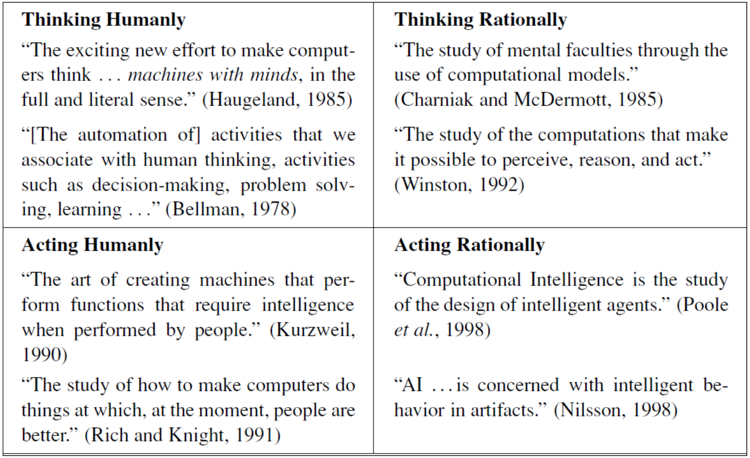
Rational (Intelligent) Agents

Jonathan Mwaura

jonathan_mwaura@uml.edu

Overview



Thinking Humanly "The exciting new effort to make computers think . . . *machines with minds*, 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 *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure: Some definitions of artificial intelligence, organized into four categories

Rationality

- Alternative approach to intelligence relies on the notion of rationality.
- Typically this is a precise mathematical notion of what it means to do the right thing in any particular circumstance.
- Provides a precise mechanism for analyzing and understanding the properties of this ideal behavior we are trying to achieve.
- A precise benchmark against which we can measure the behavior the systems we build.

Rationality

- Mathematical characterizations of rationality have come from diverse areas like logic (laws of thought) and economics (utility theory $\longrightarrow$ how best to act under uncertainty, game theory $\longrightarrow$ how self-interested agents interact).
- There is no universal agreement about which notion of rationality is best, but since these notions are precise we can study them and give exact characterizations of their properties, good and bad.
- We will focus on acting rationally

NOTE: Acting rationally has implications for thinking/reasoning

Computational Intelligence

AI/CI/MI

The science (or even art) that tries to understand and model intelligence as a computational process.

In terms of computation

Thus we try to construct systems whose computation achieves or approximates the desired notion of rationality

AI $\in$ CS

Other areas interested in the study of intelligence lie in other areas or study, e.g., cognitive science which focuses on human intelligence. Such areas are very related, but their central focus tends to be different

Attributes of an Intelligent Agent

- Autonomy - Acting without direct control by humans or others.
- Social ability - Capacity to interact with other agents and/or users
- Re-activity - Ability to react to stimuli (i.e. the percepts received via sensors and act via the actuators)
- Pro-activeness - Capacity to be goal directed and therefore act by taking initiative to fulfil their goal.

Degrees of Intelligence

Building an intelligent system as capable as humans remains an elusive goal.

- However, systems have been built which exhibit various specialized degrees of intelligence.
- Formalisms and algorithmic ideas have been identified as being useful in the construction of these “intelligent” systems.
- Together these formalisms and algorithms form the foundation of our attempt to understand intelligence as a computational process.

In this course we will study some of these formalisms and see how they can be used to achieve various degrees of intelligence.

Agency

What's an agent?

An agent is an entity that exists in an environment and that **acts** on that environment based on its **perceptions** of the environment

An intelligent agent acts to further its own interests (or those of a user).

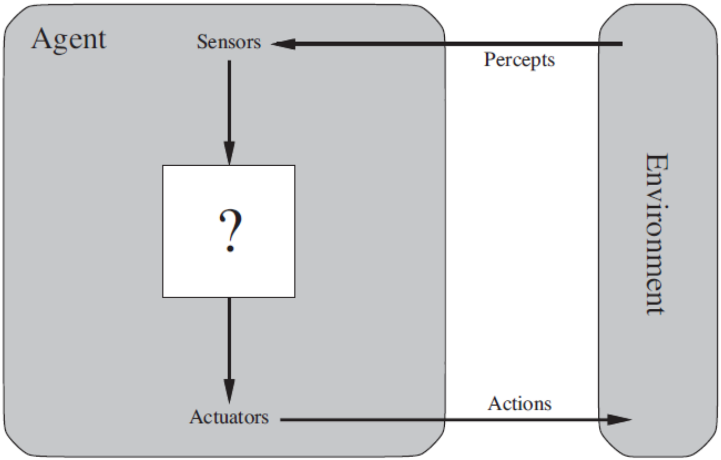


Figure: Agents interact with environments through sensors and actuators

Percept

agents perceptual (stimuli) inputs at any given instant

Percept sequence

complete history of everything the agent has ever perceived

Important

an agents choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasnt perceived

Agent Function and Program

Agent Function

Agents behavior is mathematically described by: A function mapping any given percept sequence to an action

$$AgentFunc = Percept^{*} \implies Action \tag{1}$$

In practice, the agent function is achieved through an Agent's program and a real implementation.

Vacuum Cleaner World

Percepts

Location: Right or Left

Status: Clean or dirty

Actions

Move left, Move right, suck, do nothing

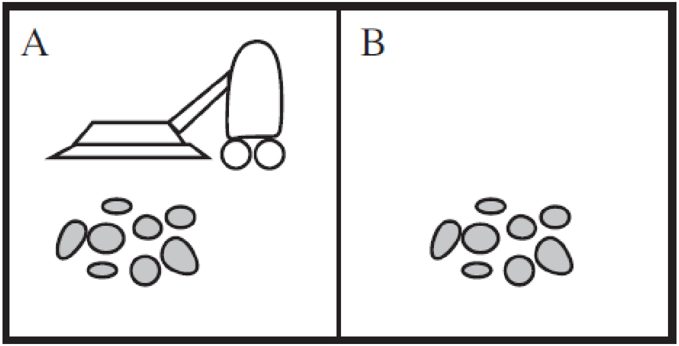
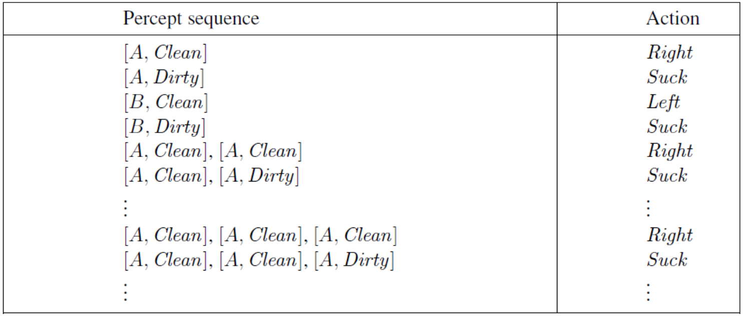


Figure: Vacuum Cleaner world



Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
$\vdots$	$\vdots$
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
$\vdots$	$\vdots$

Figure: Vacuum cleaner agent function

Concept of Rationality

Rational agents

One that does the right thing

that is; every entry in the table for the agent function is correct (rational).

What is the right thing?

The actions that cause the agent to be most successful

So we need ways to measure success

Concept of Rationality

Performance measure

An objective fitness function that determines how fit (correct) an agent is at a particular task.

Based on the mapping between percepts and actions!

If the actions are desirable then the agent is performing well.

Note: There is no universal performance measure for all agents.

Performance Measure

General rule

Design performance measures according to what one actually wants in the environment rather than how one thinks the agent should behave.

Vacuum cleaner

We want the floor clean, no matter how the agent behave

We dont restrict how the agent behaves.

Note: There is no universal performance measure for all agents.

Rationality

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

- The performance measure defining the criterion of success.
- The agents' prior knowledge of the environment
- The actions that the agent can perform
- The agents' percept sequence up to now.

Rational Agent

For each possible percept sequence; a rational agent should select

an action expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has

For Instance

On an exam; Maximise your grade based on the questions asked and what you already know!

Vacuum cleaner

Performance measure

Awards one point for each clean square at each time step, over 10000 time steps

Prior knowledge of the environment?

The geography of the environment

Only two squares

The effect of the actions

Q1

Is there circumstances where this vacuum cleaner agent will be irrational?

Omniscience

An omniscient agent

- Knows the actual outcome of its actions in advance
- No other possible outcomes
- However, impossible in real world

Learning

Does a rational agent depend on only current percept?

Learning

Does a rational agent depend on only current percept?

- No, the past percept sequence should also be used
- After experiencing an episode, the agent should adjust its behaviors to perform better for the same job next time
- This is called learning

Autonomy

If an agent just relies on the prior knowledge of its designer rather than its own percepts then the agent lacks autonomy

A rational agent should be autonomous- it should learn what it can to compensate for partial or incorrect prior knowledge.

Embodied Agents Vs Software Agents

Sometimes, the environment may not be the real world

- E.g., flight simulator, video games, Internet
- They are all artificial but very complex environments
- Those agents working in these environments are called
    - Software agent (softbots)
    - Because all parts of the agent are software

Task Environments

In general, **Task environments** are the problems while the **rational agents** are the solutions.

PEAS

Task environments are specified by analyzing the following attributes of the task :

- Performance measure
- Environment - its characteristics
- Actuators - how the agent shall interact with the environment
- Sensors - how the agent shall infer/perceive the environment.

NOTE: In designing an agent, the first step must always be to specify the task environment as fully as possible.

Automated Taxi Driver (Uber)

Performance

How can we judge the automated driver?

Which factors are considered?

- getting to the correct destination
- minimizing fuel consumption
- minimizing the trip time and/or cost
- minimizing the violations of traffic laws
- maximizing the safety and comfort, etc.

Automated Taxi Driver (Uber)

Environment

- A taxi must deal with a variety of roads
- Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, etc.
- Interact with the customer

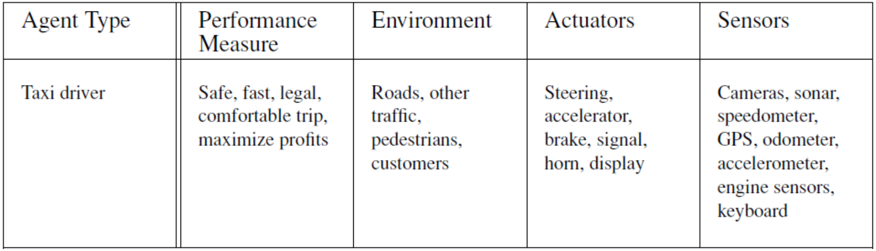
Automated Taxi Driver (Uber)

Actuators

- Control over the accelerator, steering, gear shifting and braking
- A display to communicate with the customers

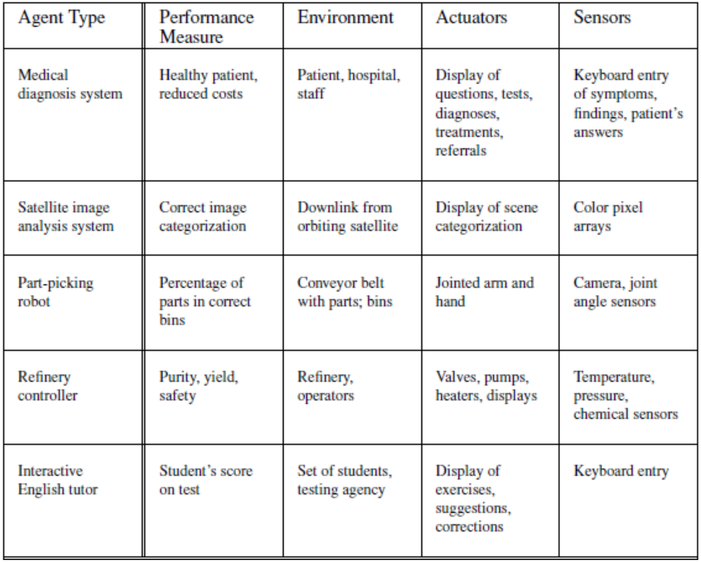
Sensors

- Detect other vehicles, road situations
- GPS (Global Positioning System) to know where the taxi is
- Many more devices are necessary



Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure: PEAS description of an automated taxi



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: Examples of agent types and their PEAS descriptions

Properties of Task Environments

Fully observable

If an agents sensors give it access to the complete state of the environment at each point in time then the environment is effectively and fully observable.

- if the sensors detect all aspects
- That are relevant to the choice of action

Partially observable

An environment might be Partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.

e.g. A local dirt sensor of the cleaner cannot tell whether other squares are clean or not.

Properties of Task Environments

Deterministic vs. stochastic

If the next state of the environment is completely determined by the current state and the actions executed by the agent, then the environment is deterministic, otherwise, it is stochastic.

Strategic environment: deterministic except for actions of other agents

Properties of Task Environments

Episodic vs. sequential

An episode = agents single pair of perception and action

- The quality of the agents action does not depend on other episodes : Every episode is independent of each other
- Episodic environment is simpler : The agent does not need to think ahead

Sequential

Current action may affect all future decisions.

e.g. An automated vehicle, game of chess etc

Properties of Task Environments

Static vs. Dynamic

- A dynamic environment is always changing over time e.g. E.g., the number of people in the street
- Static environment does not change e.g. a destination

Semi-dynamic

environment is not changed over time but the agents performance score does

Properties of Task Environments

Discrete vs. continuous

If there are a limited number of distinct states, clearly defined percepts and actions, the environment is discrete E.g., Chess game

Continuous: Taxi driving

Properties of Task Environments

Single agent VS. multiagent

Playing a crossword puzzle single agent

Chess playing two agents

Competitive multiagent environment e.g. chess playing

Partially cooperative multiagent environment e.g. Automated taxi driver etc

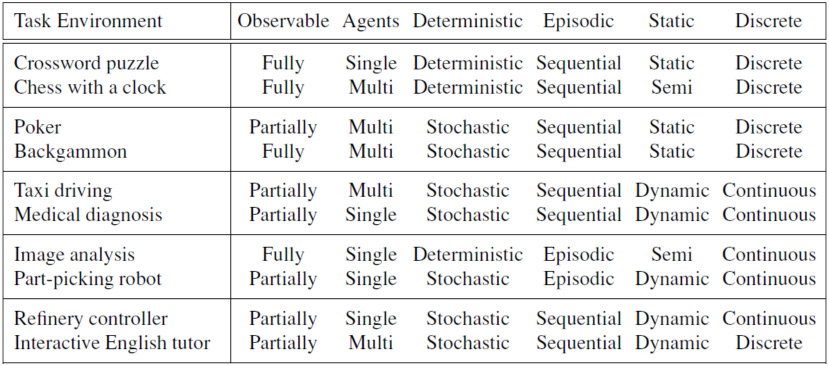
Properties of Task Environments

Known vs. unknown

This distinction refers not to the environment itself but to the agents (or designers) state of knowledge about the environment.

In known environment, the outcomes for all actions are given. (example: solitaire card games).

If the environment is unknown, the agent will have to learn how it works in order to make good decisions.(example: new video game).



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: Examples of task environments and their characteristics

Structure of agents

Agent = architecture + program

- Architecture = some sort of computing device (sensors + actuators)
- (Agent) Program = some function that implements the agent mapping = “”
- Agent Program = Job of AI

Agent programs

Input for Agent Program

Only the current percept

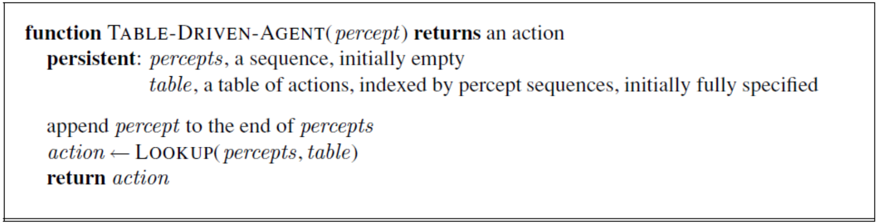
Input for Agent Function

The entire percept sequence

The agent must remember all of them

Implement the agent program as:

A look up table (agent function)



```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Figure: The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory

Factors affecting an agent program

P

The set of all possible percepts

T

Agent's lifetime

Governs the total number of percepts the agent receives

Size of look up table.

Table driven agent programs

Despite of huge size, look up table does what we want.

The key challenge of AI

- Find out how to write programs that, to the extent possible, produce rational behavior
  - From a small amount of code
  - Rather than a large amount of table entries

e.g. A five-line of the Newtons-Method rather than a huge table of square roots , cosines, sines etc

Types of agent programs

Four types:

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

Simple Reflex agents

It uses condition-action rules:

- The rules are like the form “if then ”
- Efficient but have narrow range of applicability
- Because knowledge sometimes cannot be stated explicitly
- Will only work if the environment is fully observable

NOTE: condition -action rules is also referred to as situation-action rules, production rules or if-then rules.

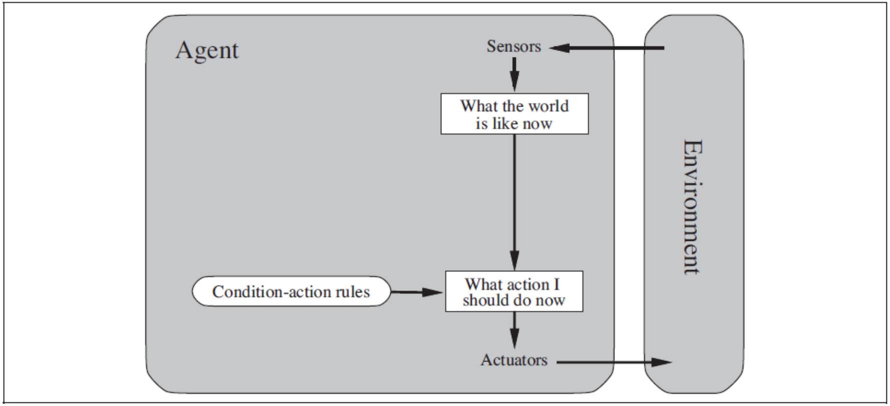
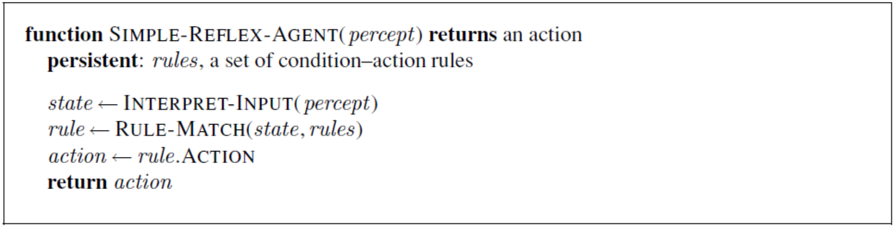


Figure: Schematic diagram of a simple reflex agent.

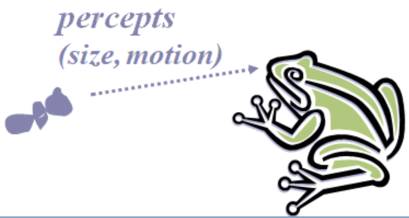


```
function SIMPLE-REFLEX-AGENT($percept$) returns an action
    persistent: $rules$, a set of condition–action rules

    $state \leftarrow$ INTERPRET-INPUT($percept$)
    $rule \leftarrow$ RULE-MATCH($state, rules$)
    $action \leftarrow rule$.ACTION
    return $action$
```

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

Reflex agent in nature



RULES:

(1) If small moving object,
    then activate SNAP

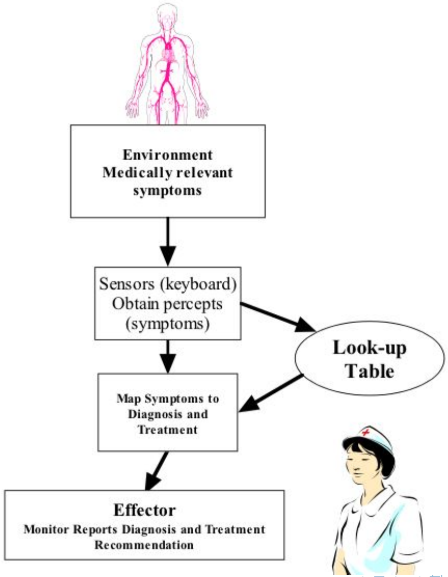
(2) If large moving object,
    then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

needed for completeness

Action: SNAP or AVOID or NOOP

A DSS as a reflex agent


Environment
Medically relevant symptoms
Sensors (keyboard)
Obtain percepts (symptoms)
Look-up Table
Map Symptoms to Diagnosis and Treatment
Effector
Monitor Reports Diagnosis and Treatment Recommendation

Model-based Reflex Agents

- For the world that is partially observable the agent has to keep track of an internal state
  - That depends on the percept history
  - Reflecting some of the unobserved aspects E.g. driving a car and changing lane
- Two types of knowledge is needed
  - How the world evolves independently of the agent
  - How the agents actions affect the world

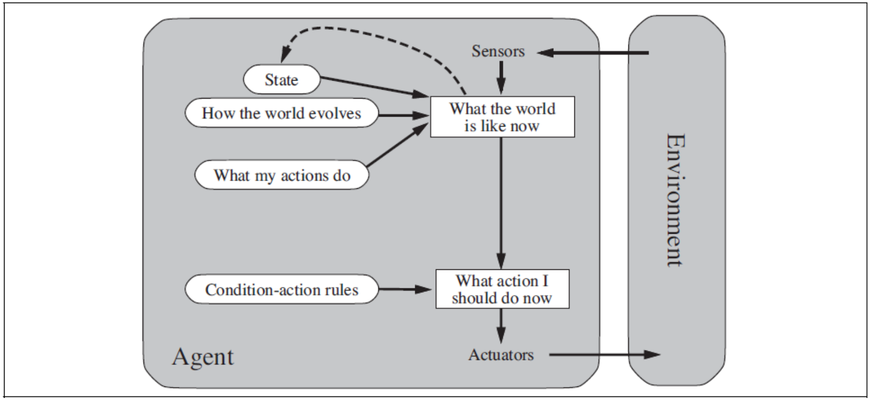


Figure: A model-based reflex agent.

```
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: 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.

Brooks Subsumption Architecture

Main idea: build complex, intelligent robots by decomposing behaviors into a hierarchy of skills, each defining a percept-action cycle for one very specific task.

Examples: collision avoidance, wandering, exploring, recognizing doorways, etc.
Each behavior is modeled by a finite-state machine with a few states (though each state may correspond to a complex function or module; provides internal state to the agent).

Behaviors are loosely coupled via asynchronous interactions.

Note: minimal internal state representation.

Brooks Subsumption Architecture

In subsumption architecture, increasingly complex behaviors arise from the combination of simple behaviors.

The most basic simple behaviors are on the level of reflexes:

- avoid an object
- go toward food if hungry
- move randomly

A more complex behavior that sits on top of simple behaviors may be go across the room.

The more complex behaviors subsume the less complex ones to accomplish their goal.

Brooks Subsumption Architecture

Planning and reasoning about our surroundings appears to require some kind of internal representation of our world. We can try things out in this representation. Much like an running a simulation of the effect of actions or a sequence of actions in our head.

General assumption for many years: The more detailed internal model, the better.

Brooks Subsumption Architecture

Brooks (mid 80s and 90s) challenged this view:
The philosophy behind Subsumption Architecture is that the world should be used as its own model.

According to Brooks, storing models of the world is dangerous in dynamic, unpredictable environments because representations might be incorrect or outdated. What is needed is the ability to react quickly to the present. So, use minimal internal state representation, complement at each time step with sensor input.

Debate continues to this day: How much of our world do we (should we) represent explicitly? Subsumption architecture worked well in robotics.

Goal-based agents

- Current state of the environment is always not enough
- The goal is another issue to achieve
    - Judgment of rationality / correctness
- Action chosen leads to goals, based on:
    - the current state
    - current percept

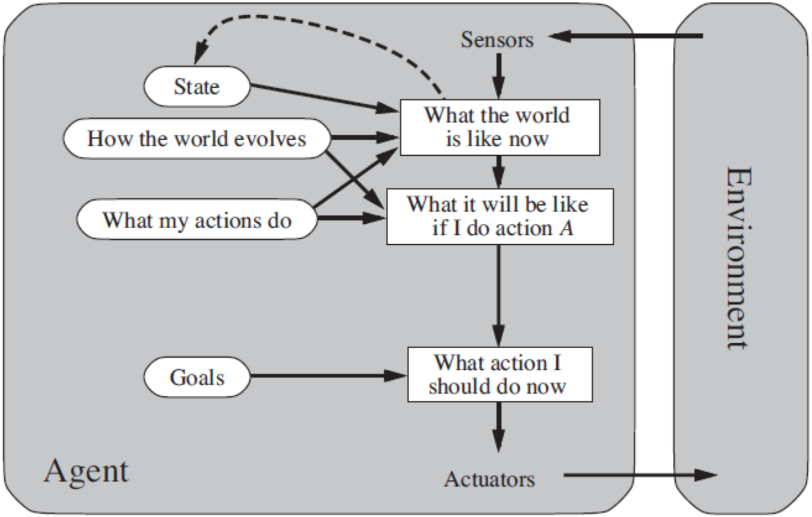

Sensors
State
How the world evolves
What my actions do
Goals
What the world is like now
What it will be like if I do action *A*
What action I should do now
Actuators
Agent
Environment

Figure: 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.

Goal-based agents

- Goal-based agents are less efficient but more flexible
- Agent : Different goals $\implies$ different tasks
- **Search** and **Rescue** operations are the main applications areas for these type of agents

Utility based agents

Goals alone are not enough to generate high-quality behavior

- Many action sequences lead to goals
- some are better and some worse

If goal means success,then utility means the degree of success (how successful it is)

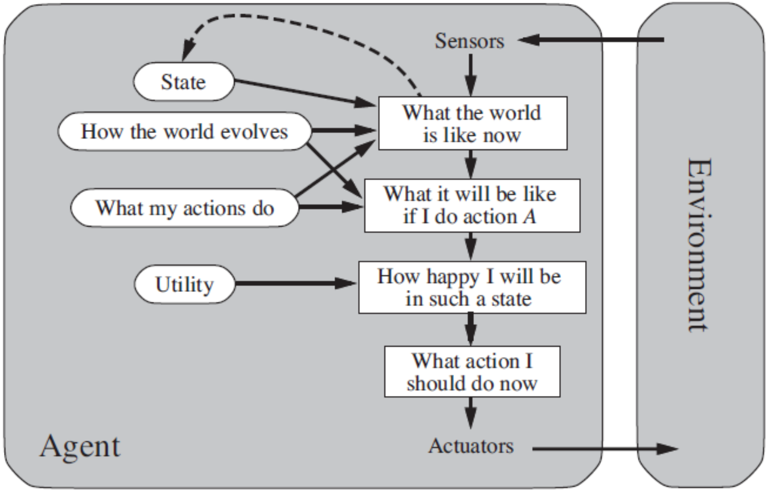

Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Actuators
Agent
Environment

Figure: 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.

Utility based agents

State A has higher utility if its more preferred that other States

Utility is therefore:

- a function that maps a state onto a real number
- a measure of degree of success

Utility based agents

Utility has several advantages:

- When there are conflicting goals,
  - Only some of the goals but not all can be achieved
  - utility describes the appropriate trade-off
- When there are several goals
  - None of them are achieved certainly
  - utility provides a way for the decision-making

Learning agents

After an agent is programmed, can it work immediately?

in AI, an agent:

- Learns via examples
- Its learning success is achieved by testing using a different set of examples.

Learning agents adapt and improve over time

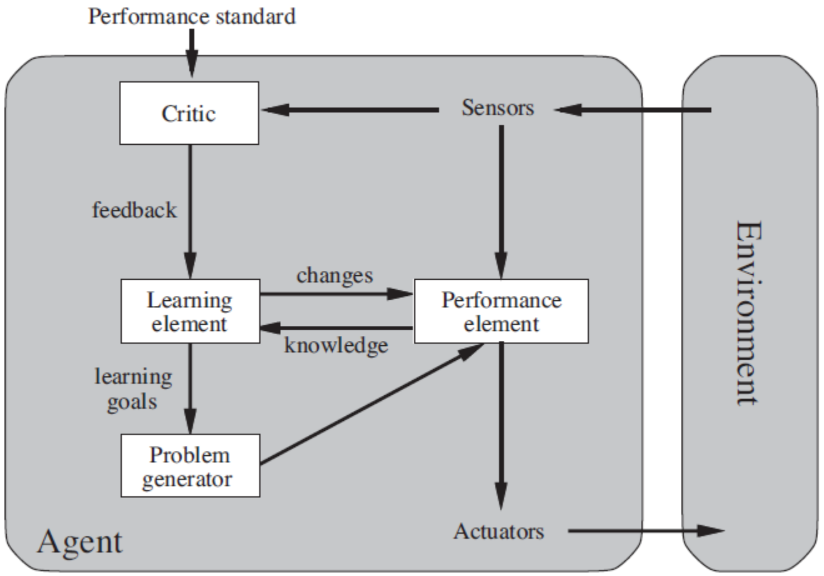


Figure: A general learning agent.

Learning agents

Four conceptual components

- Learning element : making improvement
- Performance element: Selecting external actions
- Critic: Tells the Learning element how well the agent is doing with respect to fixed performance standard.
- Problem generator: Suggest actions that will lead to new and informative experiences.

Summary:Agents

- An agent perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- A rational agent always chooses the action which maximizes its expected performance, given its percept sequence 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 percept to action and updates its internal state.
- Representing knowledge is important for successful agent design.
- The most challenging environments are partially observable, stochastic, sequential, dynamic, and continuous, and contain multiple intelligent agents

Summary: Agent types

- Table-driven agents : use a percept sequence/action table in memory to find the next action. They are implemented by a (large) lookup table.
- Simple Reflex agents: are based on condition-action rules, implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.
- Model-based reflex agents - Agents with memory have internal state, which is used to keep track of past states of the world.

Agent types

- Goal-based agents: are agents that, in addition to state information, have goal information that describes desirable situations. Agents of this kind take future events into consideration.
- Utility-based agents: base their decisions on classic axiomatic utility theory in order to act rationally.
- Learning agents: they have the ability to improve performance through learning.

The End