Problem Set for Tutorial 6 — TDDD08

Version 1.6

1. Given the program below, draw an SLDNF-forest for query $\neg \mathit{member}(X, [a, b])$ and for $\neg \mathit{member}(c, [a, b])$. What result would you expect in Prolog with the queries \+member($X, [a, b]$) and \+member($c, [a, b]$)? If the results differ, why is that the case?

```
member(X, [X|_]).
member(X, [_|L]) :- member(X,L).
```

2. A ground propositional formula can be represented in Prolog by a term over `true`, `false`, `and/2`, `or/2`, and `not/1`. For example, `and(true, or(not(false), true))` would be an example of such a formula in this representation. Assume that we want to define a predicate `true_formula(F)` which is true if `F` is a ground propositional formula in the specified representation. An attempt using negation as failure in Prolog might look as follows:

```
true_formula(true).
true_formula(and(X, Y)) :- true_formula(X), true_formula(Y).
true_formula(or(X, _Y)) :- true_formula(X).
true_formula(or(_X, Y)) :- true_formula(Y).
true_formula(not(X)) :- \+ true_formula(X).
```

   (a) Give an example of two queries, one ground and one non-ground, which produces unexpected or unintended answers.

   (b) Rewrite the above program without using negation. What is the result of the previous two queries to the resulting program?

3. Consider a program WIN:

$$w(X) \leftarrow m(X, Y), \neg w(Y). \quad m(a, b). \quad m(b, a). \quad m(b, c).$$

   (Predicate $m/2$ describes moves in a game; $w/1$ describes winning positions – a position X wins if you can move from X to a position Y in which your opponent cannot win.)

   (a) Draw the SLDNF-forest for query $w(X)$ under the Prolog selection rule. Make it clear which branches are successful derivations and what their answers are, which leaves are floundered, and which trees are finitely failed.

   (b) How is the forest changed if we add $m(c, d)$ to the program?

   (c) Construct the completion *comp*(WIN) of the program (except for equality axioms). Explain whether $\neg m(c, a)$ and $\neg w(c)$ are logical consequences of *comp*(WIN). The same for $w(b)$.

4. (Exam exercise) Consider the following general program P:

$$p(X) \leftarrow \neg q(X). \qquad q(s(Y)) \leftarrow p(Y). \qquad q(a).$$

   (a) Draw SLDNF-forests for queries $q(X)$, $p(s(b))$, and $p(s(s(b)))$. Make it clear which trees are finitely failed, which leaves are floundered, which branches are successful derivations, and what their answers are.

   (b) Construct the completion $comp(P)$ of the program (except for equality axioms). Explain whether $q(a)$ is a logical consequence of $comp(P)$; the same for $q(s(b))$.

5. Consider programs

$$P_1: \; p \leftarrow \neg p \qquad\qquad P_2: \; \begin{array}{l} p \leftarrow \neg q \\ q \leftarrow \neg p \end{array} \qquad\qquad P_3: \; \begin{array}{l} p \leftarrow \neg q \\ q \leftarrow \neg r \\ r \leftarrow \neg p \end{array}$$

   (a) Is an Herbrand interpretation $\{p\}$ a model of P_2? What about P_3?

   (b) Which of the programs have stable models? Find all of them.

6. Find the stable models of programs

$$P_4: \; \begin{array}{l} p \leftarrow p. \\ p \leftarrow r. \\ q \leftarrow \neg p. \\ s \leftarrow \neg q, r \end{array} \qquad\qquad P_5: \; \begin{array}{l} p \leftarrow \neg q. \\ q \leftarrow \neg p. \\ r \leftarrow q, \neg p. \\ f \leftarrow \neg f, p \end{array}$$

   Comment: Note that P_4 is stratified ("no recursion through negation"). Hence it has a single stable model, which can be constructed stepwise for consecutive strata (1. the clauses for p, 2. that for q, 3. that for s).

7. Assume a small Herbrand universe (with > 1 elements) and find the stable models of

$$P_6: \; \begin{array}{l} p(a) \leftarrow \neg q. \\ q \leftarrow \neg p(a). \\ p(a) \leftarrow p(X). \\ p(X) \leftarrow p(a). \end{array}$$

8. Consider a program P containing a clause

$$f \leftarrow \neg f, \vec{L}$$

   where $\vec{L}$ is a conjunction of literals, and f does occur elsewhere in P. Show that f and $\vec{L}$ are false in each stable model of P.

   This is a usual way of forcing something to be false, a special notation $:\!-\, \vec{L}$ is introduced for such a clause.