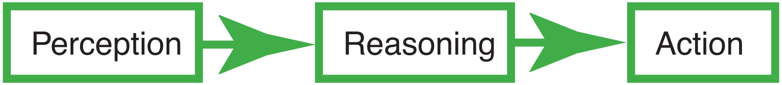
By a hierarchic system, or hierarchy, I mean a system that is composed of interrelated subsystems, each of the latter being in turn hierarchic in structure until we reach some lowest level of elementary subsystem. In most systems of nature it is somewhat arbitrary as to where we leave off the partitioning and what subsystems we take as elementary. Physics makes much use of the concept of "elementary particle," although the particles have a disconcerting tendency not to remain elementary very long . . .

Empirically a large proportion of the complex systems we observe in nature exhibit hierarchic structure. On theoretical grounds we would expect complex systems to be hierarchies in a world in which complexity had to evolve from simplicity.

– *Herbert A. Simon, 1996*

Agent Architectures

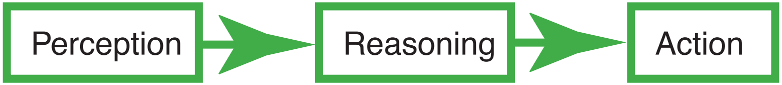
You don’t need to implement an intelligent agent as:



as three independent modules, each feeding into the the next.

Agent Architectures

You don't need to implement an intelligent agent as:

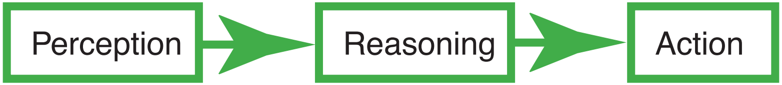


as three independent modules, each feeding into the the next.

- It's too slow.

Agent Architectures

You don't need to implement an intelligent agent as:

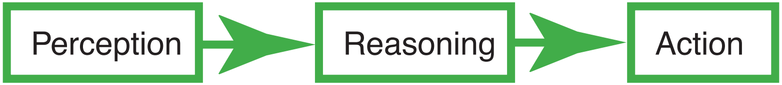


as three independent modules, each feeding into the the next.

- It's too slow.
- High-level strategic reasoning takes more time than the reaction time needed (e.g. to avoid obstacles).

Agent Architectures

You don't need to implement an intelligent agent as:



as three independent modules, each feeding into the the next.

- It's too slow.
- High-level strategic reasoning takes more time than the reaction time needed (e.g. to avoid obstacles).
- The output of the perception depends on what you will do with it.

Hierarchical Control

- A better architecture is a hierarchy of controllers.
- Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.

Hierarchical Control

- A better architecture is a hierarchy of controllers.
- Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.
- The lower-level controllers can

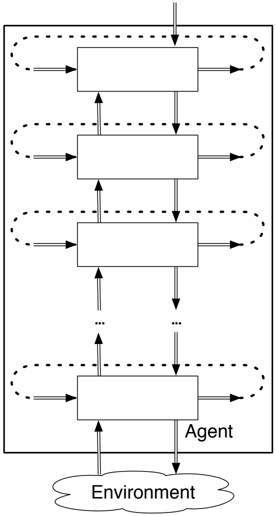
Hierarchical Control

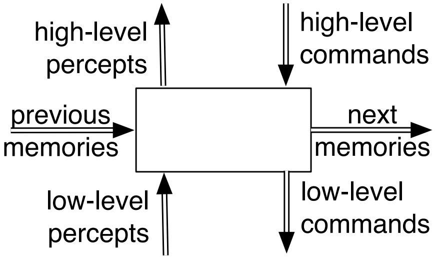
- A better architecture is a hierarchy of controllers.
- Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.
- The lower-level controllers can
  - run much faster, and react to the world more quickly

Hierarchical Control

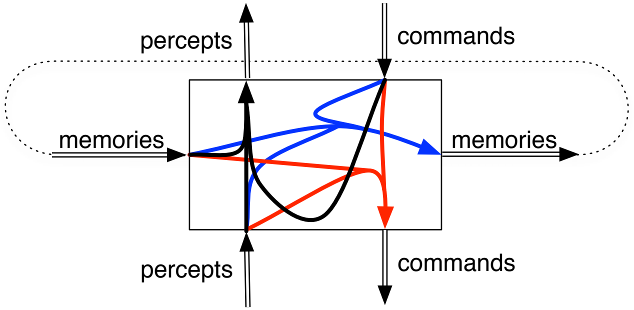
- A better architecture is a hierarchy of controllers.
- Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.
- The lower-level controllers can
  - run much faster, and react to the world more quickly
  - deliver a simpler view of the world to the higher-level controllers.

Hierarchical Robotic System Architecture





Functions implemented in a layer



- memory function $remember(memory, percept, command)$
- command function
  $do(memory, percept, command)$
- percept function $higher\_percept(memory, percept, command)$

Example: delivery robot

- The robot has three actions: go straight, go right, go left. (Its velocity doesn't change).

Example: delivery robot

- The robot has three actions: go straight, go right, go left. (Its velocity doesn't change).
- It can be given a plan consisting of sequence of named locations for the robot to go to in turn.

Example: delivery robot

- The robot has three actions: go straight, go right, go left. (Its velocity doesn't change).
- It can be given a plan consisting of sequence of named locations for the robot to go to in turn.
- The robot must avoid obstacles.

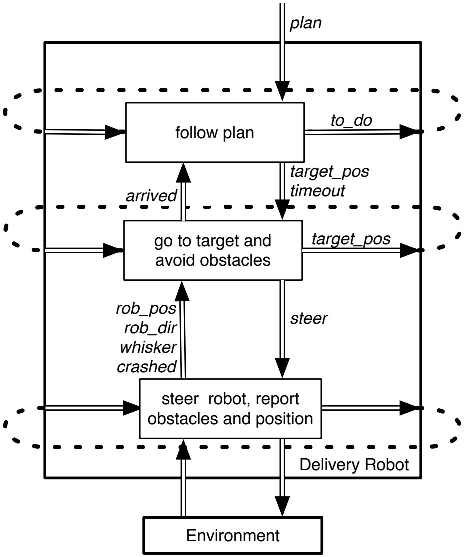
Example: delivery robot

- The robot has three actions: go straight, go right, go left. (Its velocity doesn't change).
- It can be given a plan consisting of sequence of named locations for the robot to go to in turn.
- The robot must avoid obstacles.
- It has a single whisker sensor pointing forward and to the right. The robot can detect if the whisker hits an object. The robot knows where it is.

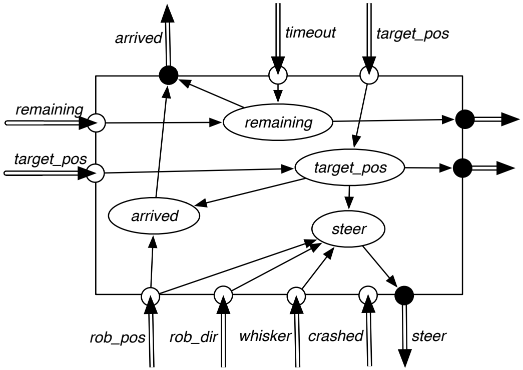
Example: delivery robot

- The robot has three actions: go straight, go right, go left. (Its velocity doesn't change).
- It can be given a plan consisting of sequence of named locations for the robot to go to in turn.
- The robot must avoid obstacles.
- It has a single whisker sensor pointing forward and to the right. The robot can detect if the whisker hits an object. The robot knows where it is.
- The obstacles and locations can be moved dynamically. Obstacles and new locations can be created dynamically.

A Decomposition of the Delivery Robot



Middle Layer



Middle Layer of the Delivery Robot

```
given $timeout$ and $target\_pos$:
    $remaining := timeout$
    while not $arrived()$ and $remaining \neq 0$
        if $whisker\_sensor = on$
            then $steer := left$
        else if $straight\_ahead(rob\_pos, robot\_dir, target\_pos)$
            then $steer := straight$
        else if $left\_of(rob\_pos, robot\_dir, target\_pos)$
            then $steer := left$
        else $steer := right$
        $do(steer)$
        $remaining := remaining - 1$
    tell upper layer $arrived()$

    $arrived = distance(previous\_goal\_pos, robot\_pos)$
```

Top Layer of the Delivery Robot

- The top layer is given a plan which is a sequence of named locations.

Top Layer of the Delivery Robot

- The top layer is given a plan which is a sequence of named locations.
- The top layer tells the middle layer the goal position of the current location.

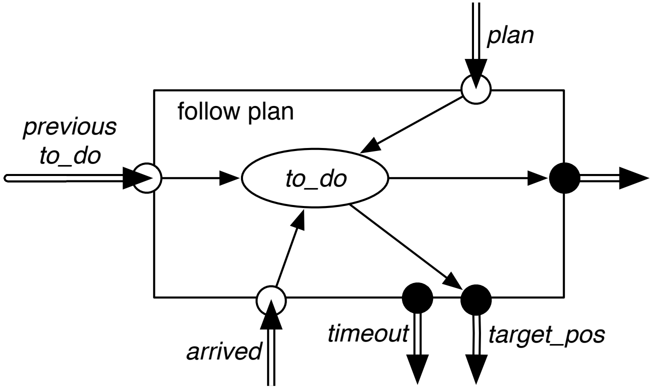
Top Layer of the Delivery Robot

- The top layer is given a plan which is a sequence of named locations.
- The top layer tells the middle layer the goal position of the current location.
- It has to remember the current goal position and the locations still to visit.

Top Layer of the Delivery Robot

- The top layer is given a plan which is a sequence of named locations.
- The top layer tells the middle layer the goal position of the current location.
- It has to remember the current goal position and the locations still to visit.
- When the middle layer reports the robot has arrived, the top layer takes the next location from the list of positions to visit, and there is a new goal position.

Top Layer

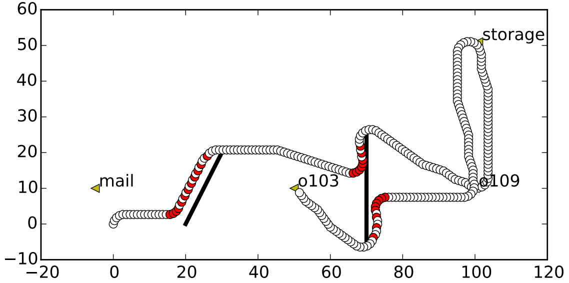


plan
follow plan
previous to_do
to_do
arrived
timeout
target_pos

Code for the top layer

```
given $plan$:
    $to\_do := plan$
    $timeout := 200$
    while not $empty(to\_do)$
        $target\_pos := coordinates(first(to\_do))$
        $do(timeout, target\_pos)$
        $to\_do := rest(to\_do)$
```

Simulation of the Robot



Robot starts at $(0, 0)$ facing up.

$$to\_do = [goto(o109), goto(storage), goto(o109), goto(o103)]$$

Red = whisker sensor on

(Run commands at the bottom of `agentTop.py` in AIPython.org)

What should be in an agent's belief state?

- An agent decides what to do based on its belief state and what it observes.

What should be in an agent's belief state?

- An agent decides what to do based on its belief state and what it observes.
- A purely reactive agent doesn't have a belief state.

What should be in an agent's belief state?

- An agent decides what to do based on its belief state and what it observes.
- A purely reactive agent doesn't have a belief state.
  A dead reckoning agent doesn't perceive the world.
  — neither work very well in complicated domains.

What should be in an agent's belief state?

- An agent decides what to do based on its belief state and what it observes.
- A purely reactive agent doesn't have a belief state.
  A dead reckoning agent doesn't perceive the world.
  — neither work very well in complicated domains.
- It is often useful for the agent's belief state to be a model of the world (itself and the environment).