Vars not used from Fablo 1

visionObstacles : $\mathbb{P}$ Pos

safeLocation : $(x, y), x \in \mathbb{N}, y \in \mathbb{N}^4$

return2NormalOperation : bool

prioritisedLocations : $\mathbb{P}$ Pos

Charges : $\mathbb{P}$ Pos

invalidMap : Message

noMoreViablePlans : Message

failureModeInvoked : Message

noPlan : Bool

systemState : (failure, noPlan, messages)

plan2C : seq Pos

plan2O : seq Pos

recheck : Bool

communicationData : (messages, data collected, sensors) — used this

GSObstacles : $\mathbb{P}$ Pos

failed2Reconnect : Bool

Completed : Message

identifiedFailure : Message

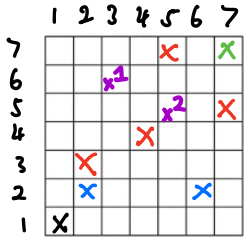
Command : Message

Types

Pos == 0…7 × 0…7

Max_Battery == 5

Message is a String



RoverState

currentPosition	: Pos	
obstacles	: ℙ Pos	
failure	: Bool	
failureReboot	: ℙ Pos	(new variable)
failureHelp	: Pos ⇸ helperIO	(new variable)
goal	: ℙ Pos	
batteryLevel	: ℕ	
chargingComplete	: Bool	
recharge	: Bool	
atGoal	: Bool	
dataUploaded	: Message	(part of communication data)
helperID	: ℕ	

(failureReboot ⊆ obstacles) ∧ (dom(failureHelp) ⊆ obstacles)

∧ (failureReboot ∩ dom(failureHelp) = ∅)

batteryLevel ∈ 0…Max_Battery

chargingComplete <=> batteryLevel = Max_Battery

currentPosition ∉ obstacles

goal ∉ obstacles

helperID ∈ {0,1,2}, 0 means no help (or main rover ID)

dom(failureHelp) ⊆ Pos

ran(failureHelp) ⊆ helperID

RoverState INIT

currentPosition	:	(1,1)
obstacles	:	{(2,3), (4,4), (5,7), (7,5)}
failure	:	False
failureReboot	:	{(2,2), (6,2)}
failureHelp	:	{(2,6) ⟼ 1, (5,5) ⟼ 2}
goal	:	(7,7)
batteryLevel	:	5
chargingComplete	:	True
recharge	:	False
atGoal	:	False
dataUploaded	:	"This is mock data"
helperID	:	0

This means at (5,5) the rover with id of 2 will come to help

*if None is state not defined, implicitly assume original variable is clear.

Scenario 1

move

ΔRoverState
next? : Pos

failure = False
recharge = False
batteryLevel > 0
next? ∉ obstacles

currentPosition' = next?
batteryLevel' = batteryLevel − 1
atGoal' = True ⟷ (currentPosition' = goal)

notifyComplete

Ξ RoverState
dataEmit! : Message

atGoal = True

dataEmit! = dataCollected

selfRecharge

ΔRoverState

failure = False
recharge = False
batteryLevel = 0

recharge' = True

charge

ΔRoverState

failure = False
recharge = True
batteryLevel < 5

batteryLevel' = batteryLevel + 1

FinishCharge

ΔRoverState

failure = False
recharge = True
batteryLevel = 5

recharge' = False
chargingComplete = True

Scenario 2

triggerRebootFailure

ΔRoverState

failure = False
currentPosition ∈ failureReboot

failure' = True
helperID' = 0

rebootRecover

ΔRoverState

failure = True
currentPosition ∈ failureReboot

failure' = False
helperID' = 0

triggerHelpFailure

Δ RoverState

---

failure = False

(currentPosition ∈ dom(failureHelp))

failure' = True

helperID' = 0

requestHelp

Ξ RoverState

---

failure = True

currentPosition ∈ dom(failureHelp)

recieveHelperID

Δ RoverState

---

failure = True

(currentPosition ∈ dom(failureHelp))

helperID' = failureHelp(currentPosition)

authenticateAndRecover

Δ RoverState

arrivingRoverID? : ℕ

---

failure = True

helperID ≠ 0

arrivingRoverID = helperID

failure' = False

helperID' = 0

* The assignment does not state however order is not correct. In terms of actual implementation I can just a way set arrivingRoverID to be where the expected helperID is.