Automated Reasoning 2012 (KB)
PROBLEMS 3
Semantic Trees, Completeness of Resolution, Factoring and Subsumption

QUICKIES:

i). Does P(f(y), y) subsume P(f(a), x)?

ii). Find all factors of P(x,x) ∨ P(a,y) ∨ Q(y,b) ∨ Q(x,z). Are any of them safe factors?

iii) Find a closed semantic tree and then find a ground refutation, and finally a general refutation for the clauses

$$G(a,b),\quad F(x) \vee \neg G(x,y),\quad \neg H(v) \vee \neg F(v),\quad \neg F(u) \vee G(u,u),\quad H(a) \vee \neg G(a,z).$$

LONGER QUESTIONS:

1.(i) *For each set of clauses in (a) and (b) below enumerate the Herbrand Universe and the Herbrand Base*. (Actually you did this in Problems 2!) Assume that the signature in each case consists of the names occurring in the given clauses. ie in (a) the predicates are {H,G,F} and the constants are {a,b}, and in (b) the predicate is P, function symbol is f and constant is a.

(a) $\neg H(a)$, $\neg Gz \vee \neg Fb$, $\neg F(y) \vee \neg H(b)$, $Gu \vee \neg Fu$, $Fx \vee Hx$

(b) $P(y,a) \vee P(f(y),y)$, $P(a,y) \vee P(y,f(y))$, $\neg P(x,y) \vee \neg P(y,x)$

(You can only enumerate part of the Herbrand Universe and Herbrand Base in (b).)

(ii) *For each set of clauses in (a) and (b) find a closed semantic tree and use it to give a set of unsatisfiable ground instances of the given clauses*.

(iii) *For each tree found in (ii) derive (from the tree) a ground resolution refutation of the unsatisfiable instances found in (ii).*

(iv) *Use the ground refutations found in (iii) to construct a (proper) resolution refutation.*

2. *Illustrate the lifting property of resolution by replacing the* ***??*** *in the following example*:

$R(x,a) \vee Q(g(x),x)$, $P(y,z) \vee \neg Q(g(y),f(z))$ ⇒⇒⇒⇒ resolve ⇒⇒⇒⇒ **??**	
⇓ take ground	take ground ⇓
⇓ instances	instance ⇓
?? and ?? ⇒⇒⇒⇒ resolve ground instances ⇒⇒⇒ ground resolvent-**??**	

3. Let S be a set of clauses including C and D, where C subsumes D.
Show that $S-\{D\} \vdash_{Res} []$ if and only if $S \vdash_{Res} []$.
That is, show $S-\{D\} \vdash_{Res} [] \rightarrow S \vdash_{Res} []$ and $S \vdash_{Res} [] \rightarrow S-\{D\} \vdash_{Res} []$.

Hint: You can use soundness and/or completeness of resolution to relate $S-\{D\} \vdash_{Res} []$ to $S-\{D\} \models_H \perp$ (ie that S-{D} has no Herbrand models).

PTO

14/10/12

4. *Devise an algorithm to check for θ–subsumption of clauses.*
i.e. to check whether the property $C\theta \subseteq D$ holds for two clauses C, D.
Note that θ could be a *renaming substitution* in which the substitution simply replaces one variable with another without constraint. e.g. if C=P(x,y) and D=P(u,v), then $C\theta \subseteq D$ and $D\theta' \subseteq C$, where θ is {x==u, y==v} and θ' is {u==x, v==y}. On the other hand, if D=P(u,u) then $C\theta'' \subseteq D$ using θ''={x==y==u}, but D does not subsume C as it would require x and y to be constrained by the substitution.

Hint: $C\theta \subseteq D$ implies $\forall C \models \forall D$, where ∀D means the universal closure of all variables in D. eg P(x,a) subsumes P(a,a) ∨ P(a,b) as $P(x,a)\{x==a\} \subseteq P(a,a) \vee P(a,b)$, and $\forall P(x,a) \models P(a,a) \vee P(a,b)$.

Try showing $\forall C \models \forall D$ for some clauses C and D using any proof method that you know and derive the algorithm from there.
Note that $\forall C \models \forall D$ does not always imply $C\theta \subseteq D$. Consider C= $\forall x(\neg P(x) \vee P(f(x)))$ and D = $\forall x(\neg P(x) \vee P(f(f(x))))$: then $C \models D$ but not $C\theta \subseteq D$.

Can you also find an algorithm for **safe factoring** of a clause C; that is, where the factor subsumes C? (Of course, you could always find a factor and then check for subsumption, but are there any better ways?)

==========

Question 5 is for you to think about for next time. It describes a way to control the formation of resolvents, by restricting which literals in a clause may be used. It does this by putting predicates in some order. You might like to consider how Prolog restricts the formation of resolvents. (In fact, for clauses with one positive literal (ie the program!) it forces the positive literal to be used. There is another restriction - what is it?)

5. **PREDICATE ORDERING**: order predicate symbols (any partial order will do but usually a simple sequence is used) and within a clause only literals with <u>minimal</u> ranking predicate in the order can be resolved upon. e.g. if F is ranked before G, and there are F-literals in a clause then the F-literals must be selected before any of the G-literals. Consequently, any G-literals in a clause can be resolved upon *only* if there are *no* F-literals in the clause. In other words, F-literals take priority over G-literals.

Try the predicate ordering method on the clauses:

$$\neg H(a),\ \neg F(x) \vee \neg H(b),\ \ Fx \vee Hx,\ \ \neg Gz \vee \neg Fb,\ \ Gu \vee \neg Fu$$

Use a saturation search and the predicate orderings G <F<H or H <F<G.

We'll consider this restriction, and some others, next week.

How might you justify soundness (easy) and completeness (consider a particular semantic tree).