COMP24412

Two hours

Question ONE is COMPULSORY

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Symbolic AI

Date: Friday 30th May 2014

Time: 14:00 - 16:00

Answer Question ONE in Section A and TWO Questions from 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? **[6 marks]**

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

| ?- X == Y.

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

| ?- X is 3*4.

| ?- X = 3*4.

| ?- 3*4 is 12.
```

b) Consider the following Prolog program.

```
p(X, X).
p(X, [H | _T]) :-
    p(X, H).
p(X, [_H | T]) :-
    p(X, T).
```

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

```
| ?- p(3, [[[a, [3], b]]]).
```

[3 marks]

ii. What would it do if you called it with

```
| ?- p(X, [[[a, [3]]]]).
```

and after each answer that it produced you typed `;` to force it to look for an alternative answer? **[2 marks]**

iii. Why have I specified the tail of the list in the second rule as `_T` and the head in the third rule as `_H`. **[1 marks]**

c) Consider the following Prolog program:

```
r(0, X) :-
    p(X).
r(I, X) :-
    assert(p(I)),
    J is I-1,
    r(J, X).
```

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

```
| ?- r(3, X).
```

[3 marks]

ii. What will happen if you force it to produce another answer? **[3 marks]**

d) Consider the following Prolog program.

```
p(X, [X | _]).
p(X, [_ | Y]) :-
    p(X, Y).

q([], _L).
q([X=Y0 | Z], L) :-
    p(X=Y1, L),
    !,
    Y0 = Y1,
    q(Z, L).
q([_X | Z], L) :-
    q(Z, L).
```

i. What would this program do if you called it with the following arguments? **[8 marks]**

```
| ?- q([a=9, b=10], [a=9, c=11, b=10]).
| ?- q([a=9, b=10], [a=9, c=11, b=10, b=11]).
| ?- q([a=9, b=10], [a=9, c=11, b=11, b=10]).
| ?- q([a=X, b=X], [a=9, c=11, b=X]).
| ?- q([a=9, b=10], [a=Y, c=11, b=Y]).
| ?- q([a=9, b=10], [a=Y, c=11]).
```

ii. What would happen if the cut in the second rule were removed? Illustrate your answer by considering what would happen in the cases above where the original program failed. **[4 marks]**.

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 your answer with examples that would be easier to account for using a feature-based grammar than with a context-free grammar. **[10 marks]**

   b) Describe how categorial descriptions of lexical items can be used to cut the number of rules required for describing how a sentence can be decomposed into its major parts. You should illustrate your answer by considering the set of rules and lexical items in Fig 1 and showing what the categorial lexical entries would look like and what rules you would still need. **[10 marks]**

```
s ==> [np, vp].
vp ==> [iverb].
vp ==> [tverb, np].
vp ==> [sverb, s].
lexEntry(you, np).
lexEntry(he, np).
lexEntry(it, np).
lexEntry(ran, iverb).
lexEntry(saw, tverb).
lexEntry(knows, tverb).
lexEntry(knows, sverb).
```

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) Explain why grammars that consist solely of sets of rewrite rules have difficulty with situations where items occur in marked/non-canonical positions.**[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 why the sentences below pose a challenge for this principle. **[8 marks]**

   (3)
   a. I broke a glass jam jar.
   b. I got caught in a jam.
   c. I want to buy a bike.

   c) Show the interpretation that the grammar and lexicon would assign to '*Every man will die*'. **[14 marks]** *Include your working–just providing the right answer without showing how you arrived at it will be worth 0*

```
[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] ==> [[cat=verb, meaning=V]].
[cat=vp, meaning=AUX:VP] ==> [[cat=aux, meaning=AUX], [cat=vp, meaning=VP]].

lexEntry('every',
     [cat=det,
      meaning=lambda(P, lambda(Q, forall(X, (P:X => Q:X))))]).
lexEntry('man', [cat=noun, meaning=lambda(U, man(U))]).
lexEntry('will', [cat=aux, meaning=lambda(A, lambda(B, future(A:B)))]).
lexEntry('die',
     [cat=verb,
      meaning=lambda(B, B:(lambda(X, exists(Z, die(Z) & patient(Z, X))))))]).
```

   d) Explain how using appropriate sets of thematic roles can account for the similarities between (1a) and (1c) and between (1b) and (1d) below. **[5 marks]**

   (1)
   a. I saw him playing his guitar.
   b. I watched him playing his guitar.
   c. I heard him playing his guitar.
   d. I listened to him playing his guitar.

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 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)).
```

   i. Explain what each element of this program is for **[6 marks]**, and show how it could be used to derive `r(2)` from {`p(X) or q(X) ==> r(X), p(2), q(2)`} **[3 marks]** and to derive `r(2)` from {`p(2) or q(2) p(X) ==> r(X), q(X) ==> r(X)`} **[4 marks]**.

   ii. It is easy for the program above to get stuck in a loop. Give a set of rules and a goal which will make this happen. **[1.5 marks]**. Describe how using a 'label' can be used to cope with this problem. **[3.5 marks]** Can you catch all and only genuine loops by the method you have described? **[2 marks]**

b) i. Outline the STRIPS notation for describing actions, illustrating your answer with a description of the action of grasping a block. **[4 marks]** Briefly describe how backwards chaining planning works. **[4 marks]**

   ii. **EITHER**
   Describe the problem of subgoal interactions. **[2 marks]** Explain how using a set of protected goals can help solve this problem. **[5 marks]**
   OR

   iii. What is the 'ramification problem' in planning. **[2 marks]** Outline how this problem could be dealt with by incorporating an inference engine that can carry out abductive reasoning into the basic backwards chaining planner that you have described above. **[5 marks]**

END OF EXAMINATION