CS:4350 Logic in Computer Science

Practice Questions and Sample Answers
(100 points)

1. (30 points) Given the following set of clauses,
   1. (p | q | s), 2. (p | -r | -s), 3. (-p | s), 4. (q | r | s), 5. (-p | -q | -r), 6. (-q | r | -s)

   where "-" means negation, please answer the following questions:

   (a) Using the orders of p, q, r, s for splitting (the value 1 is used first for each splitting variable) in DPLL, what will be the truth values of variables, at what levels, when a model is found? What will be the head/tail positions of the clauses if the initial positions are the first and last positions?

   (b) What clauses can be learned if the CDCL technique is used during the execution of DPLL in (a)?

   (c) If we modify DPLL so that it will produce all the models of the clause set, how many models will be found and in what order if the splitting order is the same as in (a)?

   (d) If we feed the clause set to a SAT solver which works as a black box and can produce only one model at a time, what clauses should be added into the clause set so that the SAT solver will produce all models of the clause set?

Answer:

(a) The following literals are assigned to be true with levels: p@1, s@1, -q@2, r@3 when the first model is found. The head/tail literals are underlined when the first model is found:
1. ($\underline{p}$ | q | $\underline{s}$), 2. ($\underline{p}$ | $\underline{-r}$ | -s), 3. $\underline{(-p}$ | $\underline{s}$), 4. (q | $\underline{r}$ | $\underline{s}$), 5. (-p | $\underline{-q}$ | $\underline{-r}$), 6. ($\underline{-q}$ | $\underline{r}$ | -s)

(b) When p@1 (BCP = { s@1 by c3}), and q@2 (BCP = { -r@2 by c5 }), clause c6 becomes conflicting. A new clause (-p | -q | -s) can be learned from c6 by resolution between c5 and c6.

(c) The following models, in the given order, can be found by a modified DPLL:
{ p, s, -q, r }, { p, s, -q, -r }, { -p, q, r, -s }, { -p, q, -r, s }, { -p, q, -r, -s }, { -p, -q, s, -r }.

(d) The rule is that if a model is found by the SAT solver and we want to find the next model, then add the negation of the model to the clause set and feed the new clause set to the solver. For example, if { p, s, -q, r } is the first model by the solver, then add the negation of the model, which is the clause (-p | -s | q | -r), to the clause set and we will get the next model. For the given example, after adding the six clauses made from the negation of the six models, the clause set becomes unsatisfiable and we knew there are only six models for this example.

2. (20 points) In number theory, **Fermat's Last Theorem** states that no three positive integers a, b, and c satisfy the equation $a^n + b^n = c^n$ for any integer value of n greater than 2.
    a. Please express Fermat's Last Theorem in the first-order logic using the following predicates: int, add, exp, where int(x) = true iff x is an integer, add(x, y, z) = true iff x+y=z, exp(x, y, z) = true iff $x^y = z$, and the usual relations over integers, such as >.
    b. Please convert the negation of the formula in (1) into clausal form.

Answer:

(a) $\neg$ ($\exists$u $\exists$v $\exists$w $\exists$n int(u) $\wedge$ int(v) $\wedge$ int(w) $\wedge$ int(n) $\wedge$ (n > 2) $\wedge$
$\forall$x $\forall$y $\forall$z (exp(u, n, x) $\wedge$ exp(v, n, y) $\wedge$ exp(w, n, z) $\rightarrow$ add(x, y, z)))

(b) The clauses from the negation of the formula in (a) are:
{ (int(a)), (int(b)), (int(c)), int(n0), (n0 > 2),
(-exp(a, n0, x) | -exp(a, n0, x) | -exp(a, n0, x) | add(x, y, z)) }
where a, b, c, and n0 are Skolem constants.

3. (20 points) For the following formula of the first order logic, either prove formally that it is valid or give a falsifying interpretation:

$$((\forall x\ p(x)) \leftrightarrow \forall x\ q(x)) \rightarrow \forall x\ (p(x) \leftrightarrow q(x)).$$

Answer: Consider the interpretation I = ({a, b}, {p, q}, { }), where p(a) = q(b) = true and p(b) = q(a) = false. Then both ($\forall$x p(x)) and ($\forall$x q(x)) are false in I, thus (($\forall$x p(x)) $\leftrightarrow$ $\forall$x q(x)) is true in I because (false $\leftrightarrow$ false) is a tautology. On the other hand, $\forall$x (p(x) $\leftrightarrow$ q(x)) is false in I because I(p(x) $\leftrightarrow$ q(x), { x = a }) = false. Thus, the given formula is false in I, thus the given formula is not valid.

4. (30 points) Please write a Prolog program myUnify(T1, T2, U), which takes T1 and T2, two terms built on constants a, b, binary function f, unary function g, and variables x, y, and z, and computes the most general unifier of T1 and T2 in U (as a list of pair(variable, term)) if T1 and T2 are unifiable.

Answer: We will implement the rule-based unification algorithm in Prolog, where myUnify handles a pair of terms.

```
% Variables allowed
myvar(x).
myvar(y).
myvar(z).

% myUnify(T1, T2, U) succeeds if T1 and T2 are unifiable
%       with mgu U, a list of pair(Var, Term).
myUnify(X, X, []) :- !.
myUnify(X, T, [pair(X, T)]) :- myvar(X), !, not(occur(X, T)).
myUnify(T, X, U) :- myvar(X), !, myUnify(X, T, U).
myUnify(g(X), g(Y), U) :- myUnify(X, Y, U).
myUnify(f(X1, X2), f(Y1, Y2), U) :-
    myUnify(X1, Y1, U1),
    subst(U1, X2, X),
    subst(U1, Y2, Y),
    myUnify(X, Y, U2),
    combine(U1, U2, U).

% occur(X, T) succeeds when variable X occurs in T.
occur(X, X) :- !.
occur(X, g(T)) :- occur(X, T).
occur(X, f(T, _)) :- occur(X, T), !.
occur(X, f(_, T)) :- occur(X, T).

% subst(U, T, R) succeeds when unifier U applies to term T and the result is the term R.
subst([], T, T).
subst([pair(X, S) | U], T, R) :- substit(X, S, T, T1), subst(U, T1, R).

% combine(U1, U2, U) succeeds when two unifiers U1 and U2 are combined into U.
combine([], U, U).
combine([pair(X, S) | U1], U2, U) :-
     subst(U2, S, S2), combine(U1, [pair(X, S2) | U2], U).

% substit(V, S, T, E) succeeds when every occurrence of V in T is replaced by S and
% the resulting term from T is E.
substit(X, V, X, V):- !.
substit(X, V, E, R) :-
    E =.. [F|Args0],
    maplist(substit(X, V), Args0, Args),
    R =.. [F|Args].

% utility function
not(P) :- call(P), !, fail.
not(_).
```