farmer.pl

Version 1

Based on this version (https://cseweb.ucsd.edu/classes/fa09/cse130/misc/prolog/goat_etc.html) from UCSD.

```
%% Travel E -> W

travel(e,w).
%% Travel W -> E

travel(w,e).

%% Possible moves.

move([X,X,Goat,Cabbage],wolf,[Y,Y,Goat,Cabbage]) :- travel(X,Y).
move([X,Wolf,X,Cabbage],goat,[Y,Wolf,Y,Cabbage]) :- travel(X,Y).
move([X,Wolf,Goat,X],cabbage,[Y,Wolf,Goat,Y]) :- travel(X,Y).
move([X,Wolf,Goat,Cabbage],nothing,[Y,Wolf,Goat,Cabbage]) :- travel(X,Y).

%% Safe conditions.

safe([X,_,X,_]). % Goat is on the same bank as farmer.
safe([X,X,_,X]). % Wolf and cabbage are on the same bank as farmer

solve([e,e,e,e],[]).
solve(State,[FirstMove|OtherMoves]) :- move(State, FirstMove, NextState), safe(NextState),

% ?- length(X,7), solve([w,w,w,w],X).

% X = [goat, nothing, wolf, goat, cabbage, nothing, goat] ;

% X = [goat, nothing, wolf, goat, cabbage, nothing, goat] ;

% X = [goat, nothing, cabbage, goat, wolf, nothing, goat] ;

% X = [goat, nothing, cabbage, goat, wolf, nothing, goat] ;

% false.


% ?- length(X,7), setof(t,solve([w,w,w,w],X),_).

% X = [goat, nothing, cabbage, goat, wolf, nothing, goat] ;

```

```
% X = [goat, nothing, wolf, goat, cabbage, nothing, goat].
```

Version 2 Additions

```
not_member(_, []) :- !.

not_member(X, [Head|Tail]) :-
     X \= Head,
    not_member(X, Tail).

solve2(AnsPath) :- solve2([[w,w,w,w]], AnsPath).
solve2([S|_], [S]) :- S = [e,e,e,e], !.
solve2([S|Path], [S|AnsPath]) :- move(S, _, NextState), safe(NextState),
                                 not_member(NextState,[S|Path]), solve2([NextState, S | Pat

% ?- solve2(X).

% X = [[w, w, w, w], [e, w, e, w], [w, w, e, w], [e, e, e, w], [w, e, w, w], [e, e, w, e],

% X = [[w, w, w, w], [e, w, e, w], [w, w, e, w], [e, e, e, w], [w, e, w, w], [e, e, w, e],

% X = [[w, w, w, w], [e, w, e, w], [w, w, e, w], [e, w, e, e], [w, w, w, e], [e, e, w, e],

% X = [[w, w, w, w], [e, w, e, w], [w, w, e, w], [e, w, e, e], [w, w, w, e], [e, e, w, e],
```