Bill J. Ellis
July 7, 2005

No one said it'd be easy: SPADEase Version 1

Contents

*Hums are short notes intended for distribution between those involved with EPSRC grant GR/T12289/01. Hums describe ϵ-baked ideas, where $1 \geq \epsilon \geq 0$. (Hum refers to both the Praxis Humming bird and Winnie-the-Pooh's indirect approach to writing: "Poetry and Hums aren't things which you get, they're things which get you.").

1 Introduction

After much effort, a first version of SPADEase has been produced[1]. This tool successfully eliminates the complexities in the research system NuSPADE to produce a viable, industrially strong, proof planner that can demonstrably prove problems beyond current automated proof support for the SPARK Approach.

The original SPADEase project advocated trying to squeeze in a speculative implementation of program analysis features. While a "plan of action" note was constructed regarding program analysis (Hum 13) this was not implemented. Nevertheless, this note, combined with the details of NuSPADE, gives some evidence that a program analysis component would be feasible given additional time.

Due to running late, this first version of SPADEase has been subjected to limited evaluation. However, as SPADEase requires zero manual configuration to analyse examples, a reasonable evaluation should be achievable in the closing days of the project.

The rest of this note collects the output from SPADEase on a single example. This provides a witness proof that SPADEase can boost proof automation and that this is achieved fully automatically.

2 Example

2.1 Prelude

Zero manual intervention was required to generate the results for this example. Two commands were executed by the user:

- **SparkExample** - A script to invoke the SPARK Examiner with the appropriate arguments and files for this example.
- **SPADEase <subprogram name>** - A script to invoke every phase of SPADEase. The single argument is the name of the subprogram being analysed. It is assumed this is in the current directory[2].

Four (out of six remaining) VCs are proved. The two undischarged VCs are not successfully planed due to very limited support for the `div` operator. This is certainly an area for refinement. This note only repeats a small fraction of the data collected during planning - the original version of this note numbered 500 pages. Only the most pertinent data is listed here.

[1] A couple of weeks later than anticipated, but delightful all the same.

[2] SPADEase includes a wide range of arguments supporting execution in any directory from any directory. However, the current platform (a combination of Windows and Cygwin) and tools (some in Poplog-Prolog and some in Sicstus) require aggressive quoting of path names to make this work. This is not important and simply avoided.

2.2 Code

```
package Array_Proofs
is

   type Numeric_Index is range 1 .. 10;
   --# assert Numeric_Index'Base is Integer;
   type Enumerated_Index is (Literal_A, Literal_B,
                             Literal_C, Literal_D, Literal_E);

   type Component is range -1000 .. 1000;

   type My_Integer is range -1E10 .. 1E10;

   type Array_N is array(Numeric_Index) of Component;
   type Array_E is array(Enumerated_Index) of Component;

   procedure P1 (Data    : in     Array_N;
                 Sum     :    out My_Integer;
                 Average :    out Component);
   --# derives Sum,
   --#         Average from
   --#            Data;

   procedure P2 (Data    : in     Array_N;
                 Sum     :    out My_Integer;
                 Average :    out Component);
   --# derives Sum,
   --#         Average from
   --#            Data;

   procedure P3 (Data    : in     Array_E;
                 Sum     :    out My_Integer;
                 Average :    out Component);
   --# derives Sum,
   --#         Average from
   --#            Data;

   procedure P4 (Data    : in     Array_E;
                 Sum     :    out My_Integer;
                 Average :    out Component);
   --# derives Sum,
   --#         Average from
   --#            Data;

end Array_Proofs;


package body Array_Proofs
is

   procedure P1 (Data    : in     Array_N;
                 Sum     :    out My_Integer;
                 Average :    out Component)
   is
   begin
      Sum := 0;
      for I in Numeric_Index loop
      --# assert Sum >= My_Integer(I - Numeric_Index'First) *
      --#                                  My_Integer(Component'First) and
```

```
   --#        Sum <= My_Integer(I - Numeric_Index'First) *
   --#                                 My_Integer(Component'Last) ;
      Sum := Sum + My_Integer(Data(I));
   end loop;
   Average := Component(Sum /
                My_Integer((Numeric_Index'Last - Numeric_Index'First) + 1));
end P1;

procedure P2 (Data    : in     Array_N;
              Sum     :    out My_Integer;
              Average :    out Component)
is
begin
   Sum := 0;
   for I in Numeric_Index loop
      Sum := Sum + My_Integer(Data(I));
   --# assert Sum >= My_Integer(I - Numeric_Index'First + 1) *
   --#                                 My_Integer(Component'First) and
   --#        Sum <= My_Integer(I - Numeric_Index'First + 1) *
   --#                                 My_Integer(Component'Last) ;
   end loop;
   Average := Component(Sum /
                My_Integer((Numeric_Index'Last - Numeric_Index'First) + 1));
end P2;

procedure P3 (Data    : in     Array_E;
              Sum     :    out My_Integer;
              Average :    out Component)
is
begin
   Sum := 0;
   for I in Enumerated_Index loop
   --# assert Sum >= My_Integer(Enumerated_Index'Pos(I)) * My_Integer(
       Component'First) and
   --#        Sum <= My_Integer(Enumerated_Index'Pos(I)) * My_Integer(
       Component'Last) ;
      Sum := Sum + My_Integer(Data(I));
   end loop;
   Average := Component(Sum /
                (My_Integer(Enumerated_Index'Pos(Enumerated_Index'Last) + 1))
       );
end P3;


procedure P4 (Data    : in     Array_E;
              Sum     :    out My_Integer;
              Average :    out Component)
is
begin
   Sum := 0;
   for I in Enumerated_Index loop
      Sum := Sum + My_Integer(Data(I));
   --# assert Sum >= My_Integer(Enumerated_Index'Pos(I)+1) * My_Integer(
       Component'First) and
   --#        Sum <= My_Integer(Enumerated_Index'Pos(I)+1) * My_Integer(
       Component'Last) ;
   end loop;
   Average := Component(Sum /
                (My_Integer(Enumerated_Index'Pos(Enumerated_Index'Last) + 1))
```

```
      );
   end P4;

end Array_Proofs;
```

2.3 SPARK Examiner

```
          *******************************************************
     SPARK95 Examiner with VC and RTC Generator Release 7.2XX / 12.04
               Praxis High Integrity Systems, Bath, England
          *******************************************************


                        DATE : 06-JUL-2005 15:57:48.23


           Reading target configuration file ...

           Examining the specification of package Array_Proofs ...

           Generating listing file array_proofs.lst ...

           Examining the body of package Array_Proofs ...

+++        Flow analysis of subprogram P1 performed: no
           errors found.

           Building model of subprogram ...

           Generating VCs ...

           Writing VCs ...

+++        Flow analysis of subprogram P2 performed: no
           errors found.

           Building model of subprogram ...

           Generating VCs ...

           Writing VCs ...

+++        Flow analysis of subprogram P3 performed: no
           errors found.

           Building model of subprogram ...

           Generating VCs ...

           Writing VCs ...

+++        Flow analysis of subprogram P4 performed: no
           errors found.

           Building model of subprogram ...

           Generating VCs ...
```

```
          Writing VCs ...

          Generating listing file array_proofs.lst ...

          Generating report file ...


-----------End of SPARK Examination--------------------------------
```

2.3.1 VCG file

```
         *******************************************************
                      Semantic Analysis of SPARK Text
      SPARK95 Examiner with VC and RTC Generator Release 7.2XX / 12.04
                  Praxis High Integrity Systems, Bath, England
         *******************************************************


                       DATE : 06-JUL-2005 15:58:03.92

                         procedure Array_Proofs.P1




For path(s) from start to run-time check associated with statement of line 9:

procedure_p1_1.
H1:    true .
H2:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
           component__first) and (element(data, [i___1]) <=
           component__last))) .
        ->
C1:    0 >= my_integer__first .
C2:    0 <= my_integer__last .


For path(s) from start to assertion of line 11:

procedure_p1_2.
H1:    true .
H2:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
           component__first) and (element(data, [i___1]) <=
           component__last))) .
H3:    0 >= my_integer__first .
H4:    0 <= my_integer__last .
        ->
C1:    0 >= (numeric_index__first - numeric_index__first) *
           component__first .
C2:    0 <= (numeric_index__first - numeric_index__first) *
           component__last .
C3:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
```

```
           component__first) and (element(data, [i___1]) <=
           component__last))) .
C4:    numeric_index__first >= numeric_index__first .
C5:    numeric_index__first <= numeric_index__last .
C6:    numeric_index__first <= numeric_index__last .
```

For path(s) from assertion of line 11 to assertion of line 11:

```
procedure_p1_3.
H1:    sum >= (loop__1__i - numeric_index__first) *
           component__first .
H2:    sum <= (loop__1__i - numeric_index__first) *
           component__last .
H3:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
           component__first) and (element(data, [i___1]) <=
           component__last))) .
H4:    loop__1__i >= numeric_index__first .
H5:    loop__1__i <= numeric_index__last .
H6:    loop__1__i <= numeric_index__last .
H7:    sum >= my_integer__first .
H8:    sum <= my_integer__last .
H9:    sum + element(data, [loop__1__i]) >= my_integer__first .
H10:   sum + element(data, [loop__1__i]) <= my_integer__last .
H11:   element(data, [loop__1__i]) >= my_integer__first .
H12:   element(data, [loop__1__i]) <= my_integer__last .
H13:   loop__1__i >= numeric_index__first .
H14:   loop__1__i <= numeric_index__last .
H15:   not (loop__1__i = numeric_index__last) .
        ->
C1:    sum + element(data, [loop__1__i]) >= (loop__1__i + 1 -
           numeric_index__first) * component__first .
C2:    sum + element(data, [loop__1__i]) <= (loop__1__i + 1 -
           numeric_index__first) * component__last .
C3:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
           component__first) and (element(data, [i___1]) <=
           component__last))) .
C4:    loop__1__i + 1 >= numeric_index__first .
C5:    loop__1__i + 1 <= numeric_index__last .
C6:    loop__1__i + 1 <= numeric_index__last .
```

For path(s) from assertion of line 11 to run-time check associated with
          statement of line 15:

```
procedure_p1_4.
H1:    sum >= (loop__1__i - numeric_index__first) *
           component__first .
H2:    sum <= (loop__1__i - numeric_index__first) *
           component__last .
H3:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
           component__first) and (element(data, [i___1]) <=
           component__last))) .
```

```
H4:    loop__1__i >= numeric_index__first .
H5:    loop__1__i <= numeric_index__last .
H6:    loop__1__i <= numeric_index__last .
H7:    sum >= my_integer__first .
H8:    sum <= my_integer__last .
        ->
C1:    sum + element(data, [loop__1__i]) >= my_integer__first .
C2:    sum + element(data, [loop__1__i]) <= my_integer__last .
C3:    element(data, [loop__1__i]) >= my_integer__first .
C4:    element(data, [loop__1__i]) <= my_integer__last .
C5:    loop__1__i >= numeric_index__first .
C6:    loop__1__i <= numeric_index__last .


For path(s) from assertion of line 11 to run-time check associated with
          statement of line 17:

procedure_p1_5.
H1:    sum >= (loop__1__i - numeric_index__first) *
           component__first .
H2:    sum <= (loop__1__i - numeric_index__first) *
           component__last .
H3:    for_all(i___1: numeric_index, ((i___1 >=
           numeric_index__first) and (i___1 <=
           numeric_index__last)) -> ((element(data, [i___1]) >=
           component__first) and (element(data, [i___1]) <=
           component__last))) .
H4:    loop__1__i >= numeric_index__first .
H5:    loop__1__i <= numeric_index__last .
H6:    loop__1__i <= numeric_index__last .
H7:    sum >= my_integer__first .
H8:    sum <= my_integer__last .
H9:    sum + element(data, [loop__1__i]) >= my_integer__first .
H10:   sum + element(data, [loop__1__i]) <= my_integer__last .
H11:   element(data, [loop__1__i]) >= my_integer__first .
H12:   element(data, [loop__1__i]) <= my_integer__last .
H13:   loop__1__i >= numeric_index__first .
H14:   loop__1__i <= numeric_index__last .
H15:   loop__1__i = numeric_index__last .
H16:   sum + element(data, [loop__1__i]) >= my_integer__first .
H17:   sum + element(data, [loop__1__i]) <= my_integer__last .
        ->
C1:    (sum + element(data, [loop__1__i])) div (
           numeric_index__last - numeric_index__first + 1) >=
           component__first .
C2:    (sum + element(data, [loop__1__i])) div (
           numeric_index__last - numeric_index__first + 1) <=
           component__last .
C3:    (sum + element(data, [loop__1__i])) div (
           numeric_index__last - numeric_index__first + 1) >=
           component__first .
C4:    (sum + element(data, [loop__1__i])) div (
           numeric_index__last - numeric_index__first + 1) <=
           component__last .
C5:    (sum + element(data, [loop__1__i])) div (
           numeric_index__last - numeric_index__first + 1) >=
           my_integer__base__first .
C6:    (sum + element(data, [loop__1__i])) div (
           numeric_index__last - numeric_index__first + 1) <=
```

```
           my_integer__base__last .
C7:    numeric_index__last - numeric_index__first + 1 <> 0 .
C8:    numeric_index__last - numeric_index__first + 1 >=
           my_integer__first .
C9:    numeric_index__last - numeric_index__first + 1 <=
           my_integer__last .
C10:   numeric_index__last - numeric_index__first + 1 >=
           numeric_index__base__first .
C11:   numeric_index__last - numeric_index__first + 1 <=
           numeric_index__base__last .
C12:   numeric_index__last - numeric_index__first >=
           numeric_index__base__first .
C13:   numeric_index__last - numeric_index__first <=
           numeric_index__base__last .


For path(s) from assertion of line 11 to finish:

procedure_p1_6.
*** true .          /* trivially true VC removed by Examiner */
```

2.4 SPADE Simplifier

2.4.1 SIV file

```
          *******************************************************
                      Semantic Analysis of SPARK Text
      SPARK95 Examiner with VC and RTC Generator Release 7.2XX / 12.04
                Praxis High Integrity Systems, Bath, England
          *******************************************************


      CREATED 06-JUL-2005, 15:58:03  SIMPLIFIED 06-JUL-2005, 15:58:44
               (Simplified by SPADE Simplifier, Version 2.18)

                         procedure Array_Proofs.P1




For path(s) from start to run-time check associated with statement of line 9:

procedure_p1_1.
*** true .          /* all conclusions proved */


For path(s) from start to assertion of line 11:

procedure_p1_2.
*** true .          /* all conclusions proved */


For path(s) from assertion of line 11 to assertion of line 11:

procedure_p1_3.
H1:    sum >= (loop__1__i - 1) * - 1000 .
H2:    sum <= (loop__1__i - 1) * 1000 .
H3:    for_all(i___1 : numeric_index, 1 <= i___1 and i___1 <= 10 -> - 1000 <=
```

```
          element(data, [i___1]) and element(data, [i___1]) <= 1000) .
H4:    loop__1__i < 10 .
H5:    sum + element(data, [loop__1__i]) >= - 10000000000 .
H6:    sum + element(data, [loop__1__i]) <= 10000000000 .
H7:    element(data, [loop__1__i]) >= - 10000000000 .
H8:    element(data, [loop__1__i]) <= 10000000000 .
       ->
C1:    sum + element(data, [loop__1__i]) >= loop__1__i * - 1000 .
C2:    sum + element(data, [loop__1__i]) <= loop__1__i * 1000 .


For path(s) from assertion of line 11 to run-time check associated with
          statement of line 15:

procedure_p1_4.
H1:    sum >= (loop__1__i - 1) * - 1000 .
H2:    sum <= (loop__1__i - 1) * 1000 .
H3:    for_all(i___1 : numeric_index, 1 <= i___1 and i___1 <= 10 -> - 1000 <=
          element(data, [i___1]) and element(data, [i___1]) <= 1000) .
H4:    loop__1__i <= 10 .
       ->
C1:    sum + element(data, [loop__1__i]) >= - 10000000000 .
C2:    sum + element(data, [loop__1__i]) <= 10000000000 .


For path(s) from assertion of line 11 to run-time check associated with
          statement of line 17:

procedure_p1_5.
H1:    sum >= - 9000 .
H2:    sum <= 9000 .
H3:    for_all(i___1 : numeric_index, 1 <= i___1 and i___1 <= 10 -> - 1000 <=
          element(data, [i___1]) and element(data, [i___1]) <= 1000) .
H4:    sum >= - 10000000000 .
H5:    sum <= 10000000000 .
H6:    sum + element(data, [10]) >= - 10000000000 .
H7:    sum + element(data, [10]) <= 10000000000 .
H8:    element(data, [10]) >= - 10000000000 .
H9:    element(data, [10]) <= 10000000000 .
       ->
C5:    (sum + element(data, [10])) div 10 >= my_integer__base__first .
C6:    (sum + element(data, [10])) div 10 <= my_integer__base__last .


For path(s) from assertion of line 11 to finish:

procedure_p1_6.
*** true .          /* all conclusions proved */
```

2.5 SPADEase-Parser

```
==================================================
SPADEase-Parser (version 1.0)
```

```
CLAM beget NuSPADE beget SPADEase

CLAM     - Developed at Edinburgh University and Heriot-Watt University
NuSPADE  - EPSRC grant GR/R24081
SPADEase - EPSRC grant GR/T11289/01

Bill J Ellis, Andrew Ireland, Andrew Cook
==================================================

(For help, provide argument: --help)

Retrieving VCG...ok
Retrieving SIV...ok
Placing loaded VCG and/or SIV on parser output...ok
Retrieving FDL...ok
Placing loaded FDL on parser output...ok
Retrieving RLS...ok
Retrieving RUL (built in)...ok
Retrieving RUL (user)...ok
Create theorem deitals from any loaded rules...ok
Create rewrite rules for any theorem detials...ok
Create wave rules for any theorem detials...
##################################################
#Creating wave rules (*=cache hit)
##################################################

*******************************************************************************************************
          ##############################################\/##
#Final save of the wave rules in memory...
#                                      ...ok
#############################################/\##

ok
Create attraction rules for any theorem detials...
##################################################
#Creating attraction rules (*=cache hit)
##################################################

*******************************************************************************************************
          ##############################################\/##
#Final save of the attraction rules in memory...
#                                            ...ok
#############################################/\##

ok
Placing loaded rules on parser output...ok
Retrieving plan data...ok
Placing plan components on parser output...ok
Placing plan strategy on parser output...ok
Saving all data on parser output...ok
Saving pages for debug...ok
Result: Success!
```

2.6 SPADEase-Planner

2.7 SPADEase-Verifier

2.7.1 Subprogram: p1 VC: 3 Conc: 1

```
consult '.\\p1.rls'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\minus_plus.rul'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\distribute.rul'.
tame_subgoal_on_conc 1.
    tame_rewrite hyp : sum>=(loop__1__i-numeric_index__first)*component__first
          : [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum>=(loop__1__i-1)*component__first : [2,2] with
          component__first to -1000 if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-numeric_index__first)*component__last :
          [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-1)*component__last : [2,2] with
          component__last to 1000 if true using p1_rules(8) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=numeric_index__first
          and i___1<=numeric_index__last->element(data,[i___1])
          >=component__first and element(data,[i___1])<=component__last) : [
          2,1,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=numeric_index__last->element(data,[i___1])>=component__first
          and element(data,[i___1])<=component__last) : [2,2,1,2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>=component__first and element(data,[
          i___1])<=component__last) : [2,1,2,2] with component__first to -1000
          if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
          <=component__last) : [2,2,2,2] with component__last to 1000 if true
          using p1_rules(8) in lefttoright.
    tame_rewrite hyp : loop__1__i>=numeric_index__first : [2] with
          numeric_index__first to 1 if true using p1_rules(3) in lefttoright.
    tame_rewrite hyp : loop__1__i<=numeric_index__last : [2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : sum>=my_integer__first : [2] with my_integer__first to
          -10000000000 if true using p1_rules(12) in lefttoright.
    tame_rewrite hyp : sum<=my_integer__last : [2] with my_integer__last to
          10000000000 if true using p1_rules(13) in lefttoright.
    tame_rewrite hyp : sum+element(data,[loop__1__i])>=my_integer__first : [2]
          with my_integer__first to -10000000000 if true using p1_rules(12) in
          lefttoright.
    tame_rewrite hyp : sum+element(data,[loop__1__i])<=my_integer__last : [2]
          with my_integer__last to 10000000000 if true using p1_rules(13) in
          lefttoright.
    tame_rewrite hyp : element(data,[loop__1__i])>=my_integer__first : [2] with
          my_integer__first to -10000000000 if true using p1_rules(12) in
          lefttoright.
    tame_rewrite hyp : element(data,[loop__1__i])<=my_integer__last : [2] with
          my_integer__last to 10000000000 if true using p1_rules(13) in
          lefttoright.
    tame_rewrite hyp : not loop__1__i=numeric_index__last : [2,1] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])>=(
```

```
          loop__1__i+1-numeric_index__first)*component__first : [2,1,2] with
          numeric_index__first to 1 if true using p1_rules(3) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])>=(loop__1__i+1-1)
          *component__first : [2,2] with component__first to -1000 if true
          using p1_rules(7) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])>=(loop__1__i+1-1)* -1000
          : [1,2] with loop__1__i+1-1 to loop__1__i-1+1 if true using
          minus_plus(1) in righttoleft.
    tame_rewrite conc : sum+element(data,[loop__1__i])>=(loop__1__i-1+1)* -1000
          : [2] with (loop__1__i-1+1)* -1000 to (loop__1__i-1)* -1000+1* -1000
          if true using distribute(1) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])>=(loop__1__i-1)*
          -1000+1* -1000 : [] with sum+element(data,[loop__1__i])>=(
          loop__1__i-1)* -1000+1* -1000 to sum>=(loop__1__i-1)* -1000 and
          element(data,[loop__1__i])>=1* -1000 if true using inequals(77) in
          righttoleft.
    tame_rewrite conc : sum>=(loop__1__i-1)* -1000 and element(data,[loop__1__i]
          )>=1* -1000 : [1] where sum>=(loop__1__i-1)* -1000.
    tame_subgoal_on_term true.
        tame_done.
    tame_all_done.
    tame_subgoal_on_term element(data,[loop__1__i])>=1* -1000.
        tame_rewrite conc : element(data,[loop__1__i])>=1* -1000 : [2] with 1*
          -1000 is -1000.
        tame_unwrap hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
          <=1000).
        instantiate loop__1__i.
        tame_subgoal_on_term loop__1__i>=1 and loop__1__i<=10.
            tame_rewrite conc : loop__1__i>=1 and loop__1__i<=10 : [1] where
          loop__1__i>=1.
            tame_rewrite conc : true and loop__1__i<=10 : [2] where
          loop__1__i<=10.
            tame_subgoal_on_term true.
                tame_done.
            tame_all_done.
            tame_subgoal_on_term true.
                tame_done.
            tame_all_done.
            tame_done.
        tame_all_done.
        tame_forwardchain loop__1__i>=1 and loop__1__i<=10->element(data,[
          loop__1__i])>= -1000 and element(data,[loop__1__i])<=1000.
        tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
            simplify.
            tame_done.
        tame_all_done.
        tame_subgoal_on_term element(data,[loop__1__i])<=1000.
            simplify.
            tame_done.
        tame_all_done.
        tame_rewrite conc : element(data,[loop__1__i])>= -1000 : [] where
          element(data,[loop__1__i])>= -1000.
        tame_done.
    tame_all_done.
    tame_done.
tame_all_done.
tame_done.
tame_finish.
```

2.7.2 Subprogram: p1 VC: 3 Conc: 2

```
consult '.\\p1.rls'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\minus_plus.rul'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\distribute.rul'.
tame_subgoal_on_conc 2.
    tame_rewrite hyp : sum>=(loop__1__i-numeric_index__first)*component__first
          : [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum>=(loop__1__i-1)*component__first : [2,2] with
          component__first to -1000 if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-numeric_index__first)*component__last :
          [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-1)*component__last : [2,2] with
          component__last to 1000 if true using p1_rules(8) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=numeric_index__first
          and i___1<=numeric_index__last->element(data,[i___1])
          >=component__first and element(data,[i___1])<=component__last) : [
          2,1,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=numeric_index__last->element(data,[i___1])>=component__first
          and element(data,[i___1])<=component__last) : [2,2,1,2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>=component__first and element(data,[
          i___1])<=component__last) : [2,1,2,2] with component__first to -1000
          if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
          <=component__last) : [2,2,2,2] with component__last to 1000 if true
          using p1_rules(8) in lefttoright.
    tame_rewrite hyp : loop__1__i>=numeric_index__first : [2] with
          numeric_index__first to 1 if true using p1_rules(3) in lefttoright.
    tame_rewrite hyp : loop__1__i<=numeric_index__last : [2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : sum>=my_integer__first : [2] with my_integer__first to
          -10000000000 if true using p1_rules(12) in lefttoright.
    tame_rewrite hyp : sum<=my_integer__last : [2] with my_integer__last to
          10000000000 if true using p1_rules(13) in lefttoright.
    tame_rewrite hyp : sum+element(data,[loop__1__i])>=my_integer__first : [2]
          with my_integer__first to -10000000000 if true using p1_rules(12) in
          lefttoright.
    tame_rewrite hyp : sum+element(data,[loop__1__i])<=my_integer__last : [2]
          with my_integer__last to 10000000000 if true using p1_rules(13) in
          lefttoright.
    tame_rewrite hyp : element(data,[loop__1__i])>=my_integer__first : [2] with
          my_integer__first to -10000000000 if true using p1_rules(12) in
          lefttoright.
    tame_rewrite hyp : element(data,[loop__1__i])<=my_integer__last : [2] with
          my_integer__last to 10000000000 if true using p1_rules(13) in
          lefttoright.
    tame_rewrite hyp : not loop__1__i=numeric_index__last : [2,1] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])<=(
          loop__1__i+1-numeric_index__first)*component__last : [2,1,2] with
```

```
          numeric_index__first to 1 if true using p1_rules(3) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])<=(loop__1__i+1-1)
          *component__last : [2,2] with component__last to 1000 if true using
          p1_rules(8) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])<=(loop__1__i+1-1)*1000 :
          [1,2] with loop__1__i+1-1 to loop__1__i-1+1 if true using minus_plus(
          1) in righttoleft.
    tame_rewrite conc : sum+element(data,[loop__1__i])<=(loop__1__i-1+1)*1000 :
          [2] with (loop__1__i-1+1)*1000 to (loop__1__i-1)*1000+1*1000 if true
          using distribute(1) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])<=(loop__1__i-1)
          *1000+1*1000 : [] with sum+element(data,[loop__1__i])<=(loop__1__i-1)
          *1000+1*1000 to sum<=(loop__1__i-1)*1000 and element(data,[loop__1__i]
          )<=1*1000 if true using inequals(80) in righttoleft.
    tame_rewrite conc : sum<=(loop__1__i-1)*1000 and element(data,[loop__1__i])
          <=1*1000 : [1] where sum<=(loop__1__i-1)*1000.
    tame_subgoal_on_term true.
        tame_done.
    tame_all_done.
    tame_subgoal_on_term element(data,[loop__1__i])<=1*1000.
        tame_rewrite conc : element(data,[loop__1__i])<=1*1000 : [2] with
          1*1000 is 1000.
        tame_unwrap hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
          <=1000).
        instantiate loop__1__i.
        tame_subgoal_on_term loop__1__i>=1 and loop__1__i<=10.
            tame_rewrite conc : loop__1__i>=1 and loop__1__i<=10 : [1] where
          loop__1__i>=1.
            tame_rewrite conc : true and loop__1__i<=10 : [2] where
          loop__1__i<=10.
            tame_subgoal_on_term true.
                tame_done.
            tame_all_done.
            tame_subgoal_on_term true.
                tame_done.
            tame_all_done.
            tame_done.
        tame_all_done.
        tame_forwardchain loop__1__i>=1 and loop__1__i<=10->element(data,[
          loop__1__i])>= -1000 and element(data,[loop__1__i])<=1000.
        tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
            simplify.
            tame_done.
        tame_all_done.
        tame_subgoal_on_term element(data,[loop__1__i])<=1000.
            simplify.
            tame_done.
        tame_all_done.
        tame_rewrite conc : element(data,[loop__1__i])<=1000 : [] where element(
          data,[loop__1__i])<=1000.
        tame_done.
    tame_all_done.
    tame_done.
tame_all_done.
tame_done.
tame_finish.
```

2.7.3 Subprogram: p1 VC: 4 Conc: 1

```
consult '.\\p1.rls'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\rotate.rul'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\minus_plus.rul'.
tame_subgoal_on_conc 1.
    tame_rewrite hyp : sum>=(loop__1__i-numeric_index__first)*component__first
          : [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum>=(loop__1__i-1)*component__first : [2,2] with
          component__first to -1000 if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-numeric_index__first)*component__last :
          [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-1)*component__last : [2,2] with
          component__last to 1000 if true using p1_rules(8) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=numeric_index__first
          and i___1<=numeric_index__last->element(data,[i___1])
          >=component__first and element(data,[i___1])<=component__last) : [
          2,1,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=numeric_index__last->element(data,[i___1])>=component__first
          and element(data,[i___1])<=component__last) : [2,2,1,2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>=component__first and element(data,[
          i___1])<=component__last) : [2,1,2,2] with component__first to -1000
          if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
          <=component__last) : [2,2,2,2] with component__last to 1000 if true
          using p1_rules(8) in lefttoright.
    tame_rewrite hyp : loop__1__i>=numeric_index__first : [2] with
          numeric_index__first to 1 if true using p1_rules(3) in lefttoright.
    tame_rewrite hyp : loop__1__i<=numeric_index__last : [2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : sum>=my_integer__first : [2] with my_integer__first to
          -10000000000 if true using p1_rules(12) in lefttoright.
    tame_rewrite hyp : sum<=my_integer__last : [2] with my_integer__last to
          10000000000 if true using p1_rules(13) in lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])>=my_integer__first : [2]
          with my_integer__first to -10000000000 if true using p1_rules(12) in
          lefttoright.
    tame_rewrite conc : sum+element(data,[loop__1__i])>= -10000000000 : [] with
          sum+element(data,[loop__1__i])>= -10000000000 to -10000000000<=
          -1000+(loop__1__i-1)* -1000 and-1000+(loop__1__i-1)*
          -1000<=sum+element(data,[loop__1__i]) if true using transitivity(10)
          in righttoleft.
    tame_rewrite conc : -10000000000<= -1000+(loop__1__i-1)* -1000 and-1000+(
          loop__1__i-1)* -1000<=sum+element(data,[loop__1__i]) : [2] with
          -1000+(loop__1__i-1)* -1000<=sum+element(data,[loop__1__i]) to
          sum+element(data,[loop__1__i])>= -1000+(loop__1__i-1)* -1000 if true
          using rotate(2) in righttoleft.
    tame_rewrite conc : -10000000000<= -1000+(loop__1__i-1)* -1000 and
          sum+element(data,[loop__1__i])>= -1000+(loop__1__i-1)* -1000 : [2]
          with sum+element(data,[loop__1__i])>= -1000+(loop__1__i-1)* -1000 to
          sum>=(loop__1__i-1)* -1000 and element(data,[loop__1__i])>= -1000 if
          true using inequals(78) in righttoleft.
```

```
tame_rewrite conc : -10000000000<= -1000+(loop__1__i-1)* -1000 and(sum>=(
      loop__1__i-1)* -1000 and element(data,[loop__1__i])>= -1000) : [1,2]
      where sum>=(loop__1__i-1)* -1000.
tame_unwrap hyp : for_all(i___1:numeric_index,i___1>=1 and
      i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
      <=1000).
instantiate loop__1__i.
tame_rewrite conc : -10000000000<= -1000+(loop__1__i-1)* -1000 and(true and
      element(data,[loop__1__i])>= -1000) : [1] with -10000000000<= -1000+(
      loop__1__i-1)* -1000 to -1000+(loop__1__i-1)* -1000>= -10000000000 if
      true using rotate(2) in righttoleft.
tame_subgoal_on_term -1000+(loop__1__i-1)* -1000>= -10000000000.
    tame_subgoal_on_term loop__1__i>=1 and loop__1__i<=10.
        tame_rewrite conc : loop__1__i>=1 and loop__1__i<=10 : [1] where
      loop__1__i>=1.
        tame_rewrite conc : true and loop__1__i<=10 : [2] where
      loop__1__i<=10.
        tame_subgoal_on_term true.
            tame_done.
        tame_all_done.
        tame_subgoal_on_term true.
            tame_done.
        tame_all_done.
        tame_done.
    tame_all_done.
    tame_forwardchain loop__1__i>=1 and loop__1__i<=10->element(data,[
      loop__1__i])>= -1000 and element(data,[loop__1__i])<=1000.
    tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
        simplify.
        tame_done.
    tame_all_done.
    tame_subgoal_on_term element(data,[loop__1__i])<=1000.
        simplify.
        tame_done.
    tame_all_done.
    tame_rewrite conc : -1000+(loop__1__i-1)* -1000>= -10000000000 : [2,1]
      with (loop__1__i-1)* -1000 to -1000*loop__1__i- -1000*1 if true using
      distrib(4) in lefttoright.
    tame_rewrite conc : -1000+(-1000*loop__1__i- -1000*1)>= -10000000000 : [
      2,2,1] with -1000*1 is -1000.
    tame_rewrite conc : -1000+(-1000*loop__1__i- -1000)>= -10000000000 : [1]
       with -1000+(-1000*loop__1__i- -1000) to -1000- -1000+
      -1000*loop__1__i if true using minus_plus(2) in lefttoright.
    tame_rewrite conc : -1000- -1000+ -1000*loop__1__i>= -10000000000 : [
      1,1] with -1000- -1000 is 0.
    tame_rewrite conc : 0+ -1000*loop__1__i>= -10000000000 : [1] with 0+
      -1000*loop__1__i to -1000*loop__1__i if true using arith(4) in
      lefttoright.
    tame_done.
tame_all_done.
tame_subgoal_on_term true and element(data,[loop__1__i])>= -1000.
    tame_subgoal_on_term true.
        tame_done.
    tame_all_done.
    tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
        tame_subgoal_on_term loop__1__i>=1 and loop__1__i<=10.
            tame_rewrite conc : loop__1__i>=1 and loop__1__i<=10 : [1]
      where loop__1__i>=1.
            tame_rewrite conc : true and loop__1__i<=10 : [2] where
```

```
          loop__1__i<=10.
                tame_subgoal_on_term true.
                    tame_done.
                tame_all_done.
                tame_subgoal_on_term true.
                    tame_done.
                tame_all_done.
                tame_done.
            tame_all_done.
            tame_forwardchain loop__1__i>=1 and loop__1__i<=10->element(data,[
          loop__1__i])>= -1000 and element(data,[loop__1__i])<=1000.
            tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
                simplify.
                tame_done.
            tame_all_done.
            tame_subgoal_on_term element(data,[loop__1__i])<=1000.
                simplify.
                tame_done.
            tame_all_done.
            tame_rewrite conc : element(data,[loop__1__i])>= -1000 : [] where
          element(data,[loop__1__i])>= -1000.
            tame_done.
        tame_all_done.
        tame_done.
    tame_all_done.
    tame_done.
tame_all_done.
tame_done.
tame_finish.
```

2.7.4 Subprogram: p1 VC: 4 Conc: 2

```
consult '.\\p1.rls'.
consult 'U:\\Bill Ellis\\DATA\\Ease/data/user-rules\\minus_plus.rul'.
tame_subgoal_on_conc 2.
    tame_rewrite hyp : sum>=(loop__1__i-numeric_index__first)*component__first
          : [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum>=(loop__1__i-1)*component__first : [2,2] with
          component__first to -1000 if true using p1_rules(7) in lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-numeric_index__first)*component__last :
          [2,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : sum<=(loop__1__i-1)*component__last : [2,2] with
          component__last to 1000 if true using p1_rules(8) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=numeric_index__first
          and i___1<=numeric_index__last->element(data,[i___1])
          >=component__first and element(data,[i___1])<=component__last) : [
          2,1,1,2] with numeric_index__first to 1 if true using p1_rules(3) in
          lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=numeric_index__last->element(data,[i___1])>=component__first
          and element(data,[i___1])<=component__last) : [2,2,1,2] with
          numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
    tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
          i___1<=10->element(data,[i___1])>=component__first and element(data,[
          i___1])<=component__last) : [2,1,2,2] with component__first to -1000
          if true using p1_rules(7) in lefttoright.
```

```
tame_rewrite hyp : for_all(i___1:numeric_index,i___1>=1 and
      i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
      <=component__last) : [2,2,2,2] with component__last to 1000 if true
      using p1_rules(8) in lefttoright.
tame_rewrite hyp : loop__1__i>=numeric_index__first : [2] with
      numeric_index__first to 1 if true using p1_rules(3) in lefttoright.
tame_rewrite hyp : loop__1__i<=numeric_index__last : [2] with
      numeric_index__last to 10 if true using p1_rules(4) in lefttoright.
tame_rewrite hyp : sum>=my_integer__first : [2] with my_integer__first to
      -10000000000 if true using p1_rules(12) in lefttoright.
tame_rewrite hyp : sum<=my_integer__last : [2] with my_integer__last to
      10000000000 if true using p1_rules(13) in lefttoright.
tame_rewrite conc : sum+element(data,[loop__1__i])<=my_integer__last : [2]
      with my_integer__last to 10000000000 if true using p1_rules(13) in
      lefttoright.
tame_rewrite conc : sum+element(data,[loop__1__i])<=10000000000 : [] with
      sum+element(data,[loop__1__i])<=10000000000 to sum+element(data,[
      loop__1__i])<=1000+(loop__1__i-1)*1000 and 1000+(loop__1__i-1)
      *1000<=10000000000 if true using transitivity(1) in righttoleft.
tame_rewrite conc : sum+element(data,[loop__1__i])<=1000+(loop__1__i-1)
      *1000 and 1000+(loop__1__i-1)*1000<=10000000000 : [1] with
      sum+element(data,[loop__1__i])<=1000+(loop__1__i-1)*1000 to sum<=(
      loop__1__i-1)*1000 and element(data,[loop__1__i])<=1000 if true using
      inequals(81) in righttoleft.
tame_rewrite conc : sum<=(loop__1__i-1)*1000 and element(data,[loop__1__i])
      <=1000 and 1000+(loop__1__i-1)*1000<=10000000000 : [1,1] where sum<=(
      loop__1__i-1)*1000.
tame_unwrap hyp : for_all(i___1:numeric_index,i___1>=1 and
      i___1<=10->element(data,[i___1])>= -1000 and element(data,[i___1])
      <=1000).
instantiate loop__1__i.
tame_subgoal_on_term true and element(data,[loop__1__i])<=1000.
    tame_subgoal_on_term true.
        tame_done.
    tame_all_done.
    tame_subgoal_on_term element(data,[loop__1__i])<=1000.
        tame_subgoal_on_term loop__1__i>=1 and loop__1__i<=10.
            tame_rewrite conc : loop__1__i>=1 and loop__1__i<=10 : [1]
      where loop__1__i>=1.
            tame_rewrite conc : true and loop__1__i<=10 : [2] where
      loop__1__i<=10.
            tame_subgoal_on_term true.
                tame_done.
            tame_all_done.
            tame_subgoal_on_term true.
                tame_done.
            tame_all_done.
            tame_done.
        tame_all_done.
        tame_forwardchain loop__1__i>=1 and loop__1__i<=10->element(data,[
      loop__1__i])>= -1000 and element(data,[loop__1__i])<=1000.
        tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
            simplify.
            tame_done.
        tame_all_done.
        tame_subgoal_on_term element(data,[loop__1__i])<=1000.
            simplify.
            tame_done.
        tame_all_done.
```

```
          tame_rewrite conc : element(data,[loop__1__i])<=1000 : [] where
        element(data,[loop__1__i])<=1000.
          tame_done.
      tame_all_done.
      tame_done.
  tame_all_done.
  tame_subgoal_on_term 1000+(loop__1__i-1)*1000<=10000000000.
      tame_subgoal_on_term loop__1__i>=1 and loop__1__i<=10.
          tame_rewrite conc : loop__1__i>=1 and loop__1__i<=10 : [1] where
        loop__1__i>=1.
          tame_rewrite conc : true and loop__1__i<=10 : [2] where
        loop__1__i<=10.
          tame_subgoal_on_term true.
              tame_done.
          tame_all_done.
          tame_subgoal_on_term true.
              tame_done.
          tame_all_done.
          tame_done.
      tame_all_done.
      tame_forwardchain loop__1__i>=1 and loop__1__i<=10->element(data,[
        loop__1__i])>= -1000 and element(data,[loop__1__i])<=1000.
      tame_subgoal_on_term element(data,[loop__1__i])>= -1000.
          simplify.
          tame_done.
      tame_all_done.
      tame_subgoal_on_term element(data,[loop__1__i])<=1000.
          simplify.
          tame_done.
      tame_all_done.
      tame_rewrite conc : 1000+(loop__1__i-1)*1000<=10000000000 : [2,1] with (
        loop__1__i-1)*1000 to 1000*loop__1__i-1000*1 if true using distrib(4)
        in lefttoright.
      tame_rewrite conc : 1000+(1000*loop__1__i-1000*1)<=10000000000 : [2,2,1]
         with 1000*1 is 1000.
      tame_rewrite conc : 1000+(1000*loop__1__i-1000)<=10000000000 : [1] with
        1000+(1000*loop__1__i-1000) to 1000-1000+1000*loop__1__i if true
        using minus_plus(2) in lefttoright.
      tame_rewrite conc : 1000-1000+1000*loop__1__i<=10000000000 : [1,1] with
        1000-1000 is 0.
      tame_rewrite conc : 0+1000*loop__1__i<=10000000000 : [1] with
        0+1000*loop__1__i to 1000*loop__1__i if true using arith(4) in
        lefttoright.
      tame_done.
  tame_all_done.
  tame_done.
tame_all_done.
tame_done.
tame_finish.
```

2.8 SPADEase-Finalizer

```
==================================================
SPADEase-Finalizer (ver 1.0)

CLAM beget NuSPADE beget SPADEase
```

```
CLAM     - Developed at Edinburgh University and Heriot-Watt University
NuSPADE  - EPSRC grant GR/R24081
SPADEase - EPSRC grant GR/T11289/01

Bill J Ellis, Andrew Cook, Andew Ireland
==================================================

(For help, provide argument: --help)



--------------------------------------------------
Running external system command:
wrap_utility .\p1.siv
--------------------------------------------------
Saving pages for debug...ok
Result: Success!
```

2.8.1 Final SIV file

```
          *******************************************************
                      Semantic Analysis of SPARK Text
      SPARK95 Examiner with VC and RTC Generator Release 7.2XX / 12.04
                Praxis High Integrity Systems, Bath, England
          *******************************************************


      CREATED 06-JUL-2005, 15:58:03  SIMPLIFIED 06-JUL-2005, 15:58:44
               (Simplified by SPADE Simplifier, Version 2.18)

                         procedure Array_Proofs.P1




For path(s) from start to run-time check associated with statement of line 9:

procedure_p1_1.
*** true .          /* all conclusions proved */


For path(s) from start to assertion of line 11:

procedure_p1_2.
*** true .          /* all conclusions proved */


For path(s) from assertion of line 11 to assertion of line 11:

procedure_p1_3.
*** true .          /* all conclusions proved */


For path(s) from assertion of line 11 to run-time check associated with
          statement of line 15:

procedure_p1_4.
```

```
*** true .          /* all conclusions proved */


For path(s) from assertion of line 11 to run-time check associated with
          statement of line 17:

procedure_p1_5.
H1:    sum >= - 9000 .
H2:    sum <= 9000 .
H3:    for_all(i___1 : numeric_index, 1 <= i___1 and i___1 <= 10 -> - 1000 <=
          element(data, [i___1]) and element(data, [i___1]) <= 1000) .
H4:    sum >= - 10000000000 .
H5:    sum <= 10000000000 .
H6:    sum + element(data, [10]) >= - 10000000000 .
H7:    sum + element(data, [10]) <= 10000000000 .
H8:    element(data, [10]) >= - 10000000000 .
H9:    element(data, [10]) <= 10000000000 .
       ->
C5:    (sum + element(data, [10])) div 10 >= my_integer__base__first .
C6:    (sum + element(data, [10])) div 10 <= my_integer__base__last .


For path(s) from assertion of line 11 to finish:

procedure_p1_6.
*** true .          /* all conclusions proved */
```