AGENT PROGRAMS
University of Malawi
Polytechnic
A. Taylor

Exam-like Questions

1. For each of the following activities, give a PEAS description of the task environment and characterize it in terms of the properties of environments.
1.Playing football
2.The TIC-TAC-TOE game
3.Shopping for used AI books on the Internet.
4.Playing a tennis match
5.Bidding on an item at an auction
6.Face-recognition agent

Skeleton Agent Program

- basic framework for an agent program

```
function SKELETON-AGENT(percept) returns
  action
  static: memory

  memory := UPDATE-MEMORY(memory, percept)
  action := CHOOSE-BEST-ACTION(memory)
  memory := UPDATE-MEMORY(memory, action)

return action
```


Table Agent Program

- agent program based on table lookup

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts, // initially empty sequence*
       table            // indexed by percept sequences
                  // initially fully specified

  append percept to the end of percepts
  action      := LOOKUP(percepts, table)

return action
```

Features of Table Lookup

- It is a simple way to specify a mapping from percepts to actions
- However, tables may become very large
  - almost all work done by the designer
  - no autonomy, all actions are predetermined
  - with well-designed and sufficiently complex tables, the agent may appear autonomous to an observer

Agent Program Types

- There are different ways of achieving the mapping from percepts to actions
- There are different levels of complexity

TYPES:
1)simple reflex agents
2)model-based agents
3)goal-based agents
4)utility-based agents
5)learning agents

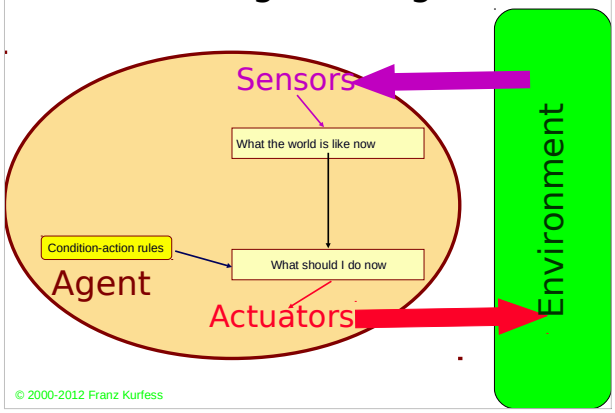
Simple Reflex Agent

- instead of specifying individual mappings in an explicit table, common input-output associations are recorded
- frequent method of specification is through condition-action rules
  - if percept then action
- similar to innate reflexes or learned responses in humans

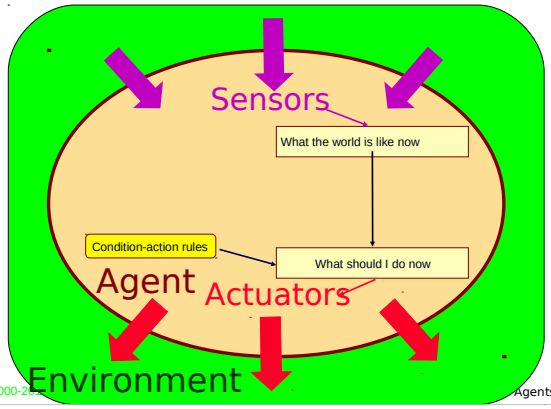
Simple reflex Agent

- A rule would specify that a if a certain condition is met, a certain action should be taken.
- A Simple Reflex Agent is typically employed when all the information of the current game state is directly observable, (eg: Chess, Checkers, Tic Tac Toe, Connect-Four) and the decision regarding the move only depends on the *current* state.
- When the agent does not need to remember any information of the past state to make a decision.
- Simple Reflex Agents can only operate on the *current* game state, and are unable to use information from older states, not even the directly previous one.

Reflex Agent Diagram



Reflex Agent Diagram 2



Reflex Agent Program

- application of simple rules to situations

```
function SIMPLE-REFLEX-AGENT(percept) returns
   action
   static: rules    //set of condition-action
   rules

   condition := INTERPRET-INPUT(percept)
   rule      := RULE-MATCH(condition, rules)
   action         := RULE-ACTION(rule)

return action
```


The Tic-Tax-Toe Game

- Write down the rules for a simple reflex agent that can play tic-tac-toe

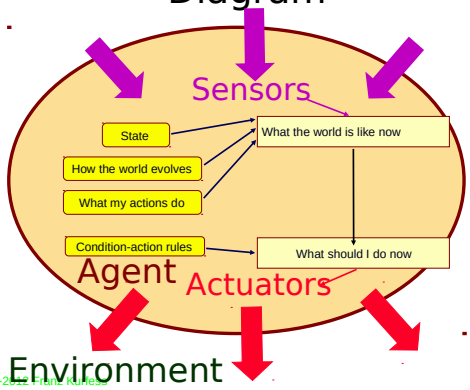
0		
	x	
	x	

- Eg., if (it is your turn to move) and (there are two zeros in a row) place an x on that row)

Model-Based Reflex Agent

- This is an improvement on the Simple Reflex Agent
- They operate in partially observed environment.
- These agents also perform action based on some condition like simple reflex agents.
- Unlike simple reflex agents, they record the history of percepts;
- an internal state maintains important information from previous percepts
  - sensors only provide a partial picture of the environment
- the internal states reflects the agents knowledge about the world. This knowledge is called a model

Model-Based Reflex Agent Diagram



Model-Based Reflex Agent Program

- application of simple rules to situations

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: rules     //set of condition-action rules
          state     //description of the current world state
          action    //most recent action, initially none

  state      := UPDATE-STATE(state, action, percept)
  rule := RULE-MATCH(state, rules)
  action     := RULE-ACTION[rule]
  return action
```

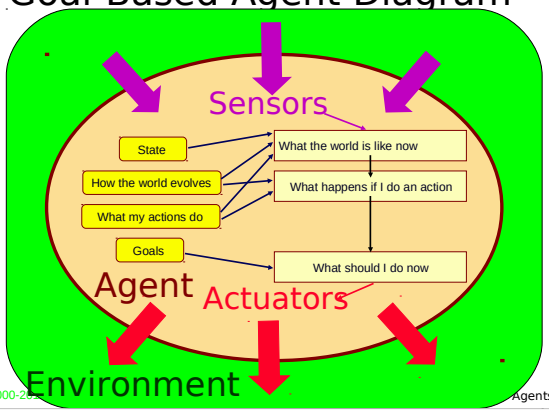

Goal Based Agents

- These agents do not make decision if condition is met alone but also see to it that these decisions are favorable in reaching their goals. They will choose among alternatives for reaching their goal unlike the other two agent types.

Goal-Based Agent

- the agent tries to reach a desirable state, the goal
- The outcomes of possible actions are considered with respect to the goal
  - easy when the results can be related to the goal after each action
  - in general, it can be difficult to attribute goal satisfaction results to individual actions
  - may require consideration of the future
    - what-if scenarios
    - search, reasoning or planning
- This is a greedy approach that may not lead to an optimal solution

Goal-Based Agent Diagram



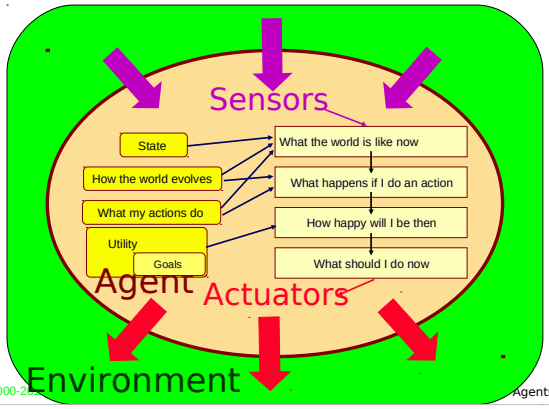
Utility-based Agents

- These agents are a type of goal-based agent with an additional functionality. These agents not only distinguish between a goal state and a non-goal state but also come equipped with a function telling the agent the degree of utility.

Utility-Based Agent

- The use of utility function
  - permits rational actions for more complex tasks
    - resolution of conflicts between goals (tradeoff)
    - multiple goals (likelihood of success, importance)
    - a utility function is necessary for rational behavior, but sometimes it is not made explicit

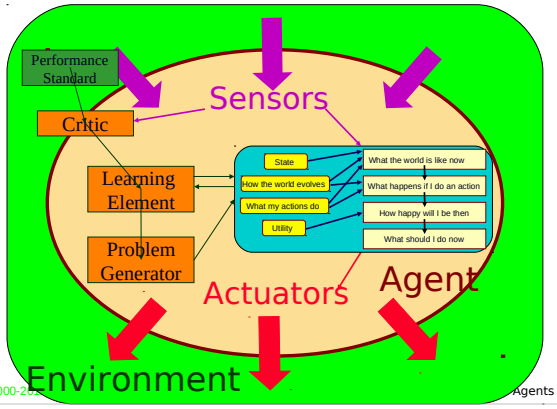
Utility-Based Agent Diagram



Learning Agent

- performance element
  - selects actions based on percepts, internal state, background knowledge
  - can be one of the previously described agents
- learning element
  - identifies improvements
- critic
  - provides feedback about the performance of the agent
  - can be external; sometimes part of the environment
- problem generator
  - suggests actions
  - required for novel solutions (creativity

Learning Agent Diagram



AI patrol game example – Tableau of Rules

- If **patrolling** and **no enemy in sight** then Patrol predefined path
- If **patrolling** and **enemy in sight**, switch mode from **patrolling** to **chasing**
- If **chasing** and **enemy in sight** , move towards enemy
- If **chasing** and **enemy not in sight** move towards **enemy's last location**
- If at **enemy's last location** and **enemy not in sight** and **cooldown timer** not started then **start cooldown timer**
- If at **enemy's last location** and **enemy not in sight** and **cooldown timer expired** then change mode to **returning to post**
- if **returning to post** and **enemy in sight** then set mode to **chasing**
- if **returning to post** and reached **patrolling path** and set mode to **patrolling**

AI patrol game example

- For the AI patrol game explain how each type of agent program will look like
- What are the differences between these different types of agents for this scenario? Advantages/ Disadvantages

E-commerce Soft-Agent Programs

Example 1

1)A shopping agent that purchases/chooses items based in the price of the product. This is a simple reflex that acts as soon as a product in a certain category has a desirable low price. There is no input from a ‘user’

E-commerce Soft-Agent Programs

Example 2

Mobile Intelligent Agent Architecture for E-Business where products are described by name and not by image or any other attribute.

The intelligent agent developed is a multi agent system functioning as Goal based Agent. The agents here mimic the functioning of a shopping human being using the Cellular phone or a PDA.

Goal based agent where not only the agent looks for condition to be met but also goal to be achieved which in our case is shopping the product of choice based on price and specification.

END