Autonomous Drone using JESS

CS-514 Project-1

-The Turingator

The Domain: - Autonomous Drone

This is a drone that can run on the road and fly in the sky. The drone is started by moving forward, after that it'll check the surroundings to make sure there are no obstacles. Every time I am taking an input from the user to enter the "Battery-Level" and "Fuel-Level". The maximum value for them is 100% which means battery is fully charged and the fuel tank is completely full. The minimum values for "battery and fuel level" is 10, to keep the drone running. If the user enters a value less than or equal to "10", then the drone stops.

The drone starts moving forward, it then sees if there is an obstacle in front, defined by "FALSE" Boolean value, any other directions which has a Boolean value of "TRUE" is asserted and the drone moves in that direction. To keep it simple the order of movement is: - front, left, right, reverse, & fly. So if the drone is cornered from all the directions when it is running on the road, it will start flying. The same order of movements are held in the air, i.e. Front, left, right, reverse. Usually in a real world scenario, a drone will check for hundreds of conditions like water-level, heat, battery, fuel, etc. And each one of them is received by a sensor. To keep it simple I am asking the user to enter the battery and fuel levels.

NOTE: - Please give the battery and fuel level values as an integer from 1 - 100, do not add % in the end. Just the numbers.

I have three templates: - **"Movement", "Fly" & "Stop".**

```
(deftemplate Movement (slot front) (slot left) (slot right)
   (slot reverse) (slot fly) (slot battery (default 100)) (slot fuel (default 100)))
(deftemplate Stop (slot front) (slot left) (slot right)
   (slot reverse) (slot fly) (slot battery (default 100)) (slot fuel (default 100)))
(deftemplate Fly (slot front) (slot left) (slot right)
   (slot reverse) (slot fly) (slot battery (default 100)) (slot fuel (default 100)))
```

There are two modules: - **startDrone and afterStart.**

startDrone has the initial rule to move forward, all other rules are in afterStart.

My knowledge base is as given below.

```
(defmodule startDrone)
(defrule moveForward1
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE))
   =>
   (if (eq (enterBatteryAndFuel "move forward") TRUE) then
   (printout t crlf "Drone is moving forward" crlf)
   (assert (Movement (front FALSE) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defmodule afterStart)


(defrule moveLeft1
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take left") TRUE) then
   (printout t crlf "Drone is taking a left turn" crlf)
   (assert (Movement (front TRUE) (left FALSE) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))


(defrule moveLeft2
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left TRUE) (right TRUE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take left") TRUE) then
   (printout t crlf "Drone is taking a left turn" crlf)
   (assert (Movement (front ?p.front) (left FALSE) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule moveLeft3
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left TRUE) (right FALSE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take left") TRUE) then
   (printout t crlf "Drone is taking a left turn" crlf)
   (assert (Movement (front ?p.front) (left FALSE) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule moveLeft4
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (left TRUE) (right TRUE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take left") TRUE) then
   (printout t crlf "Drone is taking a left turn" crlf)
   (assert (Movement (front ?p.front) (left FALSE) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))
```

```
(defrule moveRight1
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left FALSE) (right TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take right") TRUE) then
   (printout t crlf "Drone is taking a right turn" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right FALSE) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule moveRight2
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left FALSE) (right TRUE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take right") TRUE) then
   (printout t crlf "Drone is taking a right turn" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right FALSE) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule moveRight3
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left FALSE) (right TRUE) (reverse FALSE))
   =>
   (if (eq (enterBatteryAndFuel "take right") TRUE) then
   (printout t crlf "Drone is taking a right turn" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right FALSE) (reverse ?p.reverse)
          (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
          (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
     (halt)))

(defrule moveRight4
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (left TRUE) (right TRUE) (reverse FALSE))
   =>
   (if (eq (enterBatteryAndFuel "take right") TRUE) then
   (printout t crlf "Drone is taking a right turn" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right FALSE) (reverse ?p.reverse)
          (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
          (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
     (halt)))

(defrule takeReverse1
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left FALSE) (right FALSE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take reverse") TRUE) then
   (printout t crlf "Drone is taking reverse" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right ?p.right) (reverse FALSE)
          (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule takeReverse2
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (left FALSE) (right FALSE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take reverse") TRUE) then
   (printout t crlf "Drone is taking reverse" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right ?p.right) (reverse FALSE)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule takeReverse3
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (left FALSE) (right TRUE) (reverse TRUE))
   =>
   (if (eq (enterBatteryAndFuel "take reverse") TRUE) then
   (printout t crlf "Drone is taking reverse" crlf)
   (assert (Movement (front ?p.front) (left ?p.left) (right ?p.right) (reverse FALSE)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule flyUp1
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly
TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly up in the air!") TRUE) then
   (printout t crlf "Drone is flying up!" crlf)
   (assert (Fly (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule flyUp2
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left TRUE) (right FALSE) (reverse TRUE) (fly
TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly up in the air!") TRUE) then
   (printout t crlf "Drone is flying up!" crlf)
   (assert (Fly (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule flyUp3
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (left FALSE) (right FALSE) (reverse TRUE) (fly
TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly up in the air!") TRUE) then
   (printout t crlf "Drone is flying up!" crlf)
   (assert (Fly (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule flyUp4
   ?p <- (Movement {battery > 10 && fuel > 10} (front FALSE) (left TRUE) (right TRUE) (reverse FALSE) (fly
TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly up in the air!") TRUE) then
   (printout t crlf "Drone is flying up!" crlf)
   (assert (Fly (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule changeToLeftLane
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (left TRUE))
   =>
   (if (eq (enterBatteryAndFuel "change to left lane") TRUE) then
   (printout t crlf "Drone has changed lanes to left" crlf)
   (assert (Movement (front FALSE) (left FALSE) (right ?p.left) (reverse ?p.left)
            (fly ?p.left) (battery ?*b*) (fuel ?*f*)))
   else
```

```
    (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
                (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
    (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
        (halt)))

(defrule changeToRightLane
    ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (right TRUE))
    =>
    (if (eq (enterBatteryAndFuel "change to right lane") TRUE) then
    (printout t crlf "Drone has changed lanes to right" crlf)
    (assert (Movement (front FALSE) (left ?p.left) (right FALSE) (reverse ?p.reverse)
                (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
    else
    (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
                (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
    (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
        (halt)))

(defrule flyToLeft
    ?p <- (Movement {battery > 10 && fuel > 10} (left TRUE) (fly TRUE))
    =>
    (if (eq (enterBatteryAndFuel "fly to left") TRUE) then
    (printout t crlf "Drone is flying to left" crlf)
    (assert (Fly (front ?p.front) (left FALSE) (right ?p.right) (reverse ?p.reverse)
                (fly FALSE) (battery ?*b*) (fuel ?*f*)))
    else
    (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
                (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
    (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
        (halt)))
```

```
(defrule flyToRight
   ?p <- (Movement {battery > 10 && fuel > 10} (right TRUE) (fly TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly to right") TRUE) then
   (printout t crlf "Drone is flying to right" crlf)
   (assert (Fly (front ?p.front) (left ?p.left) (right FALSE) (reverse ?p.reverse)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule flyForward
   ?p <- (Movement {battery > 10 && fuel > 10} (front TRUE) (fly TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly forward") TRUE) then
   (printout t crlf "Drone is flying to right" crlf)
   (assert (Fly (front FALSE) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))

(defrule flyBackwards
   ?p <- (Movement {battery > 10 && fuel > 10} (reverse TRUE) (fly TRUE))
   =>
   (if (eq (enterBatteryAndFuel "fly backwards") TRUE) then
   (printout t crlf "Drone is flying to right" crlf)
   (assert (Fly (front ?p.front) (left ?p.left) (right ?p.right) (reverse FALSE)
            (fly FALSE) (battery ?*b*) (fuel ?*f*)))
   else
   (assert (Stop (front ?p.front) (left ?p.left) (right ?p.right) (reverse ?p.reverse)
            (fly ?p.fly) (battery ?*b*) (fuel ?*f*)))
   (printout t crlf "Battery or Fuel Low, Drone stopping!" crlf)
      (halt)))
```

I have one function to take the input from the user for battery and fuel levels: - **enterBatteryAndFuel**.

```
(deffunction enterBatteryAndFuel(?action)
   (printout t crlf "Drone is trying to " ?action crlf)
   (printout t "Please enter Battery-level in terms of percentage, for 15% enter only 15" crlf)
   (bind ?*b* (read) t)
   (printout t "Please enter fuel-level in terms of percentage, for 15% enter only 15" crlf)
   (bind ?*f* (read) t)
   (if (and (> ?*b* 10) (> ?*f* 10)) then
      (return TRUE)
      else
      (return FALSE)))
```

I start my drone with "100" battery and fuel values. These are given as the default values in the template. When the drone is running on the road, if the current fact is new it'll be asserted. The template "Movement" is asserted for drone's movement on road, "fly" while the drone is flying, and "stop" when the drone comes to a halt.

Test cases: -

Test case 1: -

```
Drone is trying to move forward

Please enter Battery-level in terms of percentage, for 15% enter only 15

100

Please enter fuel-level in terms of percentage, for 15% enter only 15

100
```

```
Drone is moving forward

 ==> f-2 (MAIN::Movement (front FALSE) (left TRUE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 100) (fuel
100))


Drone is trying to take left

Please enter Battery-level in terms of percentage, for 15% enter only 15

97

Please enter fuel-level in terms of percentage, for 15% enter only 15

96


Drone is taking a left turn

 ==> f-3 (MAIN::Movement (front TRUE) (left FALSE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 97) (fuel
96))


Drone is trying to change to right lane

Please enter Battery-level in terms of percentage, for 15% enter only 15

95

Please enter fuel-level in terms of percentage, for 15% enter only 15

93


Drone has changed lanes to right

 ==> f-4 (MAIN::Movement (front FALSE) (left FALSE) (right FALSE) (reverse TRUE) (fly TRUE) (battery 95) (fuel
93))


Drone is trying to take reverse

Please enter Battery-level in terms of percentage, for 15% enter only 15

88

Please enter fuel-level in terms of percentage, for 15% enter only 15

81


Drone is taking reverse
```

```
 ==> f-5 (MAIN::Movement (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly TRUE) (battery 88) (fuel
81))


Drone is trying to fly up in the air!

Please enter Battery-level in terms of percentage, for 15% enter only 15

76

Please enter fuel-level in terms of percentage, for 15% enter only 15

65


Drone is flying up!

 ==> f-6 (MAIN::Fly (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly FALSE) (battery 76) (fuel 65))


Drone is trying to fly backwards

Please enter Battery-level in terms of percentage, for 15% enter only 15

55

Please enter fuel-level in terms of percentage, for 15% enter only 15

45


Drone is flying to right

 ==> f-7 (MAIN::Fly (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly FALSE) (battery 55) (fuel 45))


Drone is trying to fly backwards

Please enter Battery-level in terms of percentage, for 15% enter only 15

45

Please enter fuel-level in terms of percentage, for 15% enter only 15

34


Drone is flying to right

 ==> f-8 (MAIN::Fly (front TRUE) (left FALSE) (right TRUE) (reverse FALSE) (fly FALSE) (battery 45) (fuel 34))


Drone is trying to take reverse
```

```
Please enter Battery-level in terms of percentage, for 15% enter only 15

23

Please enter fuel-level in terms of percentage, for 15% enter only 15

13


Drone is taking reverse

 ==> f-9 (MAIN::Movement (front TRUE) (left FALSE) (right TRUE) (reverse FALSE) (fly TRUE) (battery 23) (fuel
13))


Drone is trying to fly to right

Please enter Battery-level in terms of percentage, for 15% enter only 15

11

Please enter fuel-level in terms of percentage, for 15% enter only 15

11


Drone is flying to right

 ==> f-10 (MAIN::Fly (front TRUE) (left FALSE) (right FALSE) (reverse FALSE) (fly FALSE) (battery 11) (fuel 11))


Drone is trying to fly forward

Please enter Battery-level in terms of percentage, for 15% enter only 15

10

Please enter fuel-level in terms of percentage, for 15% enter only 15

10

 ==> f-11 (MAIN::Stop (front TRUE) (left FALSE) (right TRUE) (reverse FALSE) (fly TRUE) (battery 10) (fuel 10))


Battery or Fuel Low, Drone stopping!
```

Test case 2: -

```
Drone is trying to move forward

Please enter Battery-level in terms of percentage, for 15% enter only 15

56
```

```
Please enter fuel-level in terms of percentage, for 15% enter only 15

23


Drone is moving forward

 ==> f-2 (MAIN::Movement (front FALSE) (left TRUE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 56) (fuel
23))


Drone is trying to take left

Please enter Battery-level in terms of percentage, for 15% enter only 15

45

Please enter fuel-level in terms of percentage, for 15% enter only 15

22


Drone is taking a left turn

 ==> f-3 (MAIN::Movement (front TRUE) (left FALSE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 45) (fuel
22))


Drone is trying to change to right lane

Please enter Battery-level in terms of percentage, for 15% enter only 15

43

Please enter fuel-level in terms of percentage, for 15% enter only 15

21


Drone has changed lanes to right

 ==> f-4 (MAIN::Movement (front FALSE) (left FALSE) (right FALSE) (reverse TRUE) (fly TRUE) (battery 43) (fuel
21))


Drone is trying to take reverse

Please enter Battery-level in terms of percentage, for 15% enter only 15

32

Please enter fuel-level in terms of percentage, for 15% enter only 15

19
```

```
Drone is taking reverse

 ==> f-5 (MAIN::Movement (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly TRUE) (battery 32) (fuel
19))


Drone is trying to fly up in the air!

Please enter Battery-level in terms of percentage, for 15% enter only 15

31

Please enter fuel-level in terms of percentage, for 15% enter only 15

17


Drone is flying up!

 ==> f-6 (MAIN::Fly (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly FALSE) (battery 31) (fuel 17))


Drone is trying to fly backwards

Please enter Battery-level in terms of percentage, for 15% enter only 15

28

Please enter fuel-level in terms of percentage, for 15% enter only 15

16


Drone is flying to right

 ==> f-7 (MAIN::Fly (front FALSE) (left FALSE) (right FALSE) (reverse FALSE) (fly FALSE) (battery 28) (fuel 16))


Drone is trying to fly backwards

Please enter Battery-level in terms of percentage, for 15% enter only 15

25

Please enter fuel-level in terms of percentage, for 15% enter only 15

13


Drone is flying to right

 ==> f-8 (MAIN::Fly (front TRUE) (left FALSE) (right TRUE) (reverse FALSE) (fly FALSE) (battery 25) (fuel 13))
```

```
Drone is trying to take reverse

Please enter Battery-level in terms of percentage, for 15% enter only 15

22

Please enter fuel-level in terms of percentage, for 15% enter only 15

11


Drone is taking reverse

 ==> f-9 (MAIN::Movement (front TRUE) (left FALSE) (right TRUE) (reverse FALSE) (fly TRUE) (battery 22) (fuel
11))


Drone is trying to fly to right

Please enter Battery-level in terms of percentage, for 15% enter only 15

19

Please enter fuel-level in terms of percentage, for 15% enter only 15

10

 ==> f-10 (MAIN::Stop (front TRUE) (left FALSE) (right TRUE) (reverse FALSE) (fly TRUE) (battery 19) (fuel 10))


Battery or Fuel Low, Drone stopping!

For a total of 0 facts in module afterStart.
```

Test case 3: -

```
Drone is trying to move forward

Please enter Battery-level in terms of percentage, for 15% enter only 15

15

Please enter fuel-level in terms of percentage, for 15% enter only 15

12


Drone is moving forward

 ==> f-2 (MAIN::Movement (front FALSE) (left TRUE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 15) (fuel
12))
```

```
Drone is trying to take left

Please enter Battery-level in terms of percentage, for 15% enter only 15

10

Please enter fuel-level in terms of percentage, for 15% enter only 15

10

 ==> f-3 (MAIN::Stop (front FALSE) (left TRUE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 10) (fuel 10))
```

Test case 4: -

```
Drone is trying to move forward

Please enter Battery-level in terms of percentage, for 15% enter only 15

5

Please enter fuel-level in terms of percentage, for 15% enter only 15

4

 ==> f-2 (MAIN::Stop (front TRUE) (left TRUE) (right TRUE) (reverse TRUE) (fly TRUE) (battery 5) (fuel 4))


Battery or Fuel Low, Drone stopping!

For a total of 0 facts in module afterStart.


Battery or Fuel Low, Drone stopping!

For a total of 0 facts in module afterStart.
```

I have added (watch facts) to show whenever a fact is added or deleted.