LAB – 2

SIMPLE FSM
(ENUM + Switch)

NPC AI at a glance:

Attributes:

- Health
- Level
- Field of View
    - Radius
    - Distance
- Speeds for most of the states mentioned below
- Custom attributes for each state mentioned below

States:

- PATROL
- SUSPICIOUS
- ALERT
- CHASE
- ATTACK
- SEARCH
- ESCAPE
- WAIT

Summary:

The NPC AI has both fight or flee mechanic based on the attributes of **HP** and **Level** of both the player and the NPC.

Initial state is patrol. If the NPC notices player in its field of view, then it will be suspicious and will wait for a short duration before becoming alert if the player is in field of view. This window gives player window of opportunity for stealth.

If the player is not in sight in the alert state, then the NPC chase will pause and inspect the last seen location. If player is not found, then return to patrol.

If the player is still in the alert state, then the NPC will perform actions based on attributes. It will become hostile when it determines that it has a chance to win else it will try to flee and run away until it reaches a threshold distance in opposite direction. Then, it will wait for player to leave so that it can return.

If the player is weaker than the NPC, the NPC will chase player till the player is out of reach or the NPC is within attack range and in field of view. If out of view it will search last known location and after a brief search it will return to patrol if player is not found. If player is found it will enter chase again.

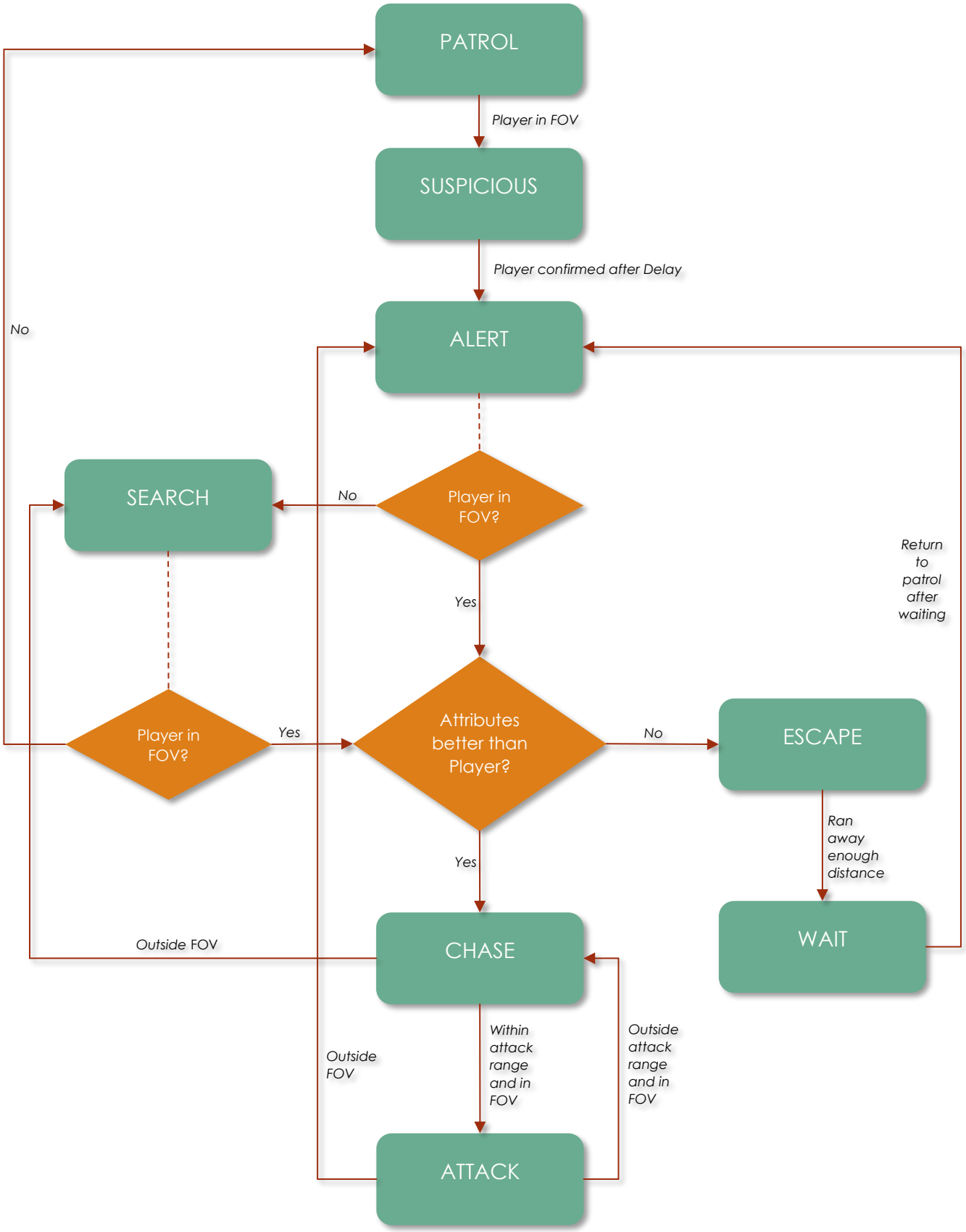

Fig 1.1: Flow chart giving an overview on the AI states.

- Add an extra state and implement it as much as you can.
  - Overhauled the states from the template
  - Added behaviours based on attributes
  - Added ***ALERT, CHASE, WAIT, SEARCH, SUSPICIOUS***

- Changes needed for accommodation of extra state:
  - More attributes needed
  - More interdependent code
  - Made the state transition a little bit more complex because of new behaviour

- Changes predicted to accommodate removal of state(s):
  - Since interdependent code, other states might get affected because of behaviour
  - Removal of some attributes
  - May make the NPC looks dumber in some ways or make it have unnatural behaviour
  - Users may see it as bug

- Nature of changes when states are removed:
  - Interdependent behaviour will get affected resulting in robot like behaviour.
    - For e.g.: In case of ***suspicious,*** it will just start ***alert*** giving player not time to react and at the same time making the NPC feel not like humans with delay in reaction
  - Certain states are intrinsically complex like ***alert***, removal of that state may break the entire FSM of the NPC since it is the heart of the NPC
  - Some states are pure transitional which not have much effect but adds to the quality of experience of the NPC
  - Some states are pure behavioural and may make the NPC react in uncertain ways
  - Programmatically it will be difficult to accomplish since complexity was increased because of the newly added state and are tightly coupled with moderate cohesion

- Sustainability of the method if frequent changes are expected:
  - **No**, the method is not sustainable
  - This method has a lot of interdependencies as well it goes against the programming principles of making a class concise and having singular logic/behaviour resulting in a single big file. Certain languages may cause performance issue if the class is too big.
  - We have only 8 states currently and an average sized games have much more states than we currently have. The code is already complicated and lots of inter references etc.
    - This is just for a single NPC. Games have a lot of NPCs.
  - Let's say we want to make a variant of the NPC but want some states to behave differently. Not only it will have a lot of code duplication it will be very complex to understand and accommodate the changes depending on the states to be changed
  - Let's take an example of a new feature:
    - The designer wants to add invisibility as a new ability to the player
    - Let's see which of the NPC states are going to be affected:
      - Highlighted in YELLOW
      - You can see in the diagram below the number of changes in the states necessary just to add Invisibility ability to the player
      - This method is time consuming, and it makes techniques such as rapid prototyping useless

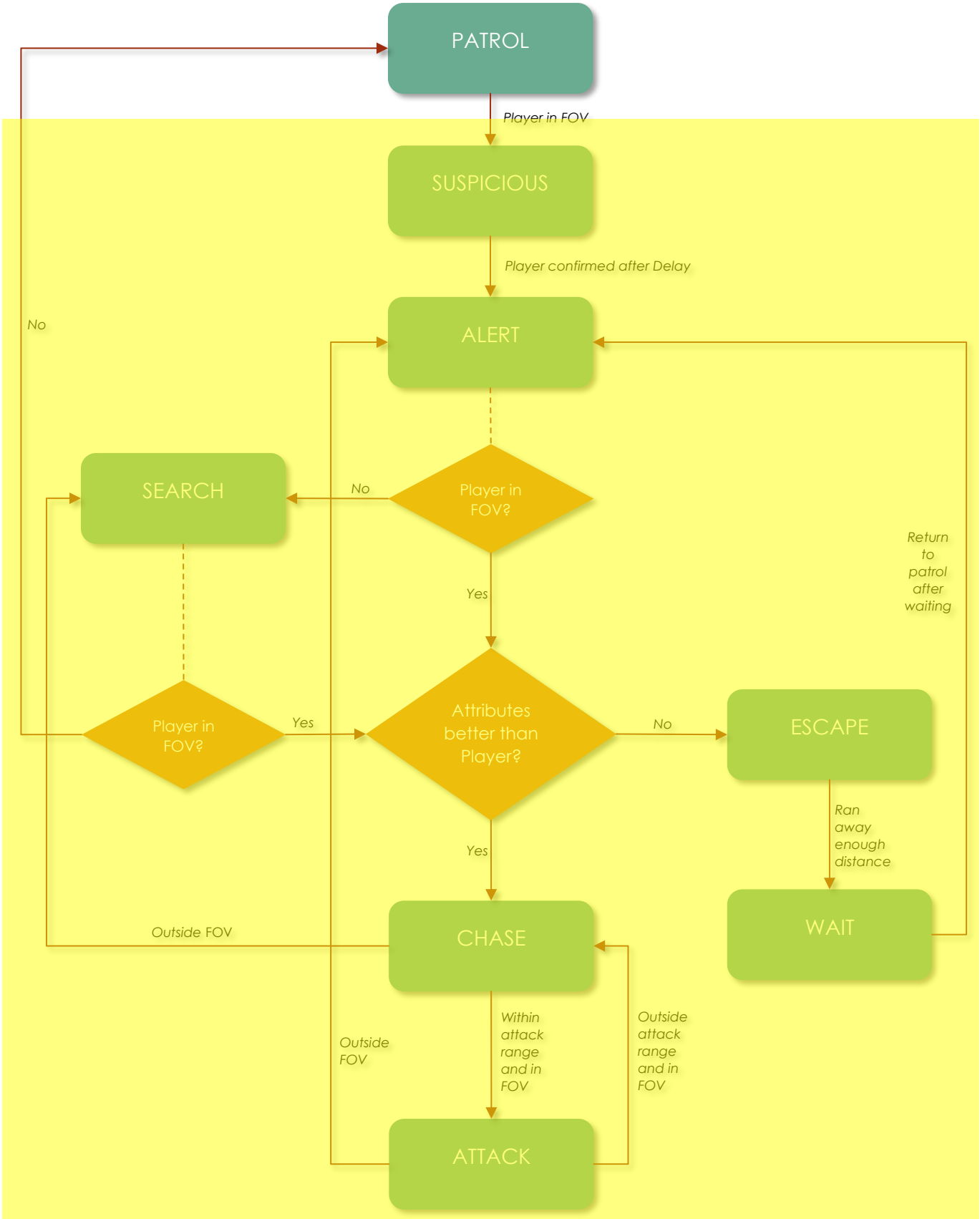

Fig 1.2: Figure showing that which states are affected just to add Invisibility to the player

- Currently, the NPC is self-contained. If in future, the designer wants to add a feature where they want to the NPC to alert other NPCs when it sees a player. This is going to very complex when the NPC will try to interact with others. It will also cause a lot of duplication of code if others are just a variation of the NPC.