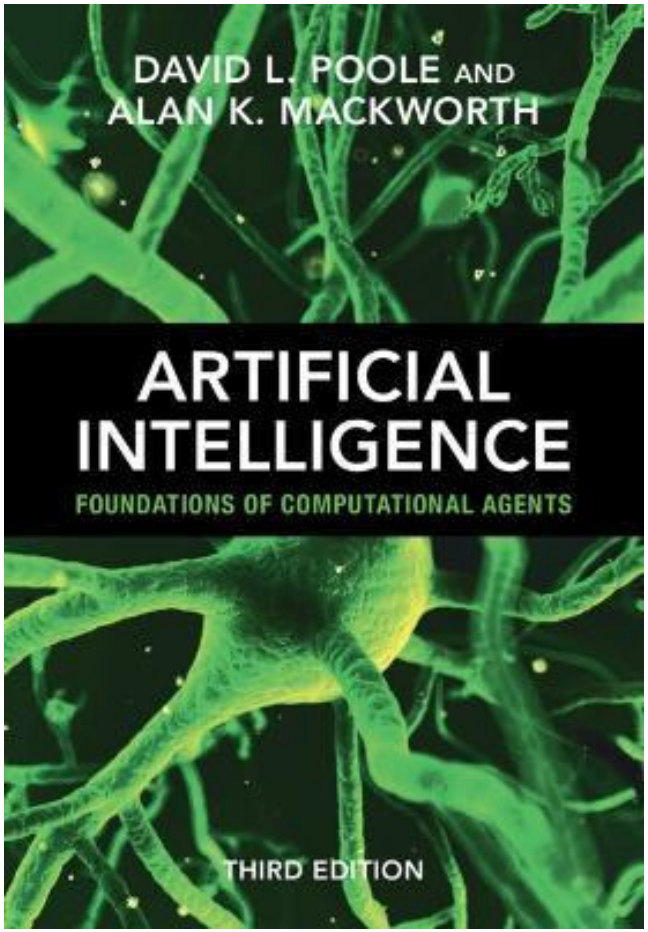
Agent Architectures & Hierarchical Control

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Gloria Beraldo

Materials

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

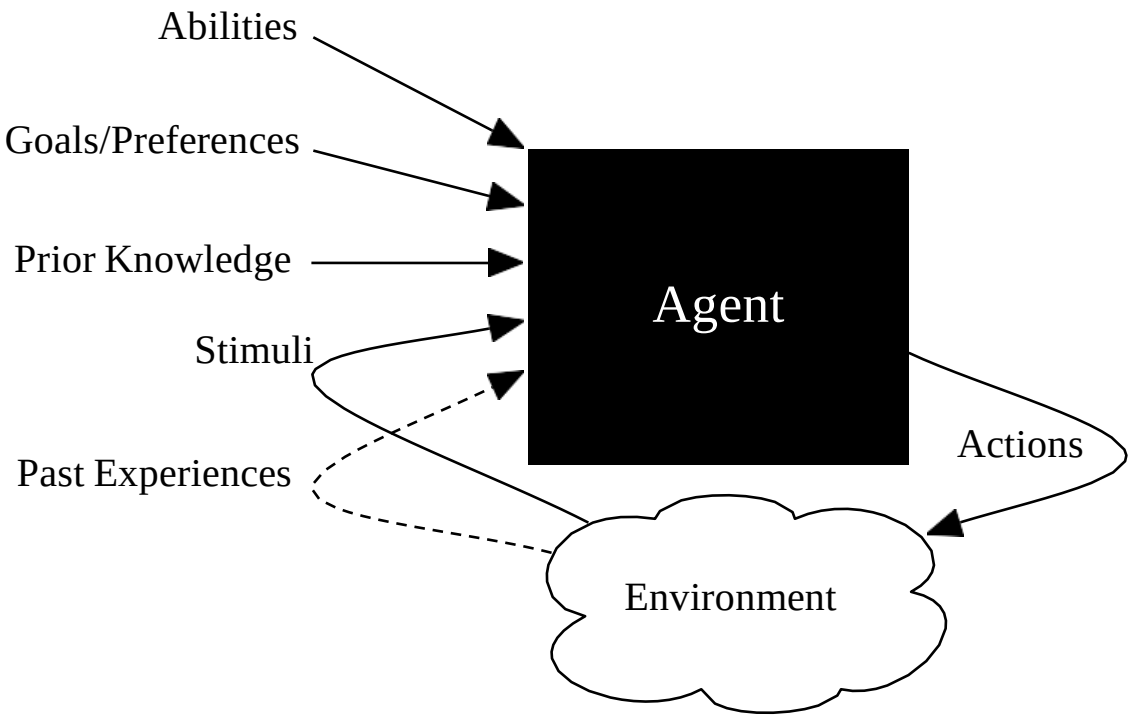


Book:
https://artint.info/3e/html/ArtInt3e.html

Chapter 2

- Agent systems and architectures
- Agent controllers
- Hierarchical controllers

Agents in the environments



- Abilities: the set of primitive actions it is capable of carrying out.
- Goals/Preferences: what it wants, its desires, its values,...
- Prior Knowledge: what it comes into being knowing, what it doesn't get from experience,...

- History of stimuli
  1. (current) stimuli: what it receives from environment now (observations, percepts)
  2. past experiences: what it has received in the past

Examples

ROBOT

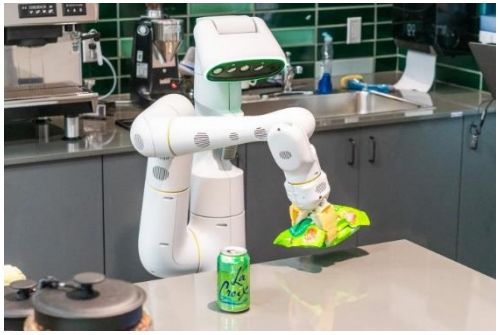
abilities: movement, grippers, speech, facial expressions,. . .

goals: deliver food, rescue people, score goals, explore,. . .

prior knowledge: what is important feature, categories of objects, what a sensor tell us,. . .

stimuli: vision, sonar, sound, speech recognition, gesture recognition,. . .

past experiences: effect of steering, slipperiness, how people move,. . .



THERMOSTAT FOR HEATER

abilities: turn heater on or off

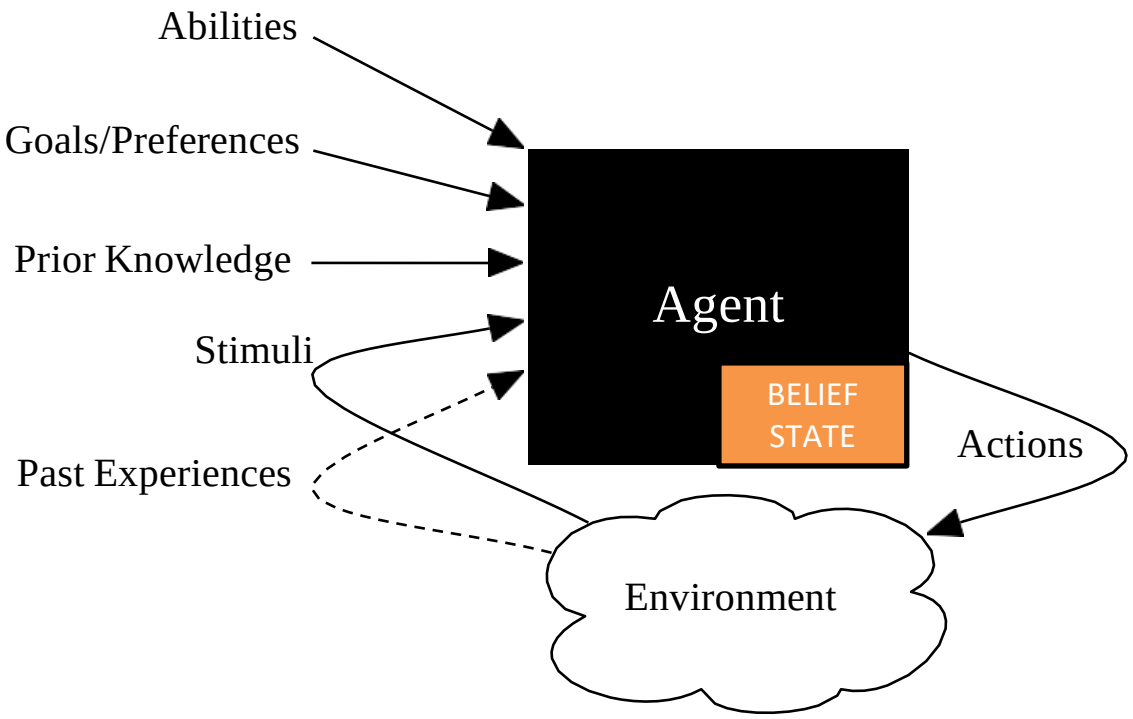
goals: conformable temperature, save fuel, save money

prior knowledge: 24 hour cycle, weekends

stimuli: temperature, set temperature, who is home, outside temperature

past experiences: when people come and go, who likes what temperature

Agents in the environments



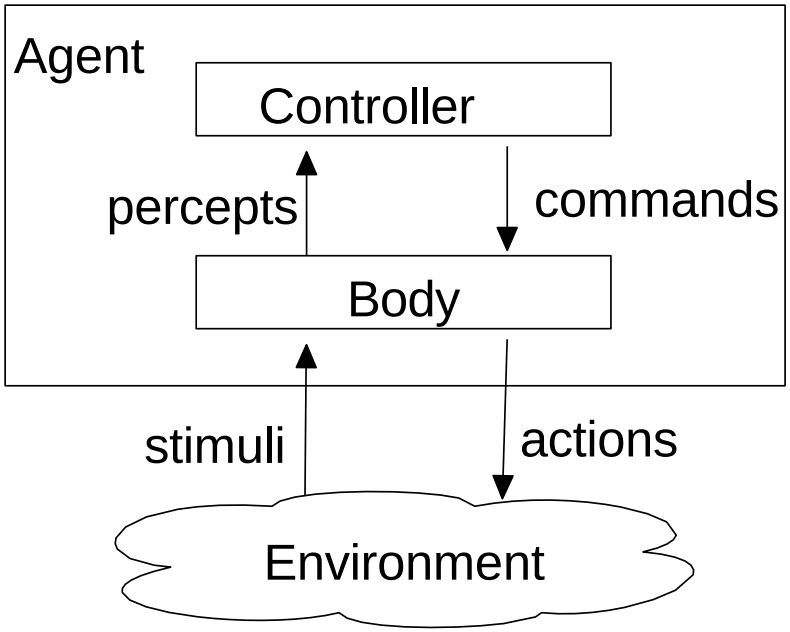
- Abilities: the set of primitive actions it is capable of carrying out.
- Goals/Preferences: what it wants, its desires, its values,...
- Prior Knowledge: what it comes into being knowing, what it doesn't get from experience,...

- History of stimuli
  1. (current) stimuli: what it receives from environment now (observations, percepts)
  2. past experiences: what it has received in the past

BELIEF STATE that encodes beliefs about its environment and it is used to choose actions

The Agent & its components

An agent interacts with the environment through its body.



An agent receives stimuli from the environment

An agent carries out actions in the environment.

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

The body is made up of:

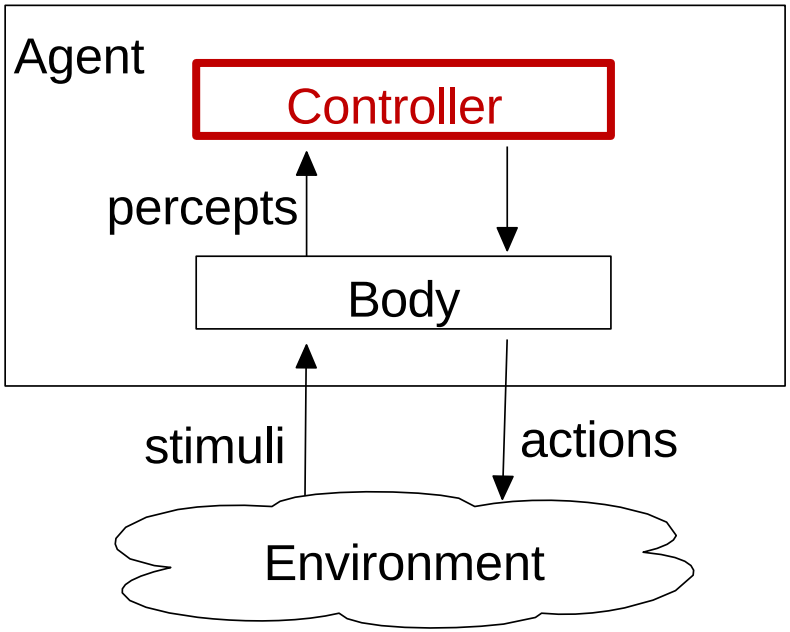
- sensors that interpret stimuli
- actuators that convert commands into actions

The controller receives percepts from the body.

The controller sends commands to the body.

The body can also have reactions that are not controlled (e.g., after emergency botton, reflexes) .

Implementing a controller



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

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*.

An agent's history at time *t* is sequence of past and present percepts and past commands.

A controller is an implementation of a causal transduction that specifies a function from an agent's history at time *t* into its action at time *t*.

Belief states

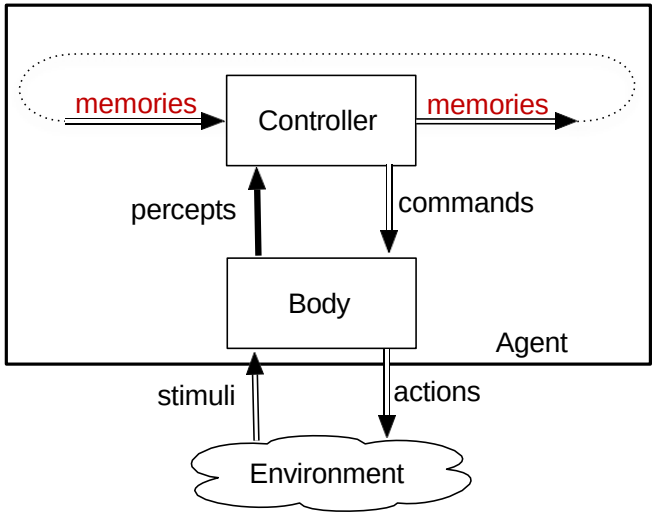
An agent doesn't have access to its entire history. It only has access to what it has remembered.

The memory or belief state of an agent at time *t* encodes all of the agent's history that it has access to.

The belief state of an agent encapsulates the information about its past that it can use for current and future actions.

At every time a controller has to decide on:
What should it do?
What should it remember?
(How should it update its memory?)
as a function of its percepts and its memory.

Functions in a controller



For discrete time, a controller implements:

- belief state function
  remember(*belief state*, *percept*)
  returns the next belief state.

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

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.

Complex agents are built modular in terms of interacting hierarchical layers

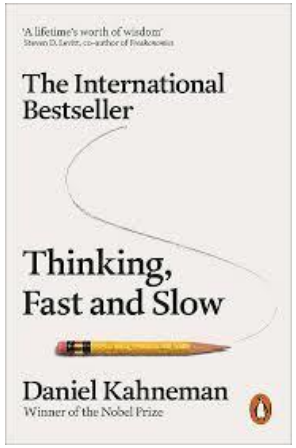
Hierarchical Control

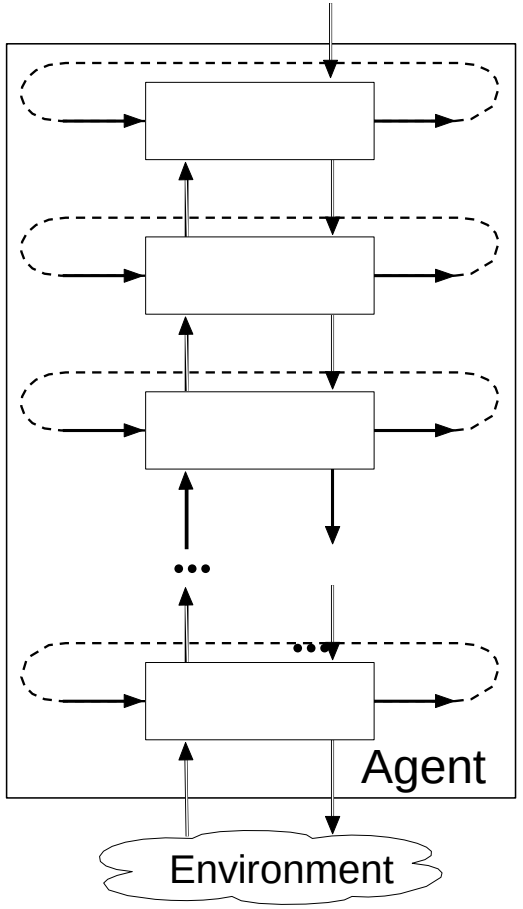
A better architecture is a hierarchy of controllers.

Each controller sees the controllers below it as a virtual body from which it gets percepts and sends commands.

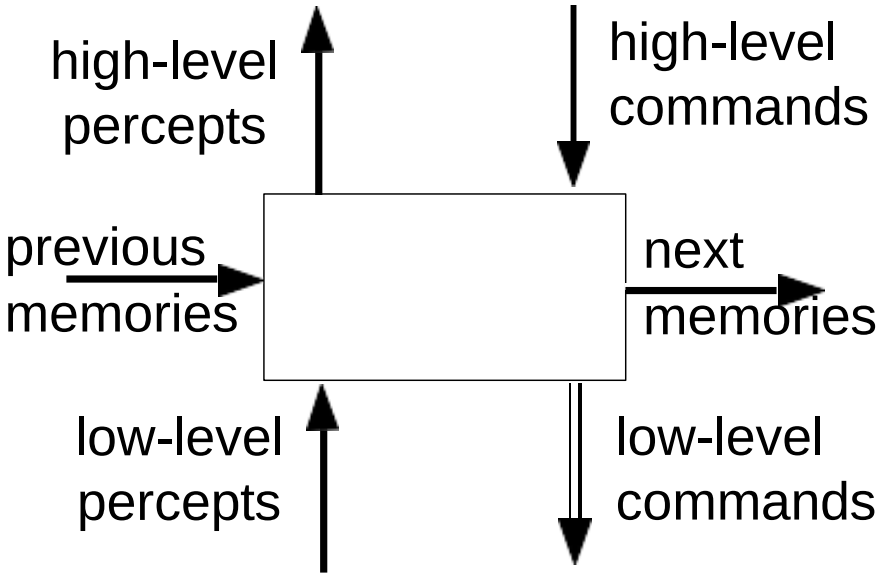
The lower-level controllers can

- run much faster, and react to the world more quickly
- deliver a simpler view of the world to the higher-level controllers.





Functions per layer



memory function

remember (*memory*, *percept*, *command*)

command function

do(*memory*, *percept*, *command*)

percept function

higher percept(*memory*, *percept*, *command*)

Knowledge & Symbol Levels

Two levels of abstraction:

- The knowledge level is in terms of what an agent knows and what its goals are.
- The symbol level is a level of description of an agent in terms of what reasoning it is doing.

The knowledge level is about the external world to the agent.

The symbol level is about what symbols an agent uses to implement the knowledge level.

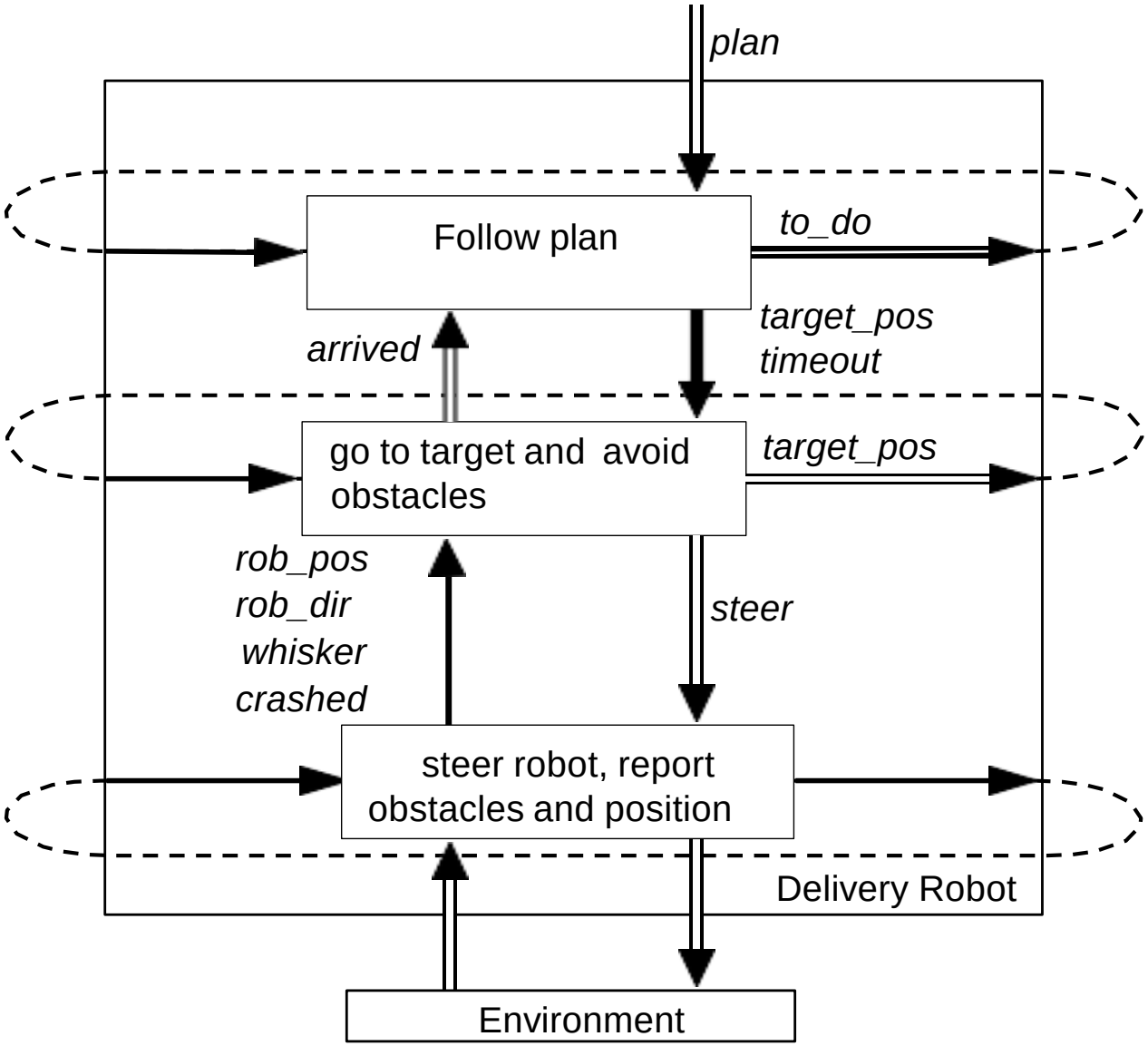
Knowledge & Symbol Levels

- A high-level description is easier for a human to specify and understand.
- A low-level description can be more accurate and more predictive. High-level descriptions abstract away details that may be important for actually solving the problem.
- The lower the level, the more difficult it is to reason with. You may not know the information needed for a low-level description.
- It is sometime possible to use multiple levels of abstraction.

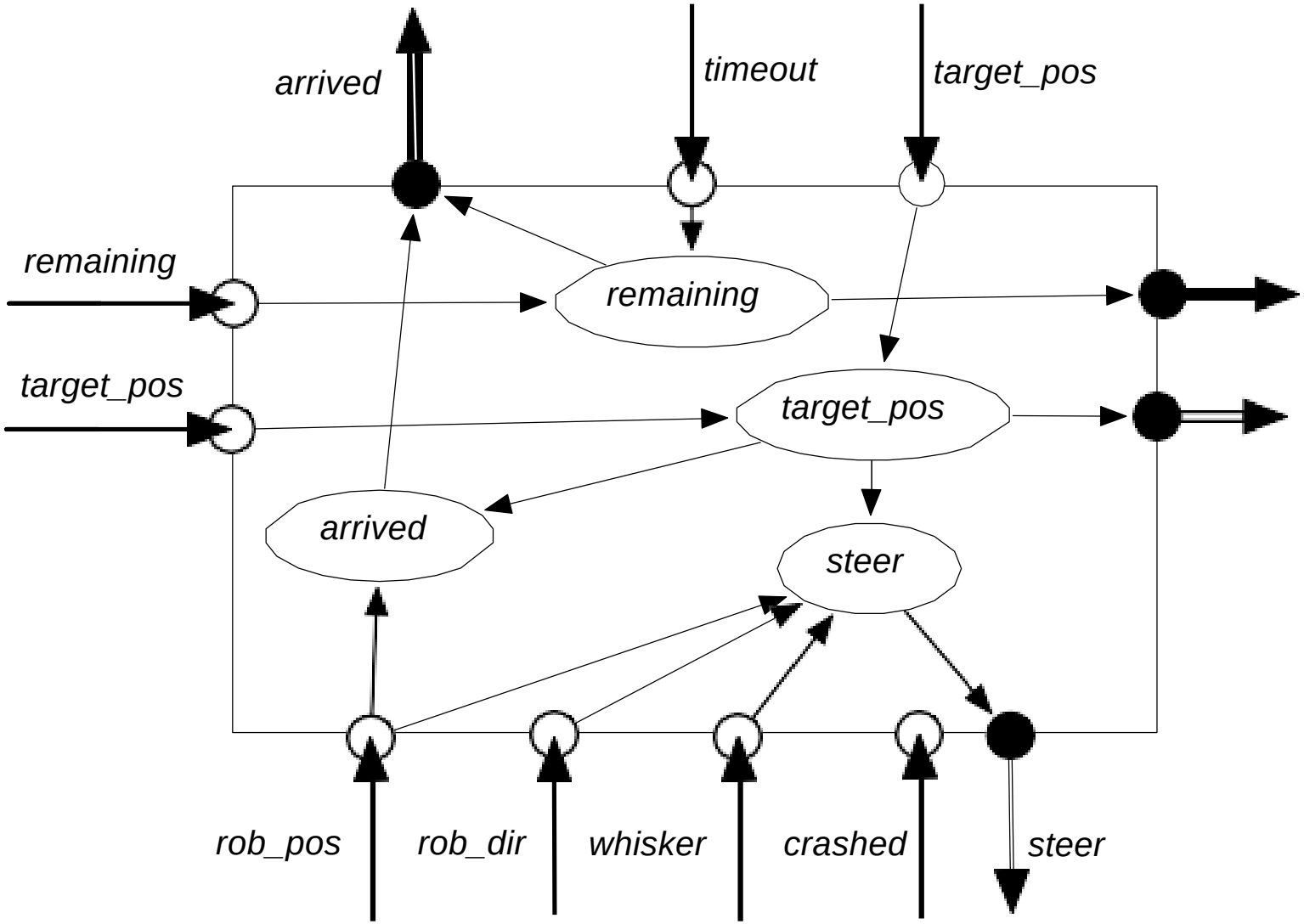
Example: delivery robot

- 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.

Delivery robot architecture



Middle layer of the delivery robot



Middle layer of the delivery robot

given *timeout* and *target pos*:

remaining := *timeout*

while not *arrived*() and *remaining* ≠ 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*

lower.do(*steer*)

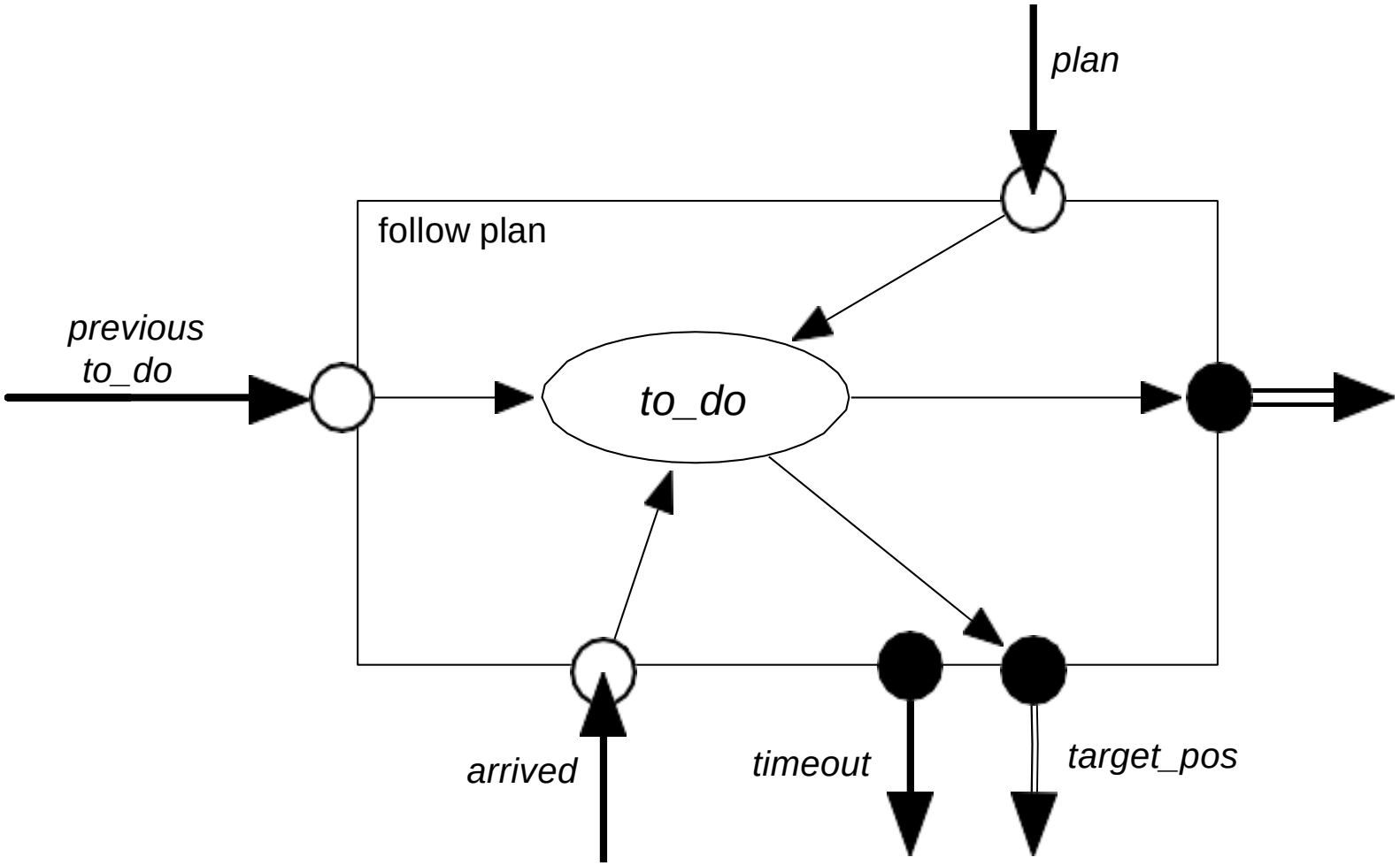
remaining := *remaining* − 1

tell upper layer *arrived*()

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 of the delivery robot



Top layer of the delivery robot

given *plan*:

to_do := *plan*

timeout := 200

while not *empty*(*to_do*)

target_pos := *coordinates*(*first*(*to_do*))

middle.do(*timeout*, *target_pos*)

to_do := *rest*(*to_do*)

to_do list is the first location
rest(to_do) the rest of the to_do list

What should be in an agent's belief state?

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

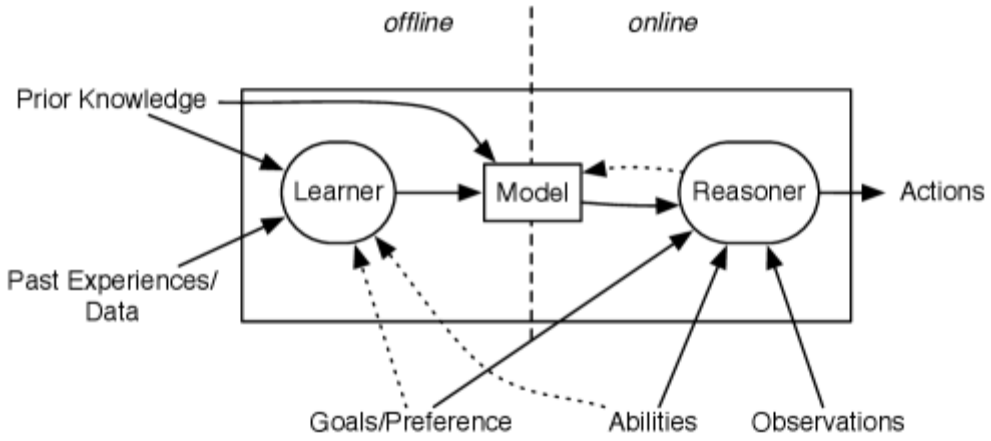
- A purely reactive agent doesn't have a belief state (i.e. action based on its perception).
- A dead reckoning agent doesn't perceive the world (i.e., it exploits forward prediction for the events in the words).

— neither work very well in complicated domains (e.g., affetcted by noise, not deterministic situations).

- It is often useful for the agent's belief state to be a model of the world (itself and the environment).

- The next belief state can be estimated using Bayes's rule (we will see in the next lectures)

Offline and Online Computation



An intelligent agent requires knowledge that is acquired at the design time, offline or online.

Offline: before the agent has to act, the agent uses prior knowledge and past experiences (either its own past experiences or data it has been given) to learn (parts of) a model that is useful for acting online.

Online: when the agent is acting, the agent uses its model, its observations of the world, and its goals and abilities to choose what to do and update its belief state. Online, the information about the particular situation becomes available, and the agent has to act.

An agent typically has much more time for offline computation than for online computation. During online computation it can take advantage of particular goals and particular observations.