Chapter 2: Architectures

Agent Architectures and Hierarchical Control

Overview:

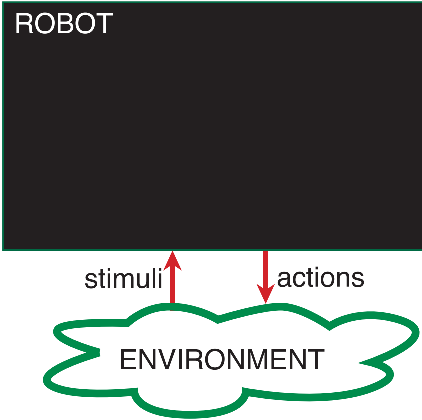
- Agents and Robots
- Agent systems and architectures
- Agent controllers
- Hierarchical controllers

Agents and Robots

A situated agent perceives, reasons, and acts in time in an environment.

- An **agent** is something that acts in the world.
- A **purposive agent** prefers some states of the world to other states, and acts to try to achieve worlds they prefer.
- Agents interact with the environment with a **body**.
- An **embodied** agent has a physical body.
- A **robot** is an artificial purposive embodied agent.

Agent Systems

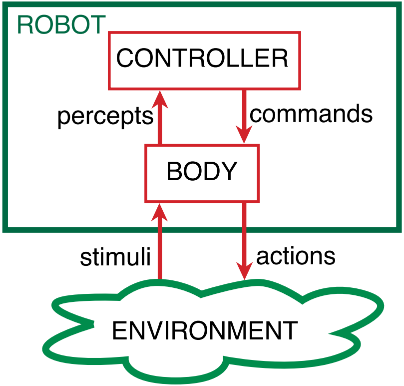


A agent system is made up of a agent and an environment.

- An agent receives stimuli from the environment
- An agent carries out actions in the environment.

Agent System Architecture

An agent is made up of a body and a controller.



- An agent interacts with the environment through its body.
- The body is made up of:
  - sensors that interpret stimuli
  - actuators that carry out actions
- The controller receives percepts from the body.
- The controller sends commands to the body.
- The body can also have reactions that are not controlled.

Implementing a controller

- A controller is the brain of the agent.
- Agents are situated in time, they receive sensory data in time, and do actions in time.
- Controllers have (limited) memory and (limited) computational capabilities.
- The controller specifies the command at every time.
- The command at any time can depend on the current and previous percepts.

The Agent Functions

- Let T be the set of time points.
- A **percept trace** is a sequence of all past, present, and future percepts received by the controller.
- A **command trace** is a sequence of all past, present, and future commands output by the controller.
- A **transduction** is a function from percept traces into command traces.
- A transduction is **causal** if the command trace up to time t depends only on percepts up to t.
- A **controller** is an implementation of a causal transduction.
- A causal transduction specifies a function from an agent's history at time t into its action at time t.

Belief States

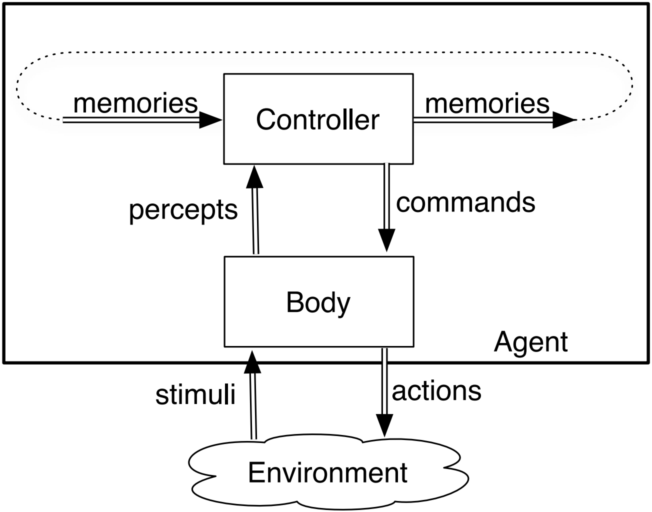
- An agent doesn't have access to its entire history. It only has access to what it has remembered.
- The memory or belief state of an agent at time t encodes all of the agent's history that it has access to.
- The memory of an agent encapsulates the information about its past that it can use for current and future actions, in particular
  - What is true in the world?
  - Beliefs about the dynamics of the environment.
  - Beliefs about what the agent itself will do in the future.
  - Expectations about the behavior of other agents.

Belief States

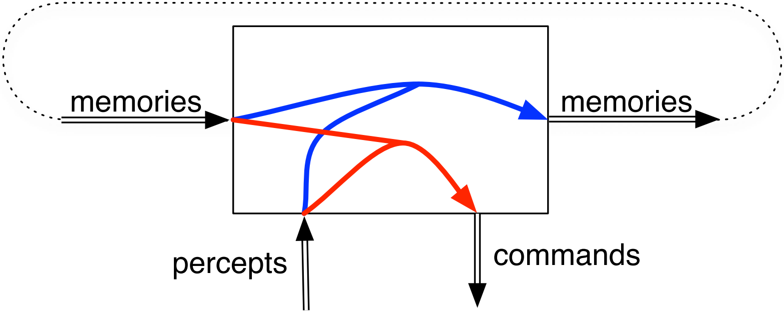
- At every point in time a controller has to decide on:
  - What should it do?
  - What should it keep in memory?

  as a function of its percepts and its memory.

Controller


memories
Controller
memories
percepts
commands
Body
Agent
stimuli
actions
Environment

Functions implemented in a controller



For discrete time, a controller implements:

- memory function *remember*(*memory*, *percept*), returns the next memory.
- command function *do*(*memory*, *percept*) returns the command for the agent.

Agent Architectures

Sequential architectures are not adequate for intelligent agents:

- They are too slow.
- High-level strategic reasoning takes more time than available to avoid obstacles.
- The output of the perception depends on what you will do with it.

Agent Architectures

In general, finding the most plausible model of the world and selecting the best action given this model cannot be separated from each other.

- The available perceptual information might depend on which model of the world is adoped.
- The choice of a model is influenced by the expected utility of the possible actions given the goals of the agent.
- Actions promising to achieve the goal(s) are preferred.

Incrementality

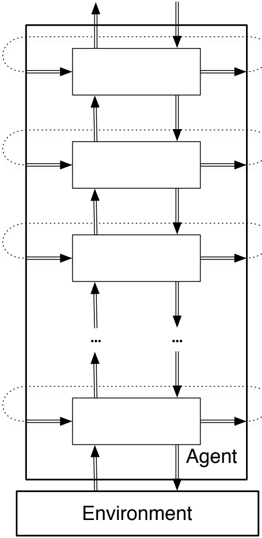
Agents have to be responsive.

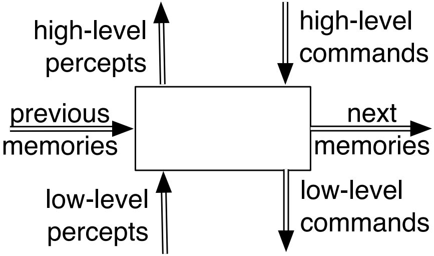
- They need to react in a timely manner
- How long should the agent wait ...
  - ... until enough perceptual evidence has been collected?
  - ... until a decision is taken and a reaction is triggered?
- Intelligent agents need to work incrementally.
- Incremental processing with short feedback cycles allows the agent to quickly adapt to the state-of-affairs in the environment.

Hierarchical Control

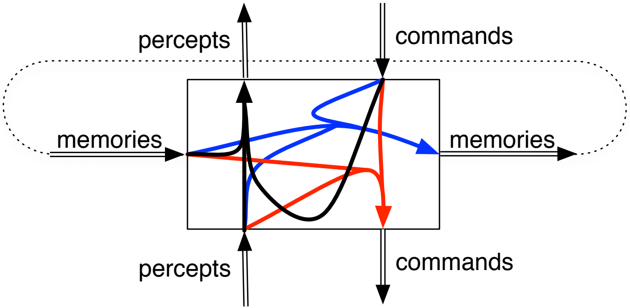
- A better architecture is a hierarchy of controllers.
- Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.
- The lower-level controllers can
  - run much faster, and react to the world more quickly
  - deliver a simpler view of the world to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

Hierarchical Robotic System Architecture





Functions implemented in a layer

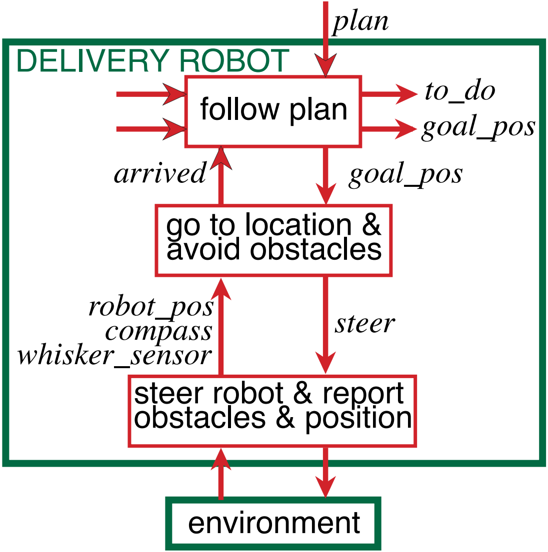


- memory function
  remember(*memory*, *percept*, *command*)
- command function
  do(*memory*, *percept*, *command*)
- percept function
  higher_percept(*memory*, *percept*, *command*)

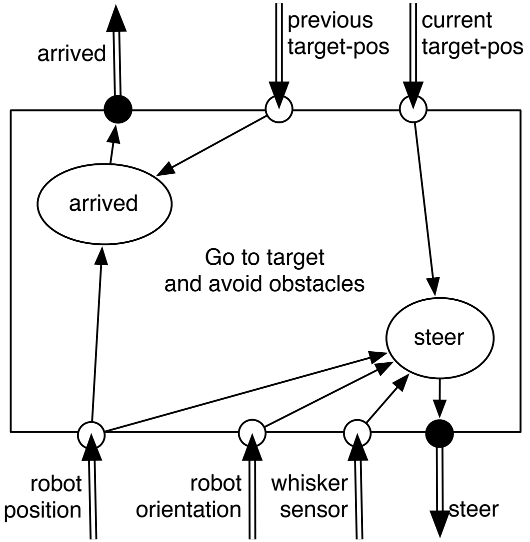
Example: delivery robot

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

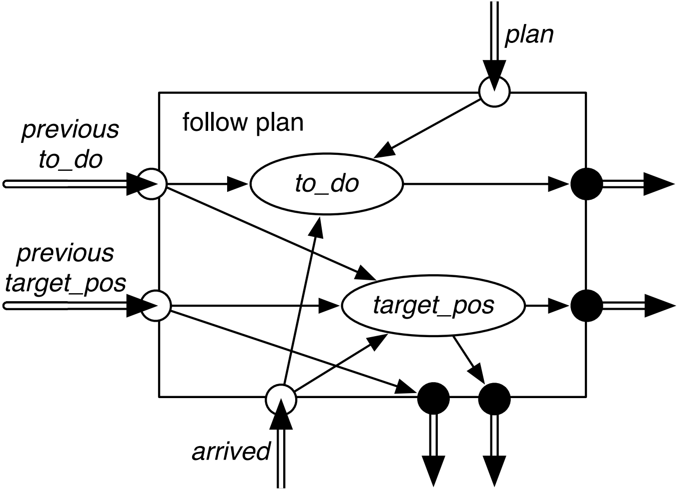
if $whisker\_sensor = on$
    then $steer = left$
else if $straight\_ahead(robot\_pos, robot\_dir, current\_goal\_pos)$
    then $steer = straight$
else if $left\_of(robot\_position, robot\_dir, current\_goal\_pos)$
    then $steer = left$
else $steer = right$

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

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
previous
target_pos
target_pos
arrived

Code for the top layer

The top layer has two belief state variables:

- *to_do* is the list of all pending locations
- *goal_pos* is the current goal position

if *arrived*

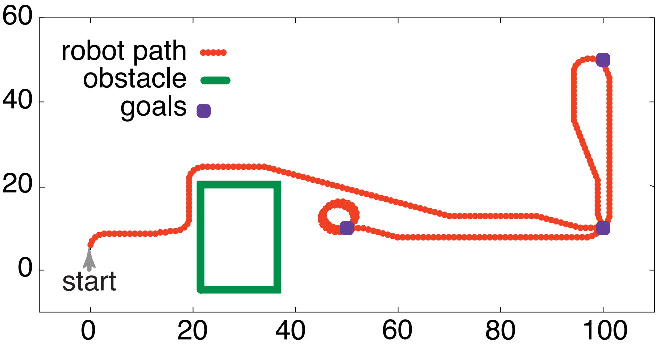
  then $goal\_pos = coordinates(head(to\_do'))$.

if *arrived*

  then $to\_do = tail(to\_do')$.

Here to_do' is the previous value for the *to_do* feature.

Simulation of the Robot



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

$arrived = true$

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).