Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Integrated Computer Science Programme — Trinity Term 2016
B.A. (Mod.) Business & Computing
B.A. (Mod.) Computer Science & Language
Year 3 Annual Examinations

Artificial Intelligence I

18 May 2016 — Sports Centre — 14:00 – 16:00

Dr Tim Fernando

Instructions to Candidates:

Attempt *two* questions. All questions carry equal marks. Each question is scored out of a total of 50 marks.

You may not start this examination until you are instructed to do so by the Invigilator.

Materials permitted for this examination:

Non-programmable calculators are permitted for this examination — please indicate the make and model of your calculator on each answer book used.

1. AI is all about the problem of

   (†) an agent acting intelligently in its environment.

   (a) What is the *Symbol System hypothesis* and what does it have to do with (†)?

   [5 marks]

   (b) What is *non-determinism* and what does it have to do with (†)? What does *search* have to do with non-determinism?

   [8 marks]

   (c) State precisely the sense in which Cantor's theorem says that the set of infinite bit strings cannot be searched.

   [5 marks]

   (d) Turning from infinite bit strings to finite bit strings, what is the problem SAT of Boolean satisfiability, and what does it have to do with finite bit strings? How is exponentiation relevant to the question of SAT and polynomial-time computability?

   [12 marks]

   (e) What is a *binary decision diagram* (BDD) and what does it mean to be *ordered* and *reduced*? Give an ordered and reduced BDD equivalent to the Boolean expression

   $$(x_1 \text{ or } x_2) \text{ and } (\overline{x_1} \text{ or } \overline{x_2})$$

   under the variable ordering $x_1 < x_2$. When is an ordered and reduced BDD satisfiable?

   [20 marks]

2. A simple way in Prolog to search a graph for a goal is as follows.

```
search(Node) :- goal(Node).
search(Node) :- arc(Node,Next), search(Next).
```

(a) Non-determinism in arc(Node,Next) above is searched depth-first. For breadth-first search, let searchBF(Node) call the predicate searchL([Node]) on lists below, which is identical to search except the predicates goal(Node) and arc(Node,Next) are revised to predicates goalL(List) and arcL(List,NextList) on lists.

```
searchBF(Node) :- searchL([Node]).
searchL(List) :- goalL(List).
searchL(List) :- arcL(List,NextList), searchL(NextList).
```

Define goalL(List) and arcL(List,NextList), assuming the predicates goal(Node) and arc(Node,Next) are already defined.

[5 marks]

(b) We can revise searchL(List) above to a frontier search seek(List) below treating List as a frontier as follows, where the predicate add2frontier(Children,More,New) builds a new frontier New from Children and More in some way.

```
seek([Node|_]):- goal(Node).
seek([Node|More]):- findall(Next,arc(Node,Next),
                                Children),
                    add2frontier(Children,More,New),
                    seek(New).
```

What more do we need to turn seek(List) into *best-first search*? Outline the modifications to seek(List) that are required to implement best-first search.

[15 marks]

(c) How does *A-star* differ from best-first? What does it mean for A-star to be *admissible*?

[10 marks]

(d) Recall that a *Constraint Satisfaction Problem* (CSP) is given by

- a finite list [X1,...,Xn] of variables Xi

- a finite list C of constraints with variables from [X1,...,Xn]
- a list [D1,...,Dn] of sets Di

and asks if there is an instantiation of variables Xi with values in Di satisfying every constraint in C.

(i) The brute-force generate-and-test approach to CSP instantiates every variable Xi before testing constraints

```
genTest(D1,...,Dn) :- node(X1,...,Xn,D1,...,Dn),
                      testC(X1,...,Xn).
node(X1,...,Xn,D1,...,Dn):- member(X1,D1),...,
                            member(Xn,Dn).
```

Assuming each set Di has size s_i, how many nodes [X1,...,Xn] are there to test? What happens to the number of nodes if we change the notion of a node so that it need only instantiate a subset of {Xi, ... , Xn} (leaving other variables un-instantiated)?

[5 marks]

(ii) Working with the latter notion of a node (as instantiating a subset of {X1, ... , Xn}), suppose

- a node is a *goal* node precisely if it instantiates all variables X1,...,Xn and satisfies every constraint in C
- whenever there is an arc from node N to node N', N' is not equal to N but instantiates every variable that N instantiates and gives it the value that N gives it.

Specify the depth and branching factor of the graph with start node $\emptyset$, where $\emptyset$ instantiates none of the variables.

[5 marks]

(iii) Keeping the assumptions in part (ii), consider the heuristic function that maps a node to the number of variables it does *not* instantiate. Under what conditions (on arcs and arc costs) is this heuristic function an underestimate? Describe what best-first search does under these conditions.

[10 marks]

3. (a) Recall that 3-coloring is the problem of assigning one of 3 colors (say, red, blue, green) to every node in a finite graph so that there is no arc between nodes of the same color. This question asks you to associate a *Representation and Reasoning System* (RRS) with this problem — or at least, the first two components of an RRS

   (i) a formal language specifying legal sentences, and

   (ii) semantics: specifying the meaning of the symbols.

   More precisely, to illustrate (i), give an example of

   - a sentence expressing a constraint for 3-coloring

   and to illustrate (ii), give an example of

   - an *interpretation* specifying the meaning of that sentence.

   [5 marks]

   (b) What does it mean for a clause `C` to be a *logical consequence of* a knowledge base `KB` (written `KB` $\models$ `C`)? Is this any different from saying the knowledge base `KB` with the additional clause `C` is satisfiable? Explain.

   [10 marks]

   (c) What does it mean for a reasoning system $\vdash$ to be *sound*?

   What does it mean for a reasoning system $\vdash$ to be *complete*?

   Use soundness and completeness to show that conditions (i) and (ii) below are equivalent. (That is, show (i) implies (ii), and conversely, (ii) implies (i).)

   (i) `KB` $\vdash$ `false`

   (ii) there is no interpretation where all clauses in `KB` are true

   [15 marks]

   (d) Consider the knowledge base

```
false :- p,q.
false :- a.
q :- p,a.
a :- r.
```

Suppose `p,r` were assumable in the knowledge base above. What are the *conflicts*? What are the *minimal* conflicts?

[10 marks]

(e) What does it mean for a reasoning system $\vdash$ to be *non-monotonic*? Does the clause `false :- q` lead to non-monotonicity? Does negation-as-failure lead to non-monotonicity? Explain.

[5 marks]

(f) What is *abduction* and how does it differ from *deduction*?

[5 marks]