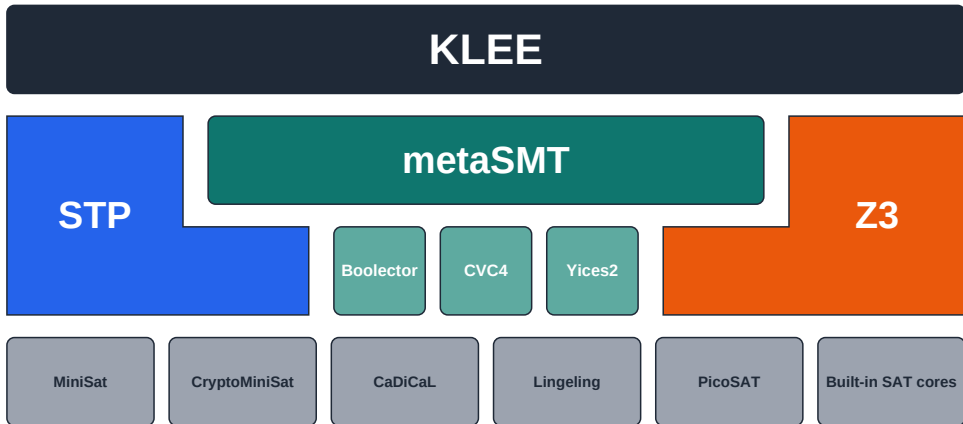


EAR: Debugging Libraries through API Call Traces

Tongjun (Albert) Zhang Cristian Cadar
Imperial College London

KLEE Workshop 2026

Solver Infrastructure of KLEE



Solvers Can Behave Unexpectedly

QF_FD: get-consequences misses a consequence that follows by unit propagation, after an earlier check-sat #10964

Closed ✓ 1 session complete #10965

Aramis Romero (aracamos) opened 2d ago

Versions: 5.1.0 (Microsoft.Z3.5.1.0.nupkg from the GitHub release, win-x64) and 4.12.2 (Microsoft.Z3 on nuget.org). Run through `Z3_eval_satlib2_string`, which is the SMT-LIB command interpreter. Not tried with the z3 executable.

```
(set-logic QF_FD)
(declare-const a0 Bool) (declare-const a5 Bool) (declare-const a2 Bool)
(declare-const x0 Bool) (declare-const x1 Bool) (declare-const x2 Bool) (decl
(declare-const x4 Bool) (declare-const x5 Bool) (declare-const x6 Bool) (decl
(declare-const y Bool)
(assert ([( _ at-most 1) a0 a1 a2 a3 a4])
(assert (or x0 x1 x2 x3 x4 x5 x6 x7))
(assert ([( _ at-most 1) x0 x1 x2 x3 x4 x5 x6 x7])
(assert (or x0 x1 x2 x3 x4 x5 x6 x7 (not y)))
(check-sat)
(get-consequences {x5} {y})
(check-sat-assuming {x5 y})
```

Expected: `get-consequences` returns `sat` with `{x5 x6 (not y)}`. With `x5` true, the `at-most-one` makes `x0-x4`, `x6` and `x7` false, and the last clause then forces `y` false.

Actual: `get-consequences` returns `sat` with no consequences. The following `check-sat-assuming {x5 y}` is `unsat`.

The consequence is reported if any one of these is changed:

- remove the first `(check-sat)`
- remove the unrelated `([( _ at-most 1) a0 ... a4])`

Assignees: Nikolaj Björner (NikolajBjörner) Copilot ✓

Labels: No labels

Type: No type

Projects: No projects

Milestone: No milestone

Relationships: None yet

Development: Preserve QF_FD consequences after a prior check-sat Z3Prover/z3

Labels: Nightly Pre-release

Solvers Can Behave Unexpectedly

3ZProver / z3 · 13
Q X D + ↻ ⌂ 🔒

[Code](#)
[Issues 44](#)
[Pull reqs](#)

[stp / stp](#)
🔍 🗑️ ↻ ⌂ 🔒

[Q](#)
[🗑️](#)
[↻](#)
[⌂](#)
[🔒](#)

QF_FD: get-consequence that fails after an earlier check

Closed ✓ 1 session complete

Aramis Romero [aracamos] opened 2 months ago

Versions: 5.1.0 (Microsoft.Z3.Solver.5.1.0.nupkg (Microsoft.Z3 on nuget.org). Run through LIB command interpreter. Not tried with .NET)

```

{set-logic QF_FD}
(declare-const x0 Bool) (declare-const x0 Bool) (declare-const x0 Bool)
(declare-const x1 Bool) (declare-const x1 Bool) (declare-const x1 Bool)
(declare-const x2 Bool) (declare-const x2 Bool) (declare-const x2 Bool)
(declare-const x3 Bool) (declare-const x3 Bool) (declare-const x3 Bool)
(assert (( _at-most 1) a0 a1 a2 a3))
(assert ((or x0 x1 x2 x3 x4 x5 x6 x7)
(assert (( _at-most 1) x0 x1 x2 x3 :
(assert ((or x0 x1 x2 x3 x4 x5 x6 x7) (=
(check-sat)
(get-consequences {x5}) {y}))
(check-sat-assuming {x5 y}))
  
```

Expected: get-consequences returns sat at most-one makes x0-x4, x6 and x7 false

Actual: get-consequences returns sat ; sat-assuming {x5 y} is unsat.

The consequence is reported if any one

- remove the first (check-sat)
- remove the unrelated ((_at-most 1

Division by zero bug #206

Closed

ccdar opened on Dec 2, 2015
Contributor · ...

Using KLEE, we found a crash bug in STP related to the way it handles division by zero. The bug is reported at [klee/klee#308](#), but I repeat it below.

The problematic query, in SMT2 format, is the following:

```

; query 6 -- To verify: Validity, Instructions: 83
{set-logic QF_AUFBV}
(declare-fun unnamed_1 ((Array (_ BitVec 32) (_ BitVec 8))) )
(assert (!let ( ?b1 const (select unnamed_1 bvi 32) ) (select unnamed_1
(check-sat)
(exit)
; FAIL -- Elapsed: 1.298289e-01
; Failure reason: SOLVER PROCESS INTERRUPTED
  
```

This doesn't crash STP from the command line, but when the query is constructed in KLEE, the following error trace is produced:

```

Fatal Error: BVConstEvaluator::division by zero error
error: STL Error: BVConstEvaluator::division by zero error
0 LBLWV-3.4.so.1 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
1 LBLWV-3.4.so.1 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
2 libpthread.so.0 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
3 libc.so.6 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
4 libc.so.6 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
5 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
6 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
7 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
8 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
9 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
10 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
11 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
12 klee 0x00002c0bc54d9042 lvm::js::PrintStackTrace(_JL_IO_FILE*)
  
```

Solvers Can Behave Unexpectedly

QF_FD: get-consequences returns sat after an earlier check

Closed

1 session complete

Aramis Romero (aracamos) opened 2

Versions: 5.1.0 (Microsoft.Z3.5.1.0.nupkg (Microsoft.Z3 on nuget.org). Run through LIB command interpreter. Not tried with i

```
(set-logic QF_FD)
(declare-const x0 Bool) (declare-const x1 Bool) (declare-const x2 Bool) (declare-const x3 Bool) (declare-const x4 Bool) (declare-const x5 Bool)
(assert ([( _at-most 1) x0 x1 x2 x3 x4 x5 x6 x7])
(assert (or x0 x1 x2 x3 x4 x5 x6 x7)
(assert ([( _at-most 1) x0 x1 x2 x3]
(assert (or x0 x1 x2 x3 x4 x5 x6 x7)
(assert (or x0 x1 x2 x3 x4 x5 x6 x7)
(check-sat)
(get-consequences (x5) (y))
(check-sat-assuming (x5 y))
```

Expected: get-consequences returns sat at-most-one makes x0-x4, x6 and x7 false

Actual: get-consequences returns sat-sat-assuming (x5 y) is unsat.

The consequence is reported if any one

- remove the first (check-sat)
- remove the unrelated ([(_at-most :

Division by zero bug

Closed

ccadar opened on Dec 2, 2015

Using KLEE, we found a crash bug in STP. The bug is reported at [klee/klee#308](#), but the problematic query, in SMT2 format, is

```
; Query 6 -- Type: Validity, Instru
(set-logic QF_AUFBV )
(declare-fun unnamed () (Array (_ &
(assert (let ( ?b1 concat (select
(check-sat)
(exit)
; FAIL -- Elapsed: 1,298,298e-01
; Failure reason: SOLVER PROCESS
```

This doesn't crash STP from the command line, the following error trace is produced

```
Fatal Error: BVCConstEvaluator::divide
error: STP Error: BVCConstEvaluator::
0 libLLVM-3.4.so.1 0x00002b8c545f
1 libLLVM-3.4.so.1 0x00002b8c545e
2 libpthread.so.0 0x00002b8c5f25
3 libc.so.6 0x00002b8c60b9
4 libc.so.6 0x00002b8c60b9
5 klee 0x000000000004ef
6 klee 0x0000000000054d
7 klee 0x000000000005a7
8 klee 0x000000000005a8
9 klee 0x000000000005a8
10 klee 0x00000000000552
11 klee 0x0000000000054e
12 klee 0x000000000005c0
```

Non-determinism in STP and C API #218

Closed

ccadar opened on May 25, 2016

Contributor

Hi, we're trying to understand the sources of non-determinism in STP. More exactly, is it enough to use the "--seed" option to get STP to execute deterministically on a certain query?

Related to this, there seems to be no C API for the "--seed" command-line option. Would it be possible to add this?

(adding @andreamattavelli and @MartinNowack who are also interested in this issue)

Thanks!

rgov on May 25, 2016

Member

Do you want the same seed on multiple hosts (with potentially different architectures, operating systems, etc.) to result in identical behavior? Or just relative to one particular host?

On 25 May 2016, at 15:51, Cristian Cadar notifications@github.com wrote:

Hi, we're trying to understand the sources of non-determinism in STP. More exactly, is it enough to use the "--seed" option to get STP to execute deterministically on a certain query?

Related to this, there seems to be no C API for the "--seed" command-line option. Would it be possible to add this?

(adding @andreamattavelli and @MartinNowack who are also interested in

Assignees

msoos

Labels

enhancement wontfix

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Notifications

Customize

Subscribe

You're not receiving notifications from this thread.

Participants

4

Solvers Can Behave Unexpectedly

The image shows a collage of GitHub discussions and an email thread related to solver behavior. The top row features three GitHub discussion snippets, and the bottom row shows an email thread.

GitHub Discussion 1: QF_FD: get-consequence

consequence that fails after an earlier check

Closed 1 session complete

Aramis Romero (aracamos) opened 2

Versions: 5.1.0 (Microsoft.Z3.5.1.0.nupkg (Microsoft.Z3 on nuget.org). Run through LIB command interpreter. Not tried with)

```
(set-logic QF_FD)
(declare-const x0 Bool) (declare-const x1 Bool) (declare-const x2 Bool) (declare-const x3 Bool) (declare-const x4 Bool) (declare-const x5 Bool)
(assert (! (at-most 1) x0 x1 x2 x3 x4 x5 x6 x7))
(assert (or x0 x1 x2 x3 x4 x5 x6 x7))
(assert (! (at-most 1) x0 x1 x2 x3))
(assert (or x0 x1 x2 x3 x4 x5 x6 x7) (check-sat))
(get-consequences (x5) (y))
(check-sat-assuming (x5 y))
```

Expected: get-consequences returns sat at-most-one makes x0-x4, x6 and x7 false

Actual: get-consequences returns sat-sat-assuming (x5 y) is unsat.

The consequence is reported if any one

- remove the first (check-sat)
- remove the unrelated (! (at-most 1

GitHub Discussion 2: Division by zero but

Closed

ccadar opened on Dec 2, 2015

Using KLEE, we found a crash bug in STP. The bug is reported at [klee/klee#308](#), but

The problematic query, in SMT2 format, is

```
; Query 6 -- Type: Validity, Instru
(set-logic QF_AUFBV)
(declare-fun unnamed () (Array (_ &
(assert (let ( ?b1 concat (select
(check-sat)
(exit)
; FAIL -- Elapsed: 1,298,298-81
; Failure reason: SOLVER PROCESS
```

This doesn't crash STP from the command KLEE, the following error trace is produced

Fatal Error: BVConstEvaluator::divide error: STP Error: BVConstEvaluator::

```
0 libLLVM-3.4.so.1 0x00002b8c545f5
1 libLLVM-3.4.so.1 0x00002b8c545e
2 libpthread.so.0 0x00002b8c5f25
3 libc.so.6 0x00002b8c60b9
4 libc.so.6 0x00002b8c60b9
5 klee 0x000000000004ef
6 klee 0x0000000000054f
7 klee 0x000000000005a7
8 klee 0x000000000005a7
9 klee 0x000000000005a7
10 klee 0x00000000000552
11 klee 0x0000000000054e
12 klee 0x000000000005c0
```

GitHub Discussion 3: Non-determinism in

Closed

ccadar opened on May 25, 2016

Hi, we're trying to understand the source it enough to use the "--seed" option to get certain query?

Related to this, there seems to be no CA Would it be possible to add this?

(adding @andreasmattavelli and @Marti issue)

Thanks!

rgov on May 25, 2016

Do you want the same seed on multiple operating systems, etc., to result in identical particular host?

On 25 May 2016, at 15:51, Cristian Cad

Hi, we're trying to understand the source exactly, is it enough to use the "--seed" deterministically on a certain query?

Related to this, there seems to be no CA Would it be possible to add this?

(adding @andreasmattavelli and @M

Email Thread: Re: [klee-dev] Reproduce solver time-out cases

Summarize this email

On 13/09/2025 23:16, Jie wrote:

- > Dear KLEE Team,
- >
- >
- >
- > I'm writing to ask about time-out queries generated by KLEE. I'm trying
- > to identify program paths that cause the solver to time out. To dump the
- > queries, I used the latest KLEE master branch with the following command:
- >
- > |klee --search=dfs --posix-runtime --write-kqueries --libc=uclicb
- > --max-solver-time=10s --max-time=600s lua.bc A -sym-files 1 40|
- >
- > Here is the output I received, which is associated to test000002
- > (confirmed by the test000002.solver.err).
- >
- > [KLEE: ERROR: lstring.c:197: Query timed out (resolve.)]
- >
- > When I run kleeaver to reproduce the timeout, I get a SAT result instead.
- > My exact command is below.
- >
- > |kleeaver --max-solver-time=10s /tmp/out/lua-ttt/test000002.kquery|
- >
- >
- > I've searched the email archive but couldn't find relevant answers to my
- > question. Could you please let me know how KLEE handles dumping queries
- > on solver time-outs? Am I missing any options or settings? Thank you for
- > your help.

Heisenbugs?

```
(assert (let ( (?B1 (concat (select unnamed (_ bv1 32) ) (select unnamed (.  
(check-sat)  
(exit)  
;   FAIL -- Elapsed: 1.298230e-01  
;   Failure reason: SOLVER PROCESS INTERRUPTED
```

This doesn't crash STP from the command line, but when the query is constructed in KLEE, the following error trace is produced:

```
Fatal Error: BVConstEvaluator:division by zero error  
error: STP Error: BVConstEvaluator:division by zero error  
0 libLLVM-3.4.so.1 0x00002b0bc545f042 llvm::sys::PrintStackTrace(_IO_FILE*)
```



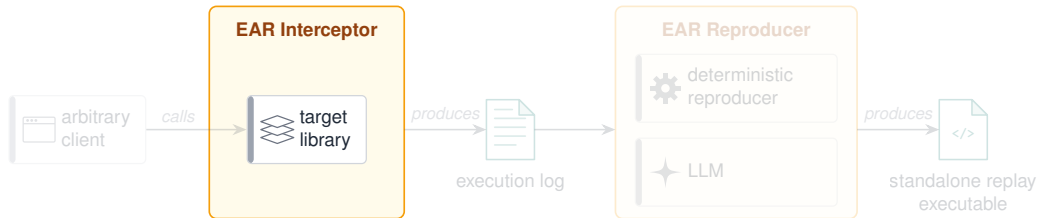
- > Here is the output I received, which is associated to test000002
- > (confirmed by the test000002.solver.err).
- >
- > |KLEE: ERROR: lstring.c:197: Query timed out (resolve).|
- >
- > When I run kleaver to reproduce the timeout, I get a SAT result instead.
- > My exact command is below.
- >
- > |kleaver --max-solver-time=10s /tmp/out/luat-tt/test000002.kquery|
- >

EAR: ExAct Reproducer

A dynamic tracer for call sequence observability



EAR Interceptor

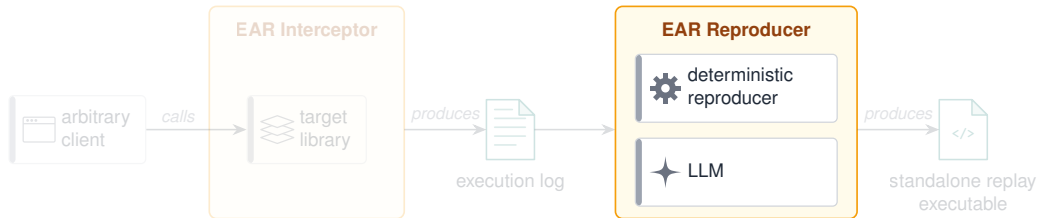


EAR Interceptor

Produces human-readable call traces

```
...
=== Function Return Intercepted ===
Function: vc_push
Return type: void
--- Argument State at Return ---
Argument count: 1
Argument types: ptr
Arguments (dereferenced at return):
    <0> (ptr) = ptr_873628
=====
=== Function Call Intercepted ===
Function: vc_bvConstExprFromInt
Argument types: ptr, i32, i32
Arguments (with recursive dereferencing):
    <0> (ptr) = ptr_873628
    <1> (i32) = 32 (0x20)
    <2> (i32) = 0 (0x0)
=====
=== Function Return Intercepted ===
Function: vc_bvConstExprFromInt
Return value: ptr_2946929 (FIRST SEEN)
...
```

EAR Reproducer



EAR Reproducer

Reverse Engineered Standalone Client

- Test case reduction
- Standalone reproducer

```
...  
Expr ptr_2945627 = vc_readExpr(ptr_873628, ptr_2878899_1, ptr_2945624_2);  
Expr ptr_2934112_38 = vc_bvConstExprFromInt(ptr_873628, 32, 41);  
vc_push(ptr_873628);  
Expr ptr_2946929 = vc_bvConstExprFromInt(ptr_873628, 32, 0);  
Expr ptr_2936572_725 = vc_bvConstExprFromInt(ptr_873628, 32, 72);  
Expr ptr_2940972_42 = vc_readExpr(ptr_873628, ptr_2878899_1, ptr_2936572_725);  
vc_DeleteExpr(ptr_2936572_725);  
...
```

Case Study: STP Crash

stp / stp

Code Issues Pull requests Agents Discussions Actions More

Division by zero bug #206

Closed

New issue

ccadar opened on Dec 2, 2015

Contributor

Using KLEE, we found a crash bug in STP related to the way it handles division by zero. The bug is reported at [klee/klee#308](#), but I repeat it below.

The problematic query, in SMT2 format, is the following:

```
; Query 0 -- Type: Validity, Instructions: 83
(set-logic QF_AUFBV)
(declare-fun unnamed () (Array (_ BitVec 32) (_ BitVec 8) ))
(assert (let ( { ?b1 (concat (select unnamed (_ bv1 32) ) (select unnamed (
(check-sat)
(exit)
; FAIL -- Elapsed: 1.298238e-01
; Failure reason: SOLVER PROCESS INTERRUPTED
```

This doesn't crash STP from the command line, but when the query is constructed in KLEE, the following error trace is produced:

```
Fatal Error: BVConstEvaluator:division by zero error
error: STP Error: BVConstEvaluator:division by zero error
0 libLLVM-3.4.so.1 0x00002b8bc545f842 llvm::sys::PrintStackTrace(_IO_FILE*)
1 libLLVM-3.4.so.1 0x00002b8bc545ee34
2 libpthread.so.0 0x00002b8bc5f25340
3 libc.so.6 0x00002b8bc6bb9cc9 gsignal + 57
4 libc.so.6 0x00002b8bc6bb9d88 abort + 328
5 klee 0x00000000004efda9
6 klee 0x0000000000549397 stp::FatalError(char const*) + 55
7 klee 0x00000000005a7657
8 klee 0x00000000005a8f80 stp::NonMemberBVConstEvaluator(stp::ST
9 klee 0x00000000005a9370 stp::NonMemberBVConstEvaluator(stp::AS
10 klee 0x0000000000553332 SimplifyingNodeFactory::CreateTerm(stp
11 klee 0x000000000054e5b7 NodeFactory::CreateArrayTerm(stp::Kind
12 klee 0x00000000005cc64a stp::SubstitutionMap::replace(stp::AST
```

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Notifications

Customize

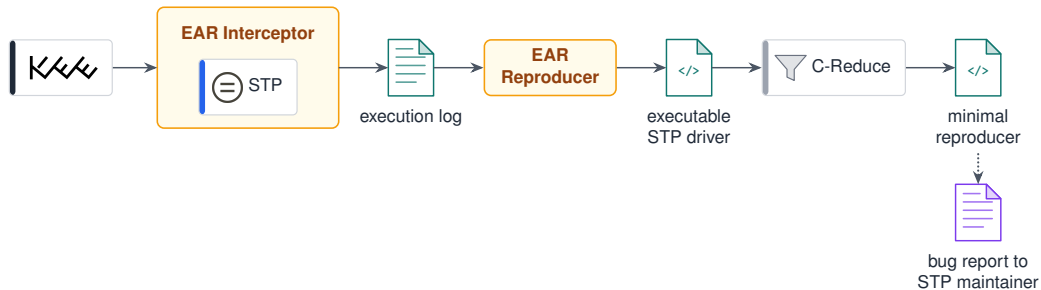
Subscribe

You're not receiving notifications from this thread.

Participants

<https://github.com/stp/stp/issues/206>

Case Study: STP Crash



Case Study: STP Crash

```
int b, a, g;
int *c = &b, *d = &b, *f = &a;
int safe_div_func_uint16_t_u_u(short p1, int p2) { return p2 == 0 ? p1 : p2; }

int main() {
    klee_make_symbolic(&b, sizeof b);
    if (safe_div_func_uint16_t_u_u(*c, 0))
        *f = (int)&b % *c;
    safe_div_func_uint16_t_u_u(a && g, *d);
}
```

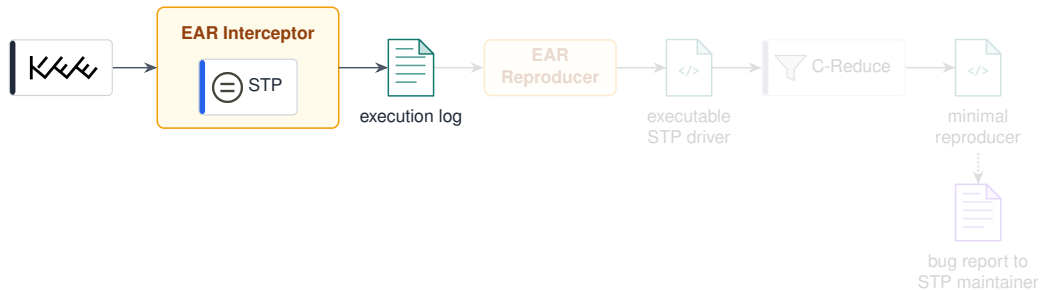
Fatal Error: BVConstEvaluator:division by zero error

Case Study: STP Crash

```
; Query 6 -- Type: Validity, Instructions: 83
(set-logic QF_AUFBV )
(declare-fun unnamed () (Array (_ BitVec 32) (_ BitVec 8) ) )
(assert
  (let ((?B1 (concat (select unnamed (_ bv1 32))
                     (select unnamed (_ bv0 32)))))
    (let ((?B2 (concat (select unnamed (_ bv3 32))
                     (concat (select unnamed (_ bv2 32)) ?B1))))
      (and (and (= false (= (_ bv0 32) ?B2))
              (= false (= (_ bv0 16) ?B1)))
           (= (_ bv0 32) (bvsrem (_ bv16864560 32) ?B2))))))
(check-sat)
(exit)
;   FAIL -- Elapsed: 1.298230e-01
;   Failure reason: SOLVER PROCESS INTERRUPTED
```

STP's command line: sat. Through the C API: sat. No crash.

Case Study: STP Crash



Case Study: STP Crash

Traced: 87 STP API calls KLEE made

ptr_34769 = 16864560, ptr_9669 = ?B2, ptr_29485 = the validity checker

=== Function Call Intercepted ===

Function: vc_sbvRemExpr

Return type: ptr

// ...

<2> (ptr) = ptr_34769 -> (UNCHANGED)

<3> (ptr) = ptr_9669 -> (UNCHANGED)

=====

// ...

=== Function Return Intercepted ===

Function: vc_notExpr

Return value: ptr_34803 (FIRST SEEN) (alloc size: 8)

--- Argument State at Return ---

Argument count: 2

Argument types: ptr, ptr

Arguments (dereferenced at return):

<0> (ptr) = ptr_29485 -> (UNCHANGED)

<1> (ptr) = ptr_29736 -> (UNCHANGED)

=====

// ...

=== Function Call Intercepted ===

Function: vc_query

Return type: i32

Arguments (with recursive dereferencing):

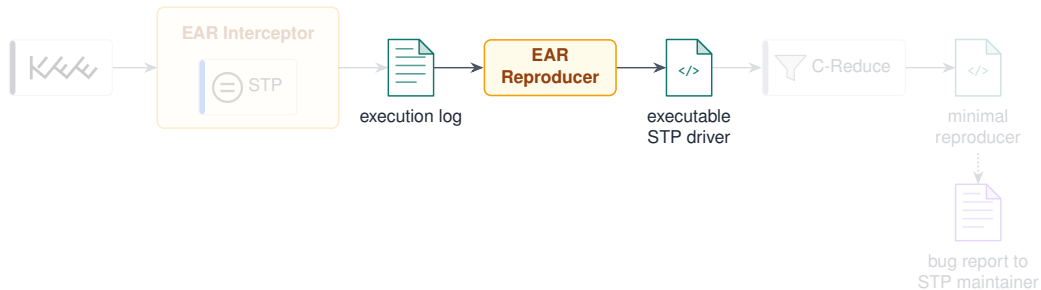
<0> (ptr) = ptr_29485 -> (UNCHANGED)

<1> (ptr) = ptr_34803 -> (UNCHANGED)

=====

// EOF

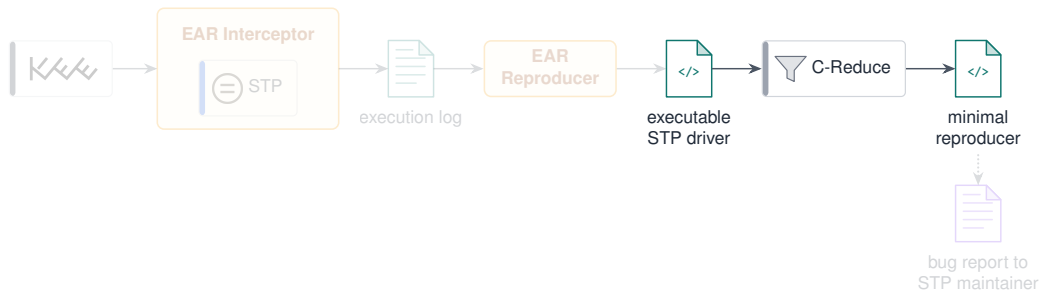
Case Study: STP Crash



Reproduced: the trace as a standalone C program with 87 calls

[illegible][illegible][illegible]

Case Study: STP Crash



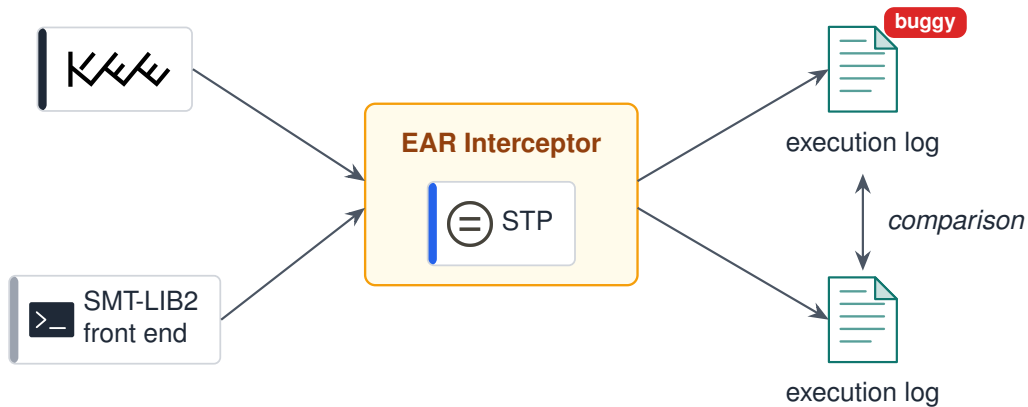
Case Study: STP Crash

CReduce-d to 36 calls, the same abort at the same place

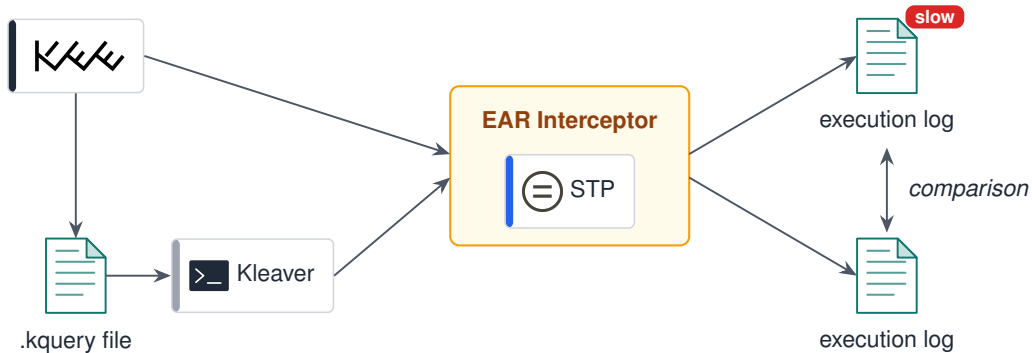
```
#include <stdlib.h>
#include <stp/c_interface.h>
#include <string.h>
int main() {
    VC a = vc_createValidityChecker();
    Type b = vc_bvType(a, 32);
    Type c = vc_bvType(a, 8);
    Type d = vc_arrayType(a, b, c);
    char e[8];
    strncpy(e, "", 8);
    Expr f = vc_varExpr(a, e, d);
    char *g = malloc(33), *j = malloc(33), *l = malloc(33),
        *n = malloc(33), *q = malloc(33), *au = malloc(33),
        *x = malloc(33), *z = malloc(33), *af = malloc(33),
        *ai = malloc(33), *al = malloc(33);
    strncpy(g, "00000000000000000000000000000000", 33);
    Expr i = vc_bvConstExprFromStr(a, g);
    strncpy(j, "00000001000000010101010100110000", 33);
    Expr k = vc_bvConstExprFromStr(a, j);
    strncpy(l, "000000000000000000000000000000011", 33);
    Expr m = vc_bvConstExprFromStr(a, l);
    Expr at = vc_readExpr(a, f, m);
    strncpy(n, "000000000000000000000000000000010", 33);
    Expr o = vc_bvConstExprFromStr(a, n);
    Expr p = vc_readExpr(a, f, o);
    strncpy(q, "000000000000000000000000000000001", 33);
    Expr r = vc_bvConstExprFromStr(a, q);
    Expr s = vc_readExpr(a, f, r);
    strncpy(au, "00000000000000000000000000000000", 33);
```

```
    Expr aa = vc_bvConstExprFromStr(a, au);
    Expr ab = vc_readExpr(a, f, aa);
    Expr ac = vc_bvConcatExpr(a, s, ab);
    Expr t = vc_bvConcatExpr(a, p, ac);
    Expr u = vc_bvConcatExpr(a, at, t);
    Expr v = vc_sbvRemExpr(a, 32, k, u);
    Expr w = vc_eqExpr(a, i, v);
    vc_assertFormula(a, w);
    strncpy(x, "00000000000000000000000000000000", 33);
    Expr y = vc_bvConstExprFromStr(a, x);
    strncpy(z, "000000000000000000000000000000011", 33);
    Expr ad = vc_bvConstExprFromStr(a, z);
    Expr ae = vc_readExpr(a, f, ad);
    strncpy(af, "000000000000000000000000000000010", 33);
    Expr ag = vc_bvConstExprFromStr(a, af);
    Expr ah = vc_readExpr(a, f, ag);
    strncpy(ai, "000000000000000000000000000000001", 33);
    Expr aj = vc_bvConstExprFromStr(a, ai);
    Expr ak = vc_readExpr(a, f, aj);
    strncpy(al, "000000000000000000000000000000000", 33);
    Expr am = vc_bvConstExprFromStr(a, al);
    Expr an = vc_readExpr(a, f, am);
    Expr ao = vc_bvConcatExpr(a, ak, an);
    Expr ap = vc_bvConcatExpr(a, ah, ao);
    Expr aq = vc_bvConcatExpr(a, ae, ap);
    Expr ar = vc_eqExpr(a, y, aq);
    Expr as = vc_notExpr(a, ar);
    vc_query(a, as);
}
```

Differential Debugging



Differential Debugging



Summary

- EAR: an API call tracer
- Extracts standalone library bug reproducer
- Analyze problematic call patterns



EAR: TESTING AND DEBUGGING LIBRARIES THROUGH API CALL TRACES

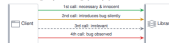
Tongjun (Albert) Zhang Cristian Cadar

t.zhang1@imperial.ac.uk c.cadar@imperial.ac.uk
Department of Computing, Imperial College London

The Need for an API Call Tracer

Debugging software library bugs triggered by client code is challenging, as isolating the library execution from that of the client's is often non-trivial.

- Proceeding library API calls set up the context
- The last call does not trigger the bug by itself
- Manually isolating the entire call trace is labor-intensive
- Certain bugs stop manifesting upon retrying



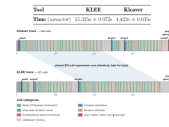
Case Study: Diagnosing Performance Issue with EAB

KLEK uses the STP Solver

- Some queries are unexpectedly slow with KLEE
- Some query replaced with Klever is no longer problematic

[illegible]

STP runs 3.5 times faster on average when the query is constructed by Klever instead of KLEE.



Our Approach

ErAct Reproductor (EAR) has two components:

- **Interceptor** records the precise sequence of API calls made by the client to the target library
- **Reproducer** consists self-contained programs that have no client dependencies, open opportunities for automated test case reduction



Differential Debugging:



Memory Object Introspection

[illegible]