ARTIFICIAL INTELLIGENCE

Russell & Norvig

Chapter 2: Intelligent Agents, part 2

Agent Architecture

- All agents have the same basic structure:
  - accept percepts from environment, generate actions
- Agent = Architecture + Program
- A Skeleton Agent:

```
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
```

- Observations:
  - agent may or may not build percept sequence in memory (depends on domain)
  - performance measure is not part of the agent; it is applied externally to judge the success of the agent

Table-driven architecture

- Why can't we just look up the answers?
  - The disadvantages of this architecture
    - infeasibility (excessive size)
    - lack of adaptiveness
  - How big would the table have to be?
  - Could the agent ever learn from its mistakes?
  - Where should the table come from in the first place?

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*

Agent types

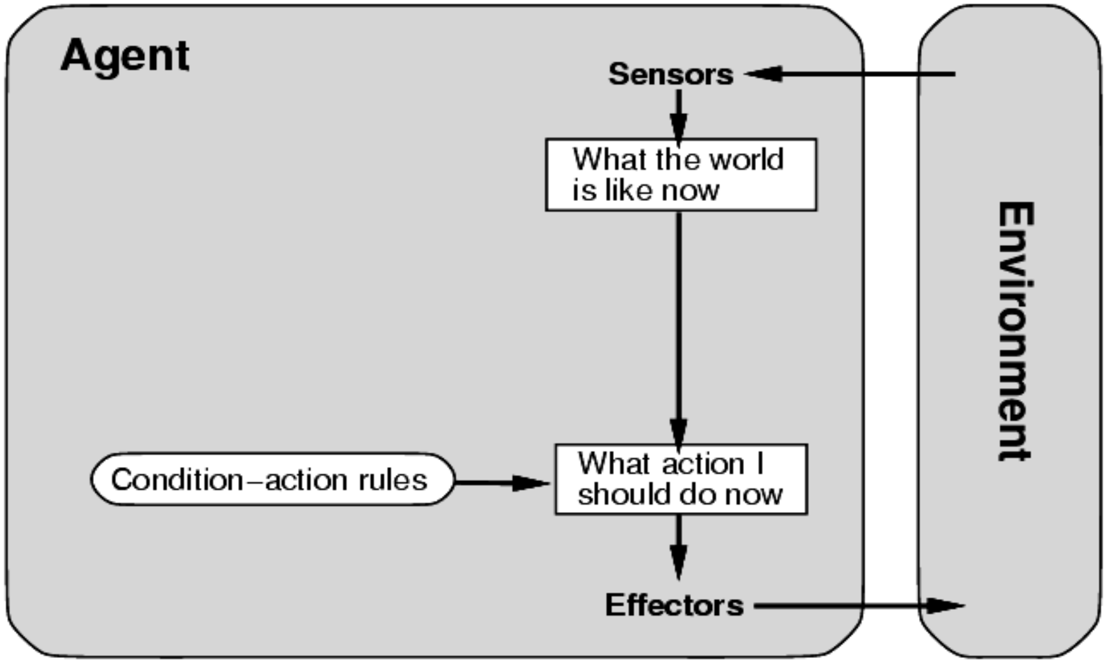
- **Simple reflex agents**
  - are based on condition-action rules and implemented with an appropriate production system. They are stateless devices which do not have memory of past world states
- **Model-based reflex agents (Reflex agent with state)**
  - have internal state which is used to keep track of past states of the world
- **Goal-based agents**
  - are agents which in addition to state information have a kind of goal information which describes desirable situations. Agents of this kind take future events into consideration
- **Utility-based agents**
  - use internal estimate for performance measure to compare future states

A Simple Reflex Agent

- We can summarize part of the table by formulating commonly occurring patterns as condition-action rules:
- Example:

  if *car-in-front-brakes*

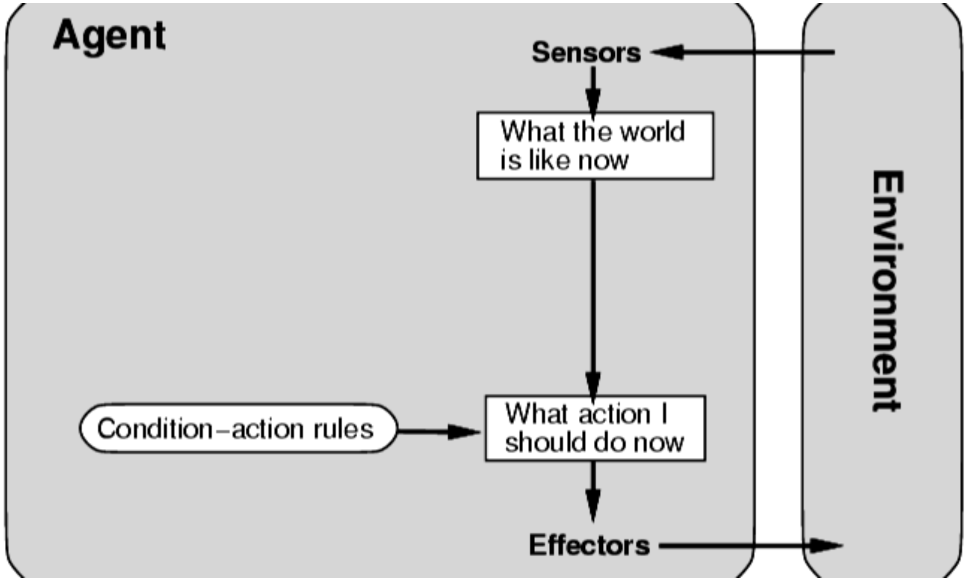
  then *initiate braking*
- Agent works by finding a rule whose condition matches the current situation
  - rule-based systems
- But, this only works if the current percept is sufficient for making the correct decision



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*

Example: Reflex Vacuum Agent

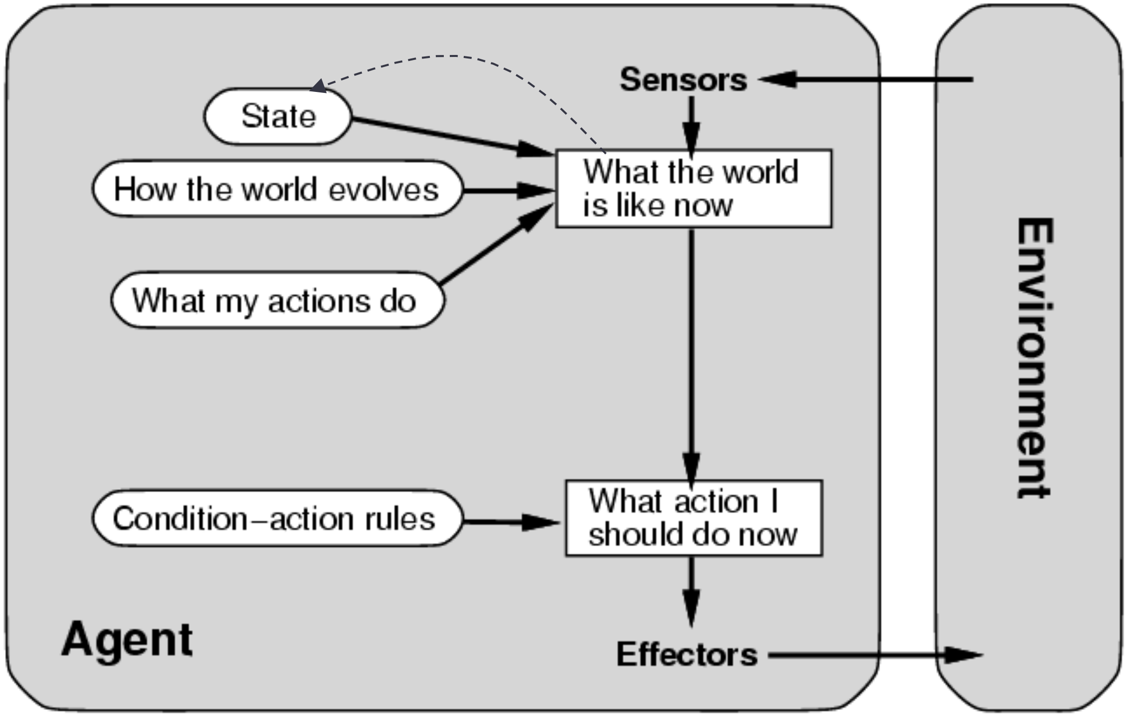


function REFLEX-VACUUM-AGENT([*location*,*status*]) returns an action

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

Model-Based Reflex Agent

- 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

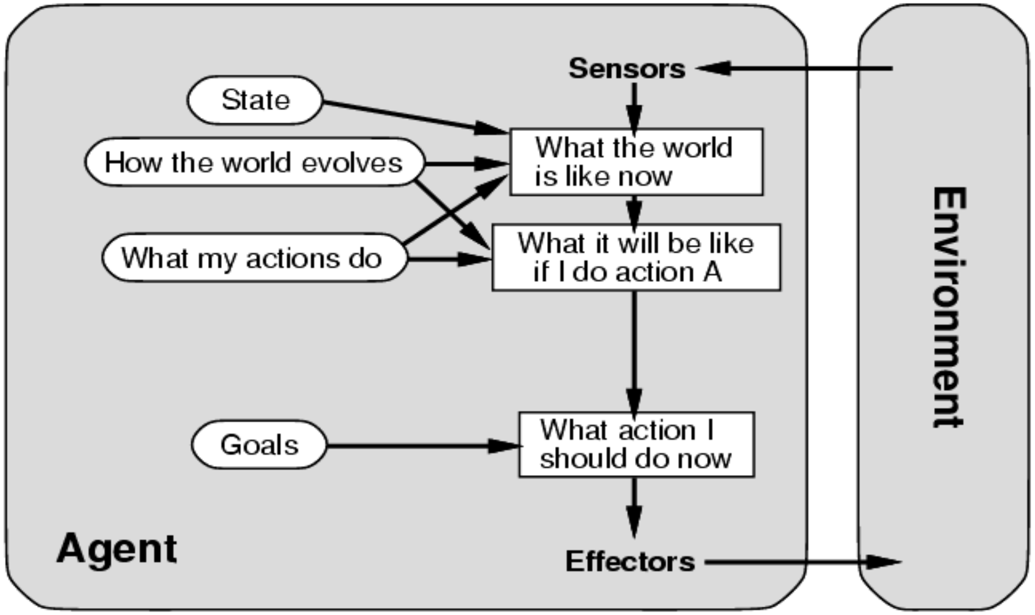


function Reflex-Agent-With-State(*percept*) **returns** action
static: *rules*, a set of condition-action rules
state, a description of the current world

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

Goal-Based Agents

- **Reasoning about actions**
  - Reflex agents only act based on pre-computed knowledge (rules)
  - Goal-based (planning) agents act by reasoning about which actions achieve the goal
  - Less efficient, but more adaptive and flexible

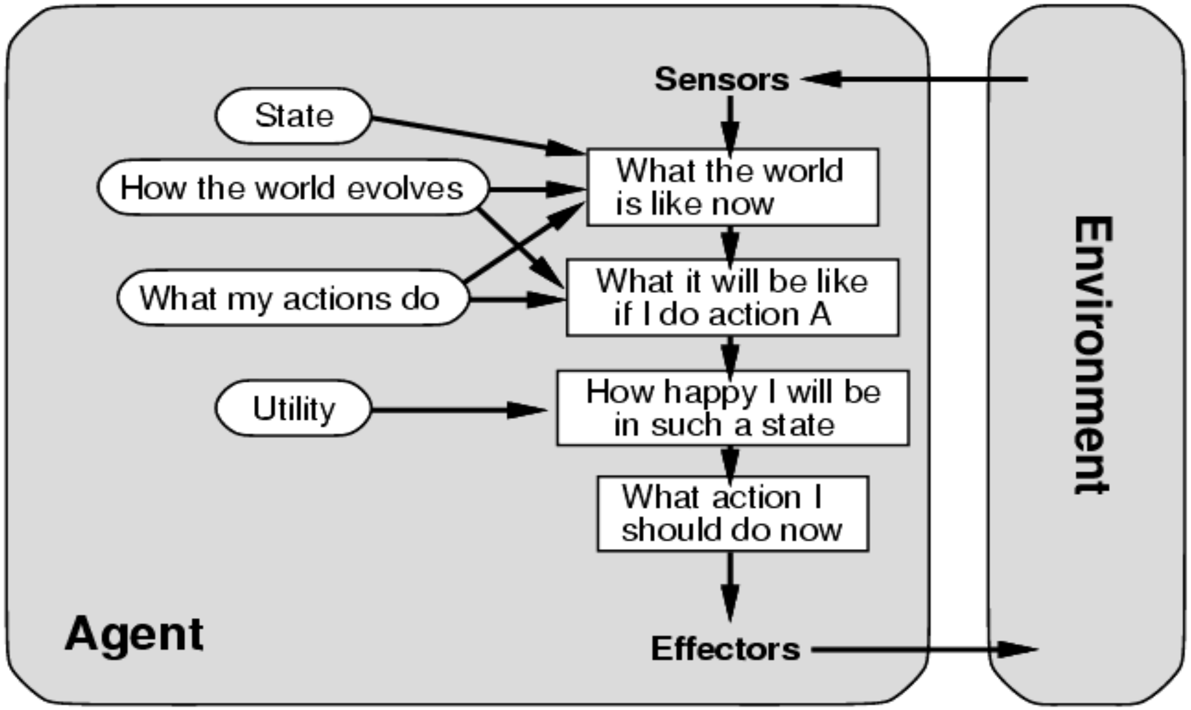


Goal-Based Agents (continued)

- **Knowing current state is not always enough**
  - State allows agent to keep track of unseen parts of world
  - Agent must update state based on changes and its actions
- **Choose between potential states using goal**
  - Can change goal without need to “reprogram” rules, for example a new destination for the taxi-driving agent
- **Search and planning** (coming soon)
  - concerned with finding sequences of actions to satisfy a goal.
  - contrast with condition-action rules: involves consideration of future "what will happen if I do ..." (fundamental difference).

Utility-Based Agent

- Utility Function
  - A mapping of states onto real numbers
  - Allows rational decisions in two kinds of situations
    - Evaluation of the tradeoffs among conflicting goals
    - Evaluation of competing goals



Utility-Based Agents (continued)

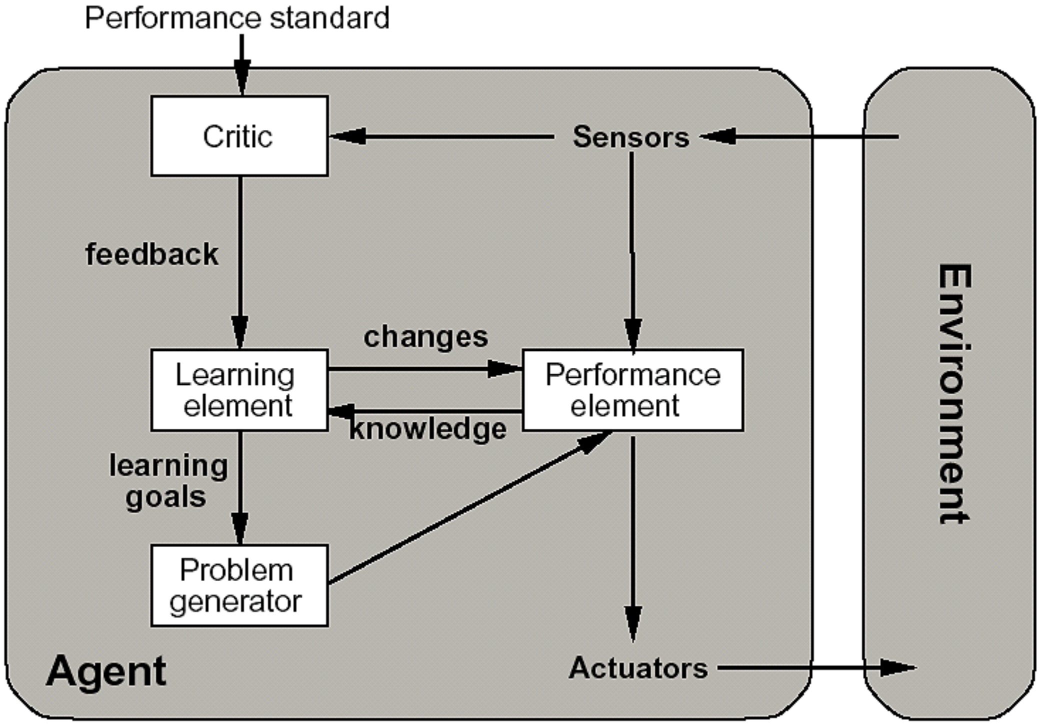
- Preferred world state has higher utility for agent
- Examples:
  - Quicker, safer, more reliable ways to get to destination
  - Price comparison shopping
  - Bidding on items in an auction
  - Evaluating bids in an auction
- Utility function: U(state) gives measure of “happiness”
- Commonly: search is goal-based and games are utility-based.

Shopping Agent Example

- **Navigating: Move around store; avoid obstacles**
  - Reflex agent: store map precompiled.
  - Goal-based agent: create an internal map, reason explicitly about it, use signs and adapt to changes (e.g., specials at the ends of aisles).
- **Gathering: Find and put into cart groceries it wants, need to induce objects from percepts**
  - Reflex agent: wander and grab items that look good.
  - Goal-based agent: shopping list.
- **Menu-planning: Generate shopping list, modify list if store is out of some item**
  - Goal-based agent: required; what happens when a needed item is not there? Achieve the goal some other way. e.g., no milk cartons: get canned milk or powdered milk.
- **Choosing among alternative brands**
  - utility-based agent: trade off quality for price.

Learning Agents

- **Four main components**
  - Performance element: the agent function
  - Learning element: responsible for making improvements by observing performance
  - Critic: gives feedback to learning element by measuring agent's performance
  - Problem generator: suggest other possible courses of actions (exploration)



Search and Knowledge Representation

- **Goal-based and utility-based agents require representation of:**
  - states within the environment
  - actions and effects (effect of an action is transition from the current state to another state)
  - goals
  - utilities
- **Problems can often be formulated as a search problem**
  - to satisfy a goal, agent must find a sequence of actions (a path in the state-space graph) from the starting state to a goal state.
- **To do this efficiently, agents must have the ability to reason with their knowledge about the world and the problem domain**
  - which path to follow (which action to choose from) next
  - how to determine if a goal state is reached OR how decide if a satisfactory state has been reached.

Intelligent Agent Summary

- An **agent** perceives and acts in an environment. It has an architecture and is implemented by a program.
- An **ideal agent** always chooses the action which maximizes its expected performance, given the percept sequence received 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 a percept to an 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.