MANCHESTER 1824
The University of Manchester

ESBMC-CHERI: Towards Verification of C/C++ Programs for CHERI Platforms with ESBMC

Franz Brauße[1] Kunjian Song[1] Fedor Shmarov[2] Rafael Menezes[1]
Mikhail R. Gadelha[3] Konstantin Korovin[1] Giles Reger[1] Lucas C. Cordeiro[1]

[1]The University of Manchester, UK

[2]University of Newcastle, UK

[3]Igalia, A Coruña, Spain

POCL'24 London, 16/01/2024

Overview of this talk

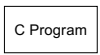
- Brief introduction to ESBMC
- Architectural changes necessary to support CHERI-C
- Challenges and solutions
- Conclusions and future work

Introduction to ESBMC

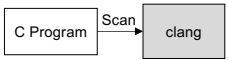
Efficient SMT-based Bounded Model Checker

- Open source logic-based automated reasoning for safety and security of C/C++ programs.
  - Apache License 2.0.
  - Used by industries: Intel, Nokia, ARM
  - Core concept: symbolic execution.
  - Concurrency support.
  - Frontends for different input languages.
- Input: Program, Output: verification success or failure + witness.
  - Witness is a trace of assignments leading to the violation.
- C/C++ frontends translate the Clang AST.

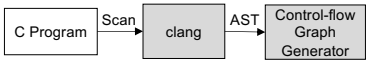
Architecture of ESBMC



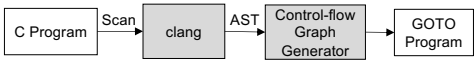
Architecture of ESBMC



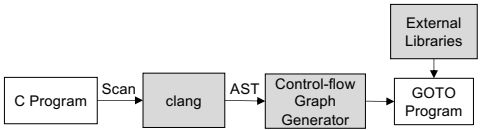
Architecture of ESBMC



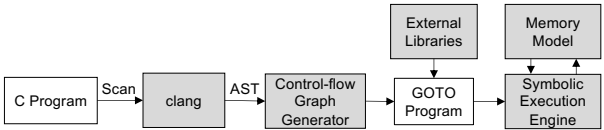
Architecture of ESBMC



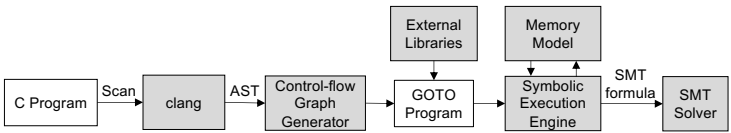
Architecture of ESBMC



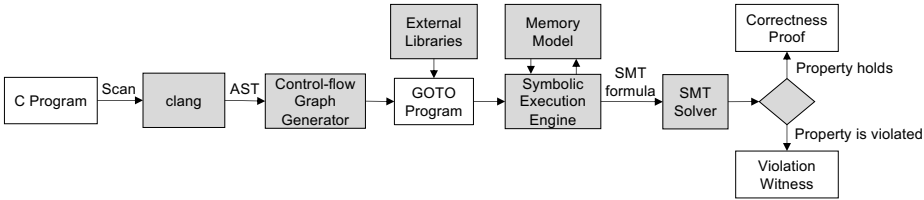
Architecture of ESBMC



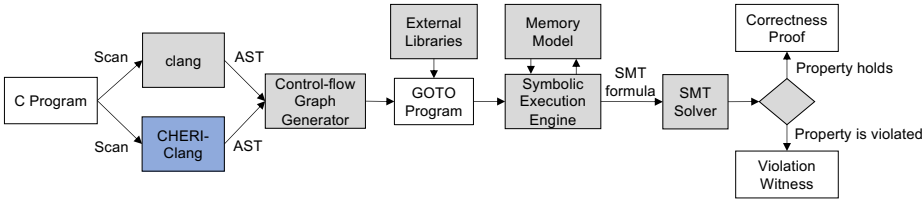
Architecture of ESBMC



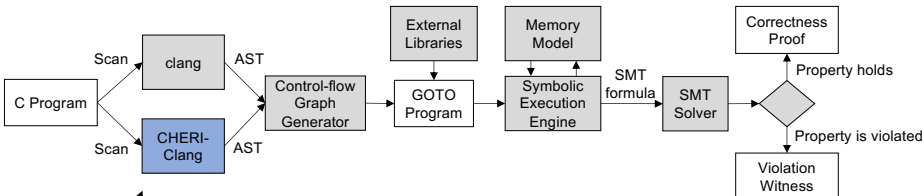
Architecture of ESBMC



Architecture of ESBMC

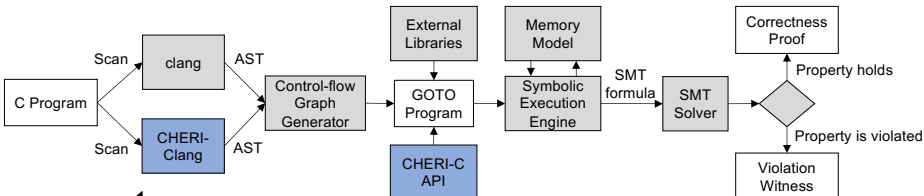


Architecture of ESBMC



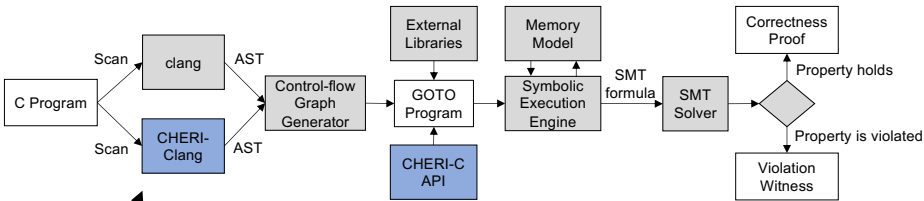
CHERI Clang/LLVM
compiler

Architecture of ESBMC



CHERI Clang/LLVM
compiler

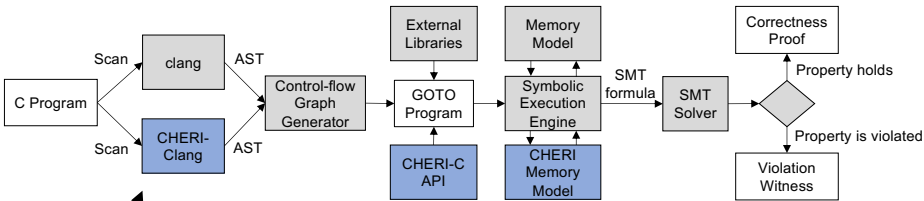
Architecture of ESBMC



CHERI Clang/LLVM compiler

Implement computational model for CHERI-C API

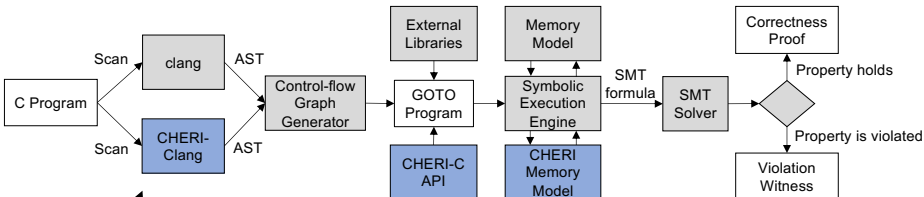
Architecture of ESBMC



CHERI Clang/LLVM compiler

Implement computational model for CHERI-C API

Architecture of ESBMC



CHERI Clang/LLVM compiler

Implement computational model for CHERI-C API

- New capability types
- Tagged memory
- Capability dereferencing

ESBMC-CHERI

Goals:

- both uncompressed and concentrate capabilities modelled bit-precise
- reasoning in CHERI-BSD execution environment of capability hardware platforms RISC-V, ARM Morello, (MIPS)
- both, hybrid and purecap mode

ESBMC-CHERI

Goals:

- both uncompressed and concentrate capabilities modelled bit-precise
- reasoning in CHERI-BSD execution environment of capability hardware platforms RISC-V, ARM Morello, (MIPS)
- both, hybrid and purecap mode

Challenges wrt. compression algorithms

ESBMC-CHERI

Goals:

- both uncompressed and concentrate capabilities modelled bit-precise
- reasoning in CHERI-BSD execution environment of capability hardware platforms RISC-V, ARM Morello, (MIPS)
- both, hybrid and purecap mode

Challenges wrt. compression algorithms

Two directions for verification of CHERI-C programs:

- Proving absence of CHERI-exceptions
  - Bit-precise model of in-memory representation of capabilities
  - Base/bounds might differ from those ESBMC already maintains
  - Additional checks for permissions
- Assuming absence of CHERI-exceptions
  - Reasoning about unmodelled external C functions (libraries) via pointer provenance
  - Simplification of ESBMC's spatial memory safety checks
  - Optimisation of operational models of `libc` functions

Challenges for Pointer Provenance

Non-provenance in 'real world'

1. International Competition on Software Verification (SV-COMP[1]):

```
void *p = malloc(n);
intptr_t i = p;
free(p);
p = malloc(n);
assert(i != p);
```

Should fail verification.

2. ARM Realm Management Monitor project[2]:

```
char buffer[n];
intptr_t i = nondet(); /* user-input */
assume(buffer <= i && i < buffer + n);
char *q = buffer + ((char *)i - buffer);
*q;
```

Expected to work.

[1] https://sv-comp.sosy-lab.org

[2] Part of Arm Confidential Compute Architecture

Conclusions

- CHERI-Clang makes frontend changes easy.
- Tagged memory & CHERI-C API using existing functionality.
- Provenance not universally compatible with existing expectations.
- In-memory representation of capabilities and their permissions are difficult.
- On the other hand: assuming CHERI semantics reduces complexity of software verification.