```
% 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.                                     %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```


```
%% A consistency checker for automata.  See "verifier.tlp".

% Check the consistency of the automaton’s description.

:- [ ’partition_graph.pl’ ].
```

```
% NOTE: The dynamic declaration is necessary for Eclipse.

:- dynamic automaton_error/0.

automaton_error.  % Will be retracted: needed to suppress a warning from Sicstus

check_consistency :-
        retractall( automaton_error ),
        check_connectedness,
        check_propositions,
        check_transitions,
        (
            automaton_error
        ->
            fail
        ;
            true
        ).


% If the graph is not connected, print a warning.

check_connectedness :-
        partition( Components ),
        length( Components, NumberOfComponents ),
        (
            NumberOfComponents =:= 1                                    % connected
        ->
            true
        ;
            write( 'WARNING: The graph is not connected!' ),
            nl,
            write( 'The partitions are: ' ),
            write( Components ),
            nl
        ).
```

```
% Make sure propositions don't clash with operators.

check_propositions :-
        proposition( P ),
        (
            \+ atom( P )
        ->
            write( 'A proposition must be an atom: ' ),
            write( '\"' ),
            write( P ),
            write( '\"' ),
            nl,
            assert( automaton_error )
        ;
            true
        ),
        (
            member( P, [ 'v', 'x', 'f', 'g', 'u', 'r' ] )
        ->
            write( '\"v\", \"x\", \"f\", \"g\", \"u\" and \"r\" ' ),
            write( 'cannot be propositions: ' ),
            write( '\"' ),
            write( P ),
            write( '\"' ),
            nl,
            assert( automaton_error )
        ;
            true
        ),
        fail.

check_propositions.
```

```
% Make sure that there is no state with no outgoing transitions, and that all
% transitions are between states.

check_transitions :-
        trans( S1, S2 ),
        (
            (var( S1 ) ;  var( S2 ) ; \+ state( S1 ) ; \+ state( S2 ))
        ->
            write( 'Transitions can only occur between states: ' ),
            write( S1 ),
            write( ' ---> ' ),
            write( S2 ),
            nl,
            assert( automaton_error )
        ;
            true
        ),
        fail.

check_transitions :-
        state( S ),
        (
            (\+ trans( S, _Set ) ; trans( S, [] ))
        ->
            write( 'No transition out of state ' ),
            write( S ),
            nl,
            assert( automaton_error )
        ;
            true
        ),
        fail.

check_transitions.

%-------------------------------------------------------------------------------
```