MASTER IN ARTIFICIAL INTELLIGENCE (UDC - USC - UVigo)
REASONING AND PLANNING exam. January 9th, 2022

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: 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:

- [x] $p \rightarrow (p \rightarrow p)$
- [ ] $(p \rightarrow p) \rightarrow p$
- [ ] $q \wedge \neg p \leftrightarrow \neg(p \rightarrow q)$
- [x] $p \wedge \neg p \rightarrow q$

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

- [x] $\neg p \vee q \vee r$
- [ ] $\neg r \vee q$
- [x] $\neg r \vee p$
- [ ] $p \vee \neg q \vee r$

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

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

1.4) Which of the following interpretations satisfy the formula $p \rightarrow \neg q$ in the logic of Here-and-There:

- [x] $H = \emptyset$, $T = \{p\}$
- [ ] $H = \{p\}$, $T = \emptyset$
- [x] $H = \{q\}$, $T = \{q\}$
- [ ] $H = \{p\}$, $T = \{p, q\}$

Explanations for the test

1.1) The formula $p \rightarrow p$ is equivalent to $\neg p \vee p \equiv \top$. Thus, the first two formulas are equivalent to $p \rightarrow \top$ and $\top \rightarrow p$ respectively. The former corresponds to $p \vee \top \equiv \top$ and is a tautology whereas the latter is false when we take interpretation $I = \emptyset$.

In the third formula, the left part amounts to $\neg(p \rightarrow q) \equiv p \wedge \neg q$. Thus, one possible counterexample is $I = \{q\}$ where $q \wedge \neg p$ is true but $p \wedge \neg q$ is false, so the double implication does not hold.

Finally, $p \wedge \neg p \equiv \bot$ (it is an inconsistency) so the fourth formula amounts to $\bot \rightarrow q \equiv \neg\bot \vee q \equiv \top \vee q \equiv \top$ and is a tautology.

1.2) The reduction to CNF can be done with the following steps:

$$\begin{aligned} & (p \wedge \neg q \leftrightarrow r) \\ \equiv\ & (p \wedge \neg q \rightarrow r) \wedge (r \rightarrow p \wedge \neg q) \\ \equiv\ & (\neg(p \wedge \neg q) \vee r) \wedge (\neg r \vee (p \wedge \neg q)) \\ \equiv\ & (\neg p \vee q \vee r) \wedge (\neg r \vee p) \wedge (\neg r \vee \neg q)) \end{aligned}$$

1.3) The explanation in each case is:

- The correct reduct with respect to $\{q\}$ is `r.`
- A program reduct never contains default negation `not`.
  The correct reduct with respect to $\emptyset$ is `p :- r. r.`
- This one was correct
- See third answer
- This one was correct

1.4) By definition, $\langle H, T\rangle \models p \rightarrow \neg q$ amounts to requiring the two conditions:

(i) If $\langle H, T\rangle \models p$ then $\langle H, T\rangle \models \neg q$

(ii) $T \models p \rightarrow \neg q$ classically

For condition (i) remember that, for negation, we have $\langle H, T\rangle \models \neg q$ amounts to $T \not\models q$ so that we actually get:

(i) If $p \in H$ then $q \notin T$.

For condition (ii), note that in classical logic, $p \rightarrow \neg q$ is the same as $\neg p \vee \neg q$, so we only accept T's where one of the two is false. This means $T = \{p\}$, $T = \{q\}$ or $T = \emptyset$ are total models of the formula and we can use $H = T$ in all these cases. But also, for $T = \{p\}$ and $T = \{q\}$ we can also take the smaller $H = \emptyset$ because (i) is trivially true, since $p \notin H$. To sum up, the HT-models of the formula are five:

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

For the second answer, note that we can never have $H = \{p\}$ and $T = \emptyset$ because $H \subseteq T$. For the fourth answer, note that $p \in H$ and due to (i) this means we should not have $q \in T$.

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 to more than one course taught by professor `enrique`.

```
query(S) :- teaches(enrique,C), teaches(enrique,D), C!=D, enrolled(S,C), enrolled(S,D).
```

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 `#minimize` clause to minimize the total number of hours assigned to classroom `25`.

```
#minimize{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 **full house**, that is, three cards with the same rank x, plus other two cards with the same rank y different from x. 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.
```