/\* membership of X in S --- from class\*/

mem(X,[]) :- fail.

mem(X,[X|\_]) :- !.

mem(X,[\_|R]) :- mem(X,R).

% intersection

interI(S1, [ ], [ ]) :- !.

interI([ ], S2, [ ]) :- !.

interI([X|R], S2, [X|S3]) :- mem(X, S2), interI(R, S2, S3). %if X is a member of Z, put it in S3.

interI([X|R], S2, S3) :- \+mem(X, S2), interI(R, S2, S3). %if X is not a member of Z, don't put it in S3.

% difference -- opposite conditions as intersection

diffI(S1, [], S1) :- !.

diffI([], S2, []) :- !.

diffI([X|R], S2, [X|S3]) :- \+mem(X, S2), diffI(R, S2, S3).

diffI([X|R], S2, S3):- mem(X, S2), diffI(R, S2, S3).

/\* append(L1, L2, L3) -- append lis  L1 to list L2 to get list  L3 -- given as reference\*/

append( [ ], L, L).

append( [X|R], L, [X|Z]) :- append(R, L, Z).

/\* mapcons(X,L1, L2) --  cons the element X to each list in L1 to get L2 -- given reference\*/

mapcons(X, [ ], [ ]) :- !.

mapcons(X, [Y|R], [ [X|Y] | Z ]) :- mapcons(X, R, Z).

% cartesian product

cartesianI(S1, [], []) :- !.

cartesianI([], S2, []) :- !.

cartesianI([X|L], S2, S3) :- mapcons(X, S2, S2\_), cartesianI(L, S2, V), append(S2\_, V, S3).

|  |  |
| --- | --- |
| Checking implementation of union. The last case shows there are no duplicates. |  |
| Power set. |  |
| Intersection with empty sets. |  |
|  |
| Intersection examples. |  |
|  |
| Set difference examples. |  |
|  |
|  |
| Cartesian product with empty sets |  |
| Cartesian product with same sets but different orders. |  |
| To check that order doesn’t matter, take difference of both products. It results in an empty set. |  |