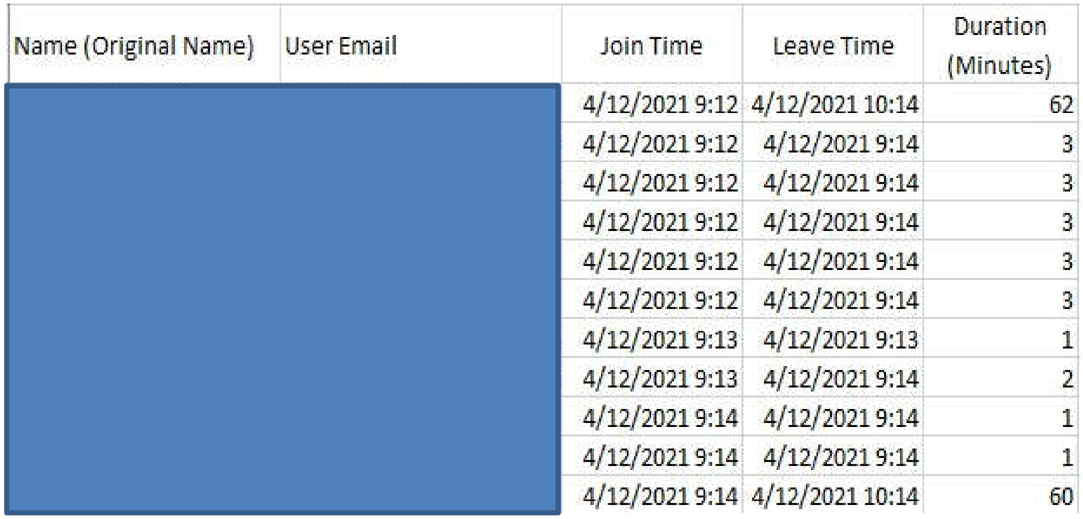
Introduction to Robotics

Felipe P. Vista IV

Class Admin Matters

Grading

- Attendance 5%



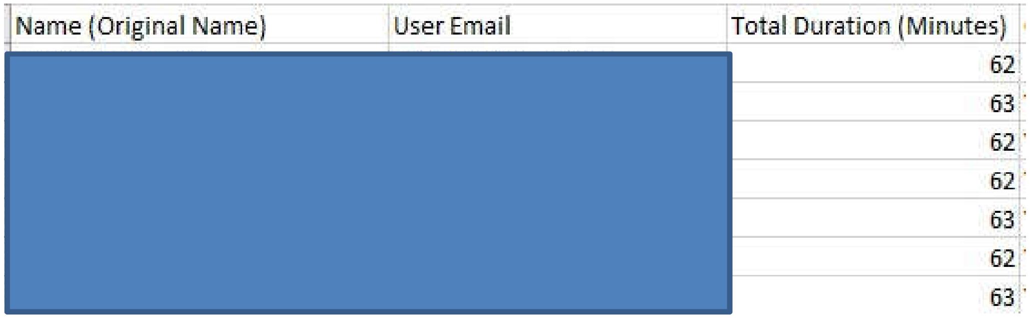
Name (Original Name)	User Email	Join Time	Leave Time	Duration (Minutes)
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 10:14	62
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:13	4/12/2021 9:13	1
[redacted]	[redacted]	4/12/2021 9:13	4/12/2021 9:14	2
[redacted]	[redacted]	4/12/2021 9:14	4/12/2021 9:14	1
[redacted]	[redacted]	4/12/2021 9:14	4/12/2021 9:14	1
[redacted]	[redacted]	4/12/2021 9:14	4/12/2021 10:14	60

Bad ZOOM User Name (Absent)

- Iphone → Not your name
- SiAko 202100001 → Wrong order
- SiAko → Name only
- 202100001 → ID Num only

ZOOM User Name (Present)

- University ID Num_Name
- 202100001 SiAko → GOOD (Present)



Name (Original Name)	User Email	Total Duration (Minutes)
[redacted]	[redacted]	62
[redacted]	[redacted]	63
[redacted]	[redacted]	62
[redacted]	[redacted]	62
[redacted]	[redacted]	63
[redacted]	[redacted]	62
[redacted]	[redacted]	63

Student Responsibilities

- Download/Install ZOOM app for online lecture
  - *Zoom profile must be your* ***OASIS ID+name*** *similar to OASIS*
  - *Ex.:* ***202061234 YourName***
  - ***If you are asked, but no reply, then you'll be out of zoom & mark absent***
- Regularly login, check OLD IEILMS for updates, notifications
  - *https://ieilmsold.jbnu.ac.kr*
  - *Presentations & lecture videos will be uploaded after class*
- Regularly check Kakao Group Chat for class
  - *Everybody must have a Kakao talk account*
  - *Search & add account "****botjok****", introduce yourself and name of class ("Robotics"), then you will be added to the group chat*

Class Admin Matters

Student Responsibilities

- Download/Install ZOOM app for online lecture
  - *Zoom profile must be your **OASIS ID+name** similar to OASIS*
  - *Ex.: **202061234 YourName***
  - *If you are asked, but no reply, then you'll be out of zoom & mark **absent***
- Regularly login, check OLD IEILMS for updates, notifications
  - *https://ieilmsold.jbnu.ac.kr*
  - *Presentations & lecture videos will be uploaded after class*
- Regularly check Kakao Group Chat for class
  - *Everybody must have a Kakao talk account*
  - *Search & add account "**botjok**", introduce yourself and name of class ("Robotics"), then you will be added to the group chat*

Intro To Robotics

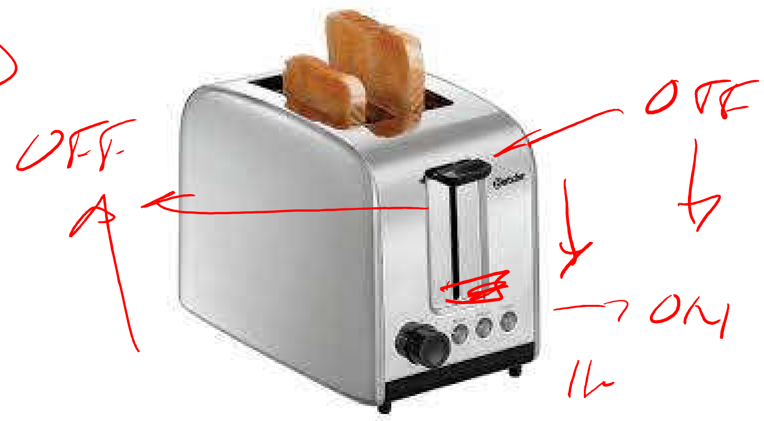
FINITE STATE MACHINES

Finite State Machines

- State Machines
- Reactive Behaviour with State
- Search & Approach
- Implementation of Finite State Machines

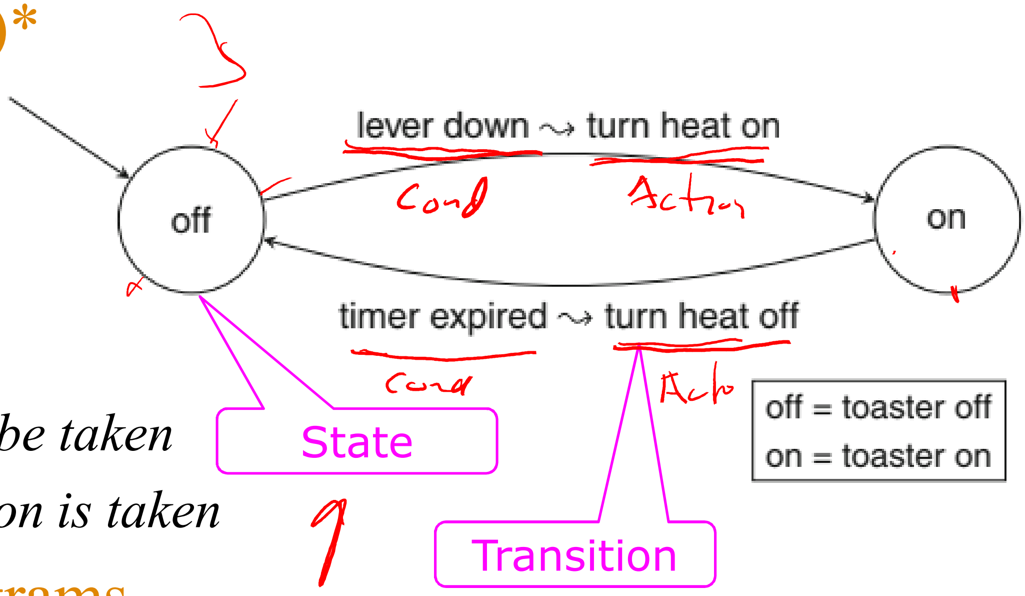
Intro

- Reactive behaviour
  - *Demonstrated by the Braitenberg vehicles & line following algorithms*
  - *Robot action depended on* **current** *values from sensor(s)*
    - **Not** from **events** that previously happened
- State
  - *Familiar concept*
- Ex: A toaster
  - *Initially*: ***OFF*** *state*
  - *Pushing lever down*: *Transition to* ***ON*** *state*
    - Heater turns on
  - *Timer expires*: *Transition to* ***OFF*** *state*
    - Heater turns off



State Machines

- Finite State Machines (FSM)*
  - *states s_i*
  - *transitions bet states s_i & s_j*
- Transition
  - *Labeled as "condition/action"*
  - *Condition: cause transition to be taken*
  - *Action: executed when transition is taken*
- FSMs displayed in state diagrams
  - **(1) State**: *denoted by circle with name*
  - **(2) Transition**: *arrow from source to target*
  - **Arrow**: *w/ condition & action of transition*
  - *Action not continuing**



* Also called ***finite automata***

* Ex:

Action "turn left" set motors so the robot turns left, but transition to next state executed w/out waiting for robot to reach specific place.

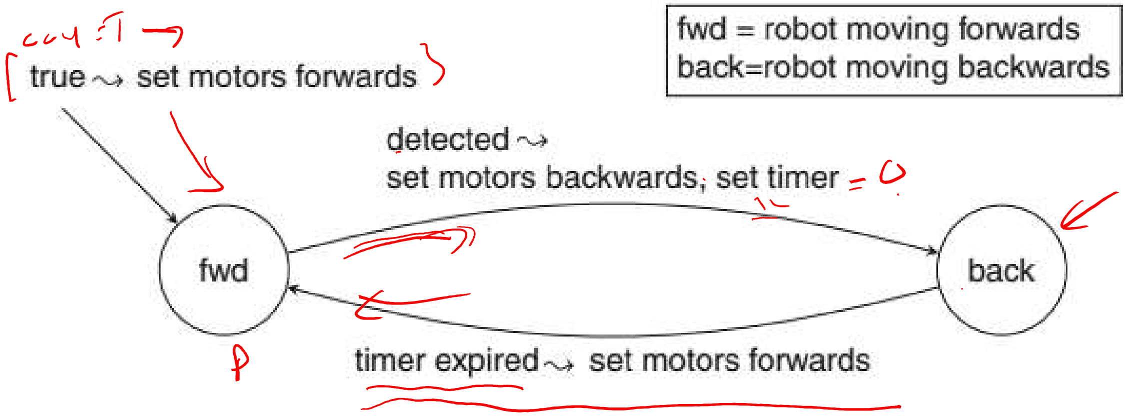
Introduction to
Robotics

Finite State Machines

- State Machines
- Reactive Behaviour with State
- Search & Approach
- Implementation of Finite State Machines

Reactive Behaviour w/ State

- **Specification (Persistent)**, for a Braitenberg vehicle, (non-reactive)
  - *Robot move forward until object detected.* — 1 T
    Then move backward for one second and reverses to move forward again. 2 T



fwd = robot moving forwards
back=robot moving backwards

true ⇝ set motors forwards

detected ⇝
set motors backwards, set timer = 0

timer expired ⇝ set motors forwards

FSM for persistent Braitenberg vehicle

* System turned ON, motors set move forward (condition always **TRUE**, this unconditionally done).
* at **fwd state**: If object detected → transition to state **back**, move backward, timer set
* at **back state**: after one second → transition to state **fwd**, move forward
** If object detected → no action performed, since no transition labeled w/ this condition
***therefore not reactive, depends on current state of robot & event happening

Reactive Behaviour w/ State

- Activity 4.1: **Specification (Consistent)**, for a Braitenberg vehicle
  - *Robot cycle through four states.*
    Changing states every second: forward, turn left, turn right, backward

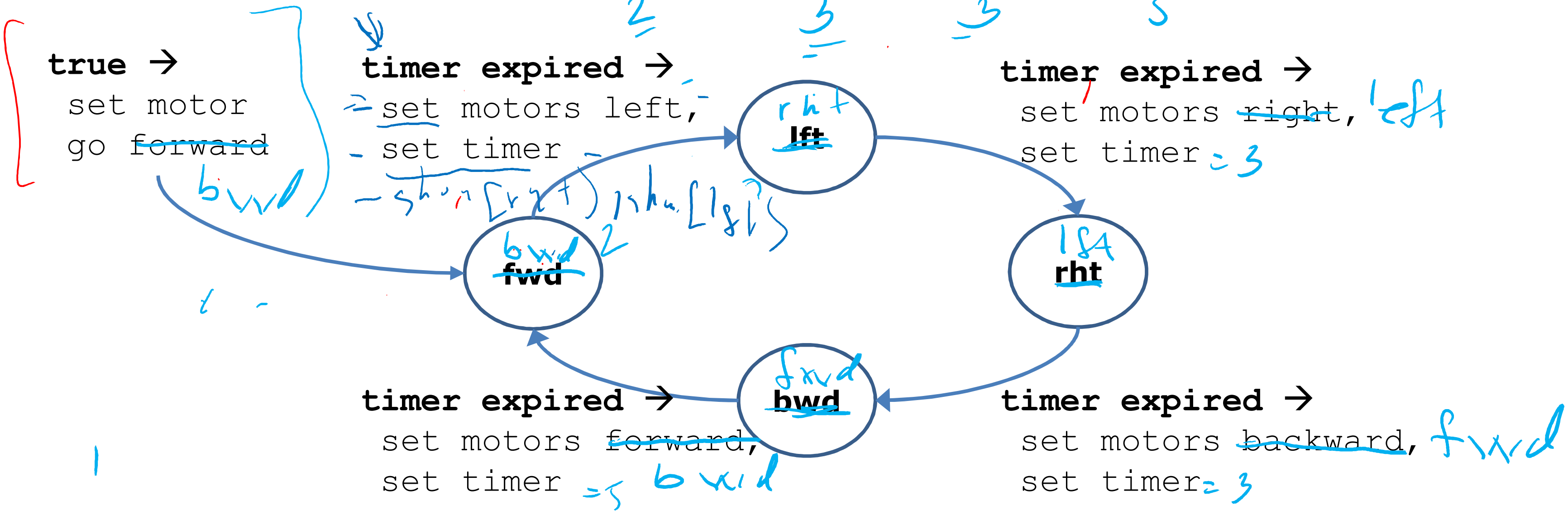
Reactive Behaviour w/ State

- Activity 4.1: **Specification (Consistent)**, for a Braitenberg vehicle
  - *Robot cycle through four states.*
    Changing states every second: forward, turn left, turn right, backward

```
fwd = robot moving forward; bwd = robot moving backward;
lft = robot turning left; rht = robot turning right
```

Reactive Behaviour w/ State

- Activity 4.1: **Specification (Consistent)**, for a Braitenberg vehicle
  - *Robot cycle through four states.* bxwd right left fwrd
    Changing states every second: forward, turn left, turn right, backward
    2 3 3 5



true →
set motor
go ~~forward~~ bwd

timer expired →
set motors left,
set timer
- shun [ryt] ishu [lgl]

timer expired →
set motors ~~right~~, left
set timer = 3

timer expired →
set motors ~~forward~~, bwd
set timer =5

timer expired →
set motors ~~backward~~, fwrd
set timer = 3

FSM for consistent Braitenberg vehicle

fwd = robot moving forward; **bwd** = robot moving backward;
lft = robot turning left; **rht** = robot turning right

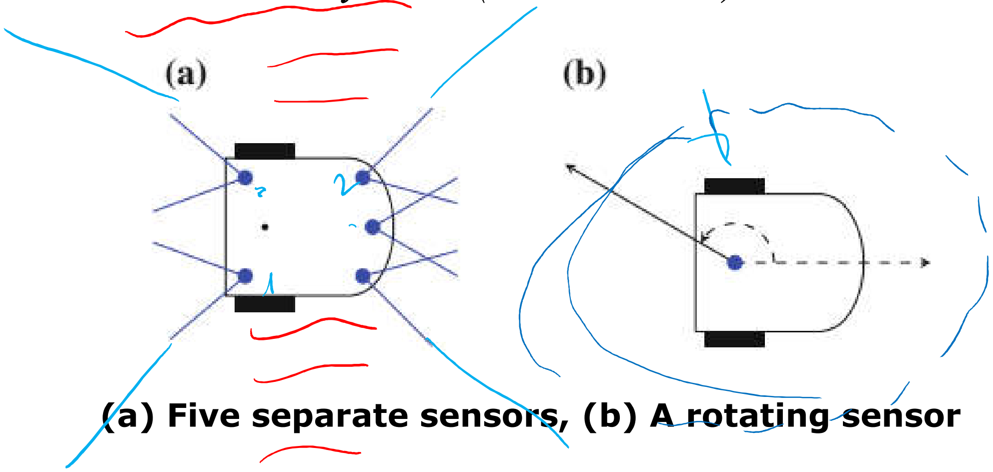
Finite State Machines

- State Machines
- Reactive Behaviour with State
- Search & Approach
- Implementation of Finite State Machines

Search & Approach

- More complex sample of robotic behaviour using states
- **Specification (Search & Approach),**
  - *Robot search left & right (±45°).*
    If object detected, robot approaches object and stops when it's near the object.

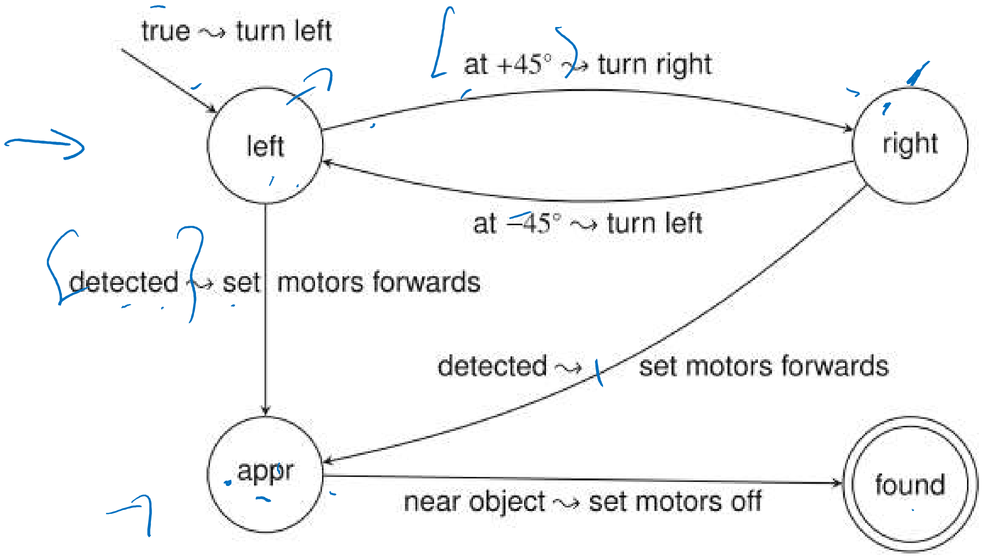
- Two possible sensor configuration to search left & right:
  (a) ***sensors fixed****, robot must turn*
  (b) ***rotating sensor****, robot stay still (we use this)*



(a) Five separate sensors, (b) A rotating sensor

Search & Approach

- **Specification (Search & Approach),**
  - *Robot search left & right (±45°).*
    If object detected, robot approaches object and stops when it's near the object.
- FSM
  - **(3) Final State** *(double circle)*
  - *Finite num of states & transitions*
  - *Behaviour can be finite or infinite*
- Current example behaviour
  - **Finite**: *robot stops when it finds an object & approaches it*
  - **Infinite**: *robot indefinitely continues search if object never found*


true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

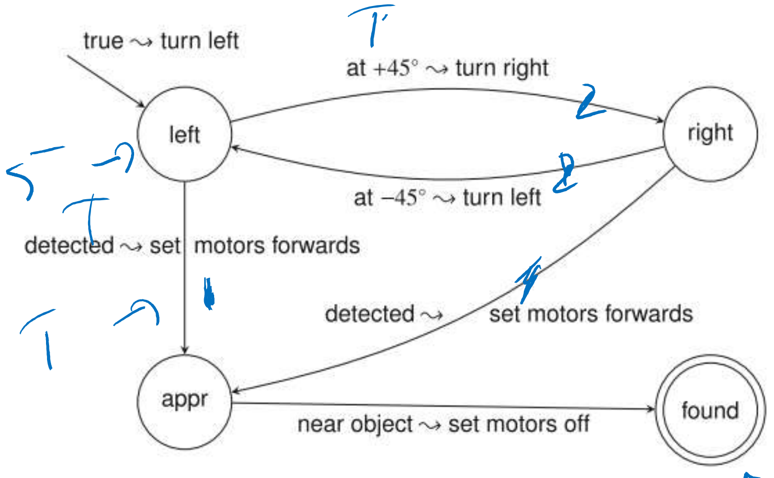
FSM for search & approach

left = robot turning left to search
right = robot turning right to search
appr = robot approaching object
found = robot found object

Final state

Search & Approach

- Infinite behaviour
  - – *Keep moving w/out stopping*
  - – *Ex: Persistent Braitenberg vehicle*
  - – *Same as "**toaster**"… why?*
    - Toast bread slice forever
- FSM
  - **(4) Nondeterminism**
  - *Any of outgoing transitions may be taken*
- Ex: states **left** & **right** has ***two***
  - *Reach edge of sector searched*
  - *Detect an object*


true ⇝ turn left
T'
at +45° ⇝ turn right
2
left
right
5
at −45° ⇝ turn left
8
T
detected ⇝ set motors forwards
T
4
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

FSM for search & approach

FS

* Object detected & search area not sector edge: transition **left/right** → state **appr** (1)
* No object detected & search area at sector edge: transition **left** → **right** or transition **right** → **left** (2)
* Object detected & search area at sector edge then arbitrary transition:
** (transition **left/right** → state **appr**) or (transition **left** → **right** or transition **right** → **left**)

Finite State Machines

- State Machines
- Reactive Behaviour with State
- Search & Approach
- Implementation of Finite State Machines

Implementation of Finite State Machines

- **Specification (Persistent)**

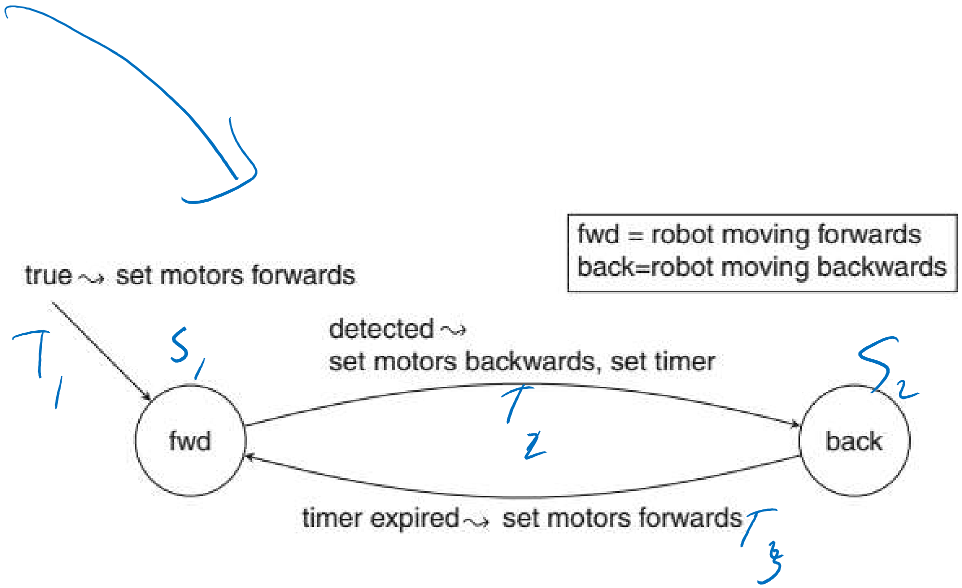
Algorithm 4.1: Persistent

- Variables used
  - *Implementing behaviours w/ states*
- Persistent vehicle
  - *Timer needed to cause an event*
- *Recall "**timer**"
  - *variable set to desired period of time*
- **Current** variable curr_state
  - *Current state of robot*
  - *Set to target state of transition at end of event handler processing*
  - *"**fwd**" & "**back**" for clarity*
    - Numerical values in computer

Implementation of Finite State Machines

- **Specification (Persistent)**

Algorithm 4.1: Persistent


fwd = robot moving forwards
back=robot moving backwards
true ⇝ set motors forwards
T_1
S_1
fwd
detected ⇝
set motors backwards, set timer
T_2
S_2
back
timer expired ⇝ set motors forwards
T_3

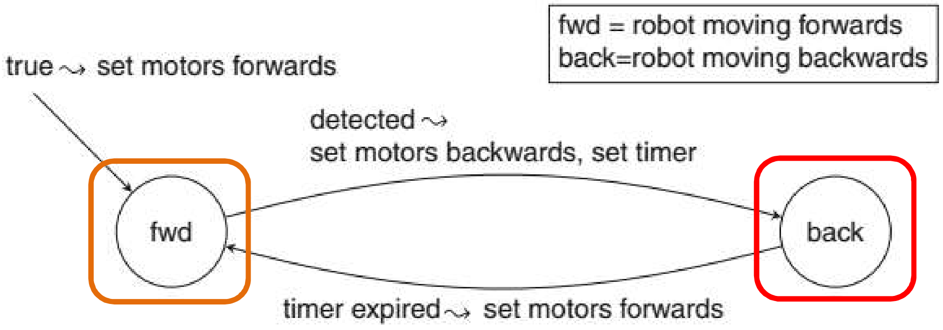
FSM for persistent Braitenberg vehicle

Implementation of Finite State Machines

- **Specification (Persistent)**

```
integer timer              // milliseconds
states current ← fwd
```

Algorithm 4.1: Persistent


fwd = robot moving forwards
back=robot moving backwards
true ⇝ set motors forwards
detected ⇝
set motors backwards, set timer
fwd
back
timer expired ⇝ set motors forwards

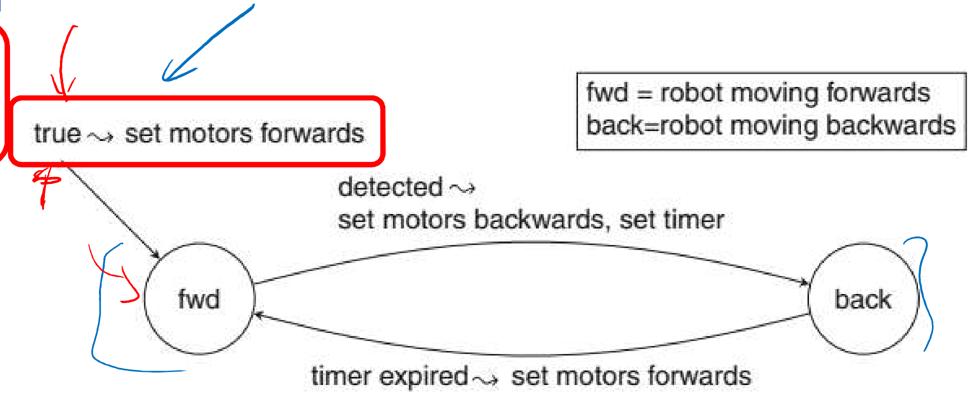
FSM for persistent Braitenberg vehicle

Implementation of Finite State Machines

- **Specification (Persistent)**

```
integer timer                    // milliseconds
states current ← fwd
left-motor-power ← 50
right-motor-power ← 50
loop   current ← fwd
```

Algorithm 4.1: Persistent


true ⇝ set motors forwards
detected ⇝
set motors backwards, set timer
fwd
back
timer expired ⇝ set motors forwards
fwd = robot moving forwards
back=robot moving backwards

FSM for persistent Braitenberg vehicle

Implementation of Finite State Machines

- **Specification (Persistent)**

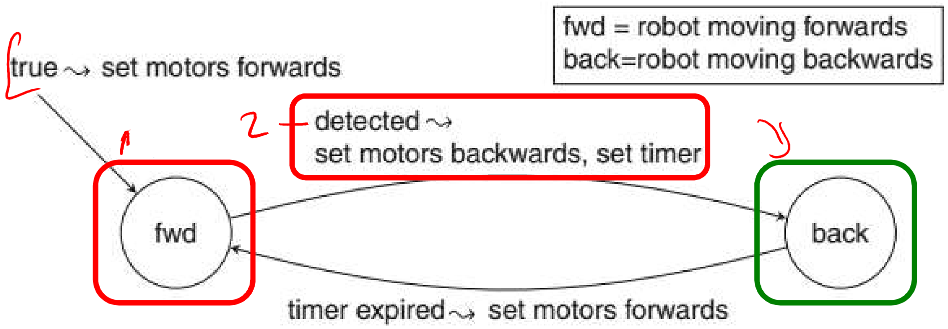
```
integer timer                    // milliseconds
states current ← fwd

left-motor-power ← 50
right-motor-power ← 50
loop
   when [current = fwd] and
        [object detected in front] 2 T2
      left-motor-power ← -50
      right-motor-power ← -50
      timer ← 1000
      current ← back
```

when curr = fwd & obj {

when curr = bwd & obj [illegible] }

Algorithm 4.1: Persistent

when object detected
if curr = fwd


fwd = robot moving forwards
back=robot moving backwards
true ⇝ set motors forwards
1
fwd
2
detected ⇝
set motors backwards, set timer
3)
back
timer expired ⇝ set motors forwards

FSM for persistent Braitenberg vehicle

if curr = bwd
}

Implementation of Finite State Machines

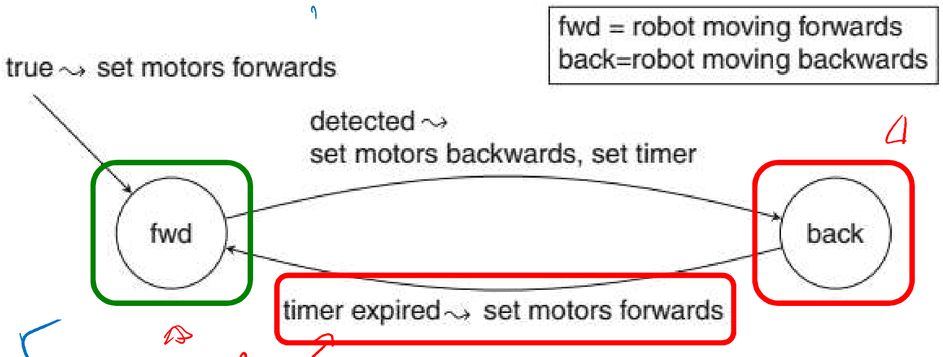
- **Specification (Persistent)**

4 x 4

4 4 4 4

```
integer timer                  // milliseconds
states current ← fwd

left-motor-power ← 50
right-motor-power ← 50
loop
   when current = fwd and
     object detected in front
      left-motor-power ← -50
      right-motor-power ← -50
      timer ← 1000
      current ← back
   when current = back and
     timer = 0
      left-motor-power ← 50
      right-motor-power ← 50
      current ← fwd
```

Algorithm 4.1: Persistent

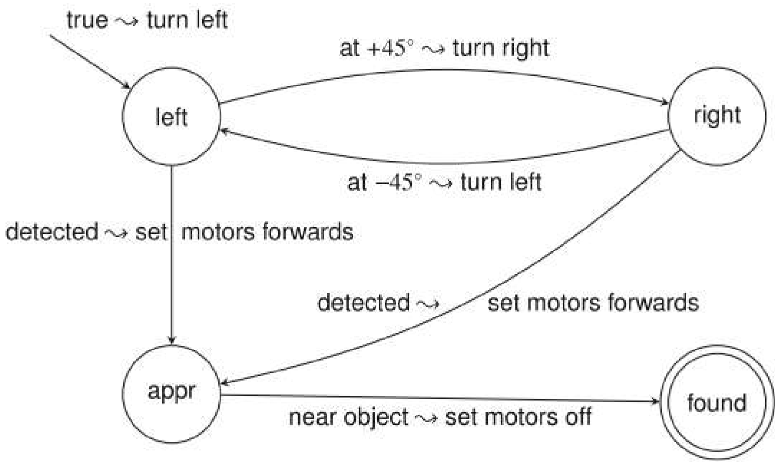

fwd = robot moving forwards
back=robot moving backwards
true ⇝ set motors forwards
detected ⇝
set motors backwards, set timer
fwd
back
timer expired ⇝ set motors forwards
4

2

FSM for persistent Braitenberg vehicle

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

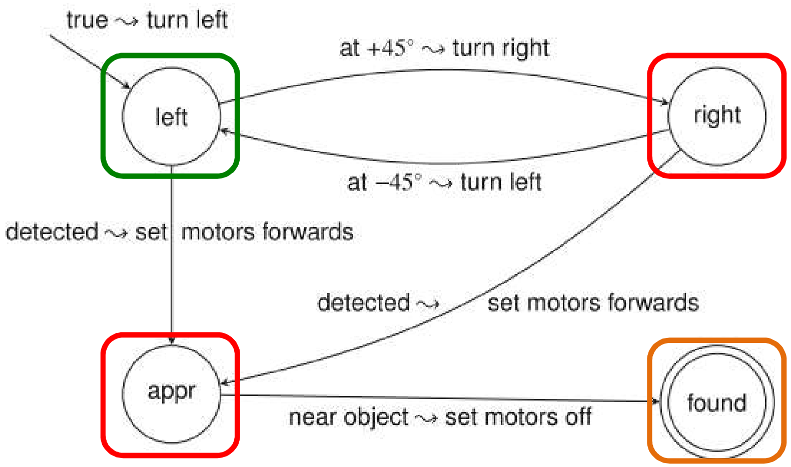

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

```
states current ← left
```

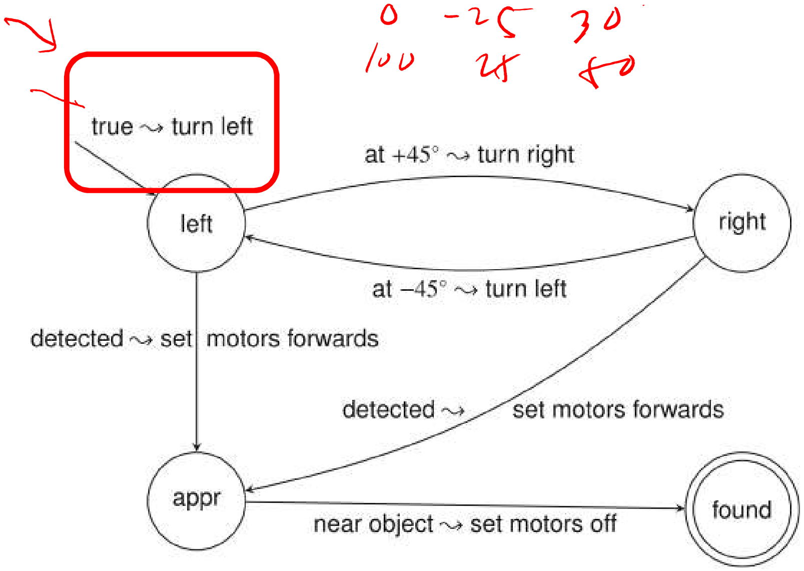

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

```
states current ← left
left-motor-power ← 25      // turn left  -25
right-motor-power ← 100                   25
loop ←
```


true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

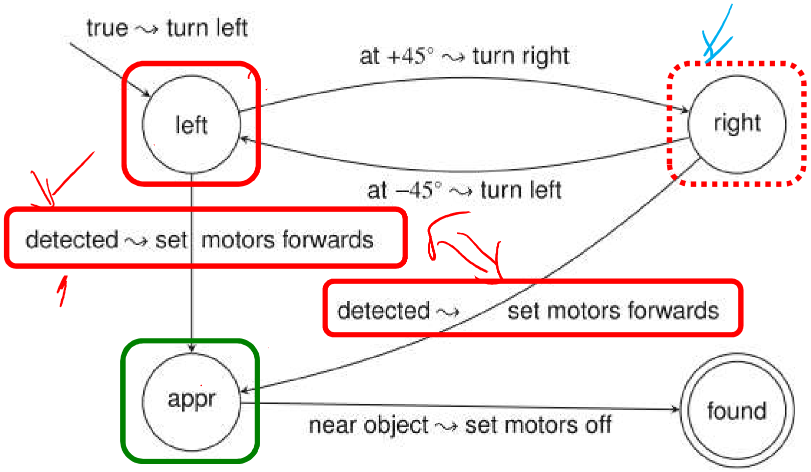
0 -25 30
100 25 80

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

```
states current ← left

left-motor-power ← 25        // turn left
right-motor-power ← 100
loop
   when object detected
     if current = left
        left-motor-power ← 100
        right-motor-power ← 100
        current ← appr
     else if current = right
        . . .
```

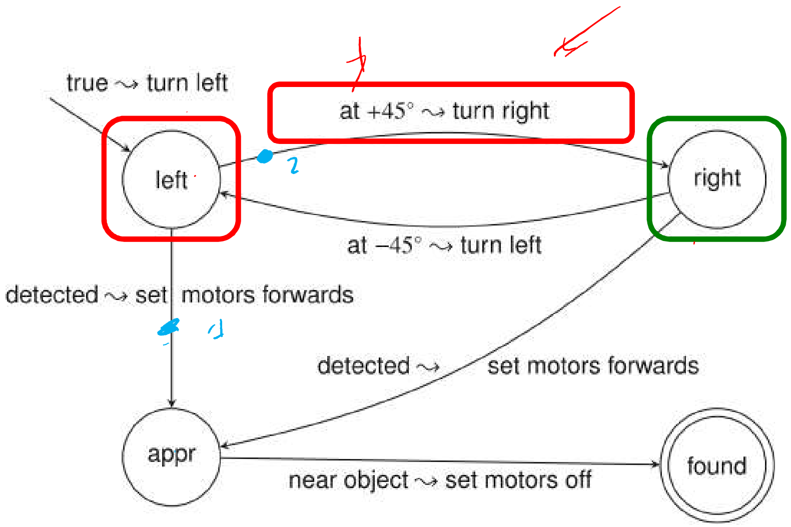

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

```
states current ← left
left-motor-power ← 25        // turn left
right-motor-power ← 100
loop
   when object detected
      if current = left
         left-motor-power ← 100        1
         right-motor-power ← 100
         current ← appr
      else if current = right
         . . .
   when at +45º
      if current = left   55
         left-motor-power ← 100 //turn right
         right-motor-power ← 50        2
         current ← right 7E 5
```

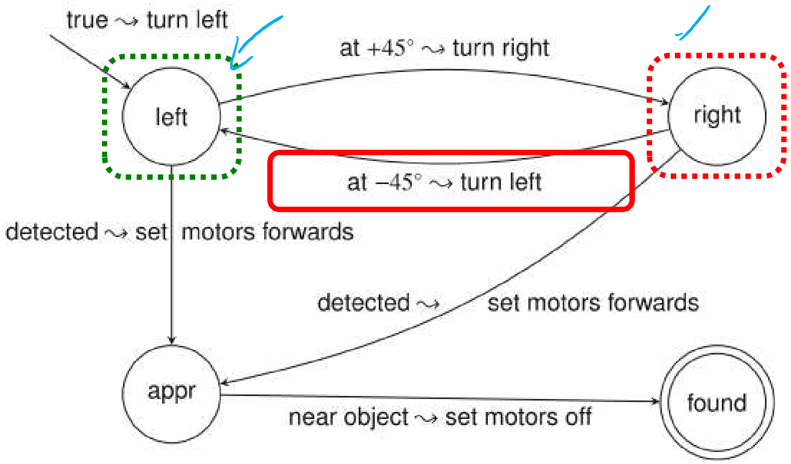

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off
2
1

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

```
states current ← left
when at -45°
   . . .
when object is very near
   if current = appr
      . . .
```

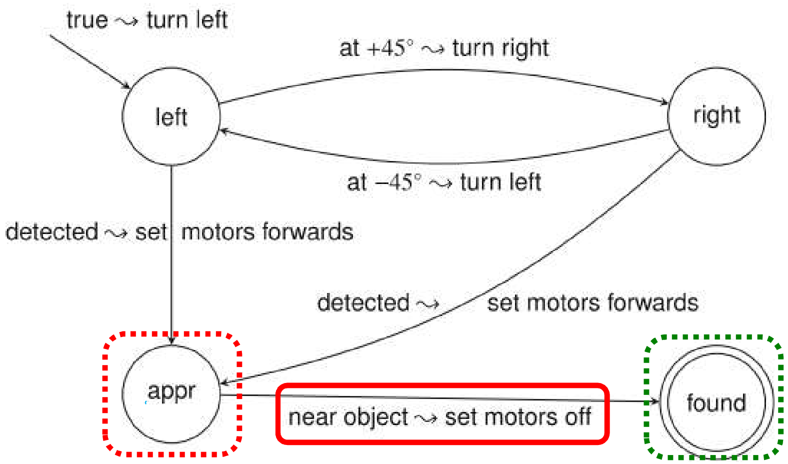

true ⇝ turn left
left
at +45° ⇝ turn right
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
near object ⇝ set motors off
found

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

```
states current ← left
when at -45°
   . . .
when object is very near
   if current = appr
      . . .
```


true ⤳ turn left
at +45° ⤳ turn right
left
right
at −45° ⤳ turn left
detected ⤳ set motors forwards
detected ⤳ set motors forwards
appr
near object ⤳ set motors off
found

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2: **Specification (Search & Approach)** (Algo Outline)

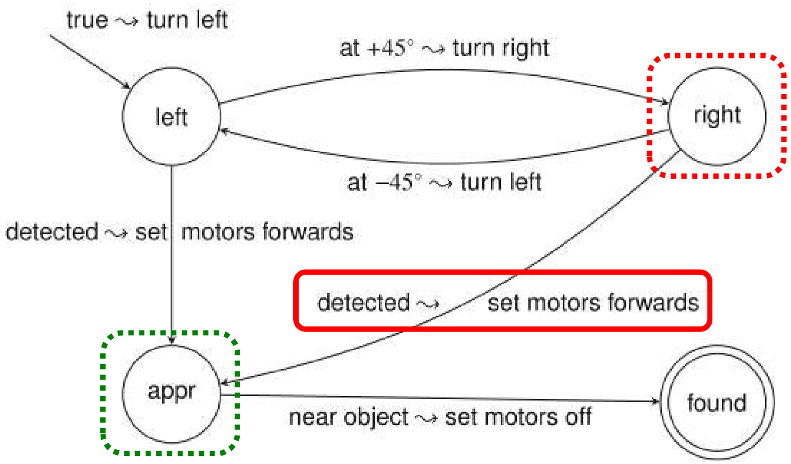
```
states current ← left

left-motor-power ← 25        // turn left
right-motor-power ← 100
loop
   when object detected
      if current = left
         left-motor-power ← 100
         right-motor-power ← 100
         current ← appr
      else if current = right
         . . .
   when at +45o
      if current = left
         left-motor-power ← 100 //turn right
         right-motor-power ← 50
         current ← right
   when at -45o
    ? . . .
   when object is very near
      if current = appr
       ? . . .
```

Algorithm 4.2: Search & Approach (Outline)

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left
left-motor-power ← 25      // turn left
right-motor-power ← 100
loop
   when object detected
      if current = left
         left-motor-power ← 100
         right-motor-power ← 100
         current ← appr
      else if current = right
         . . .
```

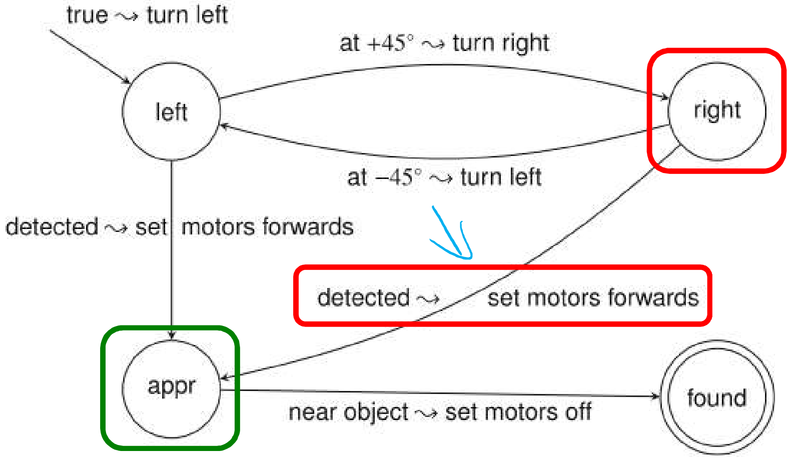

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
near object ⇝ set motors off
found

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left

left-motor-power ← 25      // turn left
right-motor-power ← 100
loop
  when object detected
    if current = left || current == right
      left-motor-power ← 100
      right-motor-power ← 100
      current ← appr
    else if current = right
      left-motor-power ← 100
      right-motor-power ← 100
      current ← appr
  when at +45°
    if current = left
      left-motor-power ← 100 // turn right
      right-motor-power ← 25
      current ← right
```

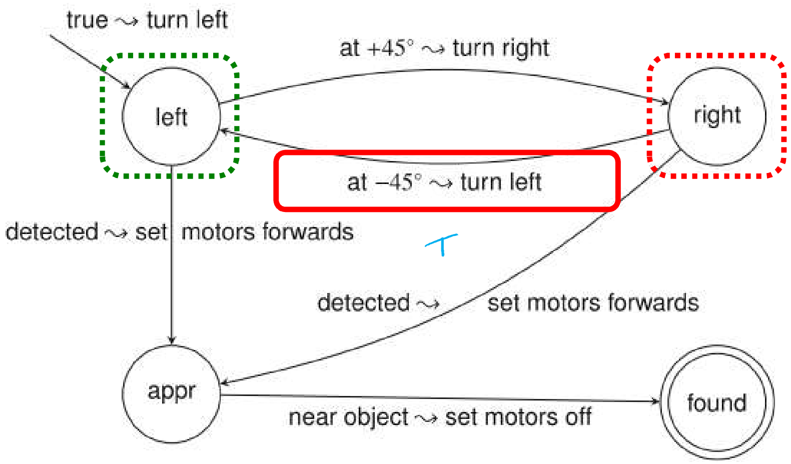

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
found
near object ⇝ set motors off

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left
when at -45º
  . . .
when object is very near
  if current = appr
    . . .
```

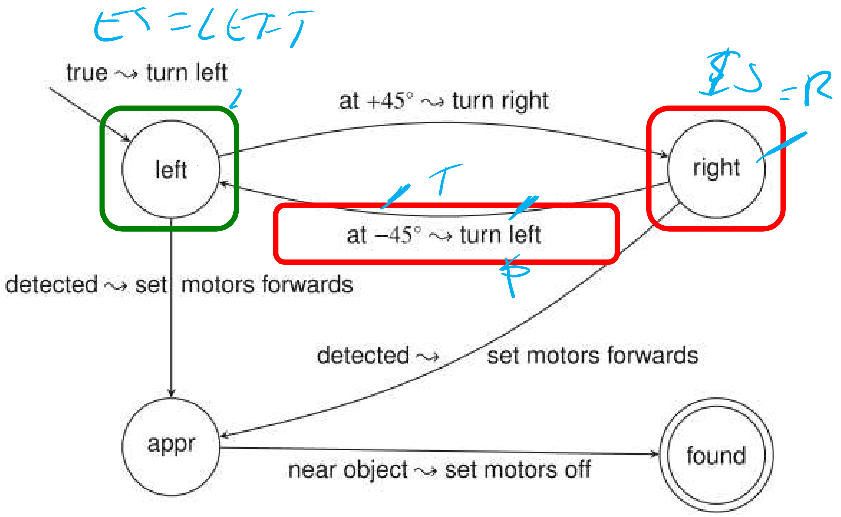

true ⤳ turn left
at +45° ⤳ turn right
left
right
at −45° ⤳ turn left
detected ⤳ set motors forwards
detected ⤳ set motors forwards
appr
found
near object ⤳ set motors off

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left

when at -45°
  if current = right
    left-motor-power ← 50  //turn left
    right-motor-power ← 100
    current ← ~~right~~ ← left
when object is very near
  if current = appr
    . . .
```

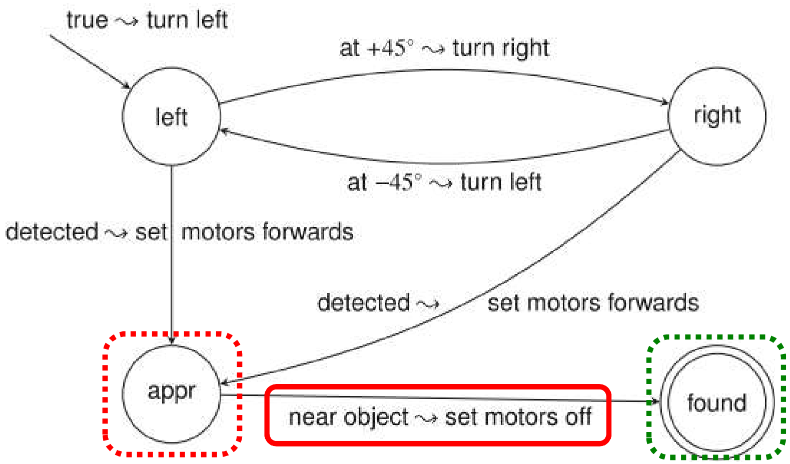


FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left
when at -45°
  if current = right
    left-motor-power ← 50  //turn left
    right-motor-power ← 100
    current ← right
when object is very near
  if current = appr
    . . .
```

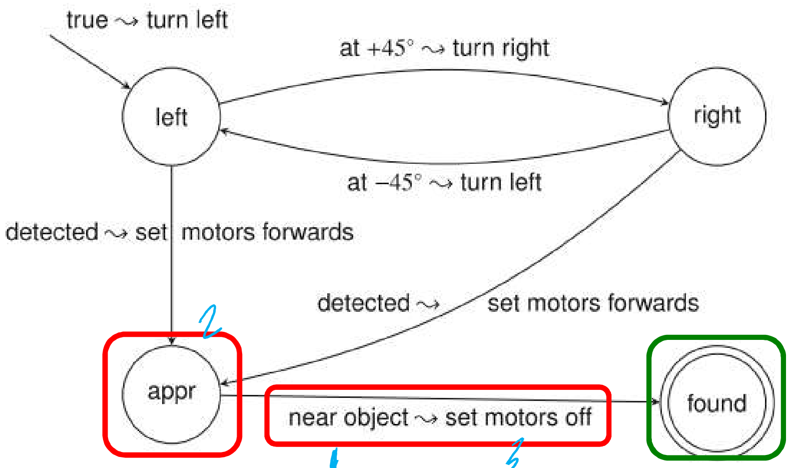

true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
appr
near object ⇝ set motors off
found

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left

when at -45º
  if current = right
    left-motor-power ← 50  //turn left
    right-motor-power ← 100
    current ← right
when object is very near 1
  if current = appr 2
    left-motor-power ← 0   // stop } 3
    right-motor-power ← 0
    current ← found (4)
```


true ⇝ turn left
at +45° ⇝ turn right
left
right
at −45° ⇝ turn left
detected ⇝ set motors forwards
detected ⇝ set motors forwards
2
appr
near object ⇝ set motors off
found
1
3

FSM for search & approach

Implementation of Finite State Machines

- Activity 4.2.a: **Specification(Search & Approach)** (Full Algo)

```
states current ← left

left-motor-power ← 25      // turn left
right-motor-power ← 100
loop
  when object detected
    if current = left
      left-motor-power ← 100
      right-motor-power ← 100
      current ← appr
    else if current = right
      left-motor-power ← 100
      right-motor-power ← 100
      current ← appr
  when at +45°
    if current = left
      left-motor-power ← 100 // turn right
      right-motor-power ← 25
      current ← right
  when at -45°
    if current = right
      left-motor-power ← 50  //turn left
      right-motor-power ← 100
      current ← right
  when object is very near
    if current = appr
      left-motor-power ← 0   // stop
      right-motor-power ← 0
      current ← found
```

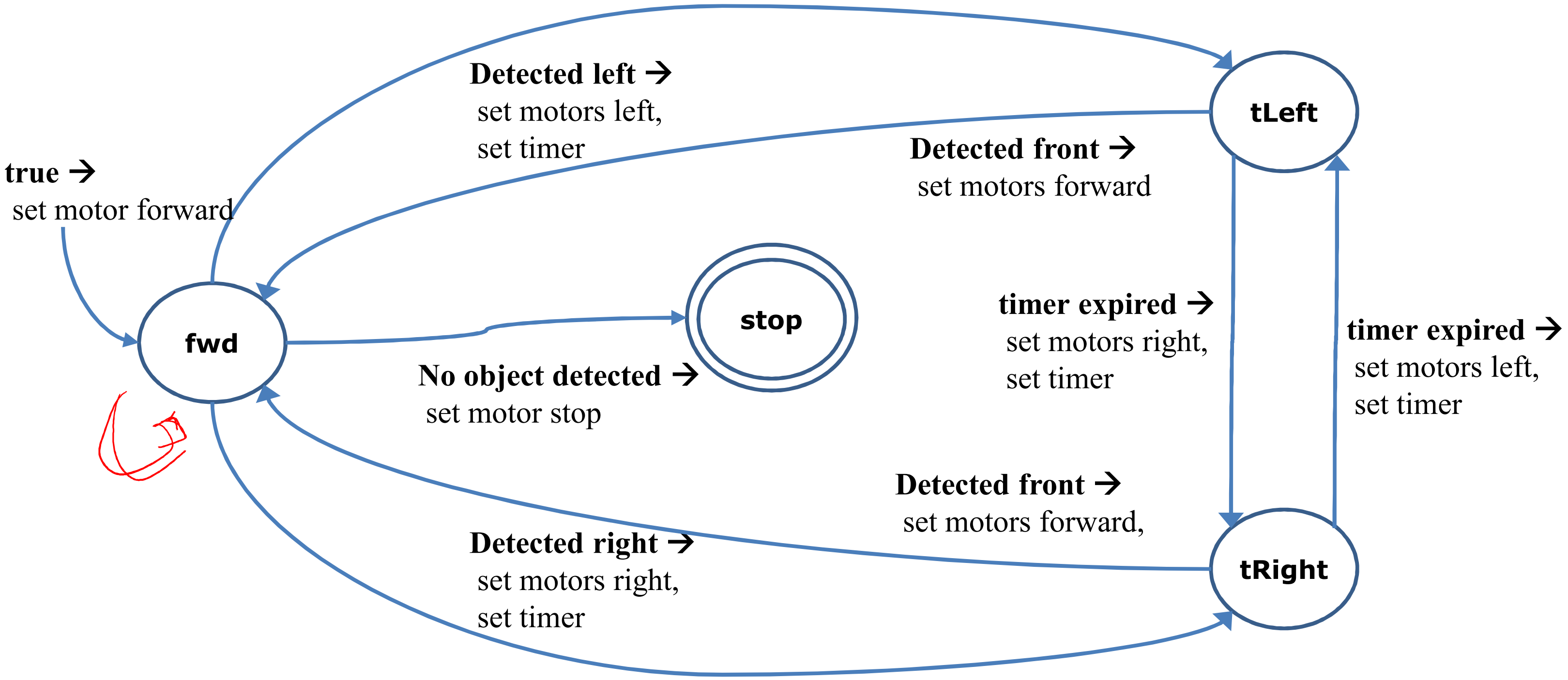
Algorithm 4.2: Search & Approach

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))
  - *Object detected in front → move forward* ✓
    Object detected at right → turn right ✓
    Object detected at left → turn left ✓
    Turning (even if no object detected) → alternate dir of turn every second ✓
    No object detected & not turning → robot stops
- Implement this specification
  - *Suggested variables:*
    - Current state, Turning direction
  - *Timer with one second period*
  - *For timer event handler*
    - Change direction to opposite direction
    - Reset timer

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))



```
fwd = robot move forward; stop = robot stop moving;
tLeft = robot turn left;  tRight = robot turn right;
```

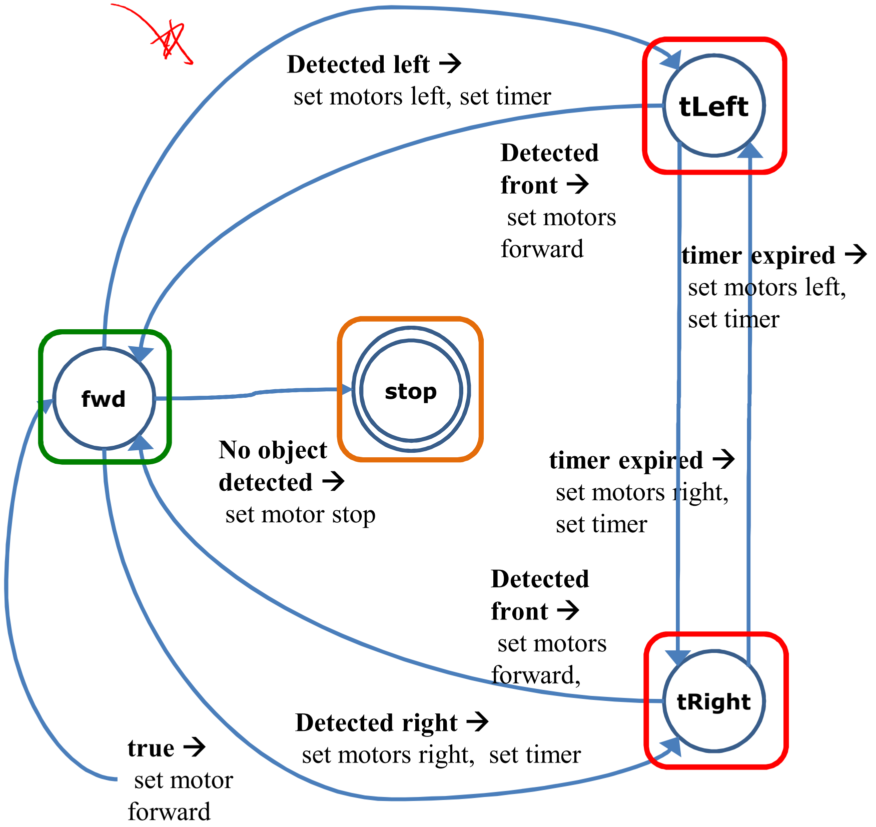
Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                    // milliseconds
states current ← fwd
direction turnDir
```

```
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
11:
12:
```

* Variables

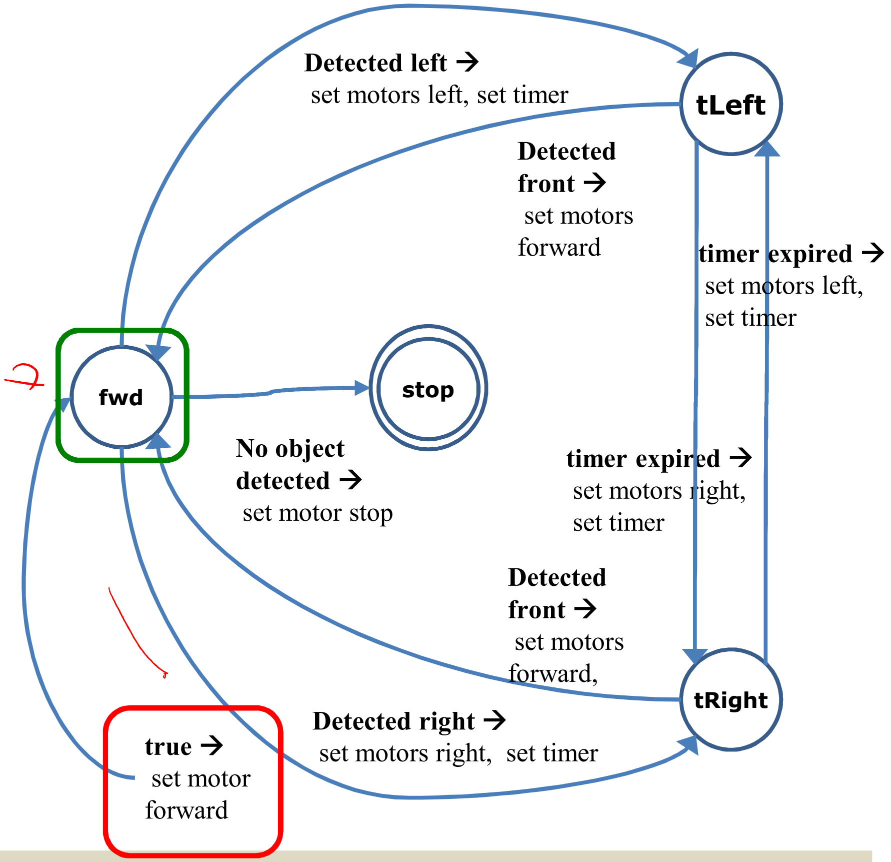


Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                    // milliseconds
states current ← fwd
direction turnDir
```

```
left-motor-power ← 50
right-motor-power ← 50
loop
```



Detected left → set motors left, set timer

Detected front → set motors forward

timer expired → set motors left, set timer

No object detected → set motor stop

timer expired → set motors right, set timer

Detected front → set motors forward,

Detected right → set motors right, set timer

true → set motor forward

*** Initial State**

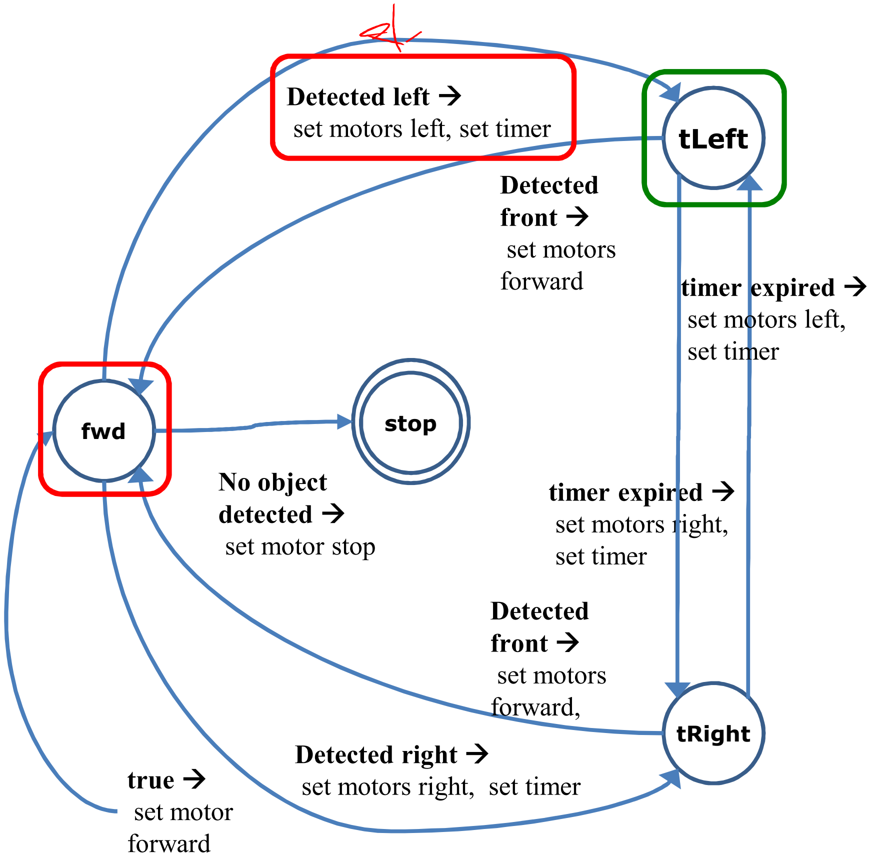
Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                // milliseconds
states current ← fwd
direction turnDir
```

```
left-motor-power ← 50
right-motor-power ← 50
loop
   when current = fwd and
     object detected in left
       left-motor-power ← -50
       right-motor-power ← 50
       current ← tLeft
       timer ← 1000
       turnDir ← left
```

*** fwd → tLeft**



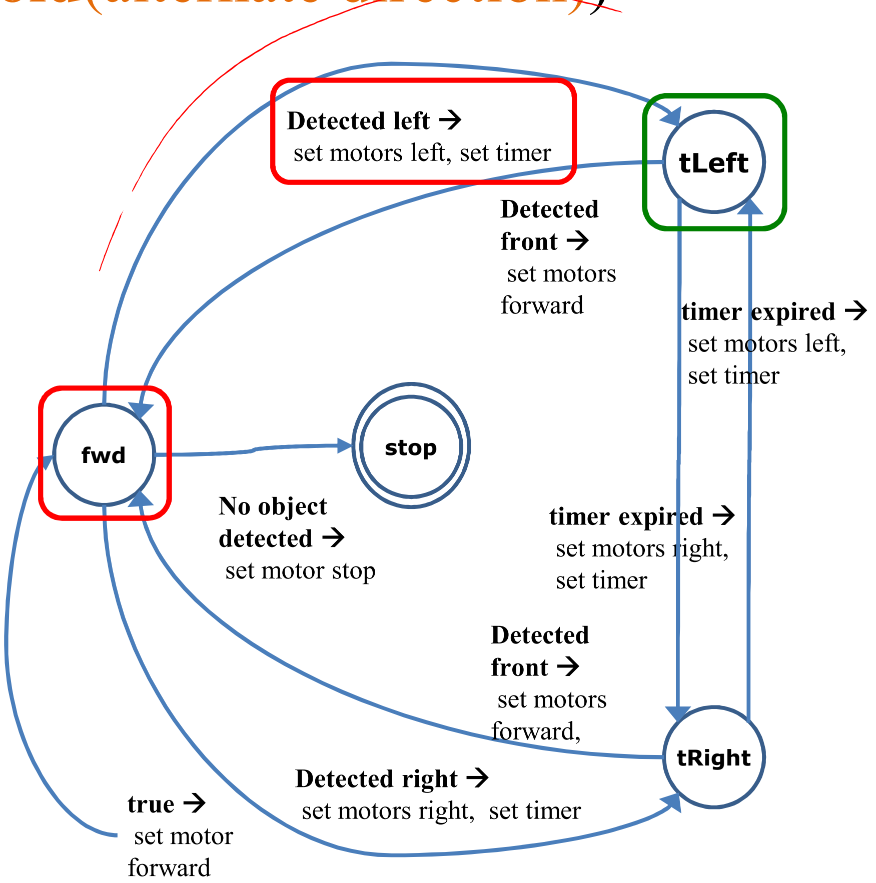
Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                    // milliseconds
states current ← fwd
direction turnDir
```

```
left-motor-power ← 50
right-motor-power ← 50
loop
   when current = fwd
      if object detected in left
         left-motor-power ← -50
         right-motor-power ← 50
         current ← tLeft
         timer ← 1000
         (turnDir ← left)
```

*** fwd → tLeft**



Detected left → set motors left, set timer

Detected front → set motors forward

timer expired → set motors left, set timer

No object detected → set motor stop

timer expired → set motors right, set timer

Detected front → set motors forward,

Detected right → set motors right, set timer

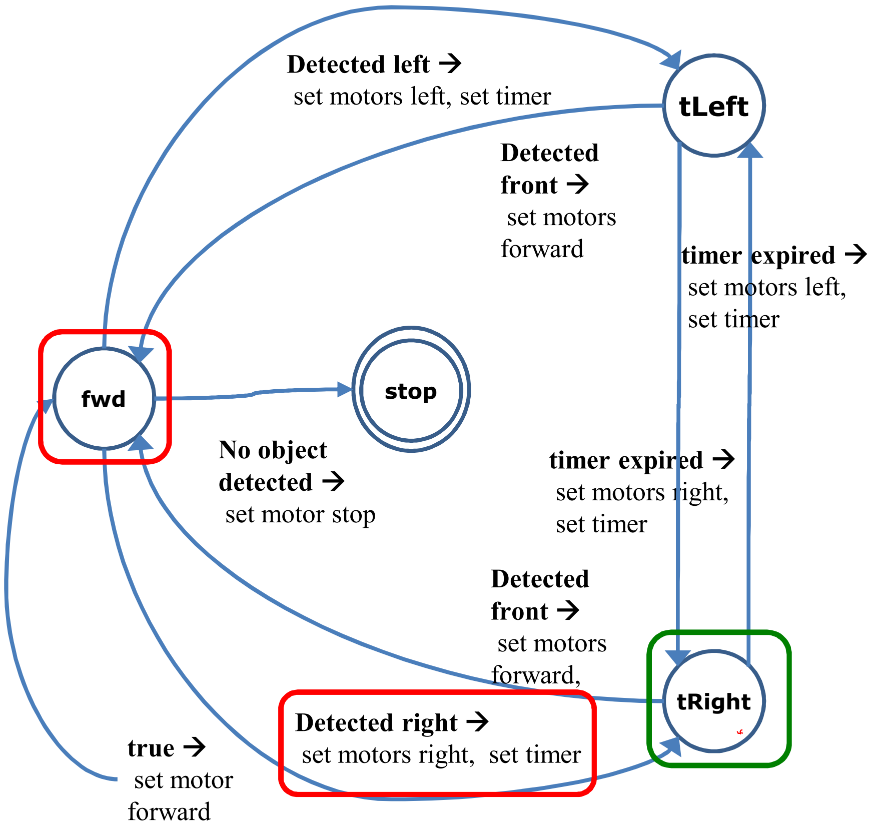
true → set motor forward

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                    // milliseconds
states current ← fwd
direction turnDir
```

```
left-motor-power ← 50
right-motor-power ← 50
loop
   when current = fwd
. . .       . . .
         else if object detected in right
            left-motor-power ← 50
            right-motor-power ← -50
            current ← tRight
            timer ← 1000
            turnDir ← right
```



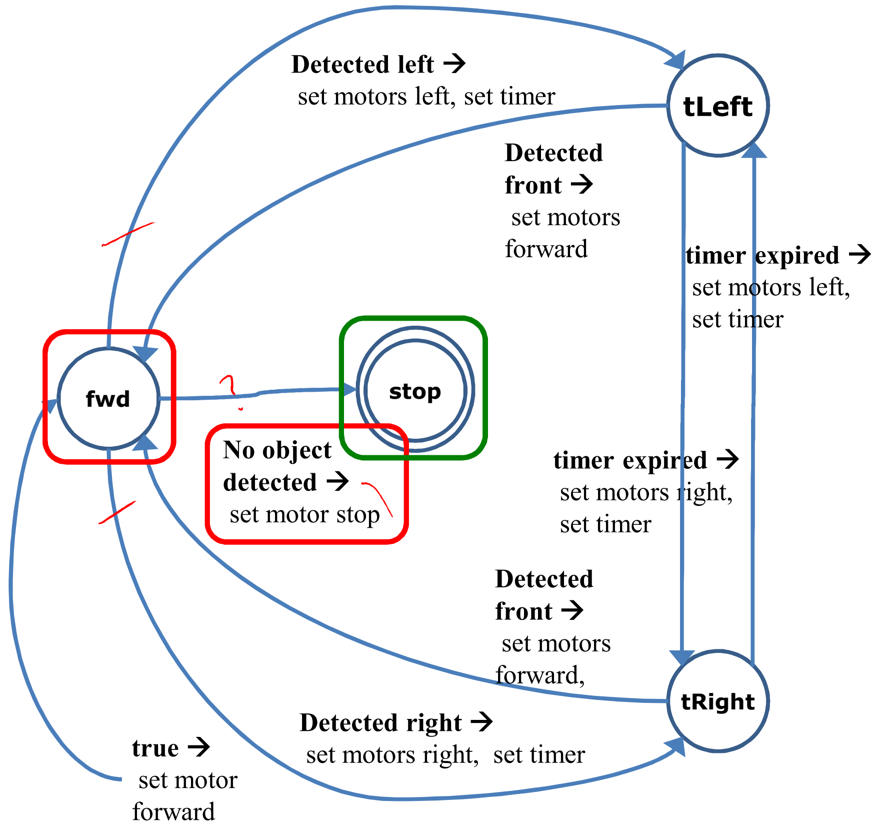
* fwd → tRight

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                    // milliseconds
states current ← fwd
direction turnDir
```

```
left-motor-power ← 50
right-motor-power ← 50
loop
   when current = fwd
. . .        . . .
        else if no object detected
           left-motor-power ← 0
           right-motor-power ← 0
           current ← stop
```



Detected left →
set motors left, set timer

tLeft

Detected front →
set motors forward

timer expired →
set motors left,
set timer

fwd

?

stop

No object detected →
set motor stop

timer expired →
set motors right,
set timer

Detected front →
set motors forward,

tRight

Detected right →
set motors right, set timer

true →
set motor forward

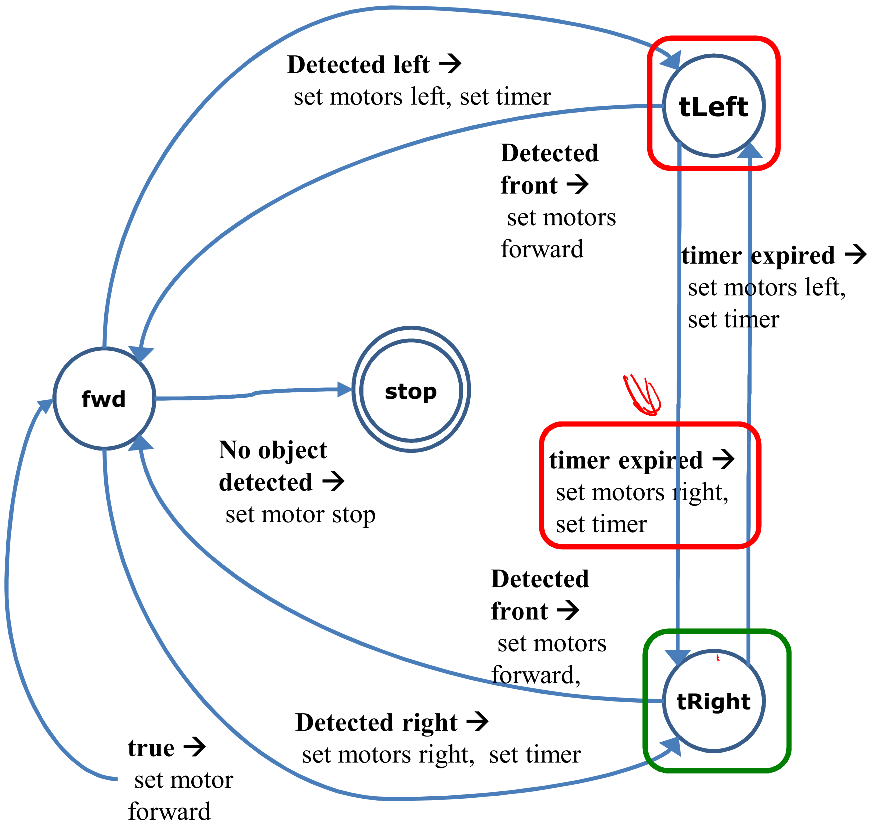
* fwd → stop

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
. . .
when current = tLeft
   if timer == 0
      if turnDir == left
         left-motor-power ← 50
         right-motor-power ← -50
         timer ← 1000
         turnDir ← right
         current ← tRight
      else
         left-motor-power ← -50
         right-motor-power ← 50
         timer ← 1000
         turnDir ← left
         current ← tLeft
```

* tLeft: timer Expires



Detected left → set motors left, set timer

Detected front → set motors forward

timer expired → set motors left, set timer

No object detected → set motor stop

timer expired → set motors right, set timer

Detected front → set motors forward,

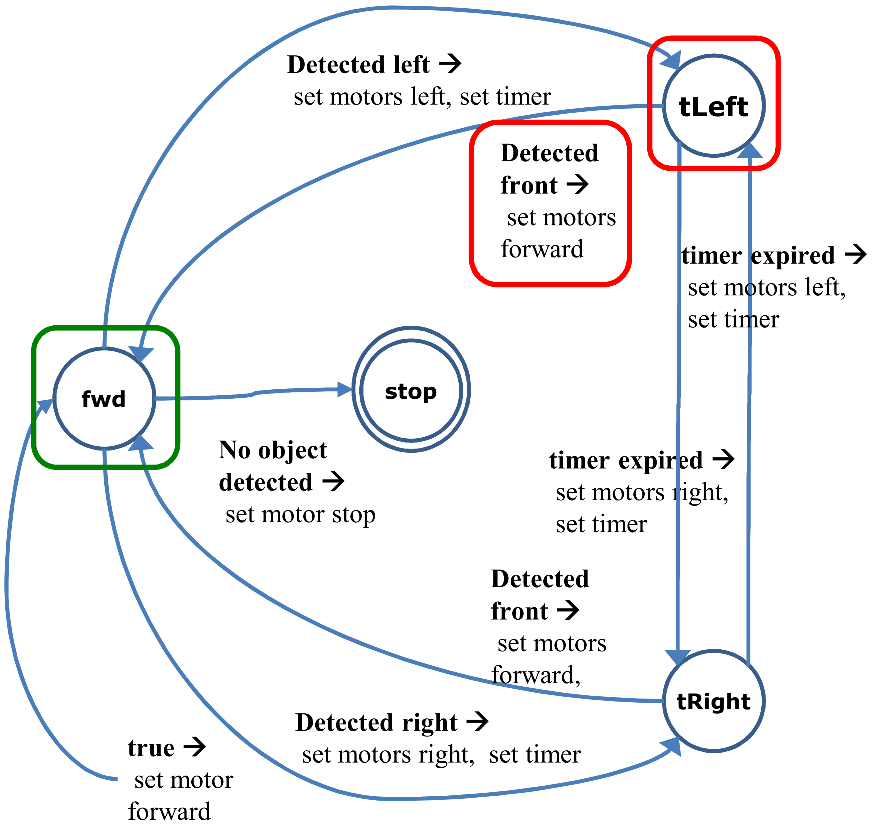
Detected right → set motors right, set timer

true → set motor forward

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
. . .
when current = tLeft
   . . .
   else if object detected in front
      left-motor-power ← 50
      right-motor-power ← 50
      current ← fwd
```

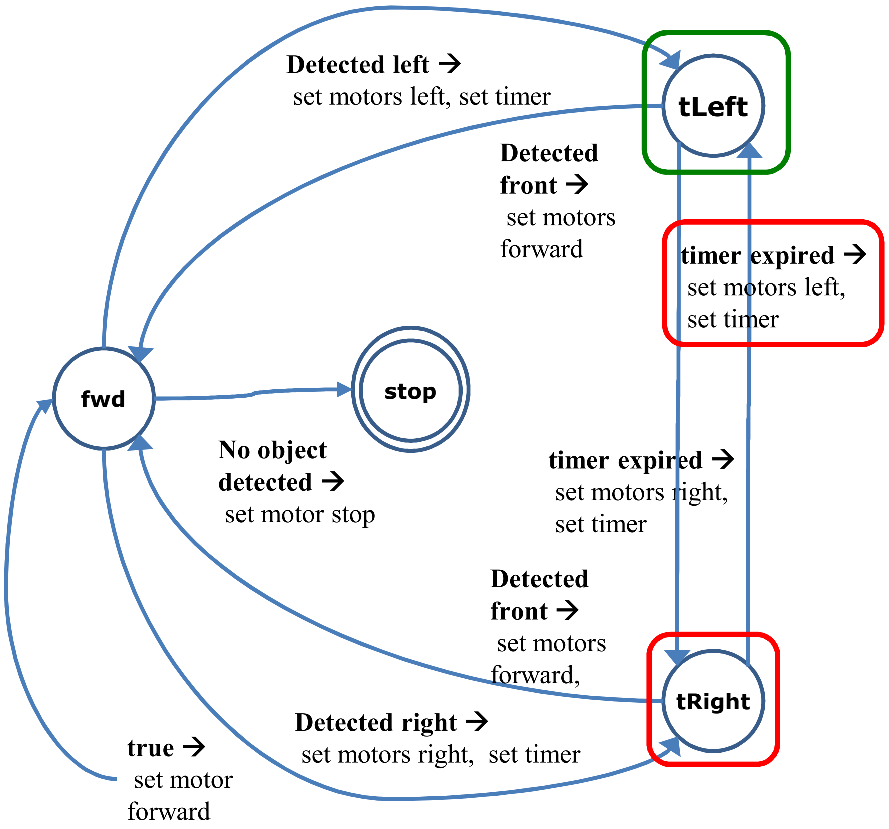


* tLeft → fwd

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
. . .
when current = tRight
   if timer == 0
      if turnDir == left
         left-motor-power ← 50
         right-motor-power ← -50
         timer ← 1000
         turnDir ← right
         current ← tRight
      else
         left-motor-power ← -50
         right-motor-power ← 50
         timer ← 1000
         turnDir ← left
         current ← tLeft
```

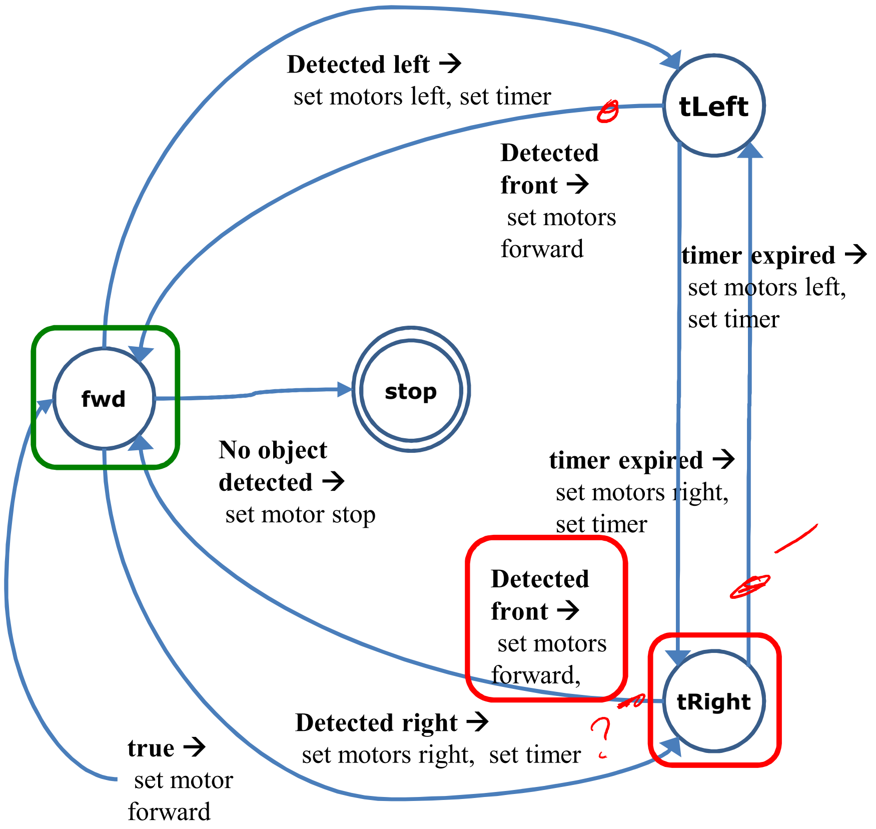

Detected left → set motors left, set timer
tLeft
Detected front → set motors forward
timer expired → set motors left, set timer
fwd
stop
No object detected → set motor stop
timer expired → set motors right, set timer
Detected front → set motors forward,
tRight
Detected right → set motors right, set timer
true → set motor forward

* tRight: timer Expires

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
. . .
when current = tRight
. . .
   . . .
   else if object detected in front
      left-motor-power ← 50
      right-motor-power ← 50
      current ← fwd
```



*** tRight → fwd**

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
integer timer                      // milliseconds
states current ← fwd
direction turnDir
```

```
left-motor-power ← 50              9 → Fwd
right-motor-power ← 50
loop
  when current = fwd
    if object detected in left
      left-motor-power ← -50
      right-motor-power ← 50       Fwd
      current ← tLeft              → Tleft
      timer ← 1000
      turnDir ← left
    else if object detected in right
      left-motor-power ← 50
      right-motor-power ← -50      Fwd
      current ← tRight             → TR
      timer ← 1000
      turnDir ← right
    else if object detected in right
      left-motor-power ← 50
      right-motor-power ← -50
      current ← tRight
      timer ← 1000
      turnDir ← right
    else if no object detected
      left-motor-power ← 0         Fwd
      right-motor-power ← 0
      current ← stop               → stop
```

Implementation of Finite State Machines

- Activity 4.3: **Specification** (**Paranoid**(alternate direction))

```
when current = tLeft
  if timer == 0
    if turnDir == left
      left-motor-power ← 50
      right-motor-power ← -50
      timer ← 1000
      turnDir ← right
      current ← tRight
    else
      left-motor-power ← -50
      right-motor-power ← 50
      timer ← 1000
      turnDir ← left
      current ← tLeft
  else if object detected in front
    left-motor-power ← 50
    right-motor-power ← 50
    current ← fwd
```

Sn

(Cn)

(C,F)

```
when current = tRight
  if timer == 0
    if turnDir == left
      left-motor-power ← 50
      right-motor-power ← -50
      timer ← 1000
      turnDir ← right
      current ← tRight
    else
      left-motor-power ← -50
      right-motor-power ← 50
      timer ← 1000
      turnDir ← left
      current ← tLeft
  else if object detected in front
    left-motor-power ← 50
    right-motor-power ← 50
    current ← fwd
```

Summary

- Maintain internal representation of its current state
  - *Required by most robotic algorithms*
- Finite State Machines
  - *Describe conditions used to decide when to change state & actions*
- State variables
  - *Implement state machines in programs*

STATE
TRANSITIONS

FS

Thank you.