```
% NOTICE: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%  COPYRIGHT (2009) University of Dallas at Texas.                              %
%                                                                               %
%  Developed at the Applied Logic, Programming Languages and Systems            %
%  (ALPS) Laboratory at UTD by Feliks Kluzniak.                                 %
%                                                                               %
%  Permission is granted to modify this file, and to distribute its             %
%  original or modified contents for non-commercial purposes, on the            %
%  condition that this notice is included in all copies in its                  %
%  original form.                                                               %
%                                                                               %
%  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,              %
%  EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES              %
%  OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND              %
%  NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR                 %
%  ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR                %
%  OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING             %
%  FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR                %
%  OTHER DEALINGS IN THE SOFTWARE.                                              %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% An auxiliary of the verifier: prepares the counterexample for presentation.


% If the path contains a loop, get the shortest prefix that exhibits the loop.
% If the path contains no loop, but ends in a variable, get the closed path.

looping_prefix( Path, Prefix ) :-
        once( looping_prefix_( Path, [], ReversedPrefix ) ),
        once( reverse_list( ReversedPrefix, Prefix ) ).

%
looping_prefix_( [], ReversedPrefixSoFar, ReversedPrefixSoFar ).

looping_prefix_( [ Last ], ReversedPrefixSoFar, ReversedPrefix ) :-
        (
            var( Last )
        ->
            ReversedPrefix = ReversedPrefixSoFar
        ;
            ReversedPrefix = [ Last | ReversedPrefixSoFar ]
        ).

looping_prefix_( [ Node | Nodes ], ReversedPrefixSoFar, ReversedPrefix ) :-
        (
            member( Node, ReversedPrefixSoFar )
        ->
            ReversedPrefix = [ Node | ReversedPrefixSoFar ]
        ;
            looping_prefix_( Nodes,
                             [ Node | ReversedPrefixSoFar ], ReversedPrefix
                           )
        ).
```

```
% reverse_list/2 is available as reverse/2 both in Eclipse and Sicstus,
% but in Sicstus one has to load a library. :-(

reverse_list( List, Reversed ) :-
        reverse_list_( List, [], Reversed ).

%
reverse_list_( [], ReversedPrefix, ReversedPrefix ).

reverse_list_( [ H | T ], ReversedPrefix, Reversed ) :-
        reverse_list_( T, [ H | ReversedPrefix ], Reversed ).
```