MASTER IN ARTIFICIAL INTELLIGENCE (UDC - USC - UVigo)
REASONING AND PLANNING exam. June 19th, 2023

Surname: _____

First Name: _____

INSTRUCTIONS This exam covers units 1-6 and is weighted with a maximum of **42 points (pt) from a total of 100 pt** in the whole course (Unit 7 is not covered in the exam and weights 8 pt). For the test, use the original statement sheet and avoid corrections or unclear marking (ask for a new blank sheet if needed). **Completion time = 2 hours**.

— EXAM —

Exercise 1 (20pt). Each question may have $n \geq 1$ correct answers. For each question, the **total score** will be: Checking all correct answers = **5pt**; Checking only correct answers, but not all = **3pt**; Checking an incorrect answer = **-3pt**; Leaving blank = **0pt**. A total negative score in Exercise 1 counts as 0 % in the rest of the exam.

1.1) Mark those formulas below that are tautologies in classical propositional logic:

- ☑ $p \vee q \leftrightarrow p \vee (\neg p \wedge q)$
- ☐ $p \vee q \leftrightarrow p \vee (\neg q \wedge p)$
- ☐ $(p \rightarrow p) \rightarrow p$
- ☐ $\neg(p \rightarrow \neg p)$

1.2) Mark those clauses that "occur in" (that is, can be derived from) the transformation of $p \rightarrow \neg(q \leftrightarrow r)$ into Conjunctive Normal Form (CNF)

- ☑ $\neg p \vee q \vee r$
- ☐ $\neg r \vee p$
- ☐ $\neg p \vee \neg q \vee r$
- ☑ $\neg p \vee \neg q \vee \neg r$

1.3) Given the following logic program `p :- not p, r.` `q :- not p.`

- ☑ the reduct with respect to $\{q\}$ is the program `p :- r.` `q.`
- ☐ the reduct with respect to $\emptyset$ is the program [ ]
- ☑ the reduct with respect to $\{p\}$ is the program [ ]
- ☐ the reduct with respect to $\{p, q\}$ is the program `r.`
- ☐ the reduct with respect to $\{q\}$ is the program `q :- not p.`

1.4) Which of the following pairs form interpretations $\langle H, T\rangle$ that satisfy the formula $\neg p \rightarrow q$ in the logic of Here-and-There:

- ☐ $H = \emptyset$, $T = \emptyset$
- ☐ $H = \{q\}$, $T = \emptyset$
- ☑ $H = \{q\}$, $T = \{q\}$
- ☑ $H = \{p\}$, $T = \{p, q\}$

Explanations for the test

1.1) The only tautology is the first formula since

$$\begin{aligned} p \vee (\neg p \vee q) &\equiv \underbrace{(p \vee \neg p)}_{\top} \vee (p \vee q) \\ &\equiv p \vee q \end{aligned}$$

The second formula is not a tautology: take the interpretation $\{q\}$ making p false and q true. Then $p \vee q$ is true but $p \vee (\neg q \wedge p)$ is false.

The third formula is equivalent to p since $(p \to p) \to p \equiv (\neg p \vee p) \to p \equiv \top \to p \equiv p$ and, thus, it is not a tautology: just take the interpretation $\emptyset$ and it becomes immediately false.

Finally, the fourth formula is again equivalent to p since $\neg(p \to \neg p) \equiv \neg(\neg p \vee \neg p) \equiv \neg\neg p \equiv p$ and we can take the interpretation $\emptyset$ to make it false.

1.2) The transformation of $p \to \neg(q \leftrightarrow r)$ into CNF corresponds to:

$$\begin{aligned} & p \to \neg(q \leftrightarrow r) \\ \equiv\ & \neg p \vee \neg((q \to r) \wedge (r \to q)) \\ \equiv\ & \neg p \vee \neg(q \to r) \vee \neg(r \to q) \\ \equiv\ & \neg p \vee (q \wedge \neg r) \vee (r \wedge \neg q) \\ \equiv\ & (\neg p \vee q \vee r) \wedge (\neg p \vee \underbrace{q \vee \neg q}_{\top}) \wedge (\neg p \vee \underbrace{\neg r \vee r}_{\top}) \wedge (\neg p \vee \neg r \vee \neg q) \\ \equiv\ & (\neg p \vee q \vee r) \wedge (\neg p \vee \neg r \vee \neg q) \end{aligned}$$

Only the first and the fourth formulas are derived from the CNF of the original formula. We cannot derive the other two clauses $\neg r \vee p$ or $\neg p \vee \neg q \vee r$.

1.3) The first answer is correct, since $p \notin \{q\}$ and thus, we just delete all the negative literals `not p` in the program. The third answer is also correct, since $p \in \{p\}$ and the negative literals `not p` do not hold with respect to the assumption, so we end up deleting all rules in the program.

The second answer is incorrect: since $p \notin \emptyset$ we should get the same program as in the first answer.

The fourth answer is incorrect because $p \in \{p, q\}$ and we should get an empty program, as in the second answer.

Finally, the fifth answer is incorrect since the reduct is always a positive program: it cannot contain any negation.

1.4) Models of $\neg p \to q$ must satisfy that if $p \notin T$ (p is not assumed) then $q \in H$ (q must be proved) and so $q \in T$ too by construction. In other words, they must satisfy $p \notin T$ or $q \in H$.

The first interpretation is not a model because $\langle H, T\rangle \models \neg p$ by $\langle H, T\rangle \not\models q$.

The second answer is not even a well-formed HT interpretation since we must always have $H \subseteq T$.

The third answer is a model because $\langle H, T\rangle \models \neg p$ but also $\langle H, T\rangle \models q$.

Finally, the fourth answer is also a model since $\langle H, T\rangle \not\models \neg p$.

Exercise 2 (5pt). A logic program contains an extensional database with facts for two predicates with the following meanings: `teaches(P,C)` = "professor `P` teaches course `C`"; `enrolled(S,C)` = "student `S` is enrolled in course `C`". Write a rule (**without aggregates**) to obtain in `query(S)` the students `S` enrolled in at least one course taught both by professor `enrique` and professor `analia` but in which student `ana` is not enrolled.

```
query(S) :- teaches(enrique,C), teaches(analia,C), enrolled(S,C), not enrolled(ana,C).
```

Exercise 3 (5pt). A logic program is used to compute several answer sets with a predicate `assigned(C,R,N)` meaning that course `C` is assigned classroom `R` for `N` hours. Write a `#maximize` clause to maximize the total number of hours assigned to classroom `25`.

```
#maximize{N,C:assigned(C,25,N)}.
```

Exercise 4 (12pt). A player of straight **poker** receives 5 cards at the beginning of the game. Write an ASP program that generates all possible initial hands (for a single player) with a **poker**, that is, four cards with the same rank x, plus a fifth card with a rank y different from x. We assume there are no jokers. Use predicate `hand(R,S)` meaning that we get a card with rank `R` for suit `S`. For instance `hand(2,diamonds)` means we got the 2 of diamonds.

```
suit(club;diamond;spade;heart).
rank(2..10;jack;queen;king;ace).
5 {hand(R,S):rank(R),suit(S)} 5.

% option 1
%full :- hand(R,_), hand(R',_), R!=R', #count{S:hand(R,S)}=3, #count{S':hand(R',S')}=2.
%:- not full.

% option 2
% :- hand(R,_), #count{S:hand(R,S)}=1. % we cannot get a rank that is not repeated

% option 3 (same but without aggregate)
repeated(R) :- hand(R,S), hand(R,S'), S!=S'.
:- hand(R,_), not repeated(R).

#show hand/2.
```