COMP24412

Two hours

Question ONE is COMPULSORY

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Symbolic AI

Date: Monday 23rd May 2016

Time: 09:45 - 11:45

Please answer Question ONE in Section A and any TWO other Questions in Section B

This is a CLOSED book examination

The use of electronic calculators is permitted provided they
are not programmable and do not store text

[PTO]

Section A

You should answer question 1: this question carries 30 marks

1. a) What will the following Prolog queries do? Note that some of them may throw exceptions. **[6 marks]**

```
| ?- X = Y, Z = 6, X == Z.

| ?- X = Y, Z = 6, Z = X, X == Z.

| ?- X = Y*4.

| ?- X is Y*4.

| ?- 3*4 is 12.

| ?- [a,b,c,d] = [A, B | C].
```

b) Consider the following Prolog program.

```
p(_H, [], []).
p(H, [H | T0], T1) :-
    !,
    p(H, T0, T1).
p(X, [H | T0], [H | T1]) :-
    p(X, T0, T1).

q([], L, L).
q([H | T], L0, L2) :-
    p(H, L0, L1),
    q(T, L1, L2).
```

i. What steps would this program carry out, and what would be the result, if you called it with

```
| ?- p(3, [1,2,3,4,3,4], L).
```

[4 marks]

ii. What would it do if you removed the cut from the definition of `p` and called it with

```
| ?- p(2, [1,2], L).
```

and then after each answer that it produced you typed `;` to force it to look for an alternative answer? **[3 marks]**

iii. What would the result be if you called

```
| ?- q([4,1], [1,2], L).
```

(using the definition of `p` that **includes** the cut: you do **not** need to go through all the steps that the calls of `p` would involve). **[3 marks]**

c) Consider the following Prolog program:

```
r(I, 1) :-
    I < 2.
r(I, X) :-
    J1 is I-1,
    r(J1, Z),
    J2 is I-2,
    r(J2, Y),
    X is Y*Z.
```

i. What steps would this program carry out and what would be the result if you called it with

```
| ?- r(2, X).
```

Note: this function is **not** the Fibonacci series.**[4 marks]**

ii. What would have happened if instead of accepting the first answer you had forced it to produce another answer? **[3 marks]**

iii. If you keep asking it for alternative answers, it will keep generating them. What would you have to do to it to make sure that it produced exactly one answer whenever you called it? **[3 marks]**

d) For each of the following pairs of terms, say whether they unify with each other (i.e. do `[A, B, C]` and `[[A, B, C] | T]` unify? For ones which do unify, what would the variables bindings be, and for ones which do not explain why they do not. **[4 marks]**.

i.
```
[A, B, C]
[[A, B, C] | T]
```

ii.
```
sign(syntax(head(cat(noun), A, B, C), D, args([])),
     E,
     structure(F, G, H, I, form(J), tree([J])),
     noun(J))
sign(A, B, structure(C, D, E, F, form(sleeps), G), H)
```

iii.
```
sign(syntax(head(A0, agree(B0), C0, D0), E0, F0), G0, H0, I0)
sign(syntax(head(A1, B1, C1, D1), E1, F1), G1, H1, I1)
```

iv.
```
sign(syntax(A0, B0, args([C0, D0])), E0, F0, G0)
sign(syntax(A1, B1, args([C1])), D1, E1, F1)
```

Section B

Answer two questions from this section. Each question carries 35 marks.

2. a) What is the difference between a 'context-free' grammar and a 'feature-based' grammar? You should illustrate answer by considering how you write a grammar that allowed

(1) a. I saw her
b. She saw me

and disallowed

(2) a. Her saw I
b. Me saw she

[8 marks]

How would you describe '*it*' in each framework, given that '*It saw me*' and '*I saw it*' are both legal sentences? **[2 marks]**

b) Outline the basic top-down and bottom-up algorithms for parsing with a simple context-free grammar. **[4 marks]** Show the steps that **one** of the algorithms would take when parsing the sentence '*he runs a shop*' using the grammar in Fig 1. You should show what each step does, not just the final parse tree. **[6 marks]**

```
s ==> [np, vp].
vp ==> [iverb].
vp ==> [tverb, np].
np ==> [det, nn].
```

```
word(a, det).
word(he, np).
word(runs, iverb).
word(runs, tverb).
word(shop, nn).
```

Figure 1: Major S and VP rules

c) What is the fundamental rule of chart parsing? **[2 marks]** Show the steps that a left-corner chart parser would perform when analysing the sentence *he knows you ate it* with the grammar in Fig 1. **[8 marks]**

d) What would you have to do to the grammar and lexicon from Fig 1 to make them cover the sentences '*He does love her*' and '*Does he love her*'? **[5 marks]**

3. a) Natural language understanding systems often translate the input text into an expression in some logic. It is common practice to use first-order logic for this purpose: give two examples of phenomena in natural language which cannot be captured in first-order logic. **[6 marks]**

   b) State the 'principle of compositionality'. **[2 marks]** Explain what lexical ambiguity, structural ambiguity and scope ambiguity are. You should illustrate your answer with examples of each kind of ambiguity. **[3 marks]** What problems do each of these forms of ambiguity pose for hopes of providing a compositional treatment of semantics? **[5 marks]**

   c) Show the interpretation that the grammar and lexicon in Fig. 2 would assign to (1) below. **[14 marks]** *Include your working–just providing the right answer without showing how you arrived at it will be worth 0, while answers that are proceeding along the right lines but don't quite get there will get partial marks*

   (1) Every man loves some woman

```
[cat=s, meaning=VP:NP]
         ==> [[cat=np, meaning=NP], [cat=vp, meaning=VP]].
[cat=np, meaning=DET:N]
         ==> [[cat=det, meaning=DET], [cat=noun, meaning=N]].
[cat=vp, meaning=V:NP] ==> [[cat=verb, meaning=V], [cat=np, meaning=NP]].

lexEntry('every',
     [cat=det,
      meaning=lambda(P, lambda(Q, forall(X, (P:X => Q:X))))]).
lexEntry('some',
     [cat=det,
      meaning=lambda(P, lambda(Q, exists(X, (P:X & Q:X))))]).
lexEntry('man', [cat=noun, meaning=lambda(U, man(U))]).
lexEntry('woman', [cat=noun, meaning=lambda(U, woman(U))]).
lexEntry('loves',
     [cat=verb,
      meaning=lambda(B,
                     lambda(C,
                            C:(lambda(X, B:lambda(Y, exists(Z, love(Z)
                                                          & agent(Z, X)
                                                          & object(Z, Y))))))]).
```

Figure 2: Semantically annotated grammar

   d) Does the interpretation that you have arrived at for (1) say that each man loves a (probably different) woman, or is there just one woman that every man loves? **[1 marks]** What would you have to do to the grammar in Fig. 2 to obtain the alternative interpretation (i.e. if your interpretation says there is a different woman for each man, how could you rewrite the grammar and/or lexicon so that there is just one woman who is loved by every man?). **[4 marks]**

4. a) The following program provides a basic implementation of the 'model generation' approach to theorem proving for first-order logic.

```
horn(P) :-
    P.
horn(P & Q) :-
    horn(P), horn(Q).
horn(P or Q) :-
    horn(P); horn(Q).
horn(P) :-
    nonvar(P),
    Q ==> P,
    horn(Q).

prove(P) :-
    horn(P).
prove(P) :-
    (R or S),
    cprove(R ==> P),
    cprove(S ==> P).

cprove(P ==> Q) :-
    nonvar(Q),
    assert(P),
    (prove(Q) -> retract(P); (retract(P), fail)).
```

Figure 3: Basic rules for SATCHMO

b) Explain what each element of this program is for **[6 marks]**, and show how it could be used to derive `r(2)` from {`p(2) or q(2)`, `p(X) ==> r(X)`, `q(X) ==> r(X)`} **[3 marks]**.

The algorithm in Fig. 3 does not contain any rules for dealing with negation. Explain how negation can be handled by this algorithm by treating `not(P)` as though it were a shorthand for `P ==> absurd`, and show how the theorem prover above could be used to derive `not(tall(john))` from `short(john)` and `not(short(X) & tall(X))` **[4 marks]**.

c) Consider the extension of first-order logic that you obtain by allowing 'default' rules of the form

```
most(X, swan(X) => white(X))
```

A sensible proof step for rules of this kind says that you can use such a rule unless

it contradicts either something which is provable without using any other default rules or some other default rule that you have used somewhere else. Show that the set of rules in Fig. 4 (below) will allow you to prove each of `black(bruce)` and `white(bruce)` in isolation, but not `black(bruce) & white(bruce)` **[8 marks]** What is meant by saying that this logic 'non-monotonic'? **[2 marks]**

```
swan(bruce).
australian(bruce).
most(X, swan(X) => white(X)).
most(X, swan(X) & australian(X) => black(X)).
forall(X, not(black(X) & white(X))).
```

Figure 4: Default rules

d) Outline the STRIPS notation for describing actions, illustrating your answer with a description of the actions of raising and lowering a robot hand, moving it around and grasping and ungrasping a block. **[4 marks]** Describe how backwards chaining planning works. You should illustrate your answer by showing how a plan to transform a situation that fits the description `on(a, b), on(b, table), on(c, table), handempty, handLow, clear(a), clear(c)` could be turned into one where the goal `on(a, c)` would be true. **[8 marks]**

END OF EXAMINATION