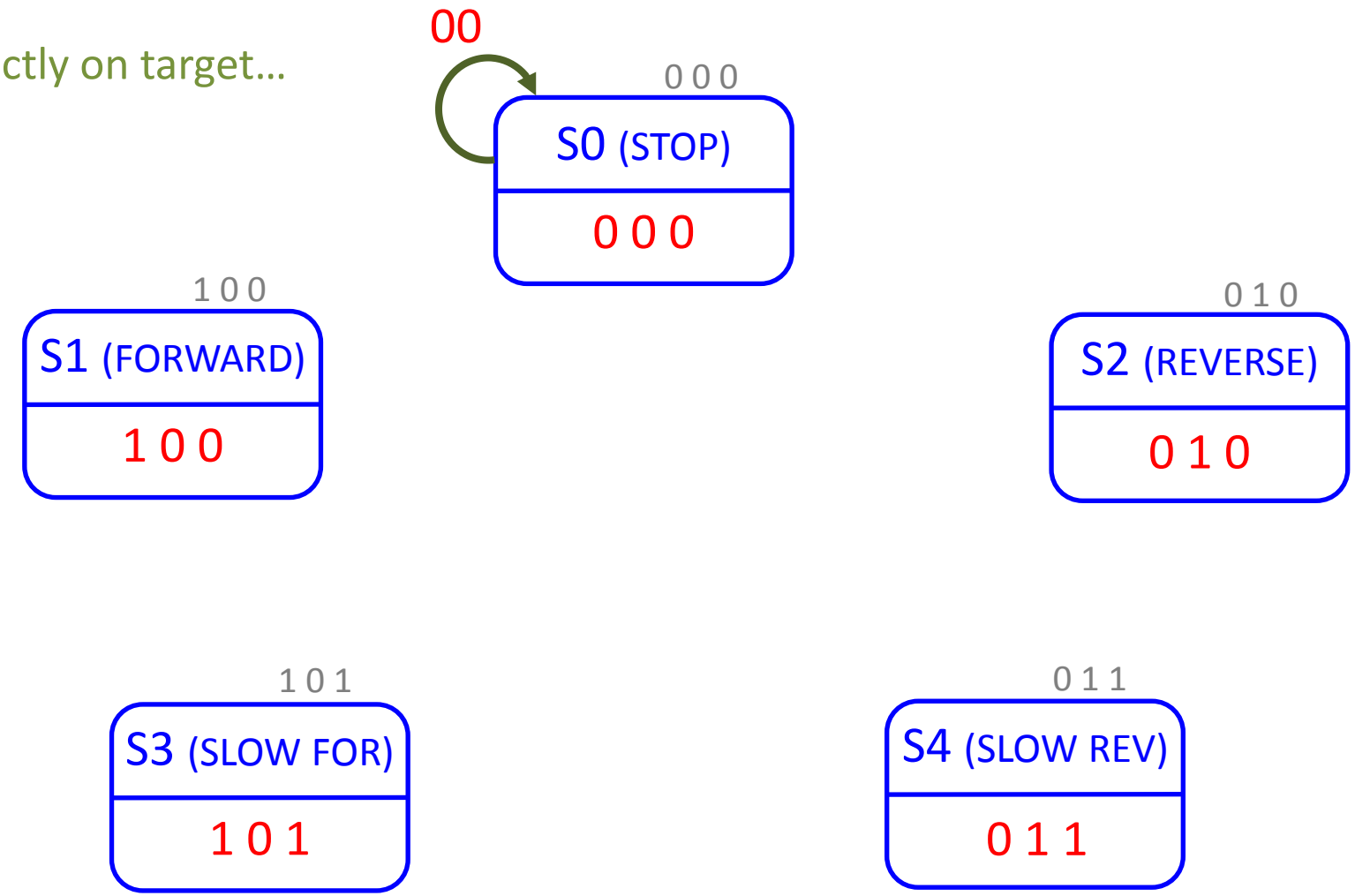
SIMPLE POSITION CONTROL

Inputs:
P N



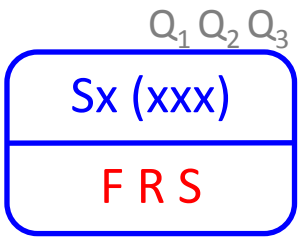
Sketch the state transition diagram for the FSM.

Exactly on target…

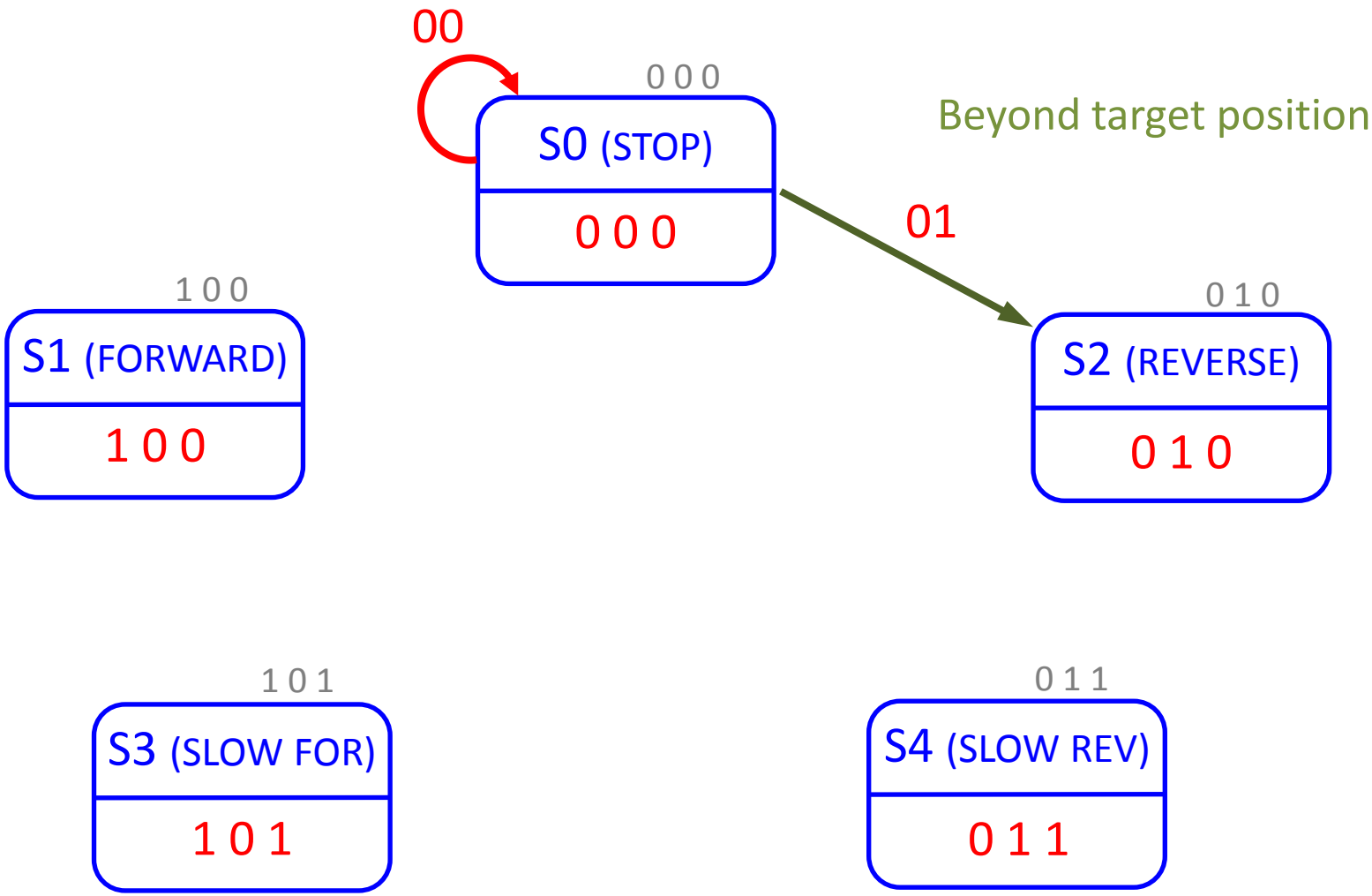


SIMPLE POSITION CONTROL

Inputs:
P N



Sketch the state transition diagram for the FSM.

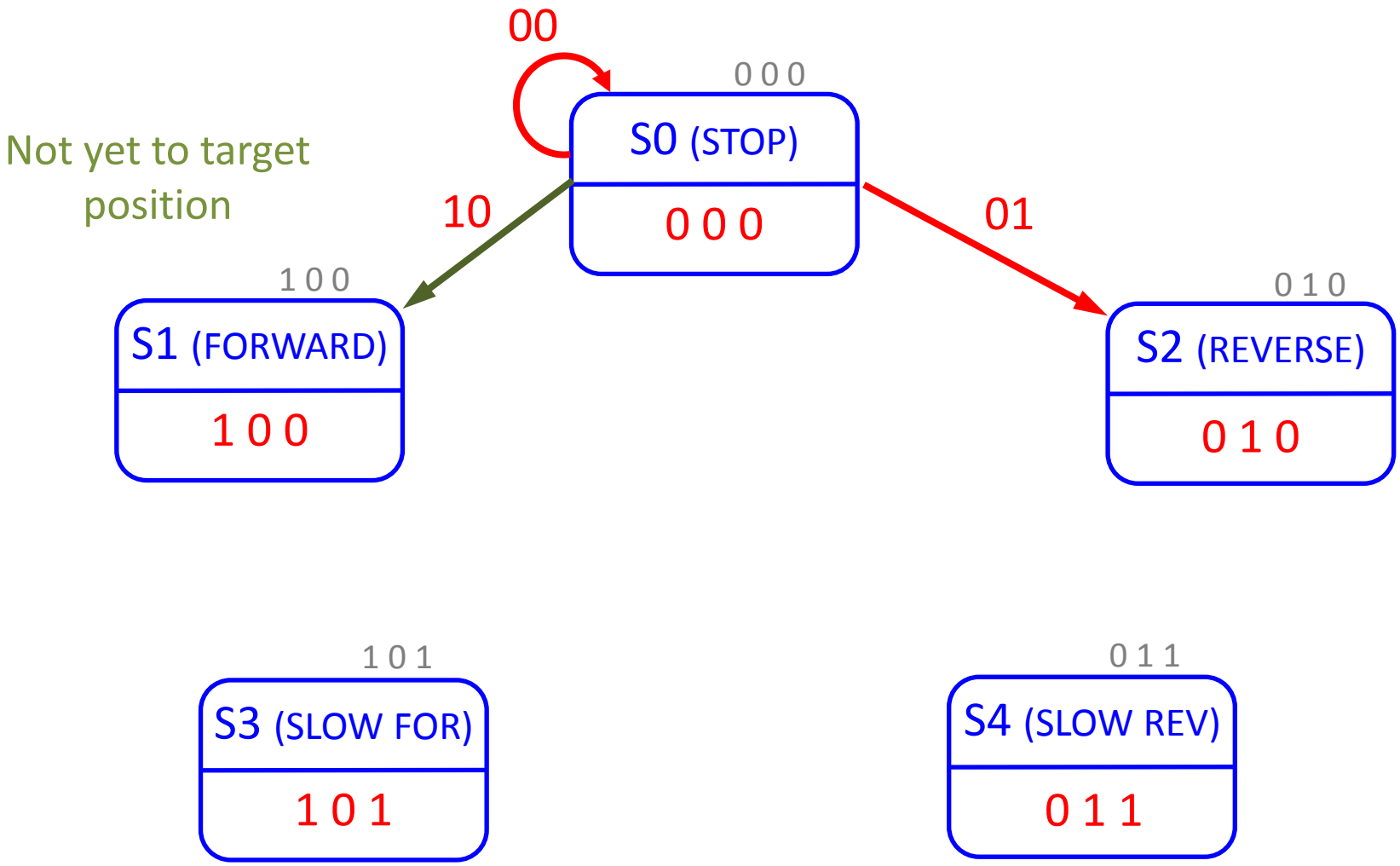


SIMPLE POSITION CONTROL

Inputs:
P N

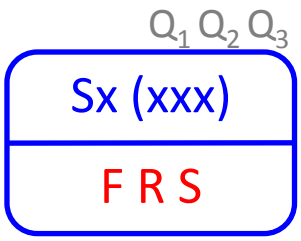
$Q_1\ Q_2\ Q_3$
Sx (xxx)
F R S

Sketch the state transition diagram for the FSM.



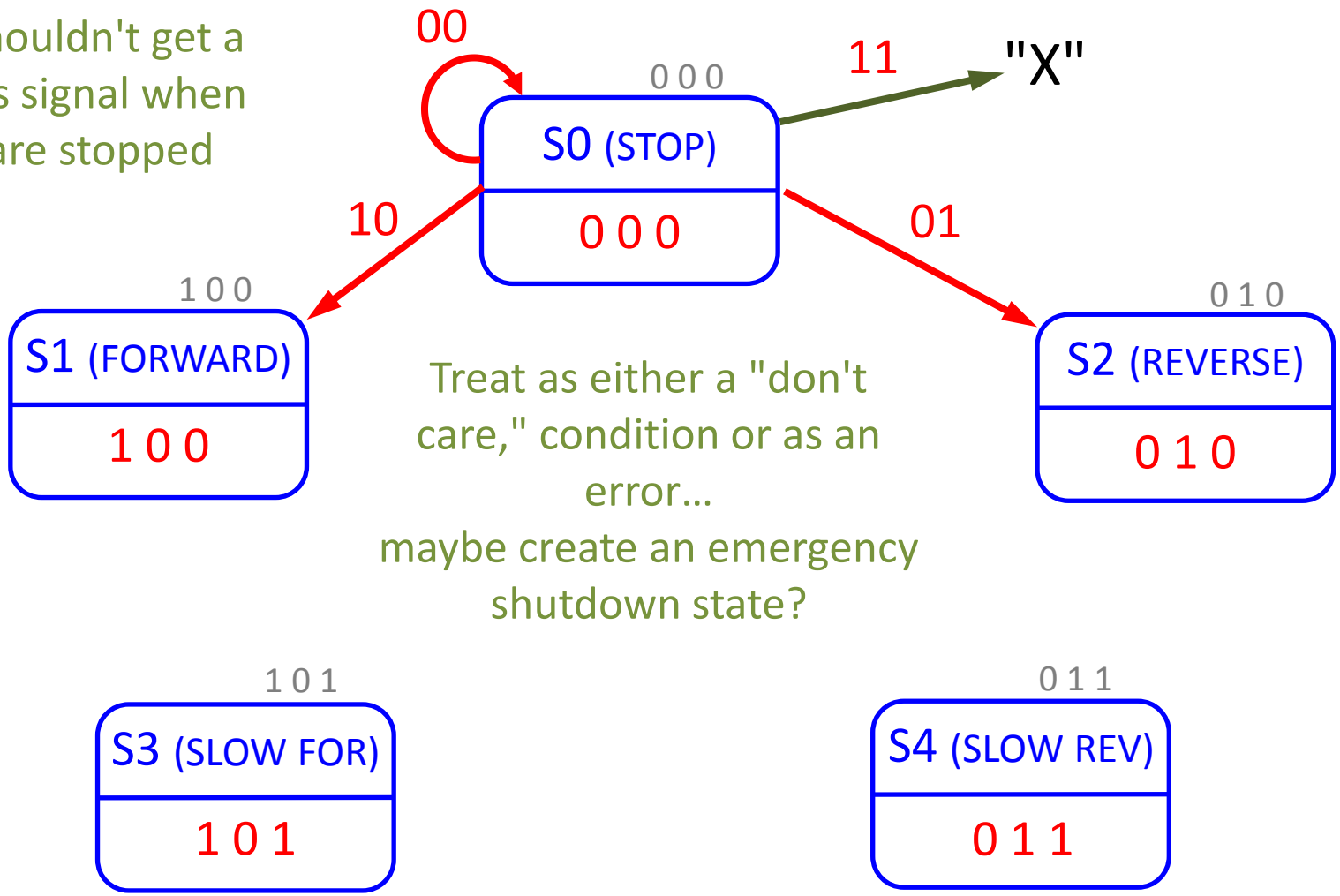
SIMPLE POSITION CONTROL

Inputs:
P N



Sketch the state transition diagram for the FSM.

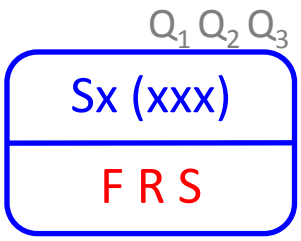
We shouldn't get a 7/8ths signal when we are stopped



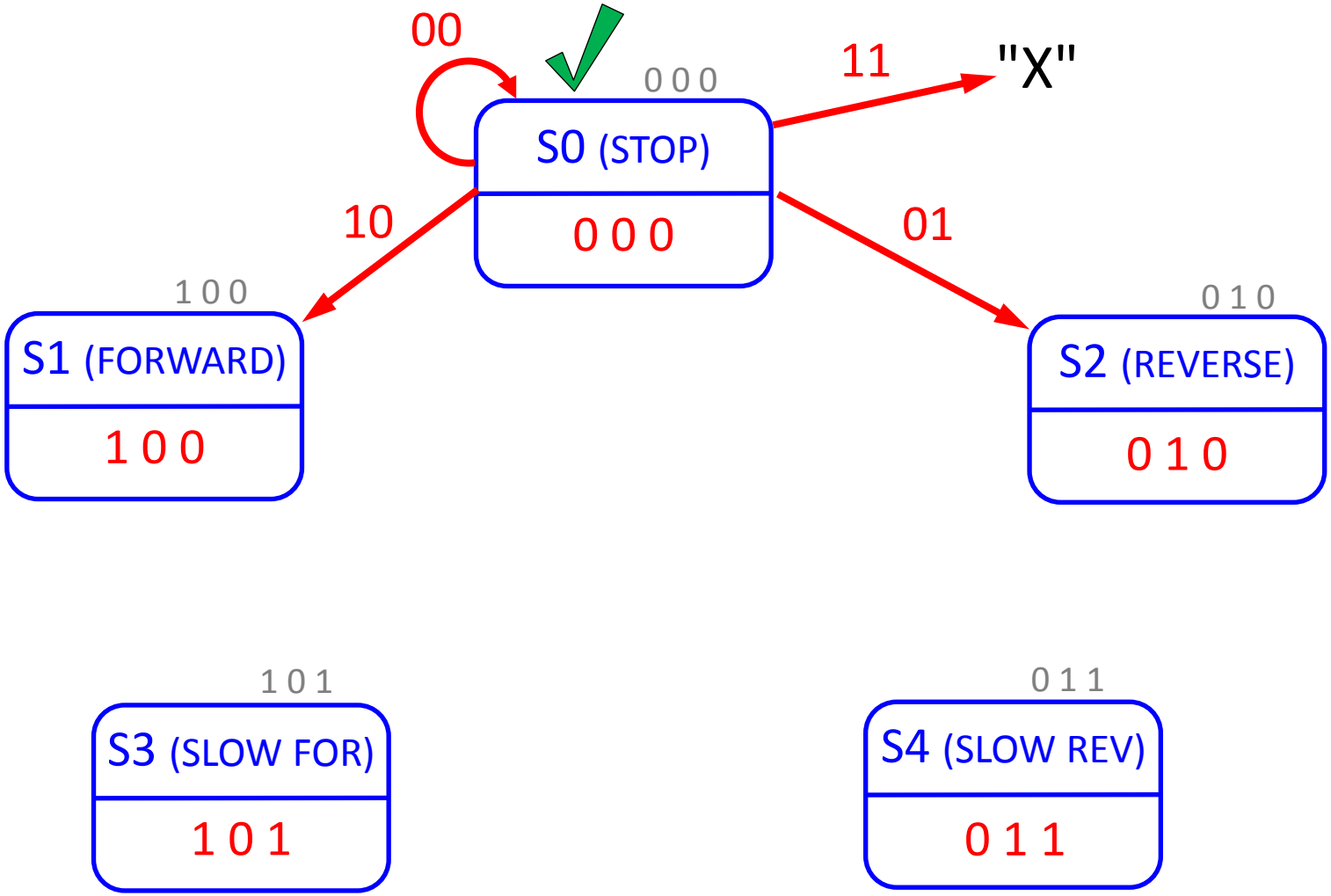
Treat as either a "don't care," condition or as an error…
maybe create an emergency shutdown state?

SIMPLE POSITION CONTROL

Inputs:
P N

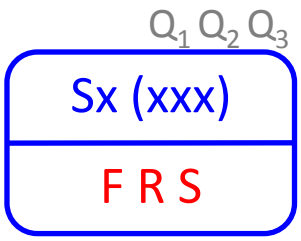


Sketch the state transition diagram for the FSM.

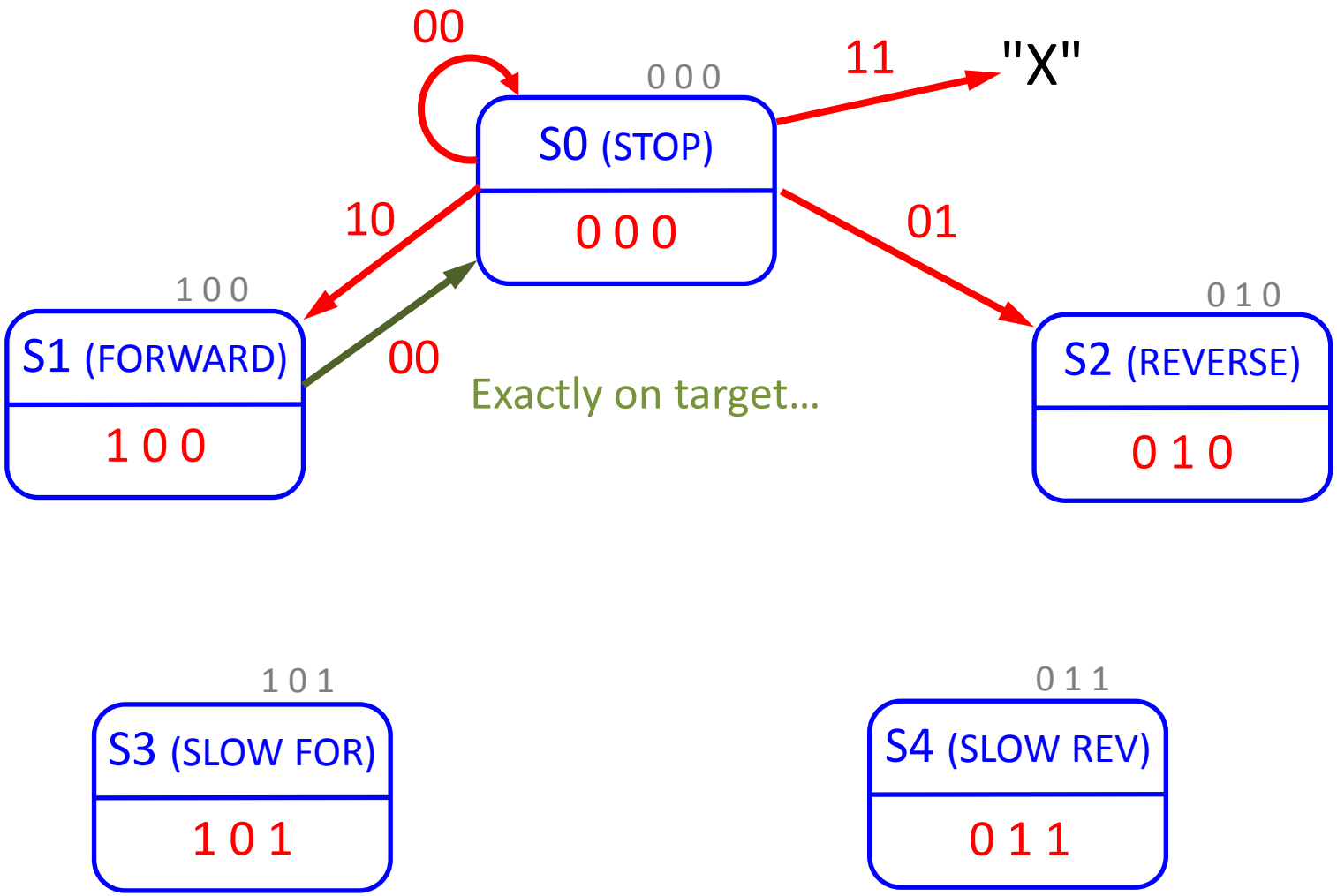


SIMPLE POSITION CONTROL

Inputs:
P N

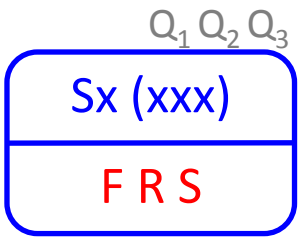


Sketch the state transition diagram for the FSM.

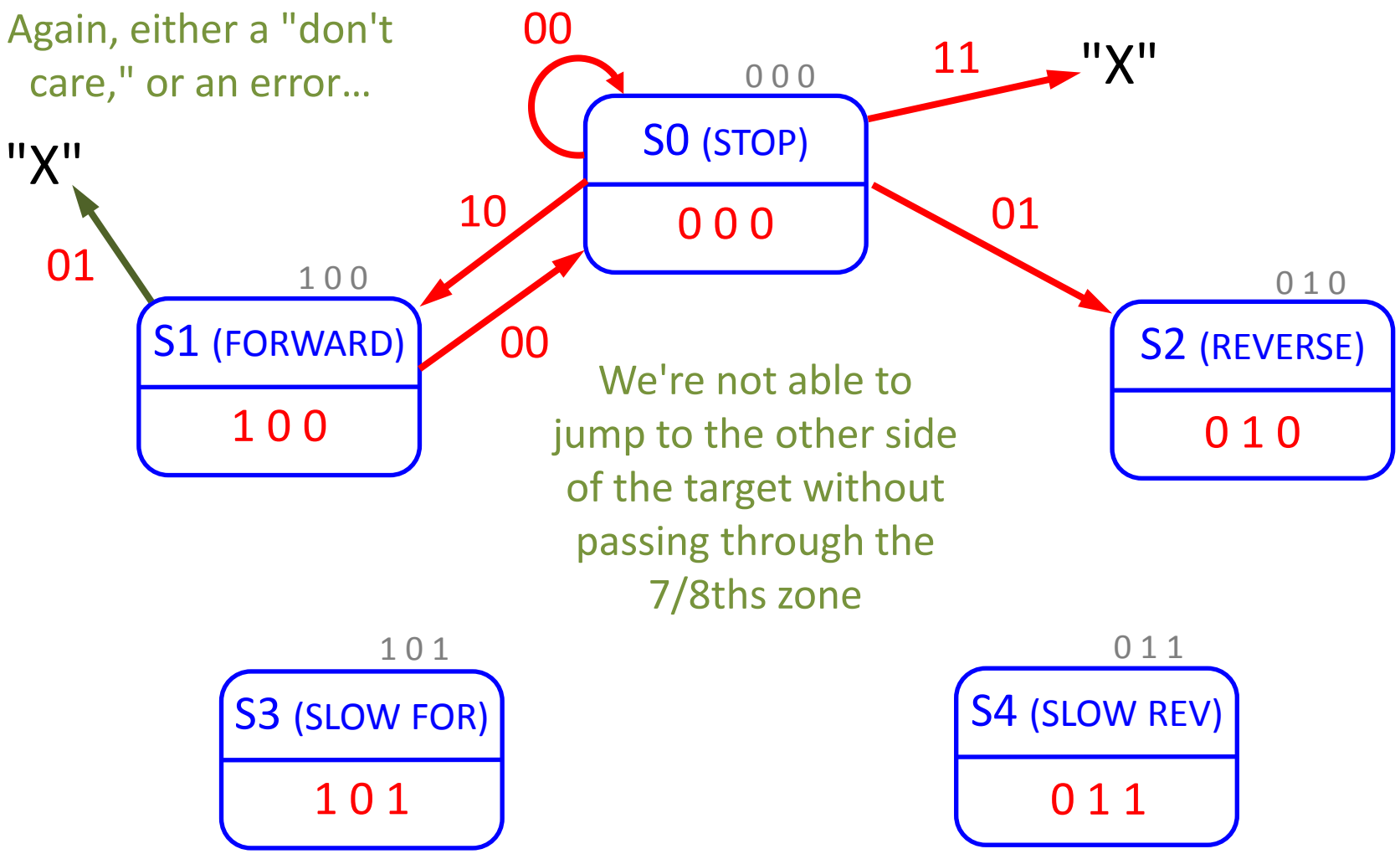


SIMPLE POSITION CONTROL

Inputs:
P N

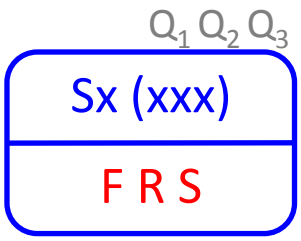


Sketch the state transition diagram for the FSM.

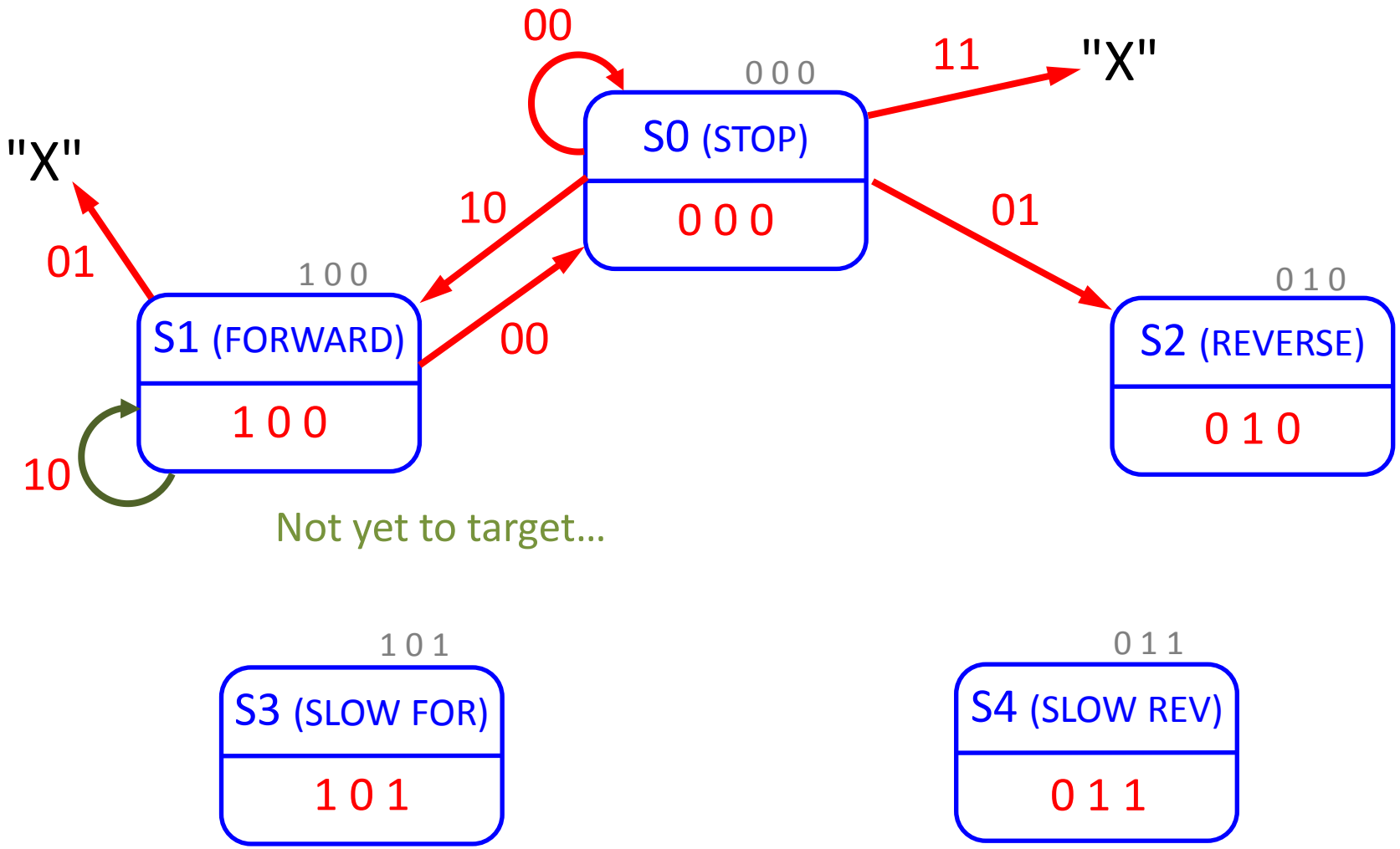


SIMPLE POSITION CONTROL

Inputs:
P N

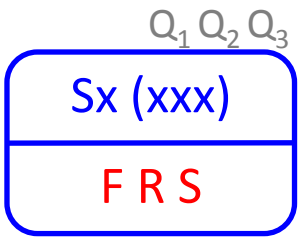


Sketch the state transition diagram for the FSM.

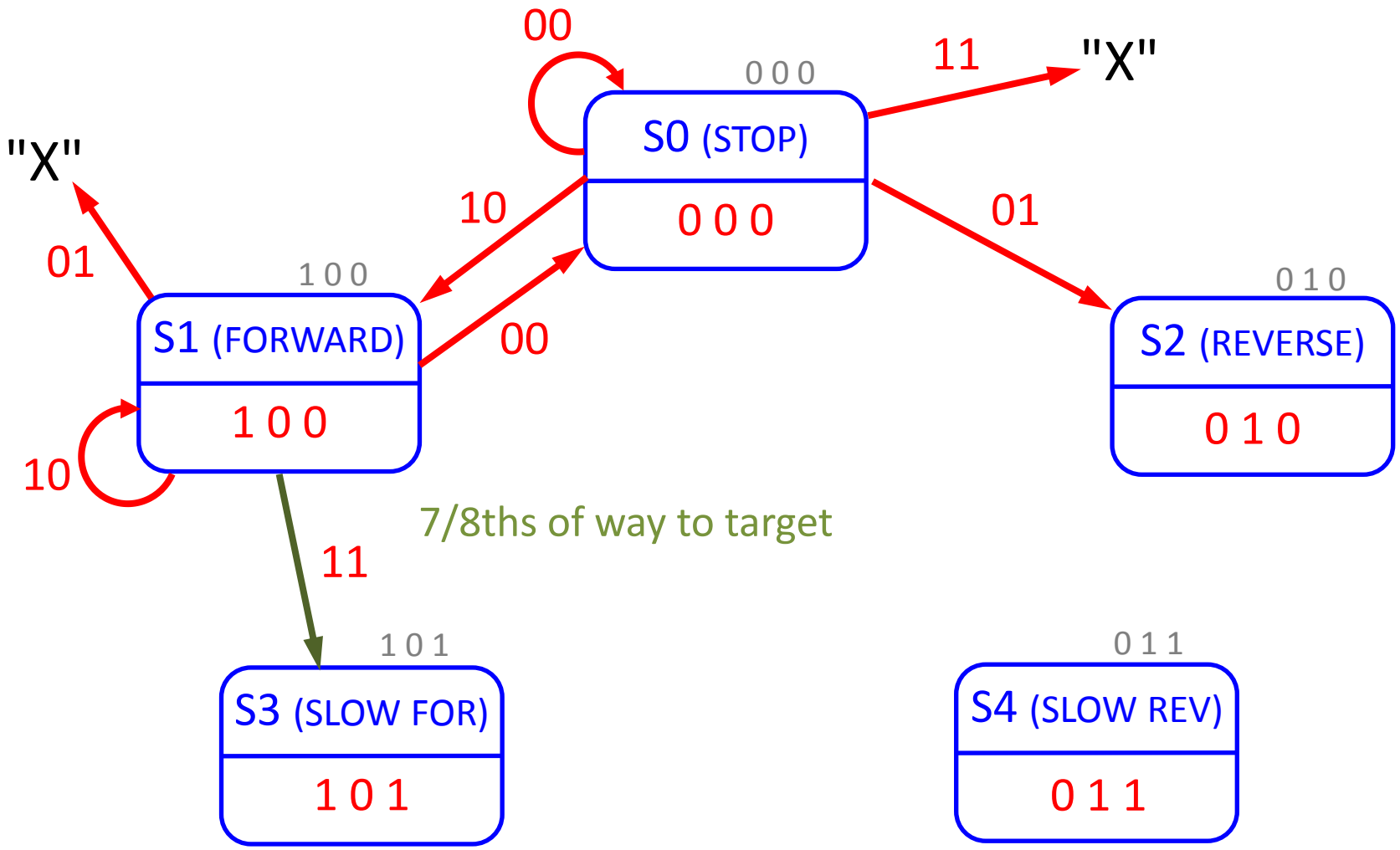


SIMPLE POSITION CONTROL

Inputs:
P N

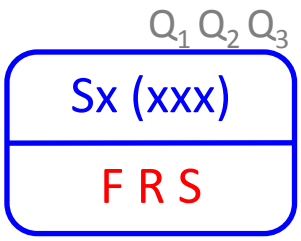


Sketch the state transition diagram for the FSM.

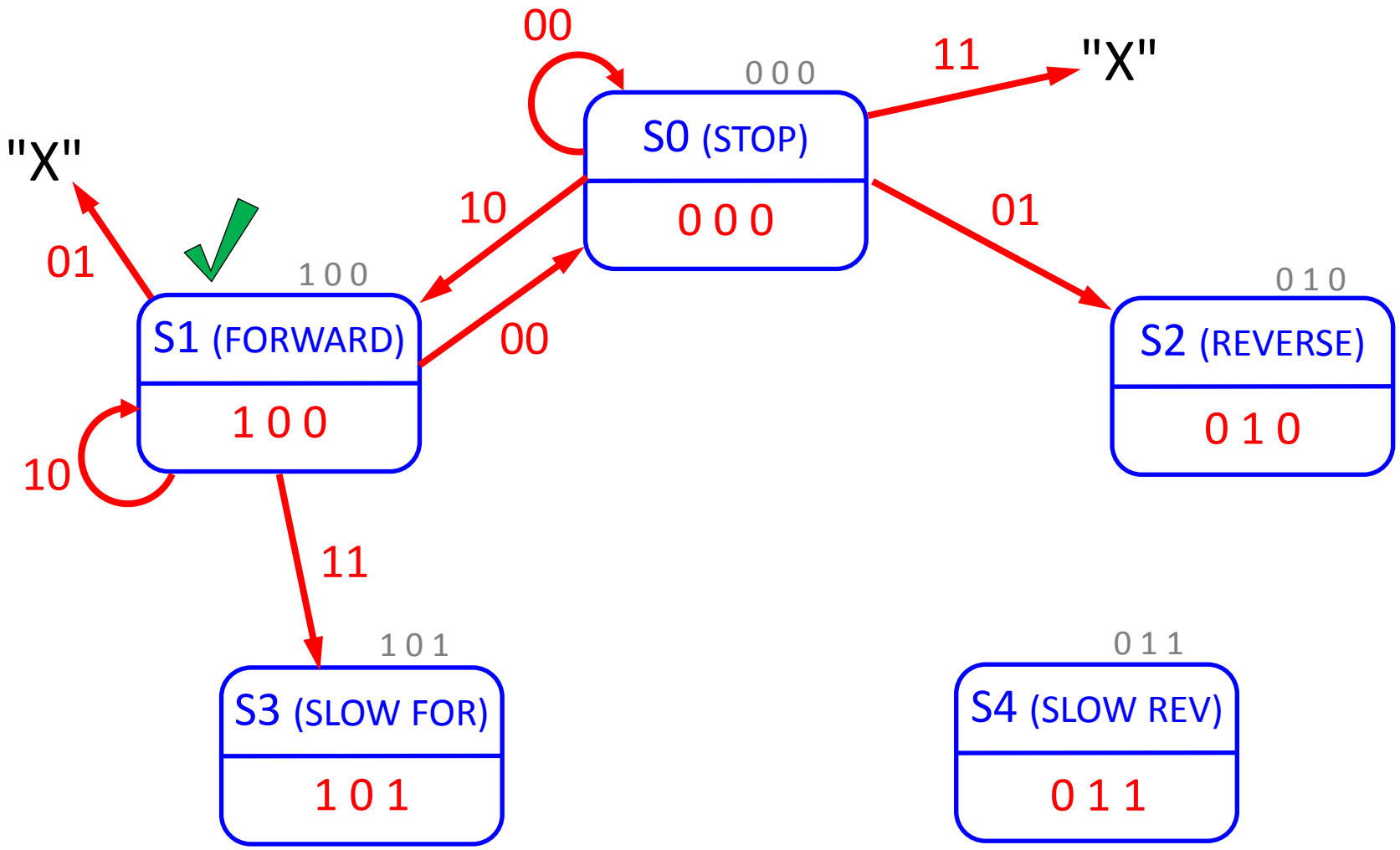


SIMPLE POSITION CONTROL

Inputs:
P N

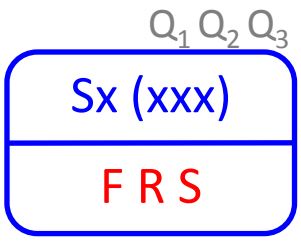


Sketch the state transition diagram for the FSM.

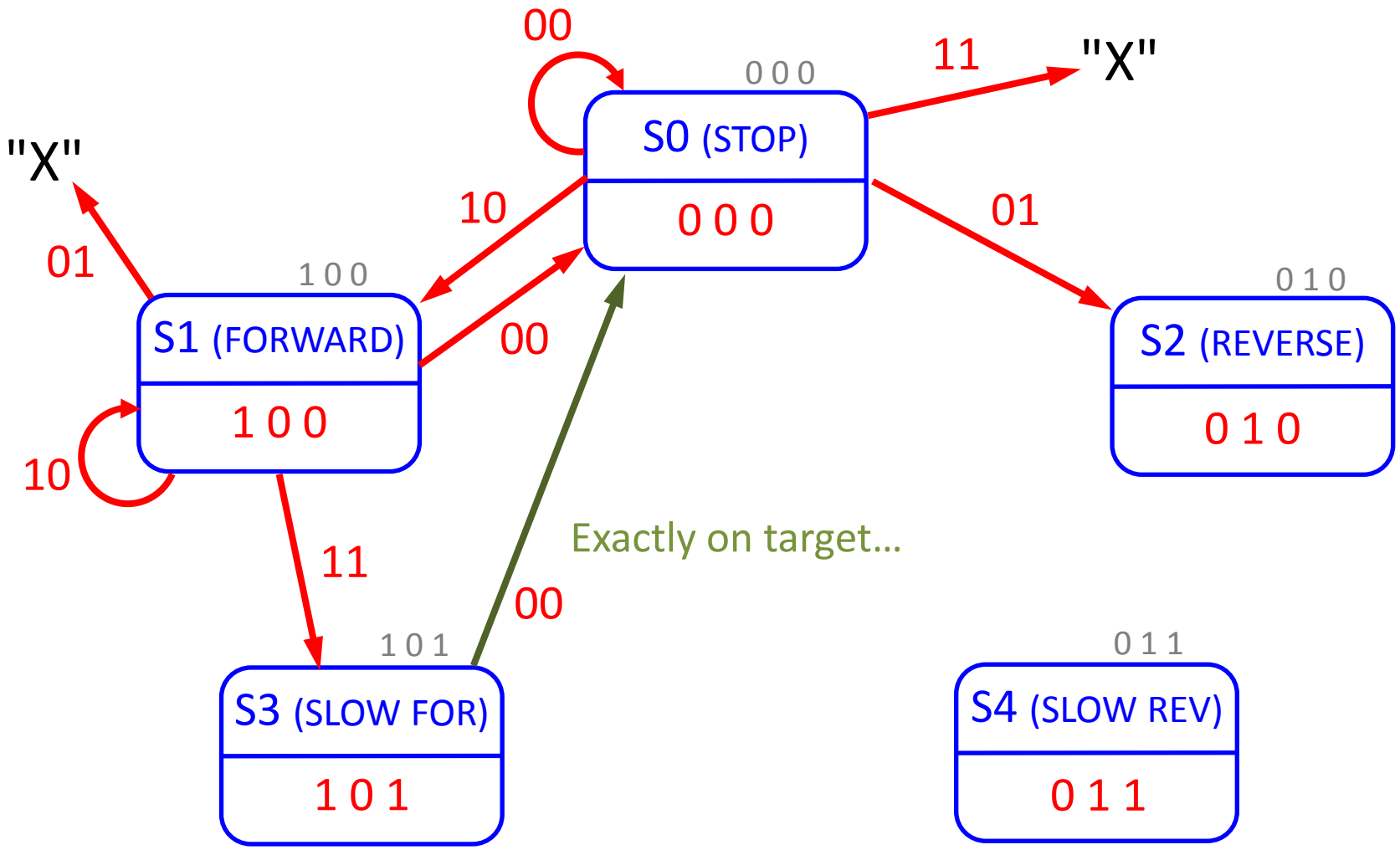


SIMPLE POSITION CONTROL

Inputs:
P N

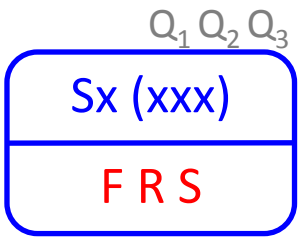


Sketch the state transition diagram for the FSM.

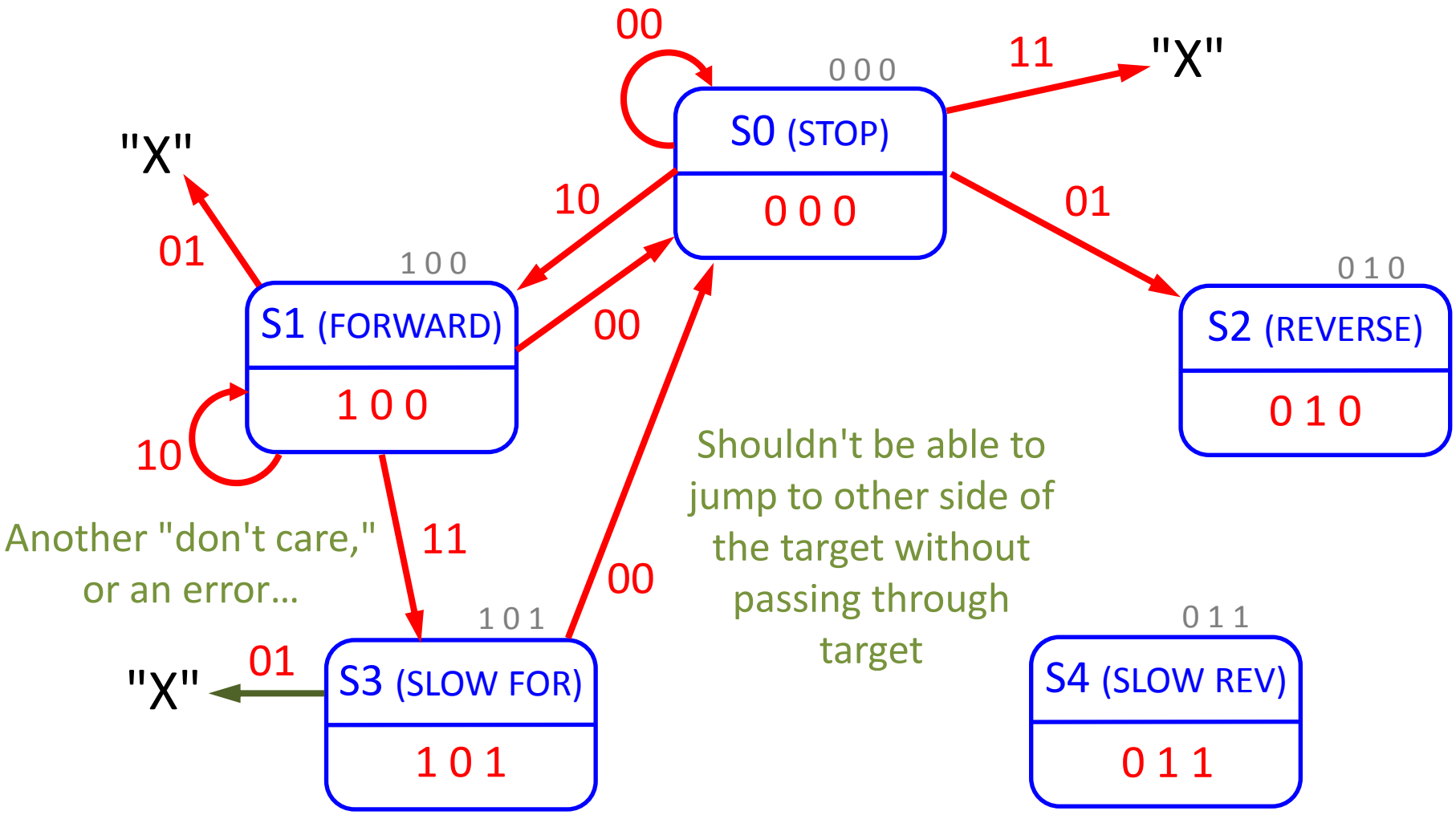


SIMPLE POSITION CONTROL

Inputs:
P N

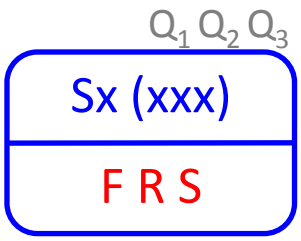


Sketch the state transition diagram for the FSM.

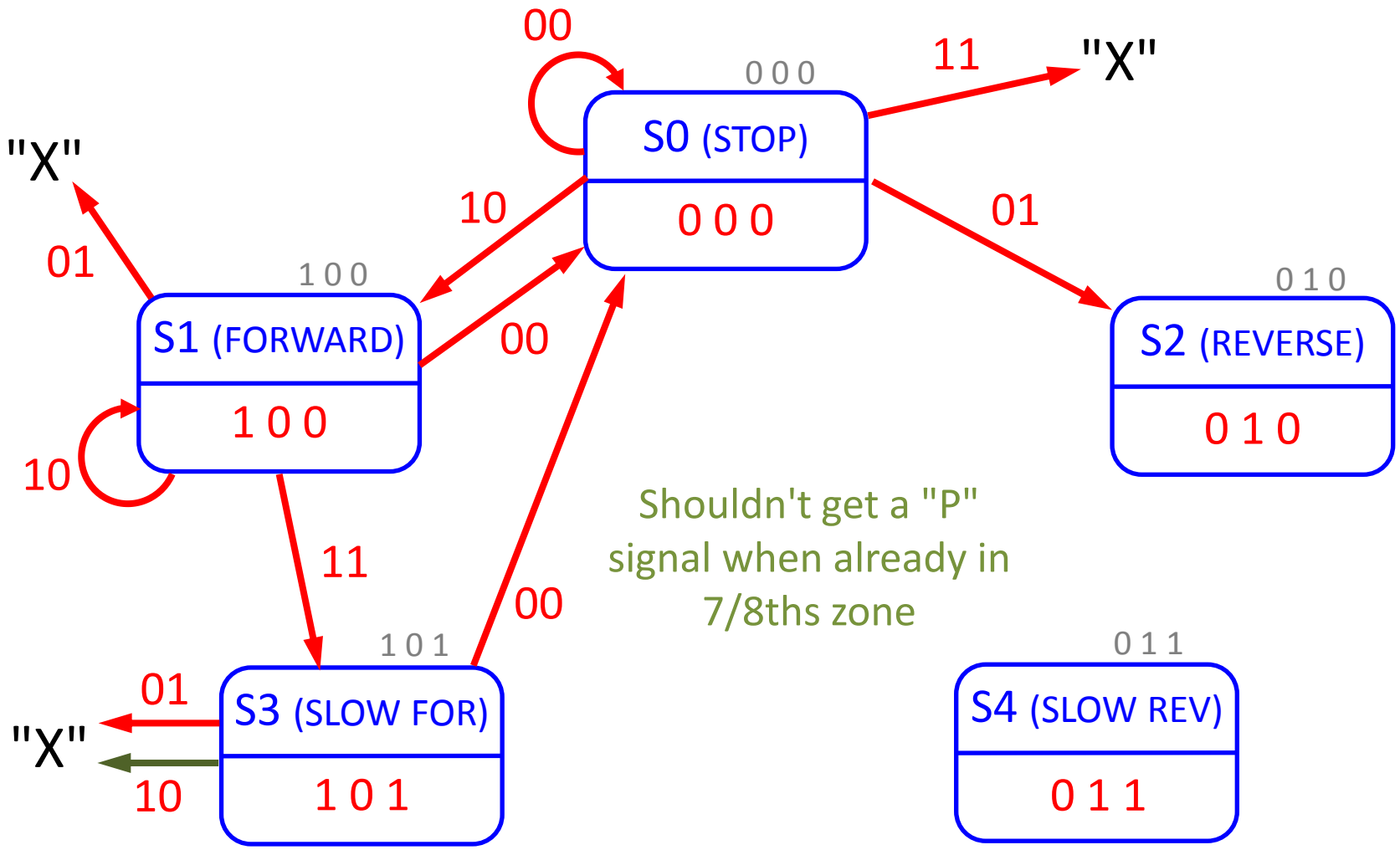


SIMPLE POSITION CONTROL

Inputs:
P N

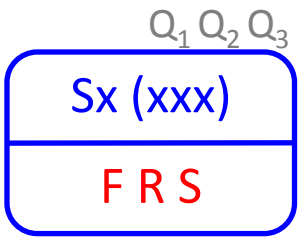


Sketch the state transition diagram for the FSM.

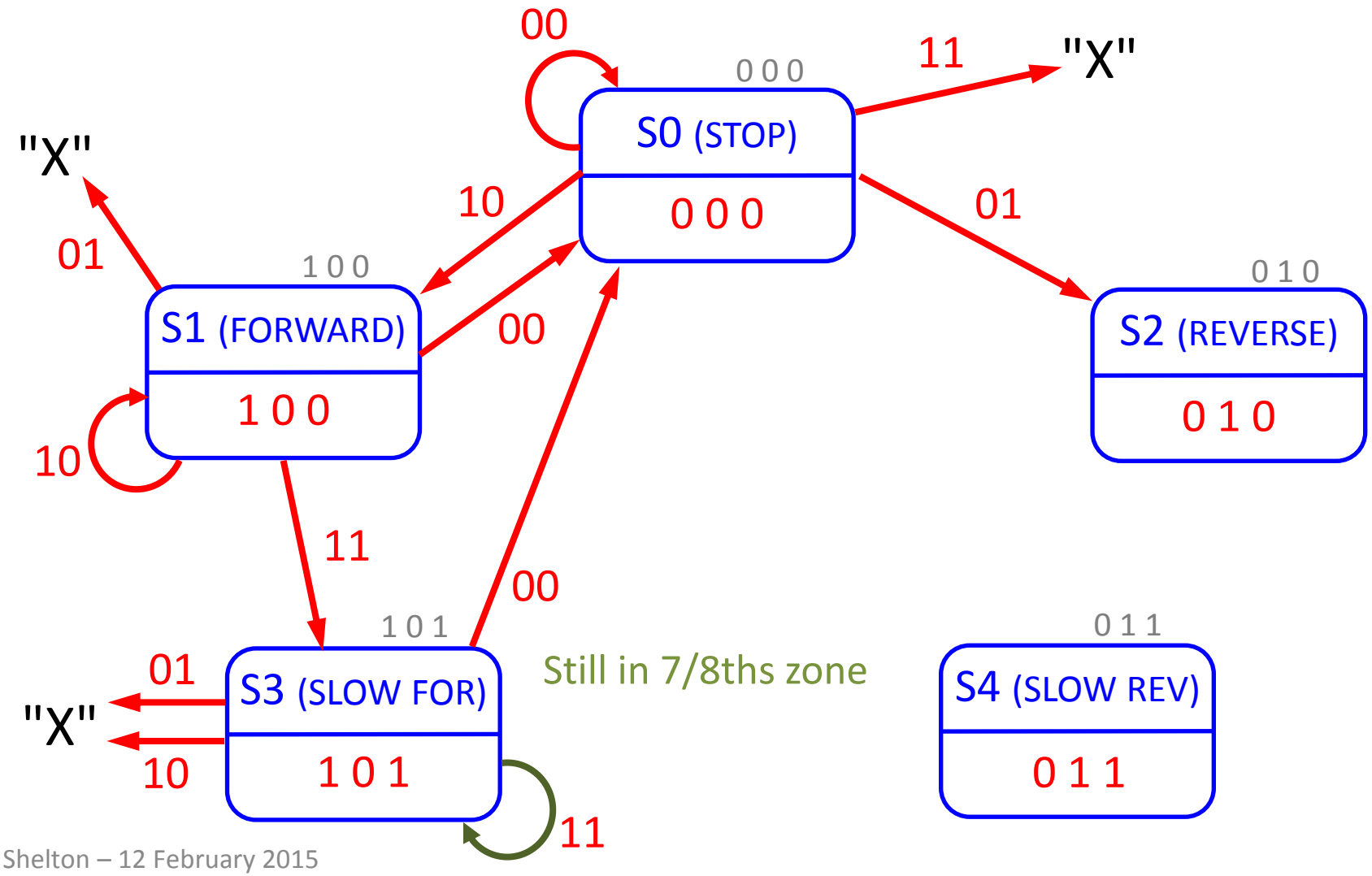


SIMPLE POSITION CONTROL

Inputs:
P N

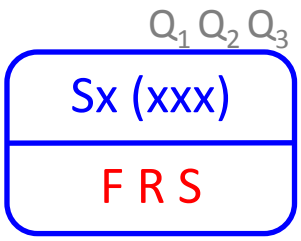


Sketch the state transition diagram for the FSM.

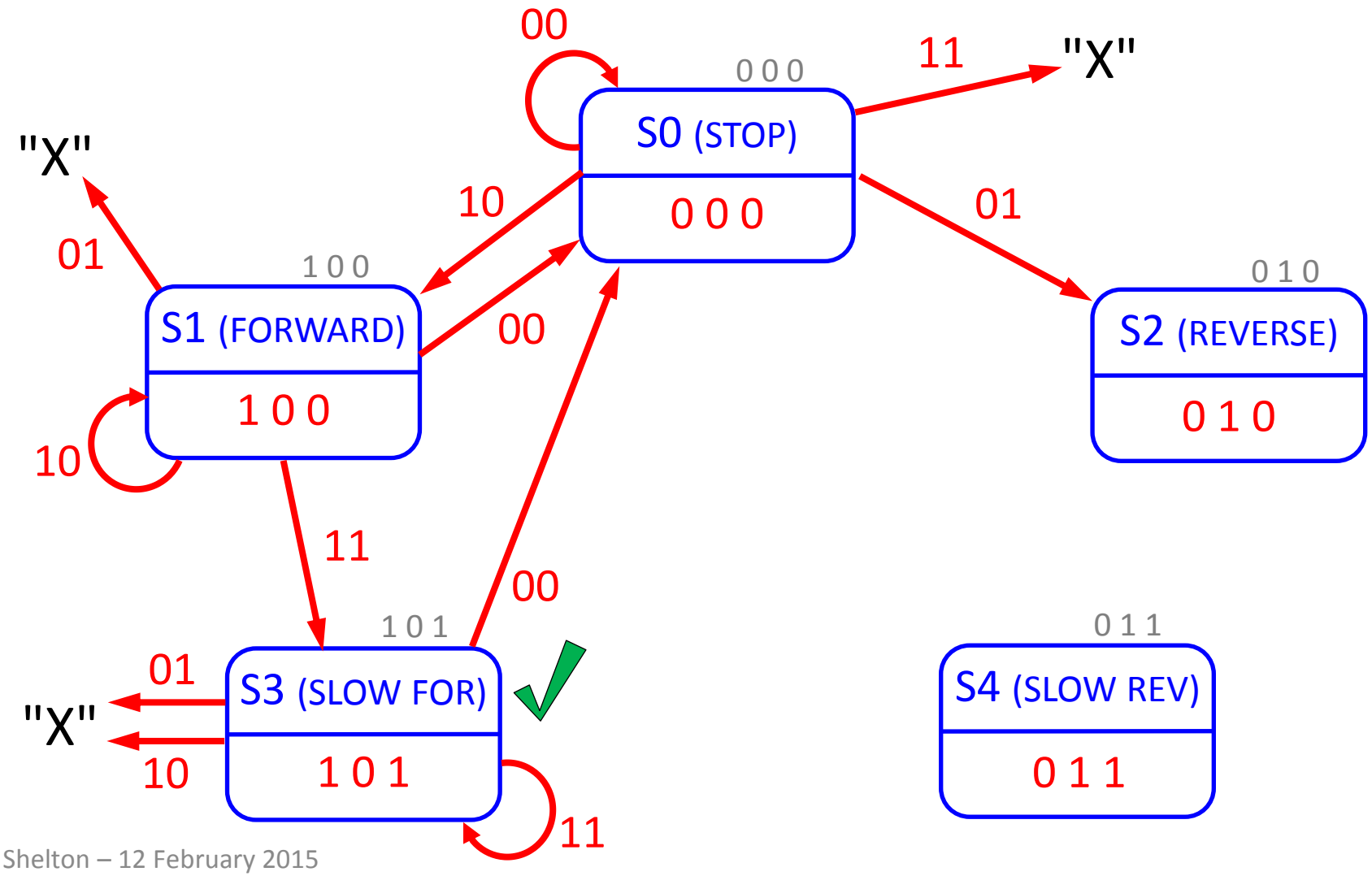


SIMPLE POSITION CONTROL

Inputs:
P N

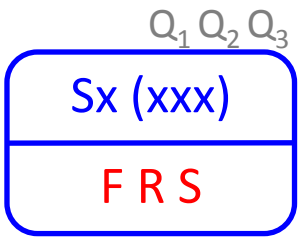


Sketch the state transition diagram for the FSM.

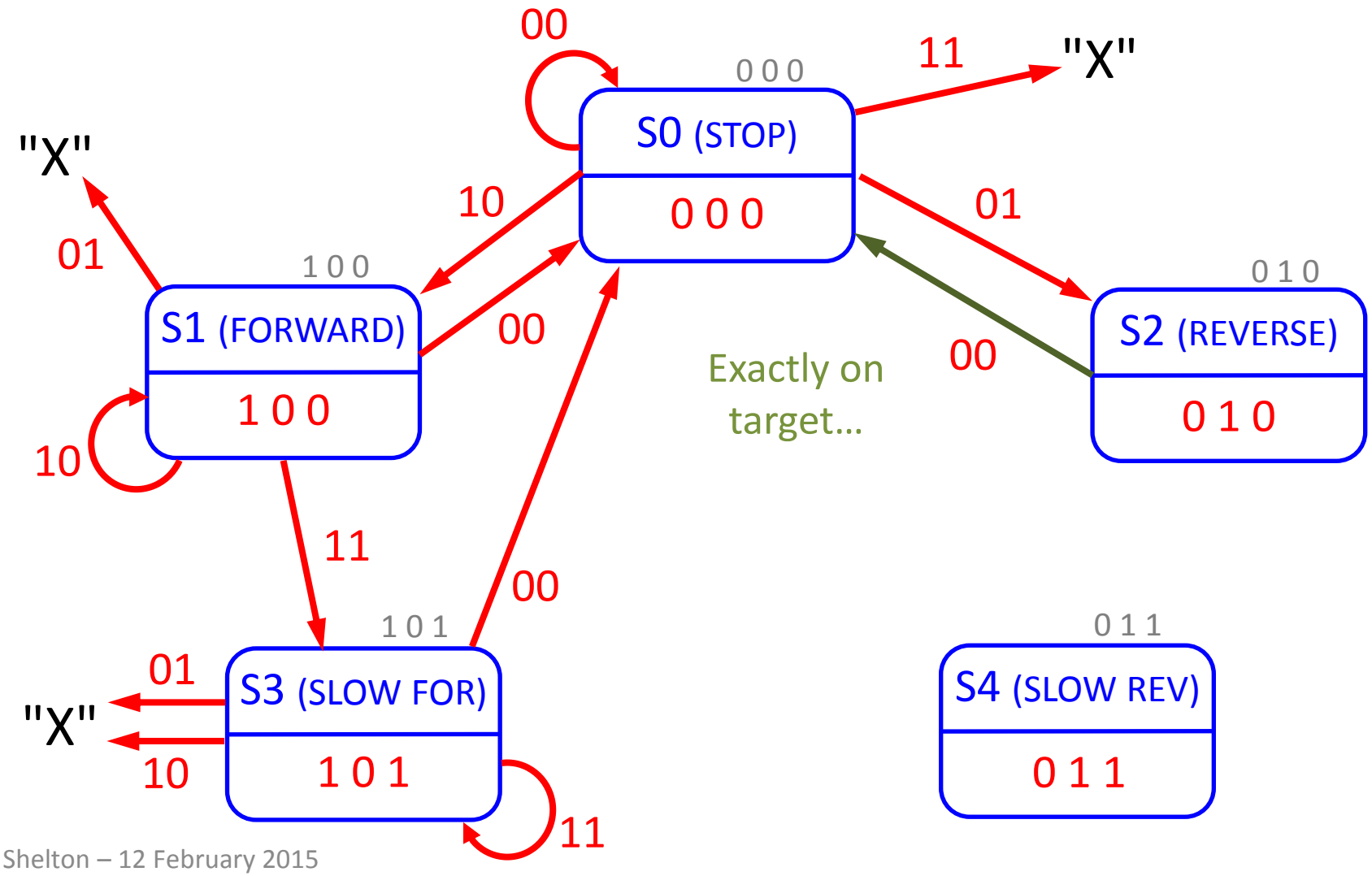


SIMPLE POSITION CONTROL

Inputs:
P N

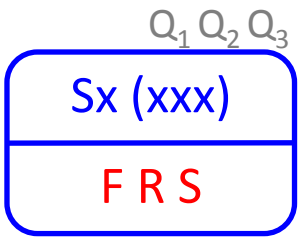


Sketch the state transition diagram for the FSM.

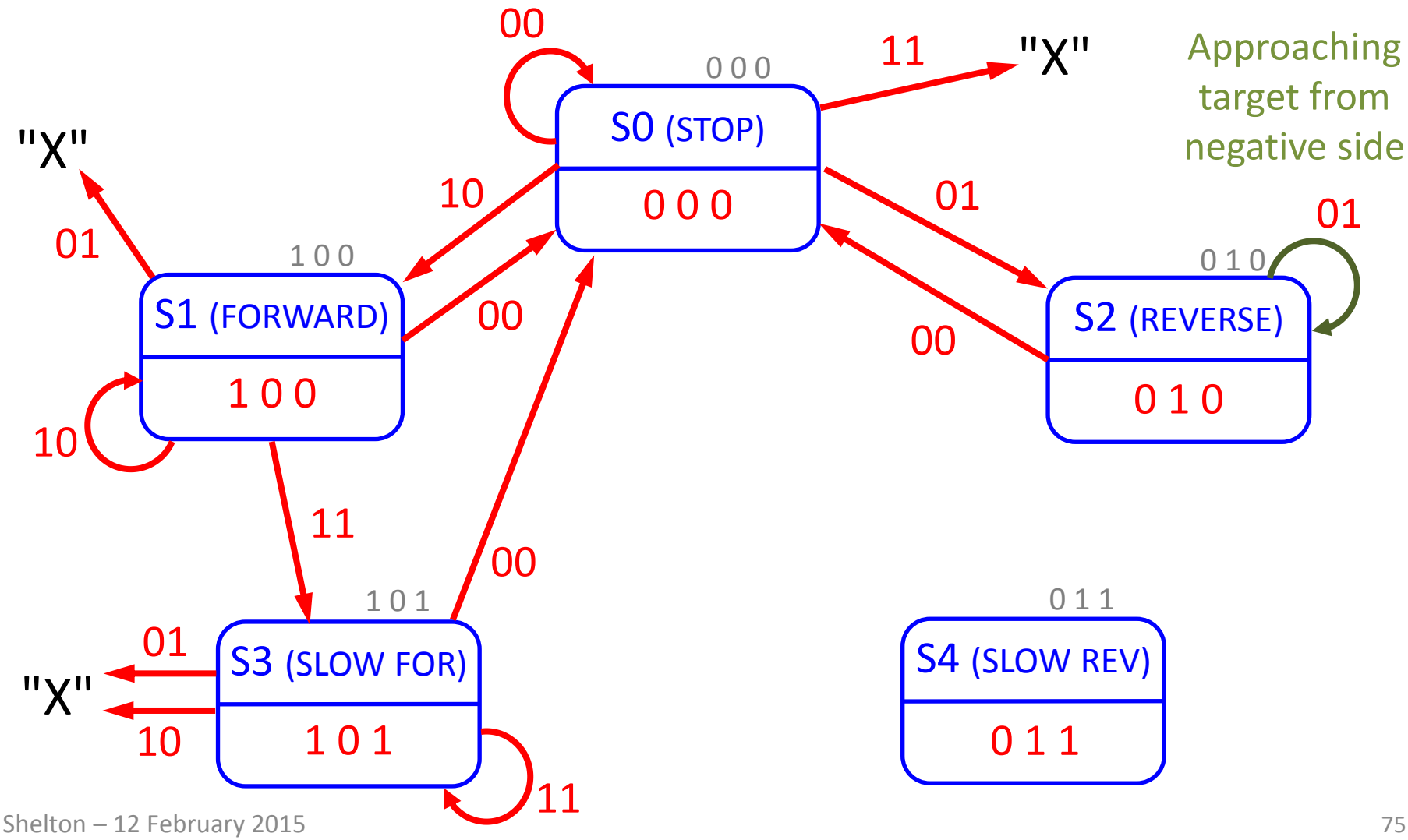


SIMPLE POSITION CONTROL

Inputs:
P N

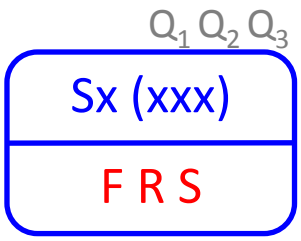


Sketch the state transition diagram for the FSM.

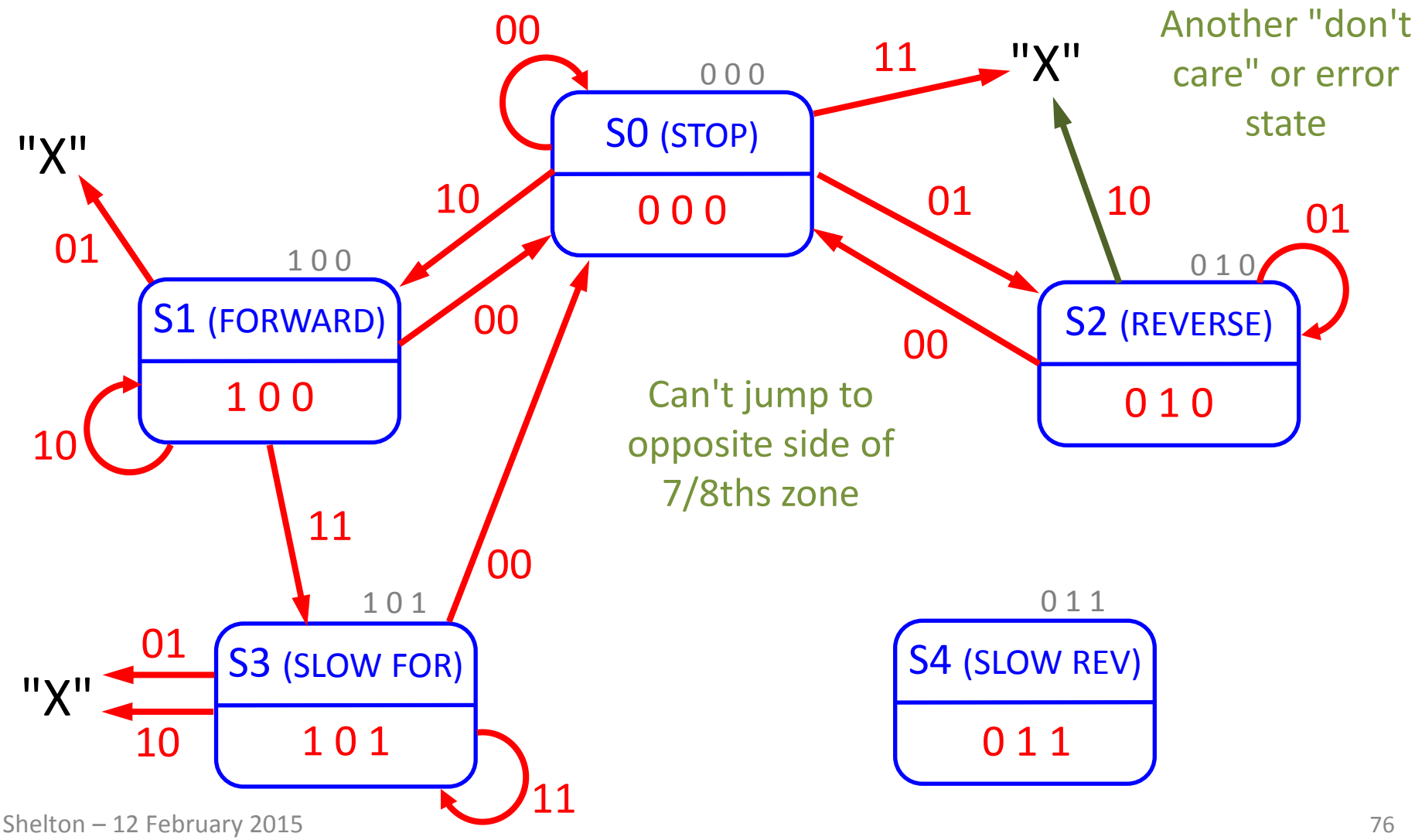


SIMPLE POSITION CONTROL

Inputs:
P N

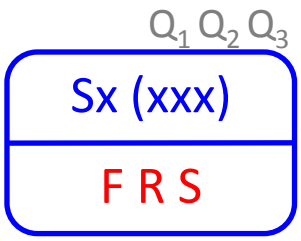


Sketch the state transition diagram for the FSM.

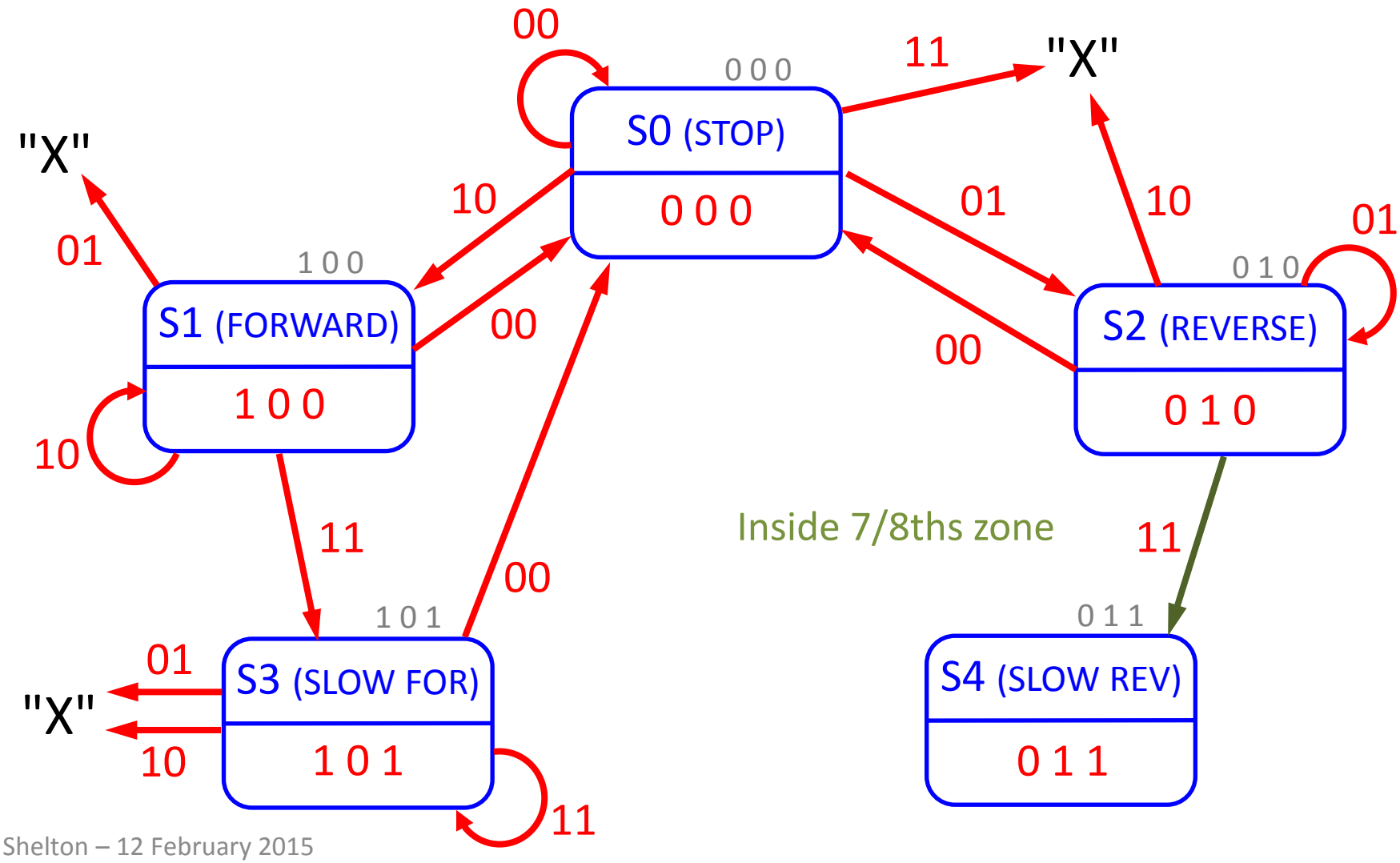


SIMPLE POSITION CONTROL

Inputs:
P N

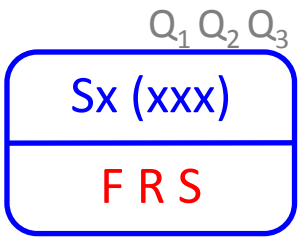


Sketch the state transition diagram for the FSM.

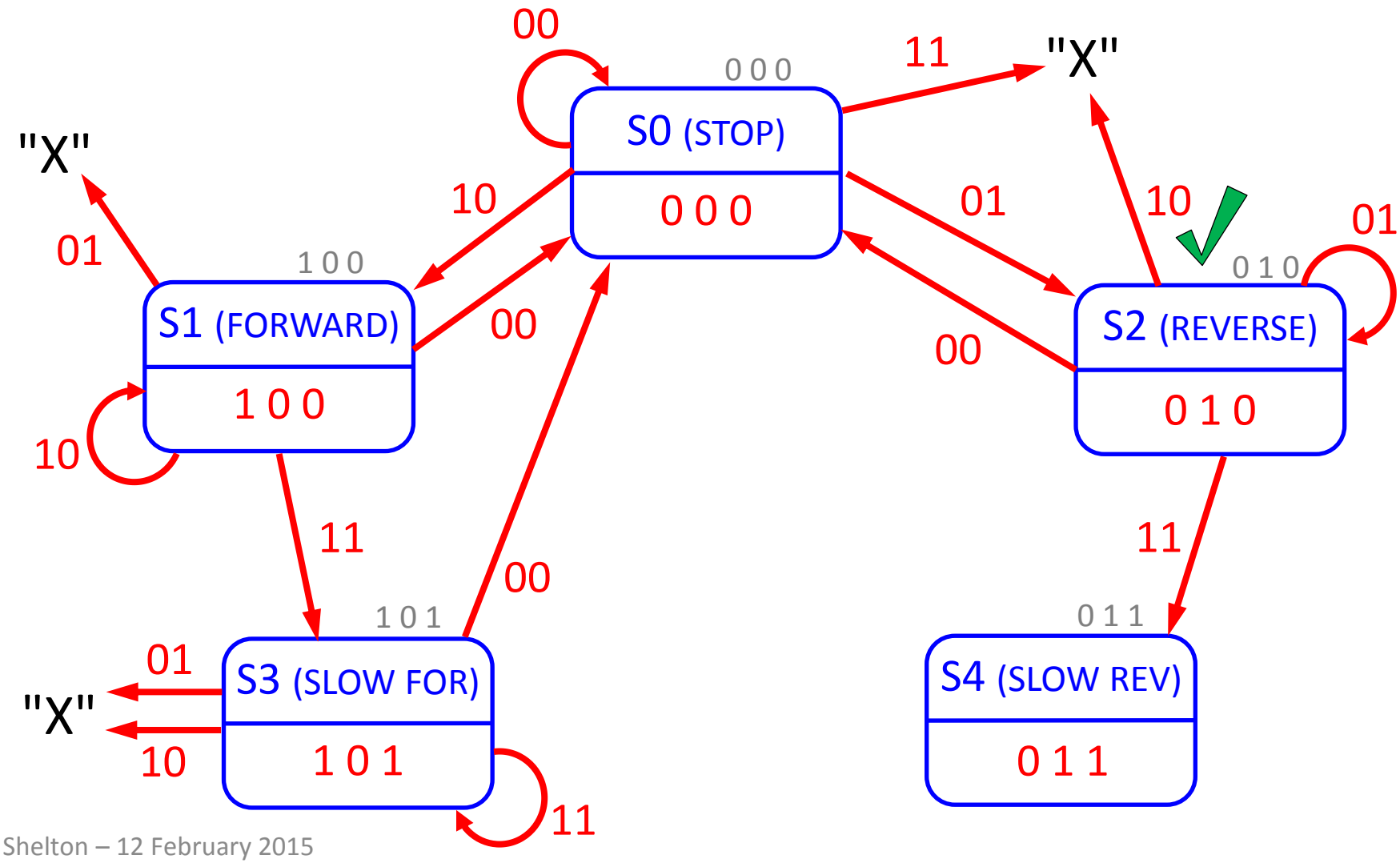


SIMPLE POSITION CONTROL

Inputs:
P N

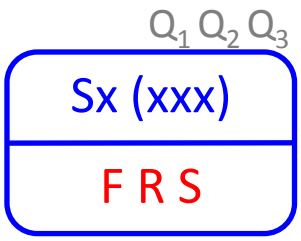


Sketch the state transition diagram for the FSM.

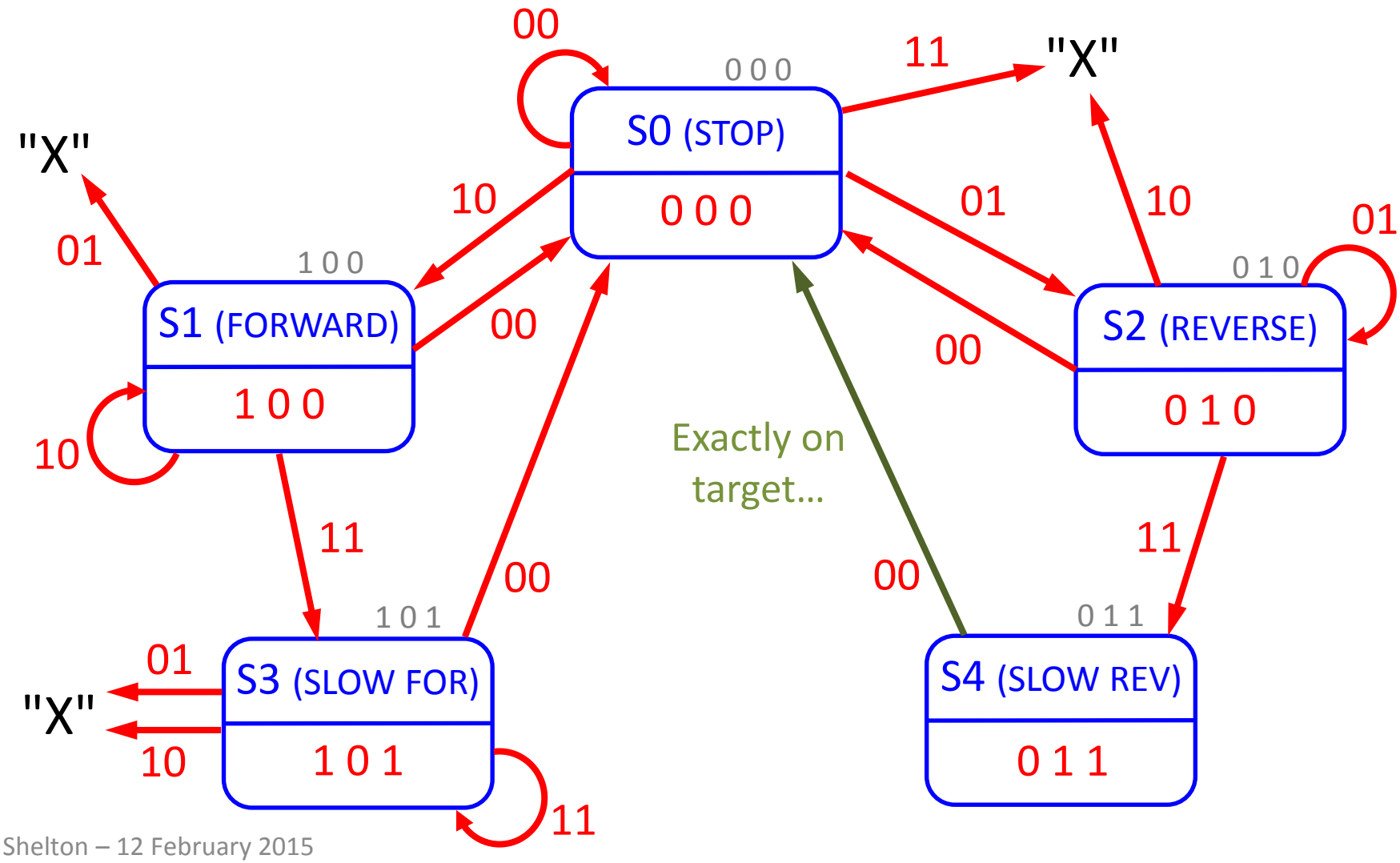


SIMPLE POSITION CONTROL

Inputs:
P N

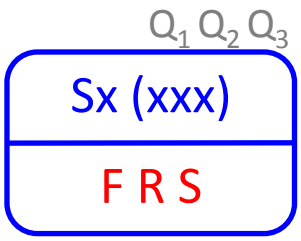


Sketch the state transition diagram for the FSM.



SIMPLE POSITION CONTROL

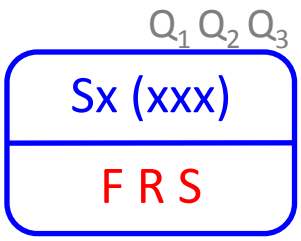
Inputs:
P N



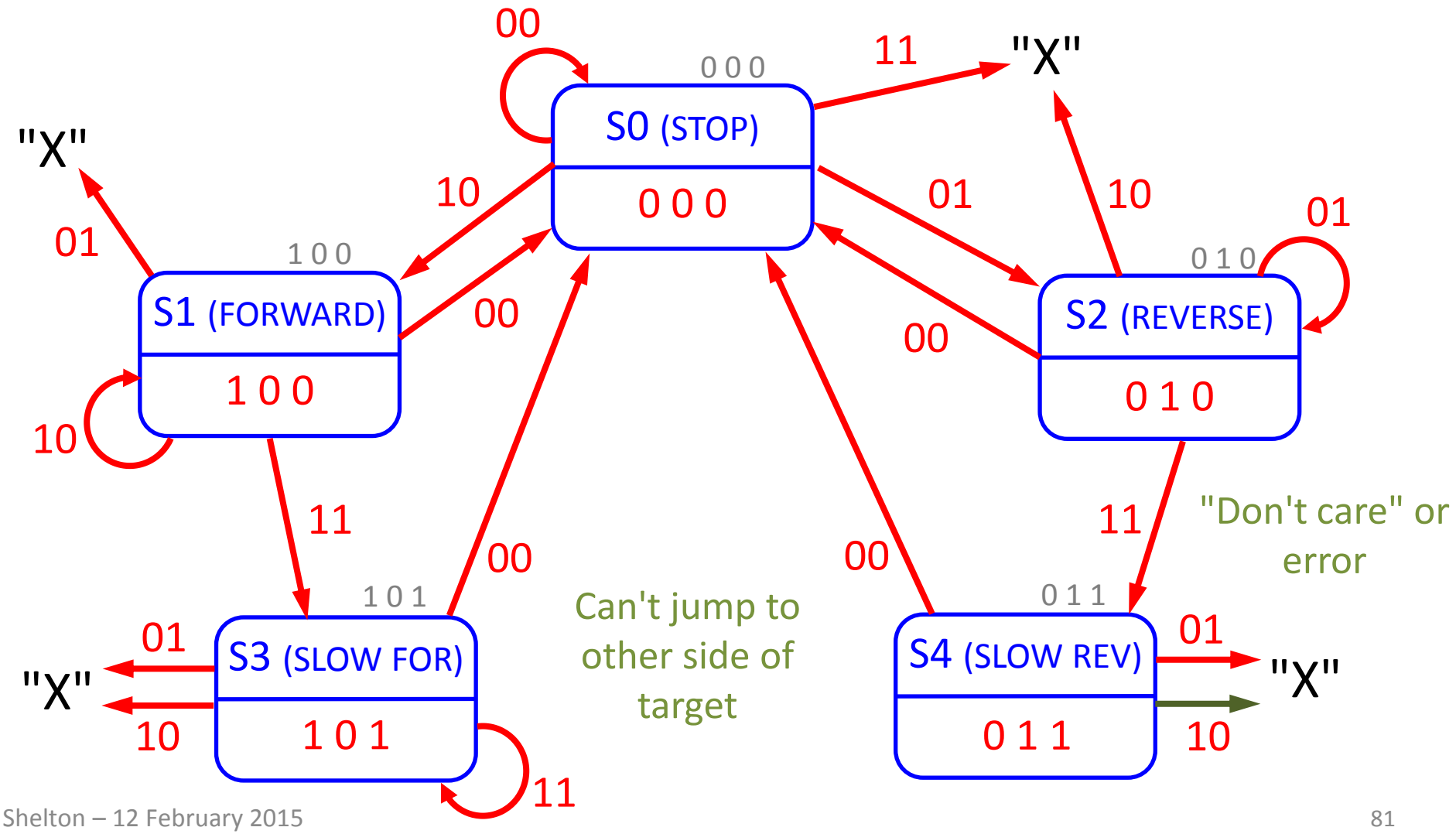
Sketch the state transition diagram for the FSM.

SIMPLE POSITION CONTROL

Inputs:
P N

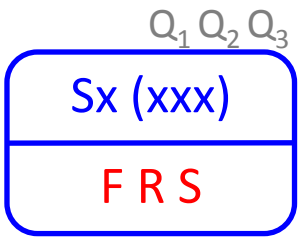


Sketch the state transition diagram for the FSM.

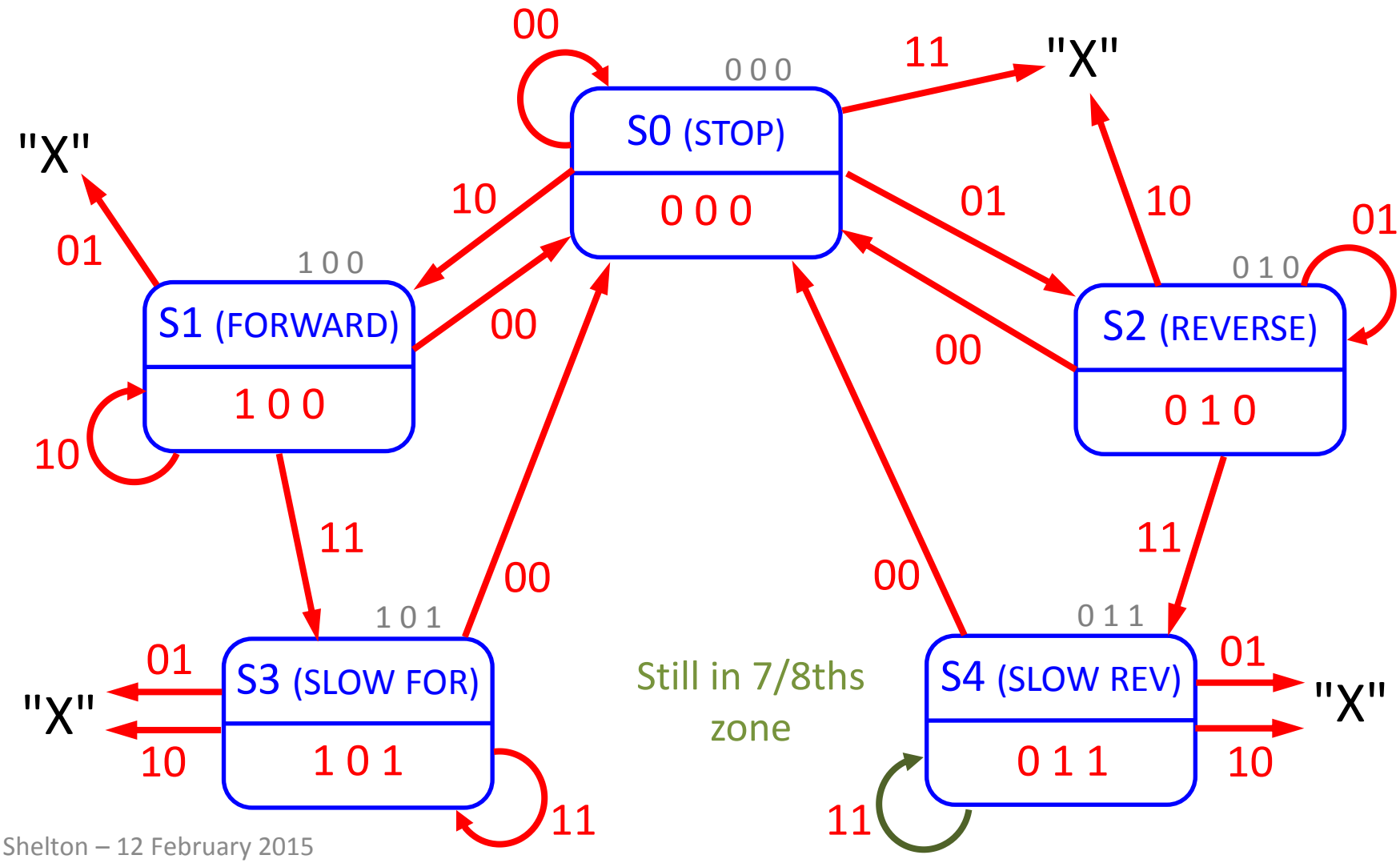


SIMPLE POSITION CONTROL

Inputs:
P N

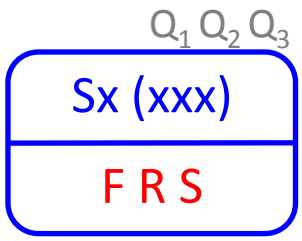


Sketch the state transition diagram for the FSM.

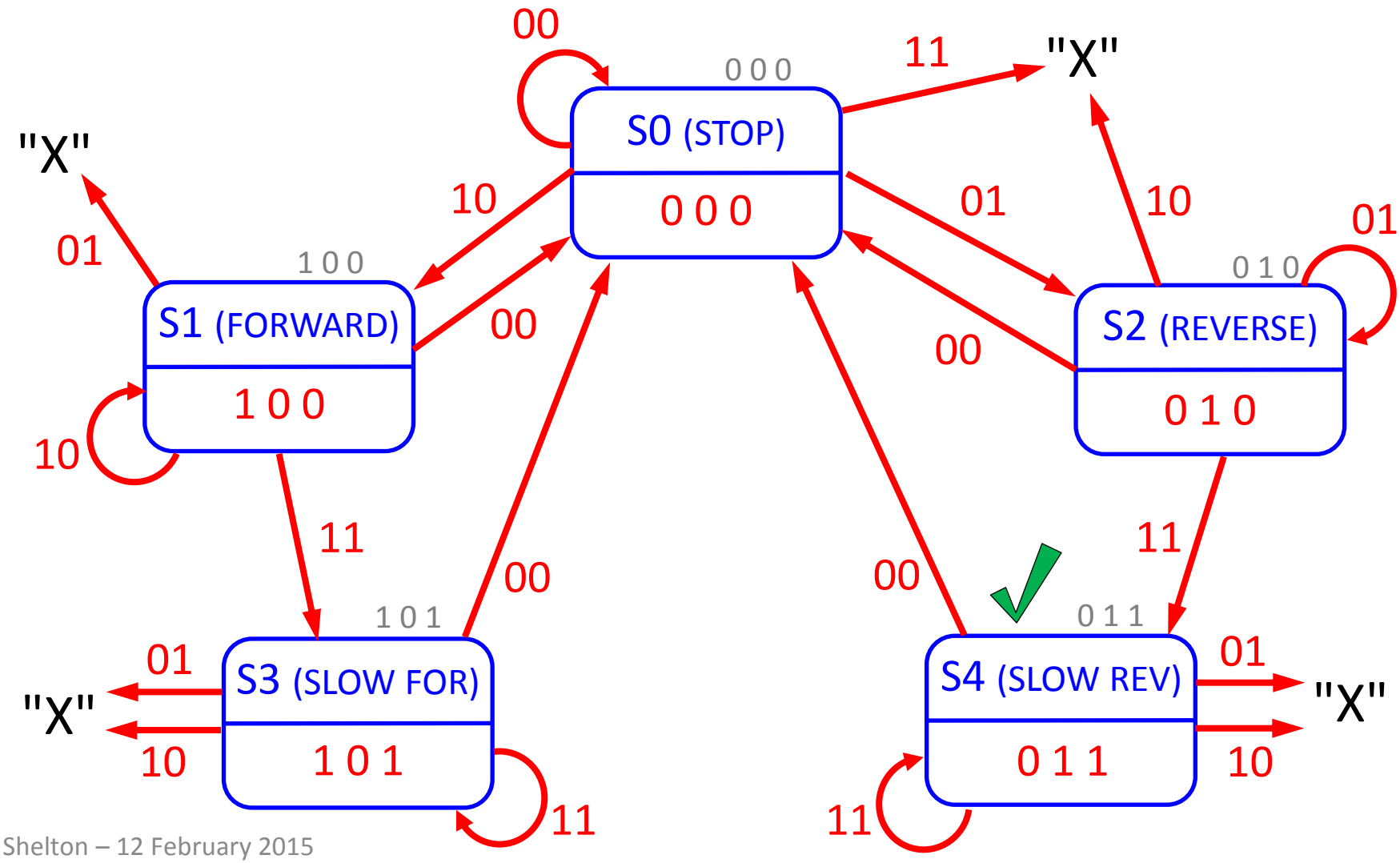


SIMPLE POSITION CONTROL

Inputs:
P N



Sketch the state transition diagram for the FSM.



SIMPLE POSITION CONTROL

Truth Table:

Input		Current State	Next State	Output		
P	N			F	R	S
0	0	S0	S0	0	0	0
0	0	S1	S0	1	0	0
0	0	S2	S0	0	1	0
0	0	S3	S0	1	0	1
0	0	S4	S0	0	1	1
0	1	S0	S2	0	0	0
0	1	S1	X	1	0	0
0	1	S2	S2	0	1	0
0	1	S3	X	1	0	1
0	1	S4	X	0	1	1
1	0	S0	S1	0	0	0
1	0	S1	S1	1	0	0
1	0	S2	X	0	1	0
1	0	S3	X	1	0	1
1	0	S4	X	0	1	1
1	1	S0	X	0	0	0
1	1	S1	S3	1	0	0
1	1	S2	S4	0	1	0
1	1	S3	S3	1	0	1
1	1	S4	S4	0	1	1

SIMPLE POSITION CONTROL

Truth Table:

Input		Current State			Next State			Output		
P	N	Q0	Q1	Q2	Q0*	Q1*	Q2*	F	R	S
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	1	0	0	0	0	1	1
0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	1	0	0	0	1	0	1
0	1	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	0	0	1	0
0	1	0	1	1	X	X	X	0	1	1
0	1	1	0	0	X	X	X	1	0	0
0	1	1	0	1	X	X	X	1	0	1
1	0	0	0	0	1	0	0	0	0	0
1	0	0	1	0	X	X	X	0	1	0
1	0	0	1	1	X	X	X	0	1	1
1	0	1	0	0	1	0	0	1	0	0
1	0	1	0	1	X	X	X	1	0	1
1	1	0	0	0	X	X	X	0	0	0
1	1	0	1	0	0	1	1	0	1	0
1	1	0	1	1	0	1	1	0	1	1
1	1	1	0	0	1	0	1	1	0	0
1	1	1	0	1	1	0	1	1	0	1

SIMPLE POSITION CONTROL

Next State Logic:

Output Logic:

$Q0^*$ ($P = 0$)		$Q1\ Q2$			
		00	01	11	10
$N\ Q0$	00				
	01				
	11				
	10				

$Q0^*$ ($P = 1$)		$Q1\ Q2$			
		00	01	11	10
$N\ Q0$	00				
	01				
	11				
	10				

$Q1^*$ ($P = 0$)		$Q1\ Q2$			
		00	01	11	10
$N\ Q0$	00				
	01				
	11				
	10				

$Q1^*$ ($P = 1$)		$Q1\ Q2$			
		00	01	11	10
$N\ Q0$	00				
	01				
	11				
	10				

$Q2^*$ ($P = 0$)		$Q1\ Q2$			
		00	01	11	10
$N\ Q0$	00				
	01				
	11				
	10				

$Q2^*$ ($P = 1$)		$Q1\ Q2$			
		00	01	11	10
$N\ Q0$	00				
	01				
	11				
	10				

SIMPLE POSITION CONTROL

Next State Logic:

$Q0^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q0^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	1	X	X	X
01	1	X	X	X
11	1	1	X	X
10	X	X	0	0

Output Logic:

$Q1^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	1	X	X	1

$Q1^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	0	0	X	X
10	X	X	1	1

$Q2^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q2^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	1	X	X
10	X	X	1	1

SIMPLE POSITION CONTROL

Next State Logic:

$$Q0^* = P \cdot \overline{Q1}$$

Output Logic:

$Q0^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q0^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	1	X	X	X
01	1	X	X	X
11	1	1	X	X
10	X	X	0	0

$Q1^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	1	X	X	1

$Q1^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	0	0	X	X
10	X	X	1	1

$Q2^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q2^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	1	X	X
10	X	X	1	1

SIMPLE POSITION CONTROL

Next State Logic:

$$Q0^* = P \cdot \overline{Q1}$$
$$Q1^* = N \cdot \overline{Q0}$$

Output Logic:

$Q0^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q0^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	1	X	X	X
01	1	X	X	X
11	1	1	X	X
10	X	X	0	0

$Q1^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	1	X	X	1

$Q1^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	0	0	X	X
10	X	X	1	1

$Q2^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q2^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	1	X	X
10	X	X	1	1

SIMPLE POSITION CONTROL

Next State Logic:

$$Q0^* = P \cdot \overline{Q1}$$
$$Q1^* = N \cdot \overline{Q0}$$
$$Q2^* = P \cdot N$$

Output Logic:

$Q0^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q0^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	1	X	X	X
01	1	X	X	X
11	1	1	X	X
10	X	X	0	0

$Q1^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	1	X	X	1

$Q1^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	0	0	X	X
10	X	X	1	1

$Q2^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q2^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	1	X	X
10	X	X	1	1

SIMPLE POSITION CONTROL

Next State Logic:

$$Q0^* = P \cdot \overline{Q1}$$
$$Q1^* = N \cdot \overline{Q0}$$
$$Q2^* = P \cdot N$$

Output Logic:

$$F = Q0$$
$$R = Q1$$
$$S = Q2$$

$Q0^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q0^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	1	X	X	X
01	1	X	X	X
11	1	1	X	X
10	X	X	0	0

$Q1^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	1	X	X	1

$Q1^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	0	0	X	X
10	X	X	1	1

$Q2^*$ ($P = 0$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	0	0
01	0	0	X	X
11	X	X	X	X
10	0	X	X	0

$Q2^*$ ($P = 1$)

N Q0 \ Q1 Q2	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	1	X	X
10	X	X	1	1

COMING UP…

Computer Systems (Sequential Logic)

- Finite state reduction