Map2Check Using LLVM and KLEE (Competition Contribution)

Rafael Menezes, **Herbert Rocha**, Lucas Cordeiro and Raimundo Barreto

ETAPS EUROPEAN JOINT CONFERENCES ON THEORY & PRACTICE OF SOFTWARE

TACAS 2018

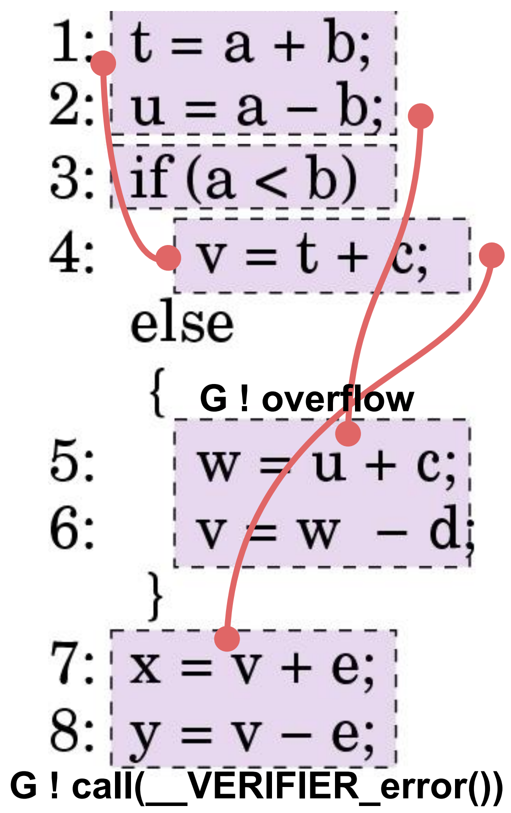
Competition on Software Verification (SV-COMP)

Map2Check

- ✓ Map2Check automatically generates and checks assertions from **safety properties** related to:
  - unreachability of an error location
  - arithmetic overflow
  - invalid deallocation
  - invalid pointers
  - memory leaks

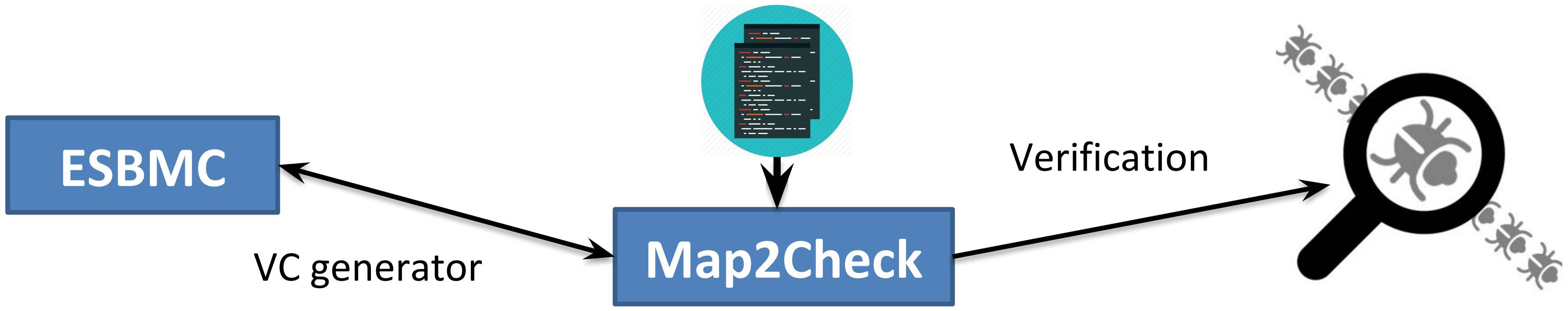
- ✓ Map2Check adopts **source code instrumentation to:**
  - monitor the program's executions
  - validate assertions with **safety properties**

Metadata



Map2Check

Old Map2Check



New Map2Check

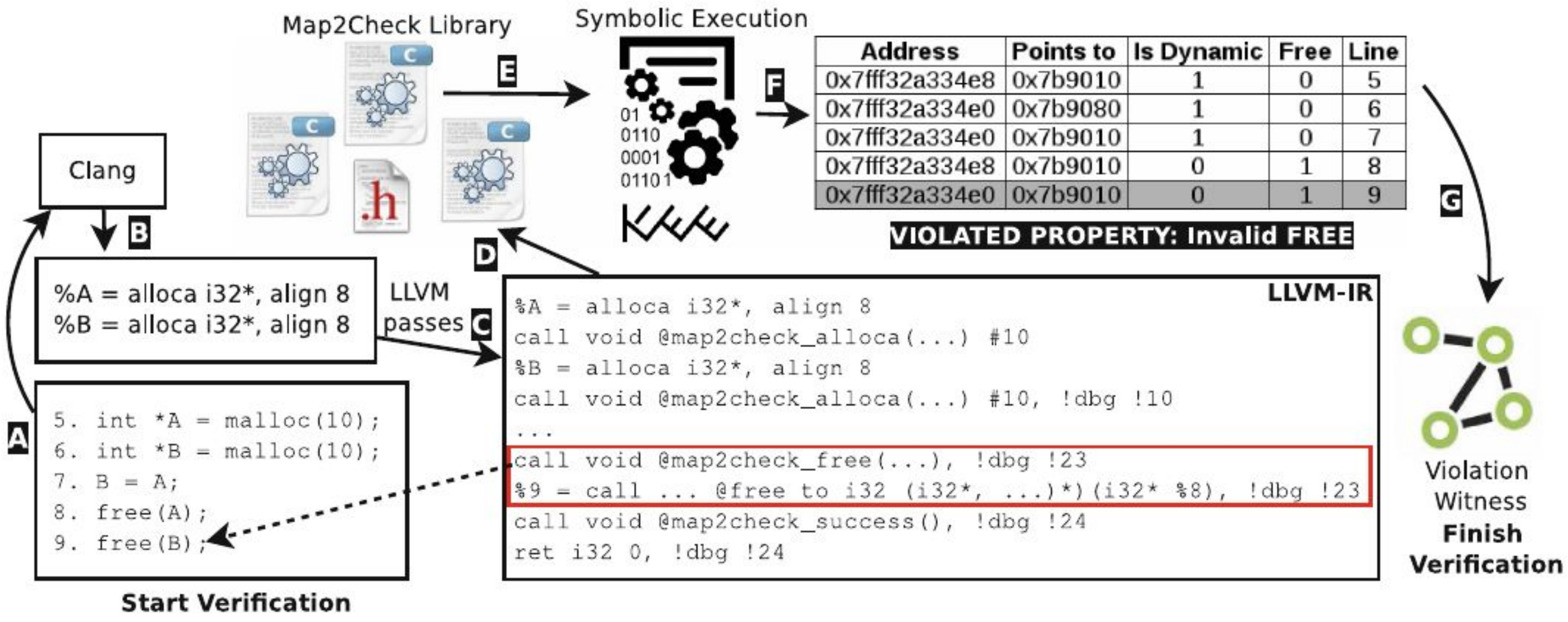
Clang

Frontend

Code Transformation

Symbolic Execution

Map2Check - Verification Approach



Address	Points to	Is Dynamic	Free	Line
0x7fff32a334e8	0x7b9010	1	0	5
0x7fff32a334e0	0x7b9080	1	0	6
0x7fff32a334e0	0x7b9010	1	0	7
0x7fff32a334e8	0x7b9010	0	1	8
0x7fff32a334e0	0x7b9010	0	1	9

VIOLATED PROPERTY: Invalid FREE

```
5. int *A = malloc(10);
6. int *B = malloc(10);
7. B = A;
8. free(A);
9. free(B);
```

```
%A = alloca i32*, align 8
%B = alloca i32*, align 8
```

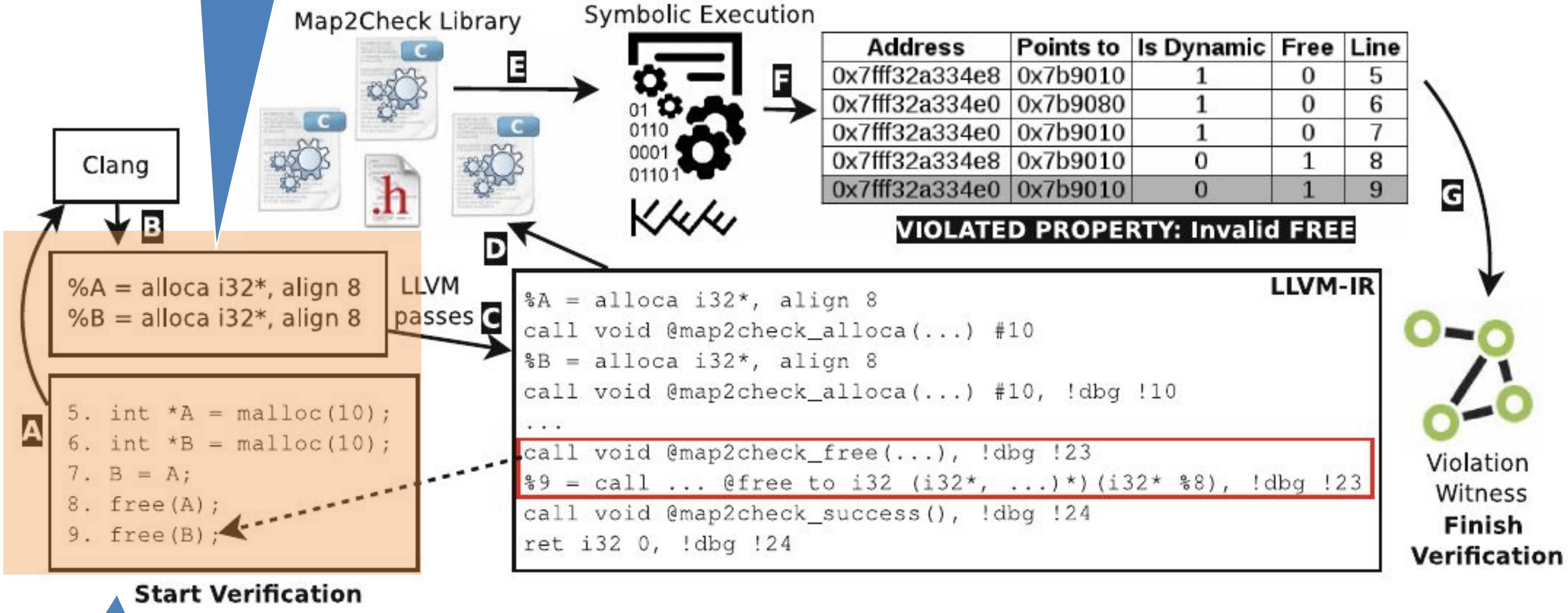
```
%A = alloca i32*, align 8
call void @map2check_alloca(...) #10
%B = alloca i32*, align 8
call void @map2check_alloca(...) #10, !dbg !10
...
call void @map2check_free(...), !dbg !23
%9 = call ... @free to i32 (i32*, ...)*)(i32* %8), !dbg !23
call void @map2check_success(), !dbg !24
ret i32 0, !dbg !24
```

Flow

Map2Check - Verification Approach

dead code elimination and constant propagation

Code optimization



```
%A = alloca i32*, align 8
%B = alloca i32*, align 8
```

```
5.  int *A = malloc(10);
6.  int *B = malloc(10);
7.  B = A;
8.  free(A);
9.  free(B);
```

Address	Points to	Is Dynamic	Free	Line
0x7fff32a334e8	0x7b9010	1	0	5
0x7fff32a334e0	0x7b9080	1	0	6
0x7fff32a334e0	0x7b9010	1	0	7
0x7fff32a334e8	0x7b9010	0	1	8
0x7fff32a334e0	0x7b9010	0	1	9

VIOLATED PROPERTY: Invalid FREE

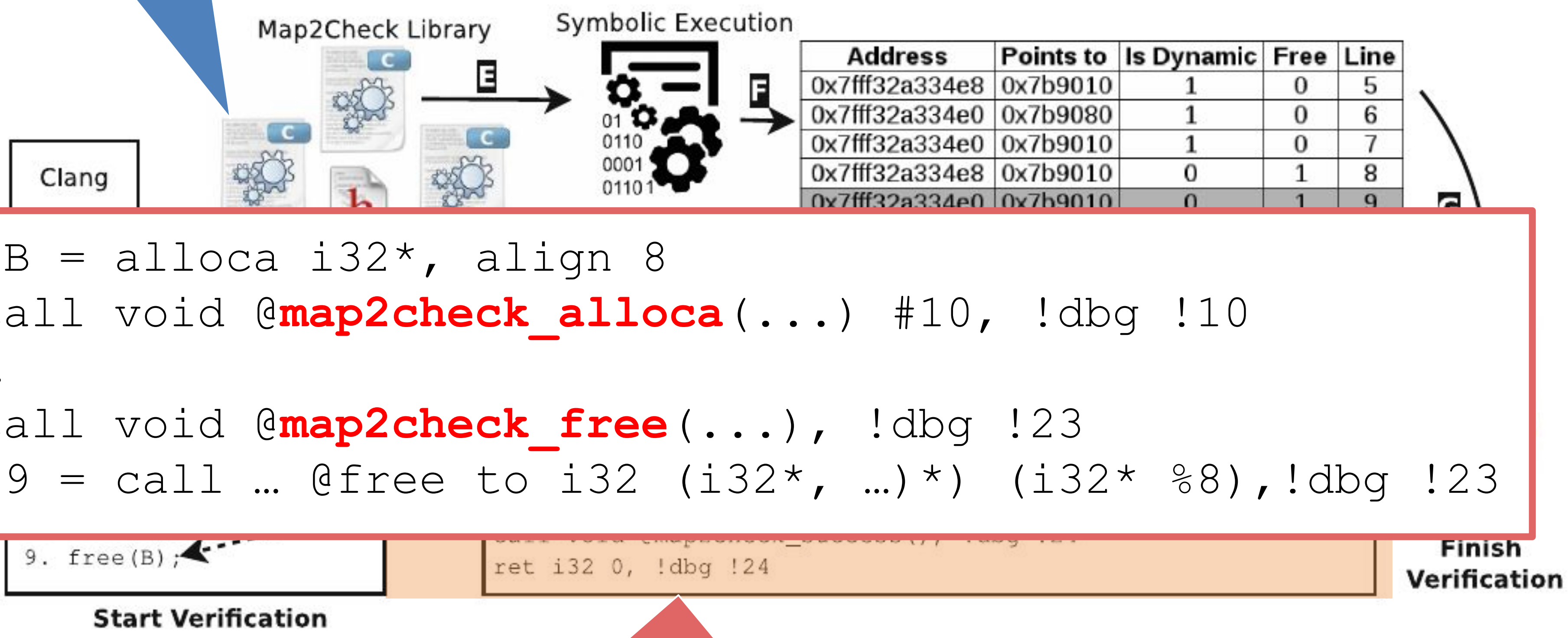
```
%A = alloca i32*, align 8
call void @map2check_alloca(...) #10
%B = alloca i32*, align 8
call void @map2check_alloca(...) #10, !dbg !10
...
call void @map2check_free(...), !dbg !23
%9 = call ... @free to i32 (i32*, ...)*)(i32* %8), !dbg !23
call void @map2check_success(), !dbg !24
ret i32 0, !dbg !24
```

Convert the C code

Map2Check - Verification Approach

Connect Map2Check library

```
#include <map2check.h>
```



Address	Points to	Is Dynamic	Free	Line
0x7fff32a334e8	0x7b9010	1	0	5
0x7fff32a334e0	0x7b9080	1	0	6
0x7fff32a334e0	0x7b9010	1	0	7
0x7fff32a334e8	0x7b9010	0	1	8
0x7fff32a334e0	0x7b9010	0	1	9

```
%B = alloca i32*, align 8
call void @map2check_alloca(...) #10, !dbg !10
…
call void @map2check_free(...), !dbg !23
%9 = call … @free to i32 (i32*, …)*) (i32* %8),!dbg !23
```

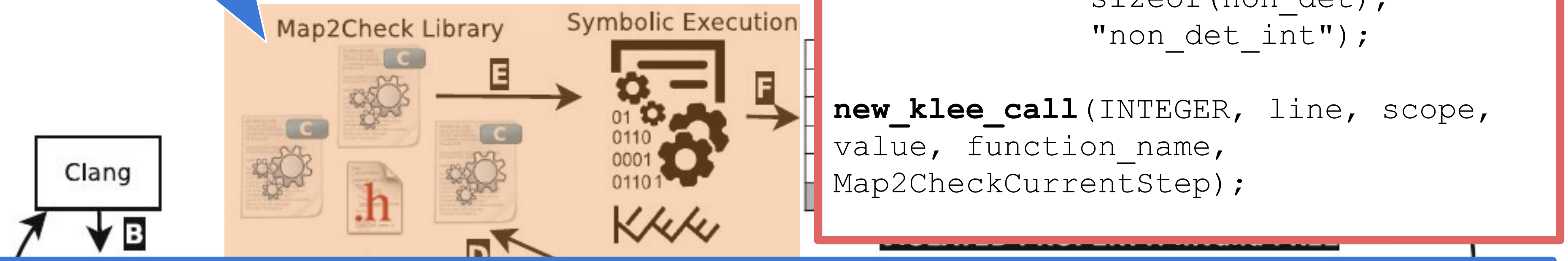
Add Map2Check library functions

Map2Check - Verification Approach

Apply further Clang optimizations

promote memory to register

Generate concrete inputs



```
klee_make_symbolic(&non_det,
              sizeof(non_det),
              "non_det_int");

new_klee_call(INTEGER, line, scope,
value, function_name,
Map2CheckCurrentStep);
```

```
Bool is_invalid_free(long address, MAP2CHECK_CONTAINER* log) {
  ...
  for(; i >= 0; i--) {
    LIST_LOG_ROW* row = (LIST_LOG_ROW*) get_element_at(i, *log);
    ...
    if(is_free || (!is_dynamic)) {
      return TRUE;
    }else {
      return FALSE;
    }
  }return TRUE;
}
```

Map2Check - Verification Approach

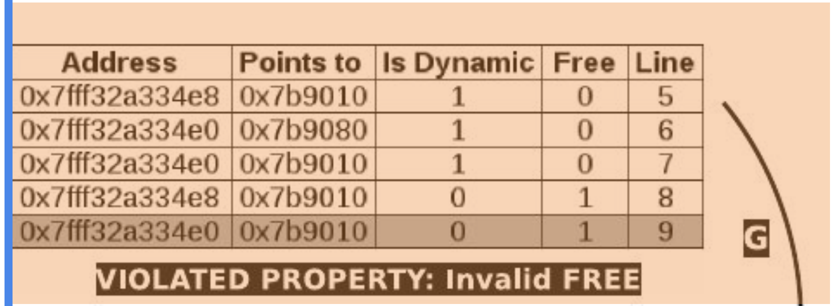
```
$ ./map2check test/tacas2018.c
...
State 5: file test/tacas2018.c
-----------------------------------------
>>Memory list log

  Line content    : free(B);
  Address         : 0x7fff32a334e0
  PointsTo        : 0x7b9010
  Is Free         : TRUE
  Is Dynamic      : FALSE
  Var Name        : B
  Line Number     : 9
  Function Scope  : main

-----------------------------------------
Violated property:
file map2check_property line 9 function
main
FALSE-FREE: Operand of free must have
zero pointer offset

         VERIFICATION FAILED
```

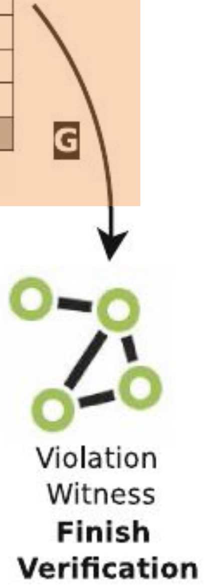
Verification result and generate witnesses



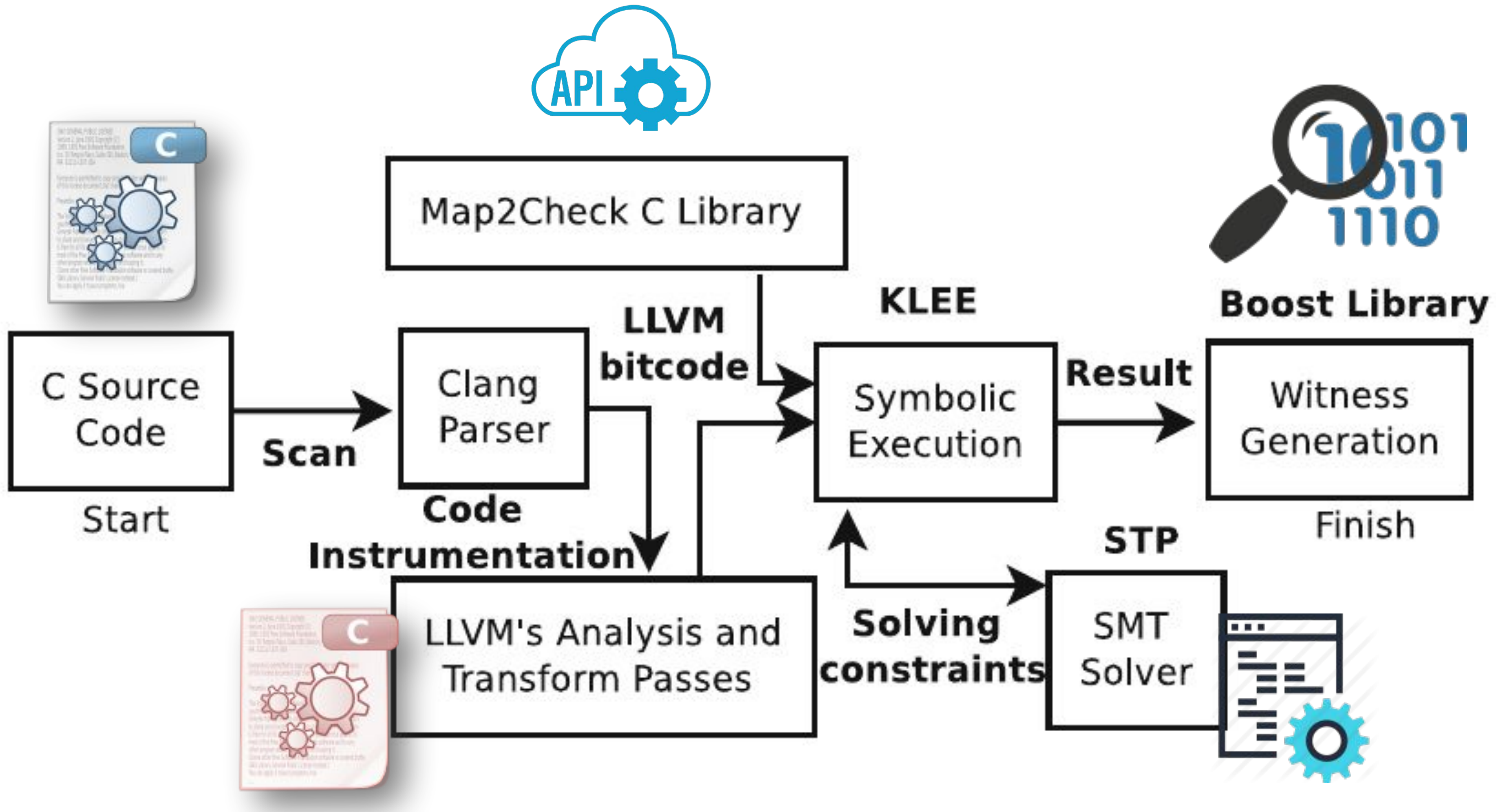
Address	Points to	Is Dynamic	Free	Line
0x7fff32a334e8	0x7b9010	1	0	5
0x7fff32a334e0	0x7b9080	1	0	6
0x7fff32a334e0	0x7b9010	1	0	7
0x7fff32a334e8	0x7b9010	0	1	8
0x7fff32a334e0	0x7b9010	0	1	9

VIOLATED PROPERTY: Invalid FREE

- KLEE output
- Basic block executed in the control flow graph
- Basic blocks as invariants

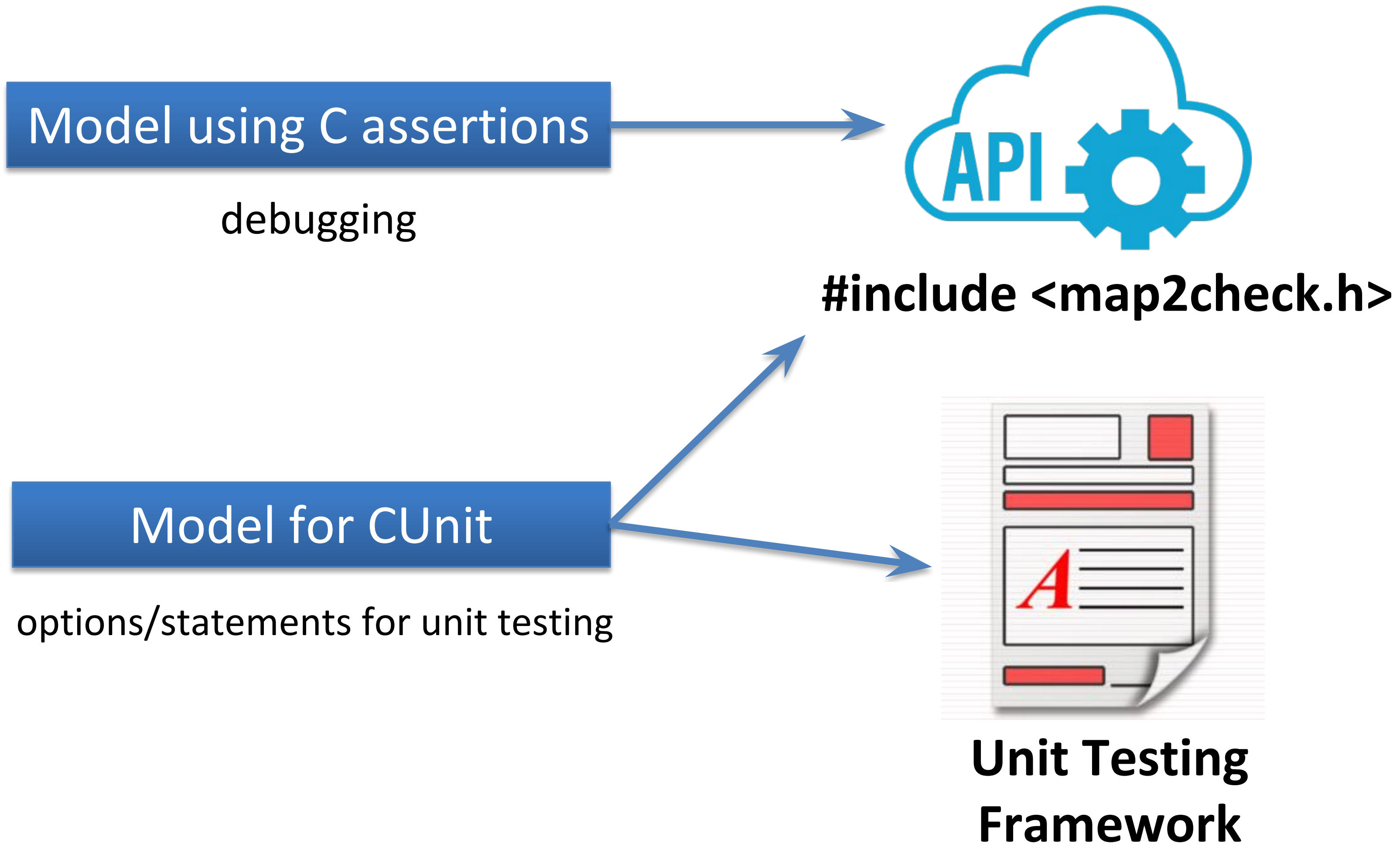


Proposed Architecture



Map2Check tool is available at https://map2check.github.io

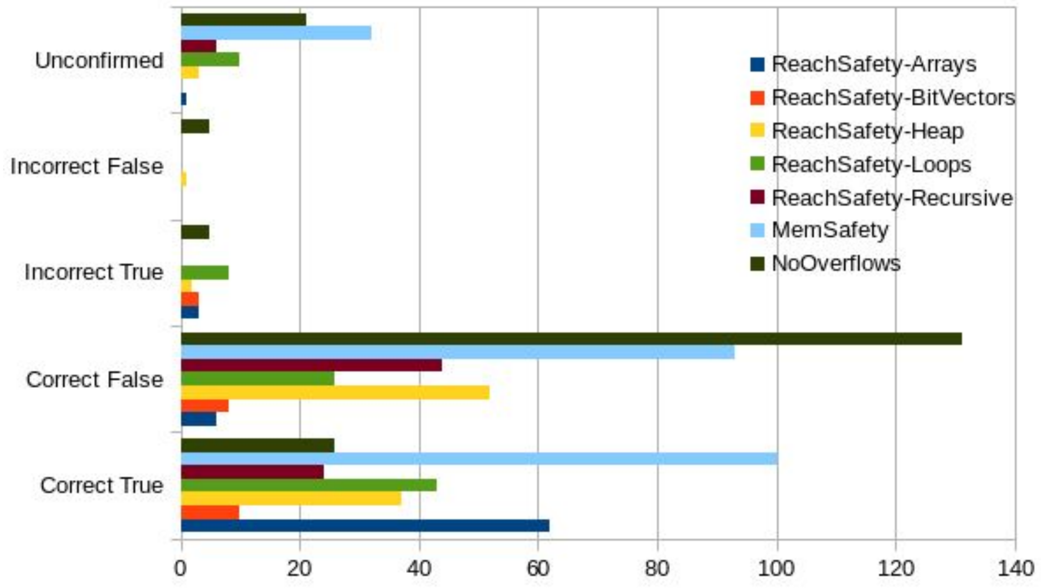
Proposed Architecture



Strengths and Weaknesses - Map2Check

SV-COMP’18 results

- ✓ **ReachSafety-Arrays** (**the highest score, i.e., 106**)
- ✓ ReachSafety-BitVectors
- ✓ ReachSafety-Heap
- ✓ ReachSafety-Loops
- ✓ ReachSafety-Recursive
- ✓ **MemSafety** (**a score of 228**)
- ✓ NoOverflows



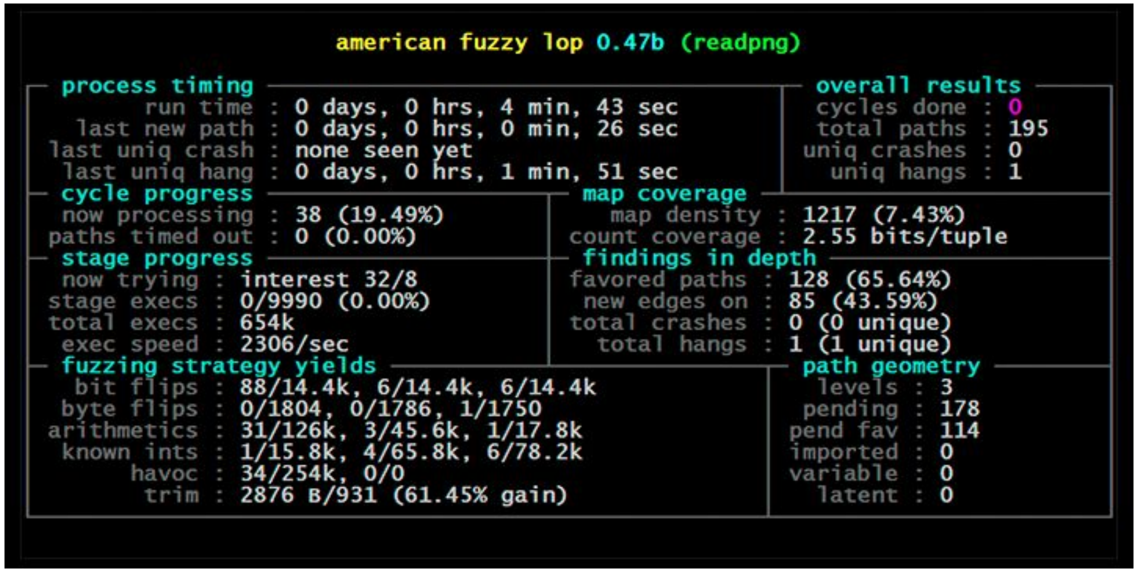
Strengths and Weaknesses - Map2Check

- ✓ Map2Check exploits **dynamic information flow** by tainting program data
- ✓ It uses **Clang/LLVM** as an industrial-strength compiler to simplify and instrument the code
- ✓ It employs **KLEE** to produce concrete inputs for different program executions
- ✓ Map2Check **bounds the loops** and recursion up to a given depth k
- ✓ Map2Check can be effective in generating and checking test cases of **memory management** for C programs

Map2Check - New plans

American fuzzy lop AFL

- Improve code exploration
- Loops

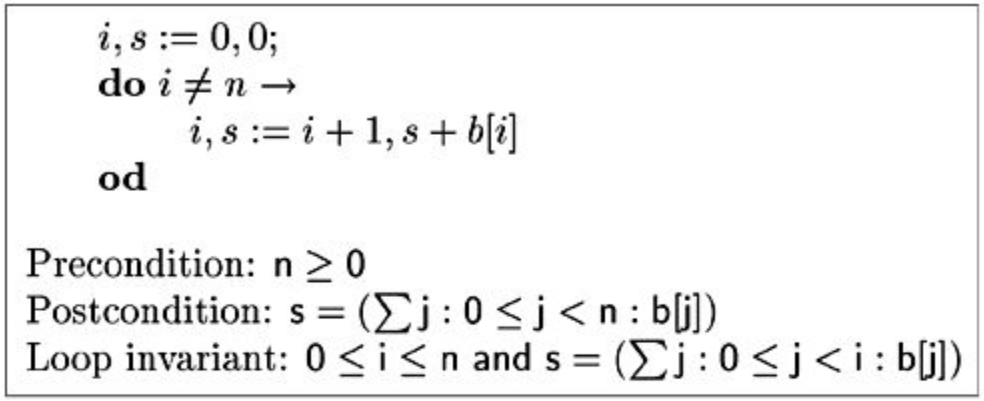


```
                    american fuzzy lop 0.47b (readpng)

 process timing ─────────────────────────────────────── overall results ──
        run time : 0 days, 0 hrs, 4 min, 43 sec        cycles done : 0
   last new path : 0 days, 0 hrs, 0 min, 26 sec        total paths : 195
 last uniq crash : none seen yet                      uniq crashes : 0
  last uniq hang : 0 days, 0 hrs, 1 min, 51 sec         uniq hangs : 1
 cycle progress ─────────────────────── map coverage ──────────────────────
  now processing : 38 (19.49%)           map density : 1217 (7.43%)
 paths timed out : 0 (0.00%)          count coverage : 2.55 bits/tuple
 stage progress ─────────────────────── findings in depth ─────────────────
  now trying : interest 32/8          favored paths : 128 (65.64%)
 stage execs : 0/9990 (0.00%)          new edges on : 85 (43.59%)
 total execs : 654k                   total crashes : 0 (0 unique)
  exec speed : 2306/sec                 total hangs : 1 (1 unique)
 fuzzing strategy yields ──────────────────────────── path geometry ───────
   bit flips : 88/14.4k, 6/14.4k, 6/14.4k              levels : 3
  byte flips : 0/1804, 0/1786, 1/1750                 pending : 178
 arithmetics : 31/126k, 3/45.6k, 1/17.8k             pend fav : 114
  known ints : 1/15.8k, 4/65.8k, 6/78.2k             imported : 0
       havoc : 34/254k, 0/0                          variable : 0
        trim : 2876 B/931 (61.45% gain)                latent : 0
```

Program invariants

- Counterexample refinement
- Data flow analysis + polyhedral invariant template



```
$i, s := 0, 0;$
do $i \neq n \rightarrow$
    $i, s := i + 1, s + b[i]$
od
```

Precondition: $n \geq 0$

Postcondition: $s = (\sum j : 0 \leq j < n : b[j])$

Loop invariant: $0 \leq i \leq n$ and $s = (\sum j : 0 \leq j < i : b[j])$

Thank you for your attention!

map2check.tool@gmail.com