MASTER IN ARTIFICIAL INTELLIGENCE (UDC - USC - UVigo)
REASONING AND PLANNING exam. January 10th, 2025

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 has at least one correct answer and its total score depends on whether you check: some incorrect answer = **-3pt**; all the correct answers = **5pt**; only correct answers, but not all = **3pt**; leaving blank = **0pt**. A total negative score in Exercise 1 counts as 0pt in the rest of the exam.

1.1) Mark those formulas below that are equivalent to $\neg(p \leftrightarrow q)$ in classical propositional logic:

- [x] $\neg p \leftrightarrow q$
- [x] $p \leftrightarrow \neg q$
- [ ] $\neg p \leftrightarrow \neg q$
- [ ] $\top$

1.2) Given the positive logic program P with rules `a :- b, c.  c :- a.  d.  e :- a, b.` mark the correct statements about the direct consequences operator T_P.

- [ ] $T_P(\emptyset) = \{\mathtt{a}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$
- [x] $T_P(\{\mathtt{a}, \mathtt{b}\}) = \{\mathtt{c}, \mathtt{d}, \mathtt{e}\}$
- [x] $T_P(\emptyset) = \{\mathtt{d}\}$
- [x] $T_P(\{\mathtt{d}\}) = \{\mathtt{d}\}$

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

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

1.4) The rule `q :- not p, not q.` used above is actually equivalent to the formula $\neg\neg p \vee \neg\neg q$ in the logic of Here-and-There (HT), but the latter is not equivalent to $p \vee q$ in that logic. Mark those HT interpretations that are HT models of $\neg\neg p \vee \neg\neg q$ but not of $p \vee q$.

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

Explanations for the test

1.1) The truth table for $\alpha \leftrightarrow \beta$ assigns true (1) to the formula iff α and β have the same truth value (both are false, or both are true). The negation $\neg(p \leftrightarrow q)$ does the opposite: it is true when p and q have a different truth value (this corresponds to an exclusive or, xor). Let us call $\gamma = \neg(p \leftrightarrow q)$ to the formula in the exercise statement.

Formula $\neg p \leftrightarrow q$ is true when $\neg p$ and q have the same value, but this happens when p and q have different value. Therefore, it is equivalent to γ.

Analogously, formula $p \leftrightarrow \neg q$ is true when p and $\neg q$ have the same value, but again, this happens when p and q have different value. Thus, it is also equivalent to γ.

Formula $\neg p \leftrightarrow \neg q$ is true when $\neg p$ and $\neg q$ have the same value, and this is the same than requiring that p and q have the same value, that is, $p \leftrightarrow q$. Clearly, it is not equivalent to γ: as a counterexample, when $p = 1$ and $q = 1$ we have that $p \leftrightarrow q$ is true but γ is false.

Finally, formula $\top$ is not equivalent to γ. Again, when $p = 1$ and $q = 1$ we have that $\top$ is true (it is always true, indeed) but γ is false. either.

1.2) Operator $T_P(I)$ collects all the heads in rules whose bodies are satisfied by I. As a result, $T_P(\emptyset)$ is always the set of facts in the program, because facts have an empty body (and so, it is always satisfied) but any other body in the program will be false in interpretation $\emptyset$. In our case $T_P(\emptyset) = \{\texttt{d}\}$. Observation: note that the facts of the program, in this case $\{\texttt{d}\}$, will always be included in $T_P(I)$ for any I we choose.

Interpretation $\{\texttt{a}, \texttt{b}\}$ makes true the bodies of all the rules, except the first one. Therefore, $T_P(\{\texttt{a}, \texttt{b}\}) = \{\texttt{c}, \texttt{d}, \texttt{e}\}$ collecting the heads of all rules, except the first one.

Interpretation $\{\texttt{d}\}$ only satisfies the body of the third rule (the fact `d.`). As a result, $T_P(\{\texttt{d}\}) = \{\texttt{d}\}$.

1.3) Answer number 1 is incorrect: the reduct with respect to $\{\texttt{p}, \texttt{q}, \texttt{r}\}$ deletes the first and the third rule, because their negative literals `not p` and `not q` are false in that interpretation. Answer number 3 is also incorrect: the program reduct never contains a negation (all `not`'s are removed). The other three answers are correct.

1.4) The formula $\neg\neg p \vee \neg\neg q$ is proved when $\neg\neg p$ is proved or $\neg\neg q$, but since both are negated formulas, this amounts to checking their truth in T, that is, checking $p \in T$ or $q \in T$. On the other hand, the formula $p \vee q$ is proved when $p \in H$ or $q \in H$.

Answer number 1 is an HT model of $\neg\neg p \vee \neg\neg q$ but is also HT model of $p \vee q$. Answers number 2 and 3 are models of $\neg\neg p \vee \neg\neg q$, because in both cases $p \in T$, and are not a models of $p \vee q$ because $p \notin H = \emptyset$ and $q \notin H = \emptyset$ in both cases. Finally, answer number 4 is not even a well-formed HT interpretation, because H must be a subset of T.

Exercise 2 (10pt). A lottery ticket in Spain consists of 5 digits, covering the interval from 00000 to 99999. Write an ASP program that generates one answer set per each lottery number that contains one or more odd digits, and one or more repeated digits. Use predicate `ticket(N,D)` to represent that the `N`-th digit in the ticket is `D`.

```
% Solution using auxiliary predicates
digit(0..9).
position(1..5).
#show ticket/2.
1 {ticket(N,D):digit(D)} 1 :- position(N).
someodd :- ticket(_,D), D\2=1.
repeated :- ticket(N,D),ticket(M,D),N!=M.
:- not someodd.
:- not repeated.

-----
% Solution using aggregates: replace the last 4 rules by
:- #count{D: ticket(_,D),D\2=1}=0.
:- #count{D: ticket(_,D)}=5.
```

Exercise 3 (4pt). The general *inertia default* can be written in `telingo` as the pair of rules

```
#program dynamic.
h(F,V) :- 'h(F,V), not c(F).
c(F) :- 'h(F,V), h(F,W), V!=W.
```

Provide a brief explanation of their meaning:

The first rule states that fluent F will hold (h) value V at the resulting state if it held value V before (the comma on the left of h means "previous state") and fluent F has not changed (c).

The second rule defines predicate c (changed). Fluent F has changed if it held value V before, but holds a different value W now.

Exercise 4 (4pt). Suppose we want to solve the Towers of Hanoi problem but using 5 pegs (`a,b,c,d,e`) and assume we use action `move(P,Q)` to move the top disk of peg `P` to the top of peg `Q`. Which is the **branching factor** of this planning problem? 10

NOTE: in the exam, no explanation was required. We include it here for didactical purposes.

The branching factor is the maximum number of actions we can open in one "expand" operation, that is, the maximum number of possible actions to be explored in a given planning search node.

In our case, the maximum number of possible actions to execute would occur if we have disks in all the pegs. Then we could move from the peg with the smallest top disk to any of the other 4 pegs. This makes 4 possible movements. The second smallest top disk can be moved to the other pegs except the one that has the smallest disk, so it can be moved to 3 pegs, that is, 3 possible movements more. Similarly, the third smallest top disk has only 2 possible choices. The fourth smallest top disk has only 1 choice (on top of the largest top disk) and, finally, the largest top disk cannot be moved anywhere else. The final amount is $4+3+2+1 = 10$. The general expression with n pegs is $\Sigma_{i=1}^{n-1} i = \frac{n\cdot(n-1)}{2}$

Exercise 5 (4pt). Write a formula in Description Logic (DL) that describes the set of foreign (*Foreign*) students (*Student*) such that all the courses they enrolled in (*enrolled*) are mandatory (*Mandatory*).

$$\mathit{Foreign} \sqcap \mathit{Student} \sqcap \forall \mathit{enrolled}.\mathit{Mandatory}$$