UNIVERSITY OF CAMBRIDGE

COMPUTER SCIENCE TRIPOS Part IB

Tuesday 2 June 2015 1.30 to 4.30 pm

COMPUTER SCIENCE Paper 4

*Answer **five** questions.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags
Rough work pad

SPECIAL REQUIREMENTS
Approved calculator permitted

1 Artificial Intelligence I

We aim to solve a *supervised learning* problem using a simple neural network taking input vectors $\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$ of features and computing the function

$$h(\mathbf{x}, \mathbf{w}) = \sigma\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

where $\mathbf{w}^T = (w_0, w_1, \ldots, w_n)$ is a vector of weights and σ is an activation function. We have a set $\mathbf{s}^T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ of m labelled training examples and seek to minimize the error

$$E(\mathbf{w}) = \sum_{i=1}^{m} (y_i - h(\mathbf{x}_i, \mathbf{w}))^2.$$

(*a*) Derive the gradient descent training algorithm for this problem. [5 marks]

(*b*) We now notice that for the particular problem of interest a solution will only make sense if a specific subset S of the weights is positive. Devise a modified version of the training algorithm that enforces this. [8 marks]

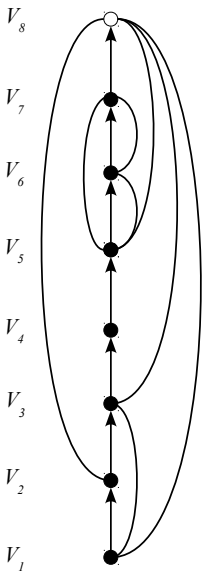
(*c*) A colleague is attempting to solve a heuristic search problem using the $A^\star$ algorithm, but is unable to decide which of a number of heuristics to use. Your colleague has a large collection of test problems, and believes that the best heuristic to use might depend on particular characteristics of the problem being solved. Explain in detail how you might apply machine learning to help your colleague. [7 marks]

2 Artificial Intelligence I

(*a*) Explain, giving a specific example, why *chronological backtracking* might be sub-optimal in solving *Constraint Satisfaction Problems (CSPs).* [4 marks]

(*b*) Explain what is meant by *backjumping.* Give **two** ways in which the approach to backjumping taken by *Gaschnig's algorithm* differs from that taken by *graph-based backjumping (GBB).* [4 marks]

Consider the following incomplete attempt to solve a CSP.



In this diagram, filled nodes are assigned variables, unfilled nodes are currently unassigned, arrows denote the order in which variables are assigned, and arcs denote constraints between variables.

(*c*) Give a detailed explanation of how in general GBB decides where to make its first backjump. Illustrate your answer using the example above, assuming that it is not possible to make a consistent assignment at V_8. [4 marks]

(*d*) In the example illustrated, assume the first backjump made is from V_8 as in part (*c*). We find after the first backjump that the variable we have arrived at has no further consistent assignments. Give a detailed explanation of how GBB decides what to do next, including in your answer a description of the set of *induced ancestors* that are relevant in making the decision. [8 marks]

3 Computer Graphics and Image Processing

Consider the transformations used in the construction and rendering of a three-dimensional model on a screen.

(*a*) List the three principal transformations in the processing pipeline and explain their rôles. [6 marks]

(*b*) Why is it convenient to represent the transformations as matrices? [2 marks]

(*c*) What are homogeneous coordinates? Explain how they are used in modelling these transformations as matrices. [2 marks]

(*d*) Derive the matrix to represent a perspective transformation for a viewer at the origin of a point in three dimensions to a point on a screen in the plane $z = d$. [5 marks]

(*e*) Perspective in classical art has *vanishing points* towards which parallel lines converge. Explain mathematically why this is the case and show how to calculate the location on the screen of the vanishing point for lines in a particular direction. [5 marks]

[*Hint:* It may be helpful to represent lines parametrically in vector form as $\mathbf{P}(s) = \mathbf{A} + s\mathbf{V}$ where $\mathbf{V}$ is a direction and $\mathbf{A}$ is any point on the line.]

4 Computer Graphics and Image Processing

Consider a curve defined by polynomial parametric segments $\mathbf{P}_i(s)$ for $i = 1, 2, \ldots m$ that interpolates a set of points $\{\mathbf{A}_i\}_{0 \le i \le m}$ in three dimensions.

(*a*) What is meant by C_k continuity at the junction of two segments? [3 marks]

(*b*) What is the least order of the polynomials that must be used to achieve C_k continuity at the junctions? [2 marks]

(*c*) Derive the Overhauser formulation for a set of weighting functions $w_{-2}(s)$, $w_{-1}(s)$, $w_0(s)$ and $w_1(s)$ so that the cubic curve segment joining $\mathbf{A}_{i-1}$ and $\mathbf{A}_i$ can be expressed as $\mathbf{P}_i(s) = w_{-2}(s)\mathbf{A}_{i-2} + w_{-1}(s)\mathbf{A}_{i-1} + w_0(s)\mathbf{A}_i + w_1(s)\mathbf{A}_{i+1}$ for $1 < i < m$. [10 marks]

(*d*) Extend this formulation to give a set of parametric patches $\mathbf{P}_{i,j}(s,t)$ for $1 < i < m$ and $1 < j < n$ interpolating a surface through an array of points $\{\mathbf{A}_{i,j}\}_{0 \le i \le m, 0 \le j \le n}$. [5 marks]

5 Databases

(a) The relational schema $R(A,\ B,\ C,\ D,\ E)$ has the following functional dependencies.

$$\begin{array}{lcl} A & \rightarrow & E \\ B & \rightarrow & D \\ A, B & \rightarrow & C \end{array}$$

Decompose this into a set of relations in BCNF. Show your working. [5 marks]

(b) By inspecting your answer to (a), describe a possible interpretation in the language of Entity-Relationship modelling. [5 marks]

(c) Heath's Rule tells us that if $R(A,\ B,\ C)$ is a relational schema with functional dependency $A \rightarrow B$, then

$$R = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R).$$

This rule is often applied in the relational decomposition process that seeks to arrive at relations in a particular normal form. For example, we might decompose R into two implemented relations $R_1(A,\ B)$ and $R_2(A,\ C)$. Some people have been very critical of this approach since it ignores the fact that the implementation of such a decomposition is normally associated with *foreign key constraints* between tables.

What is missing? Can you express, in the relational algebra, what such a missing constraint might look like for the decomposition described above using Heath's rule? Justify your answer. [5 marks]

(d) Using your answer to (c), discuss which constraints might be missing from your decomposition in question (a). [5 marks]

6 Databases

We assume that for each base table R in a relational database we have two update operations : insert($R,\ t$) which inserts tuple t into table R if t does not violate any of the constraints declared for R (fails otherwise), and delete($R,\ p$) which deletes all records in R satisfying predicate p (and fails if this would violate referential integrity constraints). Update operations are combined in programs to define transactions with ACID guarantees.

Suppose that we have defined a view $V = Q(R_1, R_2, \ldots, R_n)$, where the R_i indicate the base tables used in query Q. The designers of a new database system want to allow users to update directly such a view. That is, if we have an update of the form $U = \text{insert}(V,\ t)$ or $U = \text{delete}(V,\ p)$, then the database system must automatically generate a transaction T_U over the tables R_i such that for any database instance DB this diagram commutes:

$$\begin{array}{ccc} DB & \xrightarrow{T_U} & DB' \\ \downarrow Q & & \downarrow Q \\ V & \xrightarrow{U} & V' \end{array}$$

In other words, applying the update U directly to a view (as if it were a base table) produces the same result as applying T_U to the database and then evaluating the view query.

A major problem with this approach is that there may be multiple distinct solutions for T_U. We explore this now.

(a) Explain the difference between a *database query* and a *database view*. [2 marks]

(b) Let $V = \pi_X(R)$ be a view for some base table R and some subset X of R's attributes Y. How could this be translated into the desired transaction T_U? Discuss any problems with ambiguity that may arise. [5 marks]

(c) Let $V = \sigma_q(R)$ be a view for some base table R and predicate q. How could this be translated into the desired transaction T_U? Discuss any problems with ambiguity that may arise. [5 marks]

(d) In the design of a database schema it was discovered that a relation R violated Boyce-Codd normal form, and so it was replaced by two base tables R_1 and R_2 resulting from the standard decomposition process. Suppose users attempt to reconstruct the original relation using the view $V = R_1 \bowtie R_2$. Discuss the problems that might arise now in the construction of transaction T_U for updates to V. [8 marks]

7 Economics, Law and Ethics

(*a*) Describe three different philosophies of ethics. [12 marks]

(*b*) Which of these philosophies might be more, or less, attractive to a successful businessman who made his money from coal or steel? Justify your answer. [4 marks]

(*c*) Might a tech billionaire whose fortune is founded on network effects take a different view? Justify your answer. [4 marks]

8 Security I

(*a*) Compare and contrast the security definitions of a *pseudo-random generator* and a *pseudo-random function.* [4 marks]

(*b*) When a Windows NTFS access control entry (ACE) is inherited by a subdirectory, under which circumstances is the "inherit only" flag set or cleared, and why? [4 marks]

(*c*) What is *existential unforgeability* of a message authentication code? [4 marks]

(*d*) Which problem with CBC-MAC is fixed by ECBC-MAC, and how? [4 marks]

(*e*) A C program running on a 32-bit processor contains the following function:

```
void f(int *a, int l) {
  int *b, i;

  b = (int *) malloc(l * sizeof(int));
  if (b == NULL) return;

  for (i = 0; i < l; i++)
     b[i] = a[i];

  [...]
}
```

(*i*) How can a caller cause this function to overwrite unallocated memory? [2 marks]

(*ii*) Modify the function to remove this vulnerability. [2 marks]

9 Security I

A smartcard application requires a stack (LIFO) for storing data records. This stack can become much larger than the tiny amount of memory available on the card. Fortunately, the card terminal has enough memory, and its API offers a class `ExternalStack` that can be invoked remotely from the card. It offers the usual four methods: a constructor to initialise an empty stack, `push(`R`)` to place a data record R onto the stack, `isempty()` to test whether the stack contains no record, and `pop()` to remove and return the top record from the stack.

The integrity of the stack is crucial for the security of the application. However, the card terminal is not tamper resistant and the adversary may have full control over the `ExternalStack` object. Therefore, you have to implement a `SecureStack` wrapper class that uses `ExternalStack` as an untrusted storage provider while guaranteeing the integrity of any data returned. The trusted on-card memory available to a `SecureStack` object is only 256 bytes.

Consider how to implement on the card a class `SecureStack` that provides the same four methods by appending additional data to records before pushing them onto `ExternalStack`, and verifying any data returned against locally held values. You have a message authentication function `mac(`K`,(`R`,...))` and a cryptographic key/nonce generator function `gen()` available.

(a) Why is just appending a message authentication code to each externally stored record not sufficient? [2 marks]

(b) Write short pseudo-code for the four methods of `SecureStack` that shows how they call `ExternalStack`, how they update the on-card check data, and under which conditions a data-integrity alarm is raised. [10 marks]

(c) Express the internal check value(s) of your implementation after these calls:

```
Record r, r1, r2, r3;
SecureStack s;

s.push(r1);
s.push(r2);
r = s.pop();
s.push(r3);
```

[4 marks]

(d) Determine an upper limit for how often the `push` method of your implementation can be called securely. [4 marks]

END OF PAPER