Práctica 6: A Simple Planner in Prolog

In this práctica, the task is to implement a simple planner in Prolog. In artificial intelligence, a planner is an algorithm that computes a sequence of actions for moving a system from an initial state to a goal state. The planner we will consider performs standard depth-first search from the initial state until it encounters a state that satisfies the goal state.

Part 1. To implement the planner we will need a few new operations on lists. Recall that in Práctica 1 we implemented the following predicates:

\[
\begin{align*}
\text{miembro} & (H, [H|\_]) . \\
\text{miembro} & (X, [\_|T]) :- \text{miembro}(X, T).
\end{align*}
\]

\[
\begin{align*}
\text{quita} & (X, [X|L], L). \\
\text{quita} & (X, [H|T], [H|U]) :- \text{quita}(X, T, U).
\end{align*}
\]

\[
\begin{align*}
\text{concatena} & ([], Y, Y). \\
\text{concatena} & ([H|T], U, [H|V]):- \text{concatena}(T, U, V).
\end{align*}
\]

Your task is to implement four additional predicates for lists:

- \text{miembrotodos}(L1, L2): returns true if all the elements of L1 are also members of L2.
- \text{quitatodos}(L1, L2, L3): returns true if L3 is the result of removing all elements in L1 from L2.
- \text{iguales}(L1, L2): returns true if the lists L1 and L2 contain the same elements, although not necessarily in the same order.
- \text{miembrolista}(L1, L2): returns true if the list L2 contains a list that equals L1 according to iguales.
Part 2. A typical benchmark domain for planning is the Blocksworld, in which a gripper has to manipulate blocks on a table. An example of a Blocksworld problem is the following:

We represent blocks using the predicate block, and the gripper using the predicate hand:

\[
\begin{align*}
&\text{block}(a). \\
&\text{block}(b). \\
&\text{block}(c). \\
&\text{block}(d). \\
&\text{block}(e). \\
&\text{block}(f). \\
&\text{hand}(h).
\end{align*}
\]

We represent the current situation using predicates ontable, clear, on, held, and empty. There are four available operators of the problem:

1. Pick up a block from the table, if the block is currently on the table, the block is clear, and the gripper is empty.

2. Put down a block on the table, if the block is currently held.

3. Unstack a block from another block, if the block is currently on that block, the block is clear, and the gripper is empty.

4. Stack a block on top of another block, if the block is held and the other block is clear.
In planning, operators are represented using pre-conditions, add effects, and delete effects. We can represent the four operators of Blocksworld as follows:

\[
\text{op (} \text{pickup (} B, H ) \text{,} \\
\text{[ontable (} B \text{) , clear (} B \text{), empty (} H \text{)] ,} \\
\text{[held (} B \text{)] ,} \\
\text{[ontable (} B \text{) , clear (} B \text{), empty (} H \text{)]})} \\
\text{:} :- \text{ block (} B \text{) , hand (} H \text{).}
\]

\[
\text{op (} \text{putdown (} B, H \text{) ,} \\
\text{[held (} B \text{)] ,} \\
\text{[ontable (} B \text{), clear (} B \text{), empty (} H \text{)] ,} \\
\text{[held (} B \text{)]})} \\
\text{:} :- \text{ block (} B \text{) , hand (} H \text{).}
\]

\[
\text{op (} \text{unstack (} B, C, H \text{) ,} \\
\text{[on (} B, C \text{) , clear (} B \text{), empty (} H \text{)] ,} \\
\text{[clear (} C \text{), held (} B \text{)] ,} \\
\text{[on (} B, C \text{) , clear (} B \text{), empty (} H \text{)]})} \\
\text{:} :- \text{ block (} B \text{) , block (} C \text{), hand (} H \text{).}
\]

\[
\text{op (} \text{stack (} B, C, H \text{) ,} \\
\text{[clear (} C \text{), held (} B \text{)] ,} \\
\text{[on (} B, C \text{) , clear (} B \text{), empty (} H \text{)] ,} \\
\text{[clear (} C \text{), held (} B \text{)]})} \\
\text{:} :- \text{ block (} B \text{) , block (} C \text{), hand (} H \text{).}
\]

The first parameter is the name of the operator, which includes the objects it manipulates (for example \text{pickup (a, h)}). The second parameter is the pre-condition of the operator, the third parameter is the add effect, and the fourth parameter is the delete effect, all represented as lists.

We represent the current state and the goal state as lists of predicates. For example, the initial state of the example Blocksworld problem is given by

\[
\text{[ontable (} a \text{), on (} c, a \text{), clear (} c \text{), ontable (} b \text{), clear (} b \text{), empty (} h \text{)]},
\]

and the goal state is given by \text{[on (} a, b \text{), on (} b, c \text{)]}.  

3
Your task is to define a predicate \( \text{plan}(S, G, L, M) \) which performs the search for a plan, given a current state \( S \), a goal state \( G \), and a list \( M \) containing all previously visited states. The list \( L \) should contain all operators needed to reach the goal \( G \) from \( S \).

If all elements of the goal state are part of the current state, the goal has been reached, so the empty list contains all operators needed to reach the goal state. Otherwise, the search proceeds by selecting an operator that is applicable in the current state (i.e., all elements of its pre-condition are part of the current state), deleting and adding elements according to the delete and add effects of the operator, checking whether the resulting state is part of the list \( M \) and, in case it is not, recursively plans from the resulting state.

Two example tests follow:

?− \( \text{plan}([\text{on table}(a), \text{clear}(a), \text{on table}(b), \text{clear}(b), \text{empty}(h)],[\text{on}(a,b)],L,[]) \).

\[ L = [\text{pickup}(a, h), \text{stack}(a, b, h)] \]

?− \( \text{plan}([\text{on table}(a), \text{on}(c,a), \text{clear}(c), \text{on table}(b), \text{clear}(b), \text{empty}(h)],[\text{on}(a,b), \text{on}(b,c)],L,[]) \).

\[ L = [\text{unstack}(c, a, h), \text{putdown}(c, h), \text{pickup}(b, h), \text{stack}(b, c, h), \text{pickup}(a, h), \text{stack}(a, b, h)] \]