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 and Statistics

Integrated Computer Science Programme — **Trinity Term 2017**
B.A. (Mod.) Computer Science and Business
B.A. (Mod.) Computer Science & Language
Year 3 Annual Examinations

Artificial Intelligence I

Saturday 6 May 2017 — **RDS Main Hall** — **9:30 – 11:30**

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 in each answer book used

1. (a) What is the *Halting Problem* and what does it have to do with AI?

[8 marks]

(b) What is a *Universal Turing machine* and what does it have to do with AI and the Halting Problem?

[12 marks]

(c) What is Marr's tri-level hypothesis and what does it have to do with AI?

[10 marks]

(d) What is the *Boolean Satisfiability Problem* (SAT), and what does it have to do with the question *P=NP*? What does this question have to do with AI?

[10 marks]

(e) What is a *Constraint Satisfaction Problem* (CSP)? Is the Halting Problem a CSP? Is SAT a CSP? Explain, formulating either or both as a CSP in case either or both are.

[10 marks]

2. (a) Recall that the *3-Color* problem asks if the nodes (or vertices) of a graph can be colored by exactly one of 3 colors — say, `red`, `blue` or `green` — such that there is *no* arc (or edge) between nodes of the same color. For the sake of concreteness, consider a graph with four nodes, `V1`, `V2`, `V3` and `V4`, and five arcs, {`V1,V2`}, {`V1,V3`}, {`V2,V3`}, {`V2,V4`}, and {`V3,V4`}.

   (i) Treating the nodes `V1`, `V2`, `V3`, `V4` as Prolog variables, define two Prolog predicates `generate(V1,V2,V3,V4)` and `test(V1,V2,V3,V4)` so that we can implement a brute-force generate-and-test approach to satisfying all the *3-Color* constraints for the aforementioned graph in the Prolog predicate `generate-and-test(V1,V2,V3,V4)` given by

```
generate-and-test(V1,V2,V3,V4) :-
                    generate(V1,V2,V3,V4),
                    test(V1,V2,V3,V4).
```

   Notice that because all constraints are satisfied by coloring `V1` red, `V2` blue, `V3` green, and `V4` red, the Prolog interpreter should reply positively to the query `test(red,blue,green,red)`, but not say, to `test(red,red,red,red)`.

```
?- test(red,blue,green,red).
yes.
?- test(red,red,red,red).
no.
```

   Be sure to define `generate(V1,V2,V3,V4)` so that the Prolog interpreter responds to the query `generate-and-test(V1,V2,V3,V4)` with a coloring that satisfies all constraints.

   [10 marks]

   (ii) We can use the predicate `test(V1,V2,V3,V4)` in part (i) as a goal predicate in a simple search below that starts with coloring each of `V1,V2,V3,V4` red.

```
find(Found) :- search([red,red,red,red], Found).

search([C1,C2,C3,C4],[C1,C2,C3,C4]) :- test(C1,C2,C3,C4).

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

Give a simple definition in Prolog of the predicate `arc(Node,Next)` above, and briefly outline in English what revisions might be made to `arc` for a more sophisticated search.

[10 marks]

(b) Recall that a definite clause over 0-ary predicates can be encoded as a list of these predicates so that a knowledge base consisting of such clauses becomes a list of such lists. For example, the list

```
[ [q,p], [q,h], [q,a,b], [p,r], [h,f,p], [f,h], [a], [b] ]
```

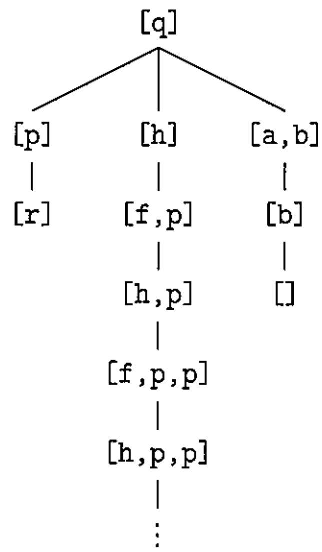
encodes the Prolog program

```
q:- p.
q:- h.
q:- a,b.
p:- r.
h:- f,p.
f:- h.
a.
b.
```

Given this program, the query

```
?- q.
```

leads in Prolog to a proof search over a tree that can be drawn as follows, where a node is a list of atoms to prove.



(i) Define predicates arc(Node,Next,KB) and goal(Node) so that the predicate query(Q,KB) below checks whether the atom Q follows from the knowledge base encoded by the list KB of lists, where

```
query(Q,KB) :- search([Q],KB).

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

[10 marks]

(ii) Notice that there is an infinite path in the search tree for q against

```
KB = [[q,p],[q,h],[q,a,b],[p,r],[h,f,p],[f,h],[a],[b]]
```

leading to a loop on the query

```
?- query(q,[[q,p],[q,h],[q,a,b],[p,r],[h,f,p],[f,h],[a],[b]]).
```

One way to get around this problem is to reverse the top-down search (from [q] to []) to a bottom-up search from [] to an appropriate goal node containing the query, as outlined below.

```
queryBU(Q,KB) :- searchBU([],KB,Q).

searchBU(Node,_,Q) :- member(Q,Node).
searchBU(Node,KB,Q) :- arcBU(Node,Next,KB),
                       searchBU(Next,KB,Q).
```

Complete the definition of searchBU(Node,KB,Q) above by defining the predicate arcBU(Node,Next,KB).

[10 marks]

(c) What is *A-star* search? Suppose you had to apply A-star to either the 3-color problem in part (a) or the KB-query problem in part (b). Which would you choose, and describe how to apply A-star to that choice.

[10 marks]

3. (a) Recall that *Datalog* is a declarative subset of Prolog that is defined so that every finite set of Datalog clauses has a *least* interpretation, and moreover, that interpretation is finite.

   (i) Describe what an *interpretation* is by giving an interpretation for the following knowledge base

```
number(0).
number(succ(X)) :- number(X).
```

   Do both clauses above belong to Datalog?

   [10 marks]

   (ii) How can a model of a Datalog knowledge base KB be understood as a *fixed point*?

   [10 marks]

   (iii) Explain how the least fixed point associated with a Datalog knowledge base KB picks out the *logical consequences of* KB.

   [5 marks]

   (iv) What is the greatest fixed point associated with a Datalog knowledge base?

   [5 marks]

   (b) Recall that a *Horn clause* is a definite clause where the head can be a special atom `false` that is true in *no* interpretation.

   (i) Let `a` and `b` be atoms. Can we use `false` to form a knowledge base whose models are precisely the interpretations where `a` *or* `b` is true? Justify your answer.

   [5 marks]

   (ii) How can we use logical consequence to tell whether a knowledge base of Horn clauses has a model?

   [5 marks]

   (iii) What is *abduction* and how can `false` be used in abduction?

   [5 marks]

(iv) We can express the statement "birds fly" as the following default rule

$$\frac{\mathsf{bird}(X)\ :\ \mathsf{fly}(X)}{\mathsf{fly}(X)}.$$

Describe how to use `false` to allow for exceptional birds such as penguins that don't fly.

[5 marks]