02156 Exercises-10

Jørgen Villadsen

2021

Exercise 1

Transform each of the following formulas to clausal form:

$$\forall x(p(x) \rightarrow \exists y q(y))$$
$$\forall x \forall y(\exists z p(z) \wedge \exists u(q(x,u) \rightarrow \exists v q(y,v)))$$
$$\exists x(\neg \exists y p(y) \rightarrow \exists z(q(z) \rightarrow r(x)))$$

Check the result using the file `qed.pl` available on CampusNet (Program files folder in the top folder) — but only when you have first *manually* transformed the formula to clausal form.

Example:

```
?-  skolem(ex(X, all(Y, p(X,Y))) =>
           all(Y, ex(X, p(X,Y)))).
(Ex1Ax2p(x1,x2) => Ax2Ex1p(x1,x2))
(Ex1Ax2p(x1,x2) => Ax3Ex4p(x4,x3))
(~Ex1Ax2p(x1,x2) \ Ax3Ex4p(x4,x3))
(Ax1Ex2~p(x1,x2) \ Ax3Ex4p(x4,x3))
Ax1Ex2Ax3Ex4(~p(x1,x2) \ p(x4,x3))
Ax1Ex2Ax3Ex4(~p(x1,x2) \ p(x4,x3))
Ax1Ax2(~p(x1,f1(x1)) \ p(f2(x2,x1),x2))
[~p(x1,f1(x1)),p(f2(x2,x1),x2)]

Yes
```

Exercise 2

Consider the use of lists as sets (no duplicate elements in the lists).

Write a deterministic program `power(+Set,?PowerSet)` that unifies `PowerSet` with the powerset of `Set` (the powerset of a set is the set of all its subsets).

For example:

```
?- power([a,b,c],P).

P = [[a, b, c], [a, b], [a, c], [a], [b, c], [b], [c], []] ;

No
```

The order of the elements does not matter.

Use the following program `sub(+Set,?SubSet)` that unifies `SubSet` with the subsets of `Set` on backtracking.

```
sub([],[]).
sub([X|A],[X|B]) :- sub(A,B).
sub([_|A],B) :- sub(A,B).
```

Exercise 3

Prolog uses a so-called logical database update view: If the execution of a program changes the clause database for the program itself then only subsequent executions of the program are affected.

First enter the query `?- dynamic p/1.` And the query `?- repeat.` Type `w` and enter a blank line to stop the loop.

Then consider the following query:

```
?- findall(X,( assertz(p(a) :- assertz(p(b))), p(X) ),T).
```

Predict the result of the query when it is repeated five times (also provide a listing of the clause database at the end).

Exercise 4

Write a program `ss(+List1,?List2)` that succeeds if and only if `List2` is a permutation of `List1` and the elements in `List2` are ordered by the standard order. Use the "Slowsort" algorithm, that is, generate permutations of the list until the result is found. Your solution should look something like this:

```
ss(List1,List2) :- permutation(List1,List2), is_ordered(List2).
```

Note that `permutation` is library auto-loaded in SWI-Prolog, which means that it can be redefined but are otherwise like built-in predicates.

Is your solution deterministic? Make it determinitic, if it is not so.

Exercise 5

Consider the following deterministic program `mysort(+List,?Sorted)` that sorts a list with arbitrary elements according to the standard order of terms and such that duplicates are merged.

```
mysort(List,Sorted) :- setof(X,member(X,List),S) -> Sorted = S ; Sorted = [].
```

```
?- mysort([b,a,c,a],S).
```

```
S = [a, b, c]
```

```
Yes
```

If `bagof` is used instead of `setof` then the resulting list is the same as the original one.

Note that duplicates are merged and in this respect it is like the SWI-Prolog built-in predicate `sort` but SWI-Prolog also has a built-in predicate `msort` that does not merge duplicates.

Write a quite simple program `mymsort(+List,?Sorted)` as an alternative to `msort` but using more or less only a single call to `setof` and the fact that in SWI-Prolog the variables are ordered by their address (the oldest variable first).

```
?- mymsort([b,a,c,a],S).
```

```
S = [a, a, b, c]
```

```
Yes
```

The basic predicate `member` and the predicate `findall` can be used.

Hint: Consider first the query: `?- setof((X,_),(X = b ; X = a),T).`