Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Symbolic AI

Date: Friday 25th May 2018
Time: 14:00 - 16:00

Please answer all Questions.

Include your working – marks will be awarded for partial answers where the appropriate method has been used

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
1. In all the parts of this question, the given Prolog program may throw an exception for one or more of the queries. Your answers should identify such cases.

a) What will the following Prolog queries do? [6 marks]

| ?- X = Y, X == Z, Y = Z. |
| ?- X = Y, Y = Z, X == Z. |
| ?- X = Y*4. |
| ?- X is Y*4. |
| ?- 12 = 3*4. |
| ?- 12 is 3*4. |

b) Consider the following Prolog program:

```
p(X0, X1) :-
    number(X0),
    number(X1),
    X0 =< X1.
p(X, [H | _T]) :-
    p(X, H).
p(X, [_H | T]) :-
    p(X, T).
q([], _). q([H0 | T0], [H1 | T1]) :-
    p(H0, H1),
    q(T0, T1).
q(L, [_H | T]) :-
    q(L, T).
```

What steps would this program carry out if you called it with the following arguments? [8 marks]

| ?- q([4], [6]). |
| ?- q([4, 0], [6]). |
| ?- q([1], [[2, 1]]). |
c) Consider the Prolog program below:

```prolog
p(X, Y) :-
    Y is X+1,
    assert(t(X, Y)).
p(X, Y) :-
    t(A, Y),
    A =< X.
```

What steps would this program carry out if you called it with the following sequence of arguments without adding anything to the database or removing anything from it in between queries? In each case, if it found one answer say what would it do if you forced it to look for alternative answers? What will be in the database afterwards?

[6 marks]

| ?- p(2, P). |
| ?- p(1, P). |
| ?- p(2, P). |
| ?- p(4, P). |
2. The program in Figure 1 provides an implementation of the model-generation theorem
prover SATCHMO:

\[
\begin{align*}
horn(X) & : - \\
& \quad fact(X); assumption(X).
prove(G) & : - \\
& \quad horn(G).
\\
horn(X) & : - \\
& \quad LHS ==> X, \\
& \quad horn(LHS).
prove(X) & : - \\
& \quad horn(P or Q), \\
& \quad prove(P ==> X), \\
& \quad prove(Q ==> X).
\\
horn(A & B) & : - \\
& \quad horn(A), \\
& \quad horn(B).
prove(P ==> Q) & : - \\
& \quad assert(assumption(P)), \\
& \quad (horn(Q) -> \\
& \quad retract(assumption(P))); \\
& \quad (retract(assumption(P)), fail)).
\\
horn(A or B) & : - \\
& \quad nonvar(A), \\
& \quad (horn(A); horn(B)).
\end{align*}
\]

Figure 1: Basic implementation of SATCHMO

a) Show the steps that this theorem prover would go through in order to derive \text{female(stacy)} \\
& \quad \text{& parent(stacy, laura)} from \text{[mother(X, Y) ==> female(X), mother(X, Y) ==> parent(X, Y), mother(stacy, laura)].} \quad \text{[6 marks]}

b) Describe how you might use labels to extend the program in Figure 1 to enable it to 
catch loops. You do not need to produce new Prolog versions of the rules in Figure 
1 – a general discussion of the issues is enough. Describe one other way that labels 
might be useful. \quad \text{[5 marks]}

c) Explain the difference between ‘material implication’ and ‘constructive 
implication’.

The final rule in Figure 1 implements one of these notions: say which, and illustrate 
your answer by showing how this implementation of SATCHMO would derive 
\text{architect(allan) ==> constructor(allan)} from \text{[architect(X) ==> builder(X), 
builder(X) ==> constructor(X)]} (you may assume that variables are denoted 
by upper-case letters and that all free variables are universally quantified). \quad \text{[7 marks]}

Page 4 of 7
3. a) What is the difference between a ‘context-free’ grammar and a ‘feature-based’ grammar? You should illustrate your answer by writing a grammar of each kind that allowed

(1) a. I slept
b. I know John
c. I know John loves Mary
d. I met John

and disallowed

(2) a. I slept John
b. I met John loves Mary

[8 marks]

b) Outline the standard top-down and bottom-up algorithms for parsing: illustrate your answer by showing the steps that one of these algorithms would go through in order to obtain an analysis of ‘I know John loves Mary’ according to one of the grammars you gave as an answer to (3a). [5 marks]

c) The lexicon in Figure 2 describes the way that words combine by specifying the items that they would like to combine with and the position where those items are expected to appear. Explain how a grammar of this kind be seen as leading to the construction of dependency tree analyses. You should illustrate your answer by showing how this grammar would assign analyses to the sentences ‘I saw him’ and ‘I know she loves him’. [7 marks]

```
np(X) :-
cat@X -- n,
args@X -- [ ].
s(X) :-
cat@X -- v,
args@X -- [ ].
tverb(X, TYPE) :-
cat@X -- v,
args@X -- [OBJ, SUBJ],
OBJ <> [np, after],
SUBJ <> [np, before].
sverb(X, TYPE) :-
cat@X -- v,
args@X -- [OBJ, SUBJ],
OBJ <> [s, after],
SUBJ <> [np, before].
```

Figure 2: Simple categorial grammar
4. a) Suppose you had a set of entailment relations between words, e.g. that man $\models$ human, human $\models$ animal, love $\models$ like, ... Outline a matching algorithm which would let you infer that the tree in Fig 3 for ‘Mary loves a man’ entails the one for ‘Mary likes a human’. You should use this example to illustrate the way your algorithm works.

```
tverb:love
  subject:name:Mary
  object:noun:man
  spec:det:exists
```

```
tverb:like
  subject:name:Mary
  object:noun:human
  spec:det:exists
```

Figure 3: Dependency trees for ‘Mary loves a man’ and ‘Mary likes a human’

b) How could you change this algorithm to enable it to cope with the fact that a simple positive sentence containing modifiers entails the same sentence without the modifiers, e.g. that ‘I saw a fat man sleeping in the park’ entails ‘I saw a man sleeping’? Again, you should illustrate your answer by referring to the trees in Figure 4.

```
sverb:see
  subject:pron:I
  iverb:sleep
    subject:noun:man
    mod:prep:in
      mod:adj:fat
      spec:det:exists
      comp:noun:park

sverb:see
  subject:pron:I
  iverb:sleep
    subject:noun:man
    spec:det:exists
```

Figure 4: Dependency trees for ‘I saw a fat man sleeping in the park’ and ‘I saw a man sleeping’
c) What changes would you have to make to the theorem prover in Figure 1 to enable it to cope with rules expressed using natural language subtrees of the kind shown in Figure 5? Show the steps that your adapted version of the theorem prover would go through in showing that ‘The earth is flat’ follows from ‘I know the earth is flat’

[8 marks]

Figure 5: Dependency trees for ‘A likes B if A loves B’ and ‘A Ps B if C knows A Ps B’