Qualifying Examination – Fall 1998
Logic Programming

Prolog is frequently praised for its ability to write meta-interpreters, i.e., extended Prolog interpreters written in Prolog itself. The simplest example is represented by the traditional *Vanilla meta-interpreter* which simply duplicates (pure) Prolog's behavior

```
interpret(true).
interpret((Goal1,Goal2)) :-
          interpret(Goal1),
          interpret(Goal2).
interpret(Goal) :-
          clause(Goal,Body),
          interpret(Body).
```

1. *[40 pts]* Extend the Vanilla meta-interpreter to produce a *tracer*, capable of generating a detailed trace of an execution. The tracer should display for each goal executed:

   - the current goal to be executed;
   - the trace of the subgoals which are solved—the trace of the subgoals should be indented with respect to the trace of the parent goal;
   - the successful or failing termination of a goal;
   - in case of backtracking over a previously satisfied goal, a message stating which goal is about to be redone.

   The possible structure of the output for the goal

   `?- member(X,[a,b]),member(X,[b,c]).`

   is shown in the following example

```
CALL: member(X,[a,b])
EXIT: member(a,[a,b])
CALL: member(a,[b,c])
    CALL: member(a,[c])
        CALL: member(a,[])
        FAIL: member(a,[])
    FAIL: member(a,[c])
FAIL: member(a,[b,c])
REDO: member(a,[a,c])
    CALL: member(X, [b])
    EXIT: member(b, [b])
EXIT: member(b,[a,c])
CALL: member(b,[b,c])
```

```
EXIT: member(b,[b,c])
```

2. *[40 pts]* Extend the Vanilla meta-interpreter to build a complete representation of the SLD resolution tree (also known as the search tree) for each goal. You can assume:

   - the goals considered generate only finite resolution trees;
   - the programs are propositional—i.e., they do not contain arguments—and pure—i.e., they do not contain built-ins.

   For example, for a program like

```
p_a.
p_b.
p :- p_a.
p :- p_b.
r_c_a.
r_d_e.
r:- r_c_a.
r:- r_d_e,p_e.
q :- p.
q :- r,p.
```

   the call to the meta-interpreter `interpret(q,T)` should instantiate `T` to a Prolog term which represents the tree depicted in figure 1. In your solution you are allowed to use any Prolog's

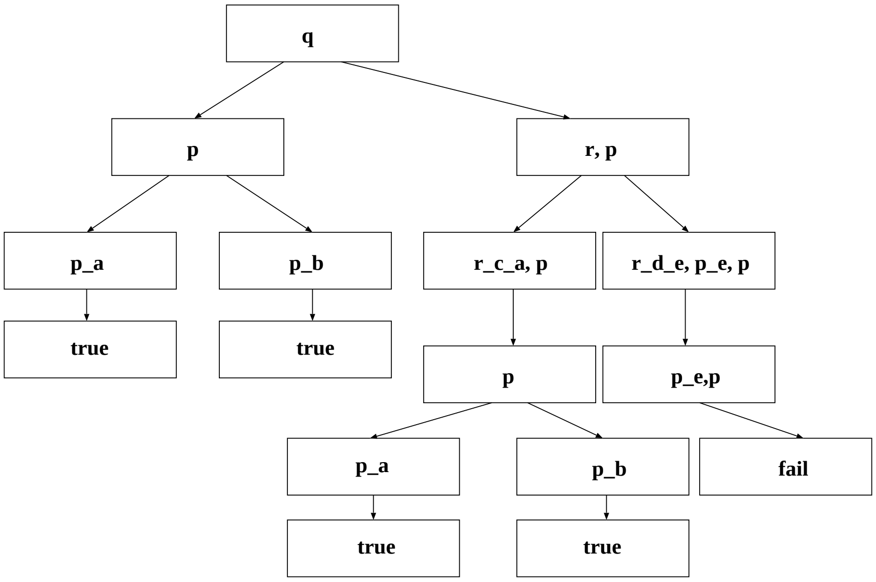

Figure 1: Resolution Tree

built-in predicate (e.g., assert, retract, findall, freeze, etc.). You are allowed to use any reasonable term representation of the resolution tree.

3. *[20 pts]* discuss how you could use a meta-interpreter to support a *fair* execution strategy for logic programming—i.e., a strategy which avoids entering non-terminating computations and missing existing solutions. Remember that the execution strategy used by Prolog (i.e., explore the resolution tree depth-first, left-to-right) is not fair.