CSE341 – Programming Languages (Fall 2024)

Homework #4

Hand-in Policy: Source code should be handed in via Teams. No late submission will be accepted.

Collaboration Policy: No collaboration is permitted. Any cheating (copying someone else's work in any form) will result in a grade of -100 for the first offense and -200 for the subsequent ones.

Grading: Each project will be graded on the scale of 100.

PROLOG in Common LISP: You will implement a simplified version of Prolog theorem prover. Recall that Prolog uses Horne clauses (a simplified version of first order predicate calculus) to define facts. A query is proven using resolution and unification. The search space is traversed using a depth-first manner with sub queries are proven from left to right and axioms are searched from top to bottom.

You will implement a single function called `prolog_ prove` in Common LISP. This function will take as input two arguments:

1. A list describing the axioms in the form `(axiom_1 axiom_2 … axiom_n)`. The list will have at least one axiom or more (assume that list is not empty).

   Each axiom (Horn clause) will follow the following format: Given
   - Variables are capital lettered words (strings).
   - Objects are small cap words.
   - Predicates are represented by a list with the first entry being the name of the predicate (small cap words) and the remaining entries as the arguments. We assume that there is at least one argument.

   Each axiom is either:
   - a list including a single predicate, e.g., `( ("father" "jim" "jill") )`, or
   - a list including a predicate for the head, a "`<`" for implication and at least one predicate listing the conditions. For example `( ("parent" "X" "Y") "<" ("mother" "X" "Y") )`.

2. A query is a list of axioms including no head body, e.g., `( ("ancestor" "X" "jill") )`.
   - We assume that we have one or zero variables in each query.

Your function will return a list as answer. If the query returns false, the list will be empty/nil. Otherwise, it will have a list of entries of the form `((variable value) …)` where each entry makes the query true.

For example:

```
(let (
        (axioms '(
                    ( ("father" "jim" "jill") )
                    ( ("mother" "mary" "jill") )
                    ( ("father" "samm" "jim") )
                    ( ("ancestor" "X" "Y") "<" ("parent" "X" "Y") )
                    ( ("ancestor" "X" "Y") "<" ("ancestor" "X" "Z") ("ancestor" "Z" "Y") )
                    ( ("parent" "X" "Y") "<" ("mother" "X" "Y") )
                    ( ("parent" "X" "Y") "<" ("father" "X" "Y") ) ) )
        (query1 '( ("ancestor" "X" "jill") ) )
        (query2 '( ("ancestor" "X" "jill") ("mother" "X" "bob") ) ) )
    (prolog_prove axioms query1)
)
```

will return `("X" "parent")`

whereas, if we change the last line to `(prolog_prove axioms query2)` will return `nil`.

Handin your code in a single Lisp file named yourstudentnumber_lastname_firstname_hw4.lisp. This file should define `prolog_ prove` function as described above.