The Structure of Agents

Four basic types embody the principles underlying almost all intelligent systems:

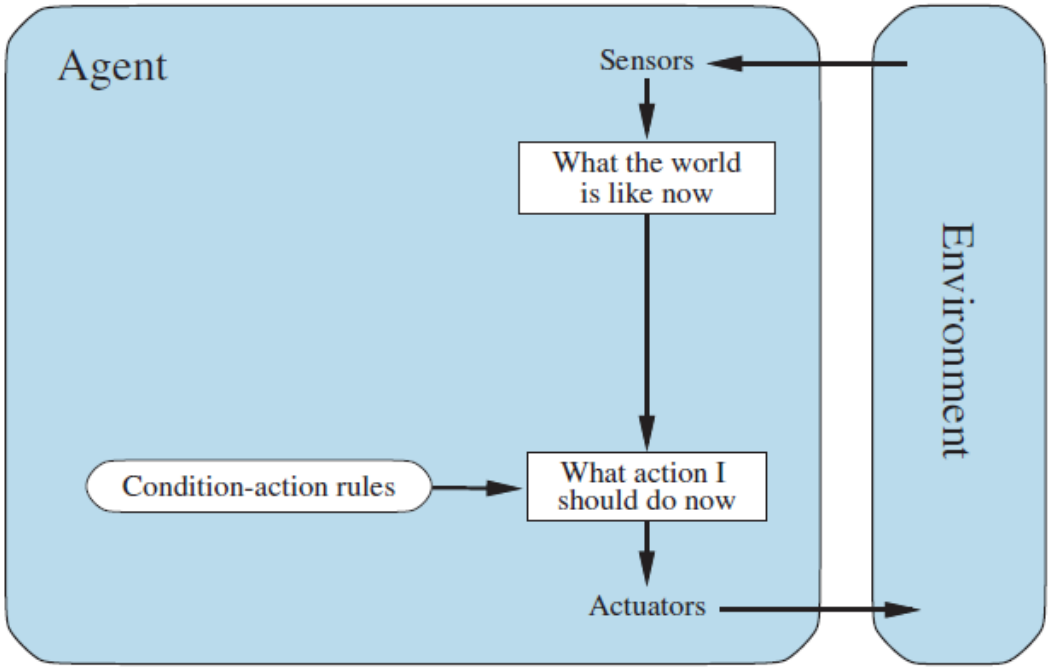
- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

All of them can be converted into

- Learning-based agents

* Figures are from the textbook site (or drawn by the instructor) unless the source is specifically cited.

Simple Reflex Agent



Rectangles: agent's current internal state
Ovals: background information used in the process.

- Select actions based on the current percepts, and ignore the percept history.

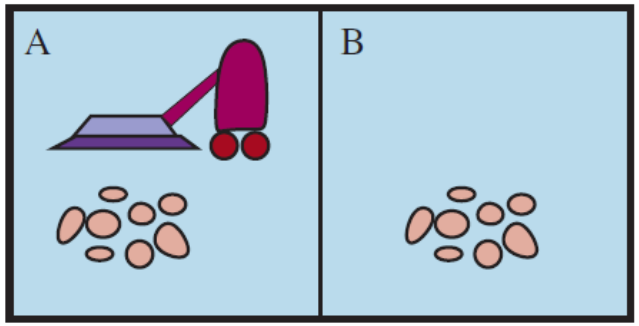
  E.g., the vacuum agent

- Implemented through *condition-action* rule.

  `if` dirty `then` suck

  `if` car-in-front-is-braking
  `then` initiate-braking

Vacuum-Cleaner World (Revisited)



if *status* == *Dirty* then return *Suck*
else if *location* == *A* then return *Right*
else if *location* == *B* then return *Left*

Simple Reflex Agent

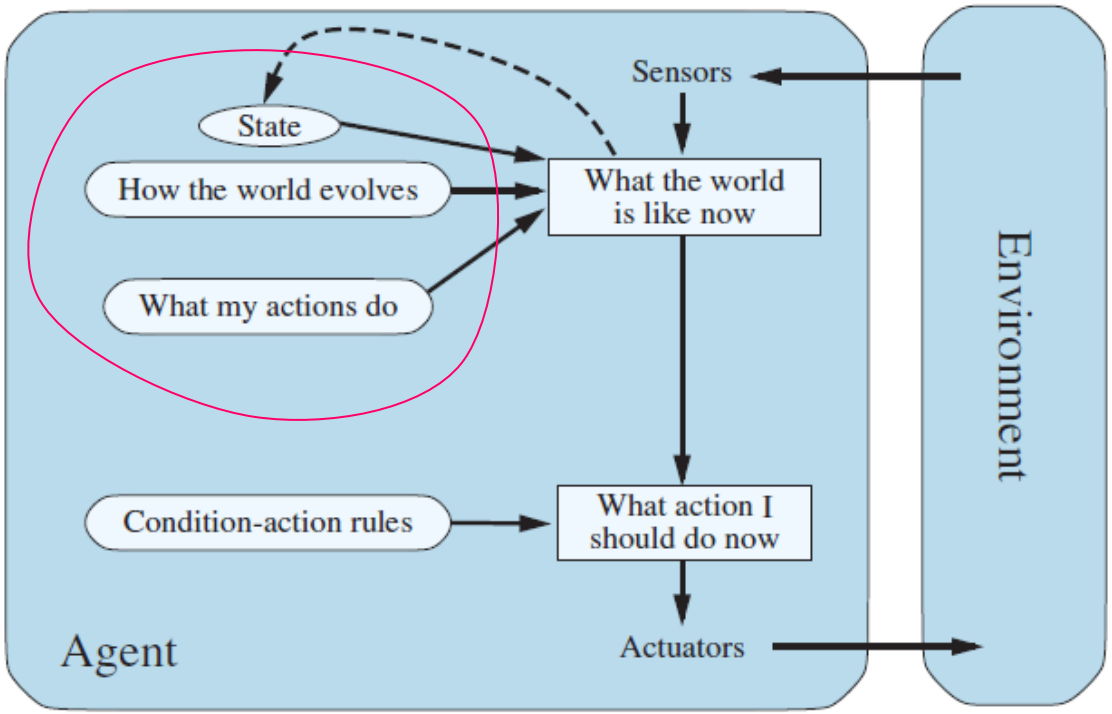
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

- Limited intelligence

  It will work **only if** the correct decision can be made based on only the current percept, i.e., **only if** the environment is fully observable.

Model-Based Reflex Agent



- Partially observable environment.
- Need to maintain some internal state.
- Update it using *knowledge*.

⇩

Model of the world

  - How does the world change?
  - How do actions affect the world?

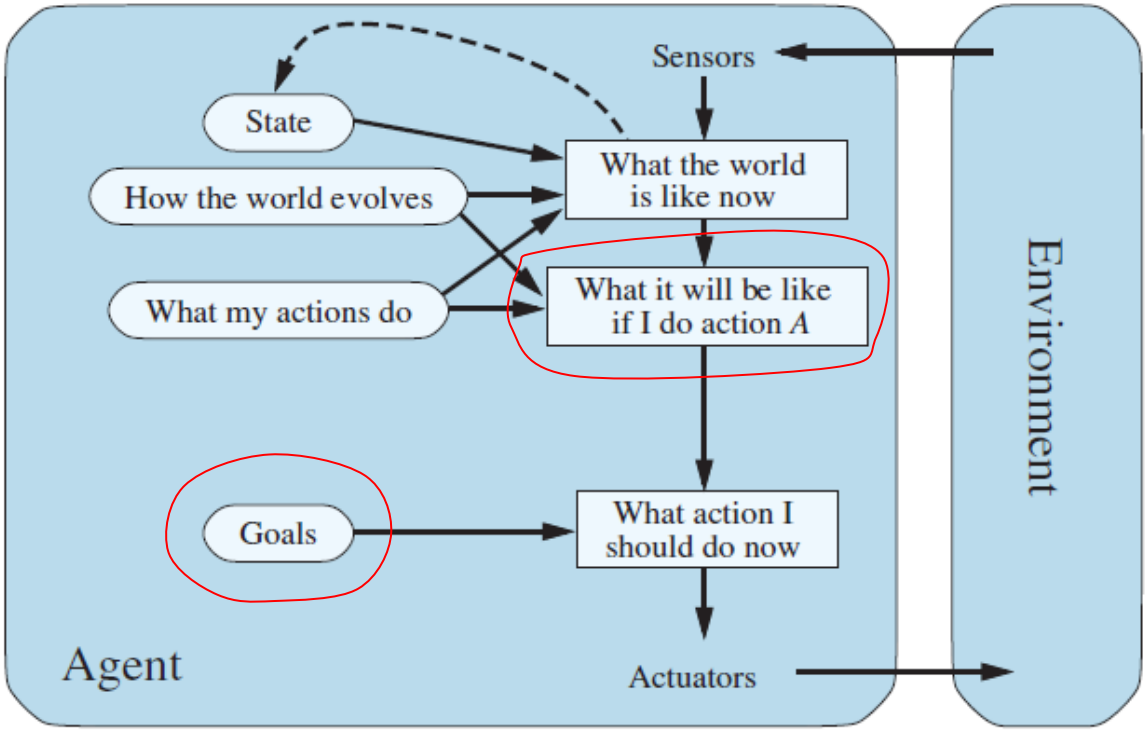
How This Agent Works

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    transition_model, a description of how the next state depends on the current state and action
    sensor_model, a description of how the current world state is reflected in the agent's percepts
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *transition_model*, *sensor_model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

- It is rarely possible to describe the exact current state of the environment.
- The maintained "state" does not have to describe the world.

Goal-Based Agent

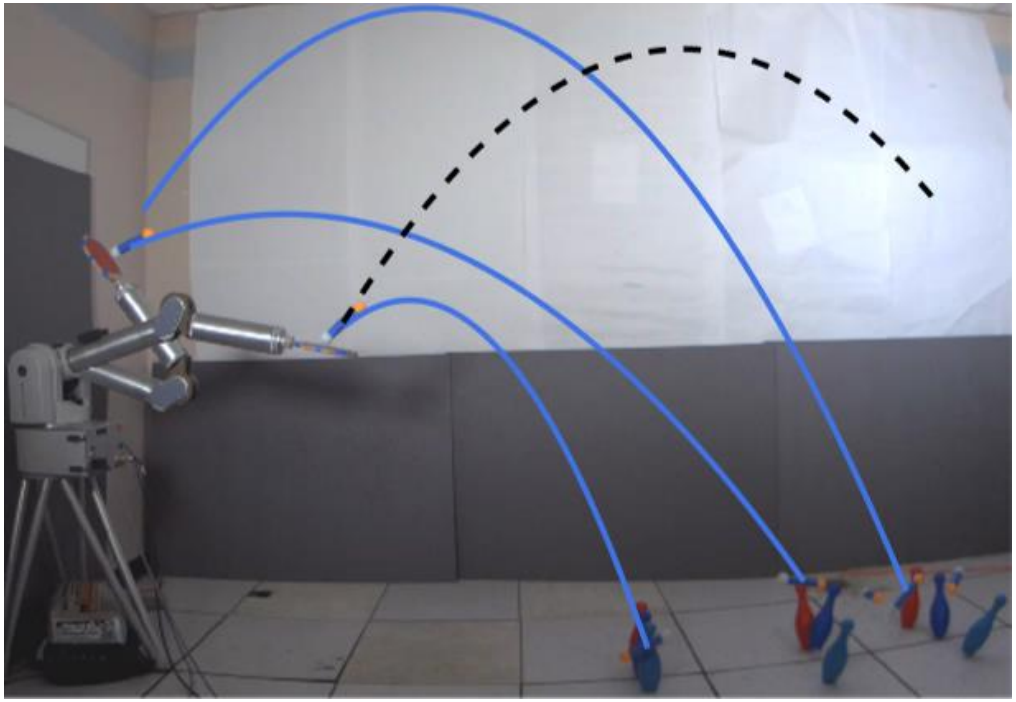


- Needs also some goal information describing desirable situations.
- *Search* and *planning*

  when a long sequence of actions is required to find the goal.
- Difference in *taking the future into account*.

Example: Robotic Batting

Goal Hit an in-flight object to a target.

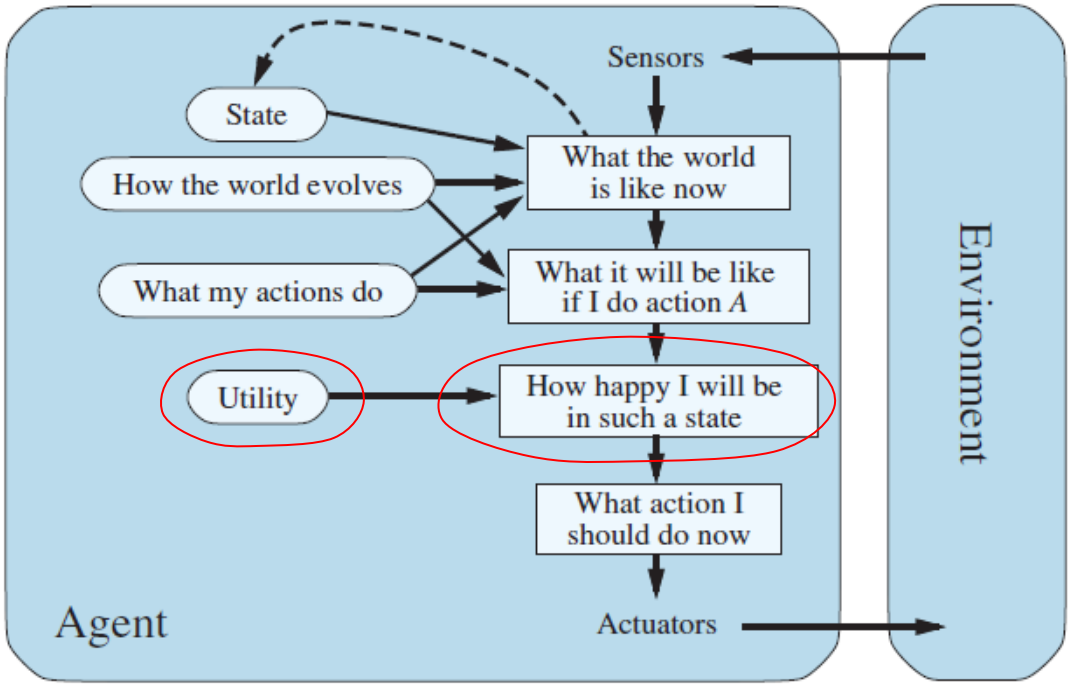


- Sensors: 2 high-speed cameras
- Actuators: 4-DOF WAM Arm
- Models: impact dynamics, aerodynamics & robot kinematics
- State estimation: uses an extended Kalman filter (EKF) to track the velocity & angular velocity of the flying object.
- Planner: determines the time instant of hitting and the bat's position and velocity at this instant.

https://www.youtube.com/watch?v=dGBevZ54E3s

IEEE Transactions on Robotics, vol. 38, no. 5, pp. 3187-3202, 2022

Utility-Based Agent

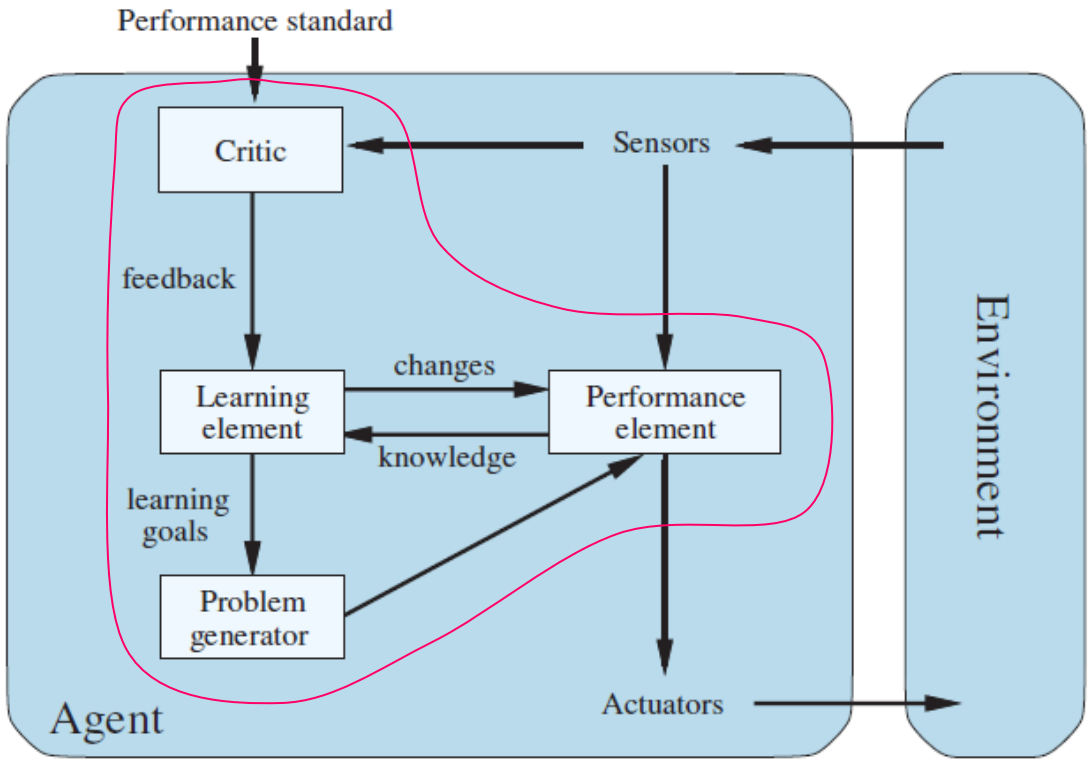


- Different ways to achieve a goal sometimes.
- Use a *utility function* that maps a (sequence of states) to a real number (*utility*)

internal performance measure

- Maximize expected utility.
- Goal improvements:
  - selection among conflicting goals
  - selection based on likelihood of success and goal importance

Learning-Based Agent

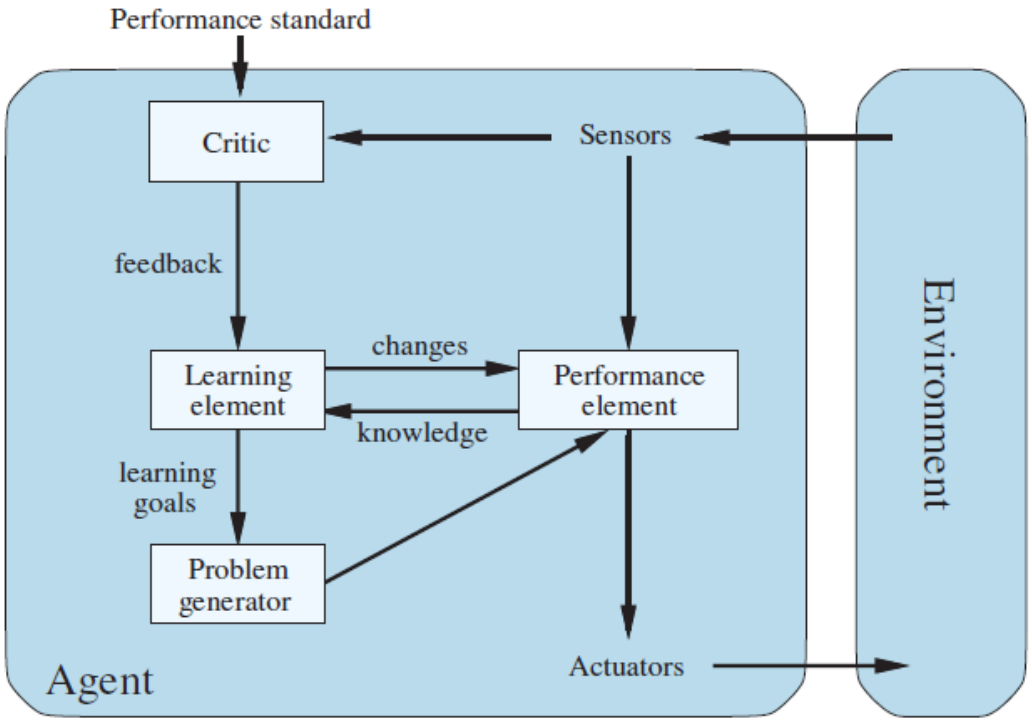


- Preferred method for creating state-of-the-art AI systems:
  - ♦ Allow operation in initially unknown environments.
  - ♦ Adapt to changes in the environment – robustness.
- Modifications of the four components to bring them in closer agreement with the available feedback

⇩

Better overall performance

Learning-Based Agent



- *Critic* provides feedback on the agent's performance based on fixed performance standard.
- *Learning element* introduces improvements in performance element.
- *Performance element* selects actions based on the percepts.
- *Problem generator* suggests actions that will lead to new and informative experiences.