CSC9YE:
Intelligent Agents I & II

Notes covering TWO Lectures on
Feb 26th & Mar 1st

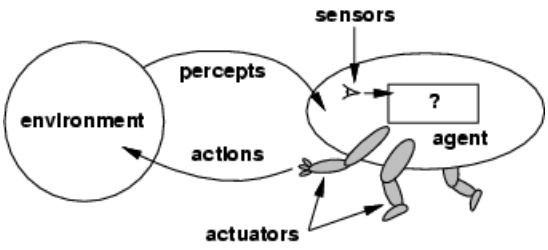
Outline

- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors) Description of Task Environment
- Environment types
- Agent types
- Reference: Chapter 2

Intelligent Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
- Human agent:
  - eyes, ears, and other organs for sensors;
  - hands, legs, mouth, and other body parts for actuators
- Robotic agent:
  - cameras and infrared range finders for sensors;
  - various motors for actuators

Agents and environments

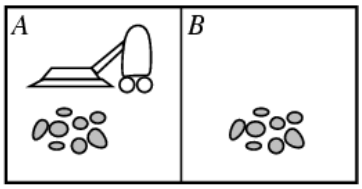


- The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$$

- The agent program runs on the physical architecture to produce *f*
- agent function = architecture + agent program

Vacuum-cleaner world



- Percepts: location and contents, e.g., [*A,Dirty*]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

A vacuum-cleaner agent

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*
.	.
.	.

Simple agent function is: If current square is dirty then suck, else move to other square. A partial tabulation of this is shown here – Can see various world agents by filling in right hand column in various ways

```
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
```

What is the right function? (i.e. right way to fill out table above, what makes agent good or bad, intelligent or stupid…)
Can it be implemented in a small agent program?

Rational agents-I

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful
- **Performance measure (PM)**: An objective criterion for success of an agent's behavior
- E.g. performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.
- Fixed performance measure evaluates *environment sequence*
  - One point per square cleaned up in time T
  - One point per clean square per time step, minus one per move?
  - Penalize for >k dirty square?
- Selection of PM is not always easy!

Rational agents-II

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

- The performance measure that defines the degree of success
- Everything the agent has perceived so far: the percept history, or percept sequence
- What the agent knows about the environment
- The actions the agent can perform

Rational agents-III

- A rational agent chooses whatever action maximises the *expected* value of the performance measure given the percept sequence *to-date*
  - *Rationality maximizes 'expected' performance while perfection maximizes 'actual' performance*
- Rationality is distinct from omniscience (all-knowing with infinite knowledge)
  - Because rationality depends on percept sequence TO-DATE
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering e.g. *exploration*)
- A rational agent is autonomous if its behavior is determined by its own experience (with ability to *learn* and adapt)
  - Need to provide agent with initial knowledge & ability to learn
  - But rational agent should be able to compensate for partial or incorrect prior knowledge (of its designer)

Rational agents-IV

- *Ideal 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.
- As a general rule, it is better to design performance measure according to what one actually wants in the environment, rather than according to how one thinks the agent should behave

PEAS Description of Task Environment

- To design the agent, we must specify the task environment (which is 'the problem' to which agents are the solution!)
- PEAS: Performance measure, Environment, Actuators, Sensors
  - *Note: AIMA 1st Edition Textbook uses: PAGE (Percepts, Actions, Goals, Environment)*
- Must first specify the setting for intelligent agent design
- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS - I

- Must first specify the setting for intelligent agent design
- Consider, e.g., the task of designing an automated taxi driver (task currently beyond capabilities of existing technology!)
  - **Performance measure**: Safe, fast, legal, comfortable trip, maximize profits
  - **Environment**: Roads, other traffic, pedestrians, customers
  - **Actuators**: Same as available to human driver e.g. control over engine through steering wheel, accelerator, brake, signal, horn, and also perhaps voice synthesizer, display screen
  - **Sensors**: Needs to know where it is on road & who else is, how fast its going etc. through e.g. Cameras, IR or sonar, speedometer, GPS, odometer, engine sensors, keyboard

PEAS - II

- Agent: Medical diagnosis system
- Performance measure: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS - III

- Agent: Part-picking robot
- Performance measure: Percentage of parts in correct bins
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

PEAS - IV

- Agent: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

Environment types-I

- Fully observable (accessible) vs. partially observable (inaccessible):
  - Fully observable if agent's sensors detect all aspects of environment relevant to choice of action
  - Relevance depends on performance measure
  - Partially observable due to noisy, inaccurate or missing sensors
  - May need internal representation of environment (i.e. may need an internal state to keep track of the world – see more later)
- Deterministic vs. stochastic (non-deterministic):
  - The next state of the environment is completely determined by the current state and the action executed by the agent.
  - Partially observable environment may appear deterministic
  - Determine from point of view of agent
  - Strategic environment if the environment is deterministic except for the actions of other agents

Environment types-II

- Episodic vs. sequential:
  - The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action)
  - and the choice of action in each episode depends only on the episode itself
  - Sequential if current decisions affect all future decisions
- Static vs. dynamic:
  - Dynamic if the environment may change while an agent is deliberating.
  - The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does

Environment types-III

- Discrete vs. continuous:
- Can apply to state of environment, time, and percepts and actions of agent
- E.g. chess is discrete in all its aspects while taxi driving is continuous
- Single agent vs. multiagent:
- An agent operating by itself in an environment.
- When is an object an agent?
- Competitive or co-operative Multi-agent environments

Environment types-IV

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable			
Deterministic			
Episodic			
Static			
Discrete			
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

- *P.S.* Above answers can change depending on how you conceptualize the environment and agents

Environment types-IV

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

Environment types-V

	Solitaire	Internet Shopping	Taxi
Fully observable			
Deterministic			
Episodic			
Static			
Discrete			
Single agent			

Environment types-V

	Solitaire	Internet Shopping	Taxi
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

- *P.S.* Above answers can *change* depending on how you conceptualize the environment and agents

Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- In principle, one can supply each possible percept sequence to see what it does. Obviously a lookup table would usually be immense
- One agent function (or a small equivalence class) is rational
- *Aim*: find a way to implement the rational agent function *concisely* (i.e. design the agent *program*)
  - Agent program will run on some computing device called, architecture

    Agent=architecture+program

Note: Distinction between Agent program & agent Function: Agent program takes current percept as input & Agent function takes entire percept history (for mapping to actions)

Table-lookup agent

function TABLE-DRIVEN-AGENT(percept) returns an action

Static: *percepts*, a sequence initially empty
table, a table of actions, indexed by percept sequences, initially fully specified

Append current *percept* to the end of *percepts*
action <- LOOKUP*(percepts,table)*
return *action*

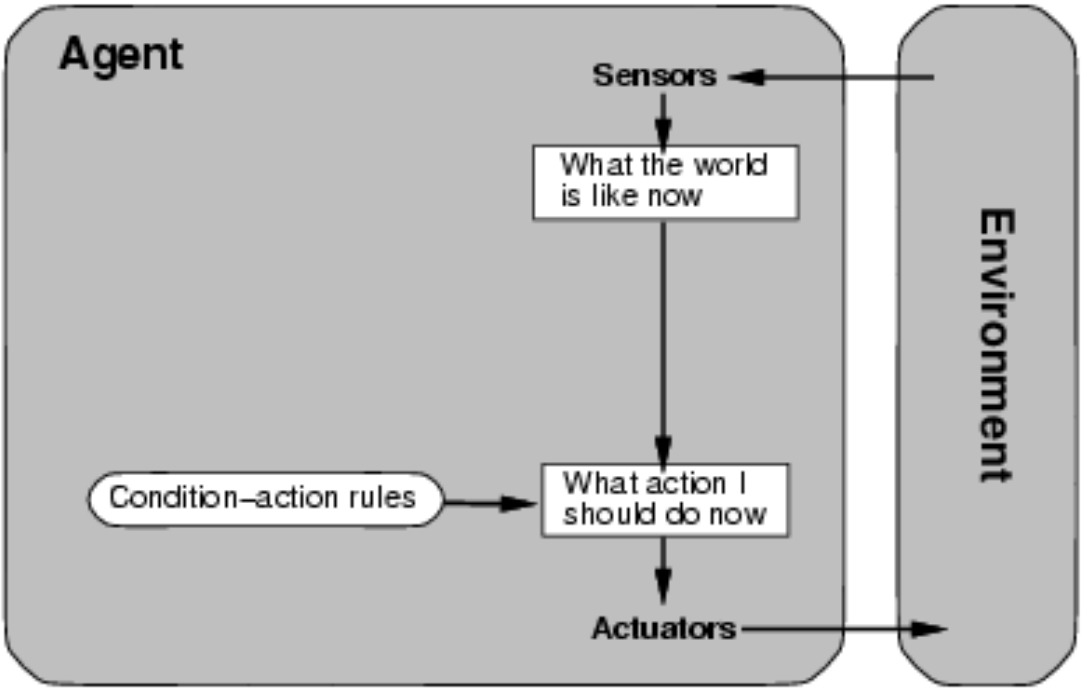
The TABLE-DRIVEN-AGENT is invoked for each new percept and returns an action each time. It keeps a track of the percept sequence using its own private data structure

- To build a rational agent in this way, the designer needs to construct a table that contains the appropriate action for every possible percept sequence.
- Drawbacks:
  - Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries
- AI Challenge: To write programs that produce rational behavior from a small amount of code rather than from a large number of table entries!
  - e.g. huge tables of square-roots used before 1970s now replaced by 5-line Newton program running on electronic calculator! – AI wants to generalize these!

Agent types

- Four basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple reflex agents-I



Simple reflex agents-II

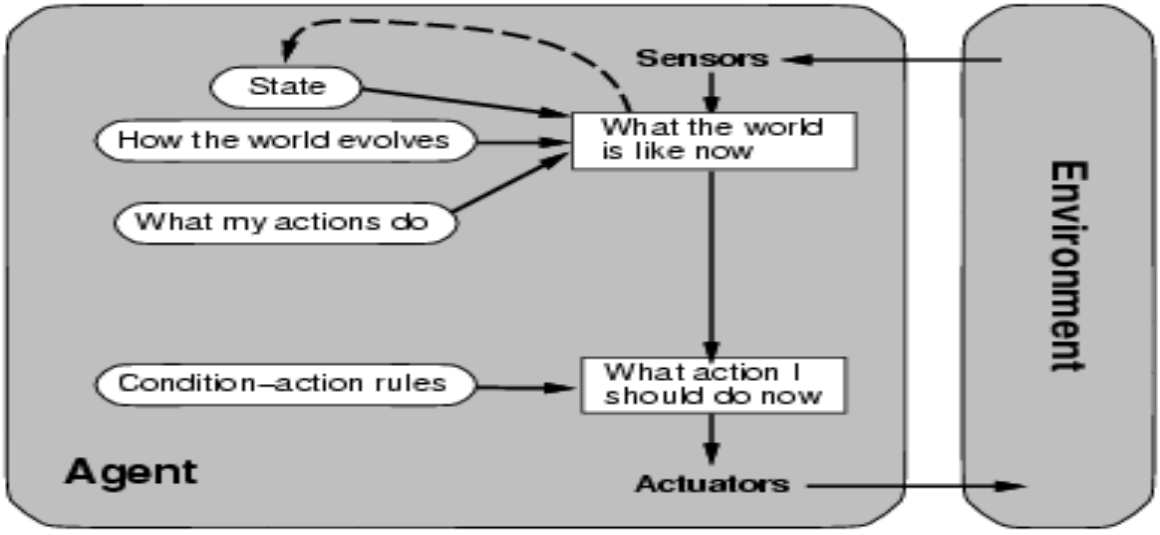
function SIMPLE-REFLEX-AGENT(*percept*) returns an *action*
Static: *rules*, a set of condition-action rules

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

- Function returns 1st rule in set of rules that matches given/current state

- These agents select actions on basis of current percept only.
- A simple Reflex-agent acts by finding a rule whose condition/state matches the current state (as defined by the percept alone),& then doing the action associated with that rule.
- Condition-Action (IF-THEN) rule: If *car-in-front-is-braking* then *initiate-breaking*

Model-based reflex agents-I



Rationale: Sensors do not give complete information of the world -> hence need an internal state (to handle partial observability), which can update two types of knowledge: (1) how the world evolves independently of the agent, and (2) how the agent's actions affect the world

Model-based reflex agents-II

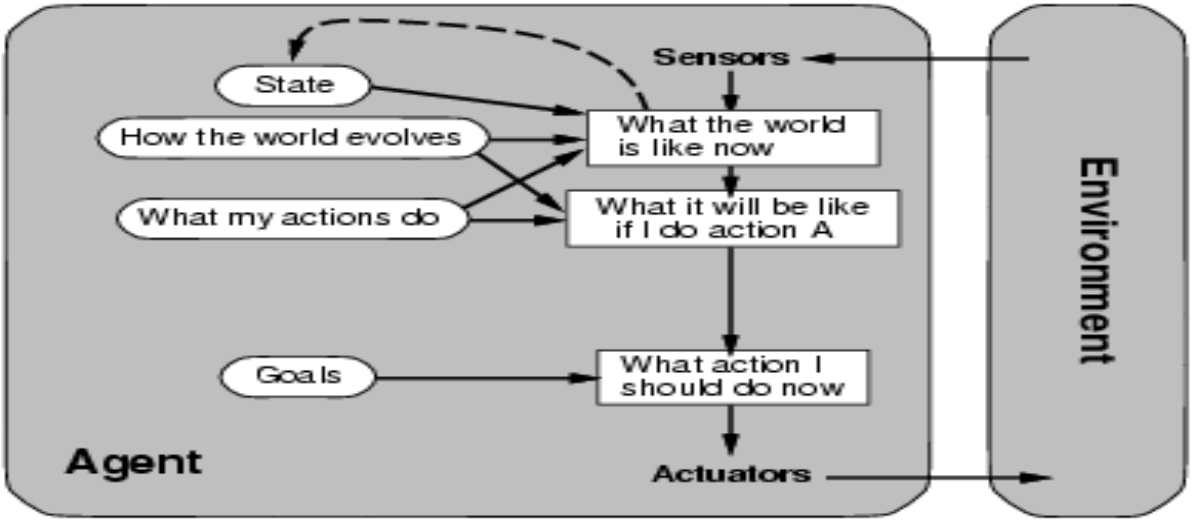
function REFLEX-AGENT-WITH-STATE(*percept*) returns an *action*

Static: *state*, a description of current world-state
rules, a set of condition-action rules
action, the most recent action, initially none

state <- UPDATE-STATE(*state*,*action*,*percept*)
rule <- RULE-MATCH(*state*, *rules*)
action <- RULE-ACTION(*rule*)
return *action*

- This agent uses an internal state to track aspects of the world not evident in current percept. It chooses an action in the same way as the reflex agent i.e. by finding a rule whose condition/state matches the current situation/state (but which is now defined by percept AND the stored internal state) & then doing the action associated with that rule

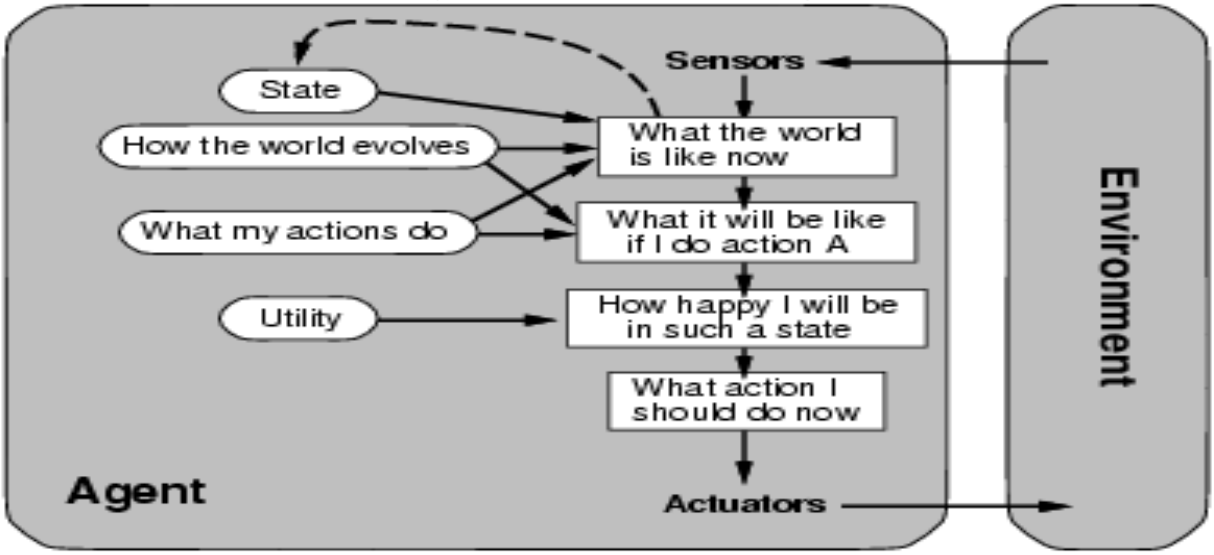
Goal-based agents



Rationale: Goal-based agent keeps track of the world state as well as a set of goals it is trying to achieve, and chooses/finds an action (or action sequence) that will eventually lead to the achievement of its goals

Choosing an action to achieve goal may involve search & planning

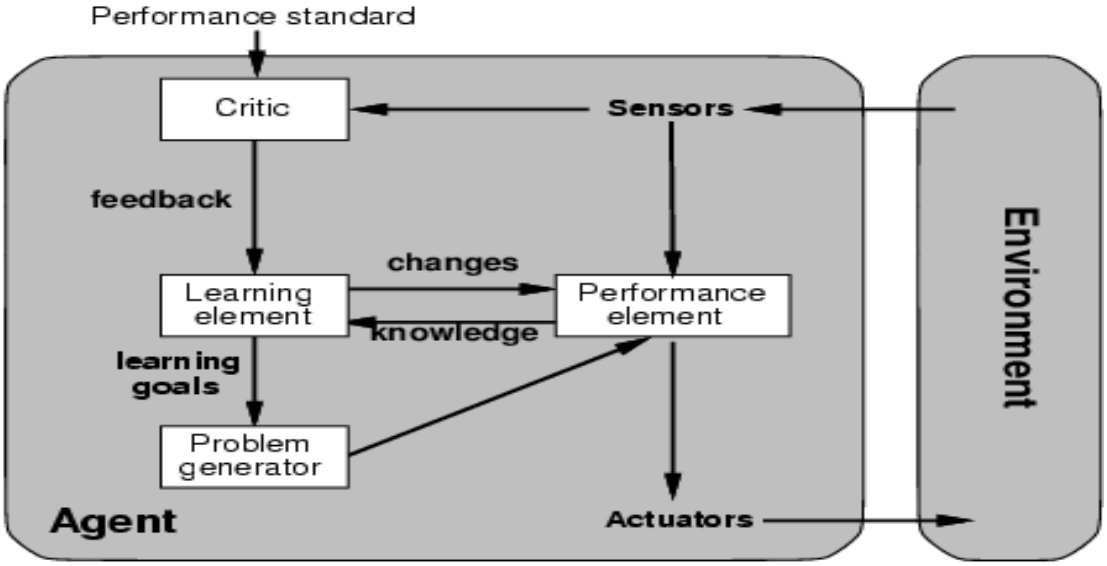
Utility-based agents



Model-based utility-based agent 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.

Rationale: Goals are not enough to generate high quality behaviour. Need to maximise agent's own expected happiness. Utility function maps states onto degree of happiness & can be used to specify trade-off between conflicting goals or to specify likelihood of achieving success against importance of goals.

Learning agents



All agents can improve their performance through learning.

•Performance element is the entire non-learning agent (selects external actions), and represents all previously considered Agents e.g. reflex, model/utility based etc.

•Learning element makes improvements to agent on basis of feedback from critic

•Problem generator may speed learning by suggesting actions that lead to new and informative experiences e.g. Exploration & discovery

Agent Design (Tut.1)

Tutorial 1 (on 5th Mar) - Q3:
For each of the following agents:
(i) Robot soccer player
(ii) Autonomous Mars Rover
Attempt the following:

(a) Develop a PEAS description of the task environment
(b) Characterise the environment according to the types given in the lecture (e.g. fully observable, deterministic etc.)
(c) Select a suitable agent design (e.g. reflex agent, model based agent etc.)

Q4.
While driving, what is the best policy:

- Always put your directional blinker on before turning?
- Never use your blinker
- Look in your mirrors and use your blinker only if you observe a car that can observe you?

What type of reasoning did you need to do to arrive at this policy (logical, goal-based or utility based)?
What type of agent design is necessary to carry out the policy (reflex, goal-based or utility based)?

More Questions for Tut.1 (Mon, 5 Mar)

Q5.
Define in your own words the following: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility based agent, learning agent

Q6.
What is the difference between a performance measure and a utility function.

Q7.
In Turing's original paper on AI (Turing, 1950, see link on course website), he predicts that, by the year 2000, a computer will have a 30% chance of passing a 5 minute Turing Test with an unskilled interrogator. What chance do you think a computer would have today? In another 50 years?