HW2 SOLUTIONS

Question 1(10 pts): (2 points each)

Please answer the following questions in 1 or 2 short sentences.

1. What does it mean for a search algorithm to be optimal?
   It means that it achieves its goal at the best possible path cost.

2. What does it mean for a search algorithm to be complete?
   It means that if there is a solution, it will find it.

3. What does it mean for a logical inference method to be sound?
   Assuming that the Knowledge Base has correct statements, it cannot achieve a truth value for a statement that is in contradiction with the real world using its rules correctly.

   (or)

   It derives only entailed sentences.

   (or)

   Whenever the inference procedure derives 'a' then KB also entails 'a'.

   (or)

   A sound inference process is one that derives true conclusions given true premises.

4. What doe sit mean for a logical inference method to be complete?
   If a statement is true, it can prove it using its inference rules.

   (or)

   It can prove any sentence that is entailed by the KB

   (or)

   Whenever KB entails 'a' then the inference procedure also infers 'a'

   (or)

   An inference procedure is complete if it can derive all true conclusions from a set of premises.

5. Consider an intelligent alarm system that has the following sentence in its knowledge base:
((Motion => Intruder) v (Noise => Intruder))

Assume that the propositions Motion and Noise take on their respective truth-values, directly from information provided by the agent's sensors. Given this sentence, show precisely the circumstances under which this agent can conclude the presence of an intruder (i.e. conclude whether or not the proposition Intruder is true).

(Motion => Intruder) ↔ (¬Motion v Intruder)

(Noise => Intruder) ↔ (¬Noise v Intruder)

((Motion => Intruder) v (Noise => Intruder)) ↔ (¬Motion v Intruder) v (¬Noise v Intruder)

↔ (¬Motion v Intruder v ¬Noise v Intruder)

↔ (¬Motion v ¬Noise v Intruder)

↔ (¬(Motion & Noise) v Intruder)

↔ ((Motion & Noise) => Intruder)

Therefore, there has to be Noise and Motion for the Alarm to conclude that there is an Intruder!

Note – Truth table is not the way to go about solving the problem and may lead you to wrong answers.

Question2 (10 pts):

Assume that we have an A^* search with the evaluation function fcost = g + Random( ) * h, where g is the actual cost of the path thus far, h is an admissible non-overestimating heuristic, and Random returns random real values between 0 and 1. Consider the behavior of an A^* search with this evaluation function as compared to the same search without the multiplication by Random( ). Be sure to discuss the following topics:

1. The qualitative effect of the multiplication by Random( ) on A^*'s performance (comparison to other search algorithms would be appropriate here).
2. The effect it has on the optimality of A^*.

Answer –

There are 4 possible cases –

(1) if random() is computed once per search, then the heuristic is still admissible and the search still optimal. Because the random value is smaller than 1, however, the heuristic will not be as efficient as it could be (will queue-up a bunch of additional nodes that a heuristic closer to the true cost-to-goal would have avoided)

2) if random() is run once per depth in the search, same as above

3) if random() is run once per loop of the general-search algorithm (i.e., once per node expansion, using the same random value for all successors created during that expansion), we can't say anything. Most likely the search becomes suboptimal. But whether or not that will happen depends on the random values.

4) if random() is run each time f() is called, idem, but it seems more obvious in this case to find an example of loss of optimality.

In case (1) if random turns out to be 0 we have uniform-cost. **(not greedy search)**

So an answer could contain the following

For part 1 -

- analysis of behavior as Random() tends to 0 and tends to 1 (note – you cannot just analyze two cases – 0 ,1 : we are interested in your opinion for values between 0 and 1)
- Comparison (specially in cases 1 and 2 – optimal solution) - it should be with Uniform cost / A* ideally but since nothing has been specified – other answers have been accepted as long as they are right.

For Part 2 -

- Discussion on optimality. It isn't just enough to say optimal / not optimal – your answer must atleast indicate the case (we are looking for atleast one of the above 4 cases) you have chosen and your analysis of that case. Note – you may even give an example here – but your argument and/or example must be consistent with your claim of optimality/non-optimality – as is mentioned in each case above.

Question 4(40 pts):

Agent Specification –

You are going to specify three agents that you would find in a restaurant, the Customer, Waiter, and Cook. The agents will have the same architecture as specified in the discussion board topic, "Intelligent Agents" Project. Here is an overview of the architecture:

1) An agent is an independent process that receives percepts from the outside world through messages. It will map the messages into internal state (perhaps with variables, perhaps with queues).

2) The scheduling thread will select an action for execution when the agent is not busy and, of course, only if there is something for it to do. It might decide to terminate a running action and do something else.

3) An action is some task the agent is capable of doing. An agent can only carry out one task at a time. If you find the agent needs to do two actions concurrently, maybe you need two agents. Since our actions do not control real-world outputs, they must send messages to other agents as their only output.

So, for each agent:

- Design the messages and how they are mapped to internal state.
- Describe the rules the scheduler uses in deciding what action the agent should perform.
- Design the actions of each agent, what they do, what internal state they change, what messages they send.

For this assignment assume the restaurant layout simply has some number of tables, each of which is occupied or not occupied.

Elaborate 2 scenarios: 1) Show what would occur if one customer comes in. (2) Show what would happen if 2 customers come in simultaneously.

In a scenario, you show in sequential form, how the agents would behave. You must show how the agent decides what to do (the scheduler decisions), and then what it does.

Here is a possible beginning to the first scenario:

1) Customer1 Agent:
   - Customer1.hungry=True; //initial state
   - if hungry then queueAction(GotoRestaurant); //scheduler rule
   - Executing GotoRestaurant:
     Customer1.location=Restaurant;
     Message to Waiter- PartyOfOne(name=Customer1)
2) Waiter1 Agent:
   - Receives message: PartyOfOne(name=Customer1) which executes
     waitingCustomers.add(Customer1)
   - if tables.isTableEmpty() then queueAction(GreetAndSeat(Customer1))
     (if no table were empty, then the action might be GreetAndWait(Customer1,10minutes);
   - etc.
3) etc.

Don't worry about movement actions. As you can see from the above sample, the GotoRestaurant action just assumes the customer is there. You might set some state in the Customer agent that stores its location.

Don't be overly detailed. Try to do everything using object-oriented pseudo-code. [For students who are implementing the agents, you might as well design Java classes.]

Solution: From Prof Dave –

Please note the following:
1) I hope the students do not try to be any more precise than I do.
2) Note how simple I made my scheduler. For this assignment we are not trying for tough logic.
3) I didn't include a scenario, but it should be easy to infer what it will look like from my print statements.
4) In the case of one customer, the waiter will execute this actions in the sequence given. Only if there are multiple customers, will his actions be intermixed.

I have not included a cook or customer agent. From the messages of the Waiter agent, you can infer the message of the others since the waiter is in the middle.

For the purposes of this homework, the cook would be trivial. One input message that takes an order and puts it on a queue. One scheduler rule to Cook food. One task to cook food which sends a message to Waiter when done.

The Customer is almost as simple. Maybe one long task which has various waits in it as the waiter does his job.

Waiter agent Specification – you can write those for the cook and customer on the same lines.

- **Waiter**
  - ***Data Structures***

    Customer – Name, number in party, order, State (one of Waiting, Seated, WaitingToOrder, ReadyToOrder, Ordered, WaitingForFood, Eating, DoneEating, NeedsCheck, HasCheck, Paid)
    myCustomers queue of Customer
    myOrders queue of Customer names whose food is ready to serve
    Table – id, number of people who can sit there;
    Restaurant array - Table, Customer
    Waiter array – waiter id and tables assignment
    Busboy array – busboy id and tables assignment

  - ***Messages***

    - **From Customer – IamInYourRestaurant(name, numberInParty)**
      Set customer.State = Waiting;
      Put customer into myCustomer array

    - **From a Waiter or Manager: CustomerAtYourTable(customer, table)**
      Set CustomerState==Seated;
      Update Restaurant(table,customer)
      Put customer into myCustomer array

    - **From Customer: IamReadyToOrder(customer, table)**
      Set Customer.State = ReadyToOrder

    - **From Customer: HereIsMyOrder(order)**
      Set Customer.State = Ordered;
      set Customer.order =order.

    - **From Cook: OrderReady(customer)**
      put customer on myOrders queue

- **From Customer: HereIsMyMoney(customer, money)**
  set Customer.State = Paid

- **From Customer: GoodBye(customer)**
  remove customer from myCustomers.
  Update Restaurant(table) = empty.

- ***Scheduler (for now assume Scheduler only acts when no action is running)***
  - if (for some customer in myCustomer) customer.state==Waiting) then Run(GreetAndSeatCustomer (customer)
  - if (for some customer in myCustomer) customer.state==Seated) then Run (IntroduceYourself)
  - if (for some customer in myCustomer) customer.state== ReadyToOrder) then Run (TakeOrder)
  - if (for some customer in myCustomer) customer.state== WaitingForFood and customer is in myOrders queue then Run (ServeFood)
  - if (for some customer in myCustomer) customer.state== Paid) then Run (SayGoodByeToCustomer)

- ***Actions***

  - **GreetAndSeatCustomer**
    Print "Customer %name in party of %numberInParty has arrived."
    table = findEmptyTable(); //need code for when all tables are full
    SendMessage(Customer, "FollowMeToTable(table));
    GoToTable(table);
    SendMessage(Customer,"PleaseHaveASeat()); //customer at table
    updateRestaurant(table, customer); //stores customer info into Restaurant array at table
    customer.State = Seated;
    waiter = findWaiterResponsibleFor(table);
    SendMessage(waiter, CustomerAtYourTable(customer, table); //this message may be to yourself
    Remove customer from myCustomer queue

  - **IntroduceYourself**
    Print "Hello %customerName, my name is %waiterName, I'll be your waiter tonight."
    SendMessage(customer, HereIsYourMenu());
    Customer.State = WaitingToOrder;}
    setTimer(5minutes, tableNumber); //When timer goes off, set Customer.State = ReadyToOrder

  - **TakeOrder**
    Print "Hello %customerName, what would you like?."
    SendMessage(customer, "WhatIsYourOrder()")
    Wait until customer.state==Ordered.
    Print "Thank you for your order;
    SendMessage(cook, HereIsOrder(customer,order)
    Set Customer.state= WaitingForFood;

  - **ServeFood**
    Print "Hello %customerName, Here is your food."
    Remove this order from myOrders.

Set Customer.state= Eating;
SendMessage(customer, HereIsYourCheck(check);
Set Customer.state= HasCheck;

- **SayGoodByeToCustomer**

  Print "Hello %customerName, Hope you enjoyed your stay."
  //Nothing else to do in this task. The GoodBye message will cause cleanup of data structures.

(Waiter =16 points; customer = 8 points; cook = 8 points; each scenario = 4 points; each of the three agents require 4 part specifications – data structures / actions / messages / scheduler – each carrying 1/4th the points for that agent)

Question 5 (50 pts): Programming

Consider First-order-logic without quantifiers. You will be dealing with proposition variables and logical connectives, implication, equivalence and, of course, parenthesis. You have to write a program that can show whether a given sentence is either valid or unsatisfiable or neither. You have to account for precedence and associativity.

Input: You will be given a text file called **problem.txt**. The file will be located in the current directory. The first line contains an integer **n** specifying the number of problems in the file. Each line will contain a problem. In each line there is a sentence composed of propositional variables, logical connectives, implication, equivalence and parenthesis. The sample file **problem.txt,** is as follows

Line number
1. 3
2. P && (Q && R) <=> (P && Q) && R
3. Smoke => Fire
4. P && ~P

Output: All the output should be written to a file called **solution.txt** in the current directory. For each problem in **problem.txt**, there will be a **corresponding line** in **solution.txt** and it will contain your program output for that sentence: one of the following three: **valid, unsatisfiable** or **neither**.

Line number
1. valid
2. neither
3. unsatisfiable

Caution: Please follow the following things
1. Do not expect any command line argument(s).
2. There can be any number of white spaces between fields in the input file.
3. Assume that only space or tab will be used for separating the fields in the input file.
4. There will be no line number or "***Line number***" string in **problem.txt**. Don't look for them.

Test cases

10

((P=>Q)<=>(~P OR Q))	Valid
((P <=> Q) <=> (Q <=> P))	Valid
((P=>Q) <=> (Q => P))	Neither
(P And ~P)	Unsatisfiable
(P And Q)	Neither
(P Or Q)	Neither
((P <=> Q) <=> (~P <=> ~Q))	Valid
(((P=> Q) And (Q=>R))=>(P=>R))	Valid
P	Neither
P=>~P	Neither

(10 cases = 50 points with 5 points per case)