PROOF FOUNDATIONS

(W) WEISER DEFINITION OF SLICING:
Given a program P, a slicing criterion C=<v,s> where v is a variable at statement s, and a slice S:
If P halts on input I, then the value of v at statement s each time s is executed in P is the same in P and S. If P fails to terminate normally, s may be executed more times in S than in P, but P and S compute the same values for v each time s is executed by P.

(A) DATA DEPENDENCE:
We say there exists a data dependence between two expressions when the first expression defines the value of a variable and the second one uses this value in at least one of the possible program executions without being any other expression modifying it.
NOTE: We consider that the arguments passed in a function call and the parameters of that function are a specific case of data dependence where the expression changes its name.

(B) CONTROL DEPENDENCE:
There exists a control dependence between two expressions when the second expression cannot be evaluated without evaluating the first expression.

(C) SEQUENTIAL REDUNDANCE:
When the return expression of a block or a function (the last expression of the block in Erlang) is a variable defined in the previous expression, this can be deleted avoiding the definition of this variable and returning the result of the previous expression, taking this expression the last position of the block and being returned in consequence.

(D) SYNTAX ERRROR:
We say there exists a syntax error in a program when the removal or modification of a chosen expression transforms the program into a non-executable state.

(E) SEMANTIC MODIFICATION:
There exists a semantic modification in an expression when the modification of one of its subexpressions modifies the behaviour of the whole expression.

(F) ABSORBING PROPERTY:
A clause of a conditional or a function statement is absorbing when its guard is always evaluated to true or its pattern always matches.

(G) FULL TEST VALIDATION:
There exists full test validation when an original program and a slice extracted from it can be executed with all possible input values of the original program and the values of the slicing criterion are the same in both executions.
NOTE: We consider in this definition also programs with slicing criteria that are independent of program inputs, where there is only one possible execution.

COLOUR LEGEND

Black: Expressions deleted by executing phase 1 (iterative slicing with the selected slicers)
Red: Expressions deleted by executing phase 2 (modified ORBS algorithm)
Green: Expressions remaining in the quasi-minimal slices
Orange: Slicing Criterion
Brown: Expressions deleted by the demonstration but not automatically by the process

NOTE1: We will not prove whether black expressions of the program code can be deleted or not because they have been deleted by phase 1. Phase 1 produces a complete slice of the original code, so we can guarantee that these expressions are not part of the slice.
NOTE2: Our slices keep the syntax of the original program (we are not interested in amorphous slices). However, in order to make the final slice executable, some modifications of the source code are compulsory (e.g., replacing calls to deleted functions with a constant called "undef"). Therefore, we allow for some modifications of the source code to produce executable slices. The modifications made never affect the behaviour of the source code, they just ensure that the final code is a valid Erlang program.

```
%-----------------------------------------------------------------------------------------
%-----------------------------------------------------------------------------------------
%-- bench9.erl
%--
%-- AUTHORS:      Anonymous
%-- DATE:         2016
%-- PUBLISHED:    Software specially developed to test the define and spec expressions.
%-- COPYRIGHT:    Bencher: The Program Slicing Benchmark Suite for Erlang
%--               (Universitat Politècnica de València)
%--               http://www.dsic.upv.es/~jsilva/slicing/bencher/
%-- DESCRIPTION
%-- The program receives a number and a state to switch the stove as inputs. If the state
%-- is ON and the number is greater than twenty, the program returns a success message,
%-- if it is ON and the number is less or equal to twenty the program returns an error
%-- message. The OFF input state will always provide a success message.
%-----------------------------------------------------------------------------------------
%-----------------------------------------------------------------------------------------
```

```
-module(bench9).
-export([main/2]).
-define(ERROR,failure).                                 %This expression is necessary to define the required macro
                                                        ?ERROR
-define(ON,enable).                                     %This expression is necessary to define the required macro
                                                        ?ON
-define(OFF,disable).                                   %This expression is necessary to define the required macro
                                                        ?OFF
-type num() :: integer().
-type word() :: string().
-type thing() :: atom().
-type switch() :: ?ON | ?OFF.
-type reply() :: word() | ?ERROR.
-spec light(N::num(),L::switch()) -> {'ok',word()} | {'error',thing()}.
light(N,L) ->                                           %Given (A), N and L are necessary w.r.t. the if expression
        if                                              %The if expression cannot be deleted because it is the
                                                        only expression of the light function and defines its
                                                        returned value. It cannot be replaced with undef (NOTE2)
                                                        because it would prevent to satisfy
                                                        (1),(2),(3),(4),(5)&(6)
            N > 20 andalso L==?ON ->                    %This clause cannot be deleted because it would prevent to
                                                        satisfy (1)
                                                        %Replace N > 20 andalso L==?ON with true (NOTE2) would
                                                        prevent to satisfy (2),(3),(4),(5)&(6) because this clause
                                                        would become absorbent (F)
                                                        %N > 20 cannot be replaced with true (NOTE2) because it
                                                        would prevent to satisfy (2)&(3) because of (E)
                                                        %L == ?ON cannot be replaced with true (NOTE2) because it
                                                        would prevent to satisfy (4) because of (E)
                                                        %Replace N with undef (NOTE2) would prevent to satisfy
                                                        (2)&(3) due to (E)
                                                        %Replace 20, L or ?ON with undef (NOTE2) would prevent to
                                                        satisfy (1) because this clause would become unreachable
                                                        because of (E). Replace L and ?ON with undef simultaneously
                                                        would prevent to satisfy (4) due to (E)
                {'ok', "Success switching ON"};         %{'ok', "Success switching ON"} cannot be deleted because
                                                        it is the only statement of the if clause and one of the
                                                        possible returned values of the light function. Replace it
                                                        with undef (NOTE2) would prevent to satisfy (1)
                                                        %"Success switching ON" cannot be replaced with undef
                                                        (NOTE2) because it would prevent to satisfy (1)
            N =< 20 andalso L==?ON ->                   %This clause cannot be deleted because it would prevent to
                                                        satisfy (2)&(3)
                                                        %N =< 20 andalso L==?ON cannot be replaced with true
                                                        (NOTE2) because it would prevent to satisfy (4),(5)&(6)
                                                        because this clause would fulfill (F)
                                                        %L == ?ON cannot be replaced with true (NOTE2) because it
                                                        would prevent to satisfy (5)&(6) because of (E)
                                                        %N =< 20 can be replaced with true. The information
                                                        apported by the condition N =< 20 is redundant because if
                                                        variable N is not greater than 20 it will surely be less
                                                        or equal than 20 and the second condition is the same for
                                                        both clauses
                                                        %Replace N, L or ?ON with undef (NOTE2) would prevent to
                                                        satisfy (2)&(3) because this clause would be converted in
                                                        an unreachable clause
                                                        %Replace L and ?ON simultaneously would prevent to satisfy
                                                        (5)&(6)
                {'error', ?ERROR};                      %{'error', ?ERROR} cannot be deleted because it is the
                                                        only statement of the if clause and one of the possible
                                                        returned values of the light function. Replace it with
                                                        undef (NOTE2) would prevent to satisfy (2)&(3)
                                                        %?ERROR cannot be replaced with undef (NOTE2) because it
                                                        would prevent to satisfy (2)&(3)
            true ->                                     %This clause cannot be deleted because it would generate
                                                        a matching error in executions (4),(5)&(6). This could be
                                                        avoided replacing the previous clause guard with true, but
                                                        this would prevent to satisfy (4),(5)&(6)
                {'ok', "Success switching OFF"}         %{'ok', "Success switching OFF"} cannot be deleted because
                                                        it is the only statement of the if clause and one of the
                                                        possible returned values of the light function. Replace it
                                                        with undef (NOTE2) would prevent to satisfy (4),(5)&(6)
                                                        %"Success switching OFF" cannot be replaced with undef
                                                        (NOTE2) because it would prevent to satisfy (4),(5)&(6).
        end.
-spec stove(P::num(),L::switch()) -> reply().
stove(P,L) ->                                           %Given (A), P and L are necessary w.r.t. the if expression
        if                                              %The if expression cannot be deleted because it is the
                                                        only expression that can assign a value to the Reply
                                                        variable (returned value of the stove() function), and in
                                                        consequence to the SC A.
```

```
            L == ?ON ->                          %This clause cannot be deleted because it would prevent
                                                 to satisfy (1),(2)&(3)
                                                 %Replace L == ?ON with true (NOTE2) would produce (F) and
                                                 thus (4),(5)&(6) are not satisfied because of (E)
                                                 %Replace L or ?ON with undef would prevent to satisfy
                                                 (1),(2) & (3). Replace both simultaneously would prevent
                                                 to satisfy (4),(5)&(6)
                {Res,Reply} = light(P,L);        %{Res,Reply} = light(P,L) cannot be replaced with undef
                                                 (NOTE2) because it assigns one of the possible values to
                                                 the returned expression of the stove() function. Neither
                                                 can it be deleted because it is the only expression of the
                                                 clause
                                                 %Replace light(P,L) with undef (NOTE2) would prevent to
                                                 reach the SC because of a matching error
                                                 %Replace {Res,Reply} with _ (NOTE2) would produce (D) with
                                                 variable Reply. This could be solved by deleting the Reply
                                                 expression in the stove function or replacing it with
                                                 undef, but this would prevent to satisfy (1),(2),
                                                 (3),(4),(5)&(6)
                                                 %Replace Reply with _ (NOTE2) would produce (D) with
                                                 variable Reply in (1),(2)&(3). Solving this error deleting
                                                 Reply or replacing it with undef would prevent to satisfy
                                                 (1),(2),(3),(4),(5)&(6)
                                                 %Given (A), P and L are necessary w.r.t. light(N,L)
            true ->                              %This clause cannot be deleted because it would prevent
                                                 to satisfy (4),(5)&(6) due to a matching error.
                {Res,Reply} = light(P,?OFF)      %Given (G), P can be deleted
                                                 %{Res,Reply} = light(P,?OFF) cannot be replaced with undef
                                                 (NOTE2) because it assigns one of the possible values to
                                                 the returned expression of the stove() function. Neither
                                                 can it be deleted because it is the only expression of the
                                                 clause
                                                 %Replace light(P,?OFF) with undef (NOTE2) would prevent to
                                                 reach the SC because of a matching error
                                                 %Replace {Res,Reply} with _ (NOTE2) would produce (D) with
                                                 variable Reply. This could be solved by deleting the Reply
                                                 expression in the stove function or replacing it with
                                                 undef, but this would prevent to satisfy (1),(2),
                                                 (3),(4),(5)&(6)
                                                 %Replace Reply with _ (NOTE2) would produce (D) with
                                                 variable Reply in (4),(5)&(6). Solving this error deleting
                                                 Reply or replacing it with undef would prevent to satisfy
                                                 (1),(2),(3),(4),(5)&(6)
        end,
        Reply.                                   %Replace Reply with undef (NOTE2) or delete it would
                                                 prevent to satisfy (1),(2),(3),(4),(5)&(6) because the
                                                 retuned value of the function will be the {Res,Reply} tuple
                                                 returned from the if expression

main(N,State) when (N > -480 andalso N =< 520)
andalso (State == ?ON orelse State == ?OFF) ->   %Given (A), N and State are necessary w.r.t. the
                                                 stove(N,State) expression
                                                 %The when (N > -480 andalso N =< 520)
                                                 andalso (State == ?ON orelse State == ?OFF) guard can be
                                                 deleted because of (W). The computed values of the slice
                                                 are not important if the original program fails to
                                                 terminate normally
        A=stove(N,State).                        %A cannot be deleted because it is the SC
                                                 %stove(N,State) is the only expression that can assign a
                                                 value to the SC. Replace it with undef would prevent to
                                                 satisfy (1),(2),(3),(4),(5)&(6)
                                                 %Given (A), N and State are necessary w.r.t. stove(P,L)
```

EXECUTION RESULTS:

	SLICING CRITERION
(1) L == ?ON && N > 20 -> State = ?ON, N = 30	SC = “Success switching ON”
(2) L == ?ON && !(N > 20) && N < 20 -> State = ?ON, N = -10	SC = failure
(3) L == ?ON && !(N > 20) && !(N < 20) -> State = ?ON, N = 20	SC = failure
(4) !(L == ?ON) && N > 20 -> State = ?OFF, N = 30	SC = “Success switching OFF”
(5) !(L == ?ON) && !(N > 20) && N < 20 -> State = ?OFF, N = -15	SC = “Success switching OFF”
(6) !(L == ?ON) && !(N > 20) && !(N < 20) -> State = ?OFF, N = 20	SC = “Success switching OFF”