
Agents and Environment

Perspectives on AI

Based on
Thought-process and
Reasoning

Based on Behaviour

Relative to human performance	Relative to ideal performance (rationality) [mathematical formulation]
Thinking Humanly “The exciting new effort to make computers think . . . <i>machines with minds</i> , in the full and literal sense.” (Haugeland, 1985) “[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . .” (Bellman, 1978)	Thinking Rationally “The study of mental faculties through the use of computational models.” (Charniak and McDermott, 1985) “The study of the computations that make it possible to perceive, reason, and act.” (Winston, 1992)
Acting Humanly “The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990) “The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)	Acting Rationally “Computational Intelligence is the study of the design of intelligent agents.” (Poole <i>et al.</i> , 1998) “AI . . . is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)
Figure 1.1 Some definitions of artificial intelligence, organized into four categories.	

Journey of AI: Challenges & Opportunities

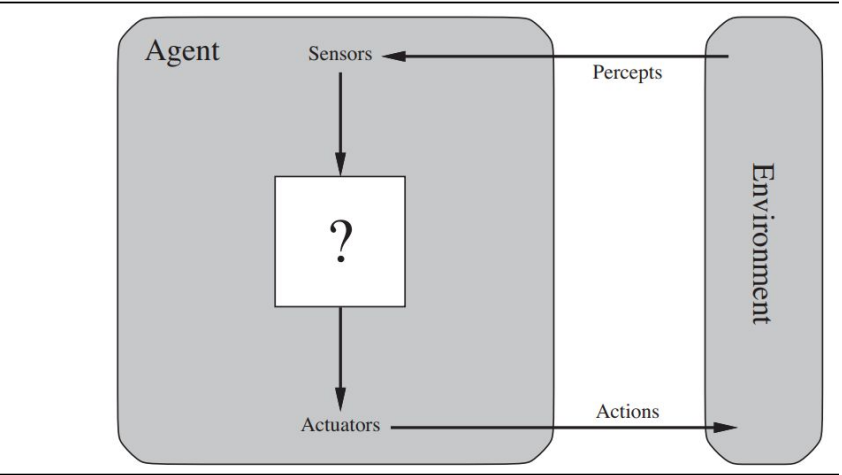
- Read Section 1.3 from book (for general on AI)
- Read from the articles circulated on MIS (for AI in process engineering applications)

Contd. from last side...

Turing test (Acting humanly):

A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer

Agent-Environment Interaction

- Percept: the agent's perceptual inputs at any given instant
 - Percept Sequence: the complete history of everything the agent has ever perceived
 - Agent Function: maps any given percept sequence to an action
- 
- The diagram illustrates the interaction between an Agent and an Environment. The Agent is represented by a large rounded rectangle on the left, containing a central white box with a question mark. Above this box is the label 'Sensors' with an arrow pointing down into the box. Below the box is the label 'Actuators' with an arrow pointing down from the box. The Environment is represented by a vertical rounded rectangle on the right. An arrow labeled 'Percepts' points from the Environment to the Sensors. Another arrow labeled 'Actions' points from the Actuators to the Environment.
- Figure 2.1 Agents interact with environments through sensors and actuators.
- Agent Program: a concrete implementation of the agent function, running within some physical system (Agent = Agent Architecture + Agent Program)
 - Performance Measure: Evaluation metric for any sequence of environment states to adjudge whether the actions done by agent is desired, from the perspective of the environment; it is specific to a case [eg., minimum time/energy/cost etc.]
[eg.: amount of dirt cleaned or cleanliness of the floor - which one more proper ?]

Rational Agent

- Rationality (maximizing expected performance) is not same as perfection/"knowing everything" (maximizing actual performance)

- An agent, acting as rational, may become irrational for circumstances (eg. needless motion of the cleaner even after cleaning up all dirt if performance measure is only cleanliness) (eg. not exploring unexplored region of environment)
- Rational choice depends upon only on the percept sequence till date
- Rational agent should do "information gathering" (eg., looking both the sides before crossing a road) that might modify future percepts and "exploration" (for initially unknown environment)

2.2.1 Rationality

What is rational at any given time depends on four things:

- The performance measure that defines the criterion of success.
- The agent's prior knowledge of the environment.
- The actions that the agent can perform.
- The agent's percept sequence to date.

This leads to a **definition of a rational agent**:

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rational Agent

- A rational agent not only gathers information but also learns as much as possible from what it perceives (except when the environment is completely known a priori)
- An agent which does not learn is typically fragile - i.e., it may fail in a new environment
- A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge (and, that involves randomness in action in initial times)
- After sufficient experience of its environment, the behavior of a rational agent can become effectively independent of its prior knowledge (so, it can do well in a variety of environments)

Task Environments

- Task environments are essentially the “problems” to which rational agents are the “solutions.”
- These are specified by Performance Measure, Environment, Actuators, Sensors

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Figure 2.5 Examples of agent types and their PEAS descriptions.

Properties of Task Environments

- Observability
 - A task environment is effectively fully observable if its sensors detect all aspects, “relevant” (as per performance measure) to the choice of action.
 - Partial observability due to faulty/no sensor for some of the “relevant” aspects
- Single and Multi Agent
 - From an agent A’s perspective, is another agent B to be treated as an agent or just another object ??

It depends upon whether B’s behavior is best described as maximizing/minimizing a performance measure whose value depends on agent A’s behavior.
 - Cooperative and Competitive agents (for the latter, randomness desired to avoid predictability)

Properties of Task Environments

- Deterministic/Stochastic
 - If the next state of the environment is completely determined by the current state and the action executed by the agent, the environment is deterministic; otherwise, it is stochastic.
 - If the environment is partially observable, it could appear to be stochastic.

An environment is said to be uncertain if it is not fully observable or not deterministic.

- Episodic/Sequential
 - Episodic: In each episode, the agent receives a percept and then performs a single action. The next episode does not depend on the actions taken in prior episodes. Many classification tasks are episodic.
 - In sequential environments, the current decision could affect all future decisions (so, the agent needs to think ahead).

Properties of Task Environments

- Static/Dynamic
 - If the environment can change while an agent is deliberating, the environment is dynamic for that agent; otherwise, it is static.
 - Semi-dynamic: if the agent's performance score would change with time
- Discrete/Continuous
 - Applicable to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
- Known/Unknown (not related to Observability)
 - It refers to the agent's state of knowledge about the laws of the environment.
 - In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
 - If the environment is unknown, the agent will have to learn how it works in order to make good decisions.

Task Environments: Properties

Most difficult scenario: Partially observable, Multiagent, Stochastic, Sequential, Dynamic, Continuous, and Unknown

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete
Figure 2.6 Examples of task environments and their characteristics.						

For details,
read
relevant
sections
from the
book

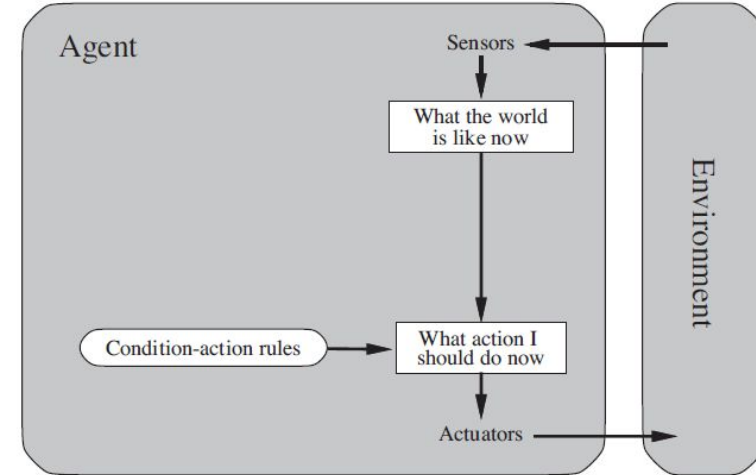
Agent Programs

- Agent Programs take the current percept as input from the sensors and return an action to the actuators
- If the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts
- Simplest approach: a look-up table, populated with outcomes for all possible percepts and actions
- **The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table.**

Four basic kinds of agent programs that embody the principles underlying almost all intelligent systems: Simple reflex agents, Model-based reflex agents, Goal-based agents, Utility-based agents

Simple Reflex Agents

- These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
- These are too simple to be implemented, yet of limited intelligence.
- Simple reflex behaviors can occur even in more complex environments.



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

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*

Such an agent may fail if the environment is not completely observable. It may also get into infinite loop for such a scenario. Way to avoid infinite loop: through randomized action by the agent

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

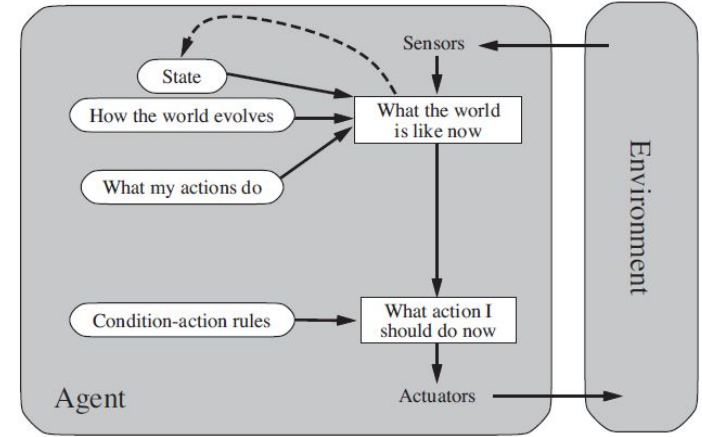
Model-based Reflex Agents

- To tackle partial observability, the agent should keep track of the unobserved part of the world. It should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state. Such internal state would be updated as per “Model”.

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
               model, a description of how the next state depends on current state and action
               rules, a set of condition-action rules
               action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.12 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: Knowledge (**best guess, not accurate**) about the world

- Some information about how the world evolves independently of the agent
- Some information about how the agent's own actions affect the world
- Representation of the models and states vary widely as per the type of environment and the particular technology of the agent design

Goal-based Agents

- The agent needs some sort of goal information that describes situations that are desirable (eg., reaching a destination, attaining a temperature)
- “Search” and “Planning” are two typical activities of goal-based agents
- Here, no direct map between percepts and actions
- No need to write new sets of condition-action rules (like for reflex agents) for change in goal or change in environment

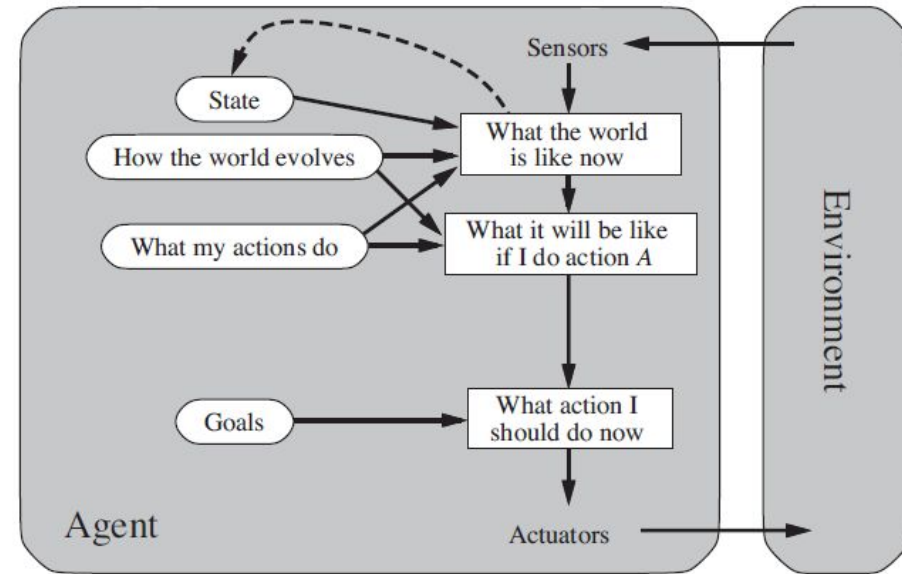


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Utility-based Agents

- There may be many paths, leading to a goal. So, a more general performance measure should allow a comparison of different world states according to exactly how good/desired they are
- An agent's utility function is essentially an internalization of the performance measure.
- If the internal utility function and the external performance measure are in agreement, an agent that chooses actions to maximize its utility will be rational according to the external performance measure.
- In case of partial observability and "expected utility" is to be considered.

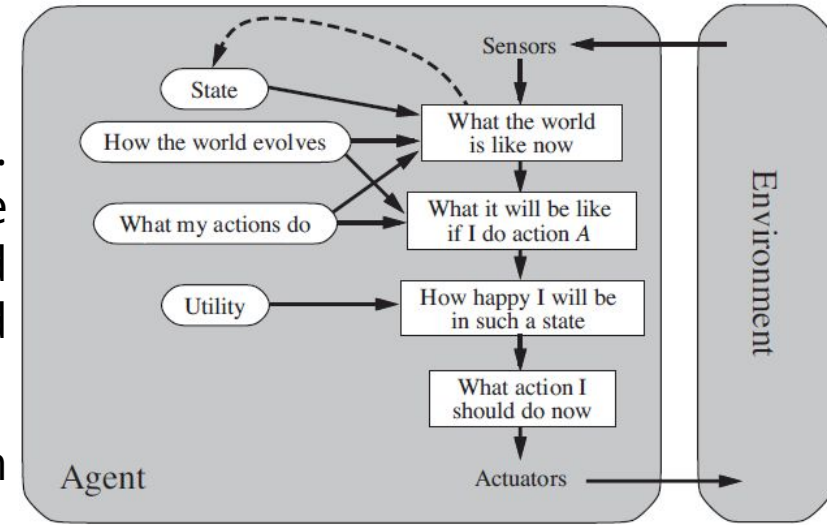


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

- Also useful for the cases of several and/or conflicting goals
- But, notion of utility is NOT the only way to be a rational agent

Distinctions among Structure of Various Agents

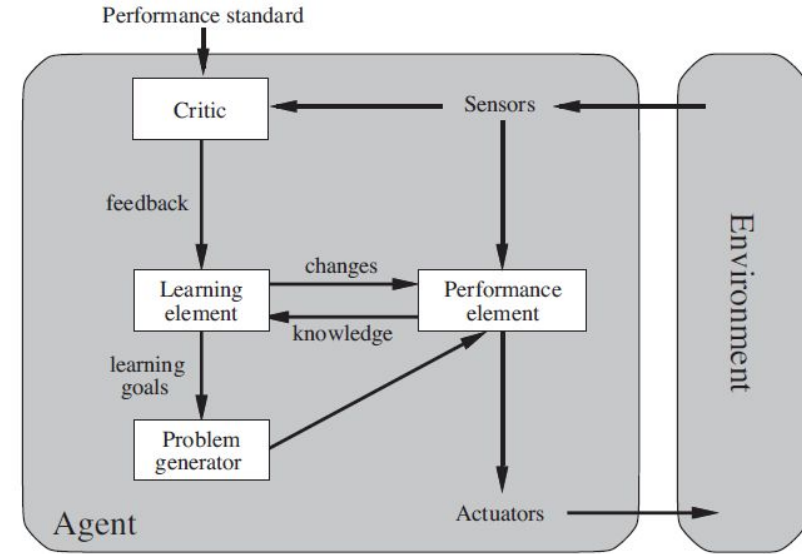
An example:

- The previous car has slowed
- The agent (the car in question) knows about that through Percept
- Reflex agents: seeing that, they would slow down the car (as per percept-action mapping of the function) [in case of unobservability, the model would help]
- Goal-based agents: to avoid collision, they would slow down
- Utility-based agents: How close the two cars can be

Similar understanding for any other instances

Learning Agents

- It may be better to build learning agents and teach them as the “Learning” allows them to operate in initially unknown environments and to become more competent than its initial knowledge
- “Learning element”: Makes improvements
- “Performance element”: Takes in Percept and Selects actions
- “Critic”: Gives feedback on how the agent is doing relative to a fixed performance standard... accordingly, the “Learning Elements” modifies “Performance element” to make improvement in future
- “Problem Generator”: Suggests actions that will lead to new and informative experiences (guides agent in exploring a little... potentially leading to suboptimal actions in the short run, but, maybe much better actions in the long run)



Learning Agents: Features

- The design of the learning element depends much on the design of the performance element. “What kind of performance element will the agent need to do this once it learns?” is the most crucial question; as, given an agent design, learning mechanisms can be constructed to improve every part of the agent.
- The critic is necessary because the percepts themselves provide no indication of the agent’s success. The agent should not modify the performance standard.
- Learning element can make changes to any of the “knowledge components” (i.e., how the world evolves, what my actions do etc.) for any of the four agent types.
- Agents have a variety of components, represented in many ways within the agent program, thereby leading to many different learning methods.
- Learning (by any method) in intelligent agents can be viewed as a process of modification of each component of the agent to bring the components into closer agreement with the available feedback information, thereby improving the overall performance of the agent.

Learning Agents: Features

- For a simple reflex agent: learning (witnessing) directly from percept sequence
- For a utility-based agent that wishes to learn utility information: performance standard distinguishes part of the incoming percept as a reward (or penalty) that provides direct feedback on the quality of the agent's past actions

Example on Interactions among Components in a Learning Agent

To make the overall design more concrete, let us return to the automated taxi example. The performance element consists of whatever collection of knowledge and procedures the taxi has for selecting its driving actions. The taxi goes out on the road and drives, using this performance element. The critic observes the world and passes information along to the learning element. For example, after the taxi makes a quick left turn across three lanes of traffic, the critic observes the shocking language used by other drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by installation of the new rule. The problem generator might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces under different conditions.

Ways of Representing the Components of Agent Programs

- Various ways for the components (which constitute an agent program) to represent the environment that the agent inhabits
- A more expressive representation can capture at least as concisely, everything a less expressive one can capture, plus some more

increasing complexity and expressive power;
so, more complex reasoning and learning

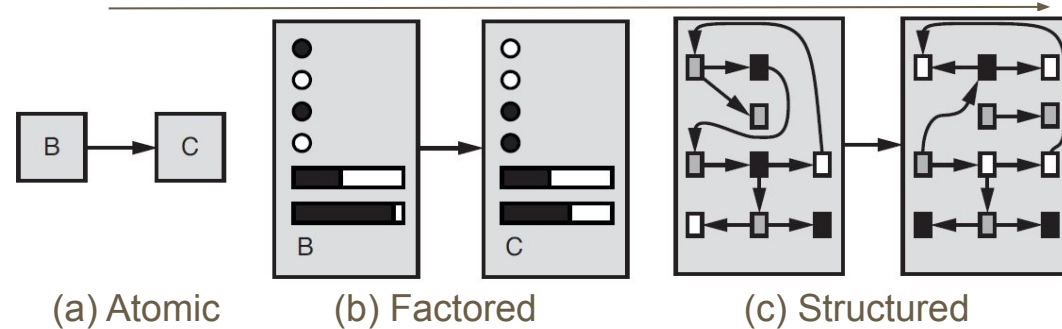


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

Atomic: states of the world (relevant information) are indivisible

Ex.: the name of the city we are in or the present temperature value which is only a “single atom of knowledge”, i.e., a “black box” whose only discernible property is that of being identical to or different from another black box

No commonalities between two different atomic states

Algorithms using such representation:
Search and game-playing, Hidden Markov models, Markov decision processes

Ways of Representing the Components of Agent Programs

Factored: Splits up each state into a fixed set of variables or attributes, each of which can have a value

[more than just atomic knowledge -
higher fidelity description than atomic]

Eg.: not only the name of the city we are in or the present temperature value; but also more detailed information about the state

Two different factored states can share some of the attributes; so, quite easy to understand transition of one state to another

Also possible to represent uncertainty (leaving the corresponding attribute blank)

Algorithms using such representation:

Constraint satisfaction algorithms, Propositional logic, Planning, Bayesian networks, and most of the Machine Learning algorithms

Structured: Representing the world as having things in it that are “related” to each other, not just variables with values

The different objects in the environment and their various and varying relationships being described explicitly (not just values or T/F)

Algorithms using such representation:

Relational databases and First-order Logic, First-order probability models, Knowledge-based learning, and much of the Natural Language understanding

To gain the benefits of expressive representations while avoiding their drawbacks, intelligent systems for the real world may need to operate with all the three representations simultaneously