# Agent Characteristics and Architectural Types

**INTENTIONAL AGENTS**

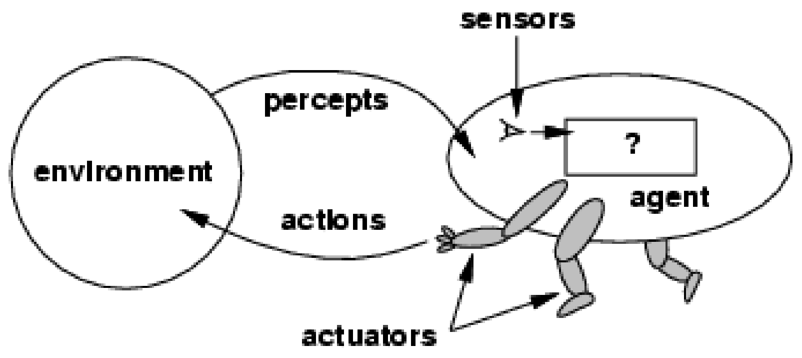
## Sources used

- www.wileyeurope.com/college/fasli
- `http://www.csc.liv.ac.uk/˜mjw/pubs/imas/`
- http://www.cse.sc.edu/~huhns/csce590/

## Defining agents

- “An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors” (Russell and Norvig 2003)
- “Agents are active, persistent (software) components that perceive, reason, act and communicate” (Huhns and Singh 1997)
- “an entity that functions continuously in an environment in which other processes take

# Agents and environments



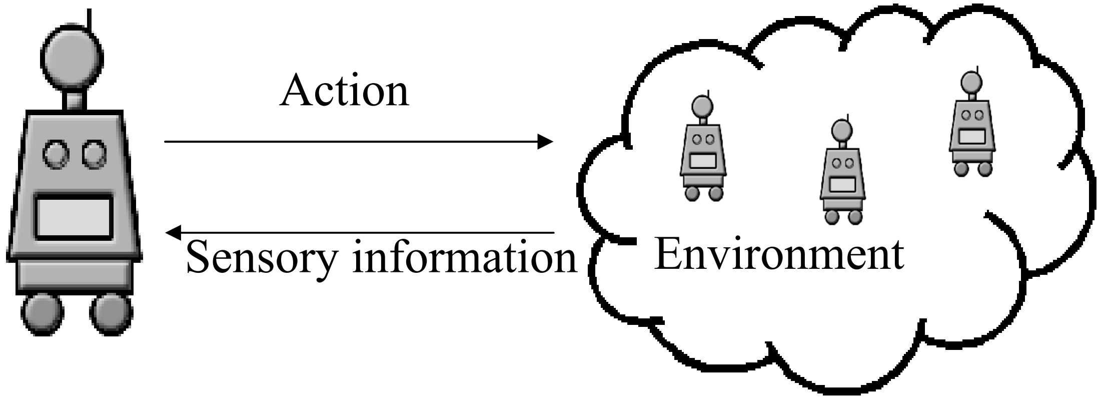
- The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^{*} \rightarrow \mathcal{A}]$$

- The agent program runs on the physical architecture to produce *f*
- agent = architecture + program

# Making decisions

- We require software agents to whom complex tasks and goals can be delegated
- Agents should be smart so that they can make decisions and take actions to successfully complete tasks and goals
- Endowing the agent with the capability to make good decisions is a nontrivial issue



# A simple view of an agent

- Environment states $S=\{s_1, s_2, \ldots\}$
- Perception $see: S \rightarrow P$
- An agent has an internal state ($IS$) which is updated by percepts:

  $next: IS \times P \rightarrow IS$
- An agent can choose an action from a set $A=\{a_1, a_2, \ldots\}$:

  $action: IS \rightarrow A$
- The effects of an agent's actions are captured via the function *do*:

  $do: A \times S \rightarrow S$

## The control loop of such an agent would

```
// control loop for Simple-Agent
begin
    IS:=$IS_0$; // initial internal state
    while true do
        p:=get-percept();
        IS:=update(IS,p);
        action:=choose-best-action(IS);
        execute(action);
    end-while
end
```

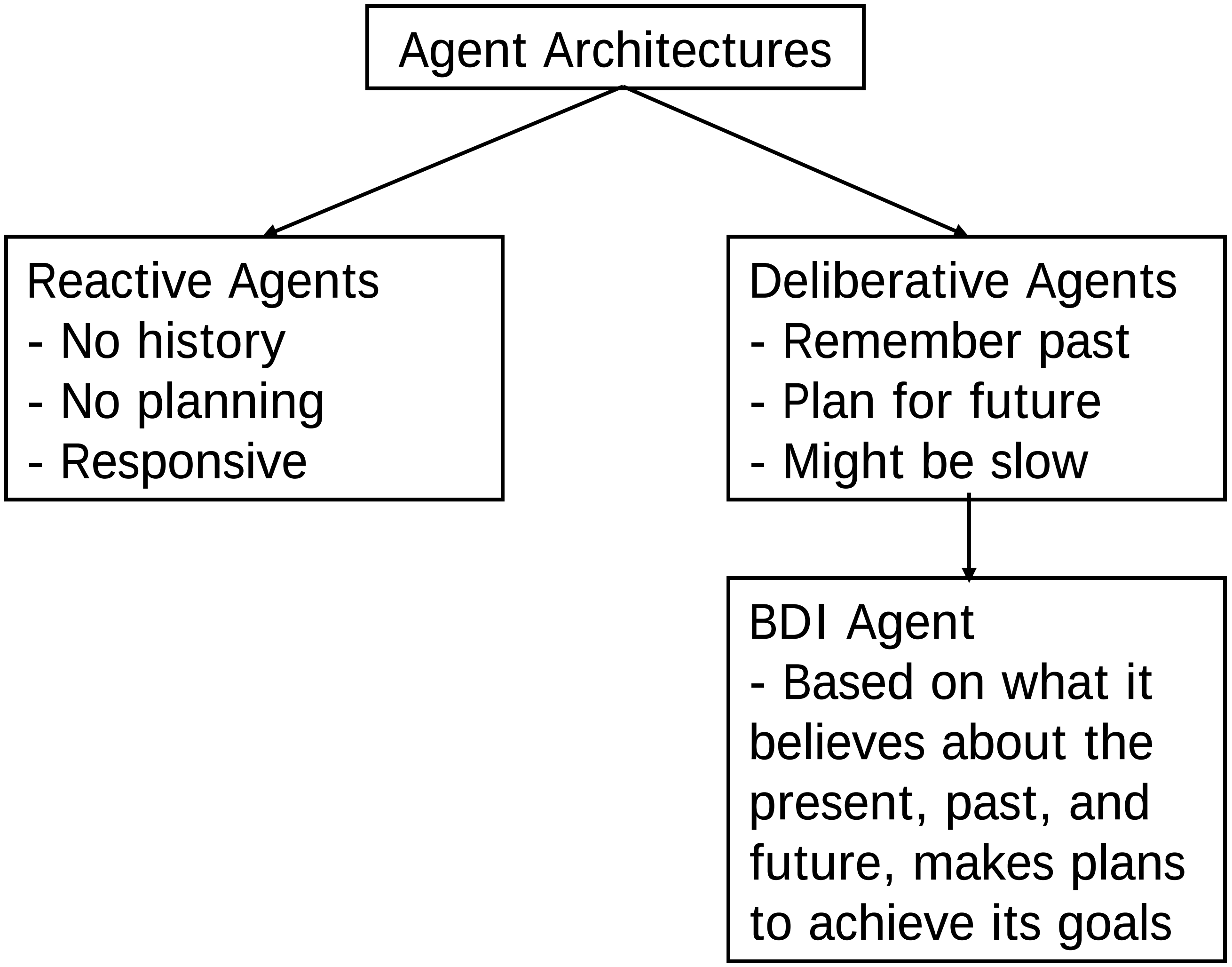
## Characteristics of agents

- Although there is no agreement regarding the definitive list of characteristics for agents, among the most important seem to be:
  - Autonomy
  - Proactiveness
  - Reactiveness
  - Social ability

## Agent Architectures, Abstractly

- Set of environmental states, $S=\{s_1,s_2\ldots.\}$
- Set of actions (of an agent), $A = \{a_1,a_2\ldots.\}$
- Agent: a function $S^* \rightarrow A$, which maps a sequence of environment states to an action
- Behavior (evolution) of an environment: a function, Env: $S \times A \rightarrow \text{Powerset}(S)$
- An agent's interaction with its environment corresponds to a history h, where
  $h = (a_0, s_0) \rightarrow (a_1,s_1) \rightarrow (a_2,s_2) \rightarrow \ldots(a_u, s_u) \rightarrow (a_{u+1},s_{u+1}) \ldots$

# Architectural Types



Agent Architectures

Reactive Agents
- No history
- No planning
- Responsive

Deliberative Agents
- Remember past
- Plan for future
- Might be slow

BDI Agent
- Based on what it believes about the present, past, and future, makes plans to achieve its goals

## Autonomy

- Difficult to pin down exactly: **how self-ruled the agent really is**
- An autonomous agent is one that can interact with its environment without the direct intervention of other agents and has control over its own actions and internal states
- The less predictable an agent is the more autonomous it appears to be to an external observer
- Absolute autonomy (complete unpredictability) may not be desirable; travel agent may exceed the allocated budget
- Restrictions on autonomy via social norms

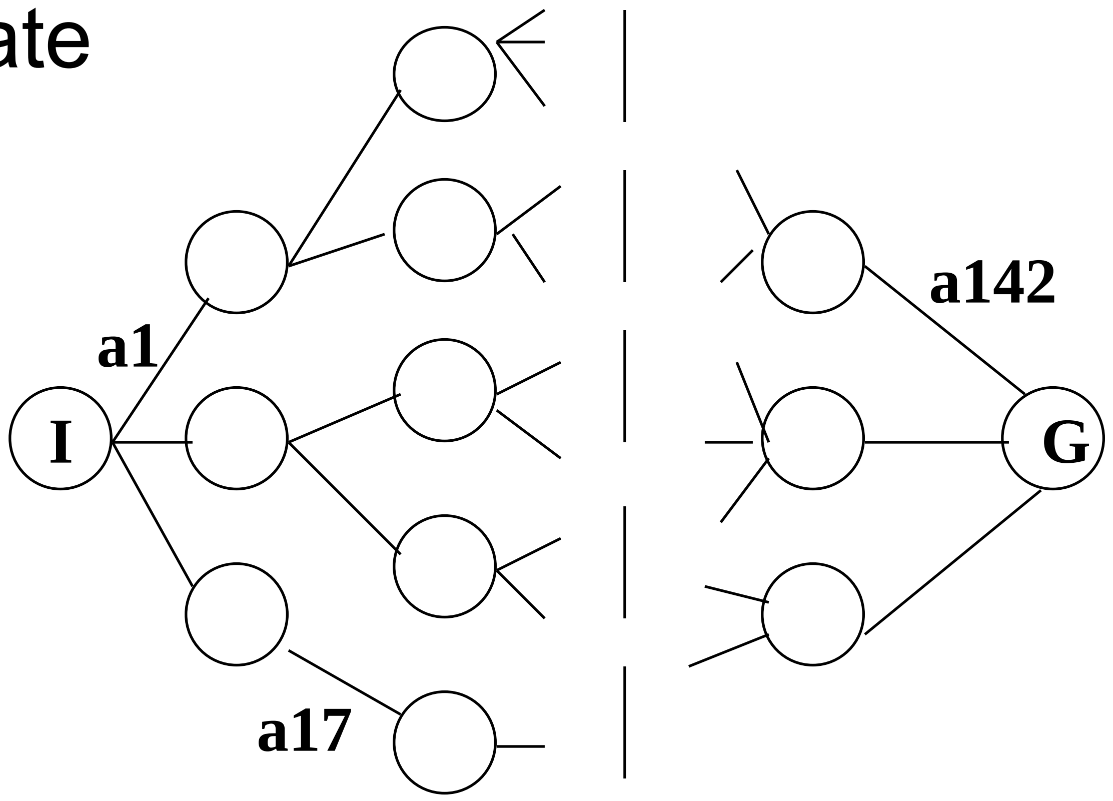
## Proactiveness

- Proactive (goal-directed) behaviour: an agent actively seeks to satisfy its goals and further its objectives
- Simplest form is writing a procedure or method which involves:
  - Preconditions that need to be satisfied for the procedure/method to be executed
  - Postconditions which are the effects of the correct execution of the procedure
  - If the preconditions are met and the procedure executes correctly, then the postconditions will be true

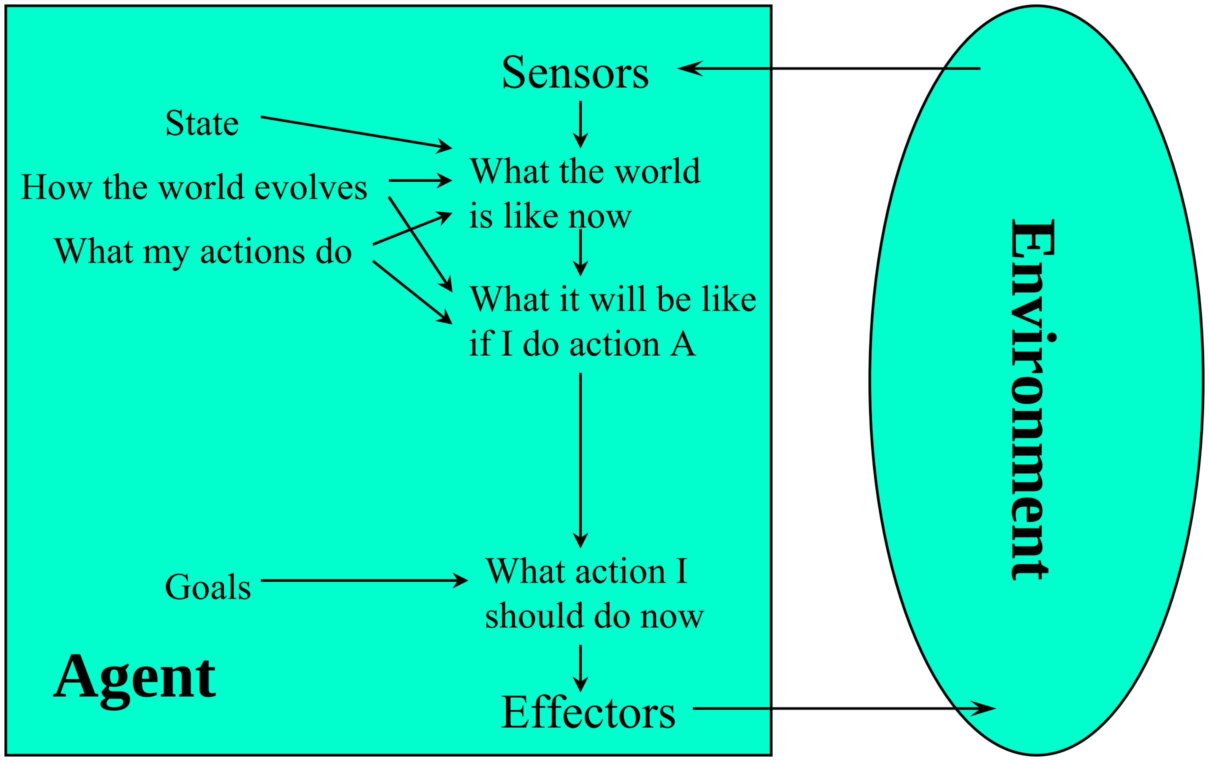
  **ASSIGNMENT**: **Choose an example (may be vacuum cleaner, robots, trade agent or anything else from your area of interest) and write such a procedure**

# Planning Systems (in general)

- Planning systems find a sequence of actions that transforms an initial state into a goal state



# A Goal-Based Agent


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Effectors
Environment

## Performance measure

- Objective: develop agents that perform well in their environments
- A performance measure indicates how successful an agent is
- Two aspects: **how** and **when**

**How** is performance assessed:

- Different performance measures will be suitable for different types of agents and environments
- **Contrast a trading agent with a vacuum cleaning agent**
- Objective performance measures are defined by us as external observers of a system

## Goal states

- One possible way to measure how well an agent is doing is to check that it has achieved its goal
- There may be a number of different action sequences that will enable an agent to satisfy its goal
- A good performance measure should allow the comparison of different world states or sequences of states

# Preferences and utilities

- Agents need to be able to express preferences over different goal states
- Each state *s* can be associated with a utility *u*(*s*) for each agent
- The utility is a real number which indicates the desirability of the state for the agent
- For two states *s* and *s'*
  - agent *i* prefers *s* to *s'* if and only if *u*(*s*)>*u*(*s'*)
  - is indifferent between the two states if and only if *u*(*s*)=*u*(*s'*)

- The agent's objective is to bring about

# Maximum Expected Utility

- In a stochastic environment the performance of an action may bring about

  $$EU(a) = \sum_{s'} P(s'|s,a)u(s')$$

  er of different outcomes
- The expected utility of an action *a* is:

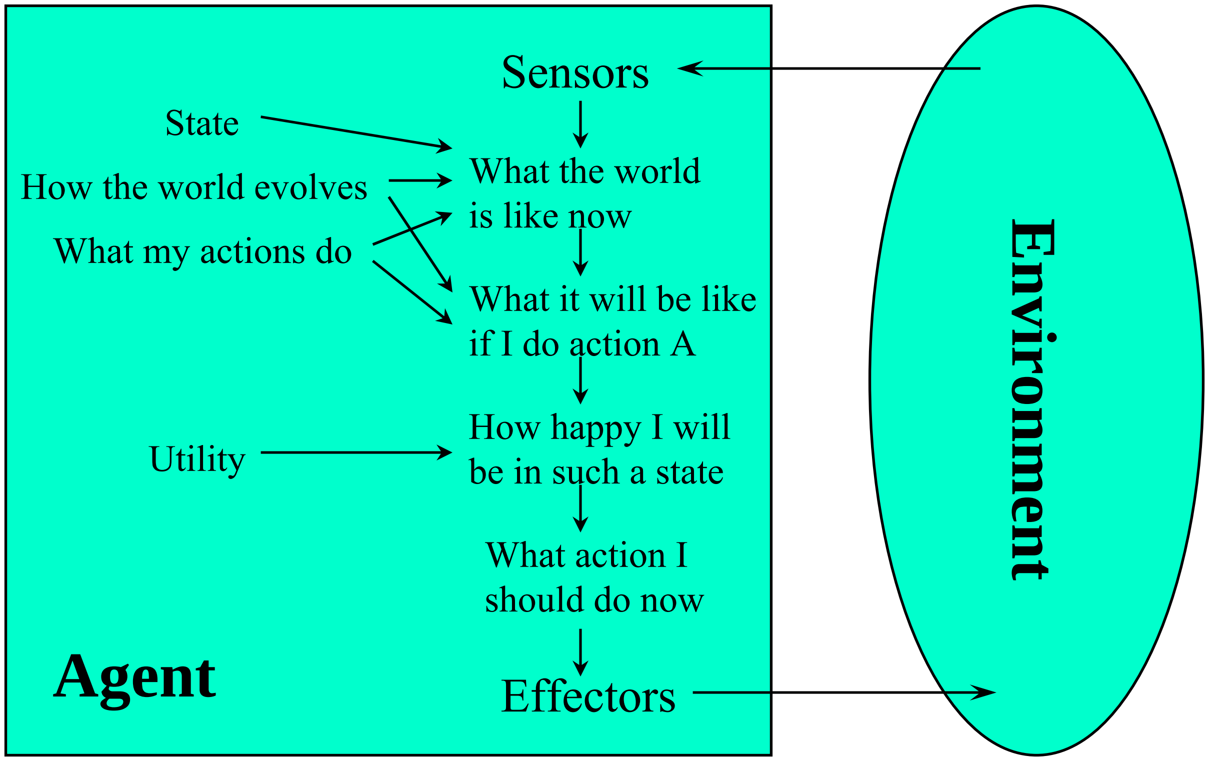
$$a^* = \arg\max_{a \in A} \sum_{s'} P(s'|s,a)u(s')$$

- The agent then chooses to perform action *a** which has the maximum expected utility (MEU):

A complete specification of the utility function allows rational decisions when:

- there are conflicting goals, only some of which can be accomplished; the utility function indicates the appropriate tradeoff;
- there are several goals that the agent can endeavour to achieve, but none of which can be achieved with certainty; the utility provides a way in which the likelihood of success can be evaluated against the importance of the goals

# A Utility-Based Agent


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Agent
Effectors
Environment

## A Utility-Based Agent

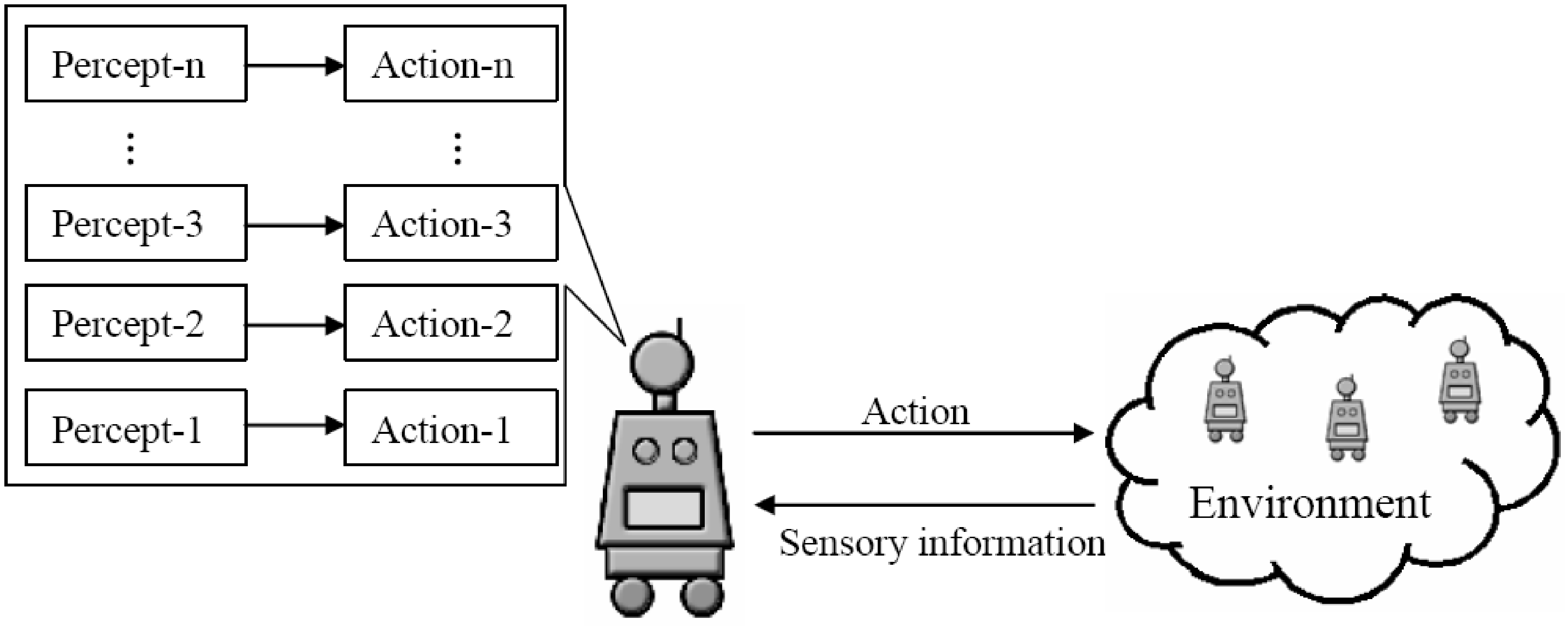
```
function Utility-Based-Agent(percept)
    static: a set of probabilistic beliefs about the
            state of the world

    Update-Probs-for-Current-State(percept,old-action)
    Update-Probs-for-Actions(state, actions)
    Select-Action-with-Highest-Utility(probs)
    return action
```

## Reactiveness

- Goal-directed behaviour as epitomised via the execution of procedures makes two limiting assumptions:
  - while the procedure executes the preconditions remain valid
  - the goal and the conditions for pursuing such a goal, remain valid at least until the procedure terminates
- Not realistic in dynamic, complex and uncertain environments
- Agents must not blindly attempt to achieve their goals, but should perceive their environment and any changes that affect their goals and respond accordingly
- Building an agent that achieves a balance between proactive and reactive behaviour is difficult

# Reactive Architecture

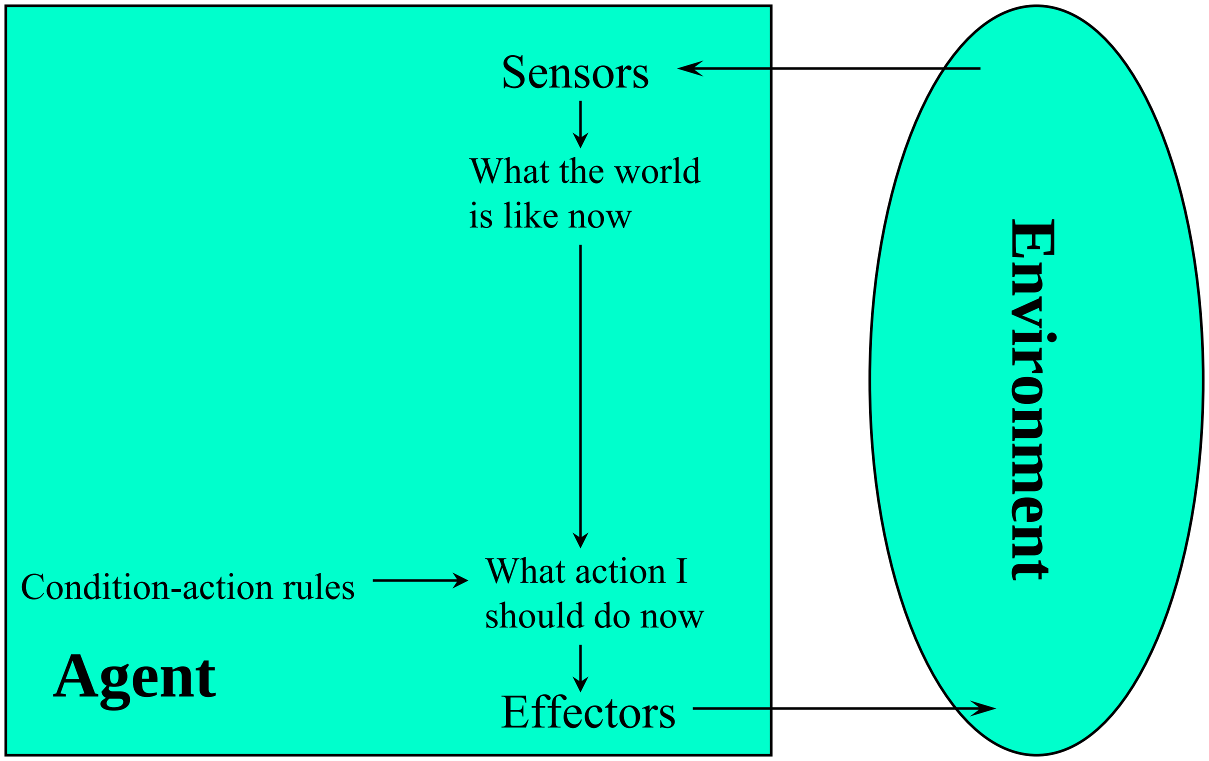
- Seeks to produce intelligent behavior without explicit
  - Symbolic representations
  - Abstract reasoning
- Intelligence is an emergent property of certain complex systems (depends on the environment too, not just the agent)
  - Cannot plan to drive a car to full detail
  - Reactively avoiding collisions while heading toward an attractor indicates intelligence


Percept-n
Action-n
Percept-3
Action-3
Percept-2
Action-2
Percept-1
Action-1
Action
Sensory information
Environment

## Reactive architecture

- The use of an internal representation and decision-making based on it is rejected
- ‘Smart’ behaviour is linked directly to the environment that the agent inhabits and can be generated by responding to changes
- The representation of the world is built into the agent’s sensory and effectory capabilities; perceptual input is mapped to actions

# A Simple Reactive Agent

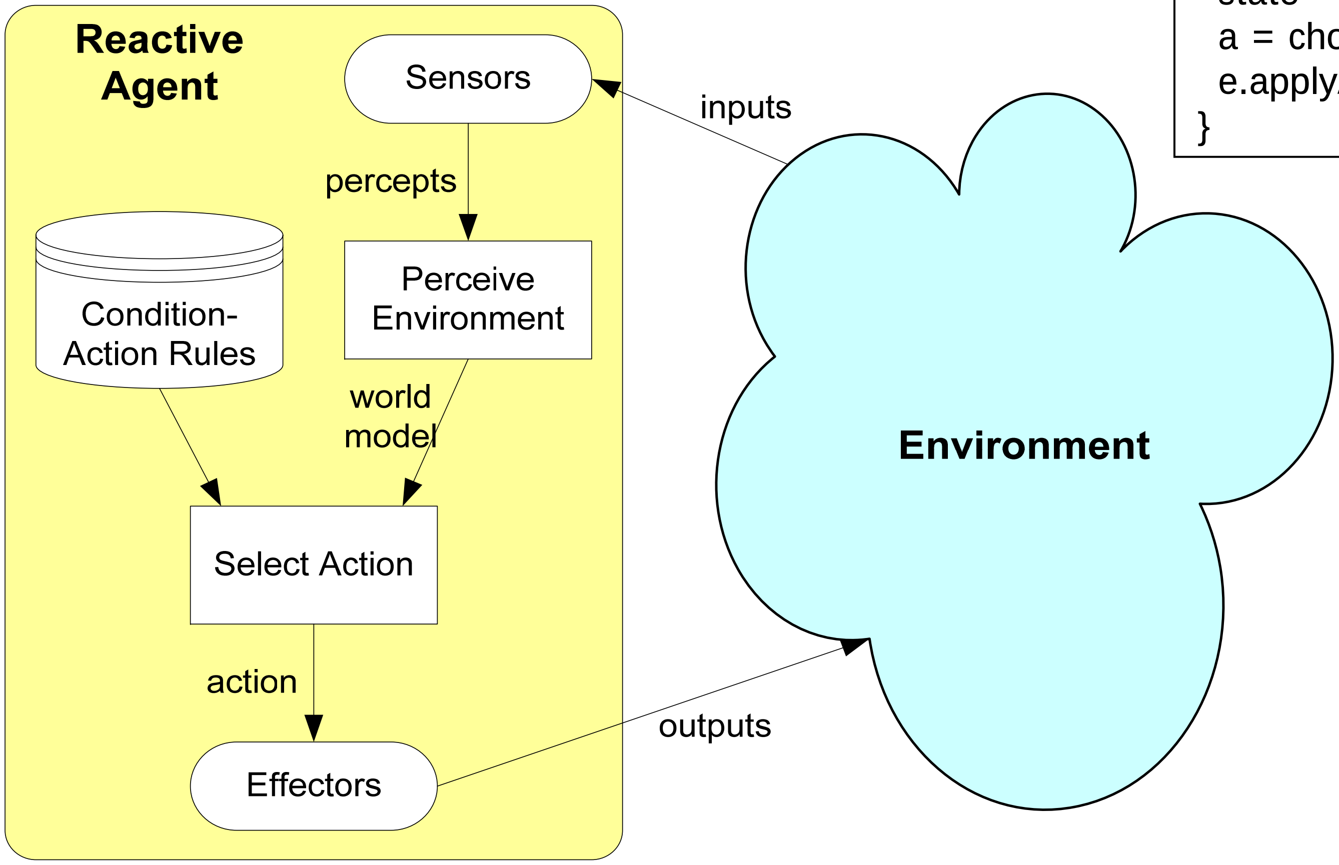


# A Simple Reactive Agent

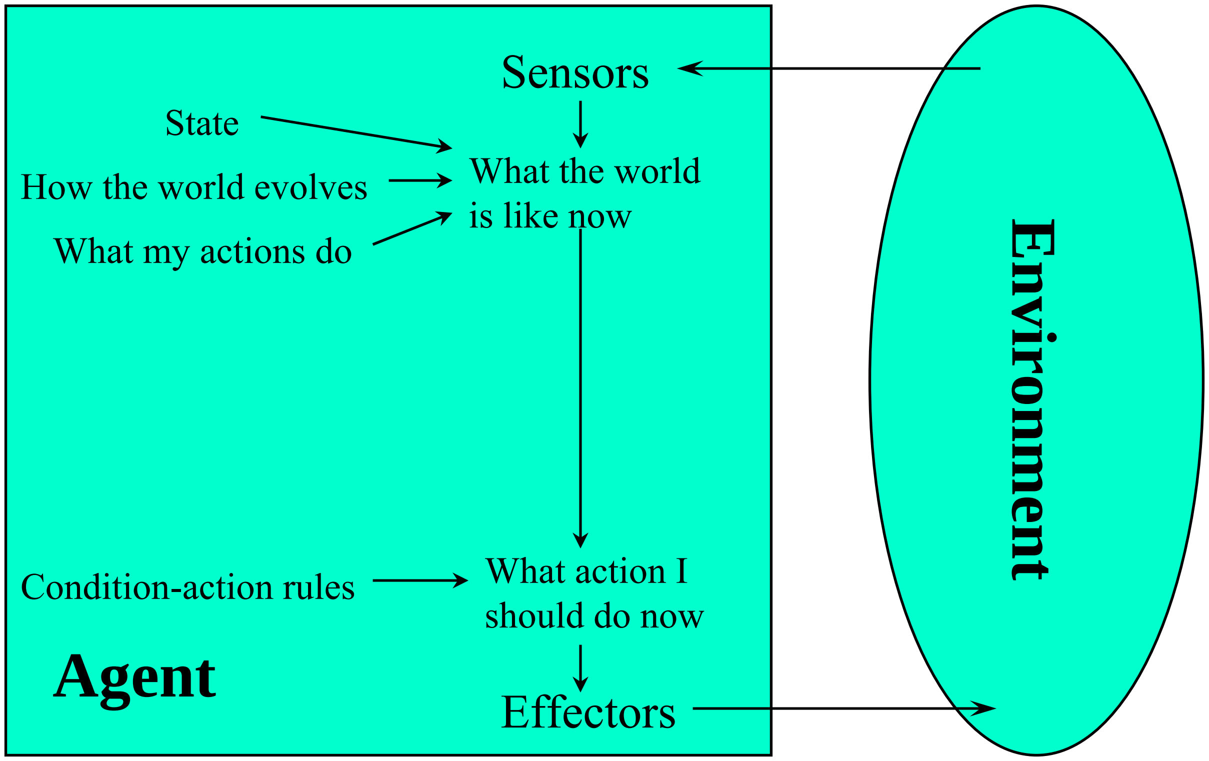
```
function Simple-Reactive-Agent(percept)
  static: rules, a set of condition-action rules

  state <–– Interpret-Input(percept)
  rule <–– Rule-Matching(state, rules)
  action <–– Rule-Action(rule)
  return action
```

# A Reactive Agent in an Environment

```
Environment e;
RuleSet r;
while (true) {
  state = senseEnvironment(e);
  a = chooseAction(state, r);
  e.applyAction(a);
}
```



# A Reactive Agent with State



## A Reactive Agent with State

```
function Reactive-Agent-with-State(percept)
  static: rules, a set of condition-action rules
          state, a description of the current world
  state <–– Update-State(state, percept)
  rule <–– Rule-Matching(state, rules)
  action <–– Rule-Action(rule)
  state <–– Update-State(state, action)
  return action
```

## Brooks' position

The first to reject the idea of a symbolic model was Brooks

- 'Real' intelligence is situated in the world and not in disembodied systems
- Intelligence is an emergent property
- Intelligent behaviour can be generated without an explicit internal representation and without explicit reasoning, but by the interaction of simple behaviours

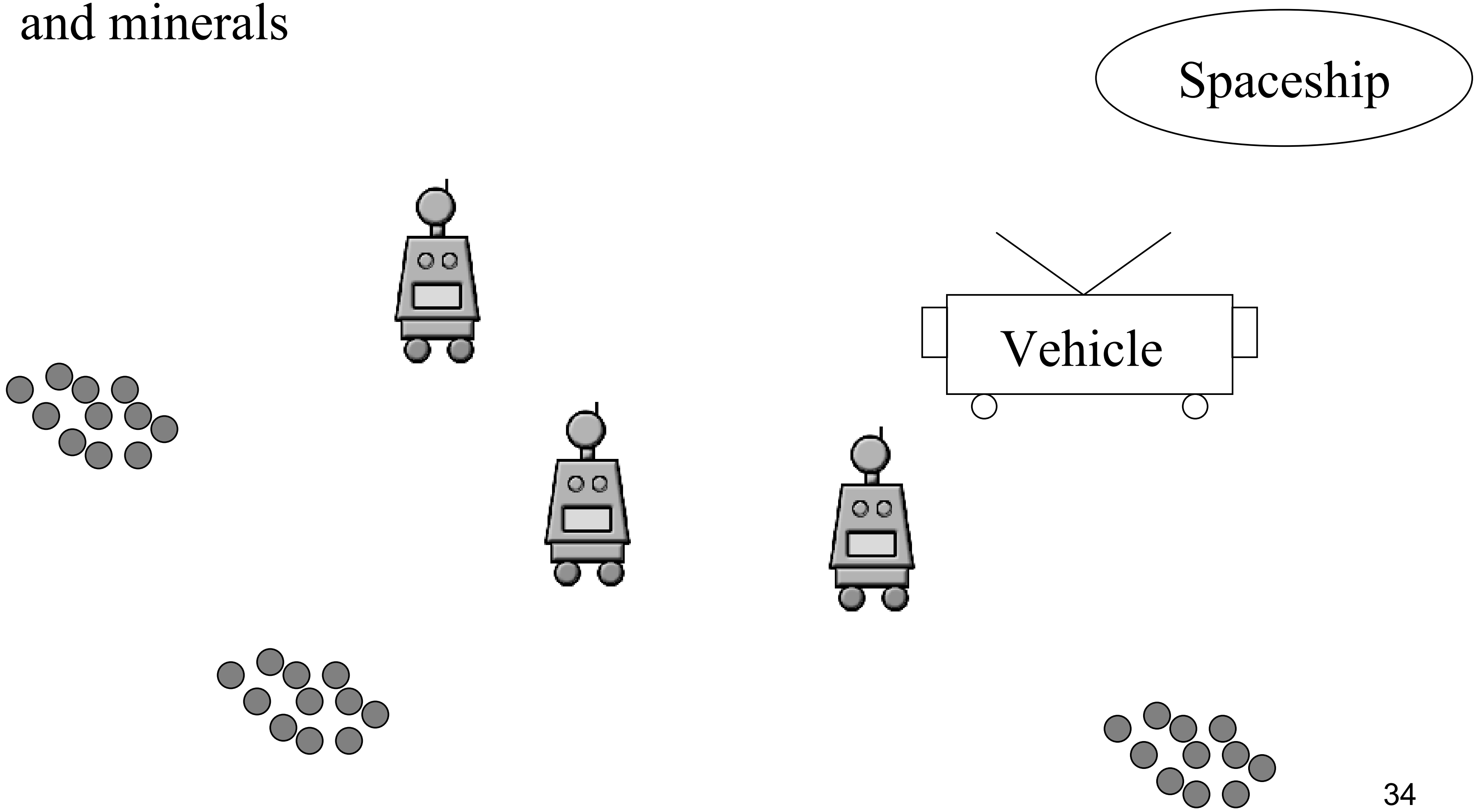
## Subsumption Architecture

Brooks proposed the Subsumption Architecture

- Task Accomplishing Behaviours (TABs): a TAB can be a finite state machine or a rule of the form *situation →action*
- Each behaviour achieves a task and can be considered as an individual action function which takes sensory input and maps it to an action to be performed
- Many behaviours can 'fire' simultaneously

# The Luc Steels scenario

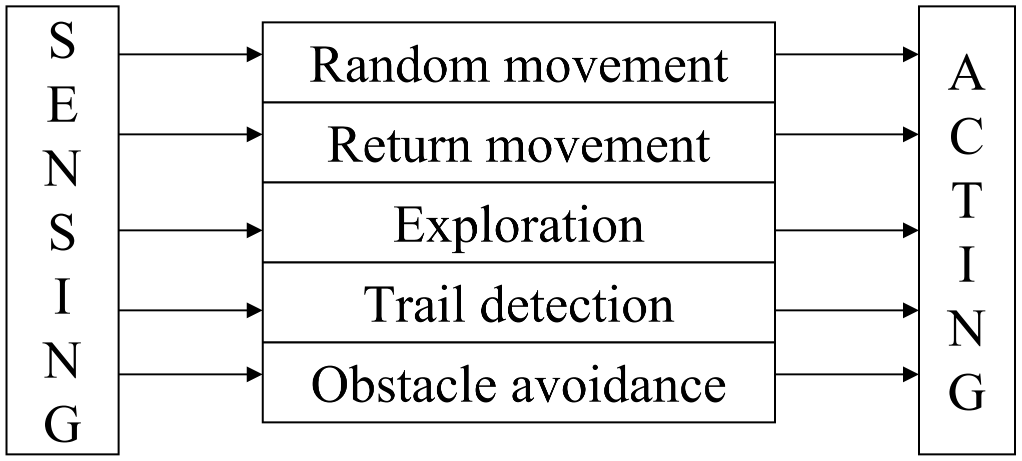
A mission to a distant planet to collect samples of rocks and minerals



- Each behaviour may encompass more than one situation-action rules, for example:

  *if detect crumb → pick up 1 and travel down the gradient*

  *if carrying samples and not at the base → drop 2 crumbs and travel up the gradient*
- The behaviours are arranged in a subsumption hierarchy



## Advantages of the reactive approach

- Simple and elegant
- An agent’s behaviour is computationally tractable
- Very robust against failure
- The power lies in numbers: complex tasks can be accomplished by a group of simple reactive agents
- Complex behaviours emerge from the interaction of simple ones

# Disadvantages of the reactive approach

- With no model of the environment, the agents need sufficient information about their current state in order to determine an action
- Short-sighted and with no planning capabilities
- Learning is difficult to achieve
- *Emergence* or *emergent behaviour* is not yet fully understood and it is even more difficult to engineer

## Social ability

- Agents are rarely isolated entities, they usually live and act in an environment with other agents, human or software
- Social ability means being able to operate in a multi-agent environment and coordinate, cooperate, negotiation and even compete with others
- This social dimension of the notion of agency must address many difficult situations which are not yet fully understood, even in the context of human behaviour

## Agents as intentional systems

- Trying to understand and analyze the behaviour of complex agents in a natural, intuitive and efficient way is a nontrivial task
- Methods that abstract us away from the mechanistic and design details of a system may be more convenient
- **Intentional stance**: ascribing to a system human mental attitudes (anthropomorphism), e.g. beliefs, desires, knowledge, wishes

## Agents as Intentional Systems

- When explaining human activity, it is often useful to make statements (‘**intentional stance**’) such as the following:

  Janine took her umbrella because she *believed* it was going to rain.

  Michael worked hard because he *wanted* to possess a PhD.

- These statements make use of a *folk psychology*, by which human behavior is predicted and explained through the attribution of *attitudes*, such as believing and wanting (as in the above examples), hoping, fearing, and so on
- The attitudes employed in such folk psychological descriptions are called the *intentional* notions

## Agents as Intentional Systems

- Is it legitimate or useful to attribute beliefs, desires, and so on, to computer systems?

How useful/legitimate is this approach? McCarthy explains:

*To ascribe certain 'beliefs', 'knowledge', 'free will', 'intentions', consciousness', 'abilities' or 'wants' to a machine or computer program is legitimate when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behaviour, or how to repair or improve it.*

## Agents as Intentional Systems

- What objects can be described by the intentional stance?
- As it turns out, more or less anything can. . . consider a light switch:

‘It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, who invariably transmits current when it believes that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires’. (Yoav Shoham)

# Agents as Intentional Systems

- The answer seems to be that while the intentional stance description is consistent,

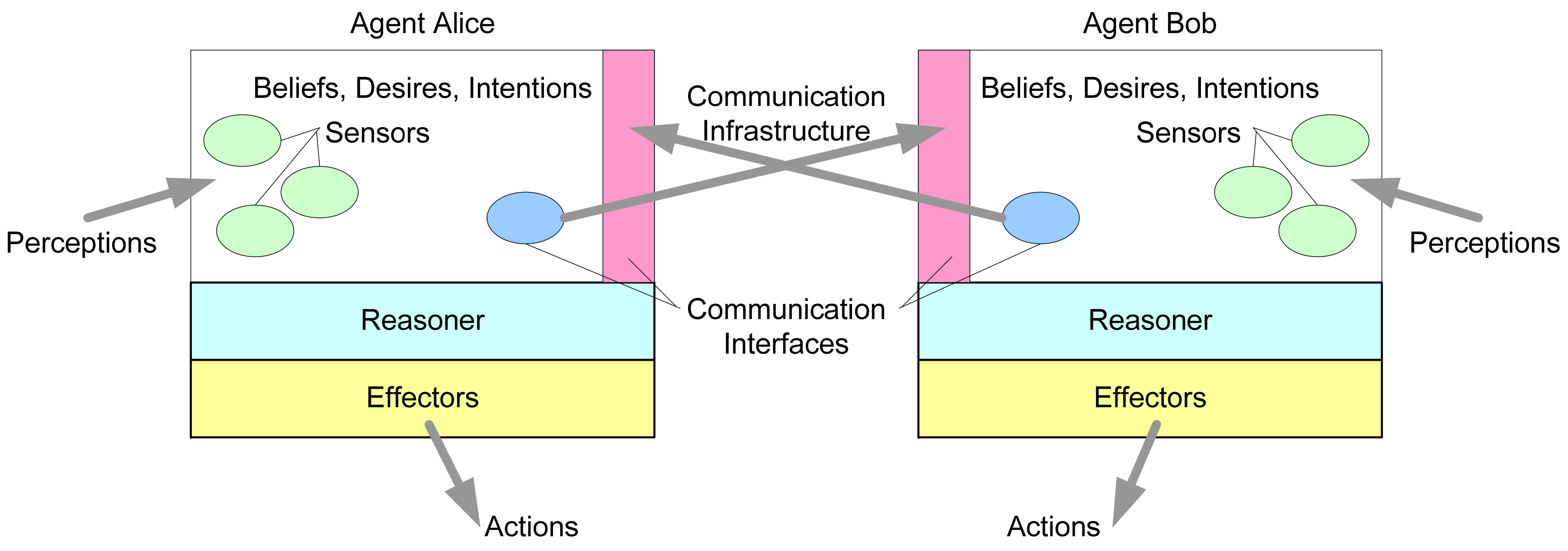
  > . . . it does not *buy us anything*, since we essentially understand the mechanism sufficiently to have a simpler, mechanistic description of its behavior.
  >
  > (Yoav Shoham)

- Put crudely, the more we know about a system, the less we need to rely on animistic, intentional explanations of its behavior
- But with very complex systems, a mechanistic, explanation of its behavior may not be practicable
- *As computer systems become ever more complex, we need more powerful abstractions and metaphors to explain their operation — low level explanations become impractical. The intentional stance is such an abstraction*

- The intentional stance provides us with a powerful abstraction tool: the behaviour of systems whose structure is unknown can be explained
- Computer systems or programs are treated as rational agents.
- An agent possesses knowledge or beliefs about the world it inhabits, it has desires and intentions and it is capable of performing a set of actions
- An agent uses practical reasoning and based on its information about the world and its chosen desires and intentions will select an action which will lead it to the achievement of one of its goals

# Cognitive Architecture for an Agent

Called a BDI (beliefs, desires, intentions) architecture



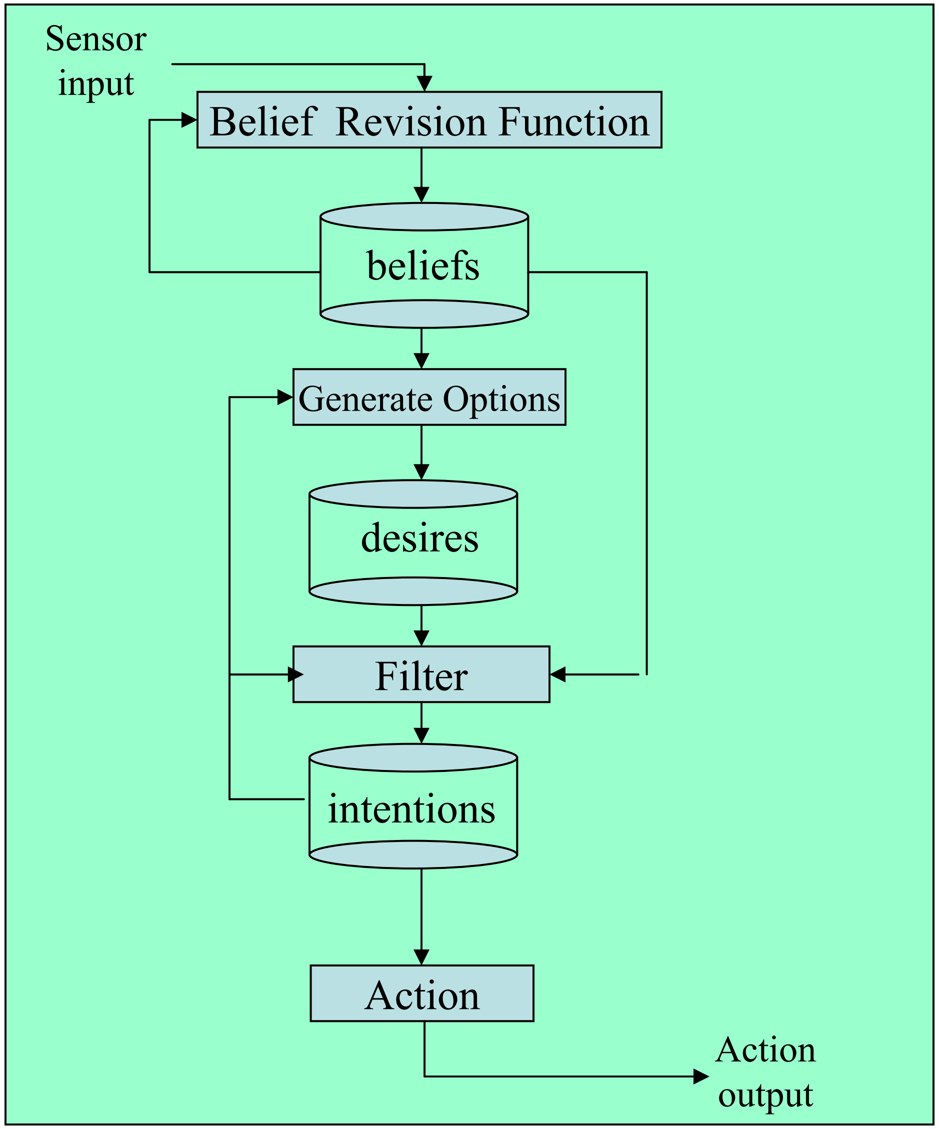
Like the reactive architecture at a coarse level, but with two differences:

- Cognitive representations
- Deeper reasoning based on the above representations

## BDI

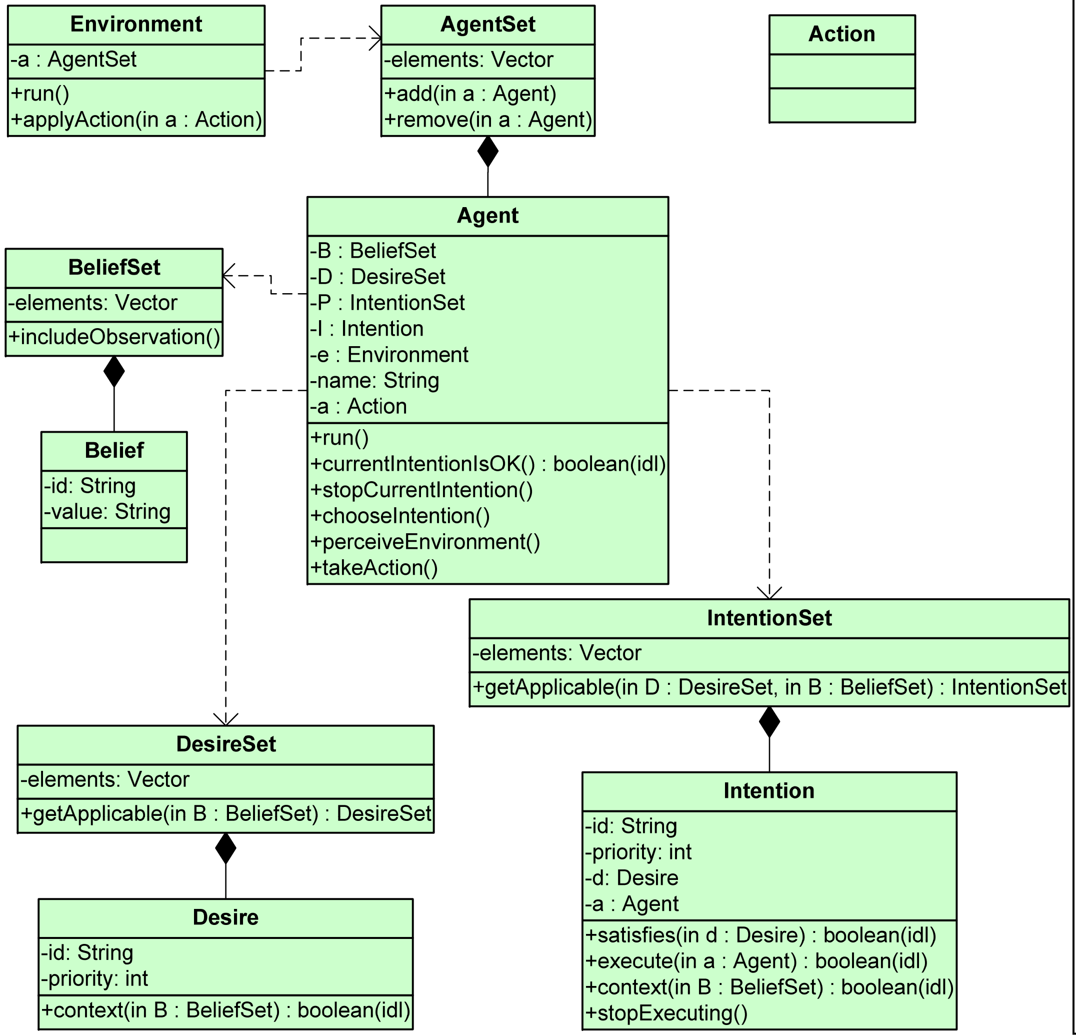
- Deciding on what goals to achieve and how to achieve them
  - *Beliefs:* the information an agent has about its surroundings
  - *Desires:* the things that an agent would like to see achieved
  - *Intentions:* the desires that an agent is working on; also involves a deeper personal commitment
- A BDI architecture addresses how beliefs, desires and intentions are represented, updated, and processed

# Generic BDI Architecture



Generating options and filtering options are together called *deliberation*

# Architecture of BDI-Based Agent



Execution Cycle:

1. New information arrives that updates beliefs and goals
2. Actions are triggered by new beliefs or goals
3. A triggered action is intended
4. An intended action is selected
5. The selected intention is activated
6. An action is performed
7. New beliefs or goals are stored
8. Intentions are updated

## Belief-Desire-Intention architecture

Based on Bratman's (1987) philosophical theory of practical reasoning

- Theoretical reasoning: the processes of deriving knowledge or reaching conclusions using one's beliefs and knowledge
- Practical reasoning: deciding what to do
  - Deliberation: deciding *what* to achieve
  - Means-end reasoning: deciding *how* to achieve it
- Beliefs: an agent's information about the world
- Desires: an agent's motivation or possible options

- Intentions: an agent's commitments

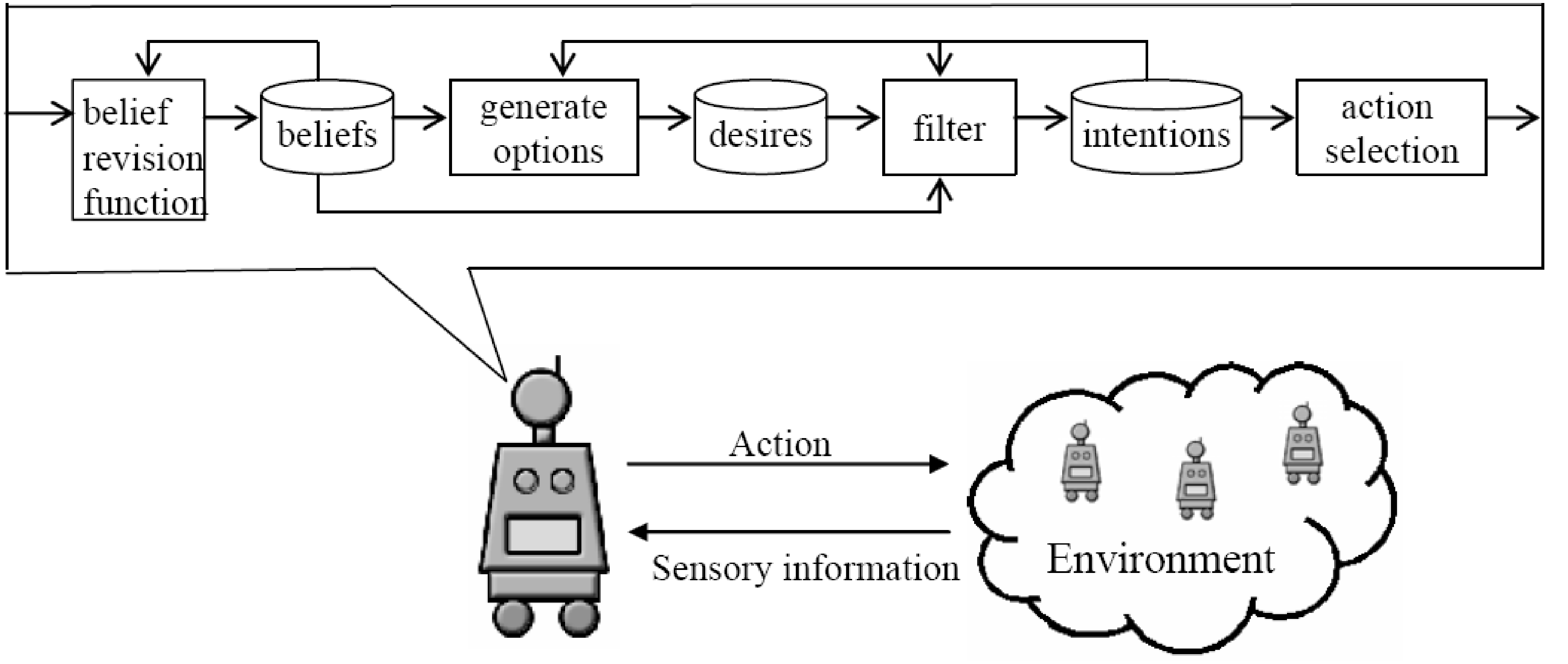
## The role of intentions

- Intentions are key in practical reasoning:
  - They describe states of affairs that the agent has committed to bringing about and as a result they are action-inducing
  - They resist reconsideration, are volitional and reasoning-centered, and one intention leads to further being generated
- Forming intentions is critical to an agent’s success

Intentions play three characteristic roles (Bratman 1987):

- An agent needs to determine ways to achieve its intentions
- An agent’s intentions lead it to adopt or refrain from adopting further intentions
- An agent is interested in succeeding in its intentions and thus it keeps track of its attempts to do so

# BDI components



- A set of current beliefs (*Beliefs*)
- A belief revision function

  *brf*: $P(Beliefs) \times P \rightarrow P\ (Beliefs)$
- A set of desires (*Desires*)
- An option generation function:

  *options*: $P(Beliefs) \times P(Intentions) \rightarrow P(Desires)$
- A set of current intentions (*Intentions*)
- A filter function

  *filter*: $P(Beliefs) \times P(Desires) \times P(Intentions) \rightarrow P\ (Intentions)$
- An action selection function

  *asf*: $P(Intentions) \rightarrow A$

Control loop for a BDI agent would take the form:

```
// control loop for BDI-Agent
begin
    B:=$B_0$;  // initial beliefs
    I:=$I_0$; // initial intentions
    while true do
        p:=get-percept();
        B:=brf(B,p); // update beliefs
        D:=options(B,I) // generate options
        I:=filter(B,D,I) // determine intentions
        action:=asf(I) // select an intention to be executed
        execute(action)
    end-while
end
```

# Advantages of the BDI approach

- Clear and intuitive
- Clear functional decomposition of the agent’s subsystems
- The formal properties can be studied (BDI logics)

## Disadvantages of the BDI approach

- Although the decomposition of the subsystems is clear, how to efficiently implement their functionality is not
- Agents need to achieve a balance between commitment and reconsideration
  - This depends on the rate of the environment change
  - Bold and cautious agents
  - If the rate of the environment change is low, then bold agents do better
  - If the rate of the environment change is high, then cautious agents do better

## Agents vs objects

In the object-oriented approach

- objects have identity, state and behaviour and communicate via messages
- A message is a procedure call which is executed in the context of the receiving object

In the agent-oriented approach

- Agents have identity, state (knowledge, beliefs, desires, intentions) and behaviour (plans to achieve goals, actions, reactions to events and even roles)

So are objects agents?

- Agents exhibit autonomy, they have control over their state, execution and ultimately behaviour
- Agents exhibit goal-directed, reactive and social behaviour
- They are persistent, self-aware and able to learn and adapt
- Agents have their own thread of control in a multi-agent system
- Control in multi-agent systems is also distributed

# Assignment 3 due on September 30

- **(1) From slide 12:** Write a procedure for proactive agents for an example (may be vacuum cleaner, robots, trade agent or anything else from your area of interest)
- **(2) From slide 16:** Give examples of performance measure for a trading agent and for a vacuum cleaning agent and discuss comparatively
- **(3)** Discuss how can the agents characteristics: AUTONOMY, PROACTIVENESS, REACTIVENESS, and SOCIAL ABILITY be implemented. Do they depend on the type of agent architecture? How is each of them related to intentionality? – max one page!
- **(4) Examine the BDI class architecture (slide 49), and with an example in mind (see slide 53) describe how would you implement each of the key components of this architecture (e.g., beliefs and desires)**