Lab tomorrow at 8:30AM !

- We will run together a simple exercise to get you started with Brahms.

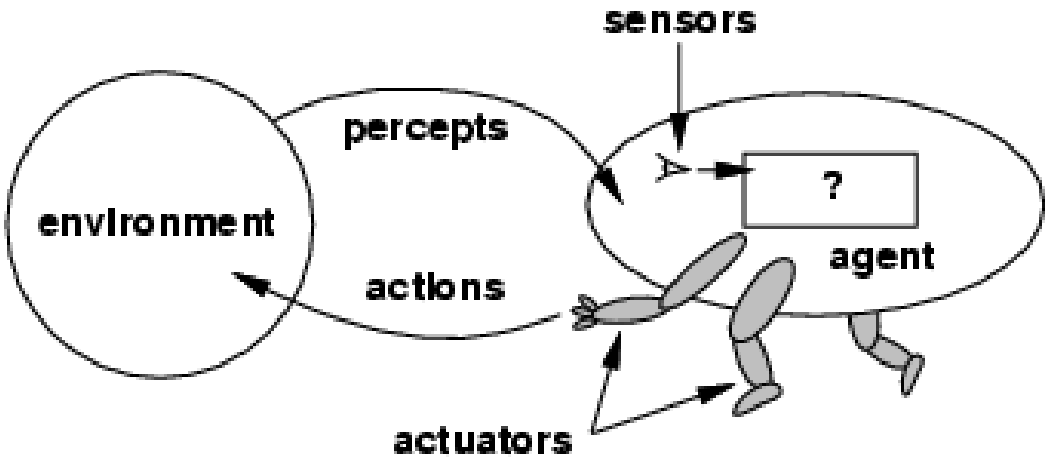
- It is already posted on the course website, under ‘Lab Materials’ link.

Wrap-Up on Agents

- Characterization of Agents and environments
- **Rationality**
- **PEAS** (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

AI as in ‘Acting Rationally’

- Definition: An **intelligent agent** perceives its environment via **sensors** and acts rationally upon that environment with its **actuators**



Rational agents

- An agent should strive to "do the right thing", based on what:
  - it can perceive and
  - the actions it can perform.

→ Performance measure: An *objective criterion for success* of an agent's behavior

- The right action is the one that will cause the agent to be most successful

Performance measure of a vacuum-cleaner agent:
E.g., •amount of dirt cleaned up; •amount of time taken,
•amount of electricity consumed, •amount of noise generated, etc.

Rational agents

- Rational Agent: For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

How do we design the agents?

- Must first specify the setting for intelligent agent design
- PEAS: Performance measure, Environment, Actuators, Sensors
- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

The taxi driver agent

Agent Type	Percepts	Actions	Goals	Environment
Taxi driver	Cameras, speedometer, GPS, sonar, microphone	Steer, accelerate, brake, talk to passenger	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers

Performance measures

Getting to the correct destination, minimizing fuel consumption and wear and tear, minimizing trip time and cost, minimizing traffic violations and disturbances of other drivers, maximizing safety and passenger comfort.

Operating environment

City streets? highways? snow and other road hazards? driving on right or left?

- **Performance measure** Safe, fast, legal, comfortable trip, maximize profits
- **Environment** Roads, other traffic, pedestrians, customers
- **Actuators** Steering wheel, accelerator, brake, signal, horn
- **Sensors** Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

Agent Type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

PEAS

- Agent: Medical diagnosis system
- Performance measure: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

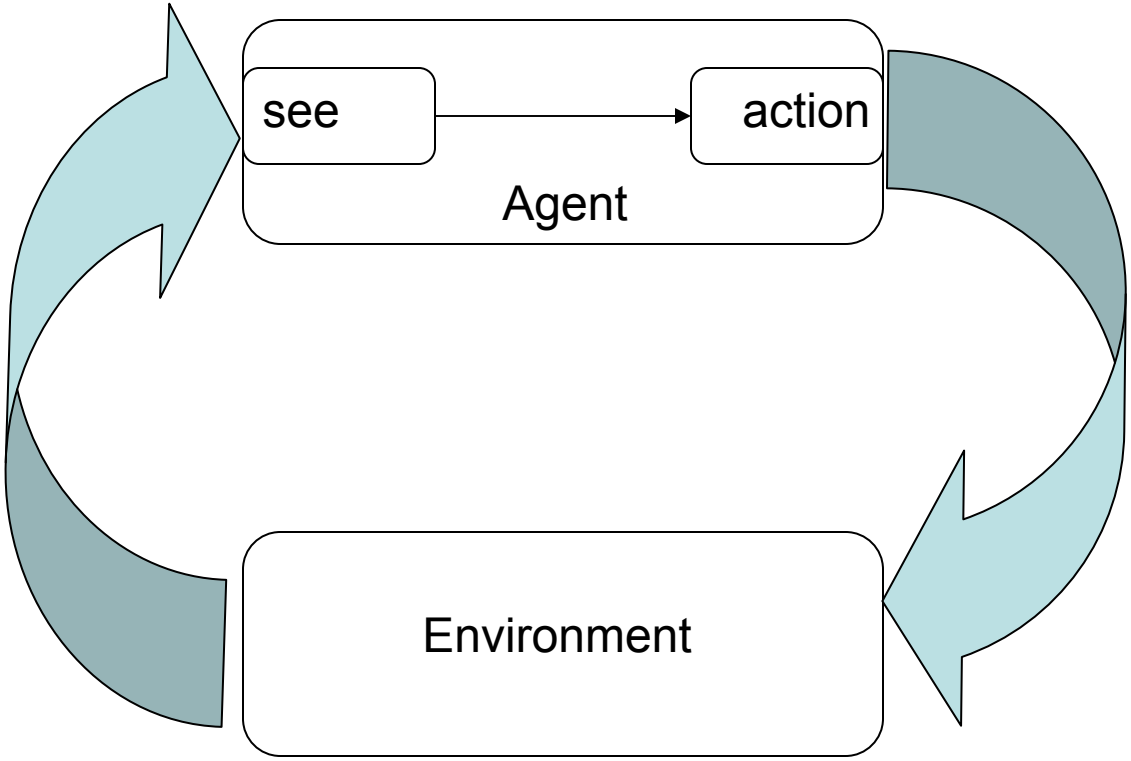
PEAS

- Agent: Part-picking robot
- Performance measure: Percentage of parts in correct bins
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

- **(0) Table-driven agents**
  - use a percept sequence/action table in memory to find the next action. They are implemented by a (large) **lookup table**.
- **(1) Simple reflex agents**
  - are based on **condition-action rules**, implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.
- **(2) Agents with memory** - **Model-based reflex agents**
  - have **internal state**, which is used to keep track of past states of the world.
- **(3) Agents with goals** – **Goal-based agents.**
  - are agents that, in addition to state information, have **goal information** that describes desirable situations. Agents of this kind take future events into consideration.
- **(4) Utility-based agents**
  - base their decisions on **classic axiomatic utility theory** in order to act rationally.
- **(5) Learning agents** – they have the ability to improve performance through learning.

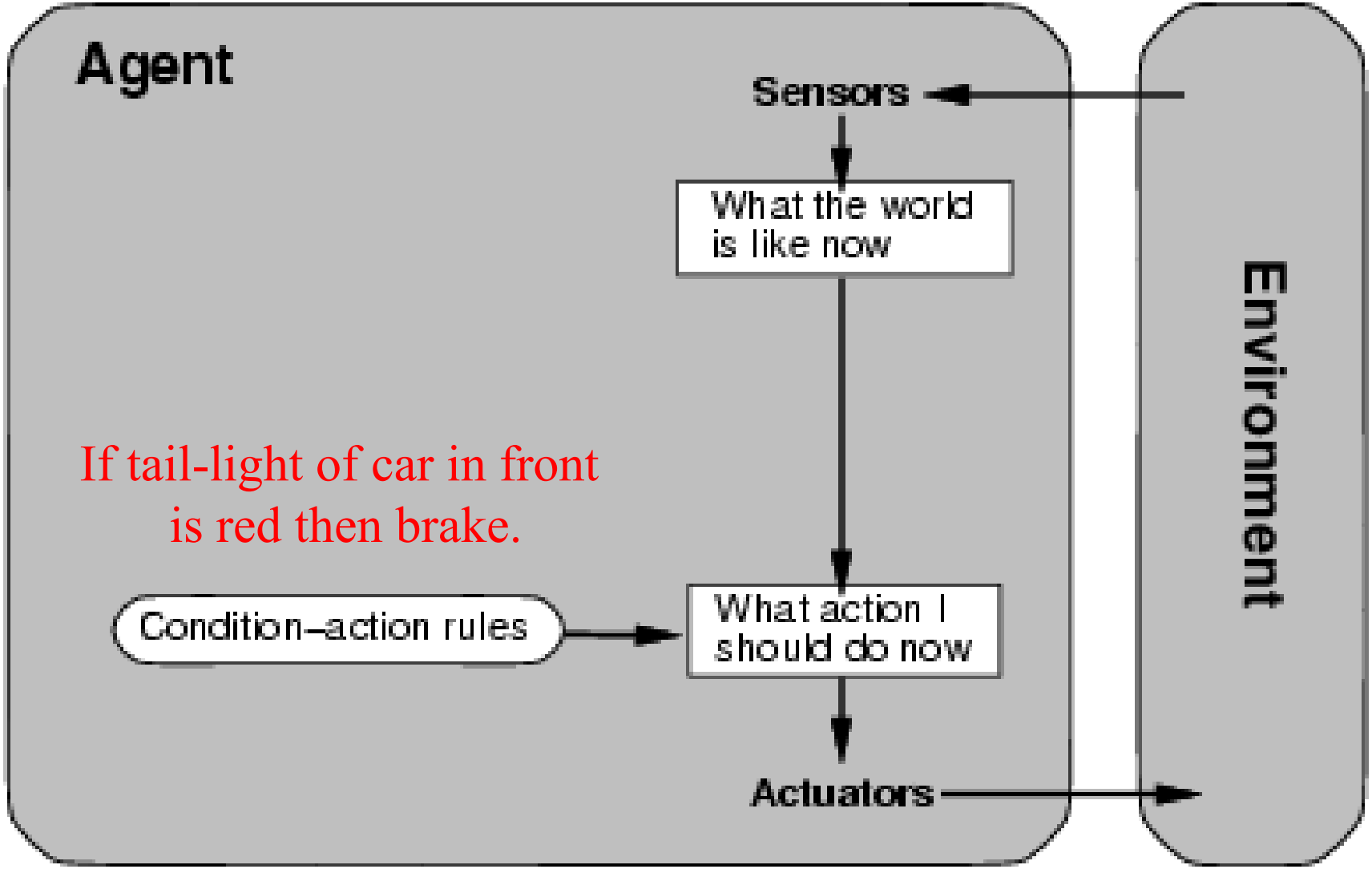
Perception

- Now introduce *perception* system:



Simple reflex agents (Rules)

Agent selects actions on the basis of *current* percept only → only possible if **environment is fully observable**.

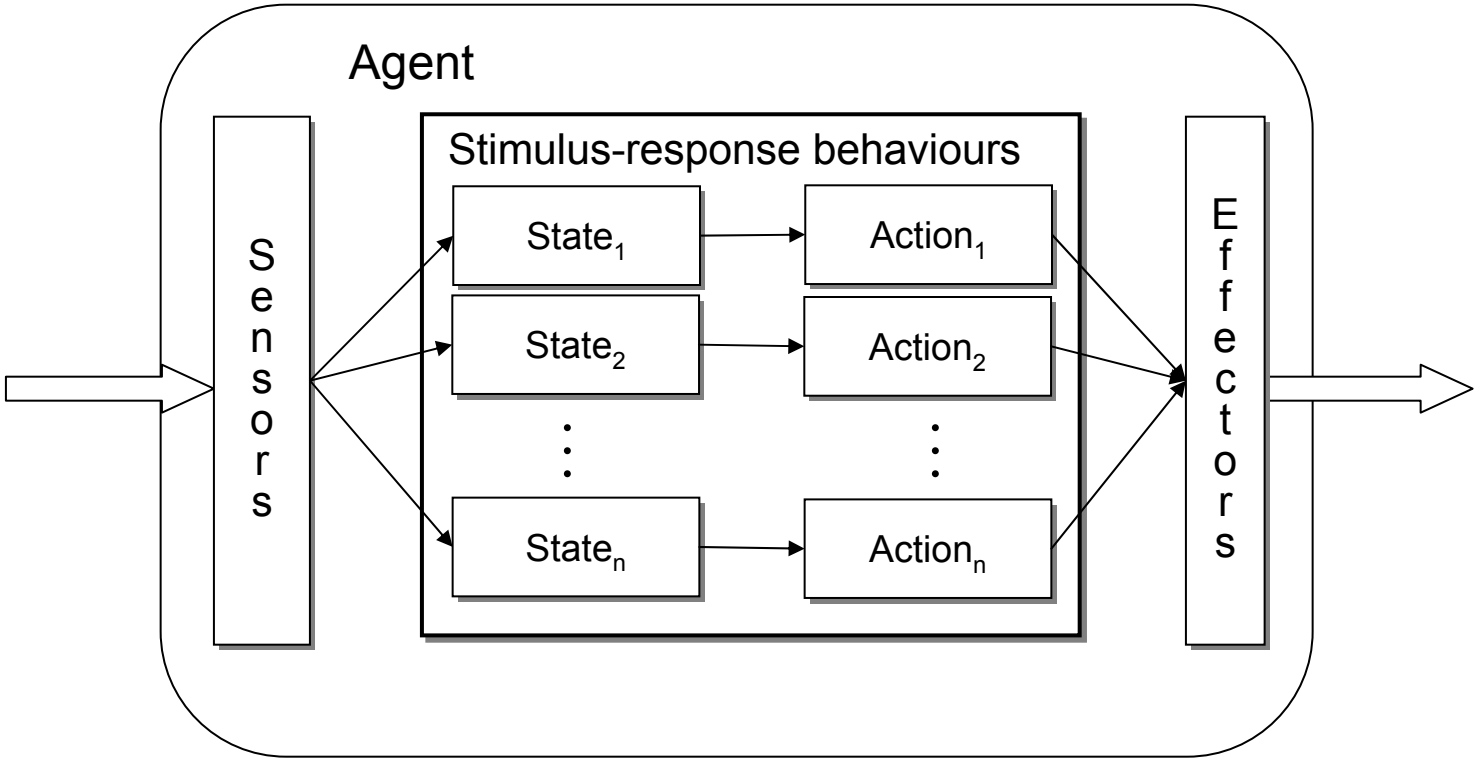


Fully vs partially observable environments

- Observability describes access to information about the world
- In a fully observable environment an agent has complete access to its state and can observe any changes as they occur in it
- Most realistic environments are only partially observable
- Partial observability can be attributed to noise in the agent's sensors or perceptual aliasing
- Partial observability is important in multi-agent systems as it also affects what the agent knows about the other agents
- The more information an agent has about its world, the easier it is to choose and perform the best action

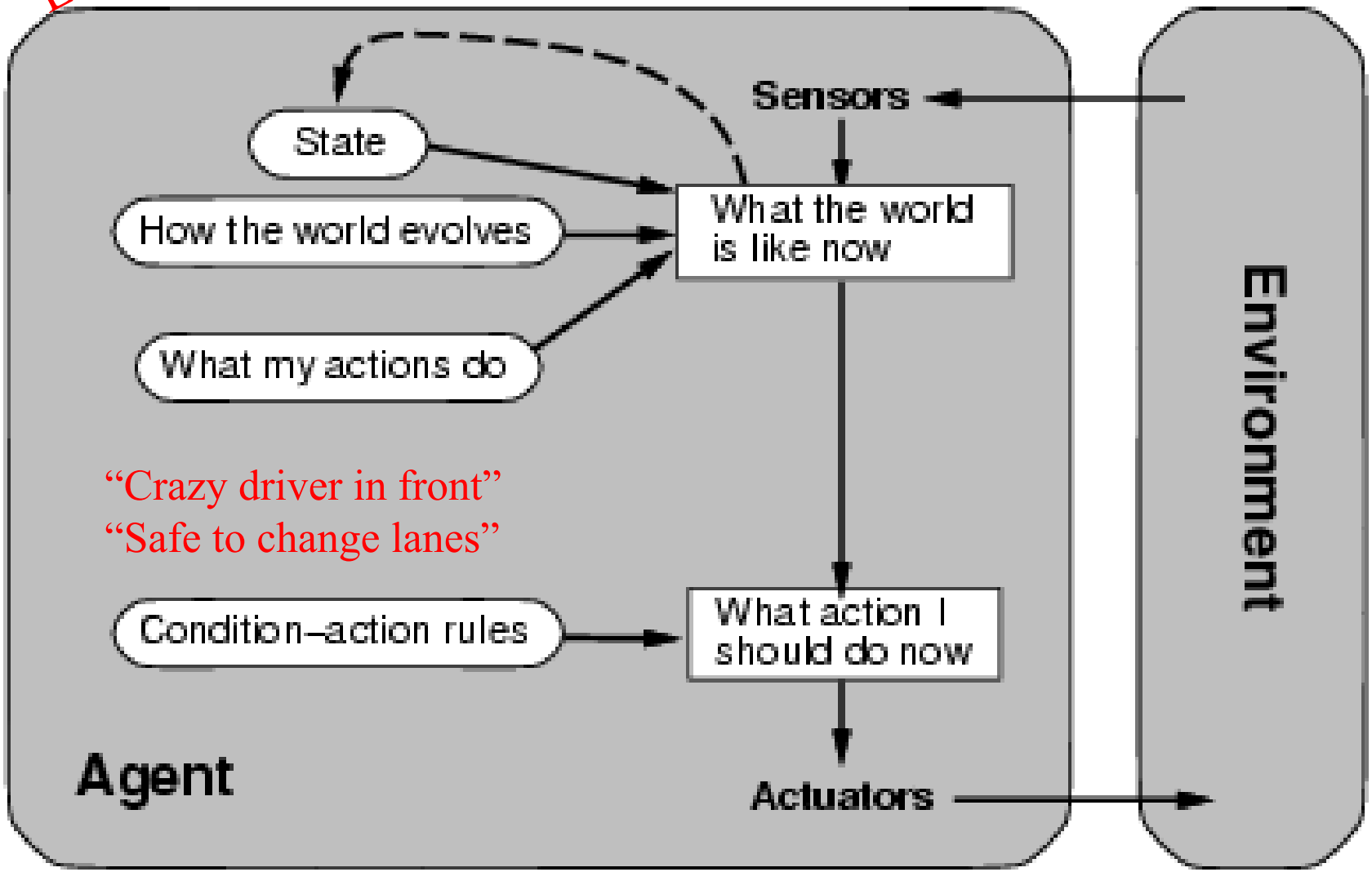
Agent Architectures

Reactive Agent



Module: Logical Agents

Model-based reflex agents



Model-based reflex agents

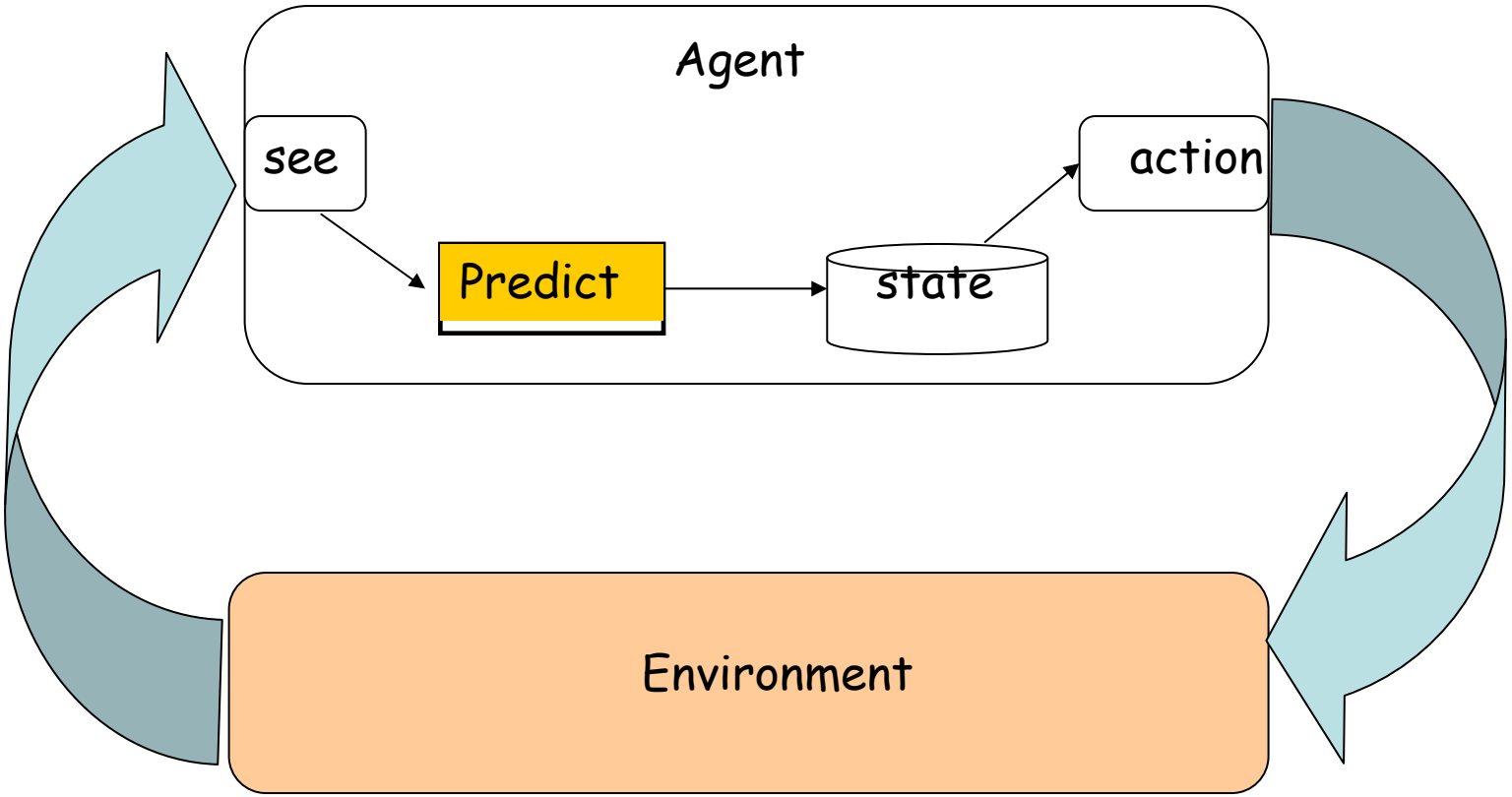
- Key difference (wrt simple reflex agents):
  - Agents have **internal state**, which is used to keep track of past states of the world.

Example: Rodney Brooks's Subsumption Architecture --- behavior based robots.

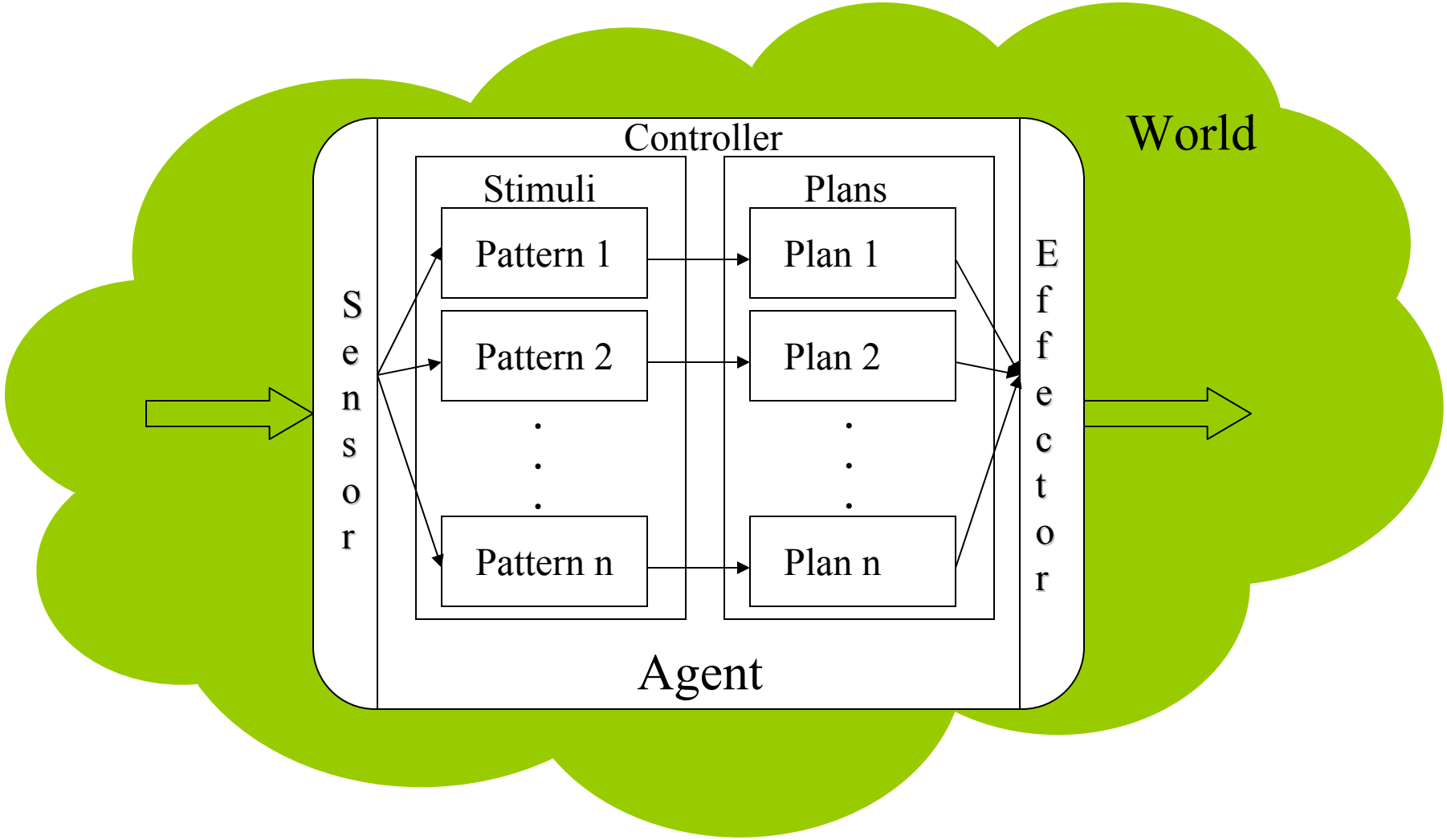
  - Agents have the ability **to represent change in the World**.

Model-based reflex agents

With internal states

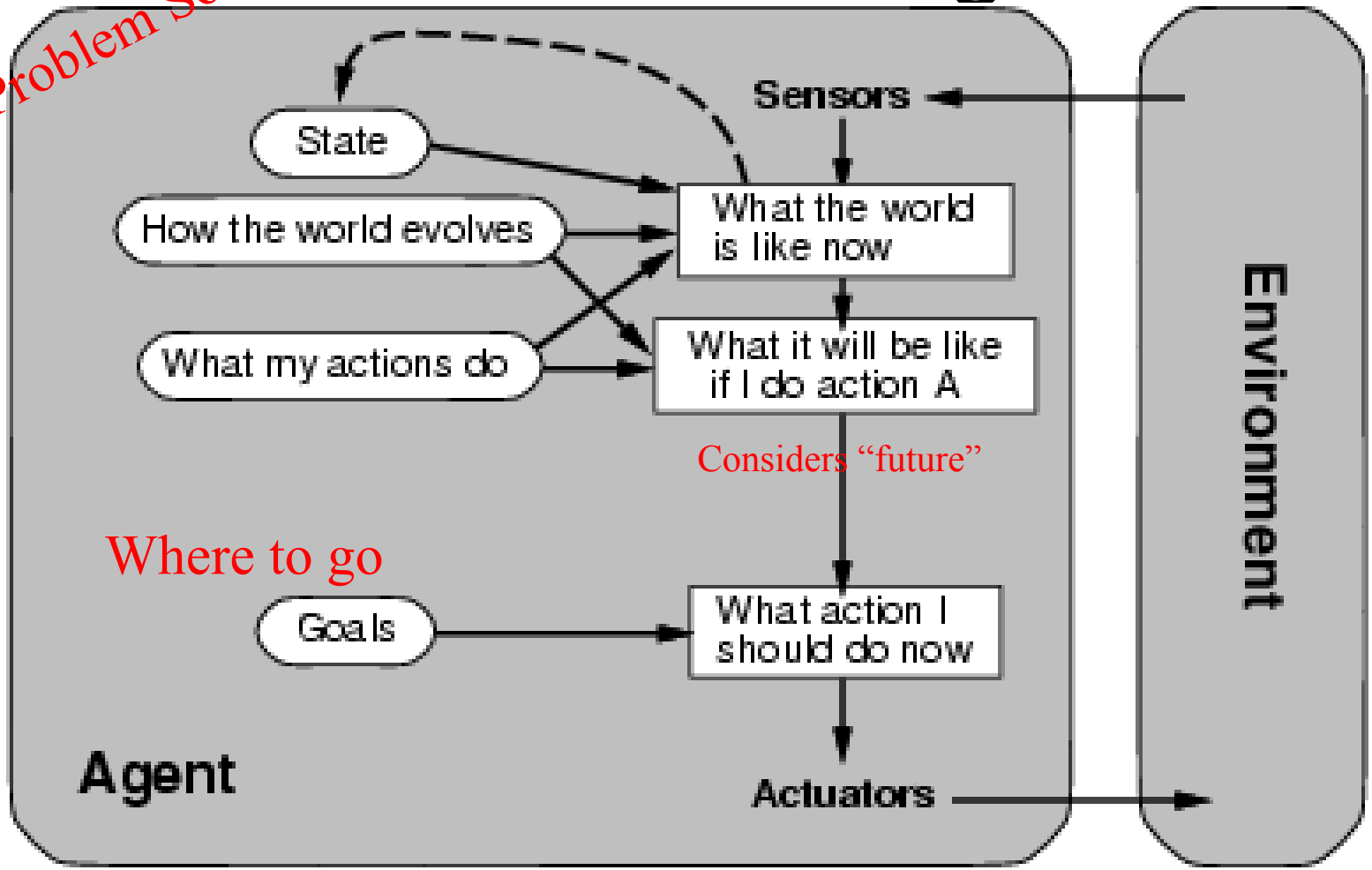


Reactive Agents



Module: Problem Solving Agents

Goal-based agents



Agent keeps track of the world state as well as set of goals it's trying to achieve: chooses Action that will (eventually) lead to the goals.

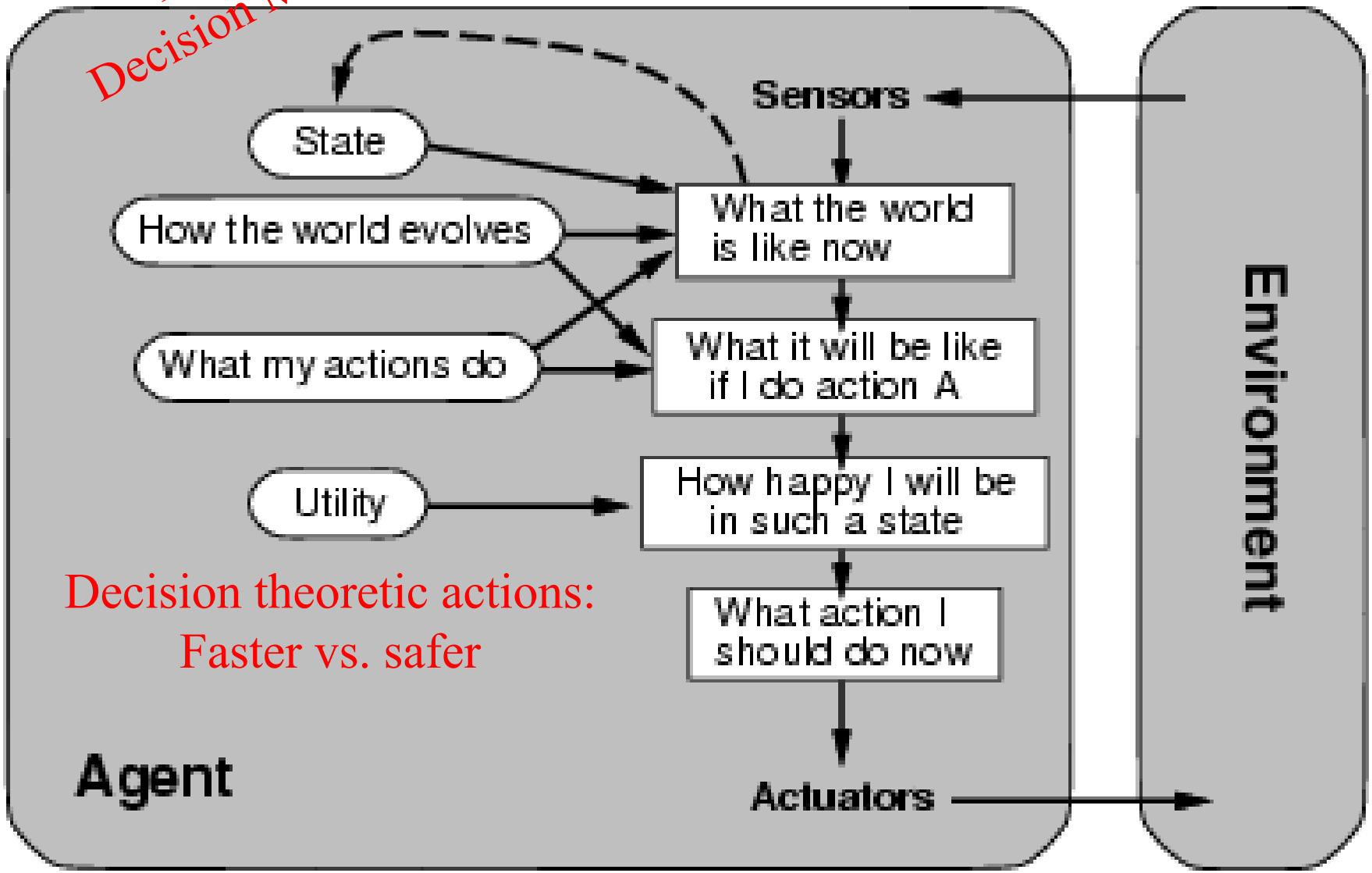
More flexible than reflex agents → may involve search and planning!

Goal-based agents

- Key difference wrt Model-Based Agents:
- In addition to state information, have **goal information** that describes 'desirable situations'.
- Agents of this kind take future events into consideration.
- Choose actions so as to achieve a (given or computed) goal.
- ***→ problem solving and search!***

Utility-based agents

Module: Decision Making



Decision theoretic actions:
Faster vs. safer

(4) Utility-based agents

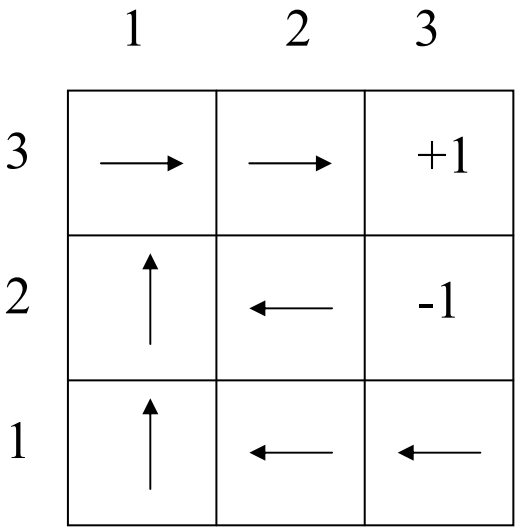
- When there are multiple possible alternatives, how to decide which one is best?
- Goals are qualitative → A goal specifies a crude distinction between a happy and unhappy state, but often need a more general performance measure that describes "degree of happiness."
- Utility function **U: State → Reals** indicating a measure of success or happiness when at a given state.
- Important for making tradeoffs →Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain).

Use decision theoretic models: e.g., faster vs. safer.

The utilities have been calculated using the value iteration method

	1	2	3
3	0.87	0.93	+1
2	0.82	0.78	-1
1	0.76	0.72	0.49

(a)

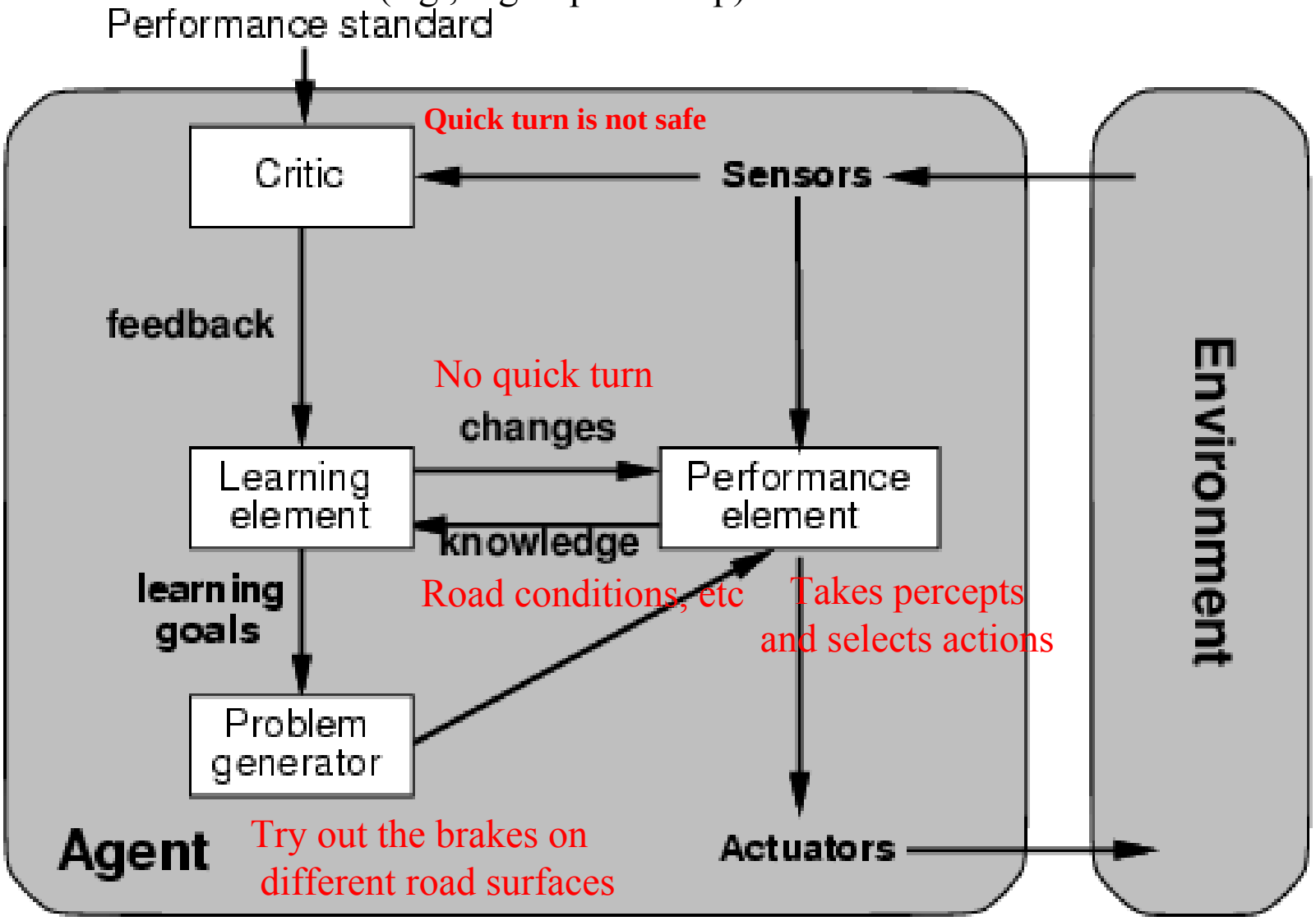


(b)

Module: Learning

More complicated when
agent needs to learn utility information
→Reinforcement learning
(reward or penalty)
(e.g., high tip or no tip)

Learning agents

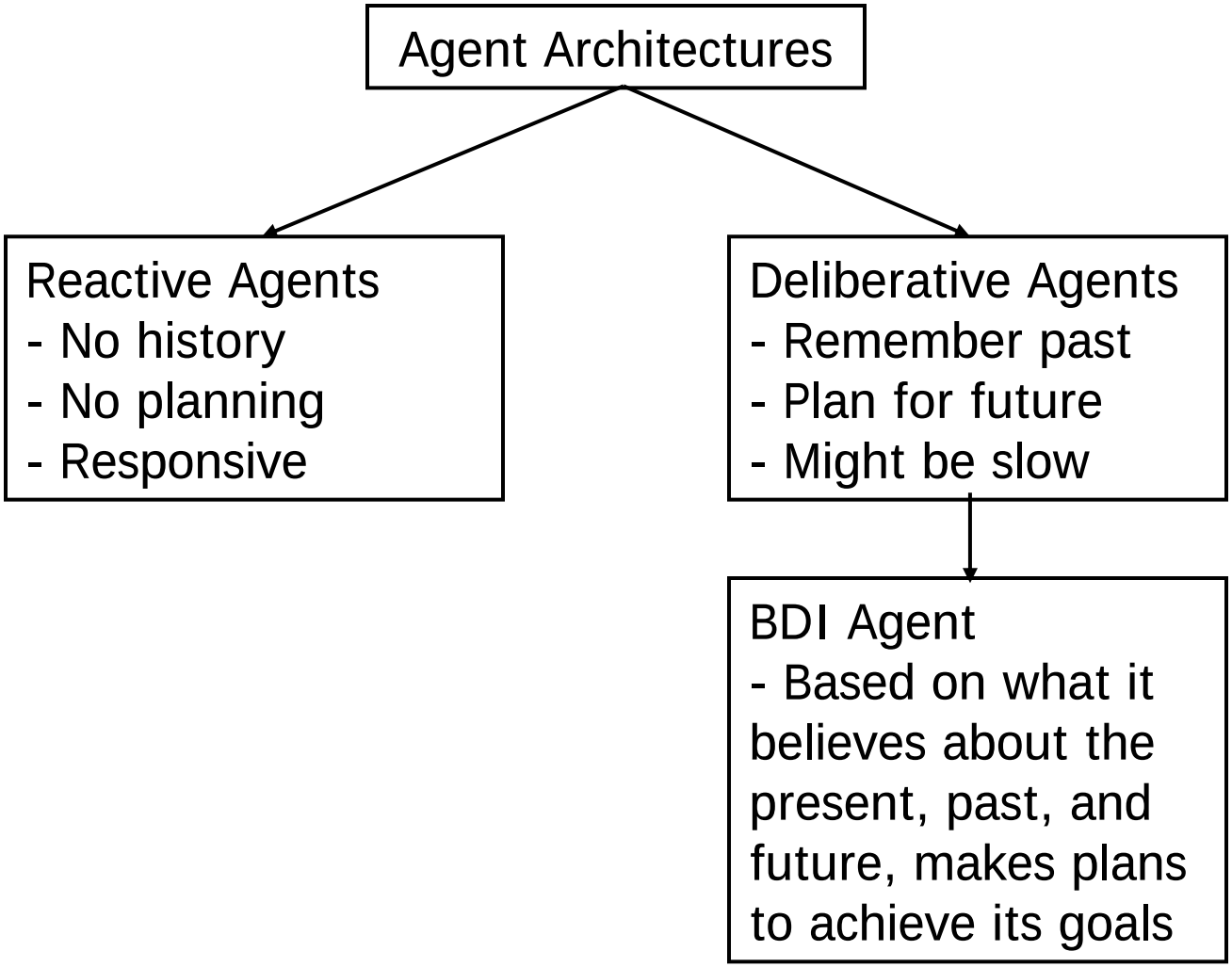


Summary

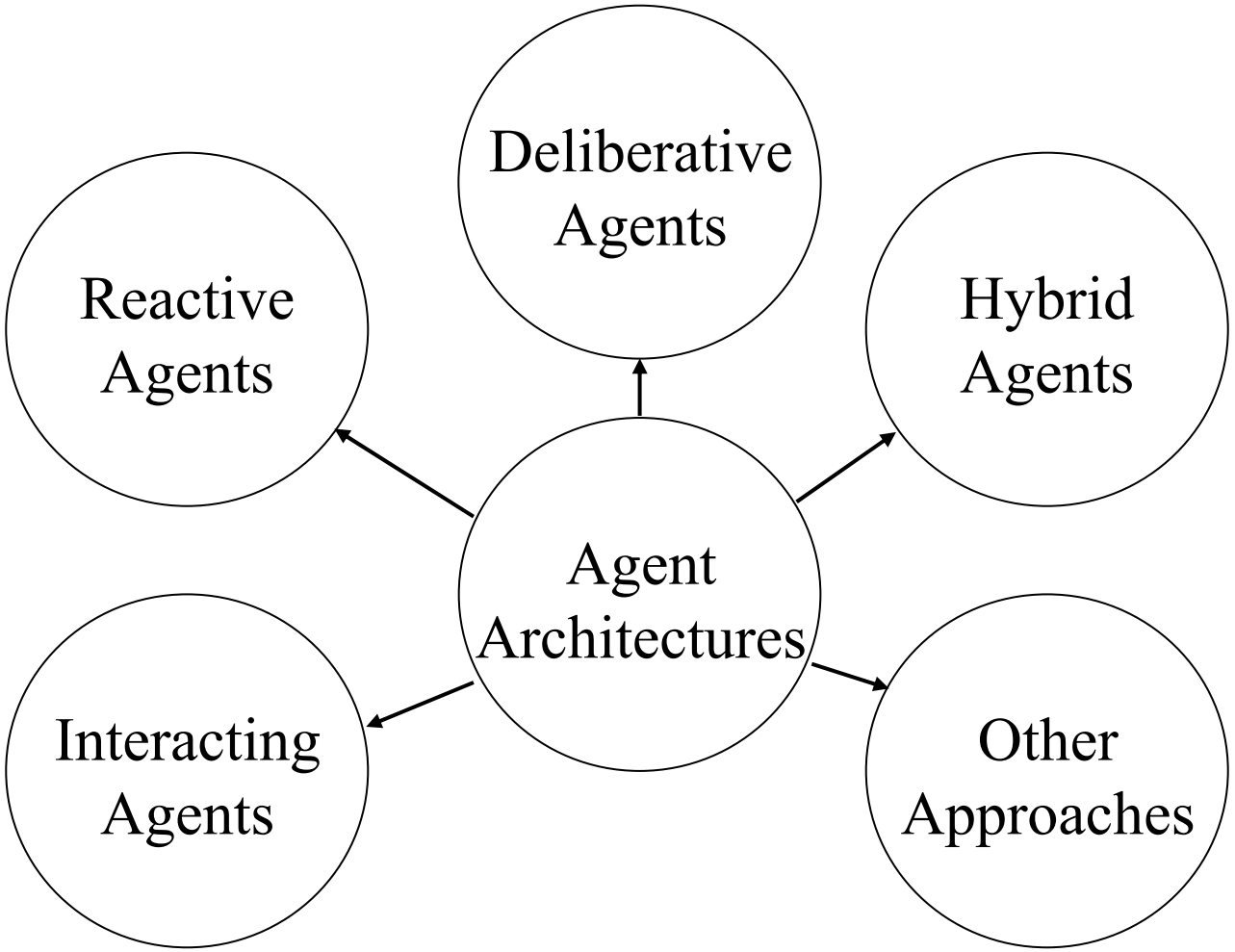
- An **agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- A **rational agent** always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An **autonomous agent** uses its own experience rather than built-in knowledge of the environment by the designer.
- An **agent program** maps from percept to action and updates its internal state.
  - **Reflex agents** respond immediately to percepts.
  - **Goal-based agents** act in order to achieve their goal(s).
  - **Utility-based agents** maximize their own utility function.
  - **Learning agents** improve their performance through learning.
- **Representing knowledge** is important for successful agent design.
- The most challenging environments are partially observable, stochastic, sequential, dynamic, and continuous, and contain multiple intelligent agents.

- **Reading: Chapter 2 R&N**

Architectural Types



Agent architectures

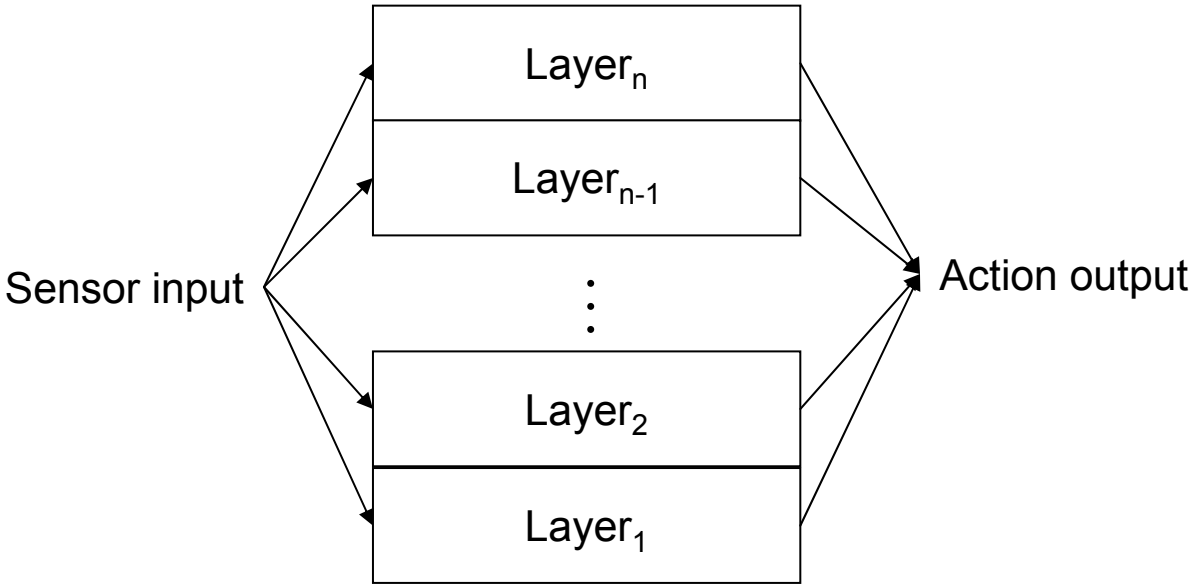


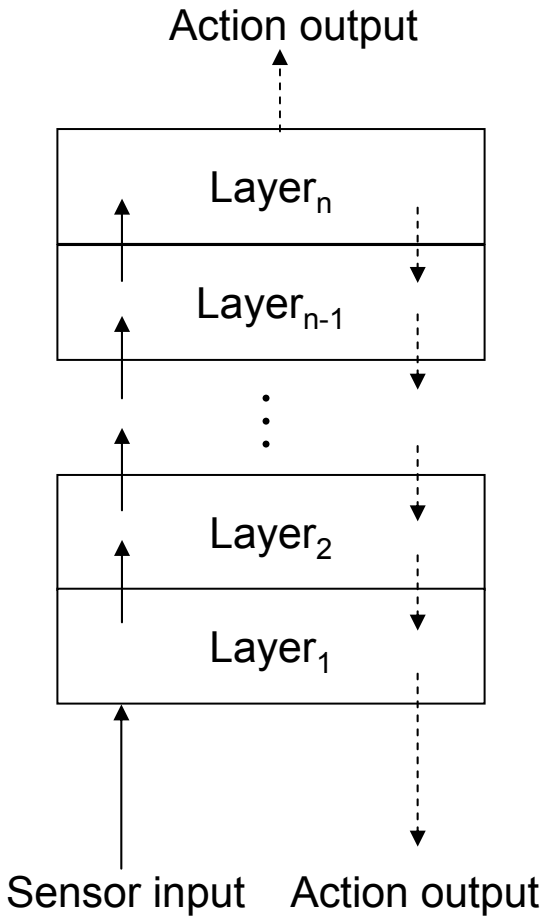
Hybrid architectures

- Combine **reactive** and **deliberative** components and form a hierarchy of interacting layers
- Each layer reasons at a different level of abstraction
- Two types of layering:
  - Horizontal layering
  - Vertical layering

Agent Architectures

Hybrid Agent

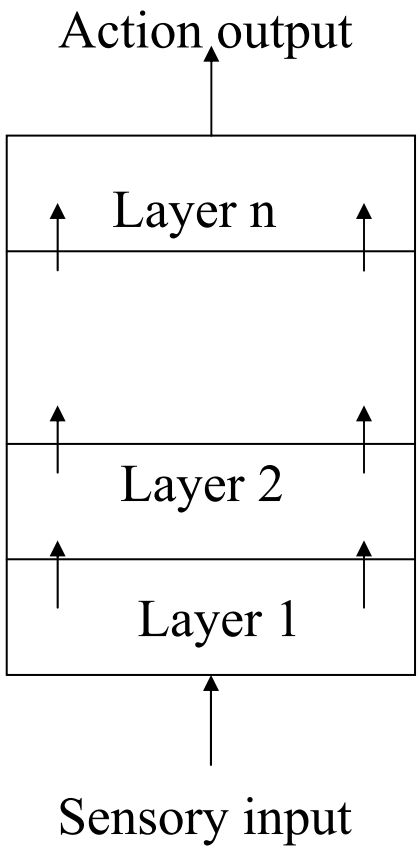




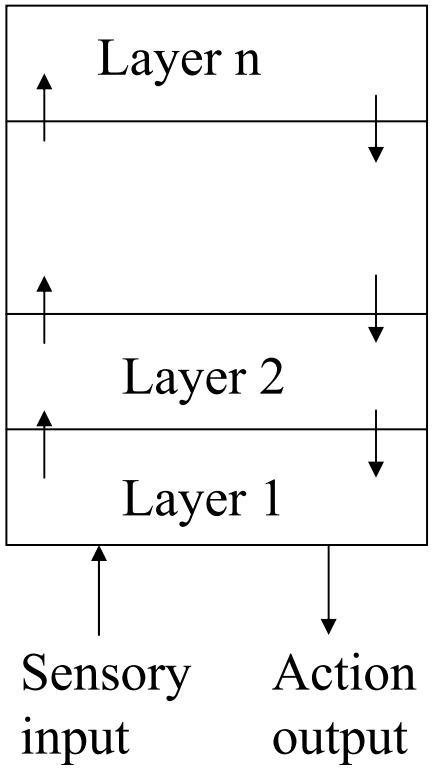
Horizontal layering

- Each layer can act as an independent agent
- For *n* different behaviours *n* layers are implemented
- The layers compete with each other in order to take control of the agent; a mediator function can be introduced

Vertical layering



One-pass



Two-pass

Advantages

- Low complexity. If there are *n* layers there are *n*-1 interfaces between them. If each layer is capable of suggesting *m* possible actions then there are at most $m^2(n-1)$ interactions
- No central control, no bottleneck in the agent's decision making

Problems

- Less flexible
- Not fault tolerant

Agent Architectures

Reactive Agent

- Each behaviour continually maps perceptual input to action output
- Reactive behaviour:

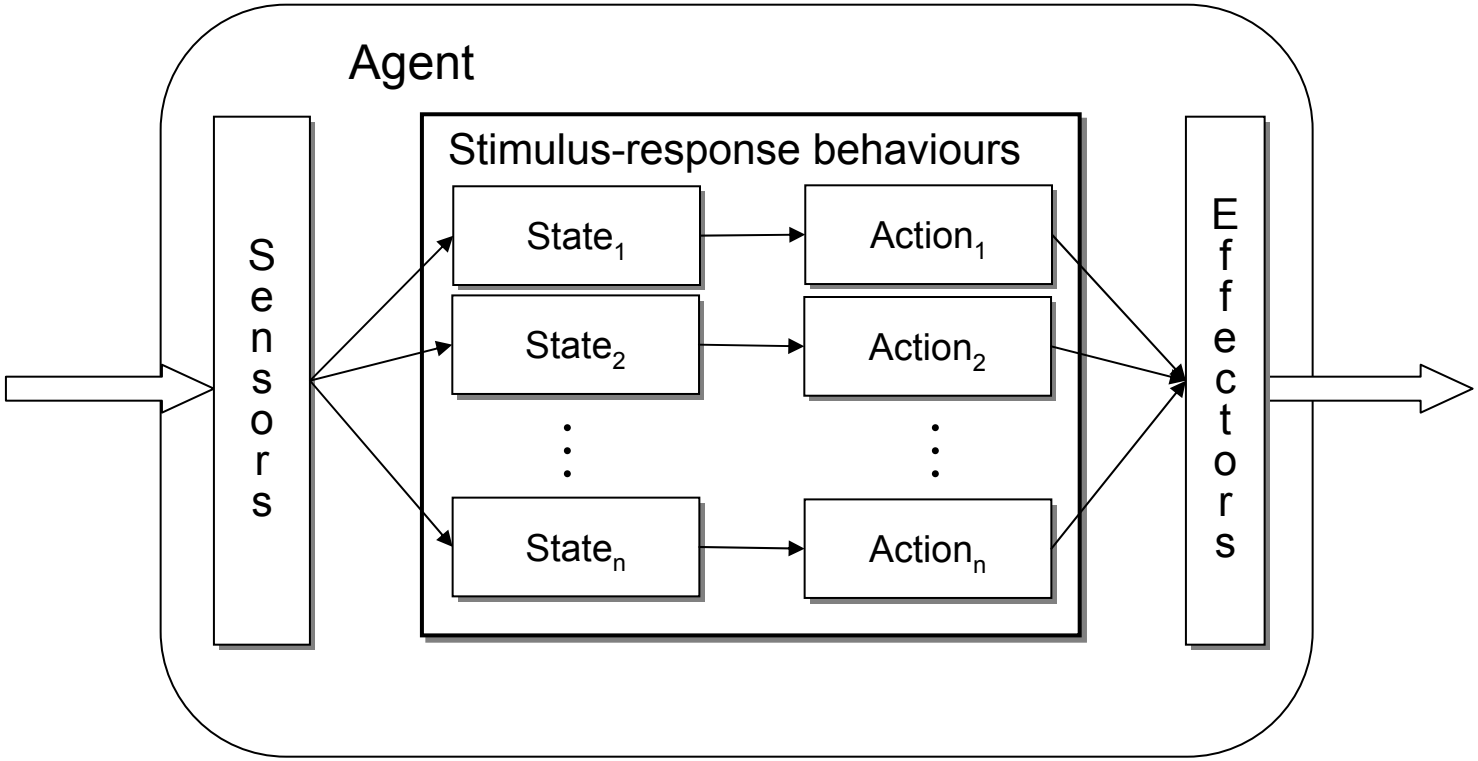
  action: S -> A

  - where S denotes the states of the environment, and A the primitive actions the agent is capable of perform.
- Example:

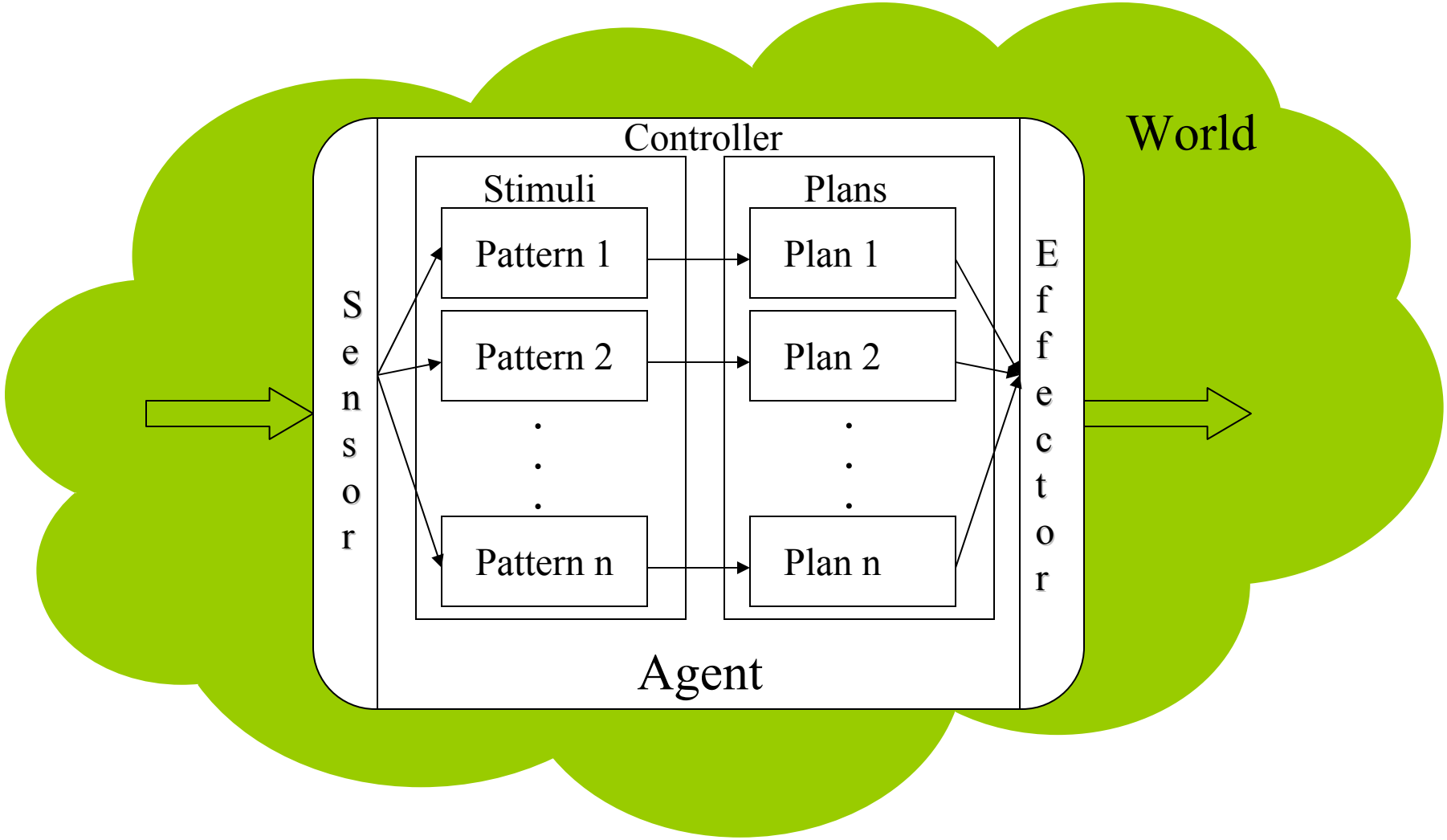
$$\text{action}(s) = \begin{cases} \textit{Heater on}, \text{ if temperature too low} \\ \textit{Heater off}, \text{ otherwise} \end{cases}$$

Agent Architectures

Reactive Agent



Reactive Agents



Agent Architectures

Reactive Agent

- Problems
  - a great deal of local information needed
  - learning?
  - Typically “handcrafted”
    - Development takes a lot of time
    - Impossible to build large systems?
    - Can be used only for its original purpose

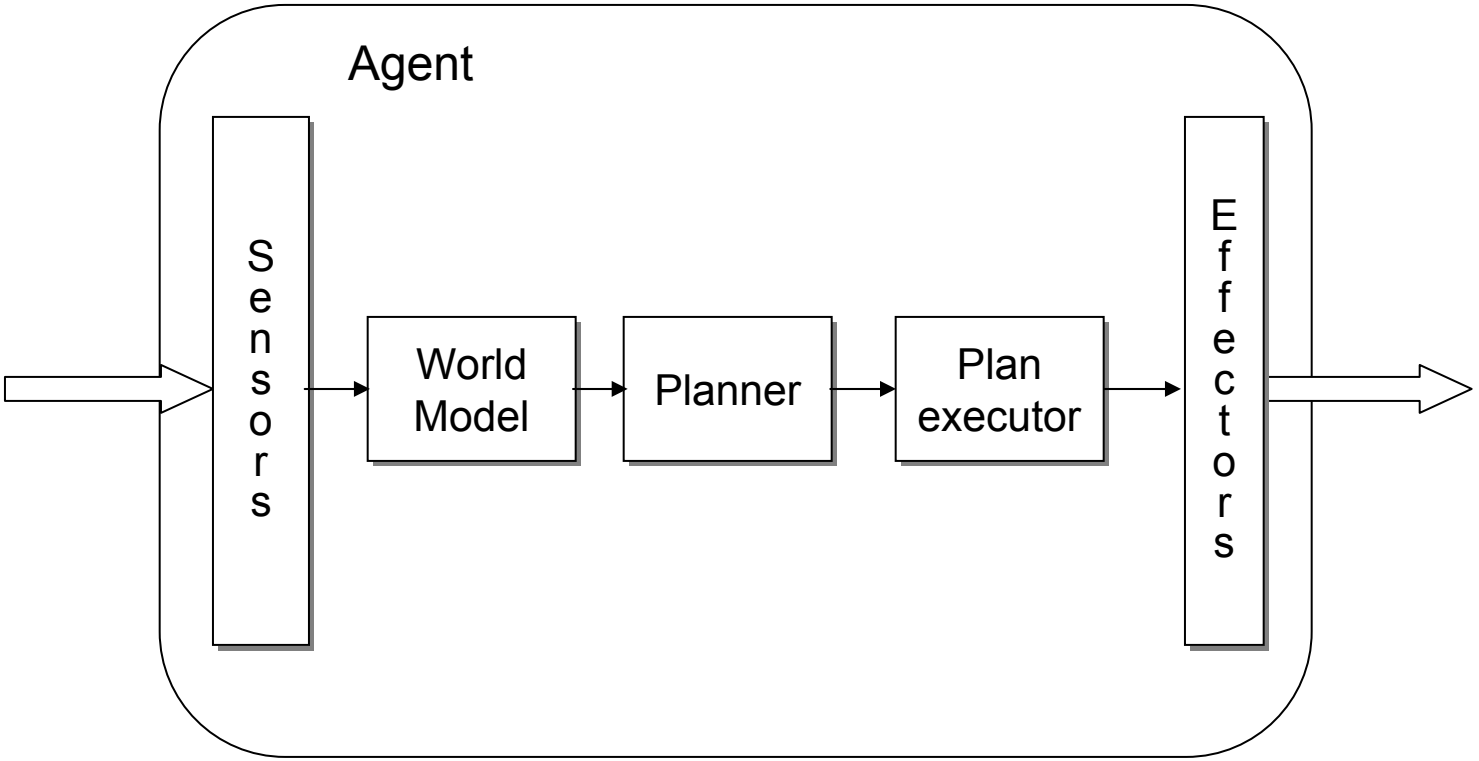
- Examples
  - Brooks: subsumption architecture
    - ref: Http://ai.eecs.umich.edu/cogarch3/Brooks/Brooks.html

Agent Architectures

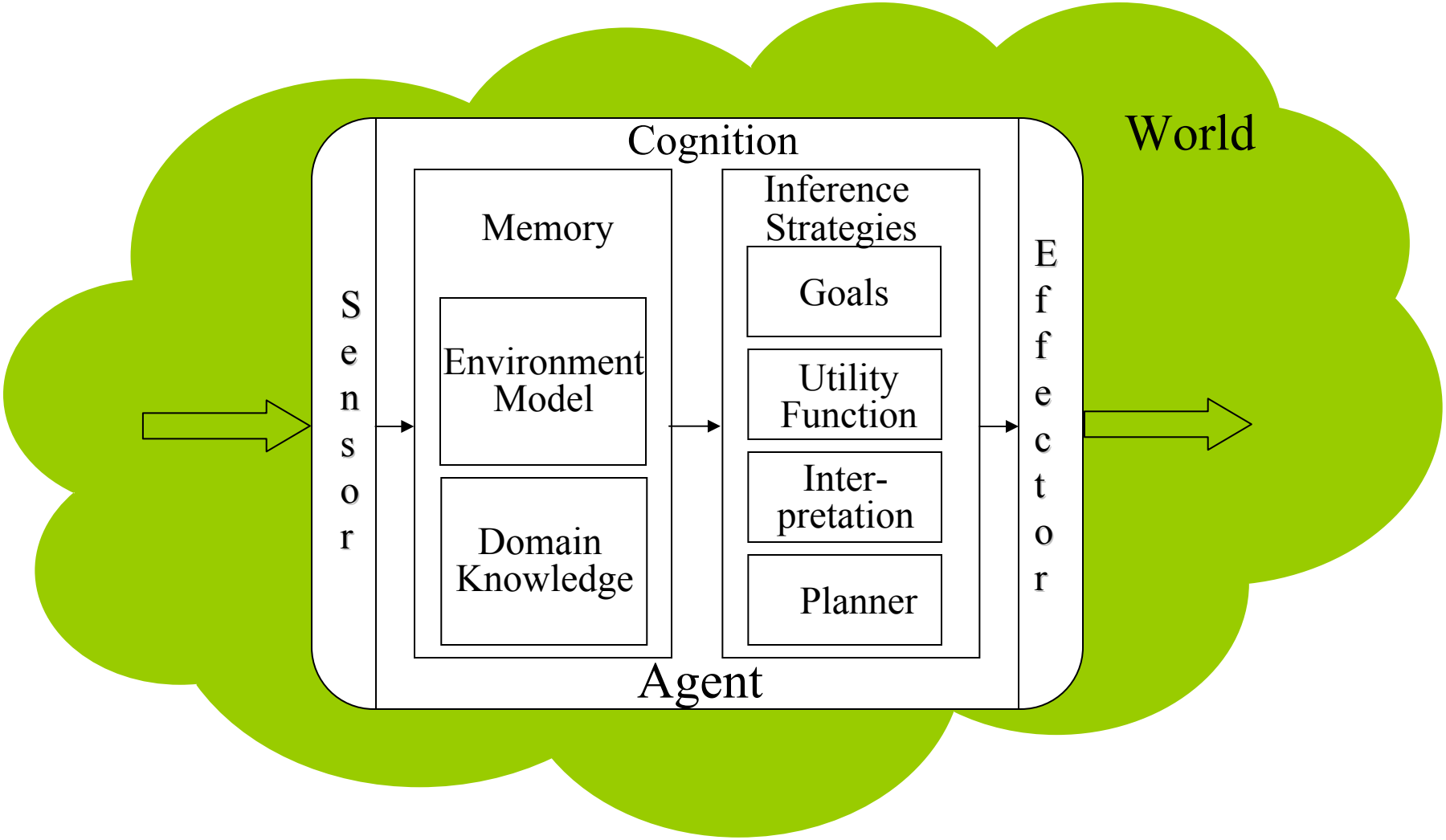
- Deliberative Agent
  - Explicit symbolic model of the world in which decisions are made via logical reasoning, based on pattern matching and symbolic manipulation
  - sense-plan-act problem-solving paradigm of classical AI planning systems

Agent Architectures

Deliberative Agent



Deliberative Agents



Agent Architectures

Deliberative Agent

- Examples of deliberative architectures
  - BDI - Brahms

Agent Architectures

Deliberative Agent

- Performance problems
  - *transduction* problem
    - time consuming to translate all of the needed information into the symbolic representation, especially if the environment is changing rapidly.
  - *representation* problem
    - how the world-model is represented in symbolically and how to get agents to reason with the information in time for the results to be useful.
- Late results may be useless
- Does not scale to real-world scenarios

Agent Architectures

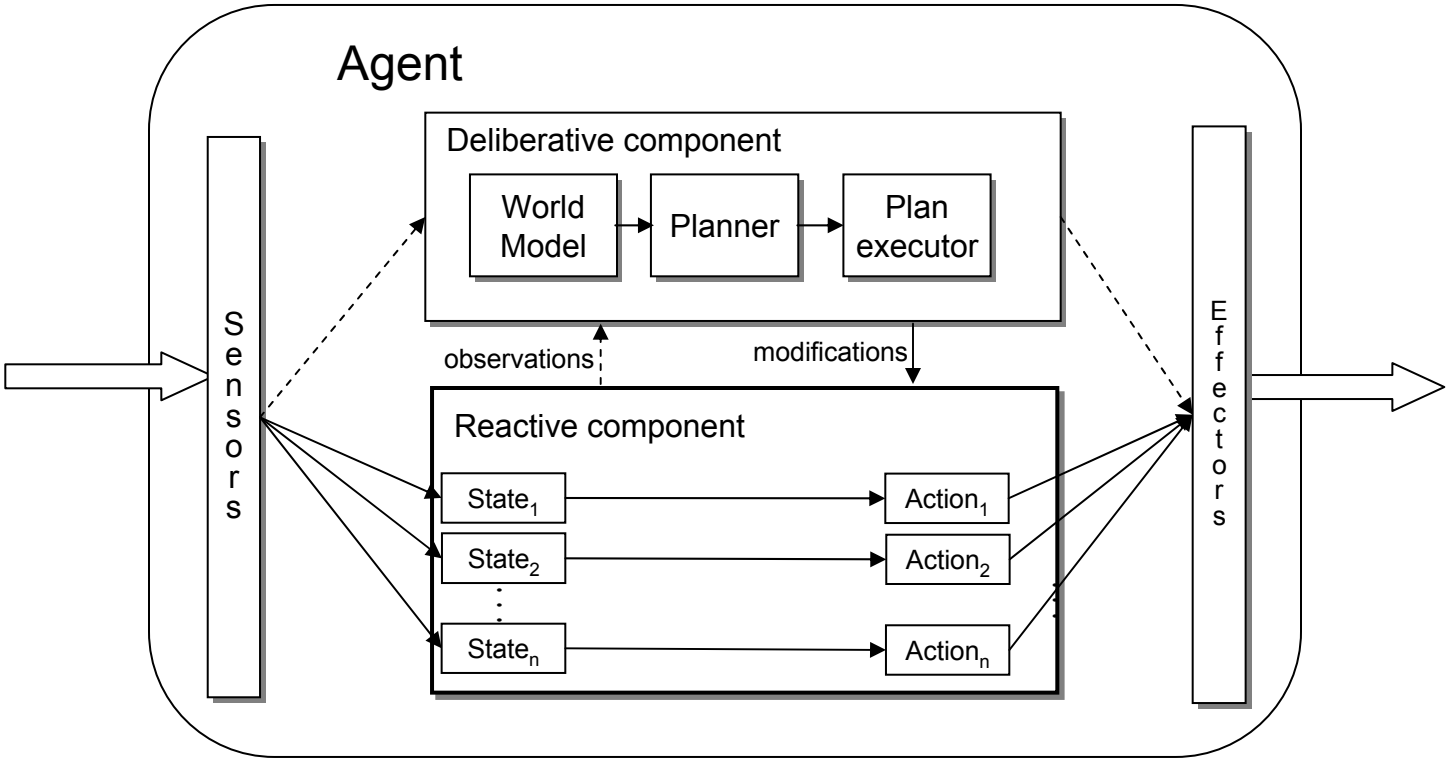
Hybrid Agent

- Combination of deliberative and reactive behaviour
  - An agent consists of several subsystems
    - Subsystems that develop plans and make decisions using symbolic reasoning (deliberative component)
    - Reactive subsystems that are able to react quickly to events without complex reasoning (reactive component)

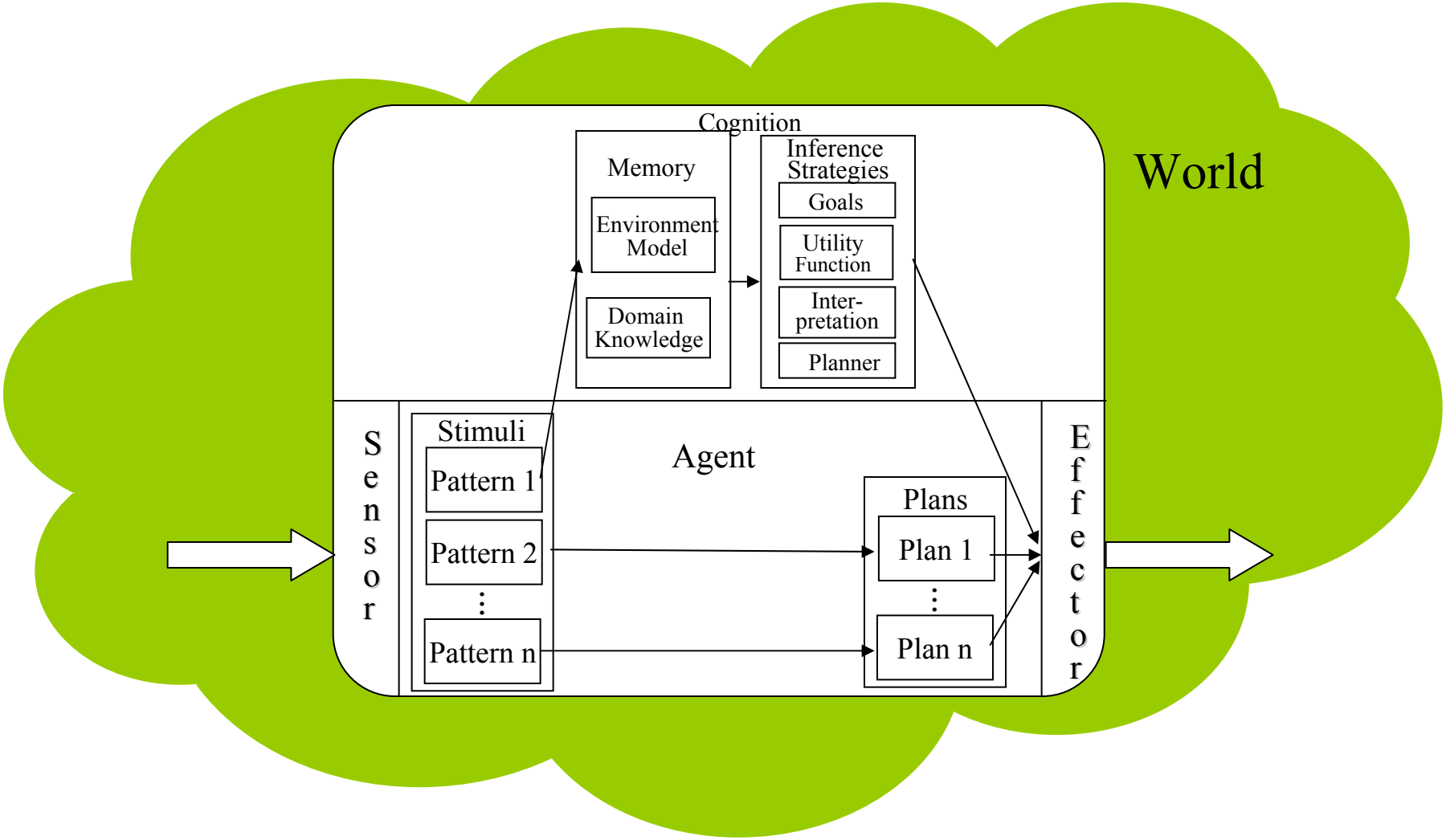
- Layered architectures

Agent Architectures

Hybrid Agent

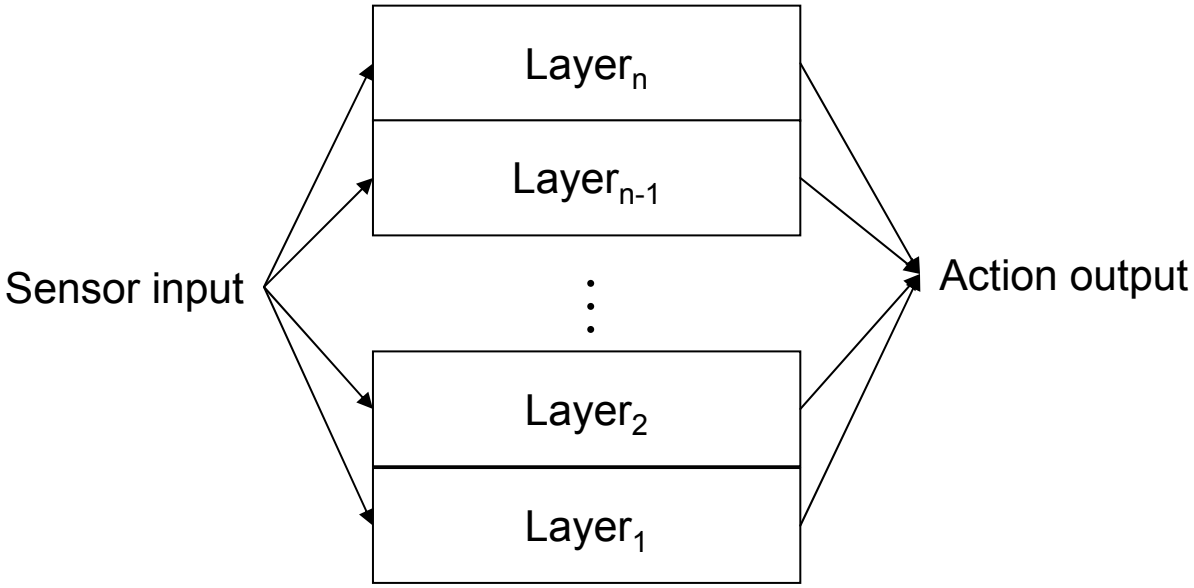


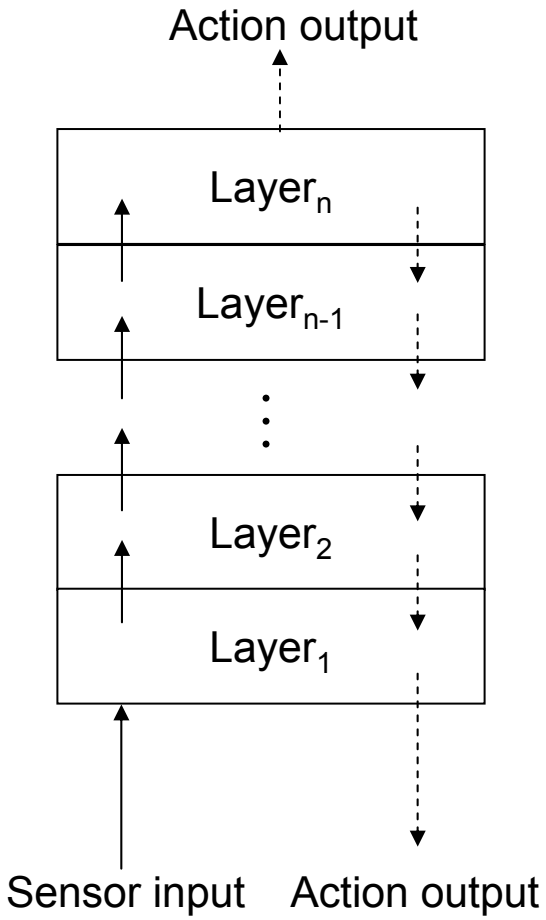
Hybrid Agents



Agent Architectures

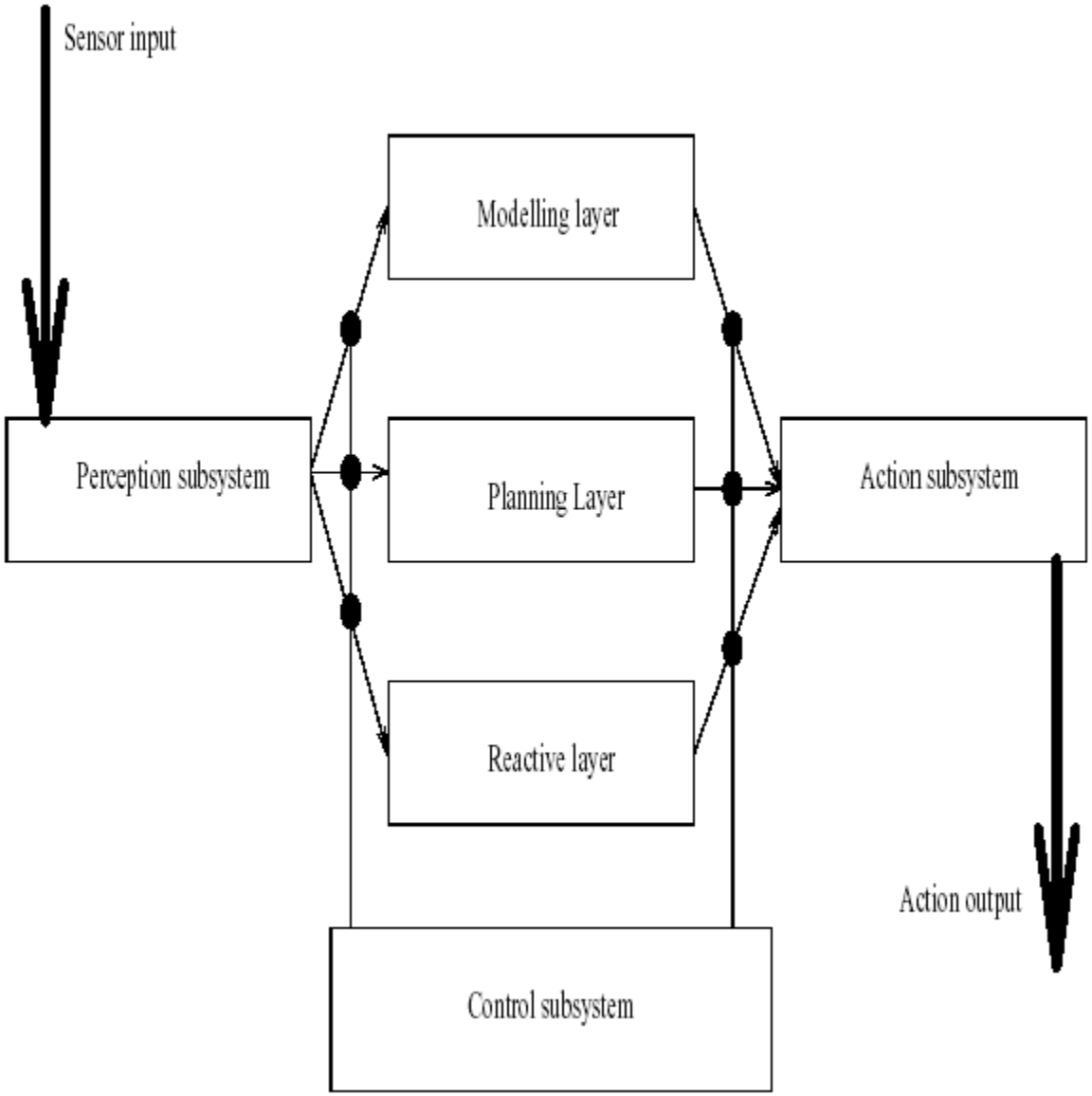
Hybrid Agent





Horizontal layering

- Each layer can act as an independent agent
- For *n* different behaviours *n* layers are implemented
- The layers compete with each other in order to take control of the agent; a mediator function can be introduced


Sensor input
Modelling layer
Perception subsystem
Planning Layer
Action subsystem
Reactive layer
Action output
Control subsystem

Ferguson – TOURING MACHINES

- The *reactive layer* is implemented as a set of situation-action rules, *a la* subsumption architecture

  Example:

```
rule-1: kerb-avoidance
   if
        is-in-front(Kerb, Observer) and
        speed(Observer) > 0 and
        separation(Kerb, Observer) <
KerbThreshHold
   then
        change-
orientation(KerbAvoidanceAngle)
```

- The *planning layer* constructs plans and selects actions to execute in order to achieve the agent's goals

- The *modeling layer* contains symbolic representations of the 'cognitive state' of other entities in the agent's environment
- The three layers communicate with each other and are embedded in a control framework, which use *control rules*

Example:

```
censor-rule-1:
    if
          entity(obstacle-6) in perception-buffer
    then
          remove-sensory-record(layer-R, entity(obstacle-
6))
```

Reactive layer

- Acts as a reactive agent and responds to changes as they occur
- Implemented through situation-action rules
- There is no model of the environment in this layer

Planning layer

- Achieves the agent's pro-active behaviour via plans based on a library of plan skeletons or schemas

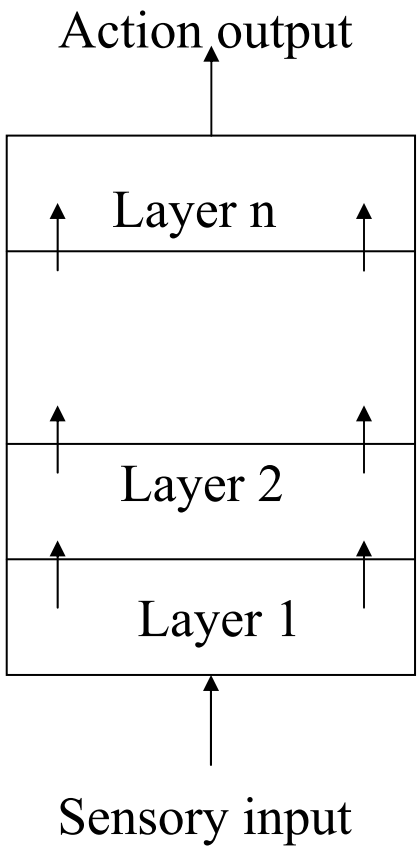
Modelling layer

- Endows the agent with reflective and predictive capabilities
- Entities are modelled as having a configuration, beliefs, desires and intentions
- Generates goals to resolve conflicts which are then propagated to the planning layer

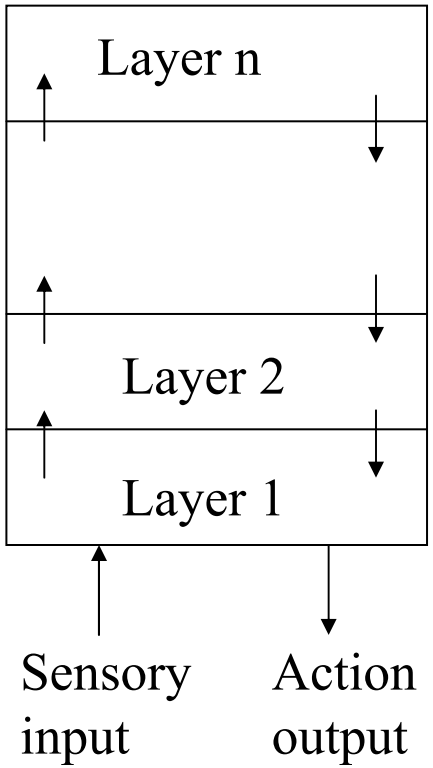
Control subsystem

- Decides which of the layers has control over the agent
- It is implemented via control rules which can either suppress sensor information between the control rules and the control layers or else censor action outputs from the control layers

Vertical layering



One-pass



Two-pass

Advantages

- Low complexity. If there are *n* layers there are *n*-1 interfaces between them. If each layer is capable of suggesting *m* possible actions then there are at most $m^2(n-1)$ interactions
- No central control, no bottleneck in the agent's decision making

Problems

- Less flexible
- Not fault tolerant

Agent Architectures

Hybrid Agent

example - InteRRaP