



EXAMINATIONS — 2003

END-YEAR

COMP 307

ARTIFICIAL INTELLIGENCE

Time Allowed: 3 Hours

Instructions: There are a total of 180 marks on this exam.
Attempt all questions.
Calculators may be used.
Non-electronic foreign language translation dictionaries may be used.

Questions

- | | |
|---------------------------------|------|
| 1. Prolog | [30] |
| 2. Search | [30] |
| 3. Rule Based Systems | [30] |
| 4. Planning | [35] |
| 5. Natural Language Processing. | [25] |
| 6. Machine Learning | [30] |

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. Prolog

[30 marks]

(a) [12 marks] Give the most general unifier for the following pairs of terms, or say that the terms do not unify. The first answer is given as an example.

property(One, Two)
property(3, 4)

Unifies with One=3, Two=4

date(D, may, Y)
date(17, M, Z)

Unifies with D=17, M=may, Y=Z

collection([1,2], X)
collection([H|T], Y)

Unifies with H=1, T=[2]

X
2+3

Unifies with X=2+3

family(leon, liz, [leon,danya])
family(X, liz, [X])

Does not unify

foo(X, [1, X, 2])
foo(a, Z)

Unifies with X=a, Z=[1,a,2]

lect(X, room(hu1), [comp307, X])
lect(Y, room(Y), Z)

Unifies with X=Y=hu1, Z=[comp307, hu1]

(Question 1 continued on next page)

(Question 1 continued)

(b) [8 marks] Matching, Backtracking and Cut

Consider the following program.

```
find(X):- p(X), q(X), r(X).
```

```
q(X):- a(X).
```

```
q(X):- b(X), !, c(X).
```

```
q(X):- d(X).
```

```
p(liz).
```

```
p(leon).
```

```
p(andrew).
```

```
a(liz).
```

```
b(liz).
```

```
b(leon).
```

```
c(liz).
```

```
d(liz).
```

```
d(leon).
```

```
d(andrew).
```

```
r(liz).
```

```
r(leon).
```

```
r(andrew).
```

What is the output generated by the following query

```
|-find(X).
```

Give all the possible answers. If an answer is found more than once, show all the occurrences.

```
X = liz ? ;  
X = liz ? ;  
X = andrew ? ;
```

(Question 1 continued on next page)

(Question 1 continued)

(c) [10 marks] Lists and Recursion

Write a program for `sublist(+List, +N, ?List1, ?List2)` where `List1` is the first `N` elements of `List` in the same order, and `List2` is the rest of `List` in the same order. For example,

```
sublist([2, 4, 5], 0, X, Y) is true with X=[], Y=[2, 4, 5]
sublist([2, 4, 5], 2, X, Y) is true with X=[2, 4], Y=[5]
sublist([2, 4, 5], 3, X, Y) is true with X=[2, 4, 5], Y=[]
```

Make sure that your program can handle the special cases of `N` being negative or larger than the length of `List`:

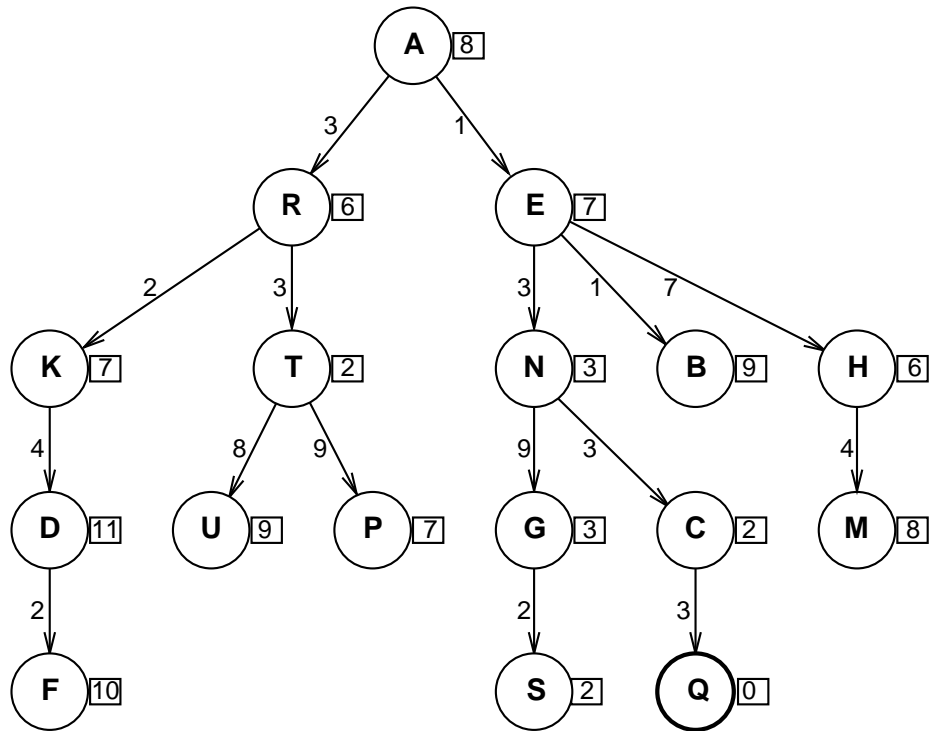
```
sublist([2, 4, 5], -5, X, Y) is false and should say no.
sublist([2, 4, 5], 100, X, Y) is true with X=[2, 4, 5], Y=[]
```

```
sublist([],N,[],[]):- N>=0.
sublist(L, 0, [], L).
sublist([H|T], N, [H|X1], L):-
    N>0,
    N1 is N-1,
    sublist(T, N1, X1, L).
```

Question 2. Search

[30 marks]

Consider the search tree shown below.



The initial state is represented by node A. The goal state is represented by node Q. The number on a link is the cost of the operator that links the two nodes. Each node has an associated number.

In the questions below, assume that the searches consider child nodes from left to right. If you make any additional assumptions, state them clearly.

For the uninformed search strategies in (a) to (d), specify the order in which the nodes will be expanded.

(a) [3 marks] **Breadth first search**

A R E K T N B H D U P G C M F S Q

(b) [3 marks] **Depth first search**

A R K D F T U P E N G S C Q (B H M)

(c) [4 marks] **Depth limited search with a limit=2**

A R K T E N B H

(Question 2 continued on next page)

(Question 2 continued)

(d) [4 marks] Iterative deepening search

A A R E A R K T E N B H A R K D T U P E N G C B H M
A R K D F T U P E N G S C Q (B H M)

For the informed searches in (e) to (g), specify the order in which the nodes will be expanded. Assume that the number on a node represents a (lower bound) heuristic estimate of the cost of reaching a goal from the node.

(e) [4 marks] Uniform cost search (best first search using the path cost only).

A E B R N K T C H D Q

(f) [4 marks] “Greedy” best first search (best first search using the heuristic estimate only)

A R T E N C Q or
A R T (P and/or K) E N C Q

(g) [4 marks] A* search

A E N C R T Q or
A E N R T C Q

For the hill climbing search below, specify the order in which the nodes will be expanded. In this case, assume that the number on a node is the value of the node, and we want to find the node with the lowest value.

(h) [4 marks] Hill climbing

A R T

Question 3. Rule Based Systems

[30 marks]

Consider the following set of rules. Goal1 and Goal2 are goals to be established. H1, H2, H3, H4 are intermediate inferences, and Q1, Q2, . . . , Q6 are the questions that can be asked of the user.

R1: if H1 and H2 then Goal1

R2: if H2 and H3 then Goal2

R3: if H4 then Goal2

R4: if Q1 then H1

R5: if Q2 then H1

R6: if Q3 then H2

R7: if Q4 then H2

R8: if Q5 then H3

R9: if Q6 then H4

Assume the rules are scanned top down, that is, a rule with a smaller number has higher priority.

(a) [8 marks] **Backward chaining**

This question uses backward chaining. If a particular question is asked, the user would respond as follows:

Q1: YES

Q2: No

Q3: No

Q4: No

Q5: YES

Q6: YES

Give the sequence of rules that a backward chaining interpreter would apply to determine whether any goal is true. (List the rule numbers only).

Assume there is **no** caching for intermediate results, and assume backtracking is always allowed (cuts are not used).

R1, R4, R6, R7, R2, R6, R7, R3, R9

(b) [8 marks] **Forward chaining**

This question uses forward chaining. Suppose Q1, Q5, Q6 are true and saved in the working memory.

Give the sequence of rules that a forward chaining interpreter would apply to determine whether any goal is true. (List the rule numbers only).

1, 2, 3, 4, 1, 2,3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3

or 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3

note that this is listing every rule that is checked

If you just listed the rules that were satisfied, it would be

R4, R8, R9, R3

(c) [6 marks] In general, would you expect forward and backward chaining to give the same conclusions on the same rule set? Why or why not?

May not be the same if a problem could have multiple conclusions. If you stop when you find the first conclusion, then forward chaining and backward chaining may reach different conclusions first because the backward chaining is governed by the ordering of both the hypotheses and the rules and follows the logical dependencies of the rules, whereas the forward

Question 4. Planning

[35 marks]

(a) [4 marks] What are the four parts of a STRIPS operator?

name and parameters
Preconditions
Adds: facts added by the operation
Deletes: facts deleted by the operation

(b) [2 marks] What constraint on the variables in the four parts must be satisfied to make it a valid STRIPS operator?

All variables in the preconditions, adds, or deletes must be specified in the parameters of the operator.

(c) [4 marks] Write a STRIPS operator for the action `unlock`. To unlock a door, the robot needs to have a key, the door needs to be locked (and shut), and the key should be the right one for the door.

```
unlock(Door, Key)
pre: hold(Key), locked(Door), shut(Door), match(Key, Door)
del: locked(Door)
add: unlocked(Door)
```

(Question 4 continued on next page)

(Question 4 continued)

(d) [4 marks] The unlock operator you wrote above almost certainly does not represent all the possible preconditions for successfully unlocking a door. Describe at least four additional preconditions for unlocking a door that a person could reason about.

(Hint: consider situations where the robot could not unlock the door successfully.)

lock not rusted,
key not bent
door has only one lock
key is not stuck to robot's hand
key is not locked inside a box that the robot is holding
the robot is on the right side of the door
the door/lock is not boarded over
the lock is not filled with chewing gum or concrete
there is not a man with a rocket launcher guarding the door

(e) [4 marks] There are also many possible effects of unlocking a door that could occur in some circumstances. For example, the door may blow up if there is a bomb wired to the lock, or a person may walk into the house if they were waiting outside the door. Show how you could represent these two conditional effects using a set of STRIPS operators.

```
unlock1(door, key)
pre: hold(key), locked(door), shut(door), match(key, door), hasbomb(door))
del: locked(door), closed(door), in(door, house)
add: mess(door)
unlock2(door, key, house)
pre: hold(key), locked(door), shut(door), match(key, door),
externaldoor(door, house), at(door, person), outside(person,house)
del: locked(door), outside(person, house)
add: unlocked(door), inside(person, house)
unlock(door, key)
pre: hold(key), locked(door), shut(door), match(key, door))
del: locked(door)
add: unlocked(door)
```

(Question 4 continued on next page)

(Question 4 continued)

Consider the following planning problem. A robot has to put together a kitset table. The goal is for the legs to be coloured black, the table top to be smooth and coloured red, and the top to be joined to the legs. The initial state has the legs and top separate, and both coloured black, and the top is not smooth. The robot has three operators, to join parts together, to paint an object a new colour (black or red), and to sand an object to remove any paint and make it smooth.

Initial State: `separate(legs)`, `separate(top)`, `colour(legs, black)`, `colour(top, black)`

Goal: `joined(top, legs)`, `smooth(top)`, `colour(legs, black)`, `colour(top, red)`

Operators:

`assemble(Obj1, Obj2):`

`pre: separate(Obj1), separate(Obj2)`

`del: separate(Obj1), separate(Obj2)`

`add: joined(Obj1, Obj2)`

`paint(Object, OldColour, NewColour):`

`pre: separate(Object), colour(Object, OldColour)`

`del: colour(Object, OldColour)`

`add: colour(Object, NewColour)`

`sand(Object, OldColour):`

`pre: separate(Object), colour(Object, OldColour)`

`del: colour(Object, OldColour)`

`add: smooth(Object), colour(Object, wood)`

(f) [6 marks] Suppose the robot used a goal-stack planner to construct a plan for this task. On the facing page, show the steps that the goal-stack planner would take, up to the point that it has added the first action to the current plan. Show the items added to and removed from the Goal Stack, the changes to the current state, and the action added to the current plan.

Assume that the subgoals of a conjunction are added to the stack in reverse order, so that the first subgoal is at the top of the stack.

colour(top, black)
colour(legs, black)
separate(top)
separate(legs)

Current State

joined(top,legs) & smooth(top) & colour(top, red) & colour(legs,black)

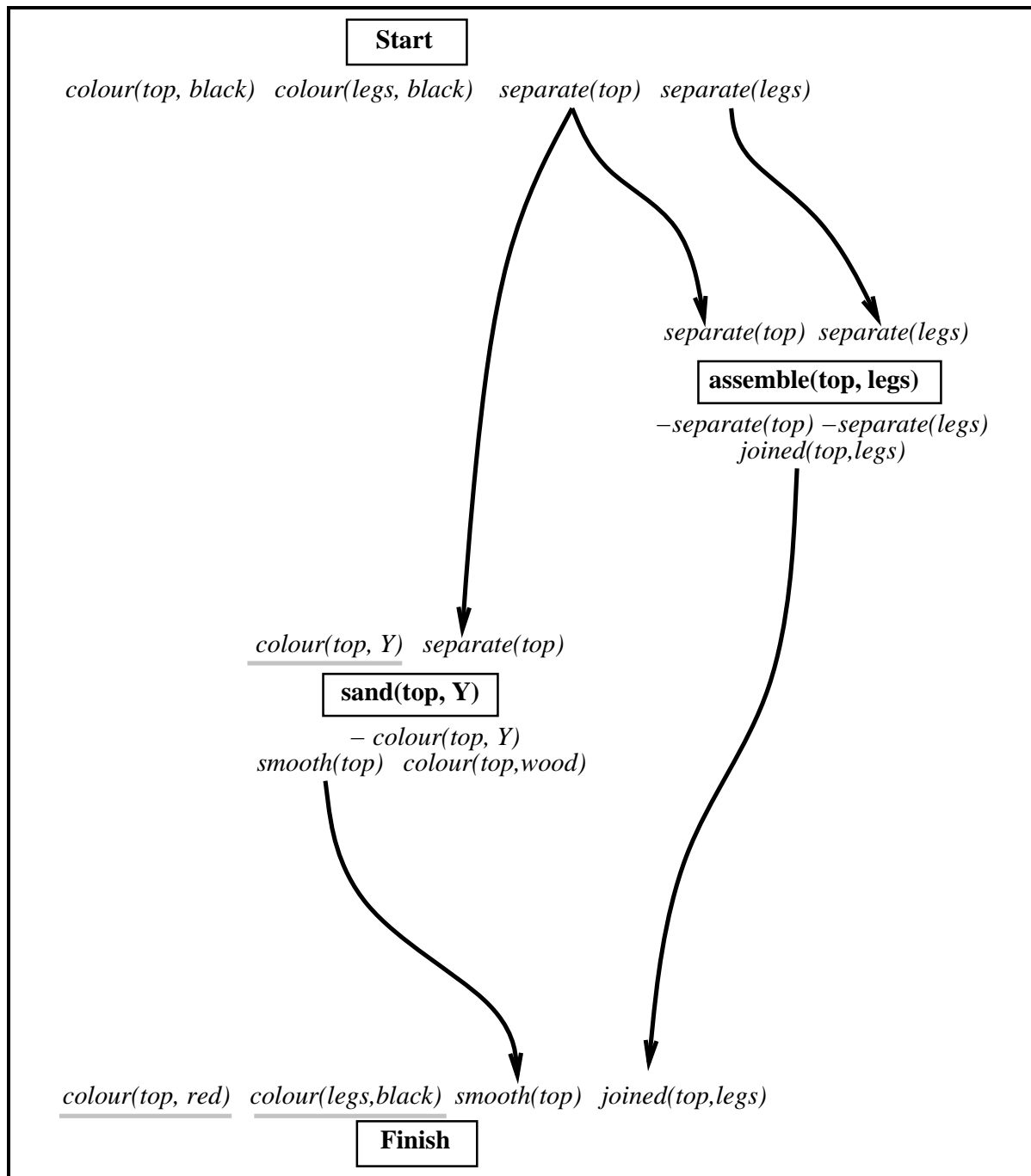
Goal Stack

Current Plan:

(Question 4 continued)

Suppose the robot used a Partial Order Planner (POP) to construct a plan for the same assembly task. The diagram below shows the state of the plan after the planner has added two actions to satisfy some of the subgoals, and has satisfied some of their preconditions.

- (h) [3 marks] Show the threat that exists in the plan, and resolve it.
- (i) [2 marks] Satisfy the other precondition of the `sand(top,Y)` action using an existing action.
- (j) [3 marks] Add a paint action to satisfy the subgoal `colour(top, red)`, and show any threats that are introduced.



SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 5. Natural Language Processing

[25 marks]

Consider the following grammar and lexicon for simple yes/no questions such as
was the toaster on the bench?
did the robot wash a bowl?
has the toaster eaten the bread and the milk?

Simple yes/no questions like these start with “auxilliary” verbs — a special, restricted set of verbs including “is”, “was”, “were”, “does”, “did”, “has”, “have”, and “had”.

s --> aux, np, pp.	lexicon(robot, noun).
s --> aux, np, vp.	lexicon(toaster, noun).
	lexicon(bench, noun).
pp --> prep, np.	lexicon(bread, noun).
	lexicon(milk, noun).
vp --> verb, np, pp.	lexicon(bowl, noun).
vp --> verb, np.	lexicon(kitchen, noun).
vp --> verb.	
	lexicon(is, aux, be).
np --> det, noun, pp.	lexicon(was, aux, be).
np --> det, noun.	lexicon(were, aux, be).
	lexicon(does, aux, do).
prep --> [P], {lexicon(P, prep)}.	lexicon(did, aux, do).
det --> [D], {lexicon(D, det)}.	lexicon(has, aux, have).
noun --> [N], {lexicon(N, noun)}.	lexicon(have, aux, have).
aux --> [V], {lexicon(V, aux, _)}.	lexicon(had, aux, have).
verb --> [V], {lexicon(V, verb)}.	
	lexicon(on, prep).
	lexicon(in, prep).
lexicon(wash, verb).	
lexicon(washing, verb).	lexicon(the, det).
lexicon(washed, verb).	lexicon(a, det).
lexicon(eat, verb).	
lexicon(eating, verb).	lexicon(and, conj).
lexicon(eaten, verb).	lexicon(or, conj).

(a) [4 marks] Show the parse tree for the question:
is the bread in the toaster?

```
s(aux(is),
  np(det(the),
    noun(bread)),
  pp(prepare(in),
    np(det(the),
      noun(toaster))))
```

(b) [5 marks] Show three possible parse trees for the question:
did the robot wash the bowl on the bench in the kitchen?

s(aux(did),
 np(det(the),
 noun(robot)),
vp(verb(wash),
 np(det(the),
 noun(bowl),
 pp(prep(on),
 np(det(the),
 noun(bench))))))
 pp(prep(in),
 np(det(the),
 noun(kitchen))))))
s(aux(did),
 np(det(the),
 noun(robot)),
vp(verb(wash), -
 np(det(the),
 noun(bowl),
 pp(prep(on),
 np(det(the),
 noun(bench)
 pp(prep(in),
 np(det(the),
 noun(kitchen))))))))))
s(aux(did),
 np(det(the),
 noun(robot)),
 -
 vp(verb(wash),
 np(det(the),
 noun(bowl)),
 pp(prep(on),
 np(det(the),
 noun(bench)
 pp(prep(in),
 np(det(the),
 noun(kitchen))))))))))

(Question 5 continued)

(c) [3 marks] Explain the different meanings of the parse trees from part (b).
Hint: which bowl is being referred to and where did the washing happen?

First: the bowl is the one on the bench; the washing happened in the kitchen.

Second: the bowl is the one on the bench and the bench is the one in the kitchen. Where the washing happened isn't specified.

Third: the washing happened on the bench that is in the kitchen. The bowl isn't specified.

(d) [5 marks] Noun phrases can contain two sub phrases, joined with a conjunction ("and" or "or").
For example:

Did the robot and the toaster eat the bread?

Is the bread in a bowl on the bench or the toaster?

Modify the grammar to allow a noun phrase to contain two sub phrases connected with conjunctions, as in the sentences above. (Note that the lexicon already contains entries for "and" and "or".)

```
np --> np1, conj, np1.  
np --> np1.  
np1 --> det, noun, pp.  
np1 --> det, noun.  
conj --> [C], {lexicon(C, conj)}.
```

Note, this only allows one conjunction; The following is also OK

```
np --> np1, conj, np1.
```

(Question 5 continued on next page)

(Question 5 continued)

(e) [8 marks] The original grammar is too general – it allows invalid sentences, like the three on the left below, although the three on the right are valid.

* is the robot washed the bowl	is the robot washing the bowl
* did the toaster eating the bread	did the toaster eat the bread
* did the toaster on the bench	is the toaster on the bench

One problem is that the auxilliary verb at the beginning of the sentence must match the form of the verb in the vp (if there is a vp).

- The “be” type auxilliaries (“is”, “was”, “were”) must have a verb that ends with “ing”.
- The “have” type auxilliaries (“has”, “have”, “had”) must have a verb that ends with “en” or “ed”.
- The “do” type auxilliaries (“do”, “did”, “does”) must have a verb with no ending.

The other problem is that a sentence with no vp (the first grammar rule), can only start with a “be” type auxilliary.

Extend the grammar and the lexicon to enforce these constraints. You do not need to write out rules that are the same as the original grammar and lexicon.

```
s --> aux(be), np, pp.  
s --> aux(Root), np, vp(Cat), {matches(Root,Cat)}.  
vp(Cat) --> verb(Cat), np, pp.  
vp(Cat) --> verb(Cat), np.  
aux(Root) --> [A], {lexicon(A, aux, Root)}.  
verb(Cat) --> [V], {lexicon(V, verb, Cat)}.  
lexicon(wash, verb, inf).  
lexicon(washing, verb, ing).  
lexicon(washed, verb, ed).  
lexicon(eat, verb, inf).  
lexicon(eating, verb, ing).  
lexicon(eaten, verb, ed).  
  
matches(be, ing).  
matches(do, inf).  
matches(have, ed).
```

Question 6. Machine Learning

[30 marks]

(a) [6 marks] Suppose that the Naïve Bayes algorithm is used to classify an instance described by a vector of n attributes (A_1, A_2, \dots, A_n) into one of two categories, C_1 and C_2 . Given an unclassified instance described by the vector (x_1, x_2, \dots, x_n) , the Naïve Bayes algorithm computes a score for each category and returns the category with the highest score. The Naïve Bayes learning algorithm analyses its training data to obtain a set of conditional probabilities for computing the scores.

Explain what probabilities the learning algorithm obtains from the training data, and how it computes the two scores for a new instance.

$$p(C_1) \text{ and } p(C_2) (= 1 - p(C_1))$$

and for each i in $1 \dots n$:

$$p(A_i = \text{true} | C_1),$$
$$p(A_i = \text{false} | C_1)$$
$$p(A_i = \text{true} | C_2),$$
$$p(A_i = \text{false} | C_2)$$
$$\text{score}(C_i) = p(C_i) \prod_{j=1}^n [p(A_j = x_j | C_i)]$$

(b) [2 marks] What is the assumption that the Naïve Bayes classifier makes about the data that is almost always wrong?

It assumes that all the attributes are conditionally independent given the category. It is wrong because the attributes are often correlated.

(c) [4 marks] Decision trees can also be used to classify instances into one of two categories. The central action of the decision tree learning algorithm is to work out what attribute to use for the decision at a node of the tree. Given a node that has some instances from each category, explain how the decision tree chooses an attribute to use for the node. State the formula for the score of an attribute. Call the attribute “A” and assume that it has just two values v_1 and v_2 . Call the classes C_1 and C_2 .

$$p(A = v_1) * p(C_1 | A = v_1) * p(C_2 | A = v_1) +$$
$$p(A = v_2) * p(C_1 | A = v_2) * p(C_2 | A = v_2)$$

(Question 6 continued on next page)

(Question 6 continued)

(d) [6 marks] Consider the following data set describing 12 species of trees of which 6 are good for timber, and 6 are not. They are described by three binary attributes: the kind of leaf, the hardness of the wood, and the pattern of the branches.

	Leaf	Wood	Branching	Class
G1	needle	soft	irregular	Good
G2	needle	soft	regular	Good
G3	needle	soft	regular	Good
G4	needle	hard	regular	Good
G5	needle	hard	regular	Good
G6	flat	hard	regular	Good
B7	flat	soft	irregular	Bad
B8	flat	hard	regular	Bad
B9	flat	soft	regular	Bad
B10	flat	soft	regular	Bad
B11	needle	hard	regular	Bad
B12	needle	hard	irregular	Bad

Which attribute would the decision tree building algorithm choose for the root of a decision tree for the timber database? Show your working and state the impurity measure you are using.

$$\text{Leaf: } 7/12 * (2/7 * 5/7) + 5/12 * (1/5 * 4/5) = (10/7 + 4/5) / 12 = 2.23/12$$

$$\text{Wood: } 6/12 * (3/6 * 3/6) + 6/12 * (3/6 * 3/6) = (9/6 + 9/6) / 12 = 3/12$$

$$\text{Branch: } 9/12 * (4/9 * 5/9) + 3/12 * (1/3 * 2/3) = (20/9 + 2/3) / 12 = 26/9/12 = 2.9/12$$

Leaf has the lowest score, therefore the algorithm will use Leaf at the root

(Question 6 continued on next page)

(Question 6 continued)

(e) [4 marks] Perceptrons are a classification mechanism that use a set of weights and a threshold. Given a set of examples (positive and negative), the perceptron learning algorithm is able to learn a threshold and set of weights. Explain the major limitation of the perceptron learning algorithm.

The instances must be linearly separable - if they are not, then the algorithm will not converge

(f) [8 marks] Draw a diagram of a multilayer feedforward neural network, and identify the input, hidden, and output nodes. Explain why the network does not suffer from the same limitation as the perceptron, particularly noting the role of the hidden nodes.

