# General BDI & TL:

The following beliefs, desires and intentions/temporal logic is applied to ALL drones.

## Beliefs:

|  |  |  |
| --- | --- | --- |
| Belief | Summary | Predicate |
| BatteryLevel | Current battery level of the drone | BatteryLevel(100)  #Battery level starts at 100, goes to 0 |
| CurrentStatus | Current status of the drone and if it requires maintenance (e.g broken battery) | CurrentStatus(<ListOf> Int) #List of components that are currently broken (empty list) if all parts are working. |
| ChargingPointLocation | The fixed location of the charging station. | ChargingPointLocation((x, y)) #Tuple containing coordinates of Charging Point |
| ForestBounds | The bounds of the forest (its size) | ForestBounds[ForestX][ForestY]  ForestX = 50  ForestY = 50 #2D array to describe forest |

## 

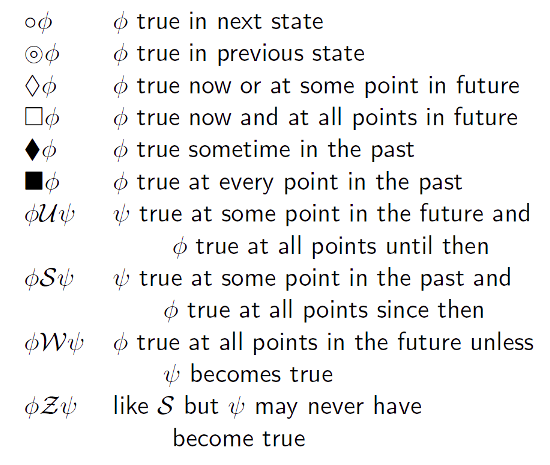
## 

## Desires:

“!φ” denotes “achieve φ”, “?φ” denotes “query φ”

|  |  |  |
| --- | --- | --- |
| Agent | Desire | Rationale |
| General (Ds, Df, Dm) | ! BatteryLevel > BatteryToChargingStation | Ensures the drones can go to charging station by themselves  Numerical comparison supported? If not then need another solution |
|  | ! At(ChargingPointLocation) | Desire for reaching the charging station |
|  | ? BrokenParts | Check for any broken parts  Belief for broken parts |
|  | ! (not BrokenParts) | Desire to remain functioning |
|  | ! (not At(ForestBounds)) | Remain inside the forest  Will be useful to have a belief when inside / outside forest |

## Temporal Logic:



Temporal logic symbols

**Symbols:** ◯ ⌾ ◇ ⯁ ⬛ ⬜ ∀ ∃ ∄ ⌐ 𝓩 Ⲫ 𝓦 Ｓ U ψ →

# 

# Survey drones [sn]

## Beliefs:

|  |  |  |
| --- | --- | --- |
| **Belief** | **Summary** | **Predicate** |
| PatrolRoute | Which section of the forest the drone has been assigned to patrol. | PatrolRoute(LinkedList<tuple>)  #A linked list of tuples containing coordinates that the drone must visit one after the other. |

## Intentions:

|  |  |
| --- | --- |
| **Intentions** | **Sample Actions** |
| **Constant** | |
| maintain\_ground\_clearance | measure\_distance\_from\_ground  adjust\_ground\_clearance |
| proximity\_sensing | scan\_for\_objects\_in\_close\_proximity  scan\_for\_objects\_in\_distant\_proximity |
| obstacle\_avoidance | adjust\_route\_for\_objects\_in\_close\_proximity  adjust\_route\_for\_objects\_in\_distant\_proximity |
| power\_supply\_level\_sensing | calculate\_battery\_level |
| hardware\_malfunction\_sensing | perform\_sensor\_check |
| software\_malfunction\_sensing | perform\_software\_check |
| communications\_listening | radio\_communications\_on  frequency\_scanning |
| fire\_sensing | thermal\_camera\_sensing  main\_camera\_sensing |
|  |  |
|  |  |
| **Events** | |
| initialise | scan\_environment  share\_environment\_information  listen\_for\_environment\_information  build\_environment\_model  decide\_on\_patrol\_area |
| fire\_detection | complete\_additional\_measurement\_to\_confirm\_fire\_detection  notify\_nearby\_drones  vote\_with\_first\_responder\_drones  *if fire:* broadcast\_fire\_alarm |
| patrol | calculate\_patrol\_route  listen\_for\_change\_in\_route  notify\_agents\_in\_close\_proximity\_for\_change\_in\_route  listen\_for\_environment\_infomation\_from\_agents\_in\_close\_proximity  Share\_environment\_beliefs\_with\_agents\_in\_close\_proximity |
| route\_recalculating | adjust\_path\_based\_on\_new\_beliefs  compile\_paths\_to\_form\_new\_route |
| battery\_replacement | broadcast\_location\_and\_battery\_information\_to\_station  listen\_to\_station\_for\_battery\_change\_information  broadcast\_to\_maintenance\_drone\_precise\_location  scan\_for\_maintenance\_drone\_in\_close\_proximity  engage\_battery\_swap\_procedure (to be decided exactly later) |
| maintenance | broadcast\_hardware\_analysis\_to\_station /  broadcast\_software\_analysis\_to\_station  listen\_for\_station\_instructions  calculate\_shortest\_path\_to\_station |
| sensor\_calibration | analyse\_current\_state\_of\_sensors  detect\_sensors\_in\_need\_of\_calibration  collect\_new\_beliefs  calculate\_calibration  test\_calibration |
| return\_to\_station | locate\_station\_location  calculate\_shortest\_path\_to\_station  go\_to\_station |
|  |  |

## Desires:

“!φ” denotes “achieve φ”, “?φ” denotes “query φ”

|  |  |  |
| --- | --- | --- |
| Agent | Desire | Rationale |
| Survey (Ds) | ? Fire | Check for fire  Belief for fire |
|  | ! (not Fire) | Keep the spot without fire, if not, should lead to an intention |
|  | ! follow(PatrolRoute) | Desire to follow their route |

## Temporal Logic:

*Listening for*: survey\_support, support\_eta

*Can send*: alert\_fire, survey\_support, request\_battery, request\_maintenance, fire\_out

⌾ ask(survey\_support) → action(move\_in\_requested\_drone\_area)

⌾ fact(fire\_confirmed) → ◇ give(alert\_fire)

⌾ fact(charging<50) → ◇ give(request\_battery)

⌾ fact(breakdown) → ◇ give(request\_maintenance) AND ◇ give(survey\_support)

(fact(fire\_fighting)Ｓ(fire\_confirmed)) AND fact(fire\_ceased) → give(fire\_out)

# Firefighting Drones

## Beliefs:

|  |  |  |
| --- | --- | --- |
| **Belief** | **Summary** | **Predicate** |
| AvailableParts | What parts are available for the maintenance drone to replace. | AvailableParts(<List>)  #A list containing all available parts for the maintenance drone. |

## Intentions:

## 

|  |  |
| --- | --- |
| **Intentions** | **Sample Actions** |
| **Constant** | |
| maintain\_ground\_clearance | measure\_distance\_from\_ground  adjust\_ground\_clearance |
| proximity\_sensing | scan\_for\_objects\_in\_close\_proximity  scan\_for\_objects\_in\_distant\_proximity |
| obstacle\_avoidance | adjust\_route\_for\_objects\_in\_close\_proximity  adjust\_route\_for\_objects\_in\_distant\_proximity |
| power\_supply\_level\_sensing | calculate\_battery\_level |
| hardware\_malfunction\_sensing | perform\_sensor\_check |
| software\_malfunction\_sensing | perform\_software\_check |
| communications\_listening | radio\_communications\_on  frequency\_scanning |
| water\_level\_sensing | get\_sensor\_reading\_on\_water\_container |
| temperature\_sensing | get\_readings\_on\_all\_sensors  calculate\_temperature\_average |
|  |  |
| **Events** | |
| initialise | check\_battery\_level  check\_water\_level  get\_incident\_location  calculate\_shortest\_path |
| put\_off\_fire | locate\_target  locate\_other\_agents  calculate\_optimal\_proximity\_to\_fire  move\_to\_optimal\_position  put\_out\_fire  check\_water\_level\_for\_refill  check\_for\_hardware\_malfunction  check\_barrery\_level\_for\_recharge |
| fill\_up\_water\_containters | check\_water\_level  calculate\_shortest\_path\_to\_water\_source |
| battery\_replacement | broadcast\_location\_and\_battery\_information\_to\_station  listen\_to\_station\_for\_battery\_change\_information  broadcast\_to\_maintenance\_drone\_precise\_location  scan\_for\_maintenance\_drone\_in\_close\_proximity  engage\_battery\_swap\_procedure (to be decided exactly later) |
| return\_to\_station | locate\_station\_location  calculate\_shortest\_path\_to\_station  go\_to\_station |
|  |  |
|  |  |

## 

|  |  |
| --- | --- |
| **Intentions** | **Sample Actions** |
| **Constant** | |
| maintain\_ground\_clearance | measure\_distance\_from\_ground  adjust\_ground\_clearance |
| proximity\_sensing | scan\_for\_objects\_in\_close\_proximity  scan\_for\_objects\_in\_distant\_proximity |
| obstacle\_avoidance | adjust\_route\_for\_objects\_in\_close\_proximity  adjust\_route\_for\_objects\_in\_distant\_proximity |
| power\_supply\_level\_sensing | calculate\_battery\_level |
| hardware\_malfunction\_sensing | perform\_sensor\_check |
| software\_malfunction\_sensing | perform\_software\_check |
| communications\_listening | radio\_communications\_on  frequency\_scanning |
|  |  |
|  |  |
| **Events** | |
| initialise | get\_information\_of\_agent\_incidence  get\_location\_of\_agent\_incidence  attain\_necessary\_tools\_for\_task  calculate\_shortest\_path\_to\_location  go\_to\_location |
| battery\_replacement | listen\_for\_agent\_location  check\_battery\_charge\_percentage (make sure it’s full)  calculate\_shortest\_path\_to\_location  go\_to\_location  scan\_for\_agent\_in\_close\_proximity  engage\_battery\_swap\_procedure (to be decided exactly later) |
| perform\_other\_maintenance | listen\_for\_agent\_location  check\_hardware\_necessary\_to\_perform\_maintenance  calculate\_shortest\_path\_to\_location  go\_to\_location  scan\_for\_agent\_in\_close\_proximity  engage\_maintenance\_procedure (to be decided exactly later) |

## Desires:

## 

“!φ” denotes “achieve φ”, “?φ” denotes “query φ”

|  |  |  |
| --- | --- | --- |
| Agent | Desire | Rationale |
| Firefighting (Df) | ? FireReported | Check for fire report |
|  | ! ∃(not (FireReport)) | There should be no fire if they do their job |
|  | ! StoredWater(Full) | Try to maintain full water |

## Temporal Logic:

*Legend: agent\_name (listens\_for) [asks\_for]*

Df (fire\_alert, backup\_call, support\_eta) [water\_out\_alert, need\_backup, fire\_out, request\_battery, breakdown\_alert, alert\_humans]:

⌾ fire\_alert → ◇ reach alert coordinates

⌾ backup\_call → ◇ reach backup coordinates

⌾ fact(charging<50) → ◇ give(request\_battery)

⌾ fact(water\_out) → ◇ give(water\_out\_alert) AND ◇ give(need\_backup)

⌾ fact(breakdown) → ◇ give(request\_maintenance) AND ◇ give(need\_backup)

⌾ fact(code\_red)→ ◇ give(alert\_humans)

(fire\_alert U fact(fire\_ceased)) → ◇ give(fire\_out)

# Maintenance Drones

## Beliefs:

|  |  |  |
| --- | --- | --- |
| **Belief** | **Summary** | **Predicate** |
| AvailableParts | What parts are available for the maintenance drone to replace. | AvailableParts(<List>)  #A list containing all available parts for the maintenance drone. |

## Intentions:

## 

|  |  |
| --- | --- |
| **Intentions** | **Sample Actions** |
| **Constant** | |
| maintain\_ground\_clearance | measure\_distance\_from\_ground  adjust\_ground\_clearance |
| proximity\_sensing | scan\_for\_objects\_in\_close\_proximity  scan\_for\_objects\_in\_distant\_proximity |
| obstacle\_avoidance | adjust\_route\_for\_objects\_in\_close\_proximity  adjust\_route\_for\_objects\_in\_distant\_proximity |
| power\_supply\_level\_sensing | calculate\_battery\_level |
| hardware\_malfunction\_sensing | perform\_sensor\_check |
| software\_malfunction\_sensing | perform\_software\_check |
| communications\_listening | radio\_communications\_on  frequency\_scanning |
|  |  |
|  |  |
| **Events** | |
| initialise | get\_information\_of\_agent\_incidence  get\_location\_of\_agent\_incidence  attain\_necessary\_tools\_for\_task  calculate\_shortest\_path\_to\_location  go\_to\_location |
| battery\_replacement | listen\_for\_agent\_location  check\_battery\_charge\_percentage (make sure it’s full)  calculate\_shortest\_path\_to\_location  go\_to\_location  scan\_for\_agent\_in\_close\_proximity  engage\_battery\_swap\_procedure (to be decided exactly later) |
| perform\_other\_maintenance | listen\_for\_agent\_location  check\_hardware\_necessary\_to\_perform\_maintenance  calculate\_shortest\_path\_to\_location  go\_to\_location  scan\_for\_agent\_in\_close\_proximity  engage\_maintenance\_procedure (to be decided exactly later) |

## Desires:

“!φ” denotes “achieve φ”, “?φ” denotes “query φ”

|  |  |  |
| --- | --- | --- |
| Agent | Desire | Rationale |
| Maintenance (Dm) | ? MaintenanceRequest | Check maintenance requests |
|  | ! ∃(not (MaintenanceRequest)) | Should be no request if they do their job |

## Temporal Logic:

⌾ ask(request\_battery) AND fact(available)→action(deliver\_battery) AND fact(⌐available)

⌾ fact(request\_completed) →fact(available)

⌾ ask(request\_battery) AND fact(⌐available) → fact(wait) W fact(available)

⌾ ask(breakdown\_call) AND fact(available)→ action(deliver\_maintenance) AND fact(⌐available)

⌾ ask(breakdown\_call) AND fact(⌐available)→ fact(wait) W fact(available)

⌾ fact(wait) AND fact(available) → action(deliver\_battery) OR action(deliver\_maintenance)