TDDC17: Artificial Intelligence

Previous exam questions and answers

SIMON AND NEWELL ARTICLE

Q1:

The following questions pertain to the course article by Newell and Simon entitled *Computer Science as an Empirical Enquiry: Symbols and Search.*

(a) What is a *physical symbol system* (PSS) and what does it consist of?

A PSS takes physical patterns (**symbols**), combines them into structures (**expressions**) and manipulates them (using **processes**) into new structures.

(b) In the article, the authors discuss two notions central to the structures of expressions, symbols and objects which are used in PSS's: 1) designation and 2) interpretation. Provide a brief description of each of these notions in the context of a PSS.

Designation: An expression designates an object if, given the expression, the system can either affect the object itself or behave in ways depending on the object.

Interpretation: The system can interpret an expression if the expression designates a process and if, given the expression, the system can carry out the process.

(b) What does the Physical Symbol System Hypothesis state?

That a PSS has the necessary and sufficient means for general intelligence.

(c) What does the Heuristical Search Hypothesis state?

A heuristical search my be applied to a PSS for general intelligent decision making and and problem solving.

(d) Do you think the Physical Symbol System Hypothesis is true, or somewhere between? Provide reasonable justification for your opinion.

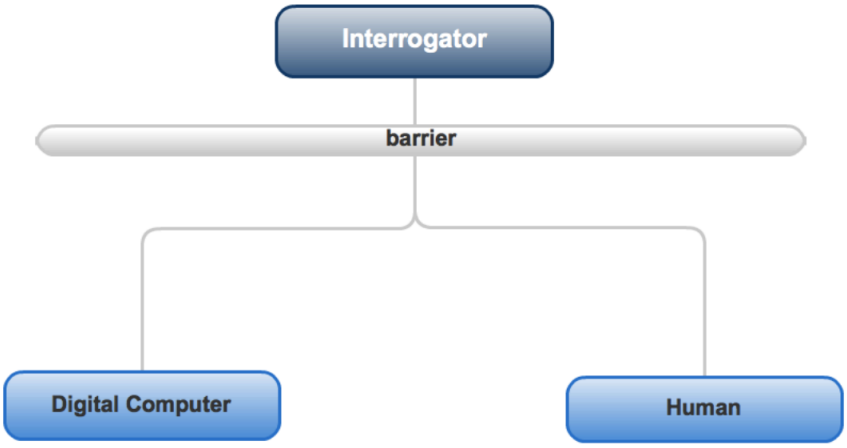
I think that since a PSS can create new structures of symbols by manipulating expressions, similar to what we do as humans when we learn new information, it should in theory be possible for a PSS to gain any new knowledge given the right building blocks.

ALAN TURING ARTICLE

Q1:

Alan Turing proposed the Turing Test as an operational definition of intelligence.

(a) Describe the Turing Test using your own diagram and explanations.[2p]



The turing test is a test of a machine's ability to exhibit intelligent behavior equivalent to, indistinguishable from, a human. Turing proposed that a human evaluator, or interrogator, would judge natural language conversations between a human and a machine that is designed to generate human-like responses. The interrogator would be aware that one of a parties would be a machine and the other a human.

If the evaluator cannot reliably tell the machine from the human (Turing originally suggested that the machine would convince a human 70% of the time after five minutes of conversation), the machine is said to have passed the test.

(b) Do you believe this is an adequate test for machine intelligence? Justify your answer.[1p]

The problem I see with using the Turing test as a means of measuring machine intelligence is that the Turing test only measures the machines ability to simulate consciousness, and to appear human. This doesn't necessarily mean that the machine is intelligent, in my mind. A machine that passes the test could implement a (very) long list of conversational rules to fool the human interrogator to believe that it is in fact thinking, even though this is far from the case.

If the machine is too good in answering questions that are considered hard for a human, such as complex combinatorial questions, then the human will know that the entity is in fact a computer, even though it is answering a hard question intelligently.

CONCEPTUAL QUESTIONS ABOUT PARTS AND TYPES OF INTELLIGENT AGENTS

Q1:

Based on the material in Chapter 2 of the course book, define succinctly in your own words, the following terms:

(a) Agent, Agent Function and Agent Program.

An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
The agents mapping from any given perception sequence to an action are determined with the **agent function**.
The actual implementation of the agent function within the agent is called the **agent program**. The agent function is just a mathematical representation of the agent program.

(b) Performance Measure, Rationality and Autonomy

A **rational agent** is one that executes the right action given a certain percept sequence. What constitutes as ”the right thing” is determined by the consequences of the executing the given actions. Executing actions will lead to the environment going through different states, where each state is more or less desirable. How desirable a sequence of states are is determined by the **performance measure**.
What is rational is determined by:

- The performance measure.
- The agents prior knowledge of the environment.
- The actions the agent can perform.
- The agents percept sequence to date.

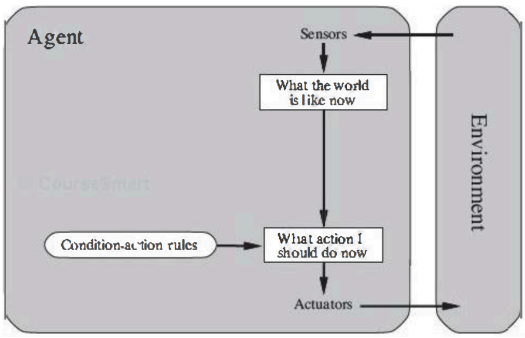
Definition of a rational agent:

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.

Autonomy is when the agent is fully self-controlled and not controlled by another entity.

(c) Reflex Agent. Also provide a schematic diagram of such an agent.

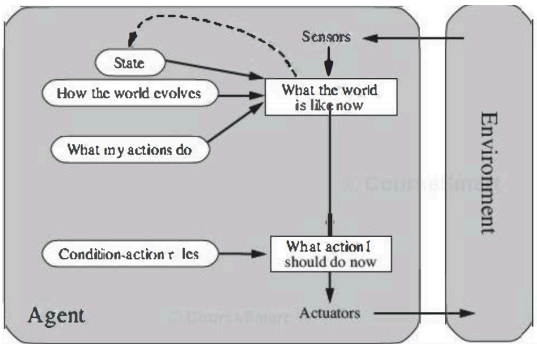
The r**eflex-agent** selects its action on the basis on the current perception, disregarding the rest of its percept history.



(d) Model-based Agent. Also provide a schematic diagram of such an agent.

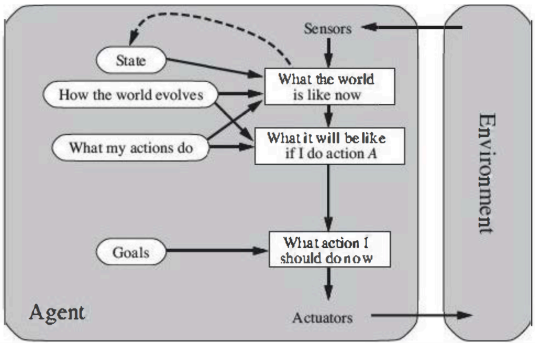
The Model-based Agent uses previous knowledge about the world to get a better understanding about how the world works. The new perception is combined with the old internal state to build a new description of the current state, based

on the agents understanding about the world.



(e) Goal-based Agent. Also provide a schematic diagram of such an agent.

The Goal-based Agent implements a goal function that gives the Agent information about what states are more desirable than others based on the goal it's working towards. **Example**: The Reflex-based Agent breaks when it sees a red light. The Goal-based Agent can in principle reason that if the car in front has its brake lights on it will slow down. Given the way the world usually evolves (Model-based approach) the car in front will then slow down, and the only way to achieve the agents goal of not hitting the car in front is to brake.



A* SEARCH

Q1:

A* search is the most widely-known form of best-first search. The following questions pertain to A* search:

(a) Explain what an admissible heuristic function is using the notation and descriptions in (c). [1p]

A heuristic function is used to rank alternatives in search algorithms at each branching step based on available information to decide which branch to follow. This is used to get a better sense of what paths are more likely to lead us to the goal. A heuristic function is said to be **admissible** if it never overestimates the cost of reaching goal. I.e the cost it estimates to reach the goal can't be higher than the actual lowest possible cost from the current point in the path.

(b) Suppose a robot is searching for a path from one location to another in a rectangular grid of locations in which there are arcs between adjacent pairs of locations and the arcs only go in north-south (south-north) and east-west (west-east) directions. Furthermore, assume that the robot can only travel on these arcs and that some of these arcs have obstructions which prevent passage across such arcs.

The Mahattan distance between two locations is the shortest distance between the locations ignoring obstructions. Is the Manhattan distance in the example above an admissible heuristic? Justify your answer explicitly. [2p]

The manhattan distance in a grid world is an admissible heuristic since it is always the shortest path from the current state (location) to the goal state. In the case that the path contains no obstacles then the estimated cost using the manhattan distance will be the same as the actual lowest cost of reaching the goal state, but the estimation using the manhattan distance will never be higher than the actual lowest cost.

(c) Let h(n) be the estimated cost of the cheapest path from a node n to the goal. Let g(n) be the path cost from the start node n_0 to n. Let f(n) = g(n) + h(n) be the estimated cost of the cheapest solution through n.

Provide a general proof that A* using tree-search is optimal if h(n) is admissible. If possible, use a diagram to structure the proof. [2p]

Let node G be a suboptimal goal node found when executing the search.
Let C* be the lowest cost of reaching the optimal goal node G*.
Since G is a goal node, the heuristic value of G is h(G) = 0, and f(G) will therefore be f(G) = g(G) + h(G) = g(G).
Since G is a sub-optimal goal the cost of reaching it will be bigger than the cost of reaching the optimal goal => f(G) > C*.
The cost of reaching the optimal goal from the current state n, is
f(n) = g(n) + h(n) - and since h(n) is an admissible heuristic, and therefore won't overestimate the cost of reaching the goal, then f(n) <= C* < f(G).
Because of this G will never be expanded and A* is optimal.

LOGIC

Q1:

Modeling action and change in incompletely represented, dynamic world is a central problem in knowledge representation. The following questions pertain to reasoning about action and change.

(a) What is Temporal Action Logic? Explain by describing the ontology used in the formalism, that is, what is taken to exist, and what notation is used in the logical language to represent those things that are taken to exist. [2p]

Temporal logic uses **facts**, **objects**, **relations** and **times** to describe things that exists.

(b) What is the frame problem? Use the Wumpus world to provide a concrete example of the problem.Represent the problem by representing an initial timepoint, an action and the result of the action using the TAL notation (either with macros or without). [2p]

The frame problem is the problem of representing the fact that most objects stay in the same state as long as they are not moved by an action.

Example:

[t1] at(agent,1,1)
[t1] at(gold, 2,3)
[t1, t2] move(agent,1,1,1,2)
[t2] at(agent,1,2)
[t2] at(gold,2,3)?

(c) What is nonmonotonic logic? How can it be used to provide solutions to the frame problem?

Nonmonotonic logic is a way of making cautious conclusions about how the world will remain until we are proven otherwise. For example that the lamp will remain on the kitchen table as long as nobody moves it. Reasoning about default.

The non-monotonic logic can say that until we get evidence that the gold in (b) is moved to a new position we assume that it will remain in the position that we saw it the last time.

Q2:
Modeling actions and change in incompletely represented, dynamic worlds is a central problem in knowledge representation. The following questions pertain to reasoning about action and change.

(a) What is the frame problem? Use the Wumpus world to provide a concrete example of the problem. [2p]

The frame problem is the problem of representing the fact that most objects stay in the same state as long as they are not moved by an action.

Example:
[t1] at(agent,1,1)
[t1] at(gold, 2,3)
[t1, t2] move(agent,1,1,1,2)
[t2] at(agent,1,2)
[t2] at(gold,2,3)?

(b) What is the ramification problem? Use the Wumpus world to provide a concrete example of the problem. [2p]

The ramification problem is the problem of representing how actions might affect other objects. For instance how walking to a new position will make the position of the keys in the pockets change as well.

Example:
[t1] at(agent,1,1)
[t1] inpocket(agent, keys)
[t1, t2] move(agent,1,1,1,2)
[t2] at(agent,1,2)
[t2] inpocket(agent, keys)?

(c) What is nonmonotonic logic? How can it be used to provide solutions to the frame problem? [2p]

Nonmonotonic logic is a way of making cautious conclusions about how the world will remain until we are proven otherwise. For example that the lamp will remain on the kitchen table as long as nobody moves it. Reasoning about default.

The non-monotonic logic can say that until we get evidence that the gold in (b) is moved to a new position we assume that it will remain in the position that we saw it the last time.

CONSTRAINT SATISFACTION

Q1:

Constraint satisfaction problems consists of a set of variables, a value domain…

(a) What is the Forward Checking technique?

Forward checking is an improved version of the backtracking algorithm. The difference between backtracking and backtracking with forward checking is that forward checking will maintain arc consistency between the current variable and the uninstantiated variables. This way, the current variable can't take on a value that will cause the uninstantiated variables to get an empty domain.

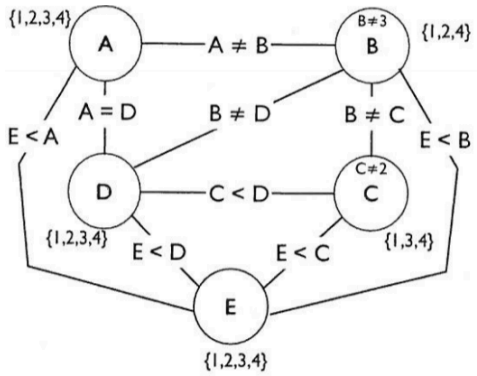
(b) What is arc consistency? Provide a definition.

A variable in a CSP is arc-consistent if every value in it's own domain satisfies the variables binary constraints. So Xi is arc consistent with respect to Xj if for every value in the domain Di it's possible to find a value in domain Dj that satisfies the binary constraint on the arc (Xi, Xj).

(c) Provide a constraint graph that is arc consistent but globally inconsistent.

If we have three variables X1, X2 and X3 that can each take on the values Dx={1,2} and they are all connected to each other. Then you can always find a value for each variable that satisfies arc-consistency with respect to it's two neighbors, but it's easy to see that it's impossible to find separate values for all three variables.

(d) Make the constraint graph in figure 1 arc-consistent using the AC-3 algorithm. For the answer, only the resulting consistent bindings for each variable in the constraint graph output by the AC-3 algorithm is needed.



function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  inputs: *csp*, a binary CSP with components (X, D, C)
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add (X_k, X_i) to *queue*
  return *true*

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i
  revised ← *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised ← *true*
  return *revised*

Answer: A={4}, B={2}, C={3}, D={4},E={1}

Q2:

Constraint satisfaction problems consist of a set of variables, a value domain for each variable and a set of constraints. A solution to a CS problem is a consistent set of bindings to the variables that satisfy the contraints. A standard backtracking search algorithm can be used to find solutions to CS problems.
In the simplest case, the algorithm would choose variables to bind and values in the variable's domain to be bound to a variable in an arbitrary manner as the search tree is generated. This is inefficient and there are a number of strategies which can improve the search. Describe the following three strategies:

(a) Minimum remaining value heuristic (MRV).
MRV chooses the variable with the least amount of values that meets the constraints.
(b) Degree heuristic. [1p]
Degree heuristic chooses the variable with the most constraints.
(c) Least constraining value heuristic. [1p]
Least constraining value chooses the value for a variable which constrains its neighbors the least.

Constraint propagation is the general term for propagating constraints on one variable onto other variables. Describe the following:

(d) What is the Forward Checking technique? [1p]
The forward checking technique propagates the constraints to a variables neighbors after a value is chosen for that variable. If choosing a value for the variable results in an empty set for one of the neighbors then the algorithm will backtrack and try another solution.
(e) What is arc consistency? [1p]
Arc consistency means that for variable X and Y, and for every value in the domain Dx, there is at least one value in the domain Dy that satisfies the constraint between X and Y.

BAYESIAN NETWORKS

3. Consider the following example:

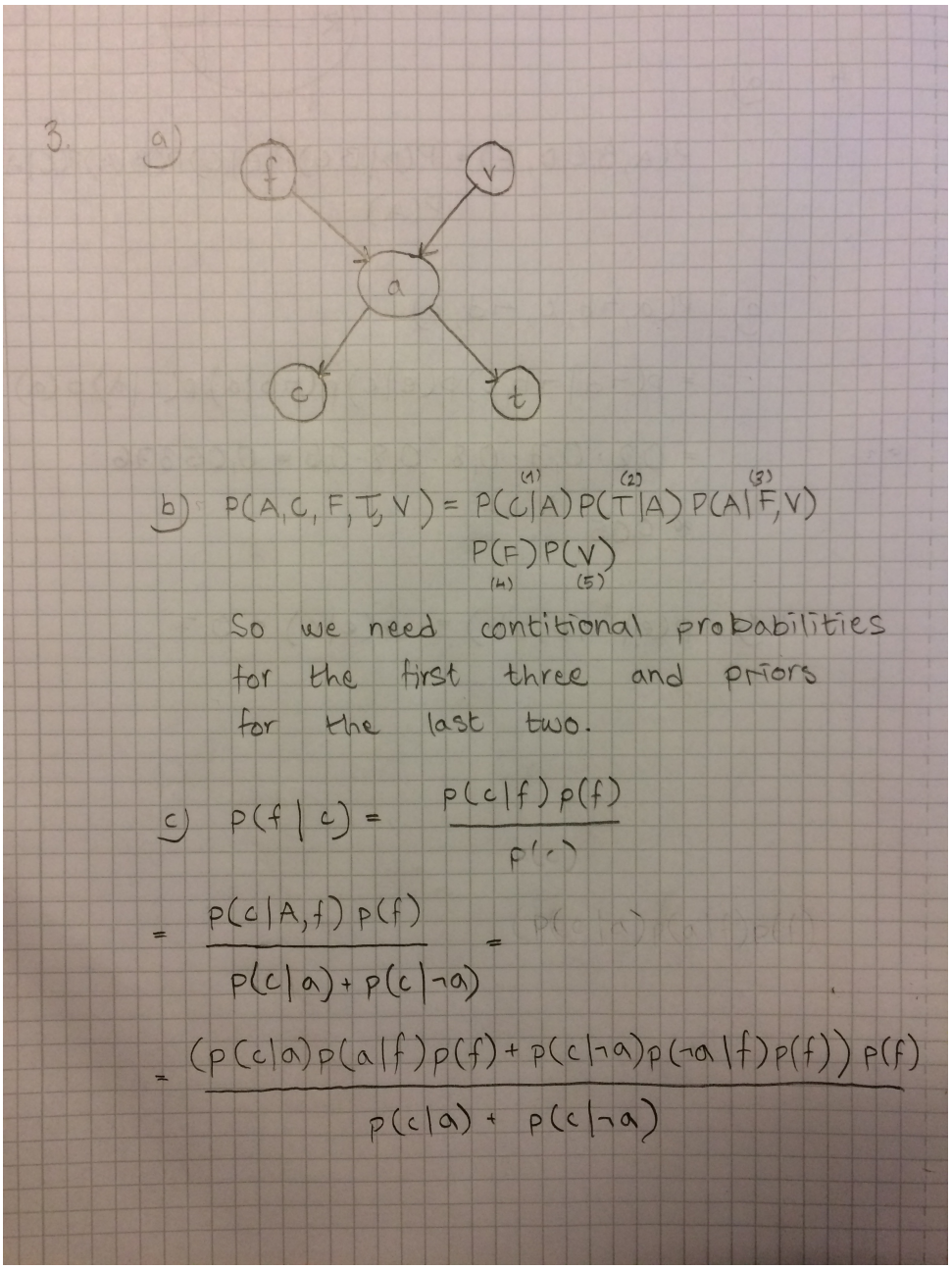
   The fire alarm in a building can go off if there is a fire in the building or if the alarm is tampered with by vandels. If the fire alarm goes off, this can cause crowds to gather at the front of the building and fire trucks to arrive.

   (a) Represent these causal links in a Bayesian network. Let a stand for "alarm sounds", c for "crowd gathers", f for "fire exists", t for truck arrives", and v for "vandalism exists". **[2p]**

   (b) Given the independence assumptions implicit in the Bayesian network, what are the conditional (or prior) probabilities that need to be specified to fully determine the joint probability distribution? In other words, what are the conditional tables associated with each node? **[2p]**

   (c) Suppose there is a crowd in front of the building one day but that no fire trucks arrive. Given this, what is the probability that there is a fire, expressed as some function of the conditional (or prior) probabilities? **[2p]**

   - Appendix 2 provides you with some help in answering these questions.



4. Use the Bayesian network in Figure 1 together with the conditional probability tables below to answer the following questions. Appendix 2 may be helpful to use.

   (a) Write the formula for the full joint probability distribution $P(A, B, C, D, E)$ in terms of (conditional) probabilities derived from the bayesian network below. **[1p]**

   (b) $P(a, \neg b, c, \neg d, e)$ **[1p]**

   (c) $P(e \mid a, c, \neg b)$ **[2p]**

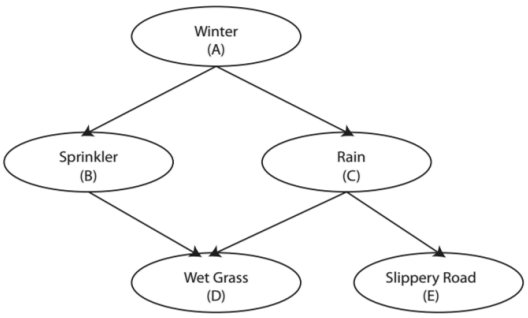


Figure 1: Bayesian Network Example

A	P(A)
T	.6
F	.4

A	B	$P(B \mid A)$
T	T	.2
T	F	.8
F	T	.75
F	F	.25

A	C	$P(C \mid A)$
T	T	.8
T	F	.2
F	T	.1
F	F	.9

B	C	D	$P(D \mid B, C)$
T	T	T	.95
T	T	F	.05
T	F	T	.9
T	F	F	.1
F	T	T	.8
F	T	F	.2
F	F	T	0
F	F	F	1

C	E	$P(E \mid C)$
T	T	.7
T	F	.3
F	T	0
F	F	1

4

a)

$$P(A,B,C,D,E) = \frac{P(D|B,C)P(E|C)P(B|A)P(C|A)}{P(A)}$$

b) $P(a, \neg b, c, \neg d, e)$

$= p(\neg d|\neg b, c)\, p(e|c)\, p(\neg b|a)\, p(c|a)\, p(a)$

$= 0{,}2 \cdot 0{,}7 \cdot 0{,}8 \cdot 0{,}8 \cdot 0{,}6 = 0{,}05376$

$\approx 0{,}05$

c)

$\alpha(p(e|a,c,\neg b,d) + p(e|a,c,\neg b,\neg d))$

$= \alpha\, p(e|c)\, p(c|a)\, p(\neg b|a)\, p(a)\,(p(d|\neg b,c) + p(\neg d|\neg b,c))$

$= \alpha\, 0{,}7 \cdot 0{,}8 \cdot 0{,}8 \cdot 0{,}6\,(0{,}8 + 0{,}2)$

$= \alpha \cdot 0{,}7 \cdot 0{,}8^2 \cdot 0{,}6 \approx \alpha \cdot 0{,}269$

$\alpha(p(\neg e|a,c,\neg b,d) + p(\neg e|a,c,\neg b,\neg d))$

$= \alpha\, p(\neg e|c)\, p(c|a)\, p(\neg b|a)\, p(a)\,(p(d|\neg b,c) + p(\neg d|\neg b,c))$

$= \alpha\, 0{,}3 \cdot 0{,}8 \cdot 0{,}8 \cdot 0{,}6\,(0{,}8 + 0{,}2)$

$= \alpha \cdot 0{,}115$

$\Rightarrow \alpha = \frac{1}{0{,}115 + 0{,}269} \approx 2{,}60$

So $p(e|a,c,\neg b) = 2{,}60 \cdot 0{,}269 \approx 0{,}7$

4. Consider the following example:

   Aching elbows and aching hands may be the result of arthritis. Arthritis is also a possible cause of tennis elbow, which in turn may cause aching elbows. Dishpan hands may also cause aching hands.

   (a) Represent these causal links in a Bayesian network. Let ar stand for "arthritis", ah for "aching hands", ae for "aching elbow", te for "tennis elbow", and dh for "dishpan hands". **[2p]**

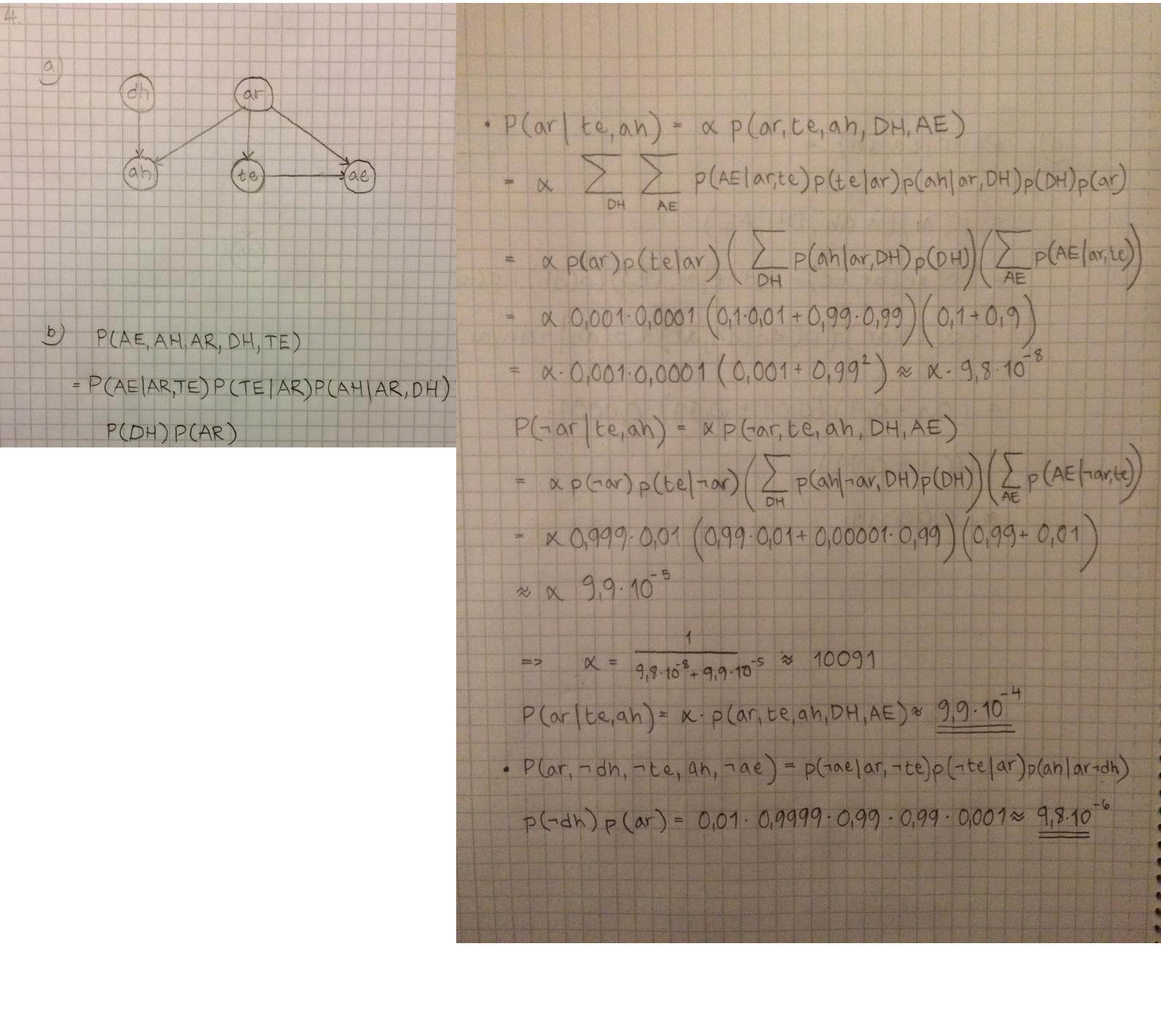
   (b) Given the independence assumptions implicit in the Bayesian network, write the formula for the full joint probability distribution over all five variables? **[2p]**

   (c) Compute the following probabilities using the formula for the full joint probability distribution and the probabilities below:

   - $P(ar \mid te, ah)$**[1p]**
   - $P(ar, \neg dh, \neg, te, ah, \neg ae)$**[1p]**
   - Appendix 2 provides you with some help in answering these questions.

Table 1: probabilities for question 5.

$P(ah \mid ar, dh) = P(ae \mid ar, te) = 0.1$
$P(ah \mid ar, \neg dh) = P(ae \mid ar, \neg te) = 0.99$
$P(ah \mid \neg ar, dh) = P(ae \mid \neg ar, te) = 0.99$
$P(ah \mid \neg ar, \neg dh) = P(ae \mid \neg ar, \neg te) = 0.00001$
$P(te \mid ar) = 0.0001$
$P(te \mid \neg ar) = 0.01$
$P(ar) = 0.001$
$P(dh) = 0.01$



PLANNING

Q1:

The following questions pertain to STRIPS-based planning. In order to use STRIPS one requires a language expressive enough to describe a wide variety of problems but restrictive enough to allow efficient algorithms to operate over it. The STRIPS language is the basic representation language for classical planners. Using terminology from logic (literals, ground terms, function free terms, etc.), describe how the following are represented in the STRIPS planning language:

(a) State representation [1p]

Each state is represented as a conjunction of literals (e.g On(Box1, Box2)) that are ground, functionless atoms. For example, Poor 'and' Unknown might represent the state of a hapless agent, and a state in a package delivery problem might be At(Truck1, Melbourne) 'and' At(Truck2, Sydney). An efficient way of representing the states is to only specify which literals are true. The rest of them can then be assumed to be false (by the closed world assumption).

(b) Goal representation [1p]

A goal is like a preposition: a conjunction of literals (positive or negative) that may contain variables, such as At(p, SFO) 'and' Plane(p). The goal is solved when we have found a sequence of actions that end in a state s that entails the goal. There can be many states that fulfills the requirements of being a goal state. For instance both At(p, SFO) 'and' Plane(p) 'and' Raining(w), and At(p, SFO) 'and' Plane(p) 'and' Sunny(w) fulfills the requirements of being a goal state by the definition above.

(c) Action representation [1p]

Any system for action description needs to solve the **frame problem** - to say what changes and what stays the same as the result of the action. Classical planning concentrate on problems where most of the actions leave most things unchanged. E.g the action of nudging an object causes that object to change its location by a change vector. The action only describes how the nudged object's location changes, and doesn't mention how every other object in the world stays in the same place.

The action schema consists of the action name, a list of all the variables used in the schema, a precondition and an effect. The precondition and effect of an action are each conjunctions of literals (positive or negated atomic sentences).The precondition defines the states in which the action can be executed, and the effect defines the result of executing the action.

The following questions pertain to non-monotonic reasoning and STRIPS:

(a) What is the Closed World Assumption? [1p]

The closed world assumption is the presumption that a statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false. The open world on the other hand means that unmentioned literals are unknown.

(b) How is the Closed World Assumption used in the context of STRIPS planning? In particular, how is it applied to state representation? [1p]

The STRIPS language only allows positive literals in the states, Since the closed world assumption is used, all literals not defined in a state are assumed to be false.

Q2:

The following questions pertain to automated planning

(a) A partial order causal link (POCL) planner begins with an initial plan and proceeds by resolving *flaws* in this plan. For example, there can be *threats* that need to be resolved. What is a *threat*? Remember that there is a specific definition of threats for POCL planning, related to the structure of a plan that is under construction but has not yet been finished. How can we resolve a threat? That is, how does a POCL planner modify or extend the plan so that the threat no longer exists?

A threat is when one part of a plan may prevent another of being realized. The two conflicting parts needs to be ordered in a way that makes the conflict disappear.

(b) Recall that the optimal delete relaxation heuristic h+(s) applies delete relaxation to a planning problem and then solves the resulting relaxed problem optimally starting in state s, taking the cost of the resulting plan as the heuristic value. As this requires optimal planning, h+ is too slow for use in planning, but we could construct other heuristics such as h1 that build on h+. What is the main idea behind h1 that allows it to avoid ”combinatorial explosions” and to be computed far more quickly than h+? What is the difference between the two very similar heuristic functions h1 and h_add (in terms of how the functions are defined and computed)?

h1(n) - The maximum of the heuristic values

Is an admissible heuristic. Does not contain as much information as the hadd(n) heuristic. Guarantees an optimal solution.

hadd(n) - The sum of all heuristic values

Is not an admissible heuristic as the sum of all of the heuristic values might be an overestimation of the lowest actual cost of reaching the goal state. This heuristic contains more information than the h1 heuristic but might not result in the algorithm reaching the goal state with the lowest cost.

(c) What is satisficing planning? Does this type of planning typically require admissible heuristics? Why or why not?

Satisficing planning means using plans that will get us to the goal but does not guarantee that we will find the optimal solution. The algorithm might take a suboptimal rout to the goal but as it might be computationally expensive to generate a perfect plan it might be worth it. As the algorithm will be fine with any solution the gets us to the goal, the heuristic function does not need to be admissable.

Q3:

The following questions pertain to partial-order planning.

(a) What is partial-order planning and how does it differ from STRIPS-based planning in terms of search space and output of the respective planning algorithms.

A partial order plan leaves the decision of **ordering of actions as open as possible**. Total order planning (as in **STRIPS) has an exact ordering** of the actions.

In STRIPS the search space are the different states (conjunctions of literals) and the actions are a way of traversing the state space, but in partial order planning the search space consists of the different partial order plans that can be executed.

(b) Describe the four basic components of partial-order planning according to Russell/Norvig.

Causal-links: What actions fulfills other actions preconditions.

Partial-order: An ordering of actions where needed. ”Action A needs to execute before action B”.

Actions: The set of available actions.

Open preconditions: Preconditions of actions that can’t be fulfilled by any available action.

Q4:

Domain-independent heuristics play an important part in many planning algorithms. Such heuristics are often based on relaxing the original planning problem.

(a) Explain the concept of relaxing a problem and how this helps us find useful heuristics.[2p]

Relaxing a problem is when create a new problem from the original problem that is easier to solve, but where all of the solutions to the original problem are still valid. A possible heuristic to the original problem can be the cost of solving the relaxed problem optimally from the current node. Since all of the solutions to the original problem are valid in the relaxed problem the cost of solving the relaxed problem from a node is never larger than in the original problem, so the heuristic is admissable.

(b) Provide a concrete example of this technique using an action schema in a planning domain of your choice. [2p]

Draw a tree of different states as the original problem, and add new arcs in the relaxed problem.

REINFORCEMENT LEARNING

Q1:

(a) As formally as possible, define *supervised learning.*

Supervised learning is when we try to infer a function that maps input values, usually called features, X to a output Y. This is done by training the model on observed cases where we have a mapping between set of features X to a respective outputs Y. The function can after training then be used on features with unobserved output and predict Y using the knowledge that it has learned during its training. The model can then be evaluated by how well it predicted the output Y.

(b) Explain the purpose of each of the terms Q(), R(), alpha, and gamma in the Q-learning update below.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R(s_t) + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Q(s, a): Is a mapping between the expected utility associated with executing an action a in a given state s.
R(s): Is the reward of being in the state s.
Alpha: Learning rate. The higher the value for alpha is in the update equation the more the future values will define the updated Q-value, and the faster the algorithm will learn.
Gamma: Determines how fast the rewards decline depending on how far into the future they are attained.

(c) Explain in more detail the *curse of dimensionality.*

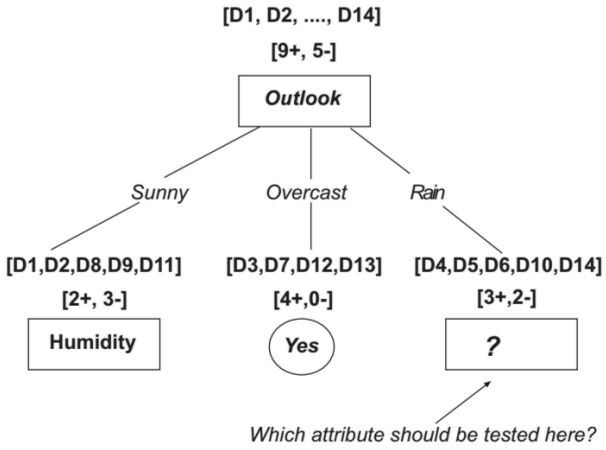
The curse of dimensionality means that as the number of features (inputs) gets larger, and therefore the dimension of the feature space gets larger, the number of training samples needed to make sure that there is several examples of outcomes for a given input needs to grow. With a fixed number of training samples the predictive power of the model will get smaller as the dimension of the feature space gets larger.

DECISION TREES

Q1:

The following question pertains Decision Tree Learning. Use the definitions and data table in the Figure 3, Appendix 2 to answer this question. Figure 2 show a partial decision tree for the Table ….

(a) What attribute should be tested in the box with the question mark on the right branch of the decision tree in the figure below? Justify your question by computing the information gain for the appropriate attributes in the Table in Figure 3.



$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v),$$

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Figure 3: Sample Table for Target Attribute PlayTennis

Entropy(S) = -p+log2(p+) - (p-)log2(p-) = -3/5log2(3/5) - 2/5log2(2/5)

Gain(Wind) = Entropy(S) - (|S_Strong|/|S| * Entropy(S_Strong) + |S_Weak|/|S| * Entropy(S_Weak)
= Entropy(S) - (2/5 * (-0/2log2(0/2) - 2/2log2(2/2)) + 3/5 * (-2/2log2(2/2) - 0/2log2(0/2)))
= Entropy(S) = -0.6log2(0.6) - 0.4log2(0.4)
= **0.97**

Gain(Temp) = Entropy(S) - (|S_Mild|/|S| * Entropy(S_Mild) + |S_Cool|/|S| * Entropy(S_Cool)
= Entropy(S) - (3/5 * (-1/3log2(1/3) - 2/3log2(2/3)) + 2/5 * (-1/2log2(1/2) - 1/2log2(1/2)))
= Entropy(S) - (3/5 * (-1/3log2(1/3) - 2/3log2(2/3)) - 2/5

= -3/5log2(3/5) - 2/5log2(2/5) - (3/5 * (-1/3log2(1/3) - 2/3log2(2/3)) - 2/5
= 0.82

Gain(Humidity) = Entropy(S) - (|S_High|/|S| * Entropy(S_High) + |S_Normal|/|S| * Entropy(S_Normal)
= Entropy(S) - (2/5 * (-1/2log2(1/2) - 1/2log2(1/2)) + 3/5 * (-2/3log2(2/3) - 1/3log2(1/3)))
= Entropy(S) - (2/5 + 3/5 * (-2/3log2(2/3) - 1/3log2(1/3)))
= -3/5log2(3/5) - 2/5log2(2/5) - (2/5 + 3/5 * (-2/3log2(2/3) - 1/3log2(1/3)))
= 0.02

Answer: The attribute that should be tested is **Wind** given that it's raining.