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

- **HoldsAt**(F, T): The fluent F holds at time T
- **Happens**(E, T): event E (i.e., the fact that an action has been executed) happened at time T
- **Initiates**(E, F, T): event E causes fluent F to hold at time T (used in domain-dependent axioms...)
- **Terminates**(E, F, T): event E causes fluent F to cease to hold at time T (used in domain-dependent axioms...)
- **Clipped**(T_1, F, T): Fluent F has been made false between T_1 and T(used in domain-independent axioms), $T_1 < T$
- **Initially**(F) : fluent F holds at time 0

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

- Categories relate each other because of subclass relation.

Disjoint(S) $\Leftrightarrow \forall c1, c2 \quad c1 \in S$ and $c2 \in S$ and $c1 \neq c2 \Rightarrow c1 \cap c2 = \varnothing$

E.g: Disjoint({Animals, Vegetables}).

- Subcategories might cover all the possible instances of the parent category.

ExhastiveDecomposition(S,c) $\Leftrightarrow (\forall i \;\; i \in c \Leftrightarrow \exists c2 \;\; c2 \in S$ and $i \in c2)$

E.g.: ExhaustiveDecomposition({Student, Professor}, PeopleInThisRoom).

- If a category can be decomposed in more categories such that:
  - they form an exhaustive decomposition
  - they are disjoint

Then we have partition.

Partition(S,c) ⇔ Disjoint(s) and ExhastiveDecomposition(S,c)

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

- Define a predicate **solve(goal)** that answers true if Goal can be proved using the clauses of the current program.
- It does not deal with pre-defined predicates.
- No need to "call" any predicate. The vanilla meta-interpreter explores the current program, searching for the clauses, until it can prove the goal, or it fails.

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

Define a Prolog interpreter **solve(Goal, Step)** that:

• It is true if **Goal** can be proved

• In case **Goal** is proved, **Step** is the number of resolution steps used to prove the goal

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.

Semantic networks are a graphical representation of knowledge that organizes information in the form of boxes or ovals which are the objects or categories and ars which are connected between boxes that show the realtion between categories. There are some limits in Semantic Networks such as:

- Negation
- Universal and existential quantified properties
- Disjunctions
- Nested function symbols

5. The candidate is invited to define a meta-interpreter for the Prolog language, where the selection of the subgoal in the current resolvent is right-most rather than left-most (as it is usually).

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

In this case, first the solve(B) wants to expand but there is not any clause solve(B) so after that the clause solve(A) can expand and this situation can make the right-most.

6. 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: Assumes everything not known to be true is false. The world is considered closed and finite.

OWA: Assumes the knowledge base is incomplete; absence of information does not imply falsity. The world is open, allowing for unknown truths.

Prolog: Traditionally follows CWA, where absent facts are considered false. Can simulate OWA with dynamic knowledge updates.

DLs: Often connect to CWA in classical logic. Some extensions and modern DLs incorporate OWA principles for handling incomplete or evolving knowledge.

7. The candidate is invited to briefly introduce the ALC Description Logics, mentioning the operators that are supported (negation, AND, ALL, EXISTS), and their meaning (possibly with a short example for each operator).

Negation: Represents the absence or negation of a concept. E.g: ¬patricide(thersandros) → It means that thersandors is not patricide

[AND d1…dn]: Each individual is a member of all the categories d1 . . . dn.

[ALL r d]: Stands for those individuals that are r-related only to individuals of class d. E.g: [ALL :HasChild Male] → All the individuals that have zero more children, but all males.

[EXISTS n r]: It stands for the class of individuals in the domain that are r-related to at least n other individuals. E.g: [EXISTS 1 :Child] All the individuals (the class of the individuals) that have at least one child;

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

Propositional Logics:

Event Calculus: Based on points of time. Reifies both fluents and events into terms. Fluents are properties whose truthness value changes over time.

Allen's Logic: Proposed to reason about intervals, rather than point in times. An interval i begins at a certain time point: we introduce the function Begin(i) that return that timepoint. An interval i ends at a certain time point: Ends(i)

9. The candidate is invited to shortly describe the RETE algorithm.

Rete is a Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Focuses on the step 1: "Match". The match step consists on computing which are the rules whose LHS is matched. LHS is usually expressed as a conjunction of patterns. It is a "many patterns" vs "many objects" pattern match problem. The Rete Algorithm avoid iteration over facts by storing, for each pattern, which are the facts that match it and focuses on determining the conflict set. When a fact is added…, removed, and modified... . It has at least two types of patterns: testing Intra elements which are compiled into alpha network and storing in alpha memory and testing Inter elemenets which are compiled in beta network and store in beta memory.

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

Knowledge Graphs are very large semantic nets that integrate various and heterogeneous information sources to represent knowledge about certain domains of discourse. Just store the information in terms of nodes and arcs connecting the nodes.

- No conceptual schema: data from various sources, with different semantics can be mixed freely
- No reasoning
- Graph algorithms used to fast traverse the graph, looking for the solution