CMSC 471/671 **Artificial Intelligence** Fall 2000
Section 0101 TuTh 5:30 - 6:45pm MP103

Project 3

Due December 12, 2000

This assignment involves implementation of part of an automated theorem prover to perform one step of resolution on a given pair of clauses. More specifically, your program should be able to do the following things:

1. **Literal Identification**: For the two given clauses, identify a pair of opposite literals P(arg-list-1) and ~P(arg-list-2), one from each clause, provided such a pair exists.
2. **Unification:** Try to unify the argument lists of two opposite literals P(arg-list-1) and ~P(arg-list-2) obtained from literal identification. Obtain their most general unifier (mgu) if they are unifiable.
3. **Resolution**: Generate the resolvent of the two clauses over P(arg-list-1) and ~P(arg-list-2), if the mgu of P(arg-list-1) and P(arg-list-2) can be found. This includes the application of the mgu to all literals of these two clauses.

Input: The input data is given below where each line contains a pair of clauses to be resolved. The clauses and predicates are represented in list notation.

1. ((p x) (q (f x))), ((not (p a)) (r y)).
2. ((p a) (not (q x))), ((not (p b)) (q (f b))).
3. ((not (p a)) (q (f x))), ((p (f y)) (not (q (g y)))).
4. ((q x a b) (p (f x a) (g x b))), ((not (p y (g y b))) (r y)).
5. ((p x (g x) (h b))), ((not (p (f u a) v u))).
6. ((poor x) (not (smart x)) (happy x)), ((not (happy z)) (exciting z)).
7. ((poor x) (not (smart x)) (happy x)), ((not (read y)) (smart y)).
8. ((not (read y)) (smart y)), ((not (happy z)) (exciting z)).

Here, a and b are constants, x, y, z, u, and v are variables. (Your program should be able to identify predicate symbols and function symbols.)

Output: If two clauses can be resolved, then output the resolvent, its parents, and the mgu, also in list notation. Otherwise, indicate why they cannot be resolved (due to the failure of unification or no pair of opposite literals can be found).

Language: Lisp is strongly recommended, though other languages are not absolutely forbidden.

Report: besides your source code and running output, write a summary of the project (problem, methods, results over input data, etc.)

Additional assignment for 25 points of extra credit

1. Extend your program for resolution to implement a resolution refutation proof procedure using the following (depth-first) strategy:

   1) At step 1: one of the two parent clauses is the negation of the goal;

   2) At step I > 1: the first of the two parent clauses is the resolvent generated from step I - 1, the second is the first sentence in the axiom set that can be resolved with the first clause (always scan axioms from A1 to A7 in that order).

   3) Return "yes" if a null clause is derived; "no" if no more resolution can be performed but a null clause has not been derived.

2. Convert the following axioms into clause form manually, and put them in list notation as input data.

   A1: Ancestor(x1, y1) <= Parent(x1, y1)

   A2: Ancestor(x2, y2) <= Ancestor(x2, z2), Parent(z2, y2)

   A3: Relative(x3, y3) <= Ancestor(z3, x3), Ancestor(z3, y3)

   A4: Parent(Ed, Fred)

   A5: Parent(Eve, Ed)

   A6: Parent(Eve, Linda)

   A7: Parent(Adam, Ed)

   where x1, y1, x2, y2, z2, x3, y3, z3 are variables, Adam, Eve, Ed, Linda, Fred are constants, and Parent, Ancestor and Relative are predicate symbols

3. Use your program for resolution refutation to prove that
   Relative(Ed, Linda)
   is a theorem of the above axioms (returning ``yes").

4. Find an example sentence which is a theorem of the above axioms but cannot be derived by the strategy given in 1. Run this sentence with your program (returning ``no").

5. After each step of resolution in 3 and 4, print the resolvent, its parents, and the mgu in list notation.