Q & A

3) Within the terminological approach towards the representation of concepts/categories and individuals/instances, the candidate is invited to illustrate the notions of
   - **Disjointness** over a set S of categories (S = {c_1, c_2, ..., c_n}, where $c_1 ... c_n$ are categories)
   - **Exhaustive** Decomposition of a category c into a set S of categories
   - **Partition** of a category c into a set of categories S

   The candidate is invited to illustrate these notions through a simple example

Disjoint(S) $\Leftrightarrow$ { $\forall c_1, c_2 \quad c_1 \in S, c_2 \in S, c_1 \neq c_2 \Rightarrow c_1 \cap c_2 = \emptyset$ } , example $\rightarrow$ Disjoint({animals, vegetables})

Exhaustive(S,c) $\Leftrightarrow$ { $\forall i \; i \in C \Leftrightarrow \exists c_2 \quad c_2 \in S \wedge i \in c_2$ }, example $\rightarrow$ Exhaustive({animals, vegetables}, Elements in Single)

Partition(S,c) $\Leftrightarrow$ Disjoint(S) $\wedge$ Exhaustive(S,c) , example $\rightarrow$ Partition({animals, vegetables}, elements in Single)

2) The candidate is invited to introduce the vanilla meta-interpreter, and to explain its clauses.

```
solve(true):-!.
solve((A,B)):- !, solve(A), solve(B).
solve(A): clause(A,B), solve(B).
```

this meta-interpreter allow us to change "the rule of the game" manipulating the order of clauses execution and insert whatever we want.
the first line is the base case, whatever the value is true (exactly present in KB), this clause is true and disallow backtrack (cut).
the second line allow us to execute the two sub-goals in left-most manner.
the third line take the clause of A and solve it recursively.

3) The candidate is invited to describe the predicates/terminology used in the definition of the Event Calculus Framework.

the event calculus framework allow us to use FOL with the concept of time. It uses the concept of point of time.
It is based on an ontology and two axioms.
the predicates are:

holdsAT(F,T): the fluent F holds at time T.
Happens(E,T): the event E happened at the T.
initiates(E,F,T): the fluent F caused by event E starts at the T.
terminates(E,F,T): the fluent F caused by event E cease at the T.
clipped(T, F, T): the fluent F such false between T0 and T, T0<T
initially(F): Fluent F happed at beginning.

the domain-dependent axioms are:

holdsAT(F,T) $\Rightarrow$ happens(E,T_1) $\wedge$ initiates(E,F,T_1) $\wedge$ $\neg$ clipped(T_1, F, T) $\wedge$ (T_1<T)
holdsAT(F,T) :- initially(F) $\wedge \neg$ clipped(0,F,T)
clipped(T_1,F,T_2) $\Rightarrow$ happens(E,T) $\wedge$ terminates(E,F,T) $\wedge$ T_1<T<T_2

4) The candidate is invited to briefly introduce the notion of Semantic Networks, and to highlight some of the limits that were present in their original formulation.

Humans are used to reason about object and classify them through categories, giving them some properties.
Semantic networks are a struct to represent knowledge in these terms.
They are characterized by ovals (or boxes) connected between labelled links.
The links represent relations among:

- object - object
- category - category
- object - category (a-like)
- object/category - property

Limitation arise due to the simplicity of this language. Despite FOL, here is missing the concept of negation, universal and existential quantifiers, disjunction and nested object function.
Moreover an inconsistent could happen due to the multiple inheritance.

3) The candidate is invited to briefly introduce the three different approaches (presented in the course), to deal with the reasoning with temporal information.

Situation calculus:

- This approach was introduced in 1963 and aims to represents facts in temporal environment. The initial state is called situation, and if "a" is an action and "s" a situation, Result(s,a) is also a situation. Moreover it introduces the fluent concept as "At(x,l,s)".
Finally does also preconditions, defined as $\phi(s) \Rightarrow Poss(a,s)$.
Given that, it uses also the successor state axioms $[F^{T+1} \Leftrightarrow Action\ Causes\ F^T \vee (F^T \wedge \neg Action\ Causes\ NOT\ F^T))$ with a different notation:
$Poss(a,t) \Leftrightarrow (F(Result(s,a)) \Leftrightarrow a = Action\ Causes\ F \vee (F(s) \wedge a \neq Action\ Causes\ Not\ F)$.
It uses the unique action axiom, which allows the execution of only one action in a situation.
So, the goal is defined as a set of fluents and actions are situations.

- Event calculus
Based on the concept of points of time, uses a set of predicates common in an ontology and two distinct axioms.
The predicates are:
Holds(F,T): the fluent holds at time T
Happens(E,T): the event E happened at the T
Initiates(E,F,T): the event E causes F to hold at the T
Terminates(E,F,T): the event E causes F to cease at the T
Initially(F): fluent F holds at the 0
Clipped(T_1,F,T): F has terminated between T_1 and T, $T_1 < T$.
Moreover the axioms are:
  - $Holds\ At(F,T) \Leftarrow Happens(E,T) \wedge Initiates(E,F,T) \wedge T_1 < T \wedge \neg Clipped(T_1,F,T)$.
  - $Holds\ At(F,T) \Leftarrow Initially(F) \wedge \neg Clipped(0,F,T)$.
  - $Clipped(T_1,F,T_2) \Leftrightarrow Happens(E,T) \wedge (T_1 < T < T_2) \wedge Terminates(E,F,T)$.
This approach allows to reason with evolving action systems, but is not safe if fluents contain variables due to the NOT operator in PROLOG.

- Allen's logic

Finally this approach introduces the duration-based representation.
Introducing the concept of interval which Begins and Ends in a certain time point, 13 predicates could be shown such as Meet, Before, After, During, Overlap, Starts, Finishes, Equals.

4) The candidate is invited to briefly introduce the Description Logics, and in particular the principal operators of the ALC fragment (AND operator; ALL operator; [EXISTS 1 r] operator; concept complement (negation)).

Description Logics is a family of languages based on constraints imposed implicitly by humans in reasoning process. Indeed, humans classify objects as belong to categories, style or multiple, reason or breeds. Moreover objects are also composed by parts which interacts among them. DL introduces 3 elements: concepts, roles, constraints and logic operators like partiality, connectors and numbers.

ALC is a type of DL and is based upon 4 fundamental operators:

AND: a AND B, represents the set of individuals which belong to both concept A and B, Also called intersection, E.G: STUDENT and TEACHERS represent all the individuals being both of them, like tutors.

ALL: ∀R.C, represents all the individuals r-related to concept C. ∀ enrolled Student is the set of individuals which are following a random type of course.

EXISTS 1 r: ∃R.C, says that there is at least one individual r-related to concept C. E.G: ∃.work for.company: there is at least one individual which works for a random company.

NOT: complement all the individuals which not belong to a class: NOT Student are all the professors for example.

2) The candidate is invited to shortly describe the RETE algorithm.

The RETE Algorithm, was developed in 70's as a way to optimize the performance of a rule-based expert system. A naive implementation of expert system compare each fact with each rule creating an high latency in high scalability problem.

This algorithm works as follows:

- Rules are typically represented as a set of conditions and actions. Each condition specifies a pattern to match against state.
- A network of nodes representing patterns is created, so that the information could be efficiently propagated between conditions.
- Alpha network (intra-elements feature) contain all the single queries to match with the fact
- Beta network (inter-elements feature) contain all the queries possibly matching multiple facts, build upon the results of alpha-net.
- The Beta results pass through a conflict resolution mechanism to establish which rules take precedence.
- Finally the rule is executed, by default in a cycle avoiding conflicts.

1) The candidate is invited to present the notion of the cut "!" operator in the Prolog language. Moreover, the candidate should illustrate its usage through a very short example.

Prolog is a reasoning logic language, it's based on backtracking and SDL algorithm. Infact, it starts to prove the query from the available KB. It attempts each way and backtrack the decision tree when reaches the failure. The cut operator it's used to avoid the backtracking after a satisfied instruction avoiding to search for other possible solution and path. It's useful e.g. to represent the if else statement.

```
even (N, Result):- K is N mod 2, K =:= 0, !, Result = True.
even (_, Result):- Result = false.
```

2) The candidate is invited to briefly introduce the Knowledge Graph paradigms, and which are the main differences w.r.t. the Semantic Web proposal.

In the early start of Web, this was seen as a pool of information where the network was accessible to everyone. But, nonetheless all this information, there were no way to automatize the reasoning process of a computer through them. Infact, was missing a common structure with descriptive information between linked page.

A first proposal was the semantic web approach: each web relation was described through a triplette: subject, relation, palette. Moreover an ontology (owl) born to define the relationships and categories of information and a query language (SPARQL) to retrieve information. Knowledge graph are an evolution of this approach, avoiding the triplette constraint and be more flexible about relations. It has a less organized structure allowing to represent the relations as in the real world. This approach is used by Google to retrieve answer to the user's question avoiding to lose him after a search click, increasing also the profilate targets.

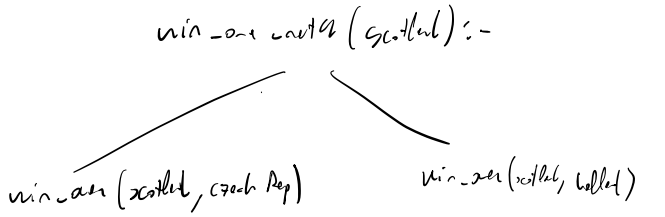
1) Given the following LPAD program:

```
win_one_match(Team) :-
      win_over(Team, _).

win_over(scotland, czechRep):0.3.
win_over(scotland, holland):0.2.
```

Which is the probability of the query `win_one_match(scotland)` ?

The candidate should illustrate the Distribution Semantics by showing the "worlds", the probability of each world, and the formula used to compute the overall probability.



W1 wo_czech_up T
wo_holland T

W1 wo_czech_up F
wo_holland T

W1 wo_czech_up T
wo_holland F

W1 wo_czech_up F
wo_holland F

$= 0.3 \cdot 0.2 + 0.7 \cdot 0.2 + 0.3 \cdot 0.8 =$

$= 0.44$

12:27 Mar 9 gen

Chesani-20210628.FundamentalsAI.testA

Apri in Libri Fine

Exam A

1) The candidate is invited to write a prolog predicate **count/2** that, given as input a term representing a predicate, it will return the number of solutions for the goal of proving that input predicate. For simplicity, it will be assumed that the proof procedure of the predicate passed as parameter will always terminate in a finite time.
For example:

```
p(a).
p(b).
p(c).
```

and the query:

```
count(p(X), Result).
```

The expected answer is:

```
Yes, Result = 3.
```

This is because there are three different ways for proving the goal **p(X)** .

---

count(Goal, Result) :- findall(Goal, Goal, Solution), length(Solution, Result).

1) The candidate is invited to describe the (operational) semantics of the predicates **setof**, **bagof**, and **findall**. Moreover, the description should be illustrated by some short examples.

Setof, bagof and findall are three build in second order formula in prolog. The first two create a list of object satisfy: $\forall x \in S, p(x)$ is true. where S is the list, p(x) is a random predicate present in KB. The difference between them is the process of unification. When the predicate has two parameters, for example, the query was satisfied like: $\forall x \in S, \forall y, p(x,y)$ is true corresponding to setof(x, p(x,y), S), not saving the list of x where y is existentially quantified. In order to do this, we have to write setof(x, y^p(x,y), S) or findall(x, p(x,y), S). In this way we are grouping by a random value of y: $\forall x \in S, \exists y, p(x,y)$ is true.

1) The candidate is invited to describe how negation is tackled in Prolog, what is NAF, what is SLDNF, and the issues related with NAF over terms containing unbounded variables.

Due to the CWA assumption in prolog and SLD algorithm, the language doesn't allow to derive negative information.

To overcome this problem, prolog adopt Negation as Failure rule.

Given a program P, we name FF(P) the set of atoms for which the proof fails in a finite tree

NF(P) = {¬A | A ∈ FF(P)}

If an atom A belongs to FF(P), is not logical consequence of P.

To solve goals containing also negative atoms, SLDNF was proposed.

- Given a list of goals $L_1 \ldots L_n$:
  - Do not select any negative literal if is not ground.
  - If the literal atom is positive, apply normal SLD.
  - If L_i is ~A (A ground) and A fails in finite tree, L is proved.
- The problem with unbound variables are that it could lead to a given degree of uncertainty, considering that is not possible to prove either truth or falseness.

2) The candidate is invited to describe the distribution semantics adopted in LPAD, using also a short program to illustrate such semantics.

LPAD is a prolog library which allow to reason upon uncertainty and probability.

Given the code:

```
sneezing(X):0.7 :- flu(X).
sneezing(X):0.8 :- has_fever(X).
```

The probability is calculated summing all the possible worlds where one of the statement is true.

We are saying there that the probability to sneeze if X has flu is 0.7, while if it has fever is 0.8. So, the probability to sneeze if it has not flu is 0.3 and not fever 0.2. The distributed world are where every one is true and possible

so the probability is calculated like conditional probability. A query like (sneezing(bob) | flu(bob) ∨ has_fever(bob)) =

$0.7 \cdot 0.8 + 0.7 \cdot 0.2 + 0.3 \cdot 0.8 = 0.94$

2) The candidate is invited to Forward reasoning in rule-based systems, and to highline the difference w.r.t. backward reasoning.

forward reasoning is a way to infer a fact given the initial data. During the process, it can create new solutions in order to reach the goal. In contrast backward reasoning starts from the goal and looks for existing data to prove it, divide the given problem in sub problem. A forward reasoning rule-based system is the RETE Algorithm.

3) The candidate is invited to introduce the notions of close world assumption and open world assumption, and to briefly discuss how Prolog and Description Logics deal with these aspects.

CWA and OWA deals with the degree of truthess in KB.
In CWA, the KB is considered to be complete, so when a fact is missing, is considered to be false. Prolog assumes this assumption, whenever it encounters a fact not present in KB, it considers be false and backtracks. It could be possible through a mutual interaction to ask to the user to expand the KB.
On the other hand, OWA considers what is not in KB as unknown.
Description Logics deals with this assumption, so whenever a fact cannot be inferred, it doesn't make false the entire logic.