Hard Dose Q3

How does the Fallback Node help in making better decisions?

Different situations need different nodes. We can't say that fallback will always make better decisions. But the advantage of fallback node is that it lets several processes to be tried even if anyone process fails in between. There are many many processes going on in a robot, if one fails the robot should not stop completely. Sequence node stops the work flow if any process fails, but fallback allows multiple processes to occur.

Why is BT better than using long if-else conditions?

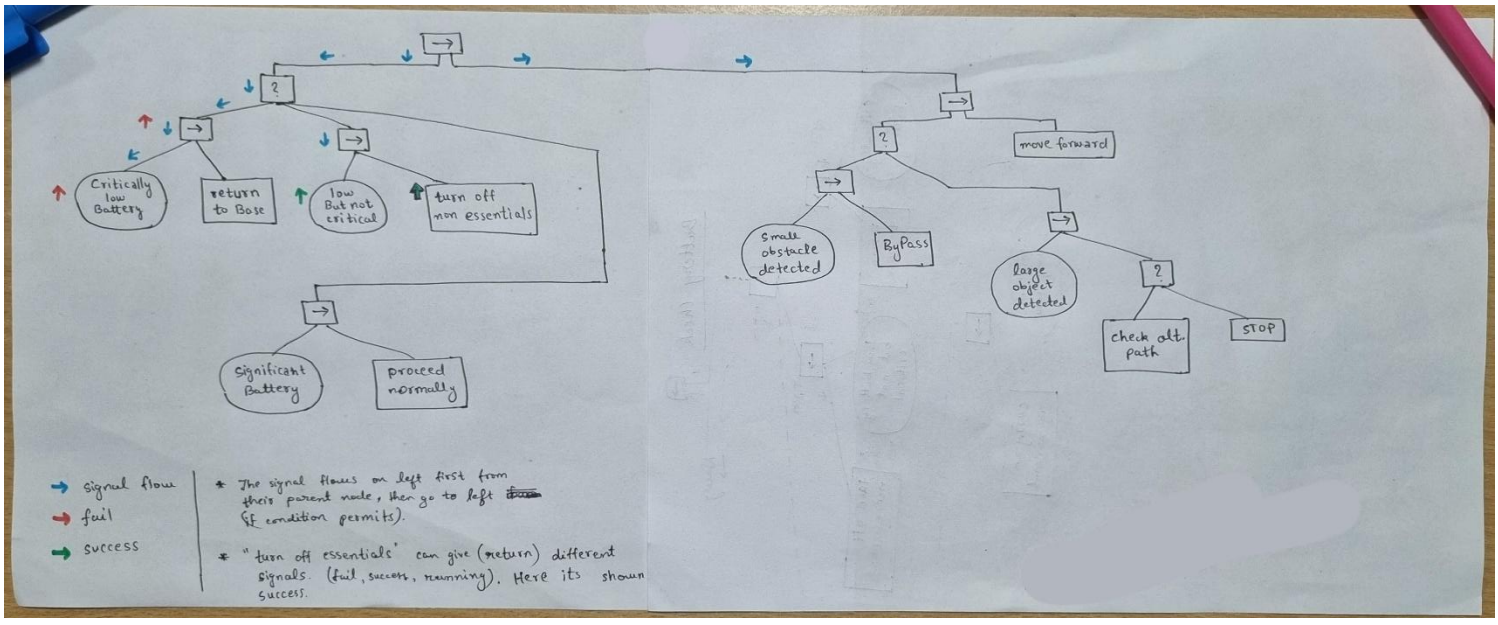
If else conditions get very complex and difficult to understand and manage while managing many conditions whereas, BT is easy to understand and manage. Modifying BT is also easier than if-else. Their structure is reusable(they can be used to handle several different situations).

What happens if the battery is low but not critically low? How does your tree handle this?

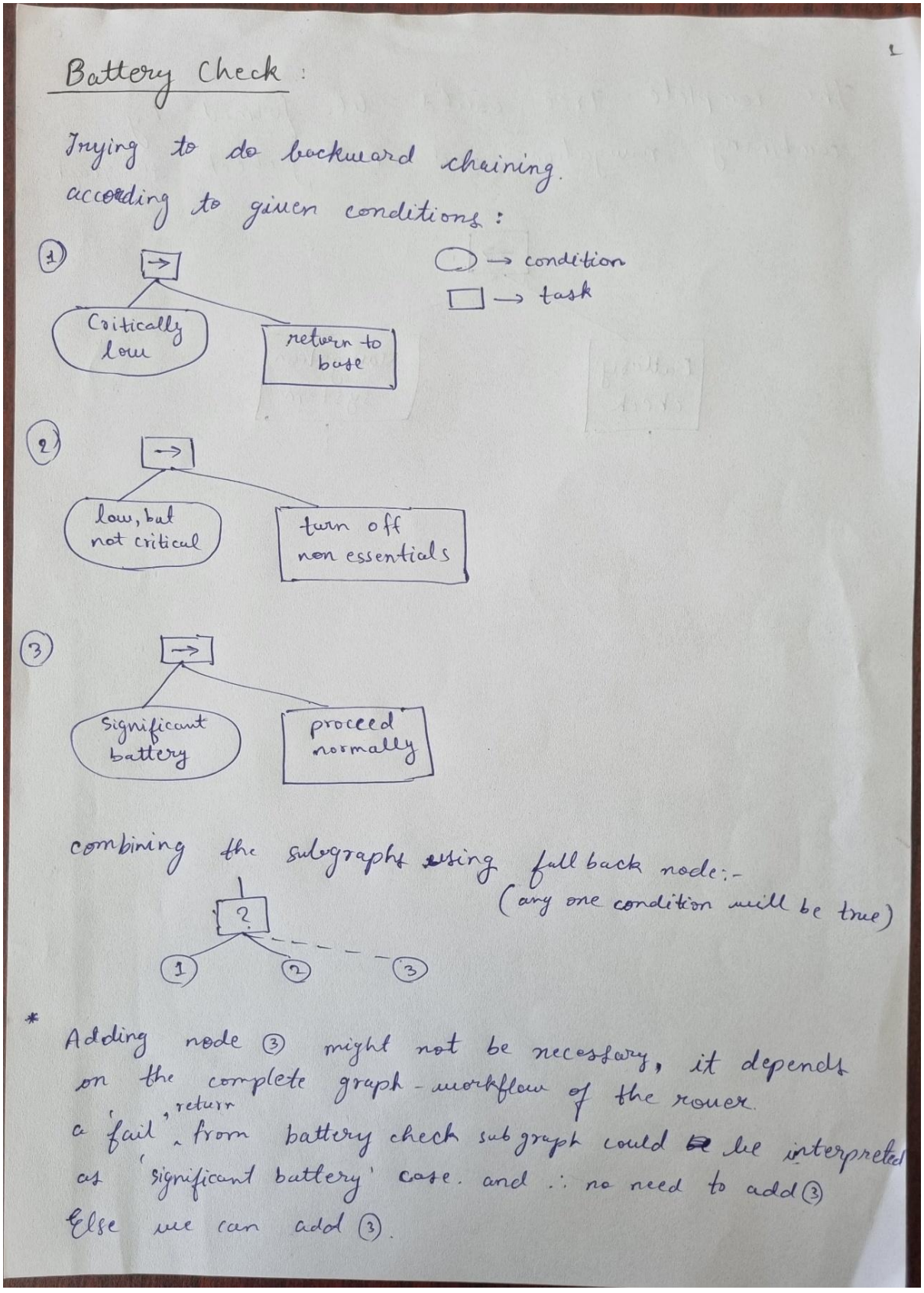
Workflow:

1. The top-most sequence node ticks batter status sub tree on left.
2. The fallback node further ticks the "critical battery condition". This condition is checked and it returns fail, thus tick moves to "low battery but not critical condition".
3. This condition is checked and it returns success, thus the adjoint task is performed, i.e. "Turn off non-essential things".
4. Now the return status of this node depends on the situation, whether the rover is able to turn off non-essentials or it fails to do so. Lets assume it does succeed, then tick moves up to navigation subtree. If It would have failed, the 3rd condition would be processed i.e. "proceed normally". The tick would then move to navigation part.

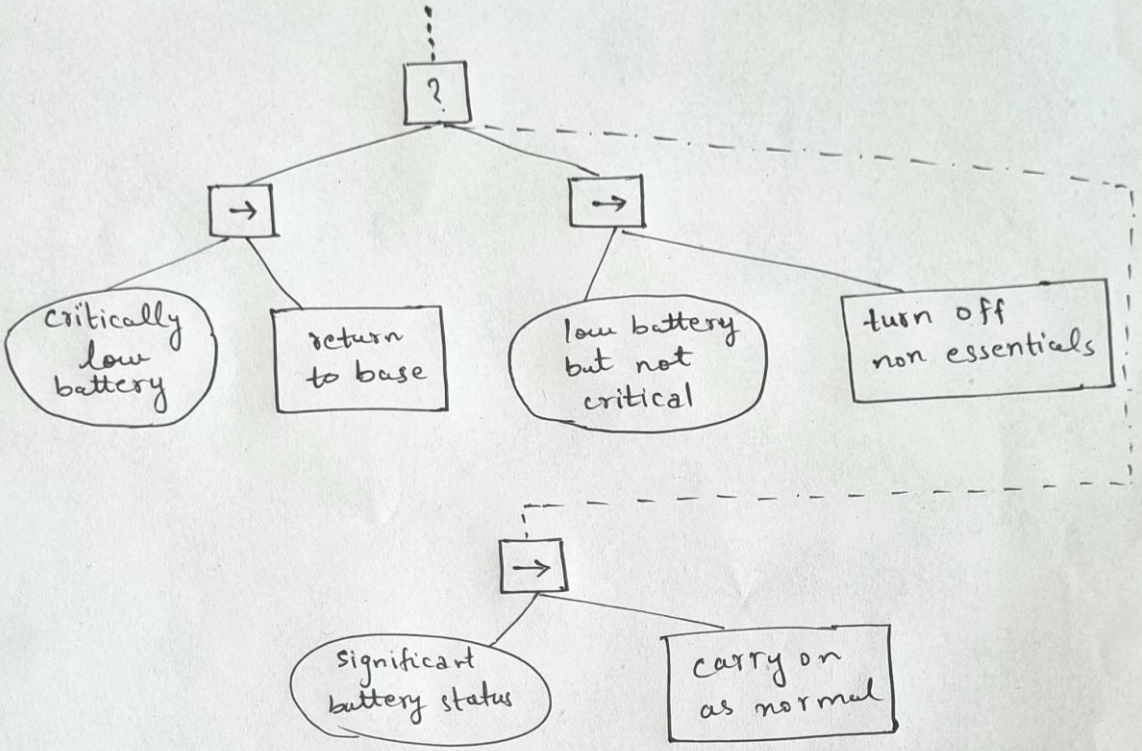
Work flow graphically for "**the battery is low but not critically low**" case:



Solving Approach:



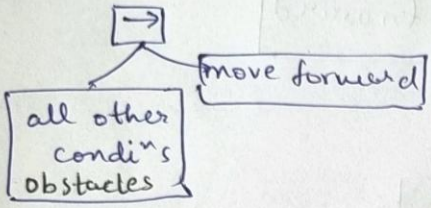
Battery Check :



Navigation :

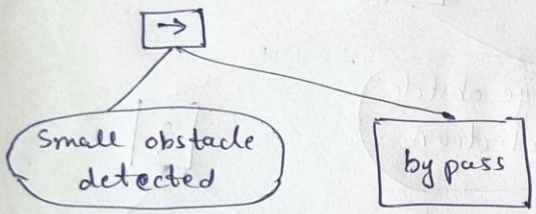
This time I tried to build logic from top most level itself.

rover must move till any obstacle detected, so its like :-

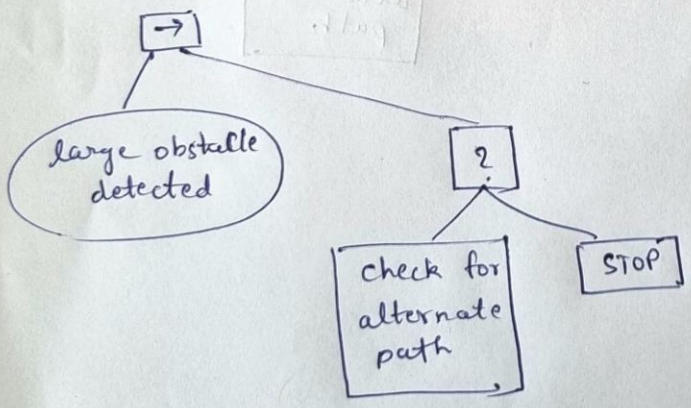


According to given conditions :-

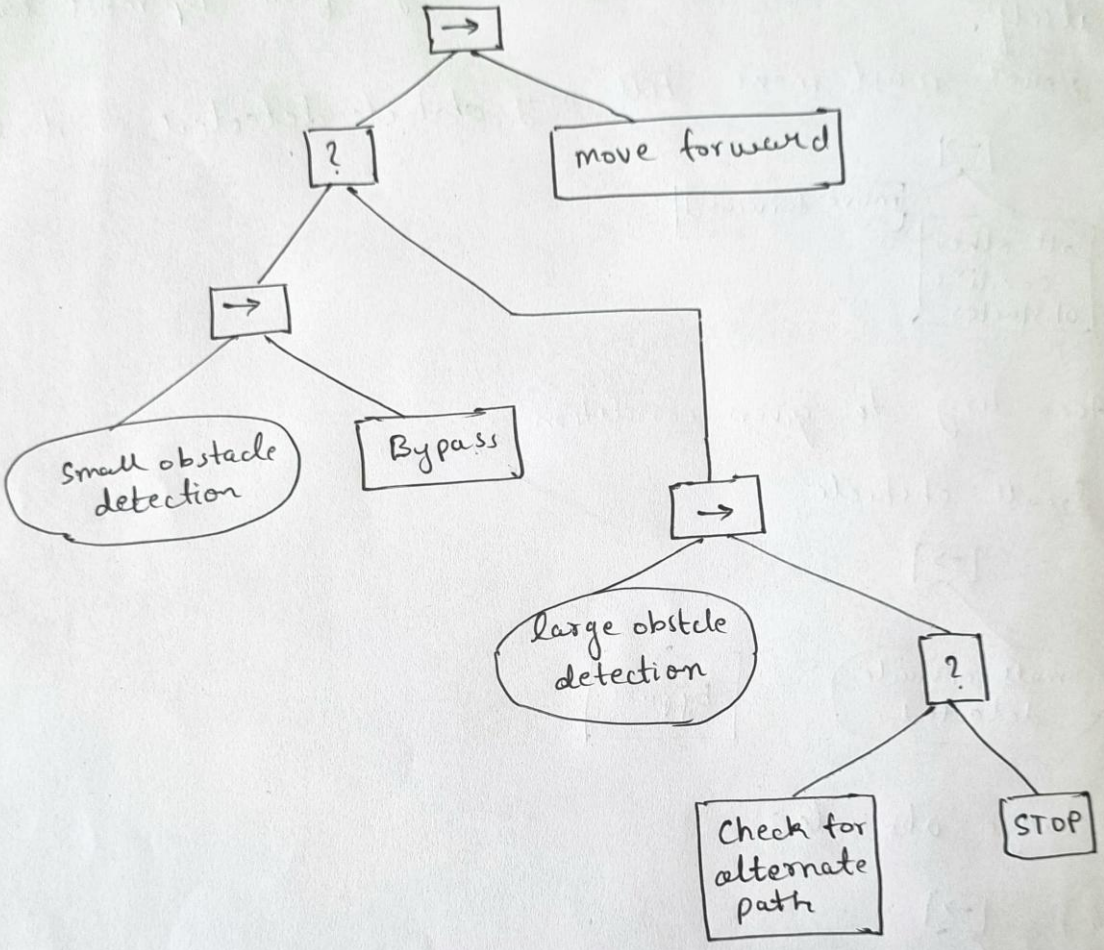
1) Small obstacle :



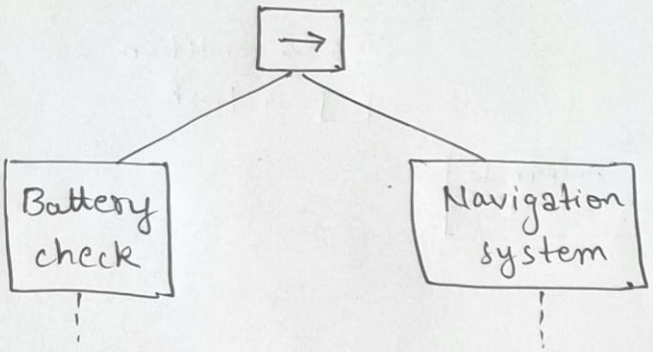
2) large obstacle :



Complete Navigation:



The complete Tree could be formed by combining "navigation" and "Battery check" sub graphs.



sequence node is used here because both systems needs to be checked.

Study reference:

Petter Ögren youtube playlist: (What are BT and how to create them was learnt)

https://www.youtube.com/playlist?list=PLFQdM4LOGDr_vYJuo8YTRcmv3FrwczdKg