```
Script started on Thu Oct 11 07:52:30 2007
demo-astree/programs  %  ./README



**************************************************
**************************************************
***                                            ***
***   Demonstration of the Astree static analyzer    ***
***             http://www.astree.ens.fr/              ***
***                                            ***
*** P. Cousot, R. Cousot, J. Feret, L. Mauborgne,   ***
*** A. Mine, X. Rival                           ***
*** [B. Blanchet (2001/03), D. Monniaux (2001/07)] ***
***                                            ***
**************************************************
**************************************************
%

**************************************************
* Astree is a VERIFIER (not a bug-finder). Hence *
* Astree is SOUND hence reports ALL potential    *
* runtime errors.                                *
**************************************************
%

***  example [CC76]:

% cat -n dichotomy-error.c
     1   /* dichotomy-error.c */
     2   int main () {
     3       int lwb, upb, m, R[100], X;
     4       lwb = 1; upb = 100;
     5       while (lwb <= upb) {
     6          m = (upb + lwb) / 2;
     7          if (X == R[m]) {
     8             upb = m; lwb = m+1; }
     9          else if (X < R[m]) {
    10             upb = m - 1; }
    11          else {
    12             lwb = m + 1; }
    13       }
    14        __ASTREE_log_vars((m));
    15   }
%

***  static analysis by Astree:

% astree --exec-fn main --no-relational --unroll 0 dichotomy-error.c \
   |& egrep --after-context 0 "(launched)|(WARN)"
%

***  (the two errors are reported two times each
***   for the two branches of the conditional.)
%

***  correcting the error:

% cat -n dichotomy.c
     1   /* dichotomy.c */
     2   int main () {
     3       int lwb, upb, m, R[100], X;
     4       lwb = 0; upb = 99;
     5       while (lwb <= upb) {
     6          m = (upb + lwb) / 2;
     7          if (X == R[m]) {
```

```
     8             upb = m; lwb = m+1; }
     9          else if (X < R[m]) {
    10             upb = m - 1; }
    11          else {
    12             lwb = m + 1; }
    13       }
    14        __ASTREE_log_vars((m));
    15    }
%

*** correction (difference with the erroneous version):

% diff dichotomy-error.c dichotomy.c
1c1
< /* dichotomy-error.c */
---
> /* dichotomy.c */
4c4
<     lwb = 1; upb = 100;
---
>     lwb = 0; upb = 99;
%

*** static analysis by Astree:

% astree --exec-fn main --no-relational dichotomy.c \
   |& egrep "(launched)|(m in )|(WARN)"
%

*******************************************************
* Astree is INCOMPLETE hence may report false alarms *
*******************************************************
%

*** example of false alarm:

% cat -n fausse-alarme.c
     1   /* fausse-alarme.c */
     2   void main()
     3   {
     4     int x, y;
     5     if ((-4681 < y) && (y < 4681) && (x < 32767) && (-32767 < x) && ((7*y*y - 1) ==
     6        y = 1 / x;
     7     };
     8   }
%

*** static analysis by Astree:

% astree --exec-fn main fausse-alarme.c |& egrep "(launched)|(WARN)"
%

*********************************************
* Astree tracks all potential buffer overruns *
*********************************************
%

*** example of uninitialization and buffer overrun:

% cat -n bufferoverrun-c.c
     1   #include <stdio.h>
     2   int main () {
     3   int x, y, z, T[9];
     4   x = T[7];
     5   y = T[8];
     6   z = T[9];
     7   printf("x = %i, y = %i, z = %i\n",x,y,z);
```

```
     8   }
     9
%

*** compilation and execution:

% gcc bufferoverrun-c.c
% ./a.out
x = 0, y = 1, z = -1073747596
%

*** static analysis with Astree:

% cat -n bufferoverrun.c
     1   int main () {
     2   int a, x, y, z, T[9];
     3   x = T[7];
     4   y = T[8];
     5   z = T[9];
     6   __ASTREE_log_vars((x,y,z));
     7   }
     8
%

% astree --exec-fn main bufferoverrun.c\
   |& egrep "(x in)|(y in)|(z in)|(WARN)"
%
*** Astree signals the definite error and considers the
*** (unpredictable) execution to be stopped (so no log).
%

************************************************
* Astree tracks all potential dangling pointers *
************************************************
%

*** example of dangling pointer:

% cat -n danglingpointer-c.c
     1   #include <stdio.h>
     2   int main () {
     3   int x, y, z, *r;
     4   x = 100;
     5   r = &x;
     6   y = *r;
     7   z = *(r+2);
     8   printf("x = %i, y = %i, z = %i\n",x,y,z);
     9   }
    10
%

*** compilation and execution:

% gcc danglingpointer-c.c
% ./a.out
x = 100, y = 100, z = -1073747800
%

*** static analysis with Astree:

% cat -n danglingpointer.c
     1   int main () {
     2   int x, y, z, *r;
     3   x = 100;
     4   r = &x;
     5   y = *r;
     6   z = *(r+2);
```

```
     7   __ASTREE_log_vars((x,y,z));
     8   }
     9
%

% astree --exec-fn main danglingpointer.c\
   |& egrep "(x in)|(y in)|(z in)|(WARN)"
%
*** Astree signals the definite error and considers the
*** (unpredictable) execution to be stopped (so no log).
%

*****************************************************
* Astree tracks potential modulo arithmetics errors *
*****************************************************
%

*** Modulo arithmetics is not very intuitive:

% cat -n modulo-c.c
     1   #include <stdio.h>
     2   int main () {
     3   int x,y;
     4   x = -2147483647 / -1;
     5   y = ((-x) -1) / -1;
     6   printf("x = %i, y = %i\n",x,y);
     7   }
     8
%

*** compilation and execution:

% gcc modulo-c.c
% ./a.out
x = 2147483647, y = -2147483648
%
*** -2147483648 / -1 = -2147483648 ???
%

*** static analysis with Astree:

% cat -n modulo.c
     1   int main () {
     2   int x,y;
     3   x = -2147483647 / -1;
     4   y = ((-x) -1) / -1;
     5   __ASTREE_log_vars((x,y));
     6   }
     7
%

% astree --exec-fn main --unroll 0 modulo.c\
   |& egrep -A 1 "(<integers)|(WARN)"
%
*** Astree signals the error and goes on with
*** an unkown value (hence the log)
%


****************************************
* Astree uses interval analysis (enhanced *
* by symbolic execution)                  *
****************************************

*** example:

% cat -n interval.c
```

```
     1   int main () {
     2    int x, y;
     3    __ASTREE_known_fact(((0 <= x) && (x <= 100)));
     4    y = x - x;
     5     __ASTREE_log_vars((x,y));
     6   }
%

*** static analysis by Astree (1 -- WITHOUT symbolic execution):

% astree interval.c --no-relational --exec-fn main \
   |& egrep "(launched)|(x in)|(y in)"
%

*** static analysis by Astree (2 -- WITH symbolic execution):

% astree interval.c --exec-fn main \
   |& egrep "(launched)|(y in)"
%

*** The symbolic abstract domain propagates the
*** symbolic value of variables (plus rounding
*** errors) to perform simplifications.
%

****************************************
* Astree uses weakly relational abstract *
* domains such as octagons...            *
****************************************

***  example:

% cat -n octagon.c
     1   /* octagon.c */
     2   void main()
     3   {
     4     int X, Y, Z;
     5     X = 10;
     6     Y = 100;
     7     while (X >= 0) {
     8       X--;
     9       Y--;
    10     };
    11      __ASTREE_assert((X <= Y));
    12   }
%

*** static analysis by Astree (1 -- WITHOUT octagons):

% astree octagon.c --no-octagon --exec-fn main |& egrep "(launched)|(WARN)"
%

*** static analysis by Astree (2 -- WITH octagons):

% astree octagon.c --exec-fn main |& egrep "(launched)|(WARN)"
%
*** Does not scale up to too many variables,
***   --> packs of variables.
%

******************************************
* Astree uses weakly relational abstract     *
* domains such as boolean decision trees... *
******************************************
%

***  example:
```

```
% cat -n boolean.c
     1   /* boolean.c */
     2   typedef enum {F=0,T=1} BOOL;
     3   BOOL B;
     4   void main () {
     5       unsigned int X, Y;
     6       while (1) {
     7          /* ... */
     8          B = (X == 0);
     9          /* ... */
    10          if (!B) {
    11             Y = 1 / X;
    12          }
    13          /* ... */
    14       }
    15   }
%

*** static analysis by Astree (1 -- **WITHOUT**
***  decision trees):

% astree boolean.c --no-relational --exec-fn main |& egrep "(launched)|(WARN)"
%

*** static analysis by Astree (2 -- **WITH**
***  decision trees):

% astree boolean.c --exec-fn main |& egrep "(launched)|(WARN)"
%

*********************************************
* Astree uses computation trace abstractions *
* (describing  sequences of states) not only *
* invariants (describing sets of states)     *
*********************************************
%

***  example:

% cat -n trace-partitioning.c
     1   void main() {
     2   float t[5] = {-10.0, -10.0, 0.0, 10.0, 10.0};
     3   float c[4] = {0.0, 2.0, 2.0, 0.0};
     4   float d[4] = {-20.0, -20.0, 0.0, 20.0};
     5   float x, r;
     6      __ASTREE_known_fact(((-30.0 <= x) && (x <= 30.0)));
     7     int i = 0;
     8     while ((i < 3) && (x >= t[i+1])) {
     9       i = i + 1;
    10     }
    11     r = (x - t[i]) * c[i] + d[i];
    12      __ASTREE_log_vars((r));
    13   }
%

*** static analysis by Astree (1 -- **WITH**
***  partitioning):

% astree --exec-fn main --no-trace --no-relational trace-partitioning.c \
   |& egrep "(launched)|(WARN)|(r in)"
%

*** static analysis by Astree (2 -- **WITHOUT**
***  partitioning):

% astree --exec-fn main --no-partition --no-trace --no-relational trace-partitioning.c \
```

```
   |& egrep "(launched)|(WARN)|(r in)"
%

************************************************
* Astree tracks potential overflows with floats *
************************************************
%

*** Floats arithmetics does overflow:

% cat -n overflow-c.c
     1   #include <stdio.h>
     2   int main () {
     3   double x,y;
     4   x = 1.0e+256 * 1.0e+256;
     5   y = 1.0e+256 * -1.0e+256;
     6   printf("x = %f, y = %f\n",x,y);
     7   }
     8
%

*** compilation and execution:

% gcc overflow-c.c
./a.out
x = inf, y = -inf
%

*** static analysis with Astree:

% cat -n overflow.c
     1   int main () {
     2   double x,y;
     3   x = 1.0e+256 * 1.0e+256;
     4   y = 1.0e+256 * -1.0e+256;
     5   __ASTREE_log_vars((x,y));
     6   }
%

% astree --exec-fn main  overflow.c |& grep "WARN"
%
*** potential computations with inf, -inf, nan, etc
*** are always signalled by Astree as potential errors
%

************************************************
* Astree handles floats, not reals or fixed point *
* arithmetics                                  *
************************************************
%

*** example of computation error in floats:
***    (x+a)-(x-a) <> 2a! with float
%

% cat -n float-float-c.c
     1   /* float-float-c.c */
     2   #include <stdio.h>
     3   int main () {
     4   float x; float a, y, z, r1, r2;
     5   a = 1.0;
     6   x = 1125899973951488.0;
     7   y = (x + a);
     8   z = (x - a);
     9   r1 = y - z;
    10   r2 = 2 * a;
    11   printf("(x + a) - (x - a) = %f\n", r1);
```

```
    12   printf("2a                  = %f\n", r2);
    13   }
%

*** compilation and execution:

% gcc float-float-c.c
% ./a.out
(x + a) - (x - a) = 0.000000
2a                = 2.000000
%

*** more precision can be better...
***    (x+a)-(x-a) = 2a with double
%

% cat -n double-double-c.c
     1   /* double-double-c.c */
     2   #include <stdio.h>
     3   int main () {
     4   double x; double a, y, z, r1, r2;
     5   a = 1.0;
     6   x = 1125899973951488.0;
     7   y = (x + a);
     8   z = (x - a);
     9   r1 = y - z;
    10   r2 = 2 * a;
    11   printf("(x + a) - (x - a) = %f\n", r1);
    12   printf("2a                = %f\n", r2);
    13   }
%

*** compilation and execution:

% ./a.out
% gcc double-double-c.c
(x + a) - (x - a) = 2.000000
2a                = 2.000000
%

*** computations with different precisions...
*** can be really catastrophic!
***    (x+a)-(x-a) <> 2a! with double+float
%

% cat -n double-float-c.c
     1   /* double-float.c */
     2   #include <stdio.h>
     3   int main () {
     4   double x; float a, y, z, r1, r2;
     5   a = 1.0;
     6   x = 1125899973951488.0;
     7   y = (x + a);
     8   z = (x - a);
     9   r1 = y - z;
    10   r2 = 2 * a;
    11   printf("(x + a) - (x - a) = %f\n", r1);
    12   printf("2a                = %f\n", r2);
    13   }
%

*** compilation and execution:

% gcc double-float-c.c
% ./a.out
(x + a) - (x - a) = 134217728.000000
2a                = 2.000000
```

```
%

*** testing is unlikely to make it
***    (x+a)-(x-a) <> 2a with double+float
%

% cat -n double-float2-c.c
     1   /* double-float2.c */
     2   #include <stdio.h>
     3   int main () {
     4   double x; float a, y, z, r1, r2;
     5   a = 1.0;
     6   x = 1125899973951487.0;
     7   y = (x + a);
     8   z = (x - a);
     9   r1 = y - z;
    10   r2 = 2 * a;
    11   printf("(x + a) - (x - a) = %f\n", r1);
    12   printf("2a                = %f\n", r2);
    13   }
%

*** only one digit difference:

% diff double-float2-c.c double-float-c.c
1c1
< /* double-float2.c */
---
> /* double-float.c */
6c6
< x = 1125899973951487.0;
---
> x = 1125899973951488.0;
%

*** compilation and execution:

% gcc double-float2-c.c
% ./a.out
(x + a) - (x - a) = 0.000000
2a                = 2.000000
%

***********************************************
* Astree takes rounding errors into account... *
***********************************************
%

*** example ((x+a)-(x-a) = 2a in double+double):
%

% cat -n double-double.c
     1   /* double-double.c */
     2   int main () {
     3   double x; double a, y, z, r1, r2;
     4   a = 1.0;
     5   x = 1125899973951488.0;
     6   y = (x + a);
     7   z = (x - a);
     8   r1 = y - z;
     9   r2 = 2 * a;
    10   __ASTREE_log_vars((r1, r2));
    11   }
%

*** static analysis by Astree:
```

```
% astree --exec-fn main --print-float-digits 10 double-double.c \
   |& egrep "(launched)|(r2 in )|(r1 in)"
%

*** example ((x+a)-(x-a) <> 2a in double+float):
%

% cat -n double-float.c
     1   /* double-float-analyze.c */
     2   int main () {
     3   double x; float a, y, z, r1, r2;
     4   a = 1.0;
     5   x = 1125899973951488.0;
     6   y = (x + a);
     7   z = (x - a);
     8   r1 = y - z;
     9   r2 = 2 * a;
    10   __ASTREE_log_vars((r1, r2));
    11   }
%

*** static analysis by Astree:

% astree --exec-fn main --print-float-digits 10 double-float.c \
   |& egrep "(launched)|(r2 in )|(r1 in)"
%

*** Note that Astree takes to worst case among all possible
*** roundings (towards +oo, -oo, 0 or closest).
%

*********************************************************
* Astree takes into account the potential accumulation *
* of rounding errors over very long periods of time... *
*********************************************************
%

*** example 1:

% cat -n rounding-c.c
     1   #include <stdio.h>
     2   int main () {
     3    int i; double x; x = 0.0;
     4    for (i=1; i<=1000000000; i++) {
     5     x = x + 1.0/10.0;
     6    }
     7   printf("x = %f\n", x);
     8   }
%

*** compilation and execution (a few seconds):

% gcc rounding-c.c
% time ./a.out
x = 99999998.745418
11.281u 0.067s 0:12.17 93.1%  0+0k 0+0io 0pf+0w
%

*** We do not find 100000000 since 1.0/10.0
*** is 0.0001100110011001100... in base 2
%

*** static analysis with Astree:

% cat -n rounding.c
     1   int main () {
     2    double x; x = 0.0;
```

```
      3    while (1) {
      4     x = x + 1.0/10.0;
      5      __ASTREE_log_vars((x));
      6      __ASTREE_wait_for_clock(());
      7    }
      8   }
%

% cat rounding.config
  __ASTREE_max_clock((1000000000));
%

% astree --exec-fn main --config-sem rounding.config --unroll 0 rounding.c\
  |& egrep "(x in)|(\|x\|)|(WARN)" | tail -2
%
*** Note that example 1 is at the origin of the
*** Patriot missile failure on Feb. 25th, 1991
%

*** example 2:

% cat -n bary.c
      1   /* bary.c */
      2   typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
      3   float  INIT,C1,I;
      4   float RANDOM_INPUT;
      5   __ASTREE_volatile_input((RANDOM_INPUT [-1.,1.]));
      6
      7   void bary () {
      8     static float X,Y,Z;
      9     if (C1>0.)
     10       {Z = Y;Y = X;}
     11     if (INIT>0.)
     12       {
     13         X=I;
     14         Y=I;
     15         Z=I;
     16       }
     17     else
     18        {X = 0.50000001 * X + 0.30000001*Y + 0.20000001*Z   ;};
     19      __ASTREE_log_vars((X,Y,Z));
     20
     21   }
     22
     23   void main () {
     24    INIT = 1.;
     25     C1 = RANDOM_INPUT;
     26     I  = RANDOM_INPUT;
     27     while (1) {
     28       bary();
     29       INIT = RANDOM_INPUT;
     30       C1 = RANDOM_INPUT;
     31       I  = RANDOM_INPUT;
     32        __ASTREE_wait_for_clock(());
     33    }
     34   }
%

*** configuration file (10 hours at 1/100th s):

% cat -n bary10.config
      1   __ASTREE_max_clock((3600000));
%

*** static analysis by Astree:

% astree --exec-fn main --config-sem bary10.config bary.c \
```

```
 |& tail -n 50 | egrep --after-context 1 "(launched)|(<float-interval: Z in)"
%

*** configuration file (100 hours at 1/100th s):

% cat -n bary100.config
     1   __ASTREE_max_clock((36000000));
%

*** static analysis by Astree:

% astree --exec-fn main --config-sem bary100.config bary.c \
 |& tail -n 50 | egrep --after-context 1 "(launched)|(<Z in)"
%

*** configuration file (1000 hours at 1/100th s):

% cat -n bary1000.config
     1   __ASTREE_max_clock((360000000));
%

*** static analysis by Astree:

% astree --exec-fn main --config-sem bary1000.config bary.c \
 |& tail -n 50 | egrep --after-context 1 "(launched)|(<Z in)"
%
*** (note that the analysis time is independent
***  of the execution time.)
%

**********************************************************
* Astree knows about truncated float computations... *
**********************************************************
%

*** example (truncated computations):

% cat -n moda_dur_3.c
     1   /* entree */
     2   double X;
     3   __ASTREE_volatile_input((X [-186.,186.]));
     4
     5   /* sortie */
     6   double RESULTAT;
     7
     8   void N()
     9   {
    10      int tronc_entier;
    11       double entree,diametre,min,rapport,troncature,plancher,multiple_inf,reste,rest
    12      int BPO;
    13      min = 0;
    14      diametre = 1.;
    15
    16      /* au choix: nouvelle entree ou retroaction */
    17      if (BPO) entree = X;
    18      else      entree = RESULTAT;
    19
    20      /* calcul du rapport de entree - min / diametre, puis de sa troncature */
    21      min = 0;
    22      diametre = 1.;
    23      rapport = (entree - min) / diametre;
    24      tronc_entier = (int) rapport;
    25      troncature = (double) tronc_entier;
    26
    27      /* calcul de la valeur plancher de ce rapport */
    28      if (rapport<0) plancher = troncature - 1;
    29      else             plancher = troncature;
```

```
     30
     31      /* calcul du reste de l'entree */
     32      reste = entree - (diametre * plancher);
     33
     34      /* calcul du multiple inferieur a l'entree*/
     35      multiple_inf = entree - reste;
     36
     37      /* calcul du multiple superieur a l'entree*/
     38      multiple_sup = multiple_inf + diametre;
     39
     40
     41      /* calcul du multiple le plus proche */
     42      if (reste < 0) reste_abs = -reste;
     43      else             reste_abs = reste;
     44      if (reste_abs <= 0.5*diametre)   plus_proche = multiple_inf;
     45      else                              plus_proche = multiple_sup;
     46
     47
     48      /* resultat */
     49      RESULTAT = plus_proche;
     50        __ASTREE_log_vars((entree,RESULTAT;mod,inter));
     51    }
     52
     53
     54    void main()
     55    {
     56      while (1) {
     57        N();
     58         __ASTREE_wait_for_clock(());
     59      }
     60    }
%

*** static analysis by Astree (1 - **WITHOUT**
*** abstract domain for modulo arithmetics):

% astree moda_dur_3.c --exec-fn main --no-mod \
  |& egrep "(launched)|(<float-interval)|(WARN)" |& tail -n 1
%

*** static analysis by Astree (2 - **WITH**
*** abstract domain for modulo arithmetics):

% astree moda_dur_3.c --exec-fn main --mod \
  |& egrep "(launched)|(<float-interval)|(WARN)" |& tail -n 1
%

*** troncation information derived by Astree:

% astree moda_dur_3.c --exec-fn main --mod \
  |& egrep --after-context 18 "(launched)|(WARN)|(direct =)" | tail -n 18
%

***********************************************
* Astree knows about synchronous programming... *
***********************************************
%

*** incorrect example:

% cat -n clock-error.c
     1    /* clock-error.c */
     2    int R, T, n = 10;
     3    void main()
     4    { volatile int I;
     5      R = 0;
     6      while (1) {
```

```
     7       if (I)
     8         { R = R+1; }
     9       else
    10         { R = 0; }
    11       T = (R>=n);
    12   /*   __ASTREE_wait_for_clock(()); */
    13   }}
%

*** configuration file:

% cat -n clock-error.config
     1   /* clock-error.config */
     2   __ASTREE_volatile_input((I [0,1]));
%

*** analysis of the incorrect example by Astree:

% astree --exec-fn main --config-sem clock-error.config clock-error.c |& egrep "(launched)
%

*** correct example:

% cat -n clock.c
     1   /* clock.c */
     2   int R, T, n = 10;
     3   void main()
     4   { volatile int I;
     5     R = 0;
     6     while (1) {
     7       if (I)
     8         { R = R+1; }
     9       else
    10         { R = 0; }
    11       T = (R>=n);
    12        __ASTREE_wait_for_clock(());
    13   }}
%

*** correction (difference with the incorrect program):

% diff clock-error.c clock.c
1c1
< /* clock-error.c */
---
> /* clock.c */
12c12
< /*   __ASTREE_wait_for_clock(()); */
---
>       __ASTREE_wait_for_clock(());
%

*** configuration file:

% cat -n clock.config
     1   /* clock.config */
     2   __ASTREE_volatile_input((I [0,1]));
     3   __ASTREE_max_clock((3600000));
%

*** analysis of the correct example by Astree:

% astree --exec-fn main --config-sem clock.config clock.c |& egrep "(launched)|(WARN)"
%

*****************************************
* Astree knows about control/command theory... *
```

```
***********************************************
%

*** filter example:

% cat -n filtre.c
     1   typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
     2   BOOLEAN INIT;
     3   float P, X;
     4   volatile float RANDOM_INPUT;
     5   __ASTREE_volatile_input((RANDOM_INPUT [-10.0,10.0]));
     6
     7   void filtre2 () {
     8      static float E[2], S[2];
     9      if (INIT) {
    10          S[0] = X;
    11          P = X;
    12          E[0] = X;
    13      } else {
    14           P = (((((0.4677826 * X) - (E[0] * 0.7700725)) + (E[1] * 0.4344376)) + (S[0
    15      }
    16      E[1] = E[0];
    17      E[0] = X;
    18      S[1] = S[0];
    19      S[0] = P;
    20   }
    21
    22   void main () {
    23      X = RANDOM_INPUT;
    24      INIT = TRUE;
    25      while (TRUE) {
    26        X = RANDOM_INPUT;
    27        filtre2 ();
    28        INIT = FALSE;
    29      }
    30   }
%

*** static analysis by Astree (1 -- WITH 2nd order
*** filter domain):

% astree filtre.c --dump-invariants --exec-fn main |& egrep "(launched)|(WARN)|(P in)"
%

*** static analysis by Astree (2 -- WITHOUT 2nd order
*** filter domain):

% astree filtre.c --exec-fn main --no-filters --dump-invariants |& egrep "(launched)|(WARN
%

***********************************************
* Astree can analyze low level memory operations *
***********************************************
%

*** example 1 (pointer casts):

% cat -n memcpy.c
     1   /* memcpy.c (polymorphic memcpy) */
     2
     3   /* byte per byte copy of src into dst */
     4   void memcpy(char* dst, const char* src, unsigned size)
     5   {
     6     int i;
     7      for (i=0;i<size;i++) dst[i] = src[i];
     8   }
     9
```

```
    10   void main()
    11   {
    12     float x = 10.0, y;
    13     int zero = 0;
    14     /* copy of x into y (well-typed) */
    15      memcpy(&y,&x,sizeof(y));
    16      __ASTREE_assert((y==10.0));
    17     /* copy of zero into y (not well-typed but allowed in C) */
    18      memcpy(&y,&zero,sizeof(y));
    19      __ASTREE_assert((y==0.0));
    20   }
%

*** static analysis by Astree:

% astree --exec-fn main --unroll 5 memcpy.c |& egrep "(launched)|(WARN)"
%

*** example 2 (unions):

% cat -n union.c
     1   /* union.c (union type) */
     2
     3   union {
     4     int type;
     5     struct { int type; int data; } A;
     6     struct { int type; char data[3]; } B;
     7   } u;
     8
     9   void main()
    10   {
    11     /* no assert failure */
    12     u.type = 12;
    13      __ASTREE_assert((u.A.type==12));
    14      __ASTREE_assert((u.B.type==12));
    15
    16     /* assert failure because the modification of u.B.data also modifies u.A.data *
    17     u.A.data = 0;
    18     u.B.data[0] = 12;
    19      __ASTREE_assert((u.A.data==0));
    20   }
%

*** static analysis by Astree:

% astree --exec-fn main --full-memory-model union.c |& egrep "(launched)|(WARN)"
%


**********************************************
* Astree has a graphic interface under X11... *
**********************************************
%

*** static analysis by Astree

% astree filtre.c --dump-invariants --exec-fn main --export-invariant stat \
   --export-file filtre.inv --export-unroll >& /dev/null
%

*** visualization of the results:
%

% visu --text-size 14 --text-font CMTT filtre.inv >& /dev/null
%

*** (scaling up with GTK+ (library to build graphical
```

```
***   user interfaces (GUIs) originally for X Window)!)
%

*****************************************
*** The end, thank you for your attention ***
*****************************************

demo-astree/programs  %
```