Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Symbolic Artificial Intelligence [MOCK EXAM QUESTIONS]

X

Time: 2 hours

Marking Scheme Included

Do not publish

This is a HYBRID exam, with sections to be answered online and questions to be answered on paper.

Answer ALL questions.

Use a SEPARATE answerbook for each paper SECTION.

For full marks your answers should be concise as well as accurate.

Marks will be awarded for reasoning and method as well as being correct.

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

Section A

Answer ALL questions. These questions should be answered ON PAPER

1. **[Modelling and Representation]**

   For this question you should select **one** of the representation formalisms covered in the course e.g. either Datalog, Prolog, or First-Order logic.

   You have graduated and got your first job as a Computer Wizard at London Zoo. Your first job is to create an intelligent system for

   a) You have a brief chat with the *Director of the Zoo* and he scribbles down some examples of facts you need to store in your system as follows:

      - There are distinct categories of animals such as Amphibian, Arachnid, Bird, Fish, Insect, Mammal, and Reptile
      - Every environment has a maximum capacity and a type of climate
      - Both Amphibians and Fish need to live in marine environments
      - Tigers are large animals, carnivores, and mammals
      - Iguanas are small animals, herbivores, and reptiles
      - Bhanu is a Lion of the Asiatic species and is kept in environment B12
      - Environment A4 is accessible from environment A2

      Describe a set of predicates that you would use to capture these facts and then use them to represent the facts in your chosen formalism. (3 marks)

Model answer: There isn't a single correct solution here. A good attempt is likely to receive full marks (see mark distribution section). We introduce the following predicates

- $is\_a(species, category), species(species)$
- $environment(name, capacity, climate)$
- $large(species), small(species)$ etc.
- $carnivore(species)$,$herbivore(species)$, $omnivore(species)$
- $species\_of(animal\_name, species)$, $subspecies(species1, species2)$
- $kept\_in(animal\_name, environment\_name)$
- $accessible\_from(environment1, environment2)$

Some of the above points don't necessarily relate to facts but to the existence of the above predicates. Some of the facts can be written as follows:

- $\forall x.(species(x) \rightarrow (is\_a(x, amphibian) \vee \ldots \vee is\_a(x, reptile)))$ - a disjunct over all the categories of animal
- $\forall x.(species(x) \rightarrow (\neg is\_a(x, amphibian) \wedge \neg is\_a(x, fish)))$ - repeated for each pair of categories
- $\forall a, s.((species\_of(a, s) \wedge (is\_a(s, amphibian) \vee is\_a(s, fish))) \rightarrow (\exists e, y.(kept\_in(a, e) \wedge environment(e, y, marine))))$
- $large(tiger) \wedge carnivore(tiger) \wedge is\_a(tiger, mammal)$
- $small(iguana) \wedge herbivore(iguana) \wedge is\_a(iguana, reptile)$
- $species\_of(bhanu, asiatic\_lion) \wedge subspecies(asiatic\_lion, lion) \wedge kept\_in(bhanu, b12)$
- $accessible\_from(a4, a2)$

Many of these require first-order logic, for example totality of categories. But it is possible to get full marks using Prolog or Datalog if a comment is added to this effect.

Distribution of Marks: After creating the model solution it has become clear that the marks for this question are too low. This is a 6 mark question. There are three parts of the problem - defining categories/species of animals, defining properties such as small/carnivore etc, and describing environments. 2 marks for each of these with 1 mark for some appropriate predicates and 1 mark for 1 or more correctly formed statement.

b) You visit Susan, the *Head of Stopping Animals from Eating Each Other*, and she says that she wants to check that carnivores are only in environments accessible from other environments containing the same species of animal. Use your formalism to design some rules and a single query to allow Susan to check this. (3 marks)

Model answer: Let's assume that Prolog is being used (even though I used FOL above). The important thing is to recursively define the transitive extension of *accessible_from*. In Prolog we simply do this as follows

```
accessible(X,Y) :- accessible_from(X,Y).

accessible(X,Y) :- accessible_from(Y,X).

accessible(X,Y) :- accessible_from(X,Z),
 accessible(Z,Y).
```

I added the second line as I'm assuming that accessibility should be symmetric, even though this isn't explicitly stated in the question. This isn't necessary. I'm going to also introduce some helper predicates:

```
contains(E,S) :- keptIn(A,E), speciesOf(A,S).

containsC(E,S) :- contains(E,S), carnivore(S).
```

Now for the query we want to find two environments that are accessible from each other such that they contain different species where one of those is carnivorous.

```
accessible(E1,E2), diff(S1,S2), contains(E1,S1),
containsC(E2,S2).
```

Distribution of Marks: 1 mark for noting that accessible must be made transitive, 1 mark for defining this transitivity and 1 mark for putting things together to answer the query.

c) Finally, some of the old Computer Wizards have been assigned to your project. They are familiar with the old system that used a relational database. Write a brief summary of how your chosen representation formalism relates to relational databases. You may find it helpful to refer to tables that might have existed in the old database and how they would relate to relations defined in your formalism. (3 marks)

Model answer: Predicates (in all formalisms) can be related to tables in a relational database. For example, a previous table might have listed animals and their species, this now becomes the predicate *species_of*. Importantly, relational databases have a *closed world assumption*, similar to Prolog and Datalog, but different form first-order logic. Rules then represent different kinds of queries that can be run on the tables (to generate new tables/views). Relational databases typically do not allow for recursively-defined queries, which is where the formalisms here are more expressive.
Distribution of Marks: 1 mark for predicates are tables. 1 mark for closed-world assumption. 1 mark for discussion of queries.

2. **[Reasoning]** Use the resolution and superposition calculus (the rules are given at the end of the paper) to give a proof that this set of formulas

$$\forall x.(is\_in(x, fruit\_salad) \rightarrow fruit(x))$$
$$\forall x.(is\_in(x, potato\_salad) \rightarrow (potato(x) \vee onion(x)))$$
$$\forall x.(potato(x) \rightarrow vegetable(x))$$
$$\forall x.(onion(x) \rightarrow vegetable(x))$$
$$\forall x.\neg(fruit \leftrightarrow vegetable(x))$$

entails

$$\neg\exists x.(is\_in(x, fruit\_salad) \wedge is\_in(x, potato\_salad)$$

You do not need to use the given clause algorithm (although may do so). Recall that you will first need to transform your problem into clausal form (the rules are given at the end of the paper). (6 marks)

Model answer:
The first step is to turn these into clauses as follows

1. $\neg is\_in(x, fruit\_salad) \vee fruit(x)$
2. $\neg is\_in(x, potato\_salad) \vee potato(x) \vee onion(x)$
3. $\neg potato(x) \vee vegetable(x)$
4. $\neg onion(x) \vee vegetable(x)$
5. $fruit(x) \vee vegetable(x)$
6. $\neg fruit(x) \vee \neg vegetable(x)$

The goal should then be negated and turned into clauses to get

7. $is\_in(a, fruit\_salad)$
8. $is\_in(a, potato\_salad)$

where *a* is a new Skolem constant.

Now we apply resolution a number of times. It is not necessary to use ordered resolution.

9. $fruit(a)$ by resolving 1 and 7
10. $potato(a) \vee onion(a)$ by resolving 2 and 8
11. $onion(a) \vee vegetable(a)$ by resolving 3 and 10
12. $vegetable(a)$ by resolving 4 and 11
13. $\neg vegetable(a)$ by resolving 9 and 6
14. *false* by resolving 13 and 6

Distribution of Marks:

Section B

Answer ALL questions. These questions should be answered ON PAPER

This is a second set of paper questions. In the actual exam there will only be one set of paper questions. The other 15 marks will be online and follow the style of the quizzes.

3. **[Modelling and Representation]**

   a) Write a set of Prolog rules to define the following predicates

   - `musician(X)` should be true if person `X` plays an instrument or sings
   - `band(X)` should be true if `X` is a list of musicians
   - `bad_band(X)` should be true if `X` is a band without a drummer (somebody who plays the drums)

   (3 marks)

Model answer:

```
musician(X) :- plays(X,Y), instrument(Y).

band([]) :- musician(X).
band([X|Xs]) :- musician(X), band(Xs).

doesnotplay(_, []).
doesnotplay(X,[Y|Ys]) :-
 plays(Y,Z), dif(Z,X),
 doesnotplay(X,Ys).

bad_band(Band) :- band(Band), doesnotplay(drums,Band).
```

The below formulation assumes that a person does not play more than one instrument e.g. if somebody plays something that is not the drums then they do not play the drums. This is a bit messy and not a great exam question because of this. The formulation should have been positive e.g. a good band is one with a drummer.

Distribution of Marks: A mark for each rule.

b) Briefly discuss advantages and disadvantages of using Prolog for this task rather than Datalog or first-order logic (2 marks)

> **Model answer:** The main advantage of using Prolog is that it has good native support for lists (which are not native to Datalog or first-order logic). A disadvantage in this case is that we really wanted negation but Prolog's negation-as-failure is generally non-logical - although it would have been possible to use negation above in such a way that it remained logical (e.g. was only applied to ground terms).
> **Distribution of Marks:** A mark per valid point made.

4. **[Reasoning]** Updated: originally a negation was missing, then it was incorrectly 'corrected' ... although the correction does produce a solvable problem, it was not the problem originally intended.

a) Translate each of the following statements in first-order logic into English

i. $\forall x.\neg(cat(x) \leftrightarrow dog(x))$
ii. $cat(garfield) \wedge dog(odie)$
iii. $likes(garfield, lasagne)$
iv. $\forall x, y, z.((cat(x) \wedge dog(y) \wedge likes(x, z)) \rightarrow \neg likes(y, z))$

(2 marks)

> **Model answer:**
>
> i. Cats and Dogs are disjoint/distinct
>
> ii. garfield is a cat and odie is a dog
>
> iii. garfield likes lasagne
>
> iv. Everything that a cat likes, a dog does not like
>
> **Distribution of Marks:** 0.5 marks for each statement

b) Transform the above formulas into clausal form. The clausification rules are given at the end of this section. (2 marks)

Model answer:

1. $cat(x) \vee dog(x)$
2. $\neg cat(x) \vee \neg dog(x)$
3. $cat(garfield)$
4. $dog(odie)$
5. $likes(garfield, lasagne)$
6. $\neg cat(x) \vee \neg dog(y) \vee \neg likes(x,z) \vee \neg likes(y,z)$

Distribution of Marks: 0.5 marks for translating each of the original statements

c) Apply the *given clause* algorithm with the rules (ordered) resolution, paramodulation, and equality resolution to show that the above set of formulas entails the formula

$$\neg likes(odie, lasagne)$$

These rules are given at the end of this section. You may select your own clause ordering and literal ordering. (6 marks)

Model answer:

We represent the given clause algorithm as a table with a column indicating at what step in the algorithm a clause is produced and a column indicating at what step in the algorithm the clause was activated (selected to be placed in the active set). Selected literals are underlined - as we only rely on selecting a single literal or a negated literal we do not need to specify a symbol ordering. A different clause selection ordering would lead to a longer table but the same proof.

			Prod.	Activ.
1	$cat(x) \vee \underline{dog(x)}$		0	
2	$\neg cat(x) \vee \underline{\neg dog(x)}$		0	
3	$\underline{cat(garfield)}$		0	1
4	$\underline{dog(odie)}$		0	2
5	$\underline{likes(garfield, lasagne)}$		0	3
6	$\neg cat(x) \vee \neg dog(y) \vee \neg likes(x,z) \vee \underline{\neg likes(y,z)}$		0	5
7	$\underline{likes(odie, lasagne)}$		0	4
8	$\neg cat(x) \vee \neg dog(garfield) \vee \underline{\neg likes(x, lasagne)}$	[6,5]	5	
9	$\neg cat(x) \vee \neg dog(odie) \vee \underline{\neg likes(x, lasagne)}$	[6,7]	5	6
10	$\underline{\neg dog(odie)} \vee \neg cat(garfield)$	[9,5]	6	7
11	$\underline{\neg dog(odie)} \vee \neg cat(odie)$	[9,7]	6	
12	$\underline{\neg cat(garfield)}$	[10,4]	7	8
13	$false$	[12,3]	8	

Distribution of Marks: 1 mark for remembering to negate the entailed formula. 1 mark for correctly applying at least one resolution step. 2 marks for correct application of the given clause algorithm. 2 marks for completing the proof.

Clausifcation Rules for First-Order Logic

Negation Normal Form

$$\begin{array}{rcl}
\neg(F_1 \wedge \ldots \wedge F_n) & \Rightarrow & \neg F_1 \vee \ldots \vee \neg F_n \\
\neg(F_1 \vee \ldots \vee F_n) & \Rightarrow & \neg F_1 \wedge \ldots \wedge \neg F_n \\
F_1 \rightarrow F_2 & \Rightarrow & \neg F_1 \vee F_2 \\
\neg\neg F & \Rightarrow & F \\
\neg\forall x_1,\ldots,x_n F & \Rightarrow & \exists x_1,\ldots,x_n \neg F \\
\neg\exists x_1,\ldots,x_n F & \Rightarrow & \forall x_1,\ldots,x_n \neg F \\
\neg(F_1 \leftrightarrow F_2) & \Rightarrow & F_1 \otimes F_2 \\
\neg(F_1 \otimes F_2) & \Rightarrow & F_1 \leftrightarrow F_2 \\
F_1 \leftrightarrow F_2 & \Rightarrow & (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1); \\
F_1 \otimes F_2 & \Rightarrow & (F_1 \vee F_2) \wedge (\neg F_1 \vee \neg F_2).
\end{array}$$

Skolemization

$$\begin{array}{l}
\forall x_1,\ldots,x_n F \;\Rightarrow\; F \\
\exists x_1,\ldots,x_n F \;\Rightarrow\; F\{x_1 \mapsto f_1(y_1,\ldots,y_m),\ldots,x_n \mapsto f_n(y_1,\ldots,y_m)\},
\end{array}$$

Clausal Normal Form

$$\begin{array}{rcll}
(A_1 \wedge \ldots \wedge A_m) \vee B_1 \vee \ldots \vee B_n & \Rightarrow & (A_1 \vee B_1 \vee \ldots \vee B_n) & \wedge \\
 & & \cdots & \wedge \\
 & & (A_m \vee B_1 \vee \ldots \vee B_n). &
\end{array}$$

Reasoning Rules for First-Order Logic

Ordered Resolution

$$\frac{l_1 \vee C \qquad \neg l_2 \vee D}{(C \vee D)\theta}\ \theta = \mathsf{mgu}(l_1, l_2)$$

where l_1 and $\neg l_2$ are selected by a well-behaved selection function.

Paramodulation and Equality Resolution

$$\frac{C \vee s = t \qquad l[u] \vee D}{(l[t] \vee C \vee D)\theta}\ \theta = \mathsf{mgu}(s,u) \qquad \frac{s \neq t \vee C}{C\theta}\ \theta = \mathsf{mgu}(s,t)$$

END OF EXAMINATION