Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-10-24
Sal (1)	G37(18)
Tid	8-12
Utb. kod	TDDD08
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Logikprogrammering Skriftlig tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	10
Jour/Kursansvarig Ange vem som besöker salen	Włodzimierz Drabent wlodzimierz.drabent@liu.se
Telefon under skrivtiden	013 - 28 2017
Besöker salen ca klockan	8:40, 10:45
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Veronica Kindeland Gunnarsson veronica.kindeland.gunnarsson@liu.se 013-28 56 34
Tillåtna hjälpmedel	* Ett papper med valfria anteckningar, 1 sida A4 eller 2 sidor A5. Anteckningarna ska signeras på samma sätt som övriga blad och bifogas tentamen vid inlämnandet. * Engelsk ordbok
Övrigt	
Antal exemplar i påsen	

Exam, TDDD08 Logic Programming

2019–10-24, 08:00 – 12:00

Means of assistance (hjälpmedel):
• A sheet of notes – 2 sided A5 or 1 sided A4. The notes should be signed in the same way as the exam sheets and returned together with the exam.
• English dictionary

You may answer in English or Swedish.

Grade limits: 3: 17 p, 4: 23 p, 5: 29 p (for total 35 p).

Remember to give explanations for all answers!
Unexplained answers may be granted 0 points.
For instance, when you write a program you should explain the relations defined by the predicates of the program, and the role of each clause.

GOOD LUCK!

---

1. Determine which of the following pairs of terms are unifiable, and provide a most general unifier (mgu) in case there is one.

   a) $p(\,f(X,Y),Y\,)$
   $p(\,f(V,g(V)),f(Z)\,)$

   b) $[f(Y,Z),Z|Y]$
   $[f(X,g(X),g(V),a,b]$

   c) $p(f(X,g(a)),b,X,f(Y,Y)\,)$
   $p(\,V,Z,g(Z),V)$

   d) $p(\,f(X),X,f(Y),f(Y)\,)$
   $p(\,Z,g(V),f(Z),V\,)$

   (4 points)

2. A directed graph is a tuple (V,E) where V is a set of vertices and $E \subseteq V \times V$ is a set of edges over V. A straightforward Prolog representation of (V,E) is as a term `graph(V,E)` where `V` is a list of vertices and `E` is a list of edges over `V` (both without duplicate elements). Write a Prolog program defining predicates:

   (a) `graph/1` checking that a term represents a graph according to the above specification,

   (b) `path/3` which is true if there exists a path between two nodes in a given graph,

   (c) `three_cycle/1` which is true if the graph contains a cycle of length three. Here, we assume that a cycle is allowed to visit a vertex at most once (except the starting vertex which by definition must be visited twice). E.g., the graph $(\{a,b,c\},\{(a,b),(b,c),(c,a)\})$ contains a cycle of length three, but the graph $(\{a,b\},\{(a,a),(a,b),(b,a)\})$ only contains cycles of length 1 and 2.

Your program may use `dif/2`, `member/2`, and `nonmember/2` for checking membership, respectively non membership, of a list. Otherwise your programs should be definite logic programs. So Prolog arithmetic, negation, `( -> ; )`, and other Prolog built-in predicates should not be used. For a full score, your `path/3` program should be able to find a path between two nodes even if the input graph contains cycles. (4 points)

3. Assume that a predicate `pre/2` (short for *precedes*) describes an ordering on terms (so that $\texttt{pre}(v_1, v_2)$ holds when v_1, v_2 are distinct terms and v_1 precedes v_2 in this ordering; for each pair of distinct terms v_1, v_2 either $\texttt{pre}(v_1, v_2)$ or $\texttt{pre}(v_2, v_1)$ holds). Write a program defining predicates:

   (a) `max/2` (maximal value) which is true if its first argument is a list, and the second argument is the maximal value of an element of the list, according to the ordering given by `pre/2` (e.g., if `pre/2` is the usual ordering $<$ over the integers then the maximal value in $[1, 2, 2, 0]$ is 2).

   (b) `rmv/2` (remove maximal value) which is true if the first argument is a list with the maximal value m, and the second argument is the first one with all occurrences of m removed.

   (c) `rbm/2` (replace by maximal) which is true if both arguments are lists of the same length, where each element in the second list is the maximal element of the first list.

   Your programs should be definite logic programs. So Prolog arithmetic, negation, `( -> ; )`, and other Prolog built-in predicates should not be used. For a full score for 3c, the program should avoid obvious inefficiencies, like unnecessary multiple traversals of a list.

   Hint: when defining `rbm/2`, begin with a simple, possibly inefficient program, and then consider the possibility of constructing the second list simultaneously with finding the maximal element.
   (4 points)

4. Consider the following definite program P:

$$p(f(h(X, X))).$$
$$p(h(V, f(V))).$$
$$p(f(g(X))) \leftarrow p(h(X, Y)), p(Y).$$

   (a) Assume that the vocabulary $\mathcal{A}$ contains one constant a, two one-argument function symbols f, g, one two-argument function symbol h, and one predicate symbol p. What is the Herbrand universe $\mathbf{U}_\mathcal{A}$ corresponding to $\mathcal{A}$?
   Describe it in the form "$\mathbf{U}_\mathcal{A}$ is the set of all ... constructed out of symbols ...", and give three example elements of $\mathbf{U}_\mathcal{A}$, so that each symbol is employed.

   (b) Is the Herbrand interpretation $I = \{\, p(h(t, f(t'))),\ p(f(t)) \mid t, t' \in \mathbf{U}_\mathcal{A} \,\}$ an Herbrand model of P?

(c) Find the set $PTR(P)$ of atomic logical consequences of the program. Alternatively, find the least Herbrand model $\mathbf{M}_P$ of the program.

(d) Give an example of a ground atom which is a logical consequence of P; the atom should not be an instance of a unary clause of the program.

(e) Give an example of a ground atom which is not a logical consequence of P. (The predicate symbol of the atom should be p.)

(f) Give an example of a non-ground atom which (1) is a logical consequence of P, and (2) is not an instance of any unary clause of the program.

(6 points)

5. For a chosen query Q and a chosen subset $P_5 \subseteq P$ of the previous program, construct two SLD-trees – one finite and one infinite. (2 points)

6. Consider the program PREFIX

$$p([\,], L).$$
$$p([H|P], [H|L]) \leftarrow p(P, L).$$

(a) Is the program correct with respect to the specification

$$S_1 = \{\, p([t_1, \ldots, t_m], [t_1, \ldots, t_n]) \in \mathbf{B}_\mathcal{A} \mid 0 \leq m \leq n \,\} \ ?$$

A brief explanation is sufficient here.

$\mathbf{B}_\mathcal{A}$ is the Herbrand base.

(b) Using a standard method, prove that the the program is correct with respect to the specification

$$S = \left\{ p([t_1, \ldots, t_m], u) \in \mathbf{B}_\mathcal{A} \;\middle|\; \begin{array}{l} m \geq 0, \text{ and} \\ \text{each } t_i \ (i = 1, \ldots, m) \\ \text{is a subterm of } u \end{array} \right\}.$$

(In other words, prove that in each answer of the program the first argument of p is a list, and each element of the list occurs somewhere within the second argument). (4 points)

7. Consider the following DCG.

```
y(0) --> [].
y(s(X)) --> [0], y(X).
y(s(X)) --> [1], y(X).
```

What is the language of `y(s(s(s(0))))`? Give an example of an element in this language by either (1) providing a derivation of the DCG, or (2) translating the grammar to Prolog and provide a proof tree or a successful SLD-derivation.

What is the language of `y(s(s(X)))`? (3 points)

8. Consider the general logic program

$$o(s(0)).$$
$$o(s(s(X))) \leftarrow \neg e(X).$$
$$e(0).$$
$$e(X) \leftarrow \neg o(X).$$

Draw SLDNF-forests for the queries $e(s(s(0)))$, and $e(s(s(X)))$. Make it clear which leaves are floundered, which trees are finitely failed, which branches are successful, and what are their answers. What answer would Prolog yield for the last query, with $\neg$ implemented as \+ ? (4 points)

9. Consider the general logic program P_9:

$$p \leftarrow \neg q.$$
$$q \leftarrow \neg p.$$

What are the Herbrand models of P, and what are the stable models of P? Compare this to the results of SLDNF-resolution for P and the queries q and p. (3 points)

10. Choose one case from the list below, and explain the notion(s).

Your explanation should be short but precise, and should show that you understand the notions. The chosen notions should not be explained in your sheet of notes.

(a) Interpretation. Model of a formula.

(b) Logical consequence.

(c) A substitution more general than another one. (Include an example of two substitutions, one more general than the other).

(d) Occur check. Its treatment in Prolog.

(e) SLD-tree, proof tree (called also "implication tree").

(f) Completeness of SLD-resolution.

(g) Incorrectness diagnosis.

(h) CLP(FD).

(i) Closed world assumption.

(1 point)