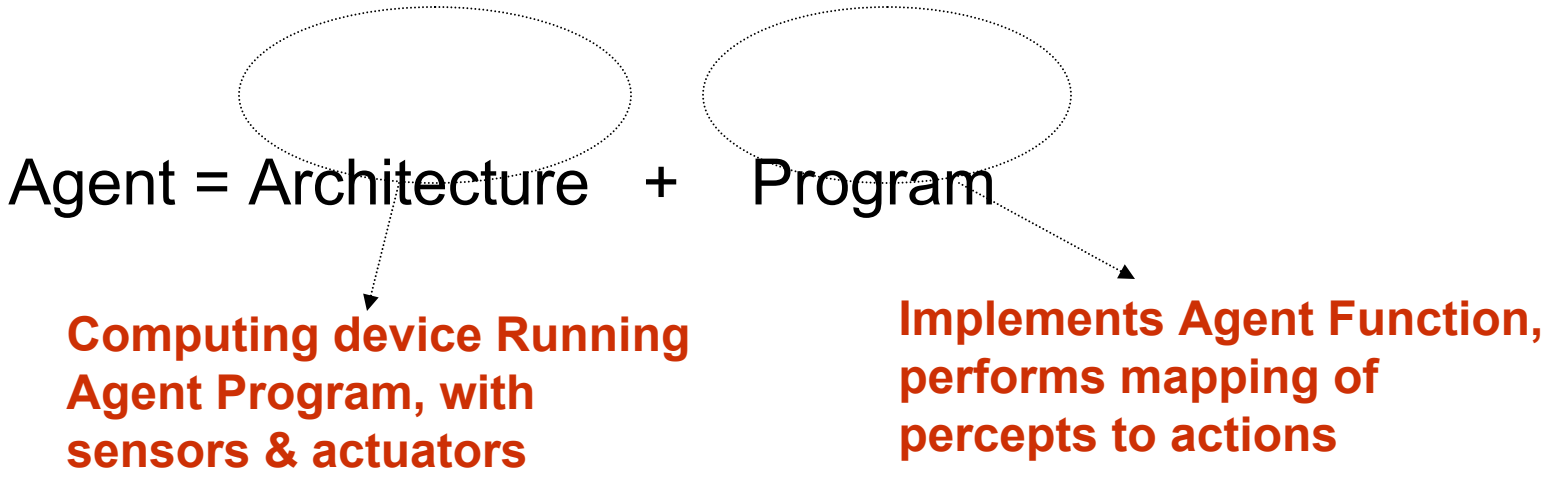
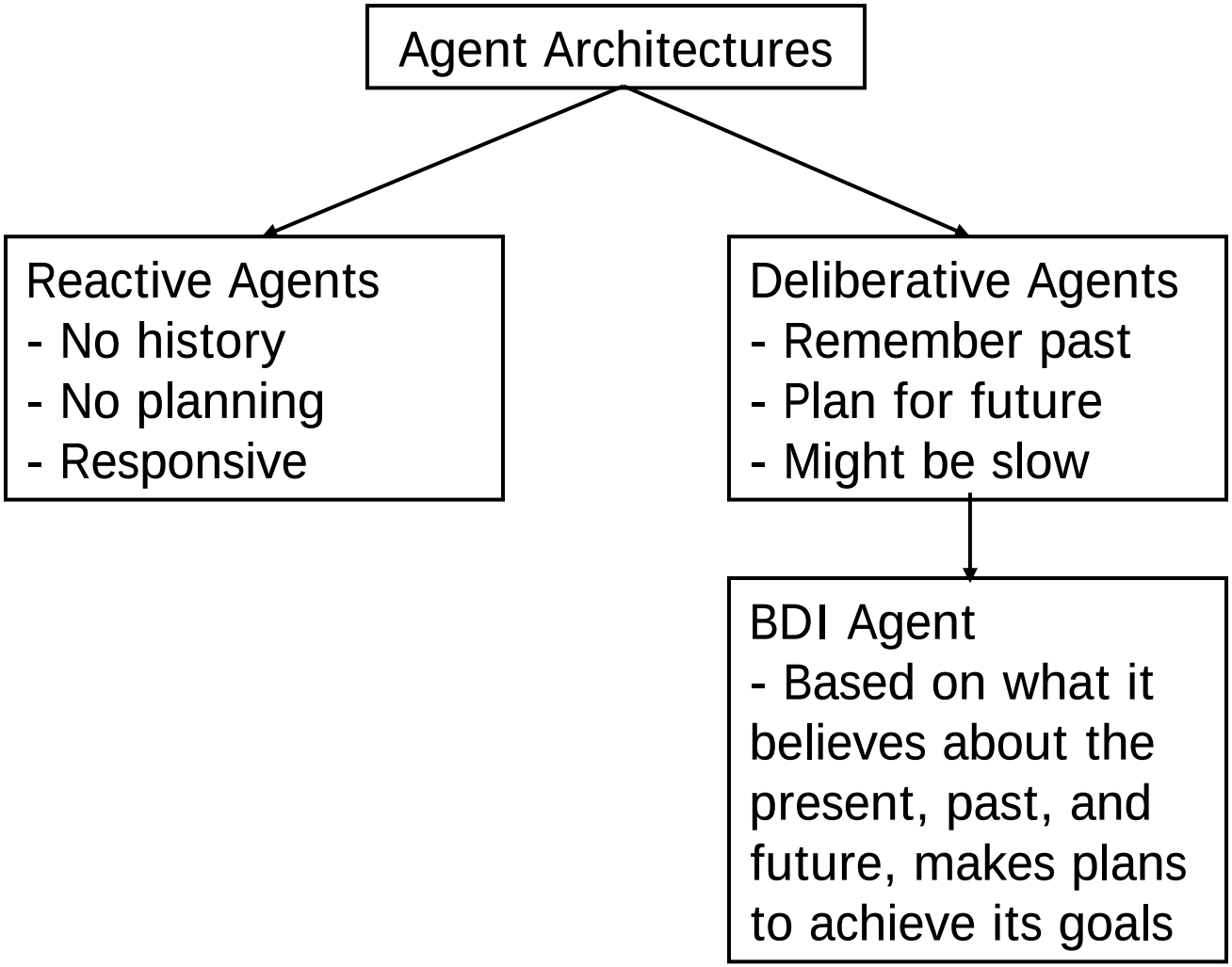
Structure of Agents

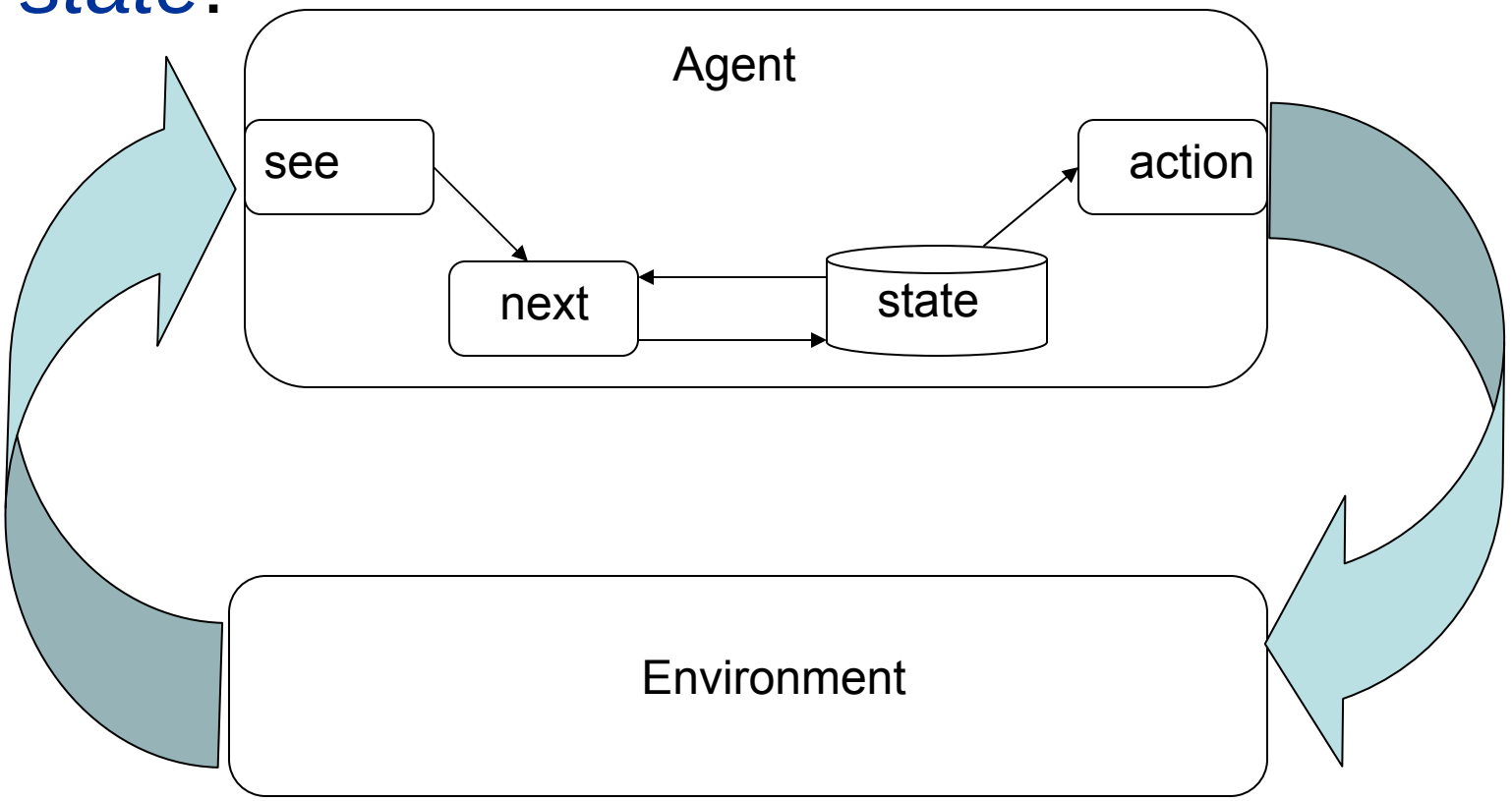


Architectural Types



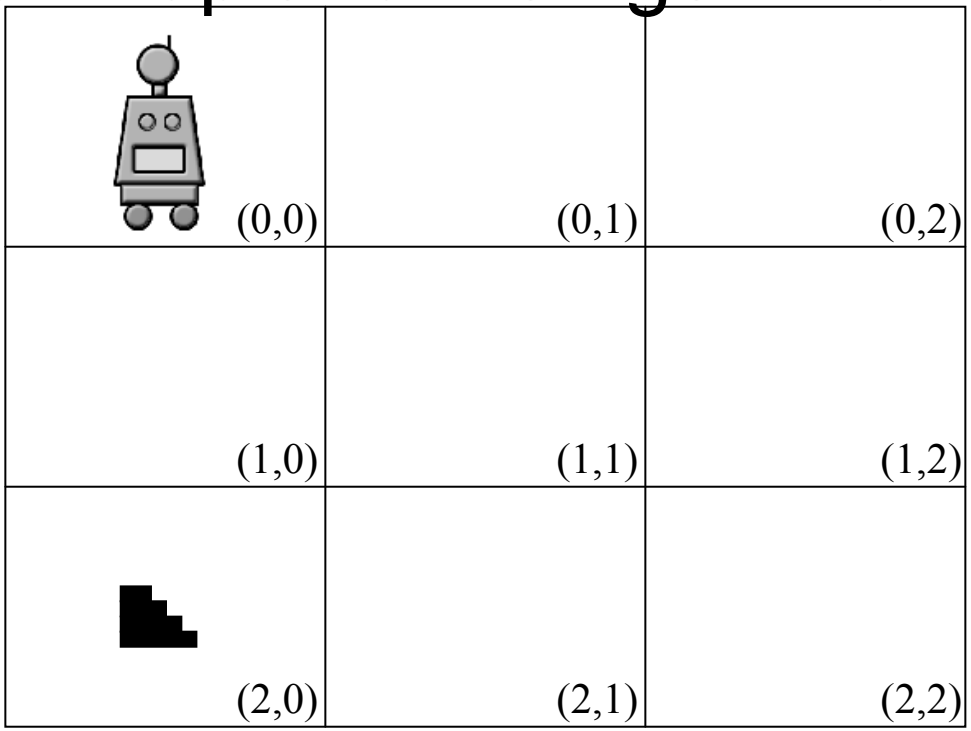
Agents with State

- We now considered agents that *maintain state*:



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

Logic-Based/Model Based

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 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:

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

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

We have to 'tell' the agent what to do at each step!

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

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

…

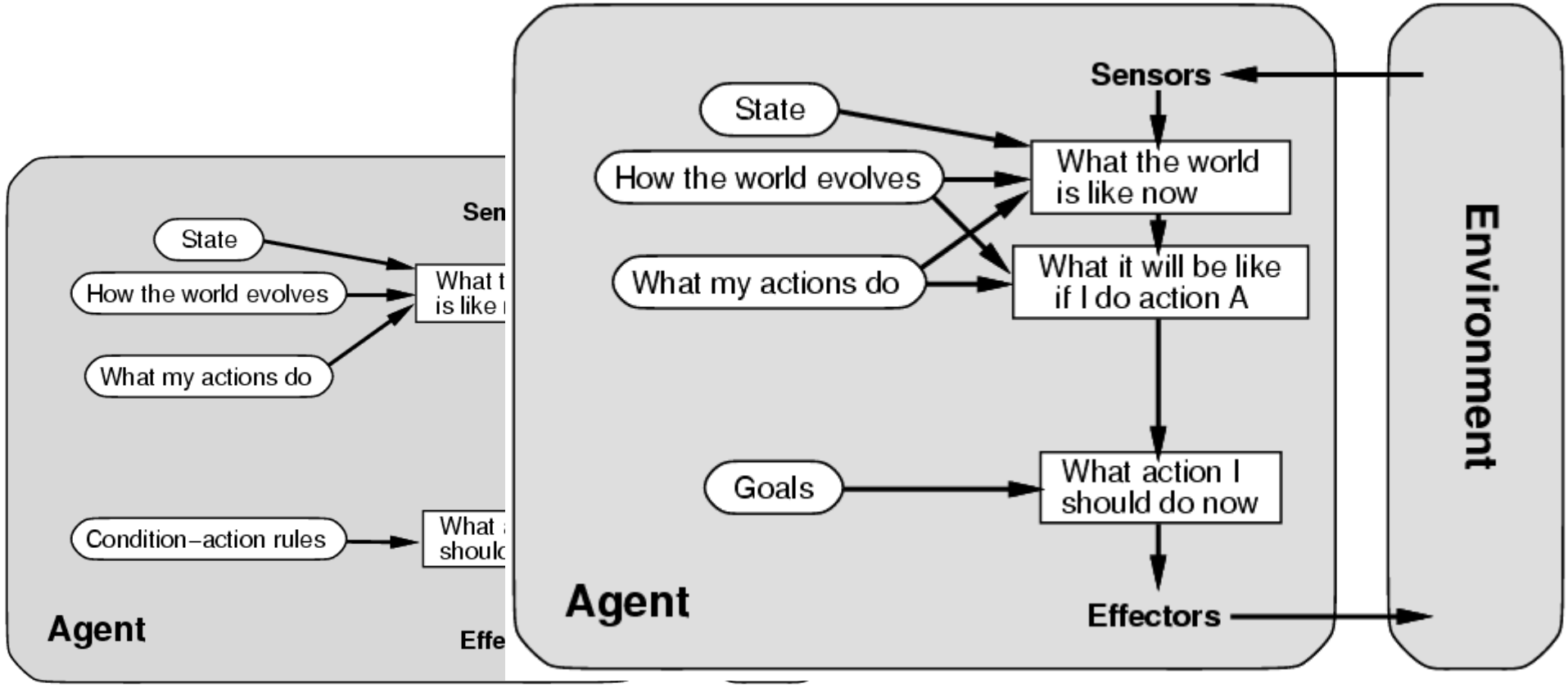
Goal-based agents

World Model (as model-based agents)

+ Goals

Goals are situations that are desirable.

The goals allow the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.



rectangles ← the current internal state; Ovals ← background information

- Reasoning about actions
  - reflex agents only act based on pre-computed knowledge (rules)
  - **goal-based (planning) act by reasoning about which actions achieve the goal**

Goal-based agents

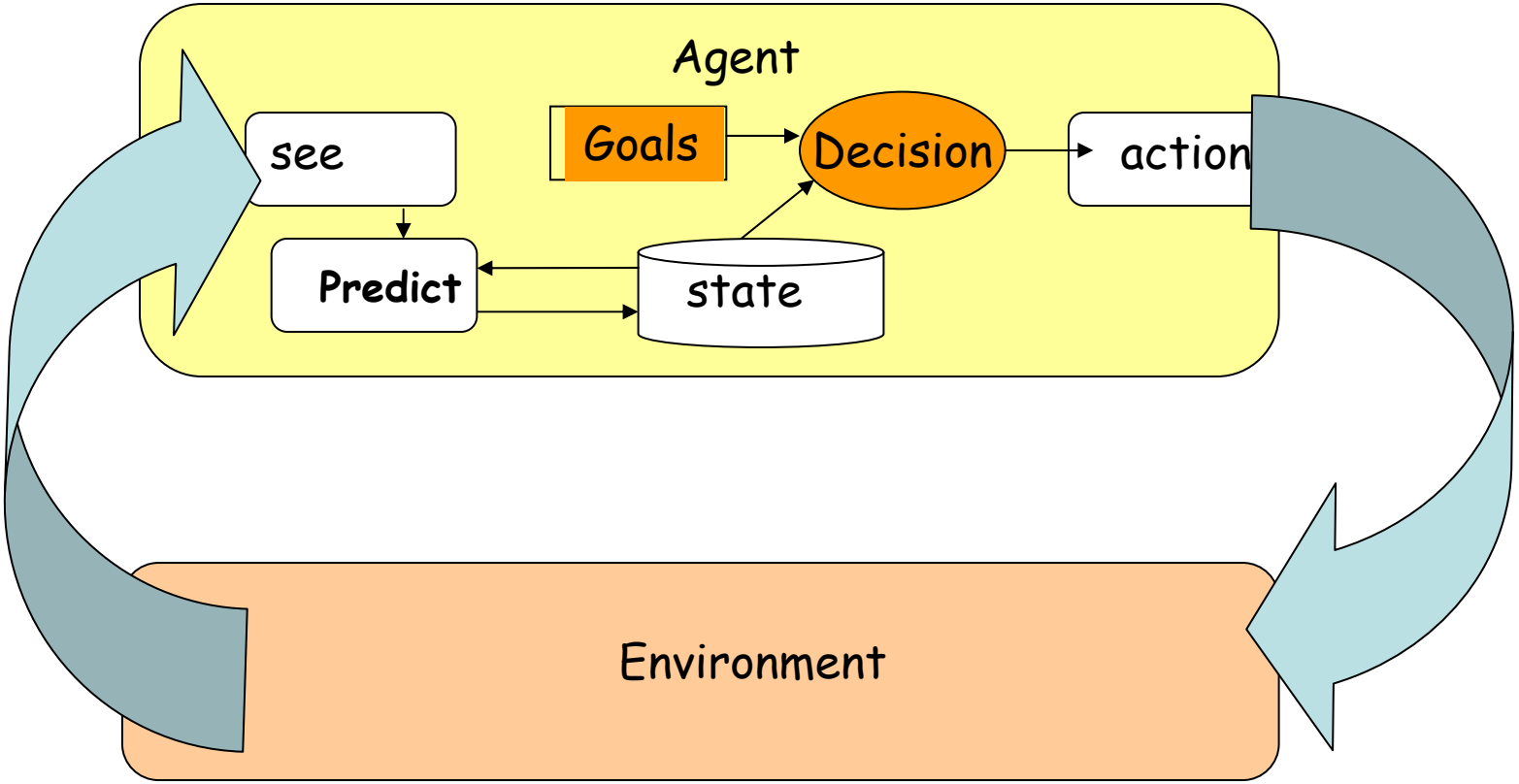
Differences from Reflexive Agents:

- **Goals are explicit**
- The future is taken into account
- **Reasoning about the future** is necessary – planning, search.

Goal-Based Agents

- Having a goal, combined with the current state information can help select the possible next action
  - Possibly agent may need to consider every *alternative action sequences* leading to the goal ➔ search for a sequence leading to goal

Goal-based agents



Problem Solving Agent

- A kind of *Goal based Agent*
- Decides what to do by finding the sequences of actions that lead to **desirable states**

Formulate Goal, Formulate Problem

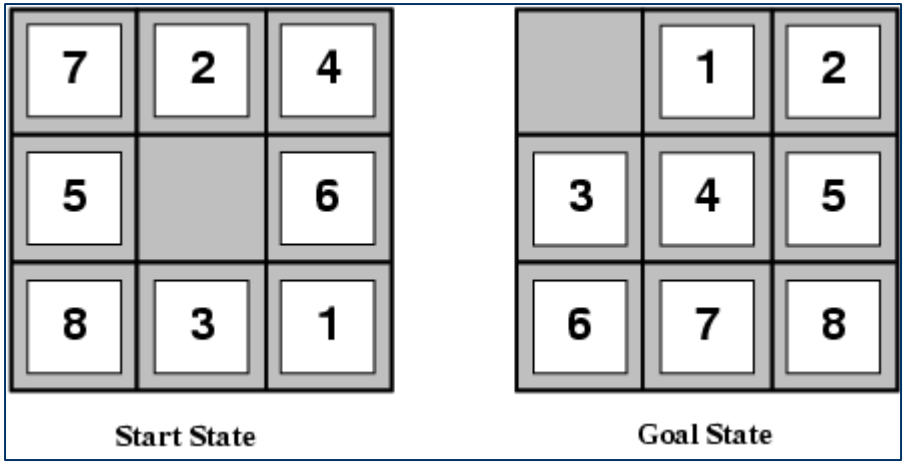
Search

Execute

Problems

- Four components of problem definition
  - Initial state
  - Possible Actions
    - Uses a *Successor Function*
      - Returns <*action*, *successor*> pair
    - *State Space*
    - *Path*
  - Goal Test
  - Path cost
    - *Step cost*
- *Problem formulation* is the process of deciding what actions and states to consider, given a goal

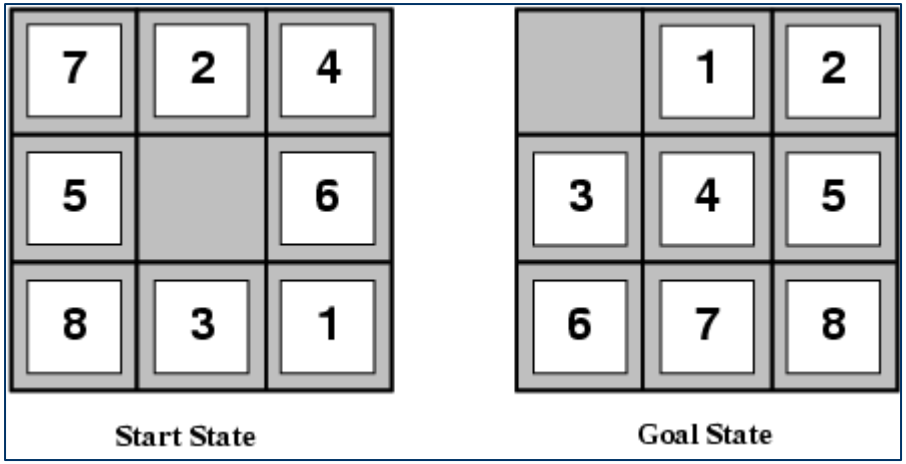
Example : The 8-Puzzle



A typical Instance of 8-puzzle

- States ?
- Initial State ?
- Successor Function ?
- Goal Test ?
- Path Cost ?

Example : The 8-Puzzle



A typical Instance of 8-puzzle

- States : Location of Tiles
- Initial State : One of States
- Successor Function: Move blank left, Right, Up, down
- Goal Test : Shown in Fig. Above
- Path Cost : 1 for each step

Solutions

- A Solution to the problem is the path from the initial state to the final state
- Quality of solution is measured by *path cost* function
  - *Optimal Solution* has the lowest path cost among other solutions

- *An Agent with several immediate options of unknown value can decide what to do by first examining different possible sequences of actions that lead to a state of known value, and then choosing the best sequence*
- ***Searching Process***
  - Input to Search : Problem
  - Output from Search: Solution in the form of Action Sequence

Problem Solving Agent

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
          state, some description of the current world state
          goal, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

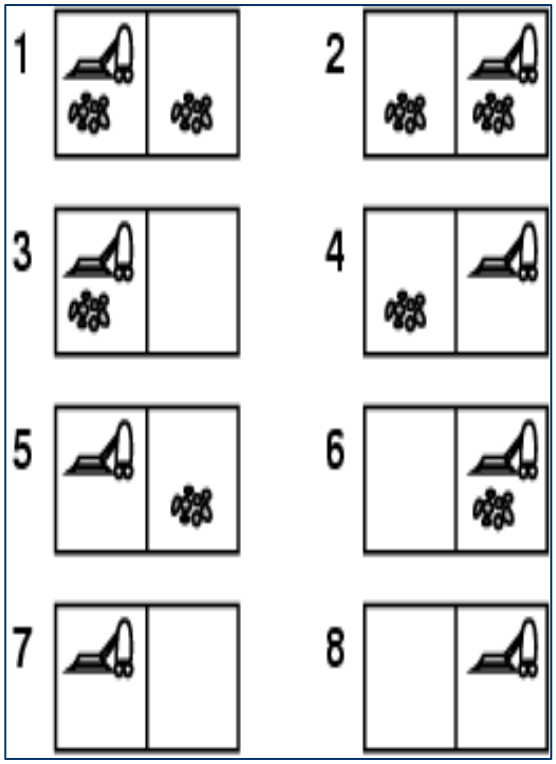
A Problem solving Agent, Assuming the environment is

- **Static**
- **Observable**
- **Discrete**
- **Deterministic**

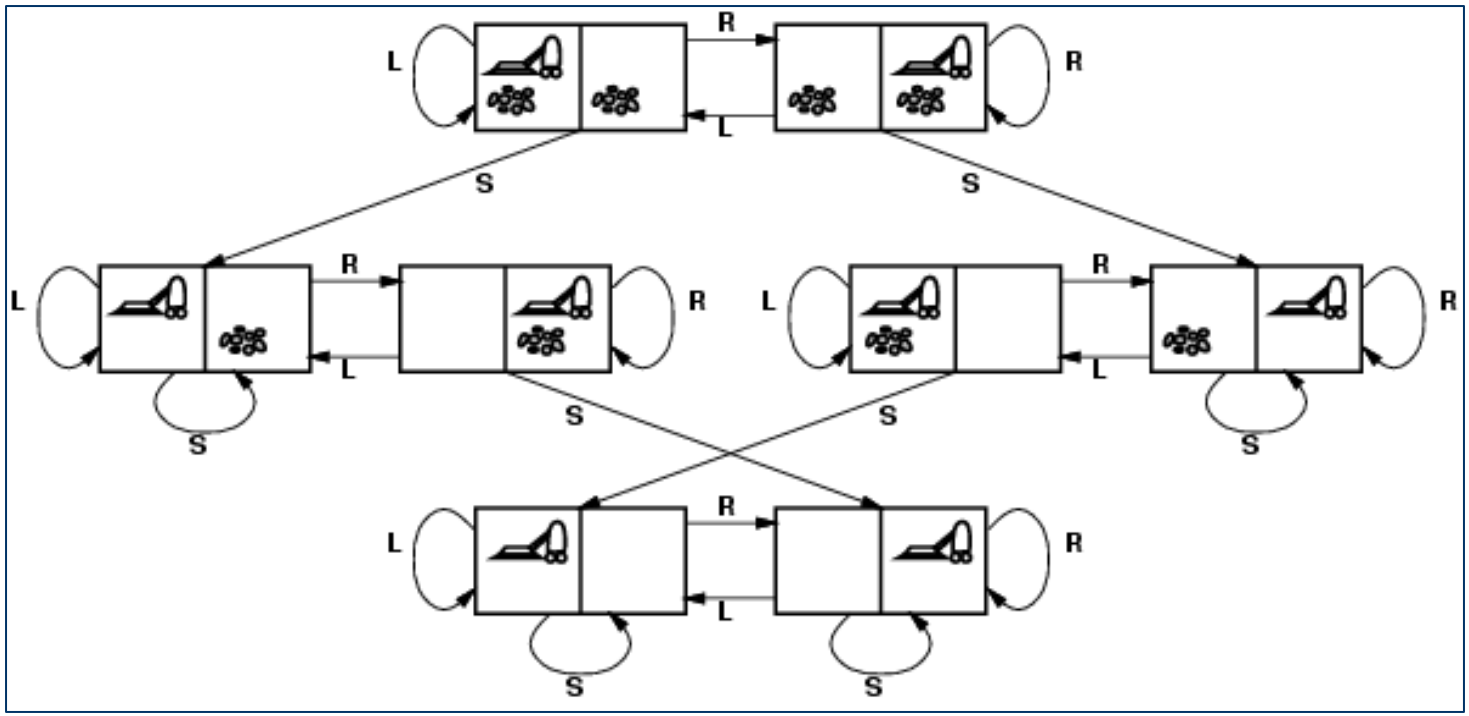
Example : Vacuum World

Problem Formulation

- States
  - 2 x 2^2 = 8 states
- Initial State
  - Any one of 8 states
- Successor Function
  - Legal states that result from three actions (Left, Right, Suck)
- Goal Test
  - All squares are clean
- Path Cost
  - Number of steps (each step costs a value of 1)



Example : Vacuum World



State Space for the Vacuum World.

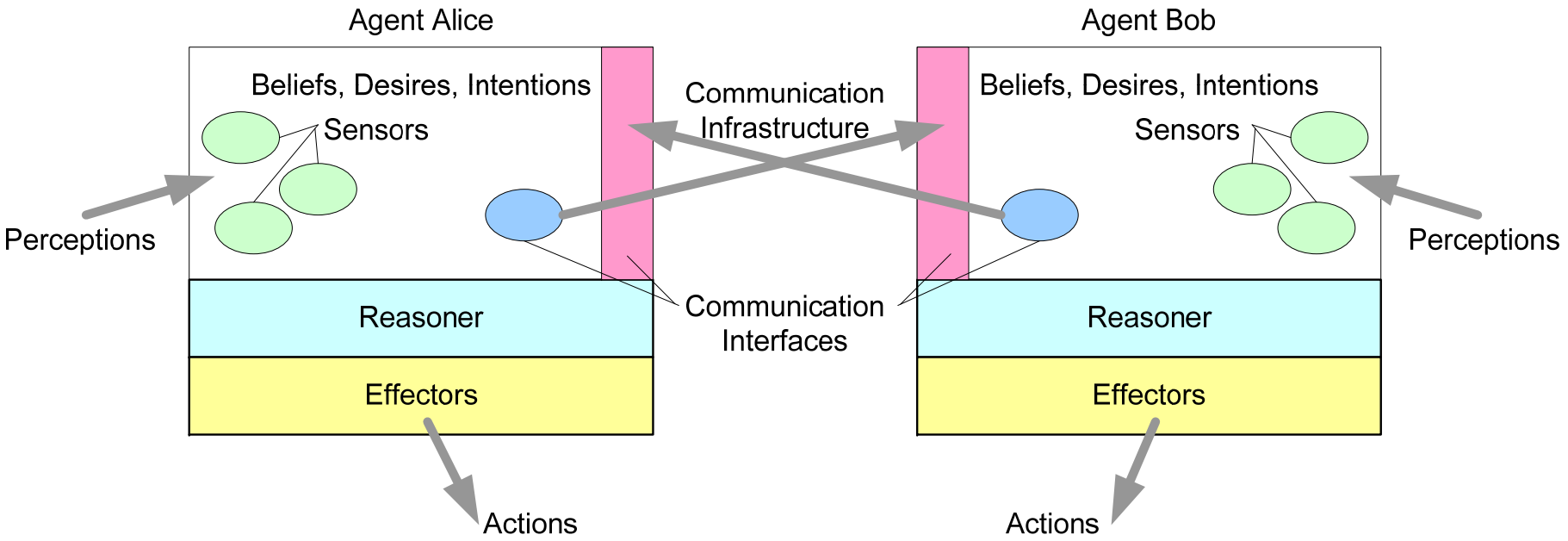
Labels on Arcs denote **L: Left, R: Right, S: Suck**

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**

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

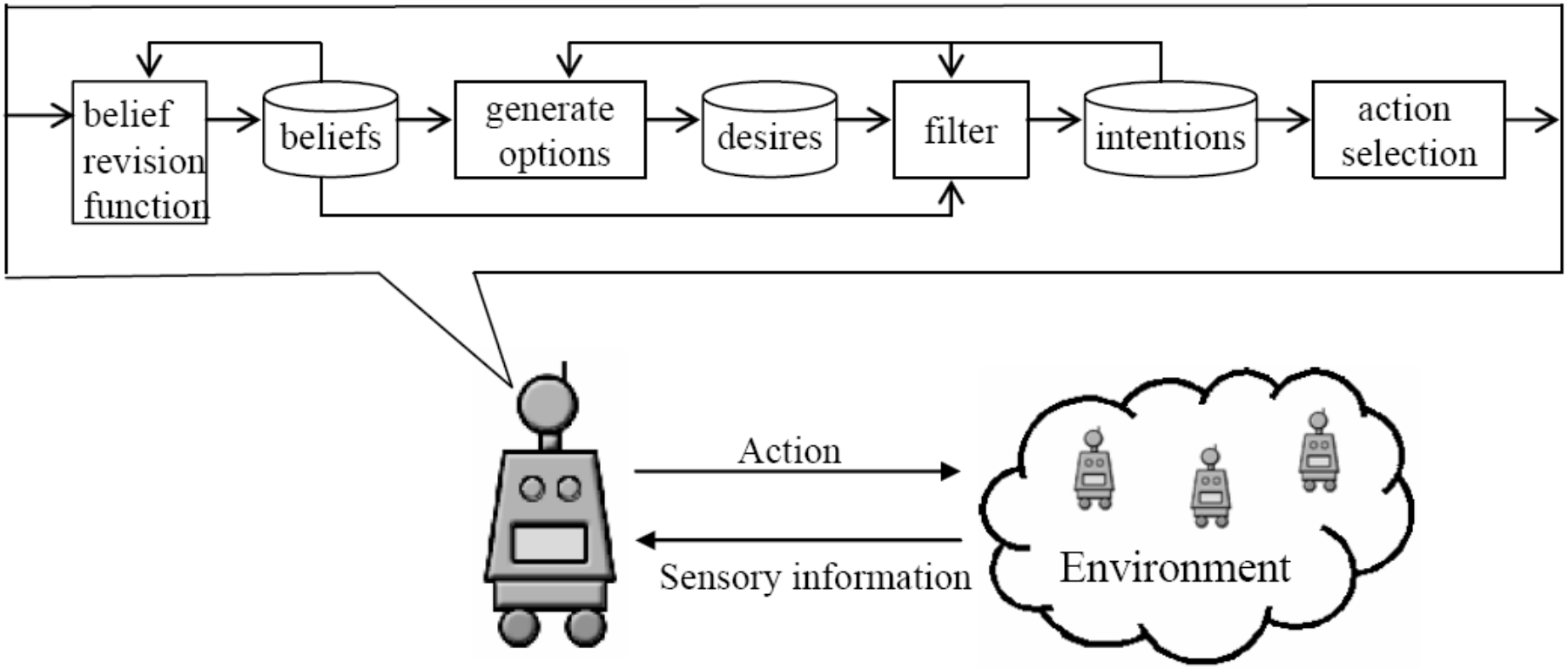
BDI as deliberative architecture

- 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**

Control loop for a BDI agent would take the form:

```
// control loop for BDI-Agent
begin
    B:=B0;  // initial beliefs
    I:=I0; // 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
```

BDI components



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

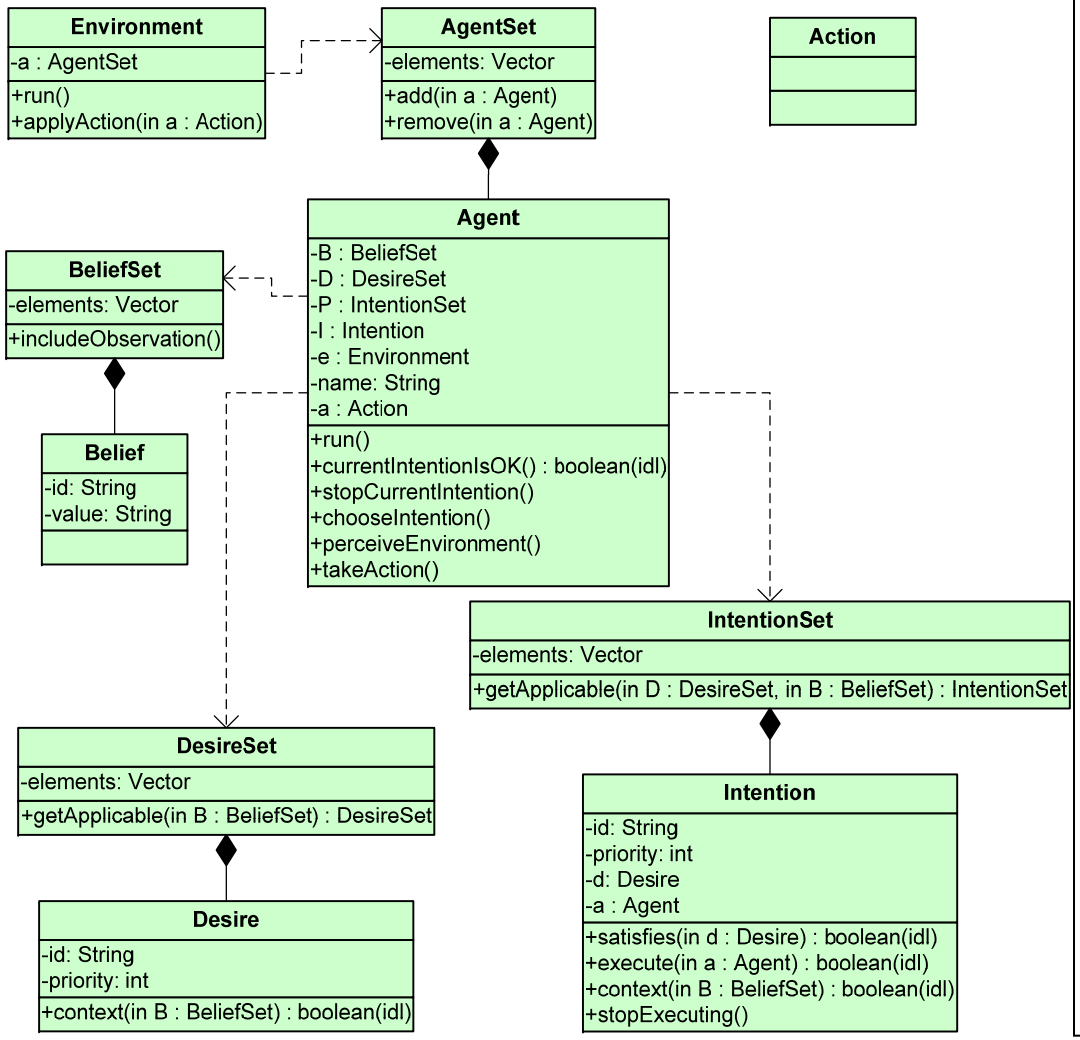
  $brf: P(Beliefs) \times P \rightarrow P(Beliefs)$
- A set of desires (*Desires*)
- An option generation function:

  $options: P(Beliefs) \times P(Intentions) \rightarrow P(Desires)$
- 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

  $asf: P(Intentions) \rightarrow A$

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

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

NEXT WEEK

- **Whole week dedicated to introduction to the Brahms agent programming language!**

- **Guest Lecturer: Dr. Nicola Bicocchi expert in agent-based programming**

ASSIGNMENT

- **DUE ON JANUARY 25**

1. Read Section 2.4 “The Structure of Agents” from the Textbook)
2. Written response to Exercises 2.5 and 2.6 (at the end of Chapter 2: Agents) shall be handed to me at class on Jan 25.