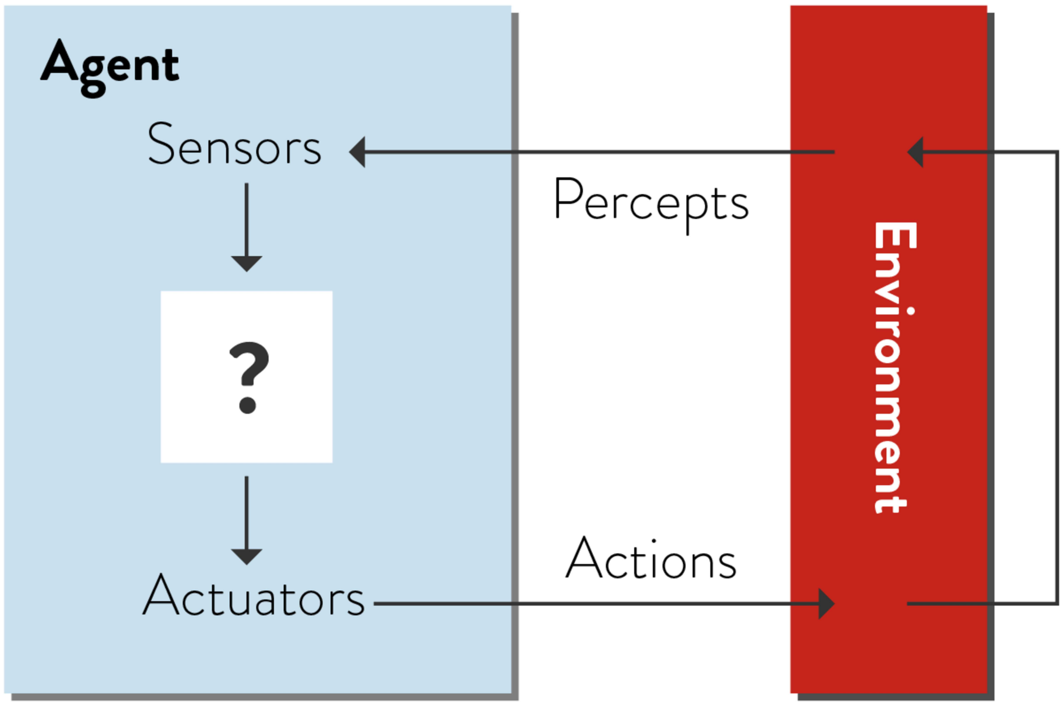
Agents

COMP3411/9814:
Artificial Intelligence

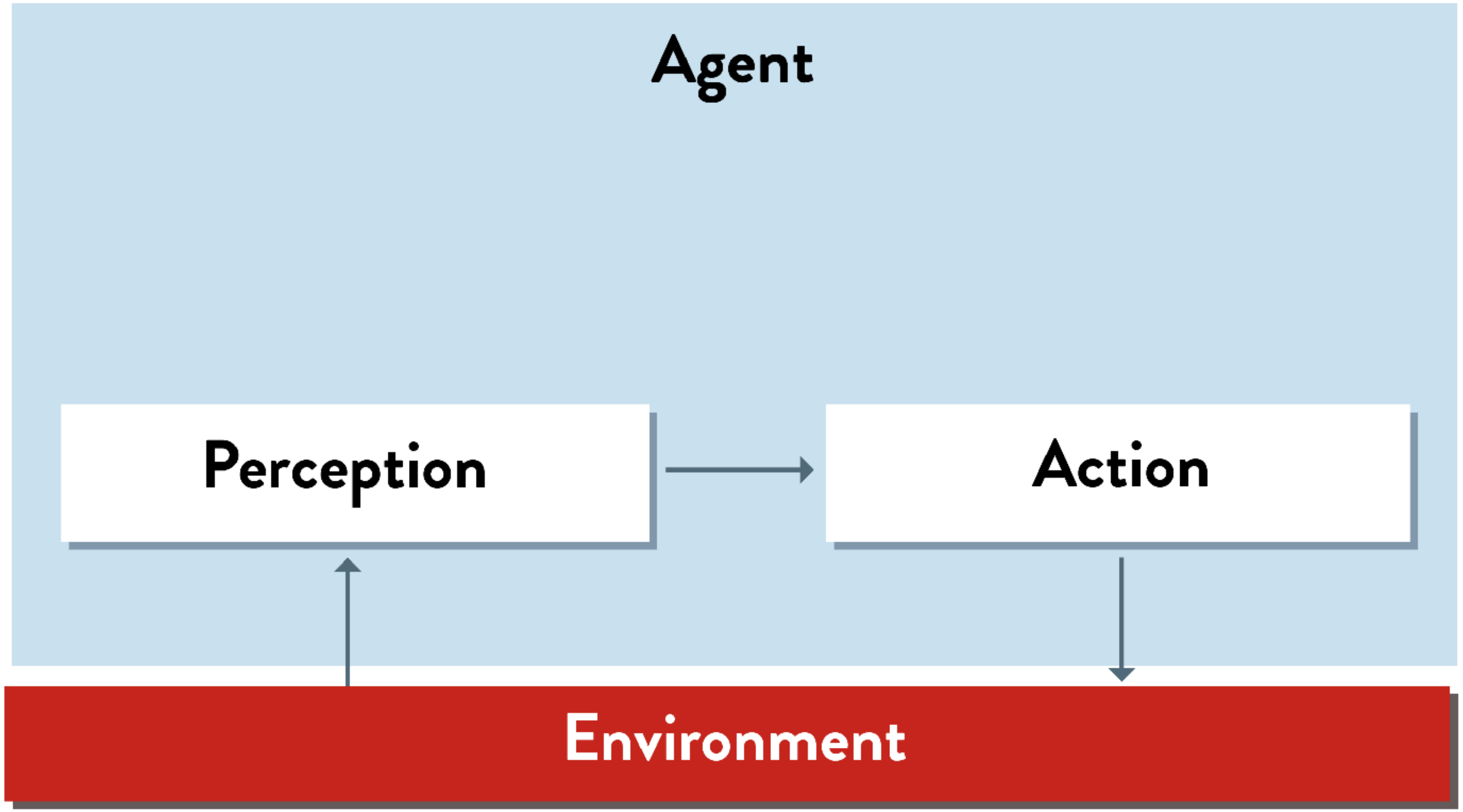
Types of Agents

- Reactive Agent
- Model-Based Agent
- Planning Agent
- Utility-based agent
- Game Playing Agent
- Learning Agent

Agent Model



Reactive Agent



Reactive Agent

- Choose the next action based only on what agent currently perceives
  - Uses a “policy” or set of rules that are simple to apply
- Sometimes called “simple reflex agents”
  - but they can do surprisingly sophisticated things

Reactive Agent

repeat

if left touch:

backup

turn right

else if right touch:

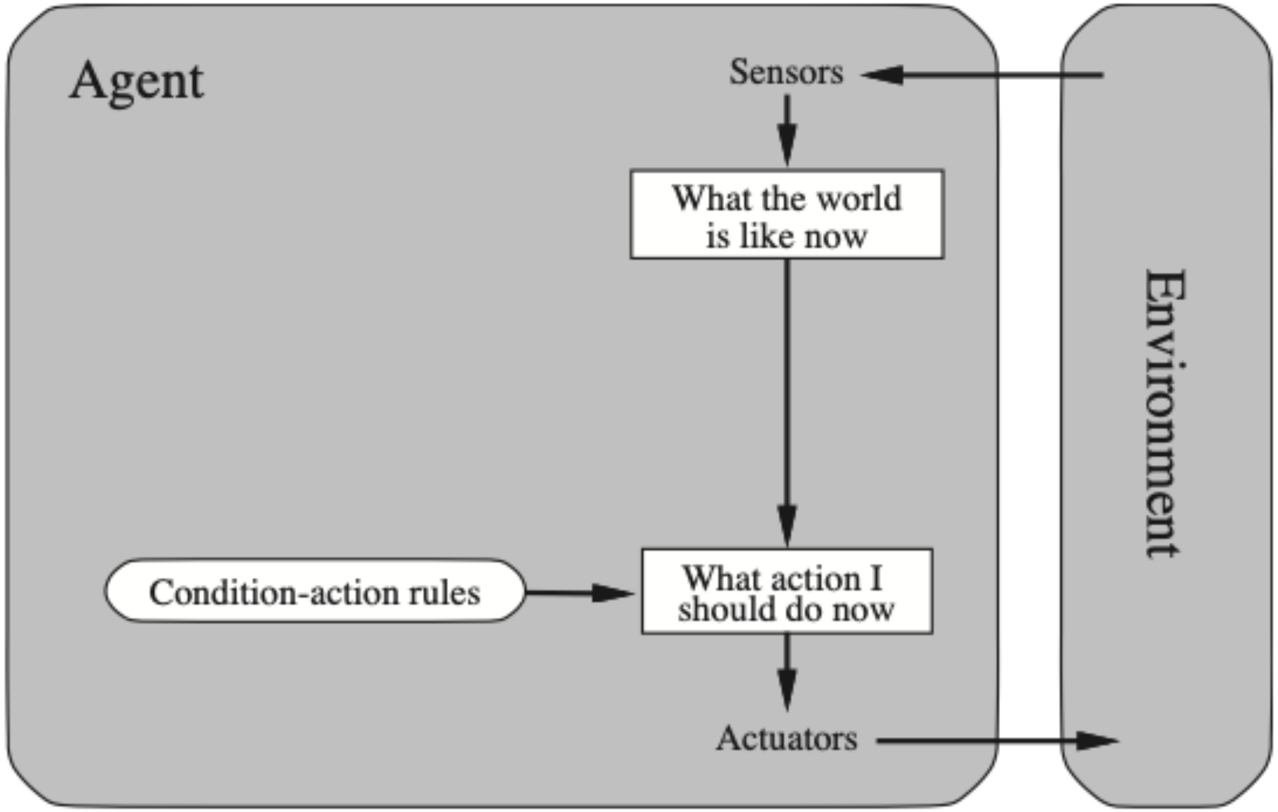
backup

turn left

else

go straight

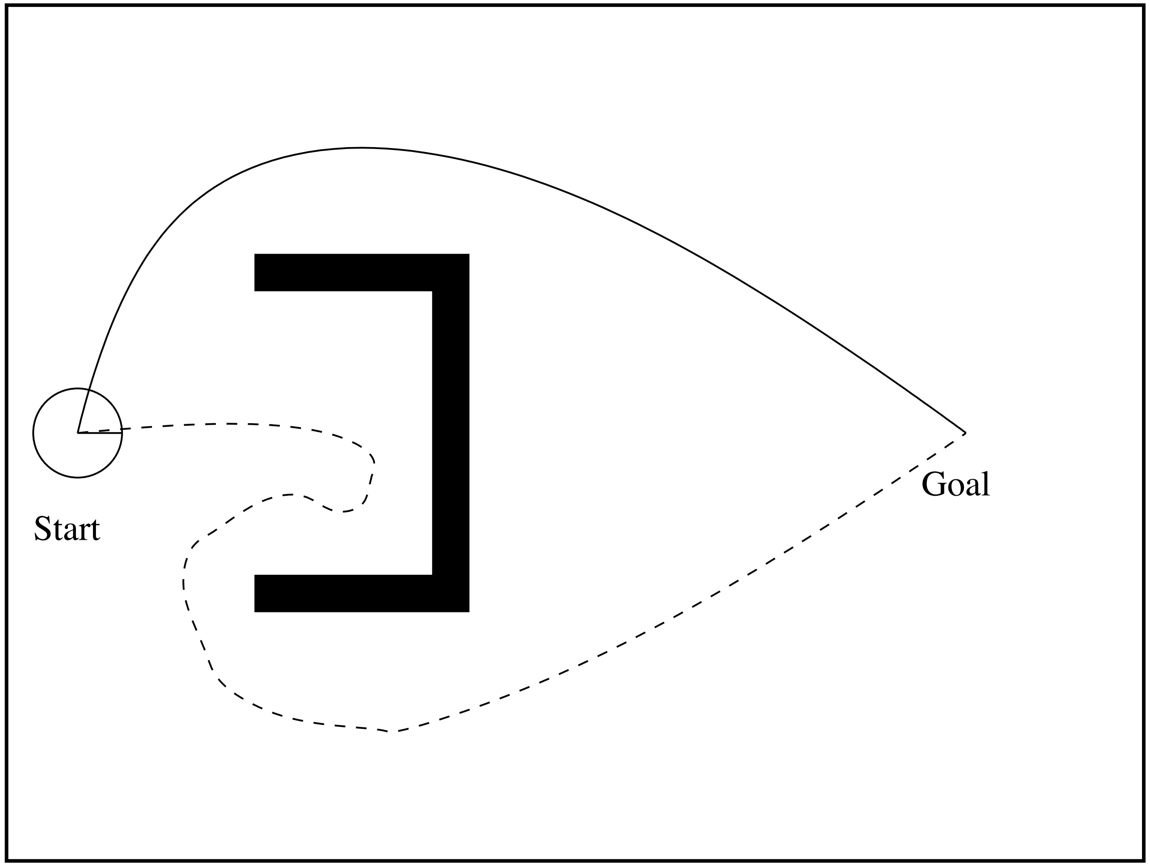
Reactive Agent

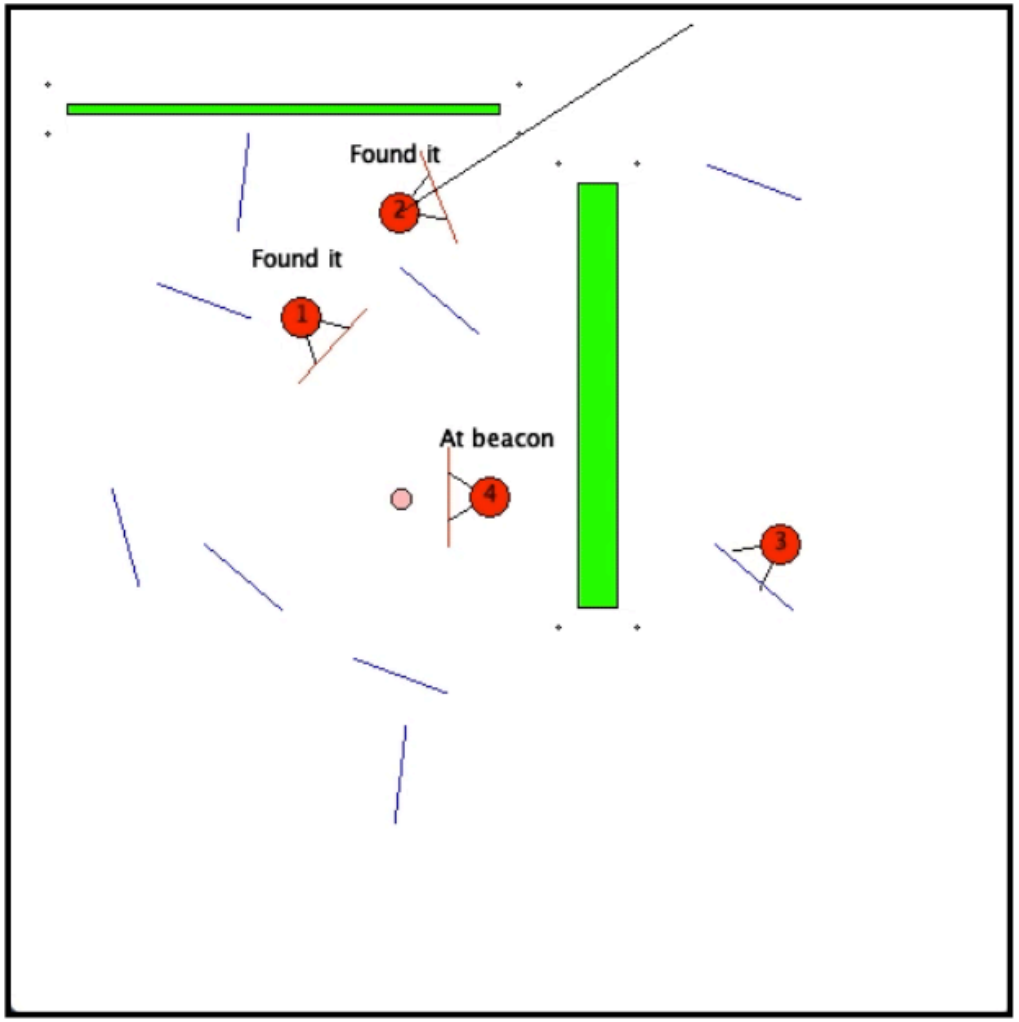


Reflex (reactive) agent — applies condition-action rules to each percept

Reactive Robots

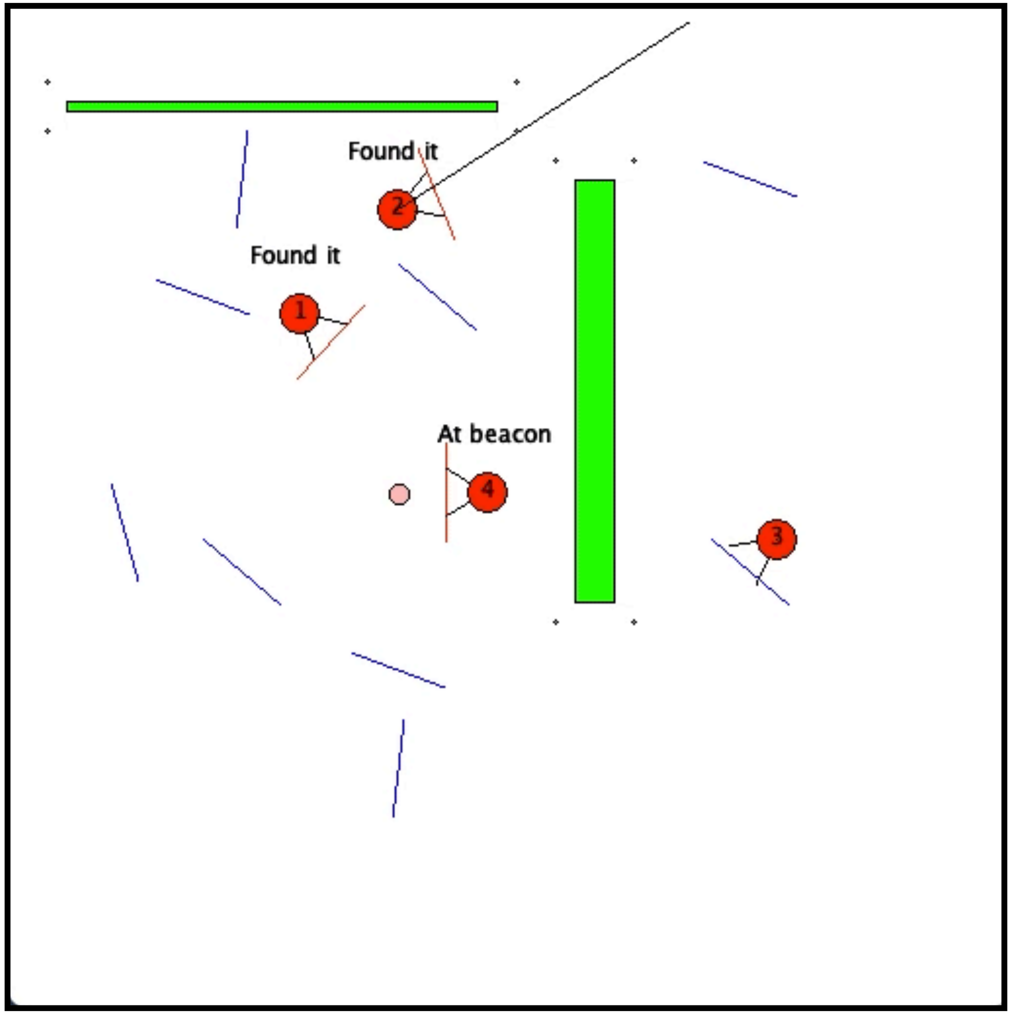
Limitations of Reactive Agents






```
botworld(512, 512,
    [
        box(1, 30, 50, 250, 55),
        box(2, 290, 90, 310, 305),

        beacon(1, 200, 250),

        bar(1, 100, 150, 20),
        bar(2, 200, 390, 275),
        bar(3, 220, 100, 75),
        bar(4, 380, 90, 20),
        bar(5, 80, 80, 275),
        bar(6, 60, 270, 75),
        bar(7, 200, 340, 20),
        bar(8, 120, 90, 275),
        bar(9, 280, 250, 75),
        bar(10, 120, 290, 40),
        bar(11, 380, 290, 40),
        bar(12, 220, 150, 40)
    ],
    [
        bot(1, "north"),
        bot(2, "south"),
        bot(3, "east"),
        bot(4, "west")
    ]
);
```

Teleo-Reactive Program (TOP)

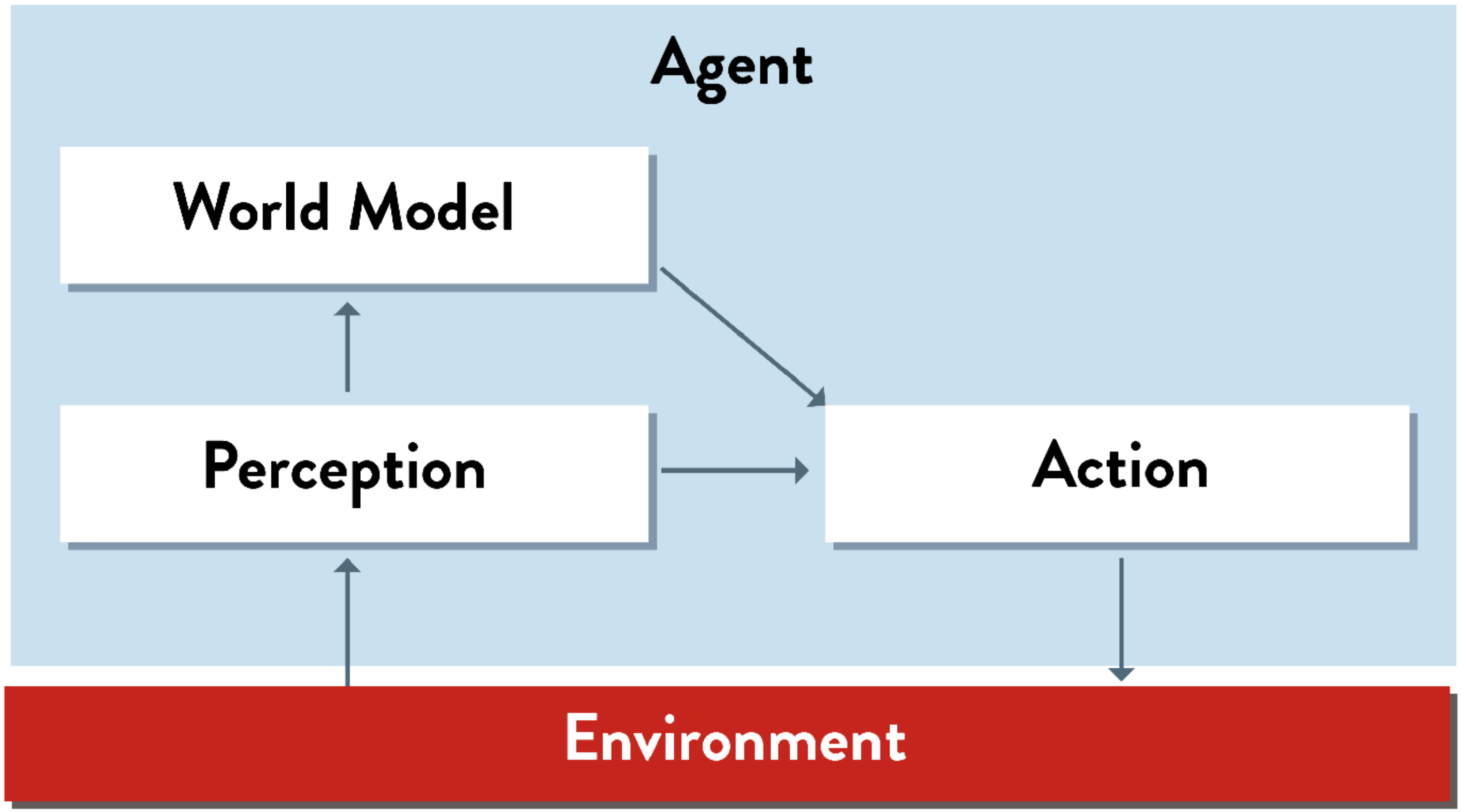



```
def bot(myNum, side) =
{
      var barNum = nearest_bar();

      atBeacon(1, side) ->
            {
                  say("At beacon");
                  turnTo(200, 250);
                  stop();
            }
|
      holding() and obstructedBeacon(1, side) -> find_opening(200)
|
      holding() -> gotoBeacon(1, side)
|
      gotoBar(barNum) ->
            {
                  grab(barNum);
                  say("Found it");
            }
|     true ->
            {
                  turn(random(-180, 180);
                  move(random(50, 300));
            };
};


def find_opening(dist) =

      obstructed_to(dist) -> turn(random(5, 15))
|
      true -> move(random(0, dist))
;
```

Limitations of Reactive Agents

- Reactive Agents have no memory or “state”
  - unable to base decision on previous observations
  - may repeat the same sequence of actions over and over
  - Escape from infinite loops is (sometimes) possible if the agent can randomise its actions.

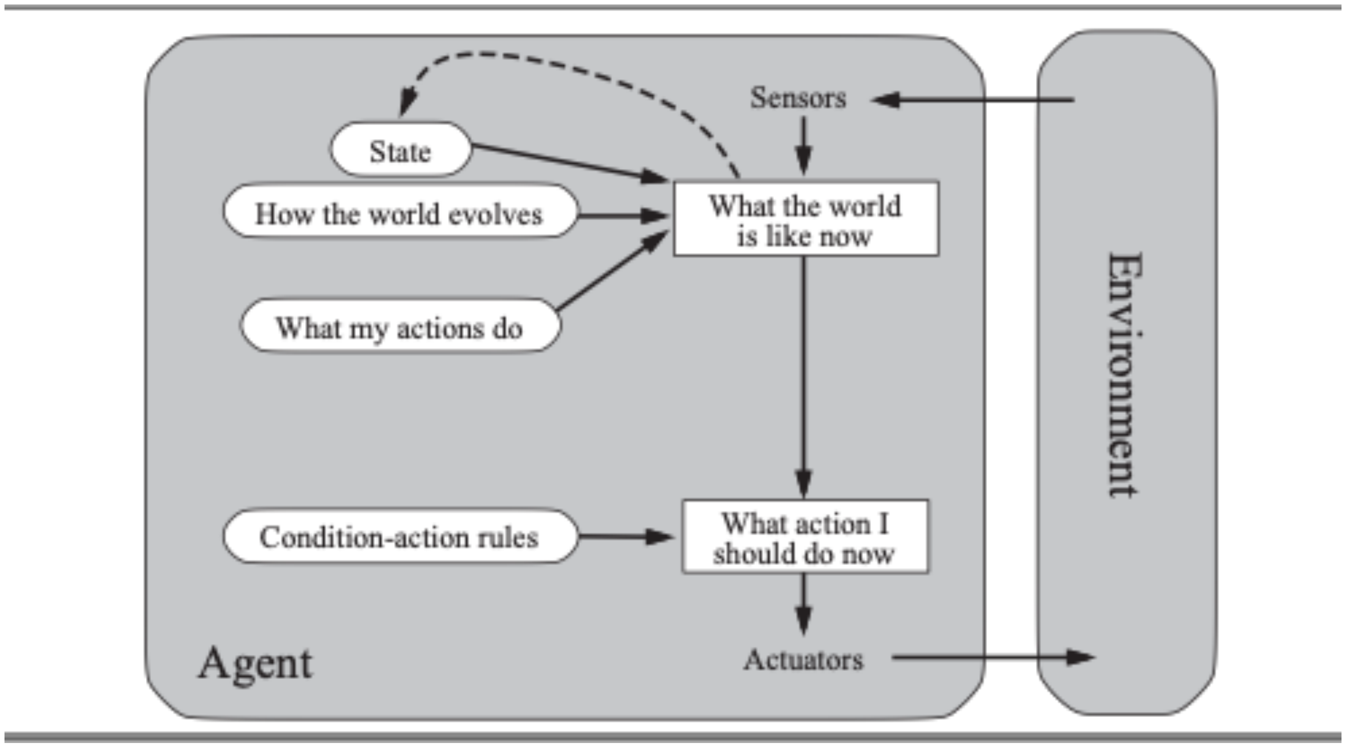
Model-Based Agent



Model-based Agents

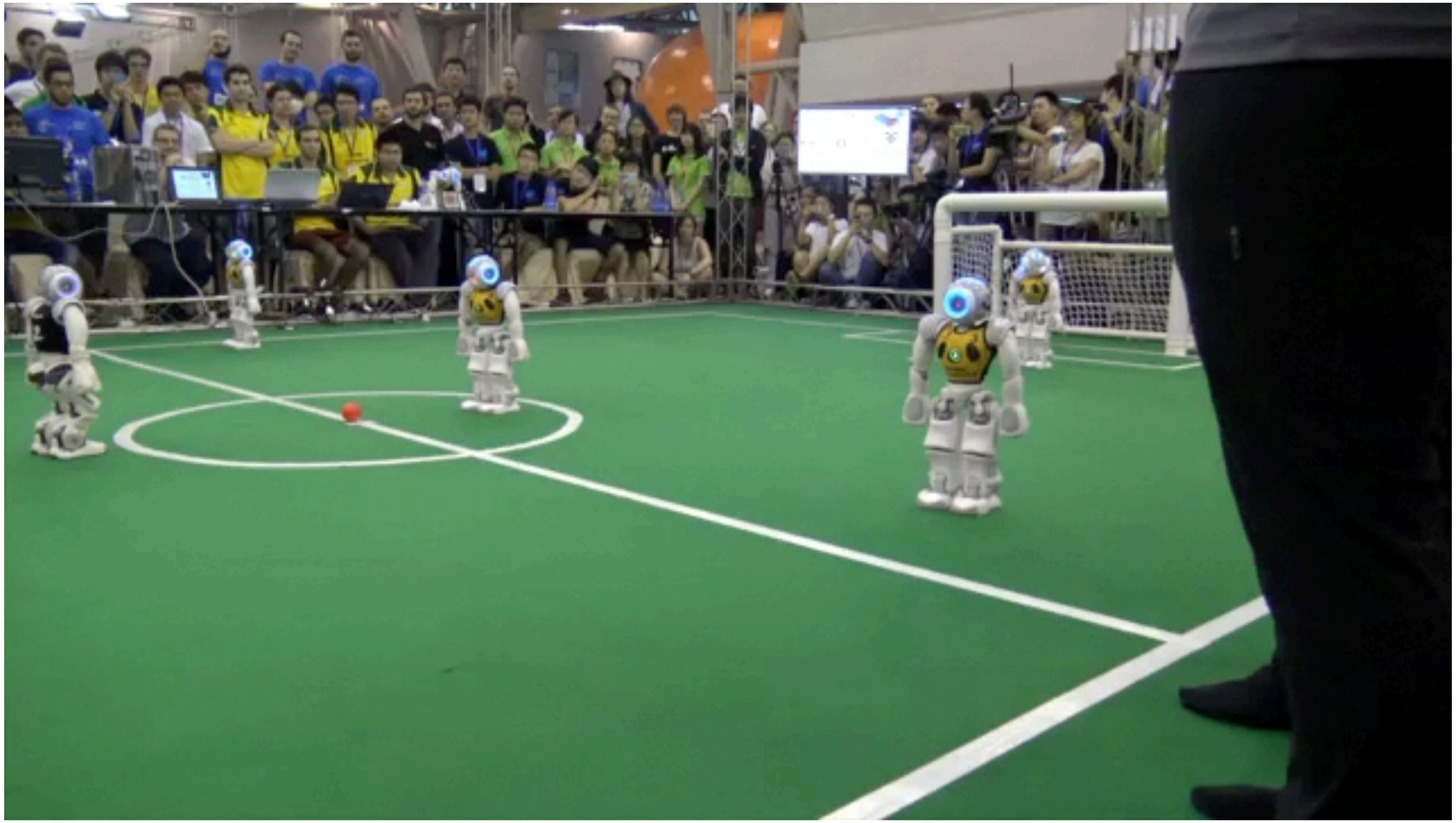
- Handle ***partial observability*** by ***keeping track of the part of the world it can’t see now***.
- Maintain internal state that depends on the percept history and remembers at least some of the unobserved aspects of the current state.
- Knowledge about “how the world works” is called a **model** of the world.
- An agent that uses such a model is called a **model-based agent**.

Model-based Reflex Agent



A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Model-based Reflex Agent

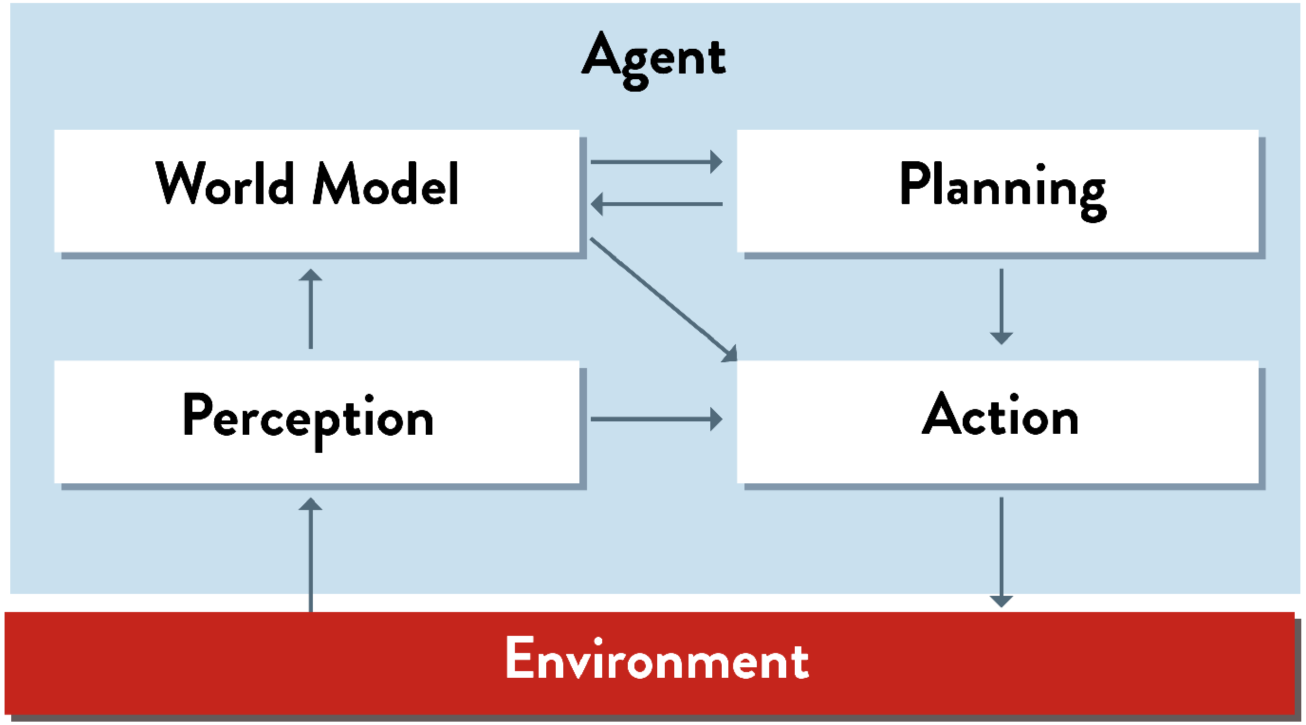


Limitations of Model-Based Agents

- An agent with a world model but no planning can look into the past, but not into the future; it will perform poorly when the task requires any of the following:
- searching several moves ahead
  – Chess, Rubik's cube
- complex tasks requiring many individual step
  – cooking a meal, assembling a watch
- logical reasoning to achieve goals
  – travel to New York

Sometimes we may need to plan several steps into the future

Planning Agent



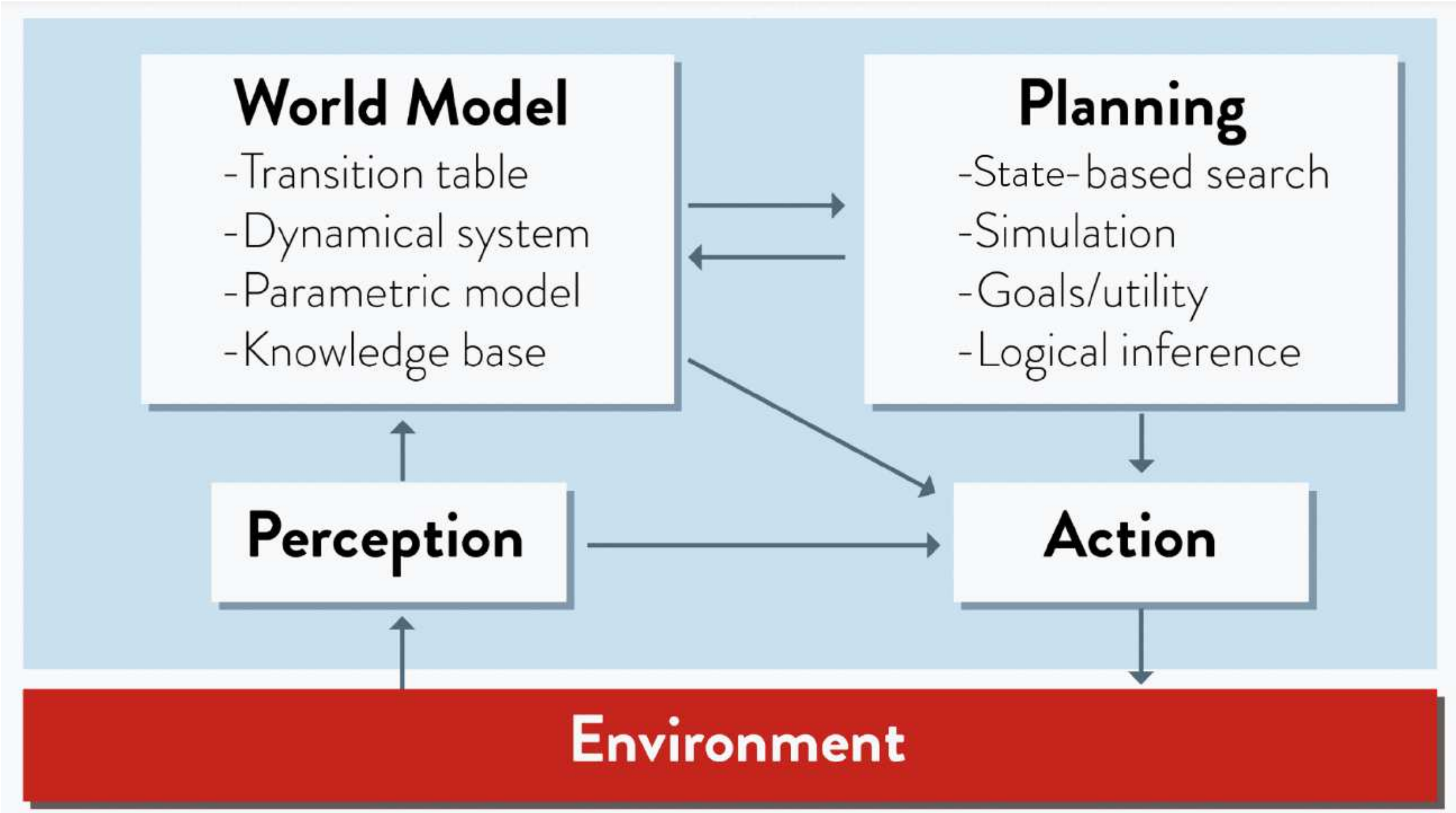
Goal-Based Agent

Planning Agent

- Decision making of this kind is fundamentally different from the condition–action rules
- It involves consideration of the future
  - "What will happen if I do such-and-such?" and
  - "Will that make me happy?"

In the reflex agent designs, this information is not explicitly represented, because the built-in rules map directly from states to actions

Models and Planning



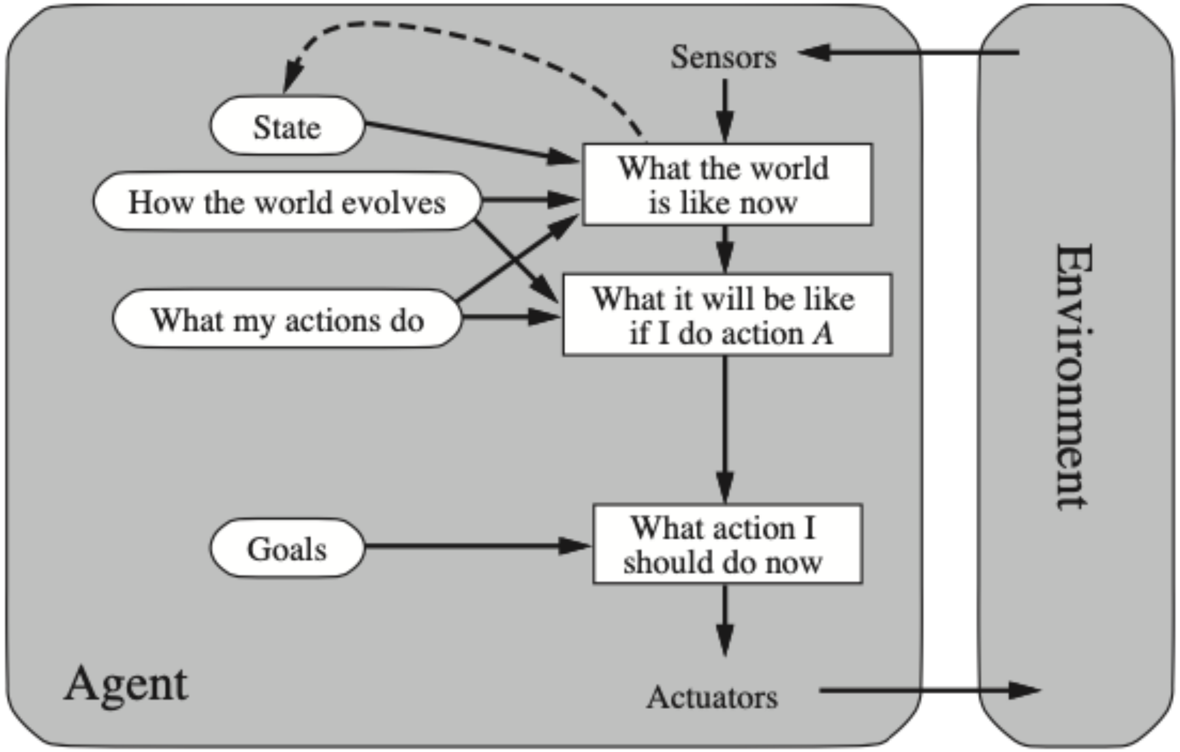
Reasoning about Future States

- What is the best action in this situation?
- Faking it
  - Sometimes an agent may appear to be planning ahead but is actually just applying reactive rules.
  - These rules can be hand-coded, or learned from experience.
  - Agent may not be flexible enough to adapt to new situations.

Planning Agent – Goal-based

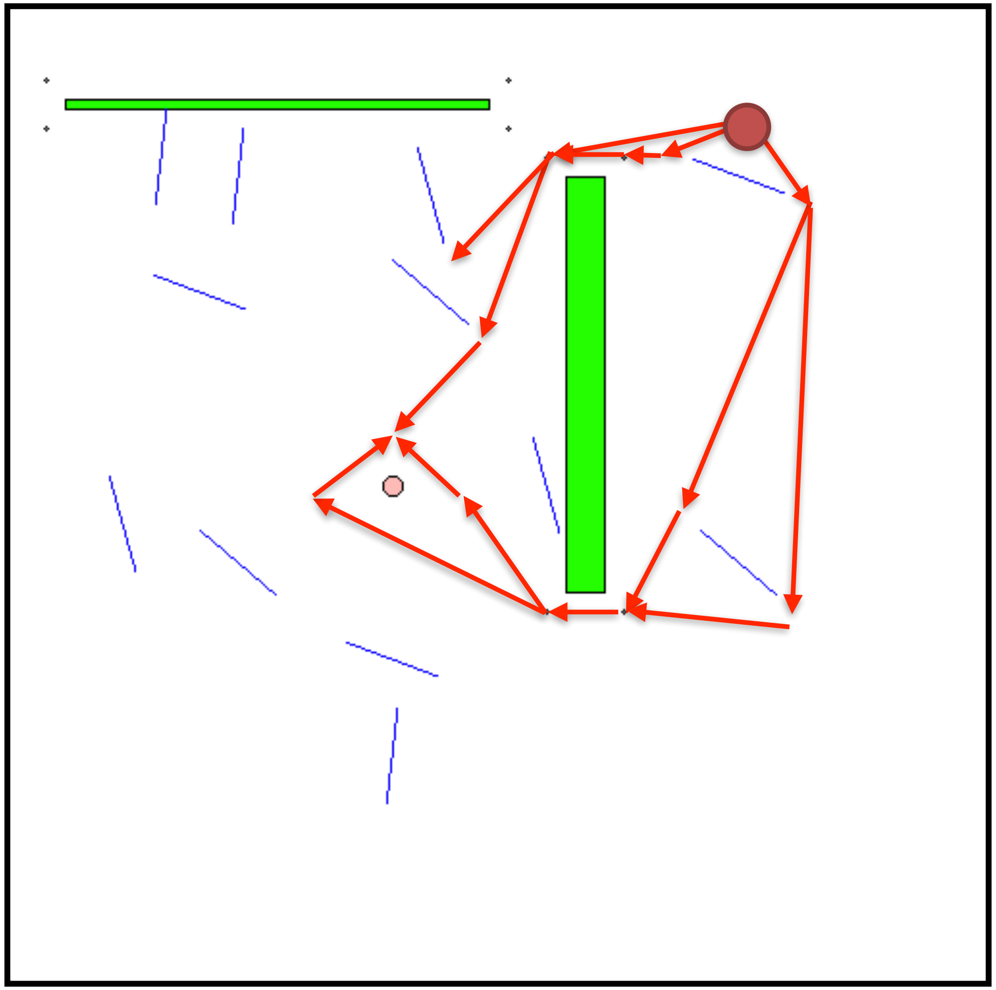
- The planning agent or goal-based agent is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.
- The agent's behaviour can easily be changed.
- But …
  - it's slower to react because it has to "think" about what it's doing.

Goal-based (teleological) agent



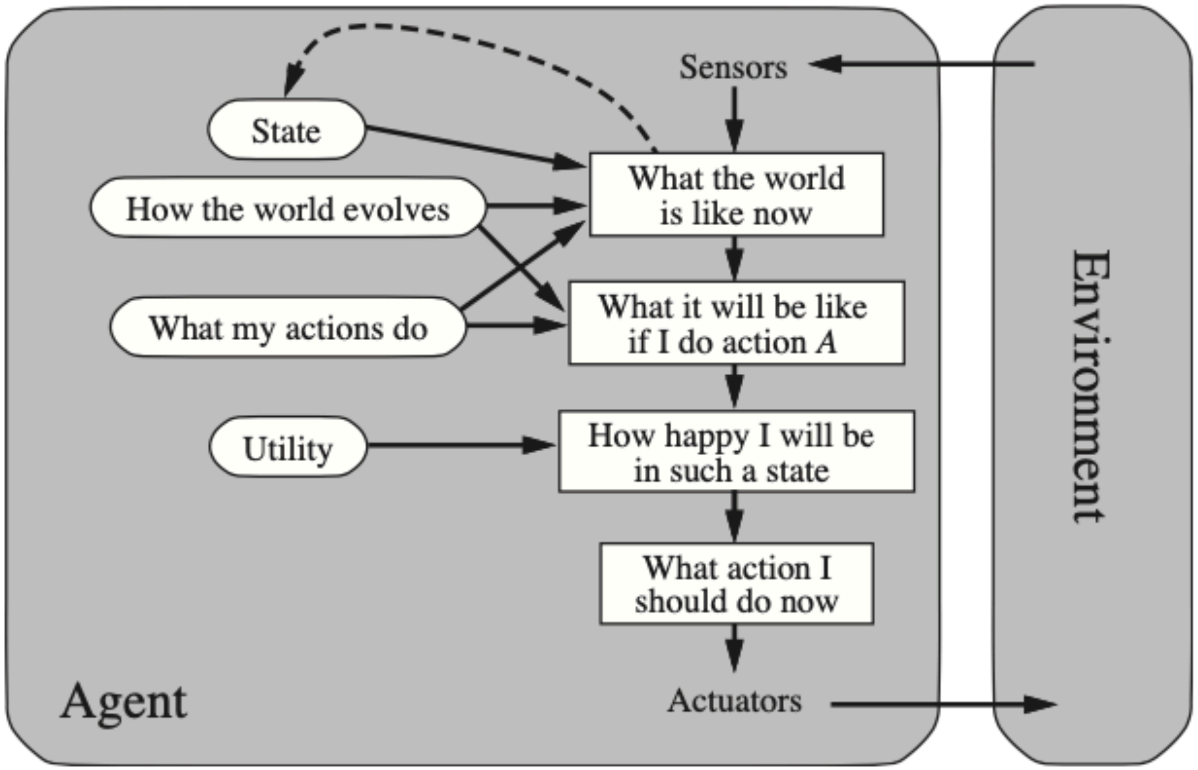
- State description often not sufficient for agent to decide what to do
- Needs to consider its goals (may involve searching and planning)

Planning usually needs search



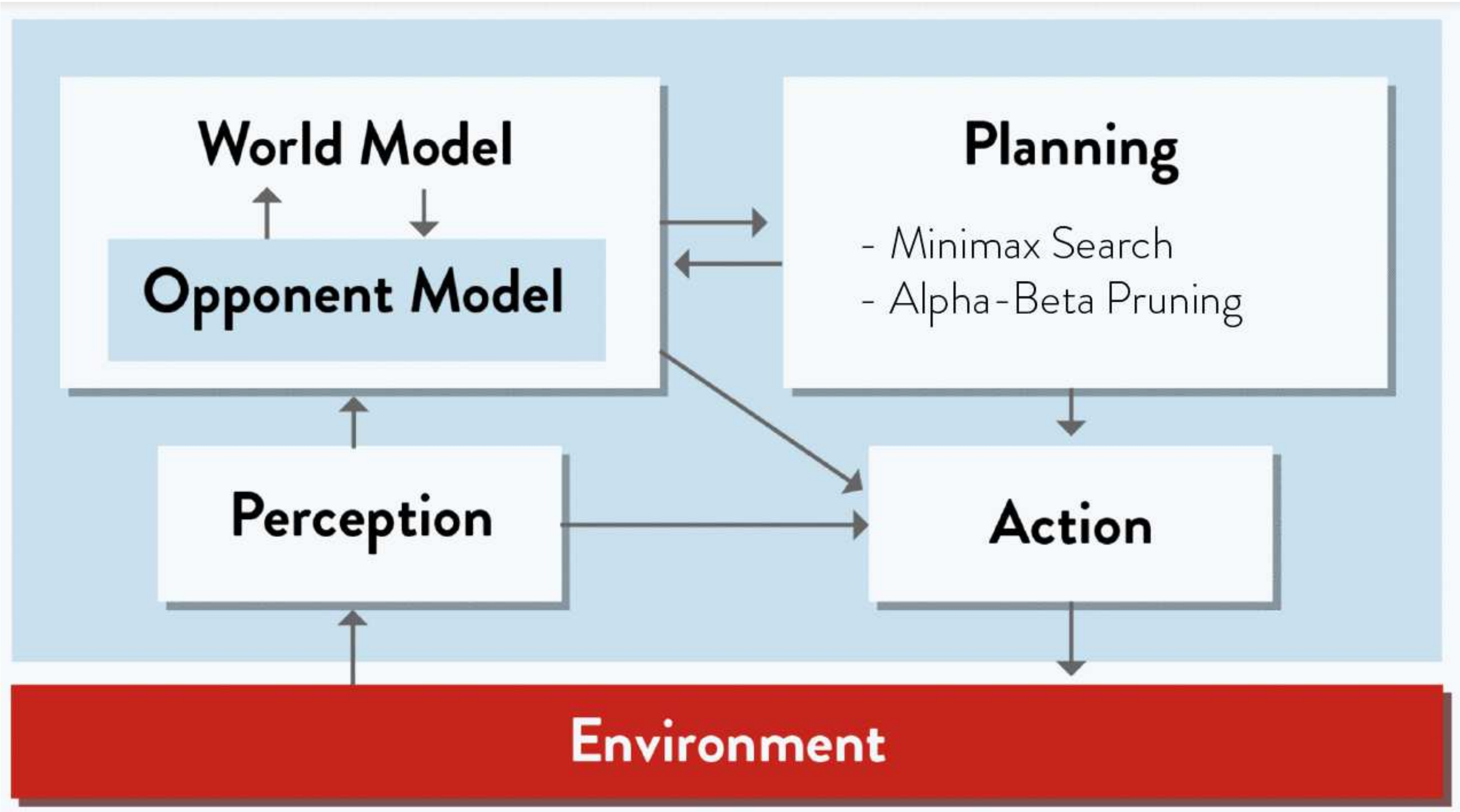
@Home Robot

Utility-based agent

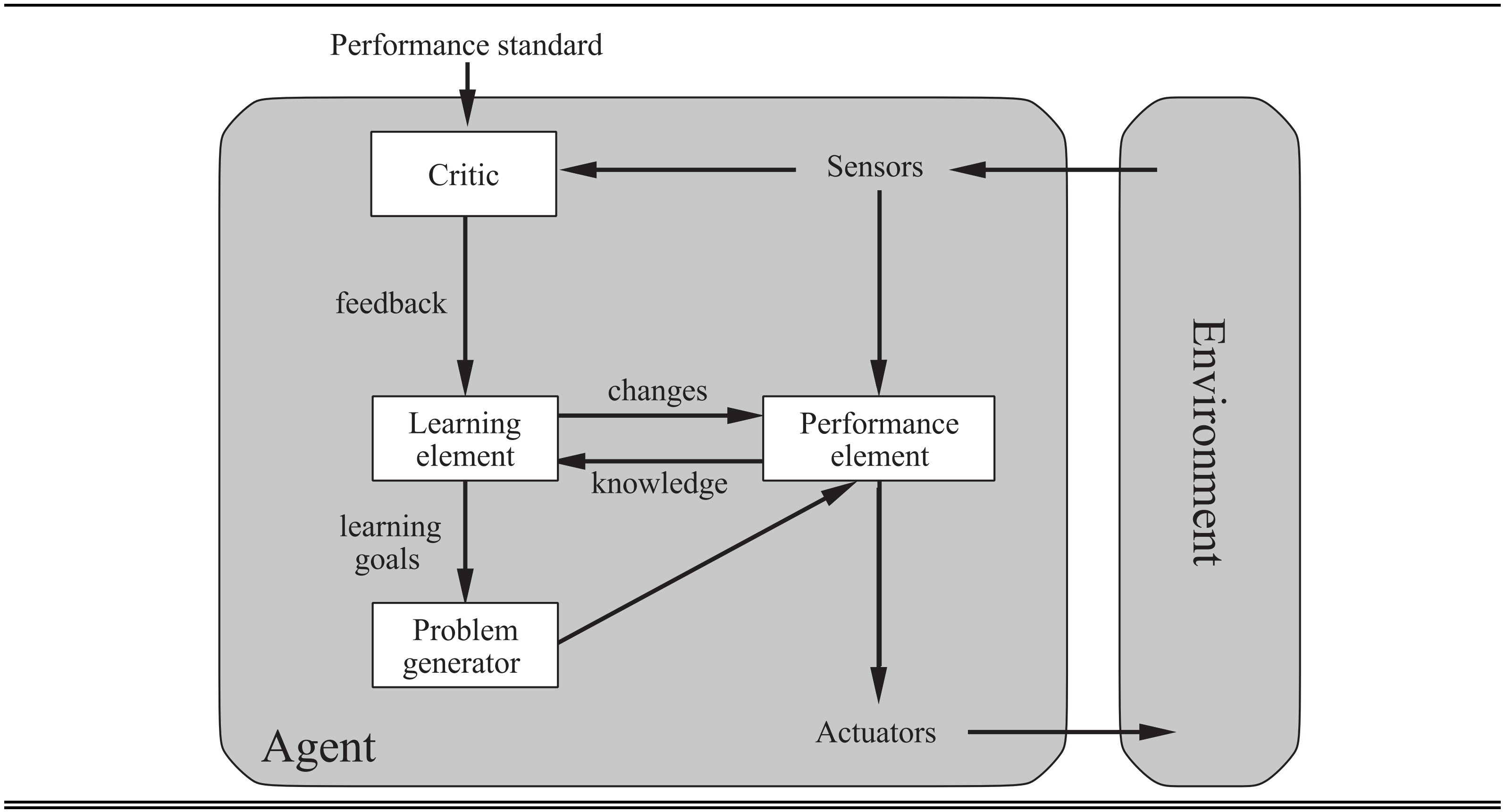


- Model-based, utility-based agent uses
  - model of world
  - utility function that measures preferences among states of world
- Chooses action that leads to best expected utility
  - Expected utility is computed by averaging over all possible outcome states
  - Weighted by probability of outcome.

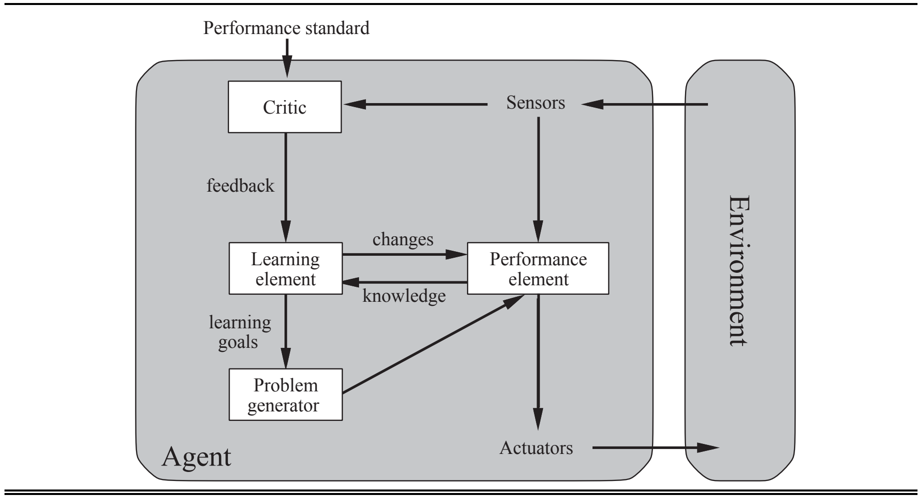
Game Playing Agent



Learning Agent

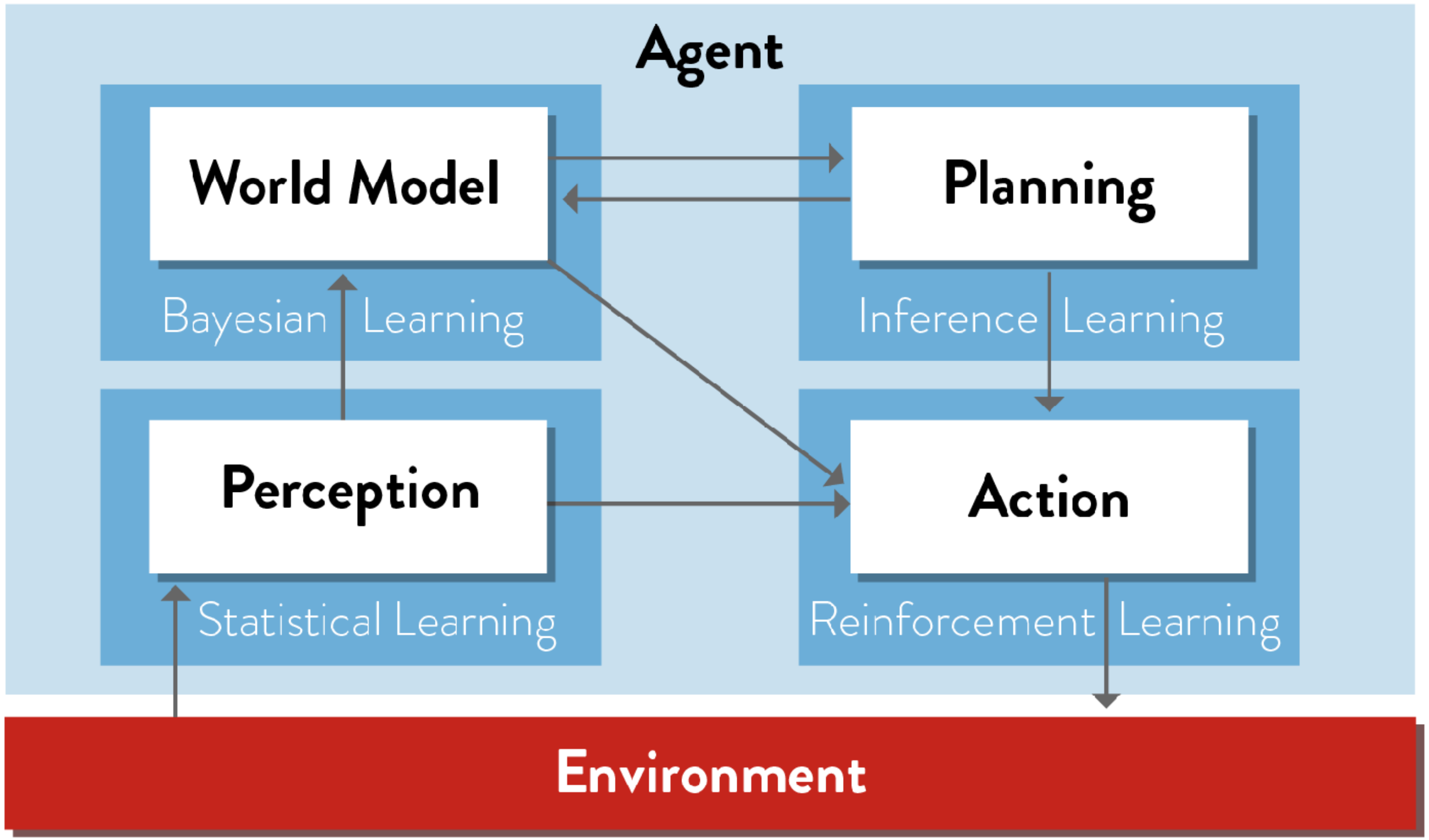


Learning Agent



- **Performance element** takes percepts; decides actions
- **Critic** gives feedback on how performance element is doing
- **Learning element** uses feedback to determine how performance element should be modified to do better in future
- **Problem generator** creates new tasks to provide new and informative experiences.

Learning Agent



Learning

- Learning is not a separate module, but rather a set of techniques for improving the existing modules
- Learning is necessary because:
  - may be difficult or even impossible for a human to design all aspects of the system by hand
  - the agent may need to adapt to new situations without being re-programmed by a human

Summary

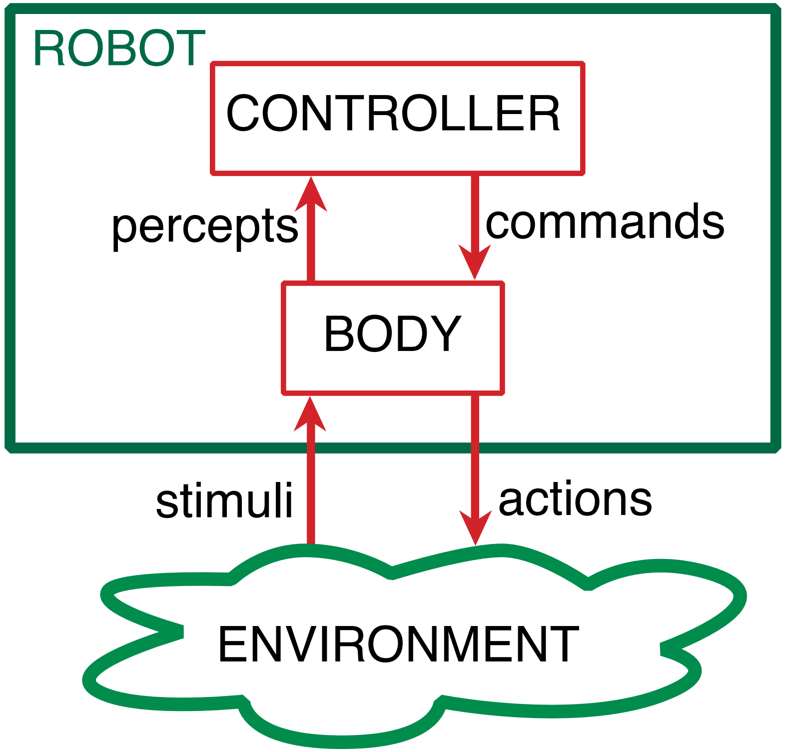
- **Reactive agents** respond directly to percepts
- **Model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept
- **Planning (Goal-based) agents** act to achieve their goals
- **Utility-based agents** try to maximise expected “happiness.”
- All agents can improve their performance through learning.

Representation and Search

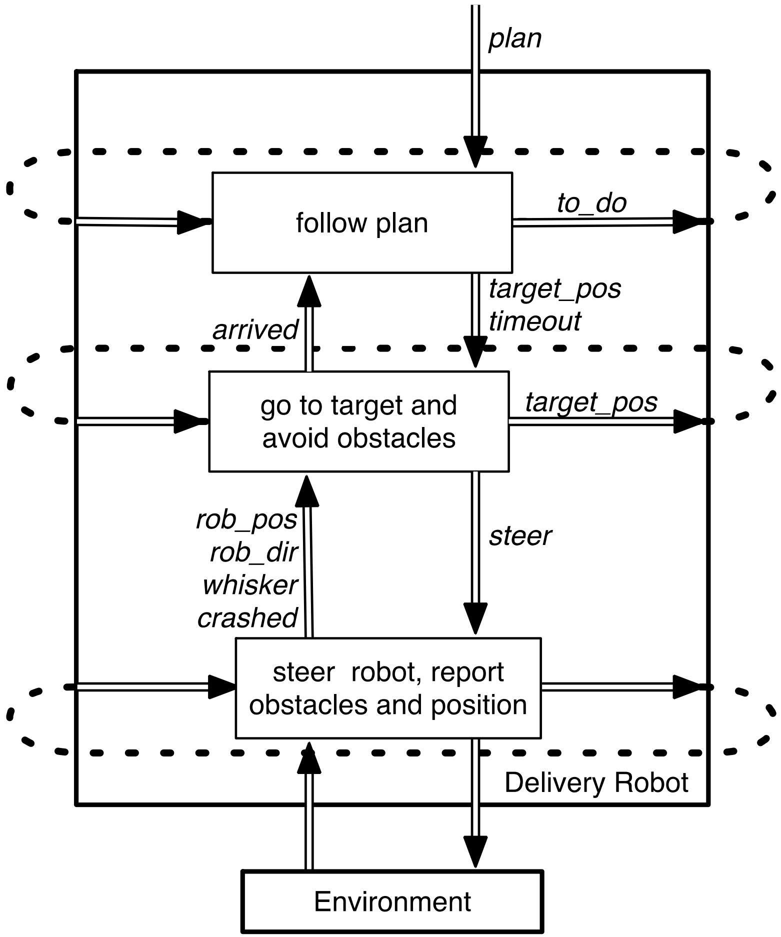
- The world model must be represented in a way that makes reasoning easy.
- Reasoning (problem solving and planning) in AI almost always involves some kind of search amongst possible solutions.

Layered Architecture

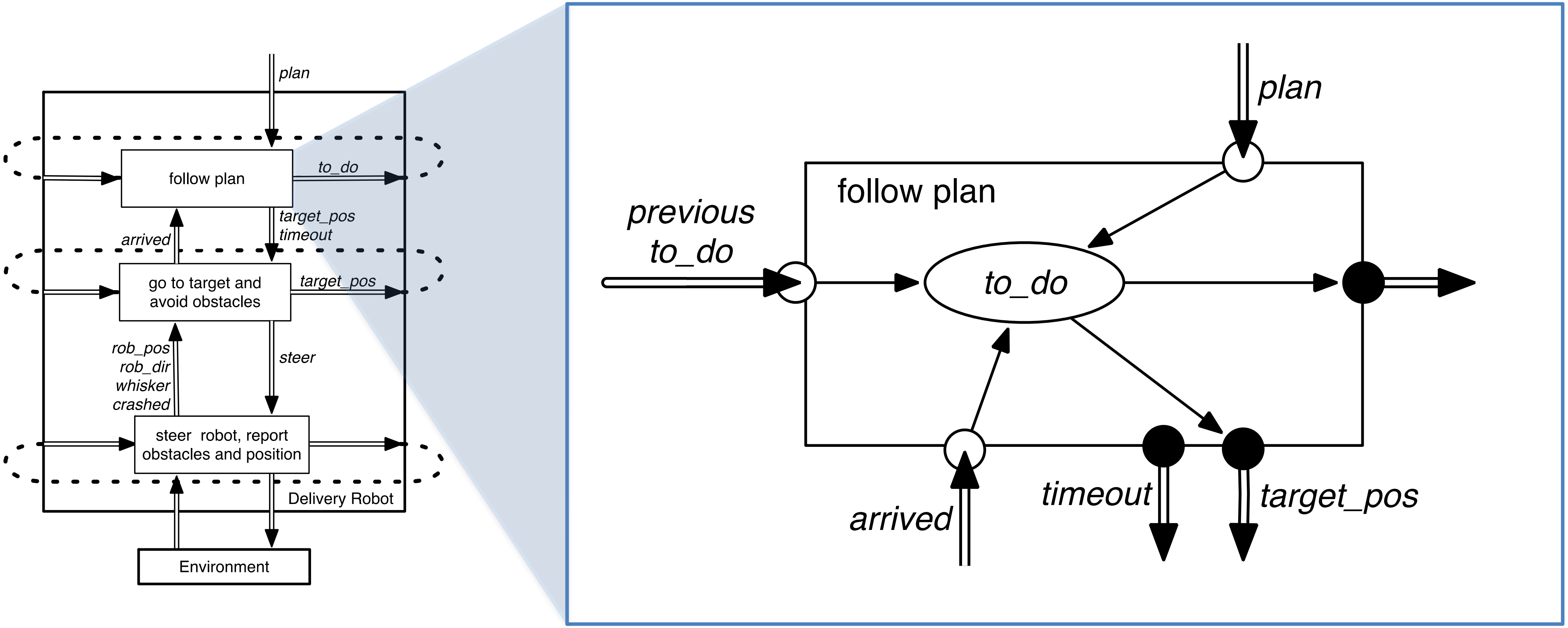
- Hierarchy of controllers
- Controller gets percepts from and sends commands to the lower layer
  - Abstracts low level features into higher level (perception)
  - Translates high level commands into actuator instructions (action)
- Controllers have different representations, programs
- Controllers operate at different time scales
- Lower-level controller can override its commands



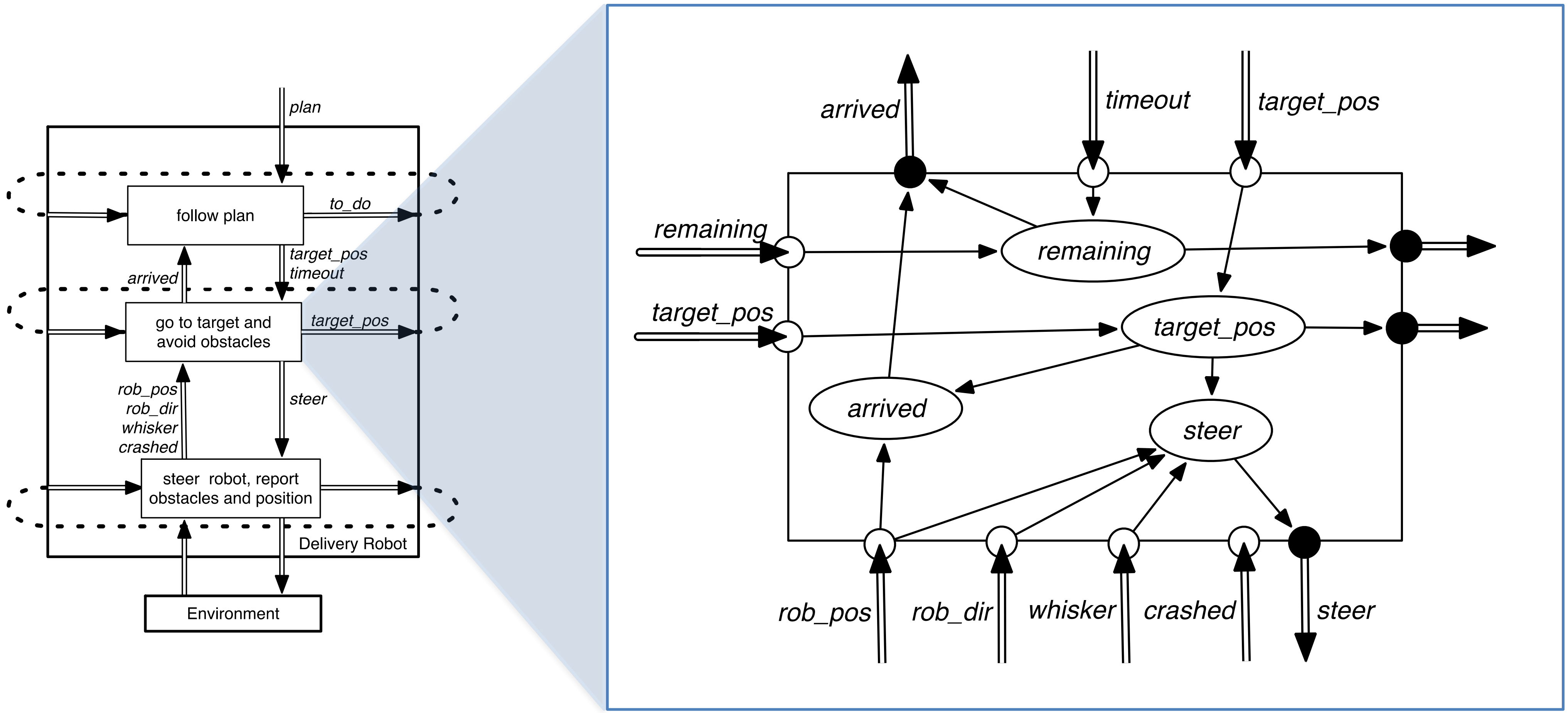
Example – Delivery Robot



Delivery Robot – Top Layer



Delivery Robot – Middle Layer



Delivery Robot – TR Code Example

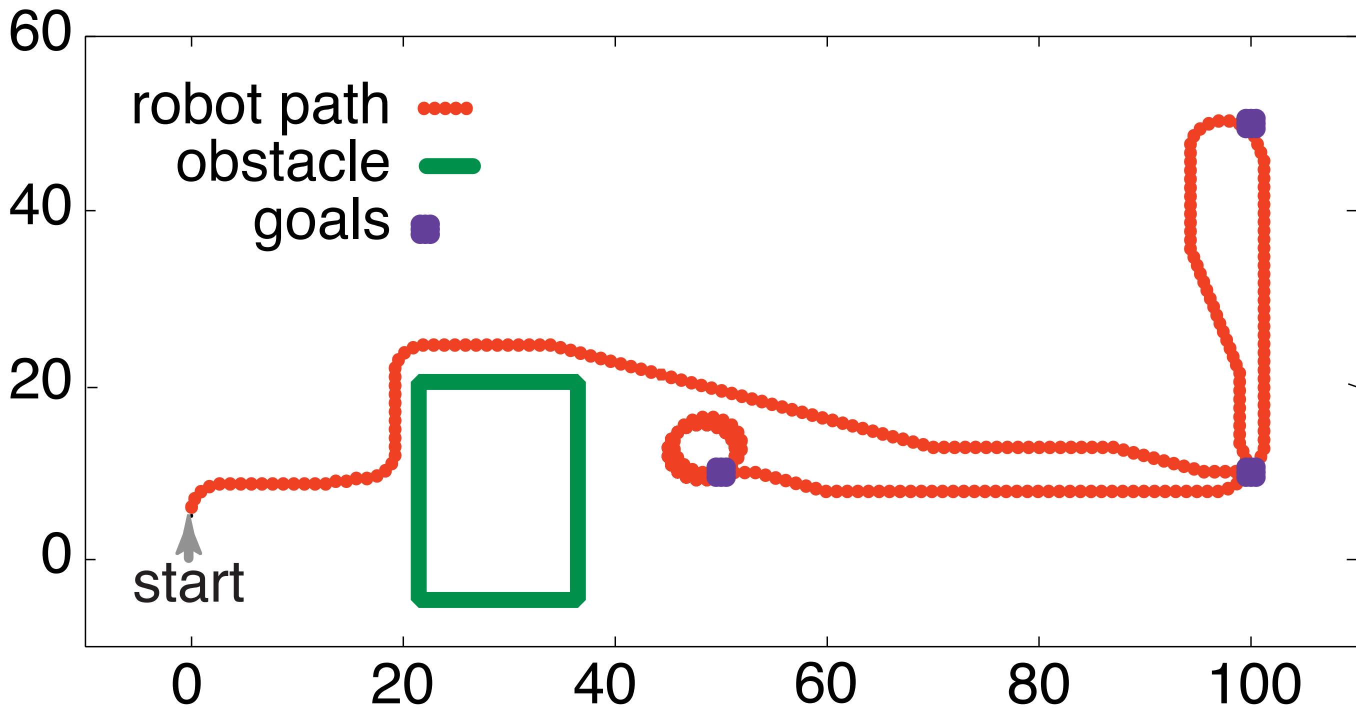
followPlan(to_do):

```
empty(to_do) → stop;

arrived() or timeout() →
{
	resetTimer(200);
	plan := rest(to_do);
}

true → goToTarget(coordinates(first(to_do));
```

goToTarget(target_pos):

```
arrived() or timeout() → {set arrived; stop;}

whisker_sensor = on → steer left;

straight_ahead(rob_pos, robot_dir, target_pos) → steer(straight);

left_of(rob_pos, robot_dir, target_pos) → steer(left);

true  → steer(right)
```

Delivery Robot – Simulation



References

- Poole &Mackworth, Artificial Intelligence: Foundations of Computational Agents, Chapter 1 & 2

- Russell & Norvig, *Artificial Intelligence: a Modern Approach*, Chapter 2.