CSE 472/572, Spring 2002

Programming Project 3
AUTOMATED THEOREM PROVING

Last Update: 5 April 2002   *********** NEW ************   **material is highlighted**

In this project, you will write programs that could have passed the CS department's old graduate-level AI Qualifying Exam questions on logic :-)

1. **(WARNING: THIS PART IS RELATIVELY EASY.)**

   (a) Given our algorithm for converting a sentence of first-order logic into clause form (handed out in lecture, and on the Web in PDF format at http://www.cse.buffalo.edu/~rapaport/572/S02/clauseform.pdf), write an algorithm (ideally, a Lisp function) that takes a sentence of first-order logic as input and that returns an equivalent sentence in clause form.

   Suggestion: When you rename variables so that variables bound by different quantifiers have unique names, you can use rewrite rules of the following form:

   $$(Q_1 v_1 F(v_1^*) \# Q_2 v_1 G(v_1^*)) \rightarrow (Q_1 v_1 F(v_1^*) \# Q_2 v_2 G(v_2^*))$$

   where the Q_i are quantifiers (either the same or different), # is either $\vee$ or $\wedge$, the v_i are variables such that 'v_1' $\neq$ 'v_2', and '$F(v_1^*)$' represents a sentence containing 0 or more occurrences of 'v_1'. An example would be:

   $$(\forall x P(a,x) \wedge \exists x R(a)) \rightarrow (\forall x P(a,x) \wedge \exists y R(a))$$

   (b) Apply your algorithm to the following sentence:

   $$\forall x[Animal(x) \Rightarrow (Predator(x) \Leftrightarrow \exists y[Animal(y) \wedge Eats(x,y)])]$$

2. **(WARNING: THIS PART IS RELATIVELY HARD.)**

   (a) Implement a unification algorithm (either the one in the text, the one (to be) given in lecture, or—if you did it correctly—your pattern-matcher from Project 1 (perhaps suitably modified)). More precisely, your algorithm should take a pair of sentences as input and either return their MGU if they are unifiable or else return a message such as "NOT UNIFIABLE". You may assume that the notation $f(x,g(x))$ can be understood as: `(f x (g x))`, if you prefer using Lispish notation.

   (b) Use your algorithm to answer this question. For each of the following pairs of terms, if they unify, show a most general unifier (mgu); if they don't, say so, and state why. Assume that *u*, *v*, *x*, *y*, and *z* are variables, and that *a*, *b*, and *c* are individual constants.

      i. $P(a,x,c)$ and $P(y,b,z)$
      ii. $P(a,x,c)$ and $P(y,b,y)$
      iii. $P(x,x,c)$ and $P(u,v,u)$
      iv. $P(x, f(x), f(y))$ and $P(f(a), f(z), z)$
      v. $P(x, f(x), f(a))$ and $P(f(z), f(z), z)$

3. (a) **THE COMPUTATIONAL IMPLEMENTATION OF THIS PART IS OPTIONAL (WARNING: THE COMPUTATIONAL IMPLEMENTATION OF THIS PART IS RELATIVELY HARD.)**
Write a resolution + unification + refutation theorem prover for *first-order predicate* logic.

(b) **DO THIS PART (AT LEAST BY HAND) WHETHER OR NOT YOU DO PART 3a.**
Using resolution, show that the following set of clauses is inconsistent. Assume that a, b, and c are individual constants, and that x and y are variables.

i. $\{On(a,b)\}$
ii. $\{On(b,c)\}$
iii. $\{Red(a)\}$
iv. $\{Green(c)\}$
v. $\{Red(b) Green(b)\}$
vi. $\{\neg Red(x) \neg Green(y) \neg On(x,y)\}$

NOTE: Please do *all* exercises at least *by hand* (in addition to, or instead of, implementing them in a programming language) as part of your *report*. I will hand out a tentative grading scheme to make it easier for you to organize your final report.

********** NEW **********

DUE AT START OF LECTURE, *FRIDAY, APRIL 19***.**

********** NEW **********