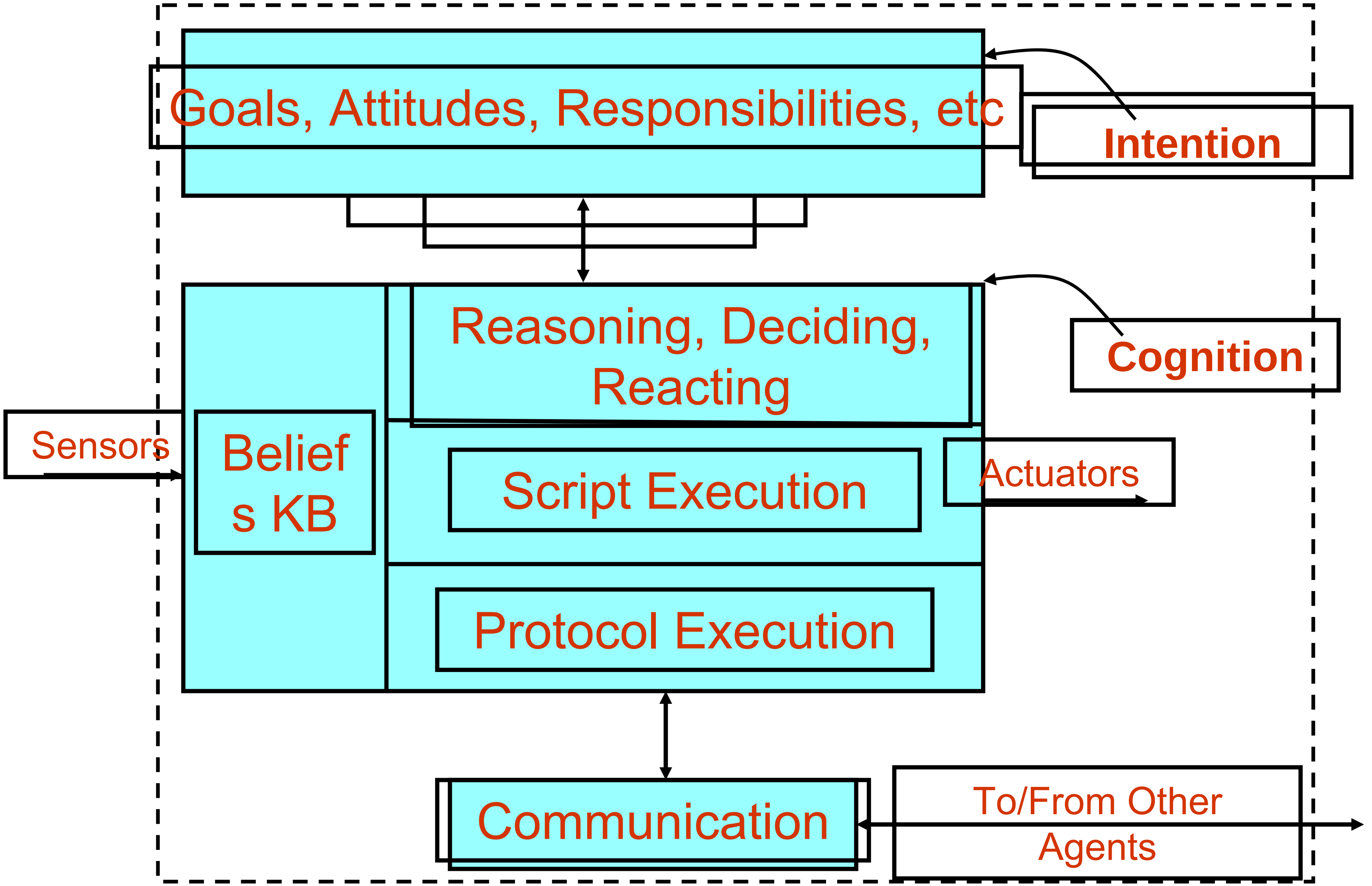
ATTENTION!

- The Lab runs EVERY TWO WEEKS!
- First Lab **tomorrow**, Jan. 14
- Second Lab Jan 28

For tomorrow at the Lab

- Think about what part of the ‘human agent’ architecture would interest you for a possible project that you can implement during the Lab using the agent programming language that we will learn (e.g. you may be interested to develop an ‘expert system’-like module for ‘Reasoning, Deciding, Reacting’, etc…)

Human (or advanced robot) agent



Structure of Agents

Agent = Architecture + Program

Computing device Running Agent Program, with sensors & actuators

Implements Agent Function, performs mapping of percepts to actions

Architectural Types

Agent Architectures

Reactive Agents
- No history
- No planning
- Responsive

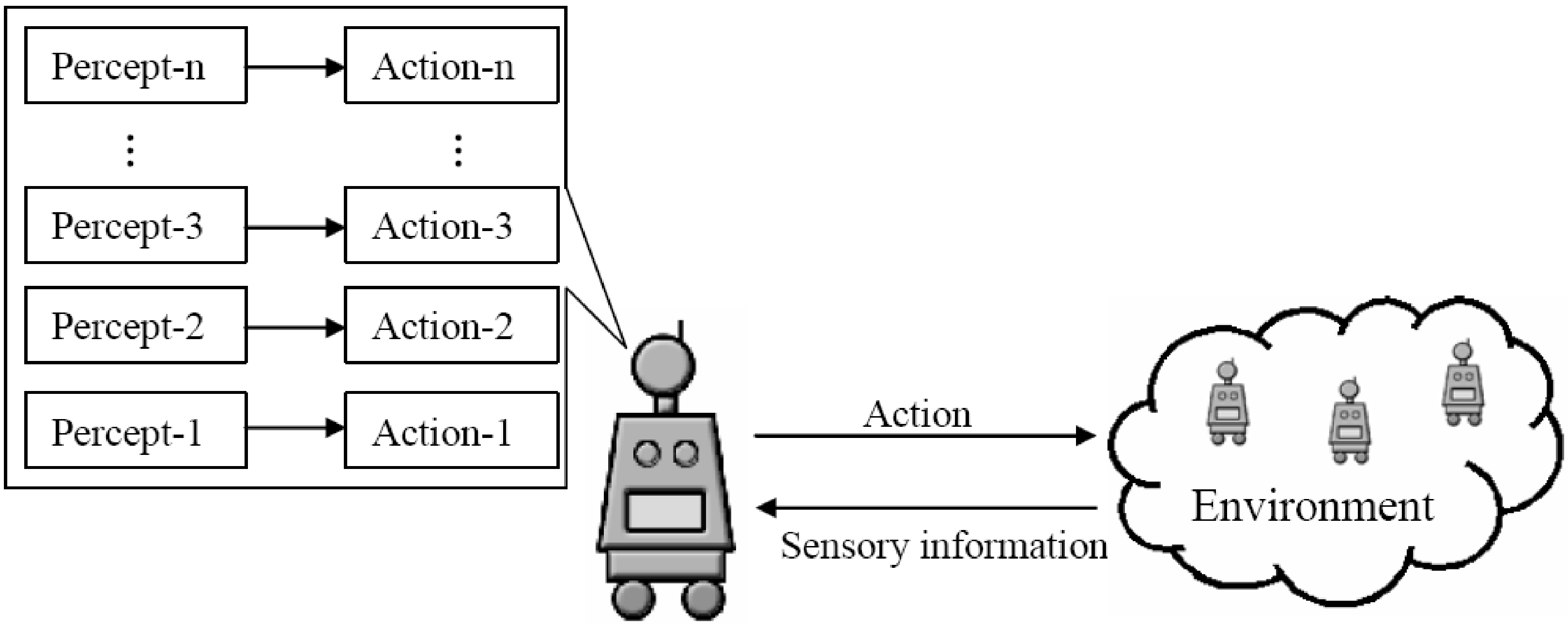
Deliberative Agents
- Remember past
- Plan for future
- Might be slow

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

Agent Architectures

- Originally (1956-1985), pretty much all agents designed within AI were *symbolic reasoning* agents
- Its purest expression proposes that agents use ***explicit logical reasoning*** in order to decide what to do
- Problems with symbolic reasoning led to a reaction against this — the so-called *reactive agents* movement, 1985–present
- From 1990-present, a number of alternatives proposed: *hybrid* architectures, which attempt to combine the best of reasoning and reactive architectures

REACTIVE ARCHITECTURE



function Skeleton-Agent(*percept*) **returns** *action*
  static: *memory*, the agent's memory of the world

  memory ← Update-Memory(*memory*, *percept*)
  action ← Choose-Best-Action(*memory*)
  memory ← Update-Memory(*memory*, *action*)
  return *action*

Generic Structure

LOOK-UP

function Table-Driven-Agent(*percept*) **returns** *action*
  static: *percepts*, a sequence, initially empty
      table, a table indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*
  action ← LookUp(*percepts, table*)
return *action*

function Simple-Reflex-Agent(*percept*) **returns** action
  static: *rules*, a set of condition-action rules

  state ← Interpret-Input(*percept*)
  rule ← Rule-Match(*state, rules*)
  action ← Rule-Action[*rule*]
  return *action*

Rule-Based

Reactive Architecture

- Seeks to produce intelligent behavior without explicit
  - Symbolic representations
  - Abstract reasoning

DELIBERATIVE !

- 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

Reactive architecture ‘movement’ was a reaction against deliberative architectures!

- 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

Brooks' position

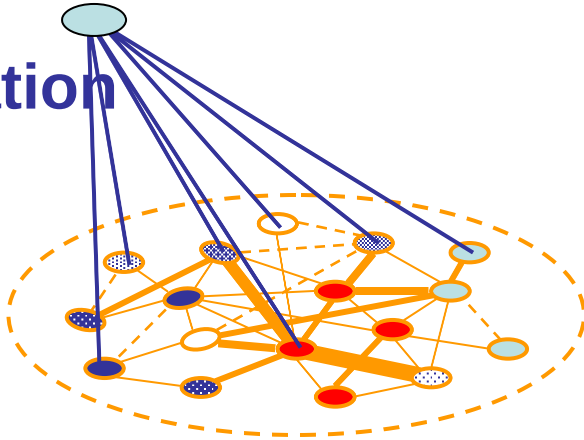
The first to reject the idea of a symbolic model was Brooks

(MIT - http://people.csail.mit.edu/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**

- **Absence of a global controller**

- **Emergence of hierarchical organization**



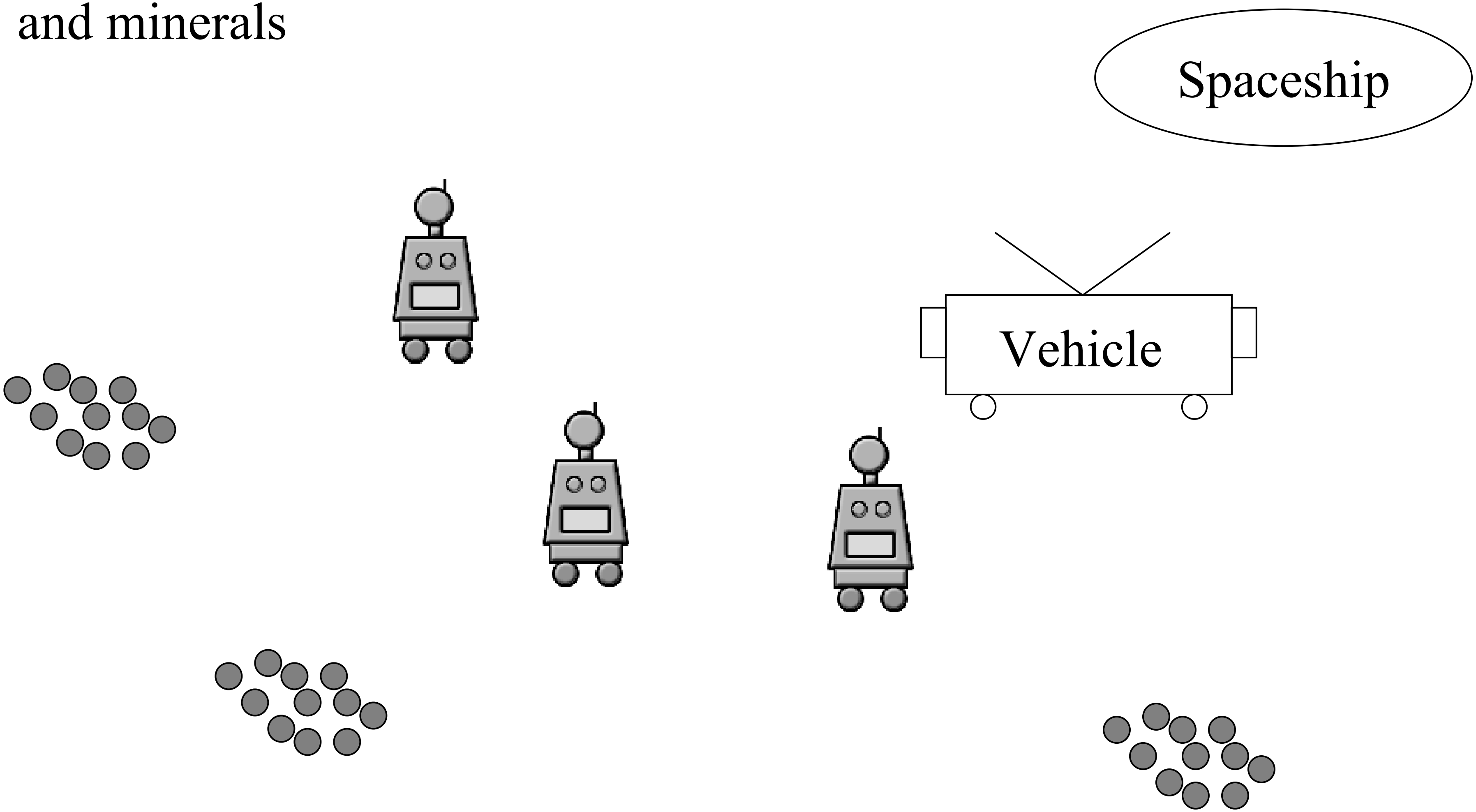
Complex Adaptive Systems

Subsumption Architecture (Brooks)

- 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
- Behaviours are arranged in layers: **the lower the layer, the higher the priority**
- Lower levels are usually associated with tasks/functions that are critical to the agent’s survival (obstacle avoidance etc.)

The Luc Steels scenario

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

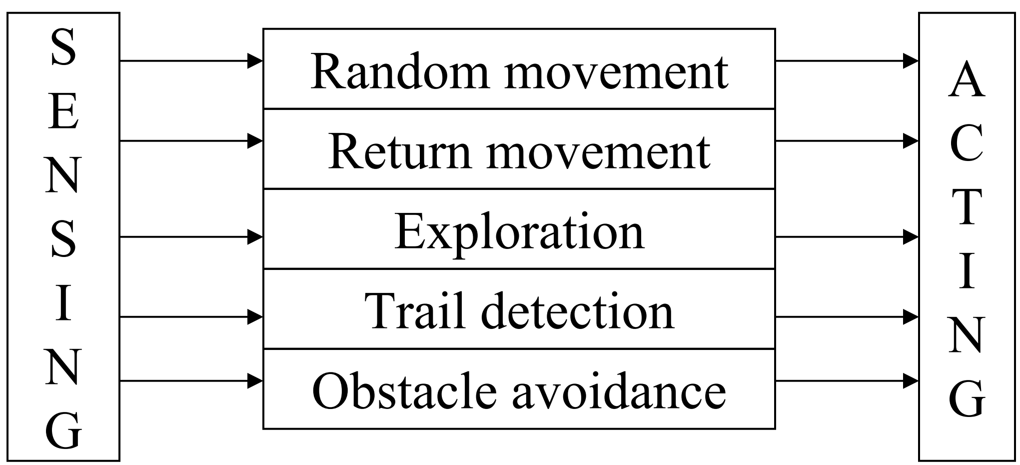


- Each behaviour may encompass more than one situation-action rules, for example:

  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



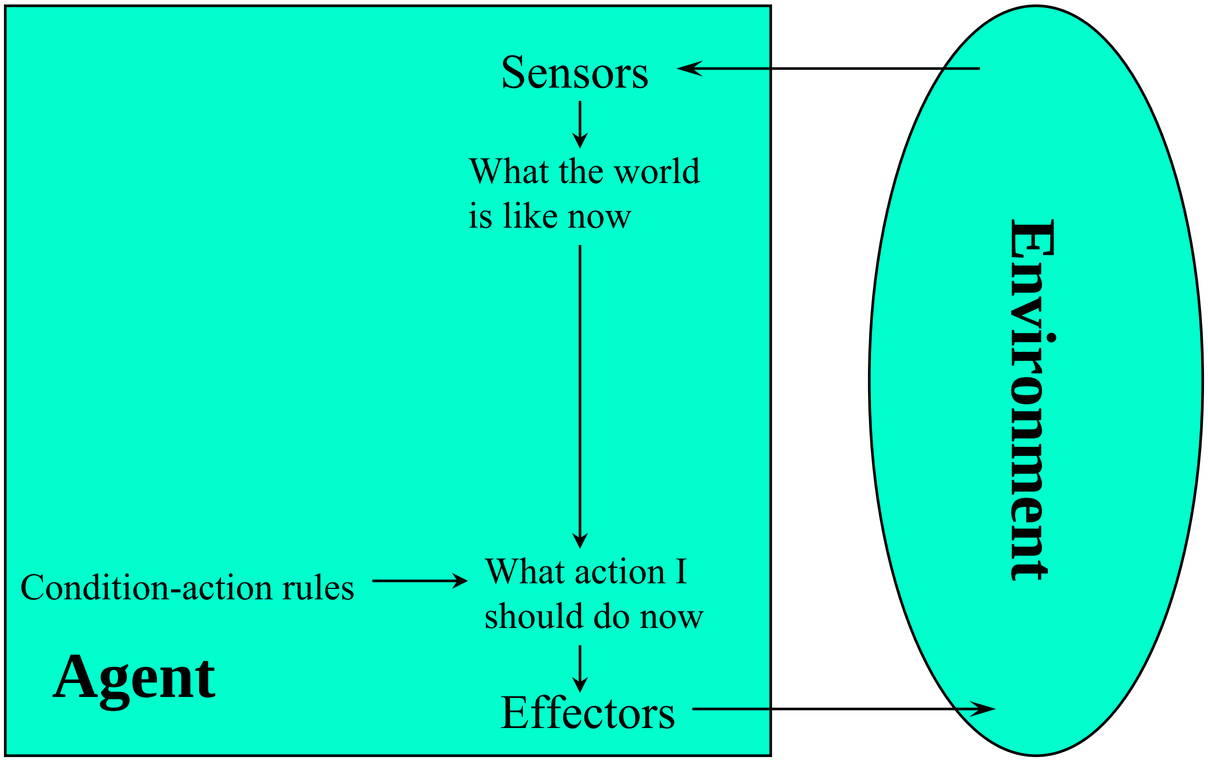
Advantages of the reactive approach

- 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* - not yet fully understood and it is even more difficult to engineer (e.g. see my paper on ‘Emergent Engineering’ – 1st under ‘Publications’)
- Hence, difficult to build task-specific agents

A Simple Reactive Agent



Perception

- The *see* function is the agent's ability to observe its environment, whereas the *action* function represents the agent's decision making process
- *Output* of the *see* function is a *percept*:

$$see : E \rightarrow Per$$

which maps environment states to percepts, and *action* is now a function

$$action : Per^* \rightarrow A$$

which maps sequences of percepts to actions

Simple reflex agents

Act only on the basis of the current percept.

The agent function is based on the

condition-action rule: condition $\Rightarrow$ action

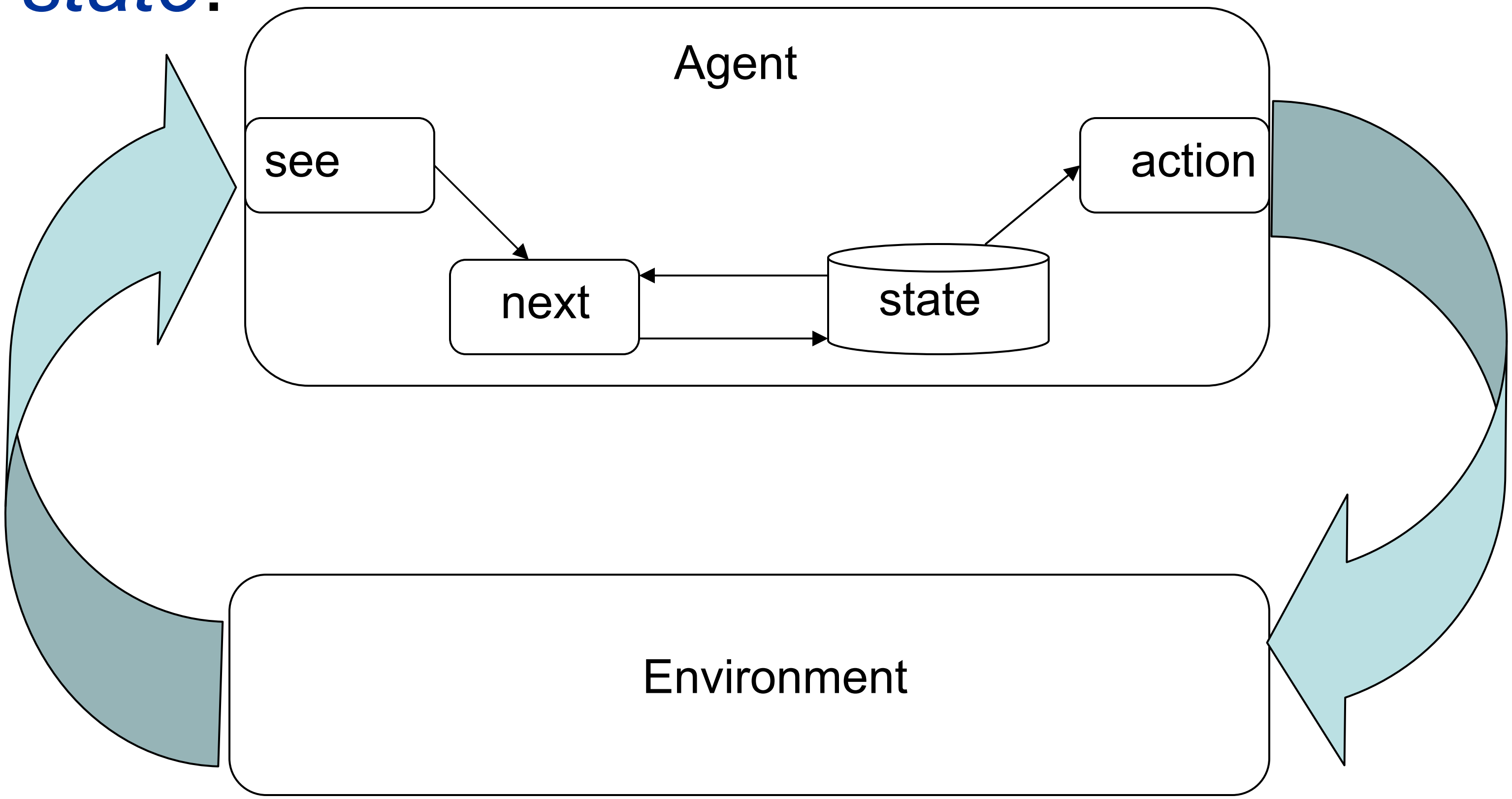
Limited functionality:

Work well only when

- the environment is fully observable and
- the condition-action rules have predicted all necessary actions.

Agents with State

- We now consider agents that *maintain state*:



Agents with State

- These agents have some **internal data structure**, which is typically used to record information about the environment state and history.
  Let I be the set of all internal states of the agent.
- The perception function see for a state-based agent is unchanged:

$$see : E \to Per$$

The action-selection function $action$ is now defined as a mapping

$$action : I \to Ac$$

from internal states to actions. An additional function $next$ is introduced, which maps an internal state and percept to an internal state:

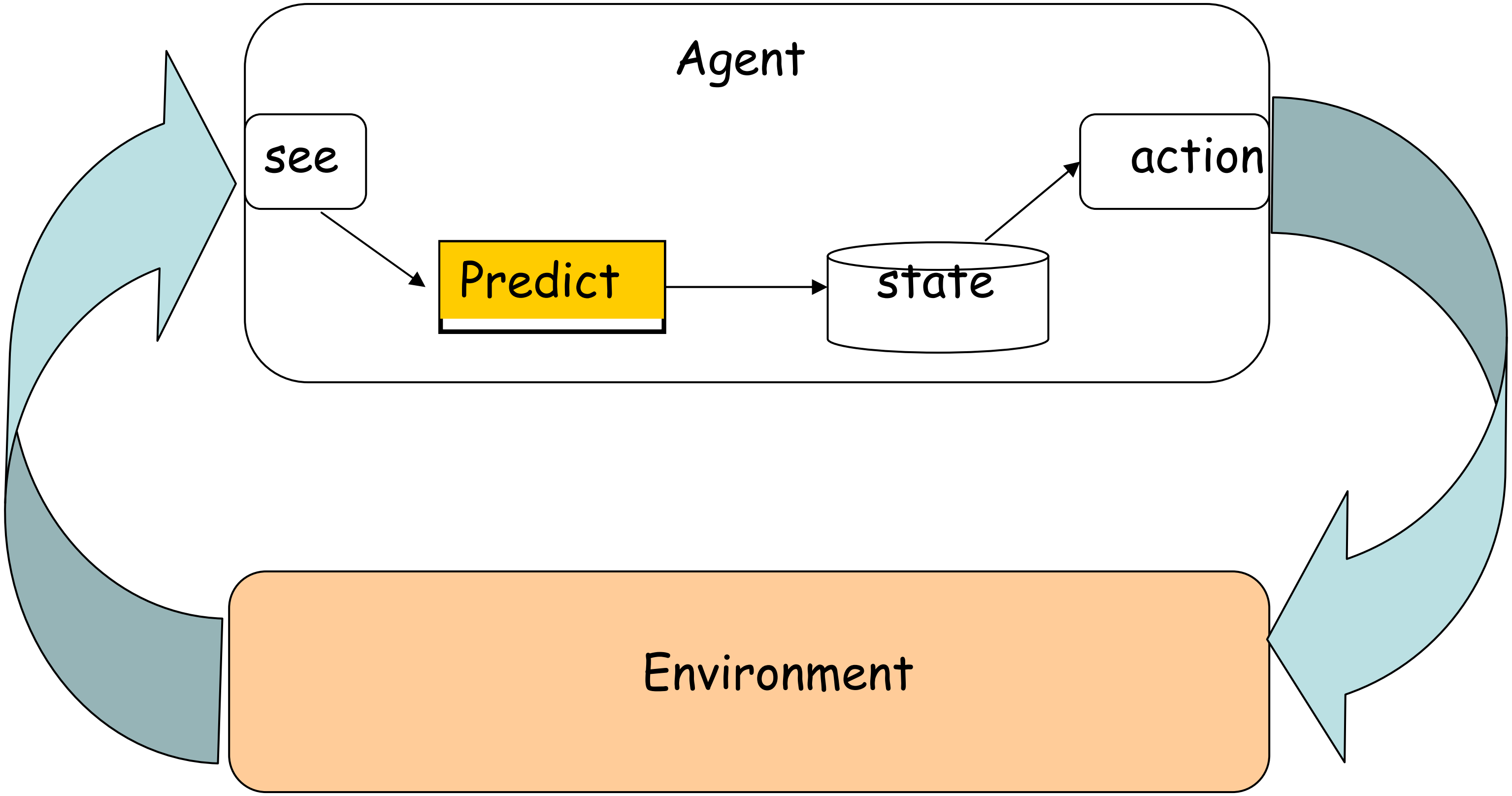
$$next : I \times Per \to I$$

Agent Control Loop

1. Agent starts in some initial internal state i_0
2. Observes its environment state e, and generates a percept $see(e)$
3. Internal state of the agent is then updated via $next$ function, becoming $next(i_0, see(e))$
4. The action selected by the agent is $action(next(i_0, see(e)))$
5. Goto 2

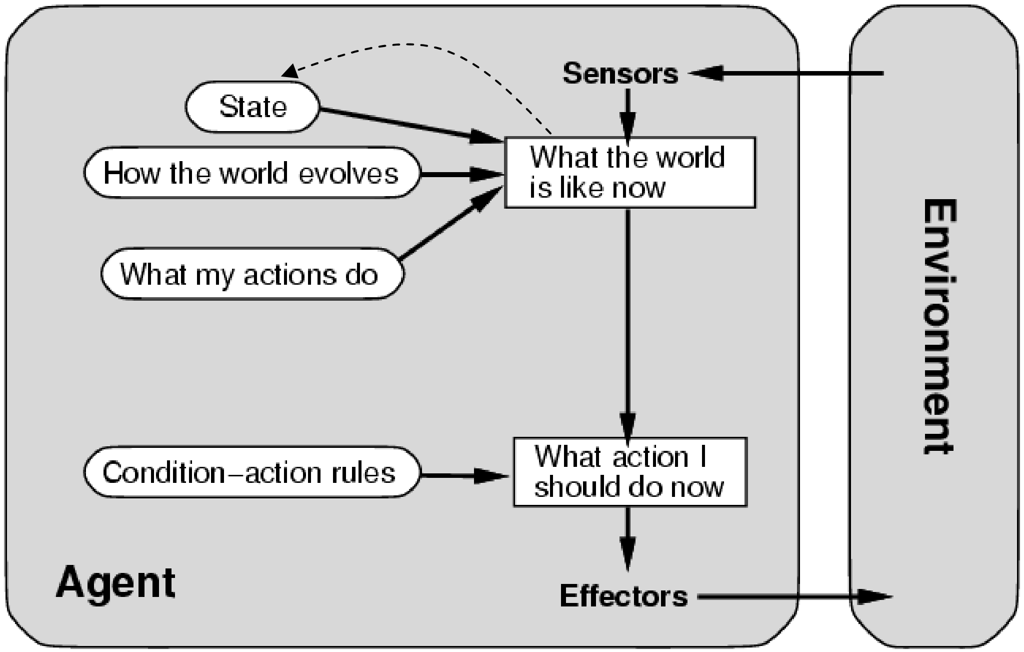
Model-based reflex agents

With internal states



Agents that Keep Track of the World

- Updating internal state requires two kinds of encoded knowledge
  - knowledge about how the world changes (independent of the agents' actions)
  - knowledge about how the agents' actions affect the world
- But, knowledge of the internal state is not always enough
  - how to choose among alternative decision paths (e.g., where should the car go at an intersection)?
  - Requires knowledge of the **goal** to be achieved



function Reflex-Agent-With-State(*percept*) **returns** action
  static: *rules*, a set of condition-action rules
    state, a description of the current world

  state ← Update-State(*state*, *percept*)
  rule ← Rule-Match(*state, rules*)
  action ← Rule-Action[*rule*]
  state ← Update-State(*state*, *action*)
  return *action*

Model-based reflex agents

- Have information about how the world behaves – Model of the World.
- They can work out information about the part of the world which they have not seen (Handle partially observable environments).

The model of the world allows them to

- Use information about how the world evolves to keep track of the parts of the world they cannot see
  - Example: If the agent has seen an object in a place and has since not seen any agent moving towards that object then the object is still at that place.
- Know the effects of their own actions on the world.
  - Example: if the agent has moved northwards for 5 minutes then it is 5 minutes north of where it was.

Deliberative Architecture

- The **classical approach** to building agents is to view them as a particular type of ***knowledge-based system***
- This paradigm is known as ***symbolic AI***
- We define **a deliberative agent or agent architecture** to be one that:
  - contains an explicitly represented, ***symbolic model of the world***
  - makes decisions (for example about what actions to perform) *via* ***symbolic reasoning***

Logic-based architecture

- The ‘traditional’ symbolic artificial intelligence approach
- The agent possesses a symbolic representation of its environment (logical formulas) and rules on how it should behave and what actions it can take
- The behaviour of the system is generated by syntactic manipulation of the symbolic representations (logical deduction)
- **Agent execution as theorem proving**: If there is a theory ϕ that explains how the agent behaves, how goals are generated and how the agent can take action to satisfy them, then this **specification can be directly executed to produce behaviour**

Logic-Based Agents

Decision making is realized through logical deduction

Agent viewed as a kind of **knowledge-based system**

- Contains an explicitly represented symbolic model of the world
- Takes decisions via symbolic reasoning

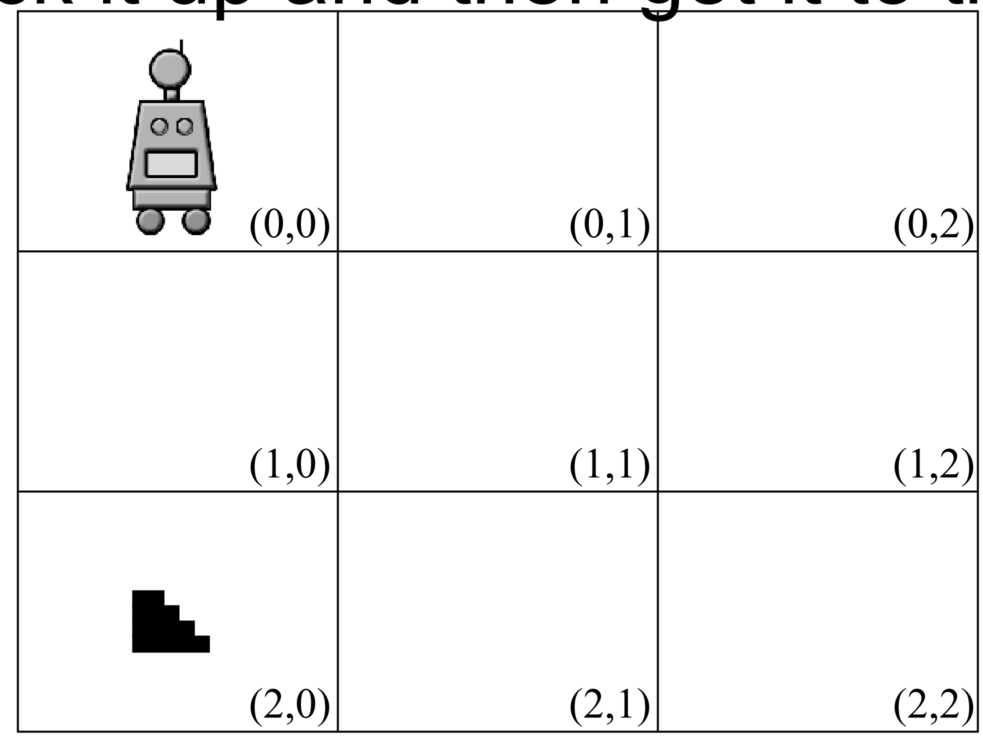
Problems:

- Translating the real world into an accurate and adequate symbolic description, in real-time
- How to symbolically represent information about complex real-world entities

- The agent's decision-making process is modelled through the **rules of inference** ρ
- *KB* $\Rightarrow_\rho \phi$: ϕ can be proven from the inference rules ρ
- **The agent programmer has to encode the inference rules ρ in a way that that enables the agent to decide what to do**

The Maze World

The agent’s objective is to discover the gold, pick it up and then get it to the exit (2,2)



Starting position (0,0) facing *East*

The state of the world is described by the following predicates

- *In*(x,y) the agent is in square with coordinates (x,y)
- *Gold*(x,y) there is gold in square (x,y)
- *Facing*(d) the agent faces $d \in$ {*North*, *South*, *East*, *West*}

Perception:

- The agent can perceive the world by detecting whether or not there is gold in a square, *gold* or *null* respectively
- It can also perceive its position on the grid and its direction

Possible actions *A*={*pick-up*, *forward*, *turn*}

When the agent turns, it turns 90 degrees clockwise

- The rules of inference ρ determine the agent's behaviour
- Rule for picking up the gold when detected:

$In(x,y) \wedge Gold(x,y) \rightarrow Do(pick\text{-}up)$

- Rules to enable the agent to move around:

$In(0,0) \wedge Facing(East) \wedge \neg Gold(0,0) \rightarrow Do(forward)$

$In(0,1) \wedge Facing(East) \wedge \neg Gold(0,1) \rightarrow Do(forward)$

$In(0,2) \wedge Facing(East) \wedge \neg Gold(0,2) \rightarrow Do(turn)$

$In(0,2) \wedge Facing(South) \wedge \neg Gold(0,2) \rightarrow Do(forward)$

…

GENERALIZING

- The environment is described by sentences in ***L*, *KB*= *P*(*L*)**
- At every moment in time t an agent's internal state is $KB_t \in KB$
- Environment states $S=\{s_1, s_2, \ldots\}$
- Perception $see: S \rightarrow P$
- The agent's internal state is updated by percepts:

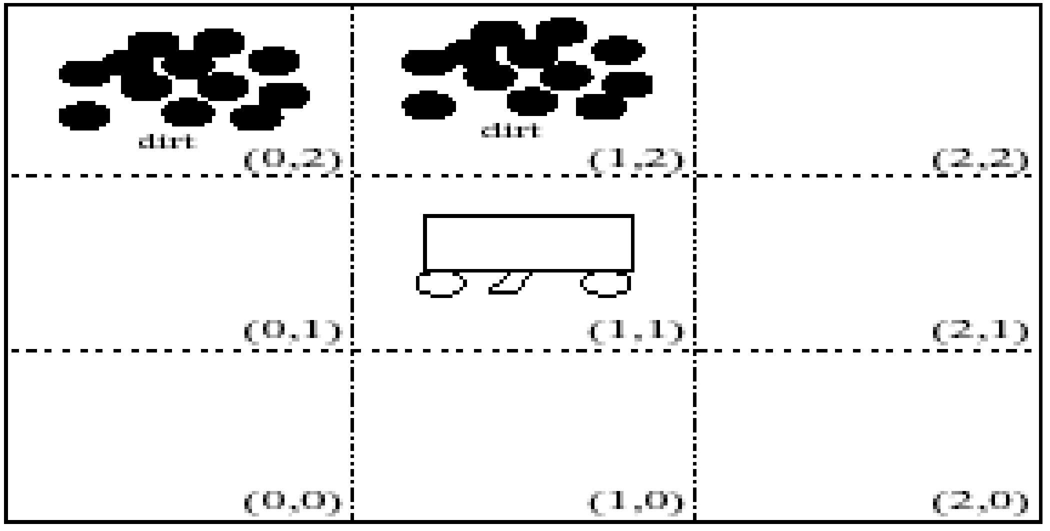
  $next: KB \times P \rightarrow KB$
- An agent can choose an action from a set $A=\{a_1, a_2, \ldots\}$:

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

  $do: A \times S \rightarrow S$

Deductive Reasoning Agents

- An example: The Vacuum World



Deductive Reasoning Agents

- Use 3 *domain predicates* to solve problem:

  In(x, y) agent is at *(x, y)*

  Dirt(x, y) there is dirt at *(x, y)*

  Facing(d) the agent is facing direction *d*

- Possible actions:

  Ac = {turn, forward, suck}

P.S. *turn* means "turn right"

Deductive Reasoning Agents

- Rules ρ for determining what to do:

$$
\begin{aligned}
In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) &\longrightarrow Do(forward) \\
In(0,1) \wedge Facing(north) \wedge \neg Dirt(0,1) &\longrightarrow Do(forward) \\
In(0,2) \wedge Facing(north) \wedge \neg Dirt(0,2) &\longrightarrow Do(turn) \\
In(0,2) \wedge Facing(east) &\longrightarrow Do(forward)
\end{aligned}
$$

- …and so on!
- Using these rules (+ other obvious ones), starting at (0, 0) the robot will clear up dirt

Deductive Reasoning Agents

- Problems:
  - How to convert video camera input to *Dirt(*0, 1*)*?
  - decision making assumes a *static* environment: *calculative* rationality
  - decision making using first-order logic is *undecidable*!
- Even where we use *propositional* logic, decision making in the worst case means solving co-NP-complete problems
  (PS: co-NP-complete = bad news!)
- Typical solutions:
  - weaken the logic
  - use symbolic, non-logical representations
  - shift the emphasis of reasoning from *run time* to *design time*
- **We will look at some examples of these approaches in this class**

More Problems…

- The "logical approach" that was presented implies adding and removing things from a database
- That's not pure logic
- Early attempts at creating a "planning agent" tried to use true logical deduction to the solve the problem

Advantages of logic-based approach

- If there is a theory ϕ which describes the agent's behaviour, all we have to do is execute this specification
- Elegant, intuitive, clear semantics

Disadvantages of logic-based approach

Two issues in building agents with traditional AI approach

- Transduction problem: how can the world be translated into a meaningful symbolic model at the right abstraction level and in time for that model to be useful (images, speech etc.)
- Representation problem: how to represent information in a symbolic form suitable for the agents to reason with and in time for the results of the reasoning to be useful (knowledge representation, reasoning and planning)

Other issues

- How to transform percepts into declarative statements that describe the environment precisely enough
- Writing down all the rules that would allow agents to operate in complex environments is unrealistic
- Assumes calculative rationality: the world does not change in a significant way while the agent is deliberating – not realistic
- Computational complexity of theorem proving is a problem. Propositional logic is decidable, but first-order logic is only semi-decidable: even if there is a proof, the theorem prover may fail to terminate
- Representing temporal information and changes is difficult