

```

1
2
3
4 *****
5 *****
6 ***                                     ***
7 *** Demonstration of the Astree static analyzer ***
8 ***      http://www.astree.ens.fr/          ***
9 ***                                     ***
10 *** P. Cousot, R. Cousot, J. Feret, L. Mauborgne, ***
11 *** A. Mine, X. Rival [2001--]             ***
12 *** [B. Blanchet (2001/03), D. Monniaux (2001/07)] ***
13 ***                                     ***
14 *****
15 *****
16 %
17
18 *****
19 * Astree is a VERIFIER (not a bug-finder). *
20 * Astree is SOUND hence should report ALL *
21 * potential runtime errors.               *
22 *****
23 %
24
25 *** example [CC76]:
26
27 % cat -n dichotomy-error.c
28 1 /* dichotomy-error.c */
29 2 int main () {
30 3   int lwb, upb, m, R[100], X;
31 4   lwb = 1; upb = 100;
32 5   while (lwb <= upb) {
33 6     m = (upb + lwb) / 2;
34 7     if (X == R[m]) {
35 8       upb = m; lwb = m+1; }
36 9     else if (X < R[m]) {
37 10      upb = m - 1; }
38 11     else {
39 12      lwb = m + 1; }
40 13   }
41 14   __ASTREE_log_vars((m));
42 15 }
43 %
44
45 *** static analysis by Astree:
46
47 % astree --exec-fn main --no-relational --unroll 0 dichotomy-error.c \
48   |& egrep --after-context 0 "(launched)|(WARN)"
49
50 dichotomy-error.c:7.15-19::[call#main@2:]: WARN: invalid dereference: dereferencing 4 byte(s) at
... offset(s) [4;400] may overflow the variable R of byte-size 400
51 dichotomy-error.c:7.15-19::[call#main@2:]: WARN: invalid dereference: dereferencing 4 byte(s) at
... offset(s) [4;400] may overflow the variable R of byte-size 400
52 dichotomy-error.c:9.19-23::[call#main@2:]: WARN: invalid dereference: dereferencing 4 byte(s) at
... offset(s) [4;400] may overflow the variable R of byte-size 400
53 dichotomy-error.c:9.19-23::[call#main@2:]: WARN: invalid dereference: dereferencing 4 byte(s) at
... offset(s) [4;400] may overflow the variable R of byte-size 400
54 %
55
56 *** (the two errors are reported two times each
57 *** for the two branches of the conditional.)
58 %
59
60 *** correcting the error:
61
62 % cat -n dichotomy.c
63 1 /* dichotomy.c */
64 2 int main () {
65 3   int lwb, upb, m, R[100], X;
66 4   lwb = 0; upb = 99;
67 5   while (lwb <= upb) {

```

```

68     6     m = (upb + lwb) / 2;
69     7     if (X == R[m]) {
70     8         upb = m; lwb = m+1; }
71     9     else if (X < R[m]) {
72    10         upb = m - 1; }
73    11     else {
74    12         lwb = m + 1; }
75    13     }
76    14     __ASTREE_log_vars((m));
77    15 }
78 %
79
80 *** correction (difference with the erroneous version):
81
82 % diff dichotomy-error.c dichotomy.c
83 1c1
84 < /* dichotomy-error.c */
85 ---
86 > /* dichotomy.c */
87 4c4
88 <     lwb = 1; upb = 100;
89 ---
90 >     lwb = 0; upb = 99;
91 %
92
93 *** static analysis by Astree:
94
95 % astree --exec-fn main --no-relational dichotomy.c \
96     |& egrep "(launched)|(m in )|(WARN)"
97
98 direct = <integers (intv+cong+bitfield+set): m in [0, 99] >
99 %
100
101 *****
102 * Astree is INCOMPLETE hence may report false alarms *
103 *****
104 %
105
106 *** example of false alarm:
107
108 % cat -n fausse-alarme.c
109     1 /* fausse-alarme.c */
110     2 void main()
111     3 {
112     4     int x, y;
113     5     if ((-4681 < y) && (y < 4681) && (x < 32767) && (-32767 < x) && ((7*y*y - 1) == x*x)) {
114     6         y = 1 / x;
115     7     };
116     8 }
117 %
118
119 *** static analysis by Astree:
120
121 % astree --exec-fn main fausse-alarme.c |& egrep "(launched)|(WARN)"
122
123 fausse-alarme.c:6.9-14:.[call#main@2:]: WARN: integer division by zero [-32766, 32766]
124 %
125
126 *****
127 * Astree tracks all potential buffer overruns *
128 *****
129 %
130
131 *** example of uninitialized and buffer overrun:
132
133 % cat -n bufferoverrun-c.c
134     1 #include <stdio.h>
135     2 int main () {
136     3     int x, y, z, T[9];
137     4     x = T[7];
138     5     y = T[8];

```

```

139     6 z = T[9];
140     7 printf("x = %i, y = %i, z = %i\n",x,y,z);
141     8 }
142     9
143 %
144
145 *** compilation and execution:
146
147 % gcc bufferoverflow-c.c
148 % ./a.out
149 x = 4096, y = 0, z = 4096
150 %
151
152 *** static analysis with Astree:
153
154 % cat -n bufferoverflow.c
155     1 int main () {
156     2 int a, x, y, z, T[9];
157     3 x = T[7];
158     4 y = T[8];
159     5 z = T[9];
160     6 __ASTREE_log_vars((x,y,z));
161     7 }
162     8
163 %
164
165 % astree --exec-fn main bufferoverflow.c\
166     |& egrep "(x in)|(y in)|(z in)|(WARN)"
167 bufferoverflow.c:5.4-8::[call#main@1:]: WARN: invalid dereference: dereferencing 4 byte(s) at
... offset(s) [36;36] may overflow the variable T of byte-size 36
168 %
169 *** Astree signals the definite error and considers the
170 *** (unpredictable) execution to be stopped (so no log).
171 %
172
173 *****
174 * Astree tracks all potential dangling pointers *
175 *****
176 %
177
178 *** example of dangling pointer:
179
180 % cat -n danglingpointer-c.c
181     1 #include <stdio.h>
182     2 int main () {
183     3 int x, y, z, *r;
184     4 x = 100;
185     5 r = &x;
186     6 y = *r;
187     7 z = *(r+2);
188     8 printf("x = %i, y = %i, z = %i\n",x,y,z);
189     9 }
190    10
191 %
192
193 *** compilation and execution:
194
195 % gcc danglingpointer-c.c
196 % ./a.out
197 x = 100, y = 100, z = -1073748436
198 %
199
200 *** static analysis with Astree:
201
202 % cat -n danglingpointer.c
203     1 int main () {
204     2 int x, y, z, *r;
205     3 x = 100;
206     4 r = &x;
207     5 y = *r;
208     6 z = *(r+2);

```

```

209     7 __ASTREE_log_vars((x,y,z));
210     8 }
211     9
212 %
213
214 % astree --exec-fn main danglingpointer.c\  

215     |& egrep "(x in)|(y in)|(z in)|(WARN)"
216 danglingpointer.c:6.4-10::[call#main@1:]: WARN: invalid dereference: dereferencing 4 byte(s) at
... offset(s) [8;8] may overflow the variable x of byte-size 4
217 %
218 *** Astree signals the definite error and considers the
219 *** (unpredictable) execution to be stopped (so no log).
220 %
221
222 *****
223 * Astree uses a predictable semantics of C *
224 * conforming with the standard and with the *
225 * implementation but only in absence of *
226 * unpredictable runtime errors. *
227 *****
228 %
229
230 % cat -n Unsafe1.c
231     1 int A[4], c;
232     2 void bad(int *p, int x, int y) {
233     3     p[4] = x;
234     4     if( c!=0 ) {
235     5         A[1003] = y;
236     6     }
237     7 }
238     8 void ok(int *q, int n) {
239     9     c = 0;
240    10     q[0] = n;
241    11 }
242    12 int main() {
243    13     ok(A,0);
244    14     bad(A,1,0);
245    15 }
246 %
247
248 *** Which errors should be raised by Astree?
249
250 % astree --exec-fn main Unsafe1.c |& egrep "(launched)|(WARN)"
251
252 Unsafe1.c:3.9-10::[call#main@12:call#bad@14:]: WARN: invalid dereference: dereferencing 4
... byte(s) at offset(s) [16;16] may overflow the variable A of byte-size 16
253 %
254 *****
255 * 5: never executed *
256 *****
257
258 % cat -n Unsafe2.c
259     1 int A[4], c;
260     2 void bad(int *p, int x, int y) {
261     3     p[4] = x;
262     4     if( c!=0 ) {
263     5         A[1003] = y;
264     6     }
265     7 }
266     8 void ok(int *q, int n) {
267     9     c = 1;
268    10     q[0] = n;
269    11 }
270    12 int main() {
271    13     ok(A,0);
272    14     bad(A,1,0);
273    15 }
274 %
275
276 *** Which errors should be raised by Astree?
277

```

```

278 % astree --exec-fn main Unsafe2.c |& egrep "(launched)|(WARN)"
279
280 Unsafe2.c:3.9-10::[call#main@12:call#bad@14:]: WARN: invalid dereference: dereferencing 4
... byte(s) at offset(s) [16;16] may overflow the variable A of byte-size 16
281 %
282 *****
283 * Execution stops at 3: with a definite error, *
284 * 5: can be executed in some implementations. *
285 *****
286
287
288 % cat -n Unsafe3.c
289     1 int A[4], c;
290     2 void bad(int *p, int x, int y) {
291     3     p[3] = x;
292     4     if( c!=0 ) {
293     5         A[1003] = y;
294     6     }
295     7 }
296     8 void ok(int *q, int n) {
297     9
298    10     q[0] = n;
299    11 }
300    12 int main() {
301    13     ok(A,0);
302    14     bad(A,1,0);
303    15 }
304 %
305
306 *** Which errors should be raised by Astree?
307
308 % astree --exec-fn main Unsafe3.c |& egrep "(launched)|(WARN)"
309
310 %
311 *****
312 * c implicitly initialized to 0. *
313 *****
314
315 *** Static analysis by Astree without implicit
316 *** initialization of globals to zero:
317
318 % astree --exec-fn main --no-global-initialization Unsafe3.c |& egrep "(launched)|(WARN)"
319
320 Unsafe3.c:5.15-16::[call#main@12:call#bad@14:]: WARN: invalid dereference: dereferencing 4
... byte(s) at offset(s) [4012;4012] may overflow the variable A of byte-size 16
321 %
322 *****
323 * A and c now uninitialized. *
324 *****
325
326 *****
327 * Astree tracks potential modulo arithmetics errors *
328 *****
329 %
330
331 *** Modulo arithmetics is not very intuitive:
332
333 % cat -n modulo-c.c
334     1 #include <stdio.h>
335     2 int main () {
336     3     int x,y;
337     4     x = -2147483647 / -1;
338     5     y = ((-x) -1) / -1;
339     6     printf("x = %i, y = %i\n",x,y);
340     7 }
341     8
342 %
343
344 *** compilation and execution:
345
346 % gcc modulo-c.c

```

```

347 % ./a.out
348 x = 2147483647, y = -2147483648
349 %
350 *** -2147483648 / -1 = -2147483648 ???
351 %
352
353 *** static analysis with Astree:
354
355 % cat -n modulo.c
356     1 int main () {
357     2 int x,y;
358     3 x = -2147483647 / -1;
359     4 y = ((-x) -1) / -1;
360     5 __ASTREE_log_vars((x,y));
361     6 }
362     7
363 %
364
365 % astree --exec-fn main --unroll 0 modulo.c\
366     |& egrep -A 1 "(<integers)|(WARN)"
367 modulo.c:4.4-18::[call#main@1:]: WARN: signed int arithmetic range {2147483648} not included in
... [-2147483648, 2147483647]
368 modulo.c:5.0-24: log:
369 --
370 <integers (intv+cong+bitfield+set): y in [-2147483648, 2147483647],
371 x in {2147483647} >
372 %
373 *** Astree signals the error and goes on predictively
374 *** but with an unkown value (hence the log)
375 %
376
377 *****
378 * Astree uses interval analysis (enhanced *
379 * by symbolic execution) *
380 *****
381
382 *** example:
383
384 % cat -n interval.c
385     1 int main () {
386     2 int x, y;
387     3 __ASTREE_known_fact(((0 <= x) && (x <= 100)));
388     4 y = x - x;
389     5 __ASTREE_log_vars((x,y));
390     6 }
391 %
392
393 *** static analysis by Astree (1 -- WITHOUT symbolic execution):
394
395 % astree interval.c --no-relational --exec-fn main \
396     |& egrep "(launched)|(x in)|(y in)"
397
398 <integers (intv+cong+bitfield+set): y in [-100, 100], x in [0, 100] >
399 %
400
401 *** static analysis by Astree (2 -- WITH symbolic execution):
402
403 % astree interval.c --exec-fn main \
404     |& egrep "(launched)|(y in)"
405
406 direct = <integers (intv+cong+bitfield+set): y in {0}, x in [0, 100] >
407 %
408
409 *** The symbolic abstract domain propagates the
410 *** symbolic value of variables (plus rounding
411 *** errors) to perform simplifications.
412 %
413
414 *****
415 * Astree uses the reduction of [non]Drelational abstract *
416 * domains such as intervals and congruences *

```

```

417 *****
418 %
419
420 *** example:
421
422 % cat -n congruence.c
423     1 /* congruence.c */
424     2 int main()
425     3 { int X;
426     4   X = 0;
427     5   while (X <= 128)
428     6     { X = X + 4; };
429     7   __ASTREE_log_vars((X));
430     8 }
431 %
432
433 *** static analysis by Astree:
434
435 % astree congruence.c --no-relational --exec-fn main |& egrep "(launched)|(WARN)|(X in)"
436
437 direct = <integers (intv+cong+bitfield+set): X in {132} >
438 %
439
440 *****
441 * Astree uses weakly relational abstract *
442 * domains such as octagons... *
443 *****
444
445 *** example:
446
447 % cat -n octagon.c
448     1 /* octagon.c */
449     2 void main()
450     3 {
451     4   int X, Y;
452     5   X = 10;
453     6   Y = 100;
454     7   while (X >= 0) {
455     8     X--;
456     9     Y--;
457    10   };
458    11   __ASTREE_assert((X <= Y));
459    12 }
460 %
461
462 *** static analysis by Astree (1 -- WITHOUT octagons):
463
464 % astree octagon.c --no-octagon --exec-fn main |& egrep "(launched)|(WARN)"
465
466 octagon.c:9.4-7::[call#main@2:loop@7>=4:]: WARN: signed int arithmetic range [-2147483649,
... 2147483646] not included in [-2147483648, 2147483647]
467 octagon.c:11.19-25::[call#main@2:]: WARN: assert failure
468 %
469
470 *** static analysis by Astree (2 -- WITH octagons):
471
472 % astree octagon.c --exec-fn main |& egrep "(launched)|(WARN)"
473
474 %
475 *** Does not scale up to too many variables,
476 *** --> packs of variables.
477 %
478
479 *** static analysis by Astree (3 -- octagon packs):
480
481 % astree octagon.c --exec-fn main --print-packs |& egrep -A 2 "List of packs"
482 List of packs
483   octagon.c@4@5 { X Y }
484 Size of packs [ 2 (octagon.c@4@5) ]
485 --
486 List of packs

```

```

487 Size of packs [ ]
488 Occurrence of each variable [ ]
489 %
490
491 *****
492 * Astree uses weakly relational abstract *
493 * domains such as boolean decision trees... *
494 *****
495 %
496
497 *** example:
498
499 % cat -n boolean.c
500     1 /* boolean.c */
501     2 typedef enum {F=0,T=1} BOOL;
502     3 BOOL B;
503     4 void main () {
504     5     unsigned int X, Y;
505     6     while (1) {
506     7         /* ... */
507     8         B = (X == 0);
508     9         /* ... */
509    10         if (!B) {
510    11             Y = 1 / X;
511    12         }
512    13         /* ... */
513    14     }
514    15 }
515 %
516
517 *** static analysis by Astree (1 -- **WITHOUT**
518 *** decision trees):
519
520 % astree boolean.c --no-relational --exec-fn main |& egrep "(launched)|(WARN)"
521
522 boolean.c:11.13-18::[call#main@4:loop@6=1:]: WARN: integer division by zero [0, 4294967295]
523 boolean.c:11.13-18::[call#main@4:loop@6=2:]: WARN: integer division by zero [0, 4294967295]
524 boolean.c:11.13-18::[call#main@4:loop@6=3:]: WARN: integer division by zero [0, 4294967295]
525 boolean.c:11.13-18::[call#main@4:loop@6>=4:]: WARN: integer division by zero [0, 4294967295]
526 %
527
528 *** static analysis by Astree (2 -- **WITH**
529 *** decision trees):
530
531 % astree boolean.c --exec-fn main |& egrep "(launched)|(WARN)"
532
533 %
534
535 *****
536 * Astree uses computation trace abstractions *
537 * (describing sequences of states) not only *
538 * invariants (describing sets of states) *
539 *****
540 %
541
542 *** example:
543
544 % cat -n trace-partitioning.c
545     1 void main() {
546     2 float t[5] = {-10.0, -10.0, 0.0, 10.0, 10.0};
547     3 float c[4] = {0.0, 2.0, 2.0, 0.0};
548     4 float d[4] = {-20.0, -20.0, 0.0, 20.0};
549     5 float x, r;
550     6 __ASTREE_known_fact((( -30.0 <= x) && (x <= 30.0)));
551     7 int i = 0;
552     8 while ((i < 3) && (x >= t[i+1])) {
553     9     i = i + 1;
554    10 }
555    11 r = (x - t[i]) * c[i] + d[i];
556    12 __ASTREE_log_vars((r));
557    13 }

```



```

558 %
559
560 *** static analysis by Astree (1 -- **WITH**
561 *** partitioning):
562
563 % astree --exec-fn main --no-trace --no-relational trace-partitioning.c \
564   |& egrep "(launched)|(WARN)|(r in)"
565
566 /* Domains: Linearization, and Integer intervals, and congruences, and bitfields, and finite
... integer sets, and Float intervals. */
567 direct = <float-interval: r in [-20., 20.] >
568 %
569
570 *** static analysis by Astree (2 -- **WITHOUT**
571 *** partitioning):
572
573 % astree --exec-fn main --no-partition --no-trace --no-relational trace-partitioning.c \
574   |& egrep "(launched)|(WARN)|(r in)"
575
576 /* Domains: Linearization, and Integer intervals, and congruences, and bitfields, and finite
... integer sets, and Float intervals. */
577 direct = <float-interval: r in [-100., 100.] >
578 %
579
580 *** static analysis by Astree (3 -- automatic
581 *** insertion of partitioning directives):
582
583 astree --exec-fn main --dump-partition trace-partitioning.c \
584   |& egrep --after-context 17 "void"
585 void (main)()
586 {
587   float (t[5]) = { -1.000000e+01, -1.000000e+01, 0.000000e+00, 1.000000e+01, 1.000000e+01};
588   float (c[4]) = { 0.000000e+00, 2.000000e+00, 2.000000e+00, 0.000000e+00};
589   float (d[4]) = { -2.000000e+01, -2.000000e+01, 0.000000e+00, 2.000000e+01};
590   float x;
591   float r;
592   __ASTREE_known_fact(((((-30. <= x) && (x <= 30.)))));
593   signed int i = 0;
594   __ASTREE_partition_control((15))
595   while (((i < 3) && (x >= t[(i + 1)])))
596   {
597     i = (i + 1);
598   }
599   r = (((x - t[i]) * c[i]) + d[i]);
600   __ASTREE_partition_merge(());
601   __ASTREE_log_vars((r));
602 }
603 %
604
605 *****
606 * Astree tracks potential overflows with floats *
607 *****
608 %
609
610 *** Floats arithmetics does overflow:
611
612 % cat -n overflow-c.c
613   1 #include <stdio.h>
614   2 int main () {
615   3 double x,y;
616   4 x = 1.0e+256 * 1.0e+256;
617   5 y = 1.0e+256 * -1.0e+256;
618   6 printf("x = %f, y = %f\n",x,y);
619   7 }
620   8
621 %
622
623 *** compilation and execution:
624
625 % gcc overflow-c.c
626 ./a.out

```

```

627 x = inf, y = -inf
628 %
629
630 *** static analysis with Astree:
631
632 % cat -n overflow.c
633     1 int main () {
634     2 double x,y;
635     3 x = 1.0e+256 * 1.0e+256;
636     4 y = 1.0e+256 * -1.0e+256;
637     5 __ASTREE_log_vars((x,y));
638     6 }
639 %
640
641 % astree --exec-fn main overflow.c |& grep "WARN"
642 overflow.c:3.4-23::[call#main@1:]: WARN: double arithmetic range [1.7976932e+308, inf.] not
... included in [-1.7976931e+308, 1.7976932e+308]
643 overflow.c:4.4-24::[call#main@1:]: WARN: double arithmetic range [-inf., -1.7976931e+308] not
... included in [-1.7976931e+308, 1.7976932e+308]
644 %
645 *** Potential computations with inf, -inf, nan, etc
646 *** are always signalled by Astree as potential errors.
647 *** Volatiles are assumed not to be inf, -inf, nan.
648 %
649
650 *****
651 * Astree handles floats, not reals or fixed point *
652 * arithmetics *
653 *****
654 %
655
656 *** example of computation error in floats:
657 *** (x+a)-(x-a) <> 2a! with float
658 %
659
660 % cat -n float-float-c.c
661     1 /* float-float-c.c */
662     2 #include <stdio.h>
663     3 int main () {
664     4 float x; float a, y, z, r1, r2;
665     5 a = 1.0;
666     6 x = 1125899973951488.0;
667     7 y = (x + a);
668     8 z = (x - a);
669     9 r1 = y - z;
670    10 r2 = 2 * a;
671    11 printf("(x + a) - (x - a) = %f\n", r1);
672    12 printf("2a                = %f\n", r2);
673    13 }
674 %
675
676 *** compilation and execution:
677
678 % gcc float-float-c.c
679 % ./a.out
680 (x + a) - (x - a) = 0.000000
681 2a                = 2.000000
682 %
683
684 *** more precision can be better...
685 *** (x+a)-(x-a) = 2a with double
686 %
687
688 % cat -n double-double-c.c
689     1 /* double-double-c.c */
690     2 #include <stdio.h>
691     3 int main () {
692     4 double x; double a, y, z, r1, r2;
693     5 a = 1.0;
694     6 x = 1125899973951488.0;
695     7 y = (x + a);

```

```

696     8 z = (x - a);
697     9 r1 = y - z;
698    10 r2 = 2 * a;
699    11 printf("(x + a) - (x - a) = %f\n", r1);
700    12 printf("2a           = %f\n", r2);
701    13 }
702 %
703
704 *** compilation and execution:
705
706 % ./a.out
707 % gcc double-double-c.c
708 (x + a) - (x - a) = 2.000000
709 2a           = 2.000000
710 %
711
712 *** computations with different precisions...
713 *** can be really catastrophic!
714 *** (x+a)-(x-a) <> 2a! with double+float
715 %
716
717 % cat -n double-float-c.c
718     1 /* double-float.c */
719     2 #include <stdio.h>
720     3 int main () {
721     4 double x; float a, y, z, r1, r2;
722     5 a = 1.0;
723     6 x = 1125899973951488.0;
724     7 y = (x + a);
725     8 z = (x - a);
726     9 r1 = y - z;
727    10 r2 = 2 * a;
728    11 printf("(x + a) - (x - a) = %f\n", r1);
729    12 printf("2a           = %f\n", r2);
730    13 }
731 %
732
733 *** compilation and execution:
734
735 % gcc double-float-c.c
736 % ./a.out
737 (x + a) - (x - a) = 134217728.000000
738 2a           = 2.000000
739 %
740
741 *** testing is unlikely to make it
742 *** (x+a)-(x-a) <> 2a with double+float
743 %
744
745 % cat -n double-float2-c.c
746     1 /* double-float2.c */
747     2 #include <stdio.h>
748     3 int main () {
749     4 double x; float a, y, z, r1, r2;
750     5 a = 1.0;
751     6 x = 1125899973951487.0;
752     7 y = (x + a);
753     8 z = (x - a);
754     9 r1 = y - z;
755    10 r2 = 2 * a;
756    11 printf("(x + a) - (x - a) = %f\n", r1);
757    12 printf("2a           = %f\n", r2);
758    13 }
759 %
760
761 *** only one digit difference:
762
763 % diff double-float2-c.c double-float-c.c
764 1c1
765 < /* double-float2.c */
766 ---

```

```

767 > /* double-float.c */
768 6c6
769 < x = 1125899973951487.0;
770 ---
771 > x = 1125899973951488.0;
772 %
773
774 *** compilation and execution:
775
776 % gcc double-float2-c.c
777 % ./a.out
778 (x + a) - (x - a) = 0.000000
779 2a                = 2.000000
780 %
781
782 *****
783 * Astree takes rounding errors into account... *
784 *****
785 %
786
787 *** example ((x+a)-(x-a) = 2a in double+double):
788 %
789
790 % cat -n double-double.c
791     1 /* double-double.c */
792     2 int main () {
793     3 double x; double a, y, z, r1, r2;
794     4 a = 1.0;
795     5 x = 1125899973951488.0;
796     6 y = (x + a);
797     7 z = (x - a);
798     8 r1 = y - z;
799     9 r2 = 2 * a;
800    10 __ASTREE_log_vars((r1, r2));
801    11 }
802 %
803
804 *** static analysis by Astree:
805
806 % astree --exec-fn main --print-float-digits 10 double-double.c \
807   |& egrep "(launched)|(r2 in)|(r1 in)"
808
809 direct = <float-interval: r2 in {2.}, r1 in {2.} >
810 %
811
812 *** example ((x+a)-(x-a) <> 2a in double+float):
813 %
814
815 % cat -n double-float.c
816     1 /* double-float-analyze.c */
817     2 int main () {
818     3 double x; float a, y, z, r1, r2;
819     4 a = 1.0;
820     5 x = 1125899973951488.0;
821     6 y = (x + a);
822     7 z = (x - a);
823     8 r1 = y - z;
824     9 r2 = 2 * a;
825    10 __ASTREE_log_vars((r1, r2));
826    11 }
827 %
828
829 *** static analysis by Astree:
830
831 % astree --exec-fn main --print-float-digits 10 double-float.c \
832   |& egrep "(launched)|(r2 in)|(r1 in)"
833
834 direct = <float-interval: r2 in {2.}, r1 in [-134217728., 134217728.] >
835 %
836
837 *** Note that Astree takes to worst case among all possible

```

```

838 *** roundings (towards +oo, -oo, 0 or closest).
839 %
840
841 *****
842 * Astree takes into account the potential accumulation *
843 * of rounding errors over very long periods of time... *
844 *****
845 %
846
847 *** example 1:
848
849 % cat -n rounding-c.c
850     1 #include <stdio.h>
851     2 int main () {
852     3   int i; double x; x = 0.0;
853     4   for (i=1; i<=1000000000; i++) {
854     5     x = x + 1.0/10.0;
855     6   }
856     7   printf("x = %f\n", x);
857     8 }
858 %
859
860 *** compilation and execution (a few seconds):
861
862 % gcc rounding-c.c
863 % time ./a.out
864 x = 99999998.745418
865 3.476u 0.008s 0:03.48 99.7% 0+0k 0+0io 0pf+0w
866 %
867
868 *** We do not find 100000000 since 1.0/10.0
869 *** is 0.0001100110011001100... in base 2
870 %
871
872 *** static analysis with Astree:
873
874 % cat -n rounding.c
875     1 int main () {
876     2   double x; x = 0.0;
877     3   while (1) {
878     4     x = x + 1.0/10.0;
879     5     __ASTREE_log_vars((x));
880     6     __ASTREE_wait_for_clock(( ));
881     7   }
882     8 }
883 %
884
885 % cat rounding.config
886 __ASTREE_max_clock((100000000));
887 %
888
889 % astree --exec-fn main --config-sem rounding.config --unroll 0 rounding.c\
890 |& egrep "(x in)|(\|x\!)| (WARN)" | tail -2
891 direct = <float-interval: x in [0.1, 2.0000005e+08] >
892 |x| <= 1.0000001*((0. + 0.10000001/(1.0000001-1))*(1.0000001)^clock -
... 0.10000001/(1.0000001-1)) + 0.10000001 <= 2.0000005e+08
893 %
894 *** Note that example 1 is at the origin of the
895 *** Patriot missile failure on Feb. 25th, 1991
896 %
897
898 *** example 2:
899
900 % cat -n bary.c
901     1 /* bary.c */
902     2 typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
903     3 float INIT,C1,I;
904     4 float RANDOM_INPUT;
905     5 __ASTREE_volatile_input((RANDOM_INPUT [-1.,1.]));
906     6
907     7 void bary () {

```

```

908     8  static float X,Y,Z;
909     9  if (C1>0.)
910     10 {Z = Y;Y = X;}
911     11 if (INIT>0.)
912     12 {
913     13     X=I;
914     14     Y=I;
915     15     Z=I;
916     16 }
917     17 else
918     18 {X = 0.50000001 * X + 0.30000001*Y + 0.20000001*Z  };
919     19 __ASTREE_log_vars((X,Y,Z));
920     20
921     21 }
922     22
923     23 void main () {
924     24 INIT = 1.;
925     25 C1 = RANDOM_INPUT;
926     26 I = RANDOM_INPUT;
927     27 while (1) {
928     28     bary();
929     29     INIT = RANDOM_INPUT;
930     30     C1 = RANDOM_INPUT;
931     31     I = RANDOM_INPUT;
932     32     __ASTREE_wait_for_clock();
933     33 }
934     34 }
935 %
936
937 *** configuration file (10 hours at 1/100th s):
938
939 % cat -n bary10.config
940     1 __ASTREE_max_clock((3600000));
941 %
942
943 *** static analysis by Astree:
944
945 % astree --exec-fn main --config-sem bary10.config bary.c \
946 |& tail -n 50 | egrep --after-context 1 "(launched)|(<float-interval: Z in)"
947 <float-interval: Z in [-1.7111293, 1.7111294],
948 %
949
950 *** configuration file (100 hours at 1/100th s):
951
952 % cat -n bary100.config
953     1 __ASTREE_max_clock((36000000));
954 %
955
956 *** static analysis by Astree:
957
958 % astree --exec-fn main --config-sem bary100.config bary.c \
959 |& tail -n 50 | egrep --after-context 1 "(launched)|(<Z in)"
960 <float-interval: Z in [-215.19279, 215.1928], Y in [-215.19279, 215.1928],
961 %
962
963 *** configuration file (1000 hours at 1/100th s):
964
965 % cat -n bary1000.config
966     1 __ASTREE_max_clock((360000000));
967 %
968
969 *** static analysis by Astree:
970
971 % astree --exec-fn main --config-sem bary1000.config bary.c \
972 |& tail -n 50 | egrep --after-context 1 "(launched)|(<Z in)"
973 <float-interval: Z in [-2.1294954e+23, 2.1294955e+23],
974 %
975 *** (note that the analysis time is independent
976 *** of the execution time.)
977 %
978

```

```

979 *****
980 * Astree knows about truncated float computations... *
981 *****
982 %
983
984 *** example (truncated computations):
985
986 % cat -n moda_dur_3.c
987     1 /* entree */
988     2 double X;
989     3 __ASTREE_volatile_input((X [-186.,186.]));
990     4
991     5 /* sortie */
992     6 double RESULTAT;
993     7
994     8 void N()
995     9 {
996    10     int tronc_entier;
997    11     double
... entree,diametre,min,rapport,troncature,plancher,multiple_inf,reste,reste_abs,multiple_sup,
... plus_proche;
998    12     int BPO;
999    13     min = 0;
1000   14     diametre = 1.;
1001   15
1002   16     /* au choix: nouvelle entree ou retroaction */
1003   17     if (BPO) entree = X;
1004   18     else     entree = RESULTAT;
1005   19
1006   20     /* calcul du rapport de entree - min / diametre, puis de sa troncature */
1007   21     min = 0;
1008   22     diametre = 1.;
1009   23     rapport = (entree - min) / diametre;
1010   24     tronc_entier = (int) rapport;
1011   25     troncature = (double) tronc_entier;
1012   26
1013   27     /* calcul de la valeur plancher de ce rapport */
1014   28     if (rapport<0) plancher = troncature - 1;
1015   29     else           plancher = troncature;
1016   30
1017   31     /* calcul du reste de l'entree */
1018   32     reste = entree - (diametre * plancher);
1019   33
1020   34     /* calcul du multiple inferieur a l'entree*/
1021   35     multiple_inf = entree - reste;
1022   36
1023   37     /* calcul du multiple superieur a l'entree*/
1024   38     multiple_sup = multiple_inf + diametre;
1025   39
1026   40
1027   41     /* calcul du multiple le plus proche */
1028   42     if (reste < 0) reste_abs = -reste;
1029   43     else           reste_abs = reste;
1030   44     if (reste_abs <= 0.5*diametre) plus_proche = multiple_inf;
1031   45     else           plus_proche = multiple_sup;
1032   46
1033   47
1034   48     /* resultat */
1035   49     RESULTAT = plus_proche;
1036   50     __ASTREE_log_vars((entree,RESULTAT;mod,inter));
1037   51 }
1038   52
1039   53
1040   54 void main()
1041   55 {
1042   56     while (1) {
1043   57         N();
1044   58         __ASTREE_wait_for_clock();
1045   59     }
1046   60 }
1047 %

```

```

1048
1049 *** static analysis by Astree (1 - **WITHOUT**
1050 *** abstract domain for modulo arithmetics):
1051
1052 % astree moda_dur_3.c --exec-fn main --no-mod \
1053 |& egrep "(launched)|(<float-interval)|(WARN)" |& tail -n 1
1054 <float-interval: entree in [-18328581., 19048581.],
1055 %
1056
1057 *** static analysis by Astree (2 - **WITH**
1058 *** abstract domain for modulo arithmetics):
1059
1060 % astree moda_dur_3.c --exec-fn main --mod \
1061 |& egrep "(launched)|(<float-interval)|(WARN)" |& tail -n 1
1062 <float-interval: entree in [-186.09999, 186.10001],
1063 %
1064
1065 *** troncation information derived by Astree:
1066
1067 % astree moda_dur_3.c --exec-fn main --mod \
1068 |& egrep --after-context 18 "(launched)|(WARN)|(direct =)" | tail -n 18
1069 <float-interval: entree in [-186.09999, 186.10001],
1070 RESULTAT in [-186.09999, 186.10001] >
1071 <modulo:
1072   there exists an integer i in ((entree) - 0.)/1. + [-0.5;0.50000001]
1073 such that: RESULTAT = 1.*i + [-3.3328896e-13;3.3328896e-13]
1074 >
1075 <modulo:
1076   tronc_entier = Arr_0(((entree) - 0.)/1. + [0.;0.]) + [-0.;0.]
1077 there exists an integer i in ((entree) - 0.)/1. + [-0.5;0.50000001]
1078 such that: plus_proche = 1.*i + [-3.3328896e-13;3.3328896e-13]
1079 there exists an integer i in ((entree) - 0.)/1. + [-1.;8.2645002e-14]
1080 such that: reste=entree - 1.*i + [-1.6600055e-13;1.6600055e-13]
1081 there exists an integer i in ((entree) - 0.)/1. + [-1.;8.2645002e-14]
1082 such that: plancher = i + [-4.1633364e-14;4.1633364e-14]
1083 troncature = Arr_0(((entree) - 0.)/1. + [0.;0.]) + [-0.;0.]
1084 rapport=((entree) - 0.)/1. + [-8.2645002e-14;8.2645002e-14]
1085 there exists an integer i in ((entree) - 0.)/1. + [-0.5;0.50000001]
1086 such that: RESULTAT = 1.*i + [-3.3328896e-13;3.3328896e-13]
1087 %
1088
1089 *****
1090 * Astree knows about synchronous programming... *
1091 *****
1092 %
1093
1094 *** incorrect example:
1095
1096 % cat -n clock-error.c
1097   1 /* clock-error.c */
1098   2 int R, T, n = 10;
1099   3 void main()
1100   4 { volatile int I;
1101   5   R = 0;
1102   6   while (1) {
1103   7     if (I)
1104   8       { R = R+1; }
1105   9     else
1106  10       { R = 0; }
1107  11     T = (R>=n);
1108  12 /* __ASTREE_wait_for_clock(( )); */
1109  13 }}
1110 %
1111
1112 *** configuration file:
1113
1114 % cat -n clock-error.config
1115   1 /* clock-error.config */
1116   2 __ASTREE_volatile_input((I [0,1]));
1117 %
1118

```



```

1119 *** analysis of the incorrect example by Astree:
1120
1121 % astree --exec-fn main --config-sem clock-error.config clock-error.c |& egrep
... "(launched)|(WARN)"
1122
1123 clock-error.c:8.12-15::[call#main@3:loop@6>=4:]: WARN: signed int arithmetic range [-2147483647,
... 2147483648] not included in [-2147483648, 2147483647]
1124 %
1125
1126 *** correct example:
1127
1128 % cat -n clock.c
1129     1 /* clock.c */
1130     2 int R, T, n = 10;
1131     3 void main()
1132     4 { volatile int I;
1133     5   R = 0;
1134     6   while (1) {
1135     7     if (I)
1136     8       { R = R+1; }
1137     9     else
1138    10       { R = 0; }
1139    11     T = (R>=n);
1140    12     __ASTREE_wait_for_clock();
1141    13 }}
1142 %
1143
1144 *** correction (difference with the incorrect program):
1145
1146 % diff clock-error.c clock.c
1147 1c1
1148 < /* clock-error.c */
1149 ---
1150 > /* clock.c */
1151 12c12
1152 < /* __ASTREE_wait_for_clock(); */
1153 ---
1154 > __ASTREE_wait_for_clock();
1155 %
1156
1157 *** configuration file:
1158
1159 % cat -n clock.config
1160     1 /* clock.config */
1161     2 __ASTREE_volatile_input((I [0,1]));
1162     3 __ASTREE_max_clock((3600000));
1163 %
1164
1165 *** analysis of the correct example by Astree:
1166
1167 % astree --exec-fn main --config-sem clock.config clock.c |& egrep "(launched)|(WARN)"
1168
1169 %
1170
1171 *****
1172 * Astree knows about control/command theory... *
1173 *****
1174 %
1175
1176 *** filter example:
1177
1178 % cat -n filtre.c
1179     1 typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
1180     2 BOOLEAN INIT;
1181     3 float P, X;
1182     4 volatile float RANDOM_INPUT;
1183     5 __ASTREE_volatile_input((RANDOM_INPUT [-10.0,10.0]));
1184     6
1185     7 void filtre2 () {
1186     8   static float E[2], S[2];
1187     9   if (INIT) {

```

```

1188     10     S[0] = X;
1189     11     P = X;
1190     12     E[0] = X;
1191     13     } else {
1192     14     P = (((((0.4677826 * X) - (E[0] * 0.7700725)) + (E[1] * 0.4344376)) + (S[0] *
... 1.5419)) - (S[1] * 0.6740477));
1193     15     }
1194     16     E[1] = E[0];
1195     17     E[0] = X;
1196     18     S[1] = S[0];
1197     19     S[0] = P;
1198     20 }
1199     21
1200     22 void main () {
1201     23     X = RANDOM_INPUT;
1202     24     INIT = TRUE;
1203     25     while (TRUE) {
1204     26     X = RANDOM_INPUT;
1205     27     filtre2 ();
1206     28     INIT = FALSE;
1207     29     }
1208     30 }
1209 %
1210
1211 *** static analysis by Astree (1 -- WITH 2nd order
1212 *** filter domain):
1213
1214 % astree filtre.c --dump-invariants --exec-fn main |& egrep "(launched)|(WARN)|(P in)"
1215
1216     X in [-10., 10.], P in [-13.388092, 13.388093],
1217 %
1218
1219 *** static analysis by Astree (2 -- WITHOUT 2nd order
1220 *** filter domain):
1221
1222 % astree filtre.c --exec-fn main --no-filters --dump-invariants |& egrep "(launched)|(WARN)|(P
... in)"
1223
1224 filtre.c:14.6-114.: [call#main@22:loop@25>=4:call#filtre2@27:]: WARN: double->float conversion
... range [-inf., inf.] not included in [-3.4028234e+38, 3.4028235e+38]
1225     P in [-3.4028234e+38, 3.4028235e+38], RANDOM_INPUT in [-10., 10.] >
1226 %
1227
1228 *****
1229 * Astree can analyze low level memory operations *
1230 *****
1231 %
1232
1233 *** example 1 (pointer casts):
1234
1235 % cat -n memcpy.c
1236     1 /* memcpy.c (polymorphic memcpy) */
1237     2
1238     3 /* byte per byte copy of src into dst */
1239     4 void memcpy(char* dst, const char* src, unsigned size)
1240     5 {
1241     6     int i;
1242     7     for (i=0;i<size;i++) dst[i] = src[i];
1243     8 }
1244     9
1245    10 void main()
1246    11 {
1247    12     float x = 10.0, y;
1248    13     int zero = 0;
1249    14     /* copy of x into y (well-typed) */
1250    15     memcpy(&y,&x,sizeof(y));
1251    16     __ASTREE_assert((y==10.0));
1252    17     /* copy of zero into y (not well-typed but allowed in C) */
1253    18     memcpy(&y,&zero,sizeof(y));
1254    19     __ASTREE_assert((y==0.0));
1255    20 }

```

```

1256 %
1257
1258 *** static analysis by Astree:
1259
1260 % astree --exec-fn main --unroll 5 memcpy.c |& egrep "(launched)|(WARN)"
1261
1262 %
1263
1264 *** example 2 (unions):
1265
1266 % cat -n union.c
1267     1 /* union.c (union type) */
1268     2
1269     3 union {
1270     4     int type;
1271     5     struct { int type; int data; } A;
1272     6     struct { int type; char data[3]; } B;
1273     7 } u;
1274     8
1275     9 void main()
1276    10 {
1277    11     /* no assert failure */
1278    12     u.type = 12;
1279    13     __ASTREE_assert((u.A.type==12));
1280    14     __ASTREE_assert((u.B.type==12));
1281    15
1282    16     /* assert failure because the modification of u.B.data also modifies u.A.data */
1283    17     u.A.data = 0;
1284    18     u.B.data[0] = 12;
1285    19     __ASTREE_assert((u.A.data==0));
1286    20 }
1287 %
1288
1289 *** static analysis by Astree:
1290
1291 % astree --exec-fn main --full-memory-model union.c |& egrep "(launched)|(WARN)"
1292
1293 union.c:19.19-30::[call#main@9:]: WARN: assert failure
1294 %
1295
1296
1297 *****
1298 * Astree has a graphic interface under X11... *
1299 *****
1300 %
1301
1302 *** static analysis by Astree
1303
1304 % astree filtre.c --dump-invariants --exec-fn main --export-invariant stat \
1305     --export-file filtre.inv --export-unroll >& /dev/null
1306 %
1307
1308 *** visualization of the results:
1309 %
1310
1311 % visu --text-size 14 --text-font CMTT filtre.inv >& /dev/null
1312 %
1313
1314 *** (scaling up with GTK+ (library to build graphical
1315 *** user interfaces (GUIs) originally for X Window)!)
1316 %
1317
1318 *****
1319 *** The end, thank you for your attention ***
1320 *****
1321
1322

```



Quit



Intervals



Clocks



Congruences



Trees



Octagons



Filters



Geom. dev.



Symbolics



Pointers



Structure



Help

Search string:  Next Previous First Last Goto line:  Current

```

tre.c:14:6
| # main @ filtre.c:22:5
loop @ filtre.c:25:2
iter >= 4
  call # filtre2 @ filtre.c:27:4
invariant
iter = 3
iter = 2

```

filtre.c

```

static void filtre2(int E[2], int S[2])
{
  if (INIT) {
    S[0] = X;
    P = X;
    E[0] = X;
  } else {
    P = ((((((0.4677826 * X) - (E[0] * 0.7700725)) + (E[1] * 0.4344376)) + (S[0] * 1.5419)) - (S[1] * 0.6740477)));
    E[1] = E[0];
    E[0] = X;
    S[1] = S[0];
    S[0] = P;
  }
}

```

Functions Sources

```

filtre2
main

```

Invariant info

**Second order filter with complex roots**

```

Last input      : E[1]
Previous input  : E[0]
Last output     : P
Previous output : S[1]
coef e1        : -0.7700725
coef e2        : 0.46778261
coef e3        : 0.43443761
coef a         : 1.5419001
coef b         : -0.6740477
Number of unrolling : 39
Bound on the input : 13.390565
Bound on atomic rounding errors : 5.5358767e-06
Gain for the first outputs : 6.5877578
Gain due to the last inputs : 5.3786085
Gain due to the first inputs : 0.015089829
Bound on (overall) rounding errors : 8.9995463e-05
Bound on the output : 88.213821

```