MANCHESTER
1824
The University of Manchester

ESBMC v7.3: Model Checking C++ Programs using Clang AST

Kunjian Song, Mikhail R. Gadelha, Franz Brauße, Rafael S. Menezes, Lucas C. Cordeiro

26th Brazilian Symposium on Formal Methods 2023

- Motivations
- ESBMC v7.3 Architectural Overview
- ESBMC v7.3 in Action
- Evaluation
- Future Work
- Download and Installation of ESBMC v7.3

ESBMC v7.3

- Motivations
- ESBMC v7.3 Architectural Overview
- ESBMC v7.3 in Action
- Evaluation
- Future Work
- Download and Installation of ESBMC v7.3

Motivations

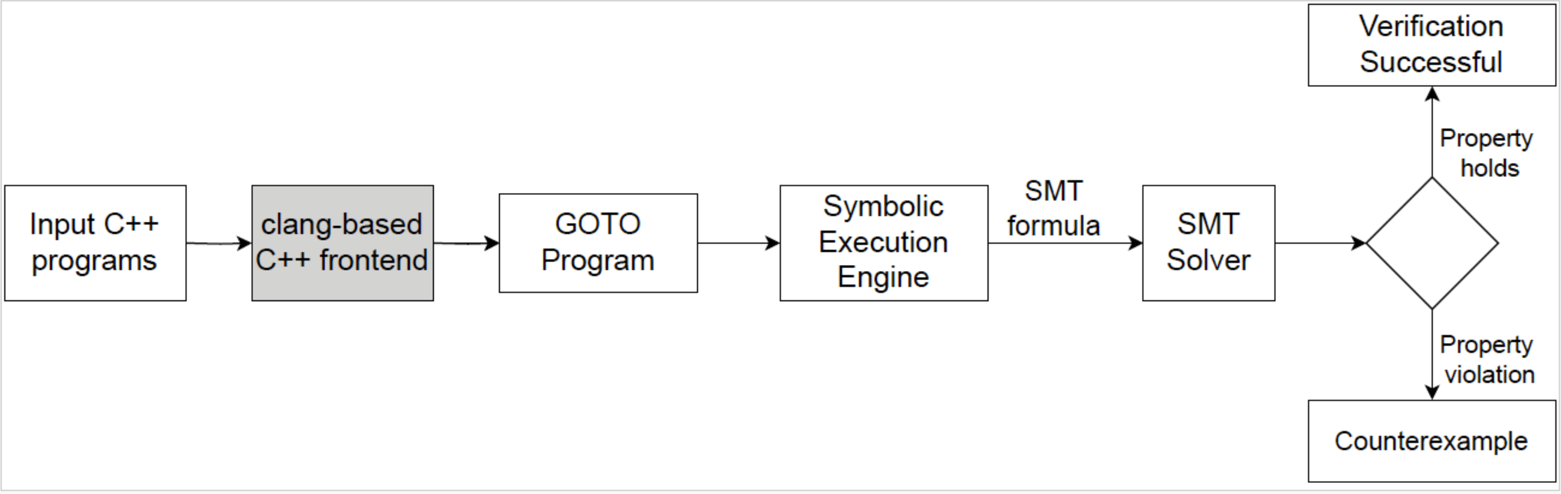
- Memory safety issues remain a major source of security vulnerabilities in C++ programs.
- Formal verification of C++ programs is more challenging than C programs due to the sophisticated features, such as templates, inheritance, polymorphism and STL libraries.
- ESBMC reaches its maturity in version 2.1 for C++03 verification, but:
  - ESBMC v2.1 uses a Flex and Bison-based frontend from CBMC.
  - Hard-to-maintain code and cannot evolve to support C++11 features.
  - Unable to cope with recursive templates.
- Solution – Replace the old frontend with a new frontend converting clang AST to ESBMC IR (intermediate representation).

ESBMC v7.3

- Motivations
- ESBMC v7.3 Architectural Overview
- ESBMC v7.3 in Action
- Evaluation
- Future Work
- Download and Installation of ESBMC v7.3

ESBMC v7.3

ESBMC v7.3 Architectural Overview



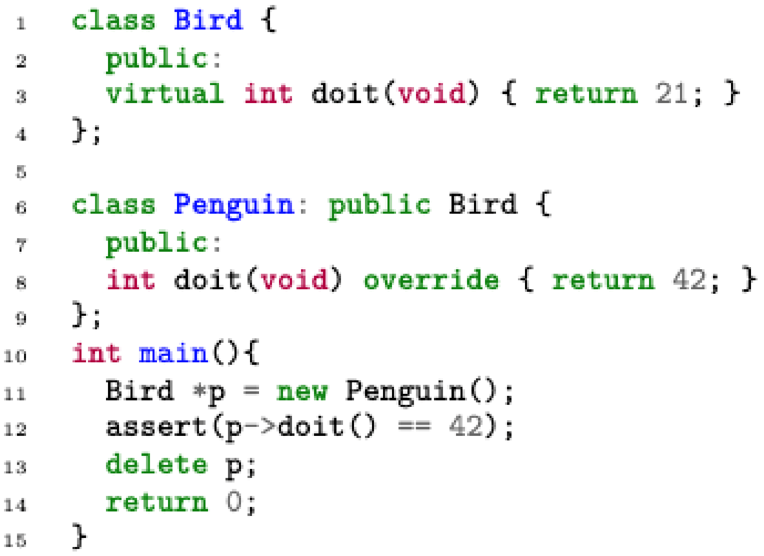
- Complete Redesign of the frontend
  - The grey block represents the new clang-based C++ frontend integrated into ESBMC v7.3, converting clang AST to ESBMC's GOTO IR.
- The new frontend provide comprehensive insights into the object models.
  - Seamless conversion of C++ polymorphism code to ESBMC's IR.
- Enhanced type checking for templates.
  - Explicit-template specialization, partial template specialization and recursive templates.

ESBMC v7.3

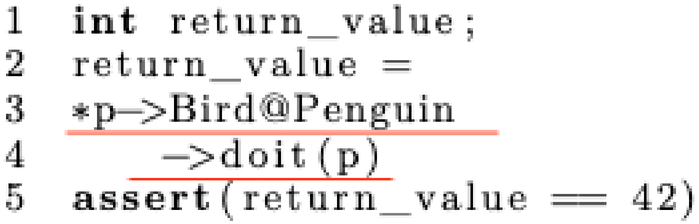
- Motivations
- ESBMC v7.3 Architectural Overview
- **ESBMC v7.3 in Action**
- Evaluation
- Future Work
- Download and Installation of ESBMC v7.3

ESBMC v7.3

ESBMC v7.3 in Action: Polymorphism

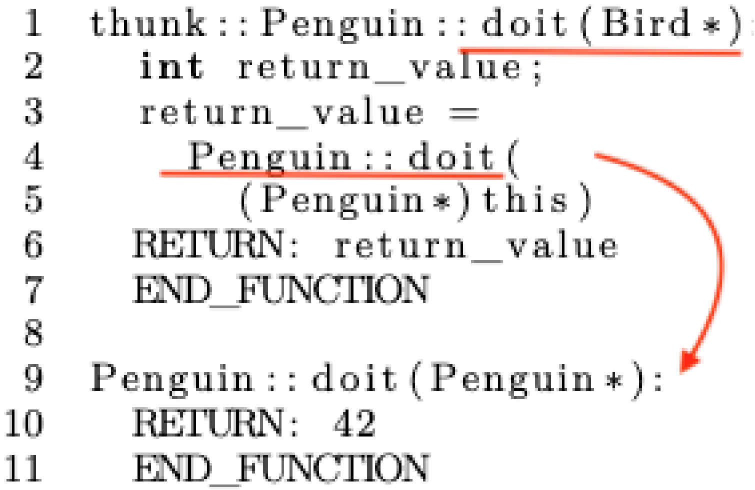


```cpp
class Bird {
  public:
  virtual int doit(void) { return 21; }
};

class Penguin: public Bird {
  public:
  int doit(void) override { return 42; }
};
int main(){
  Bird *p = new Penguin();
  assert(p->doit() == 42);
  delete p;
  return 0;
}
```



```
int return_value;
return_value =
*p->Bird@Penguin
      ->doit(p)
assert(return_value == 42)
```

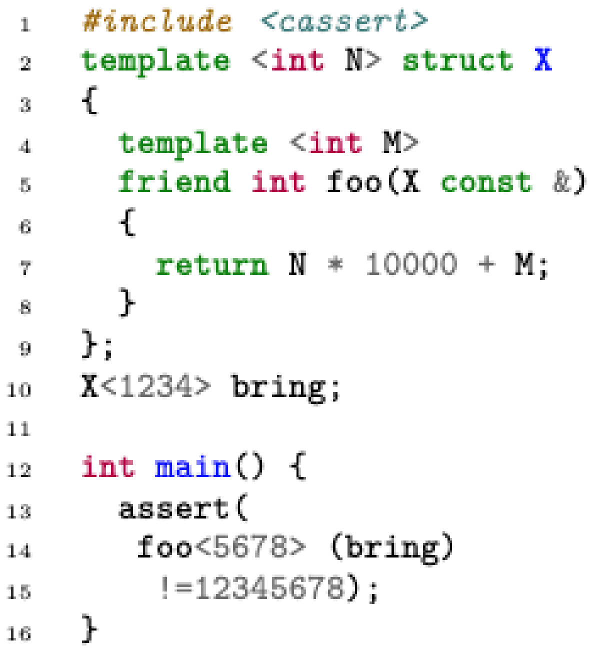
Dynamic dispatch using the virtual pointer of Penguin class.



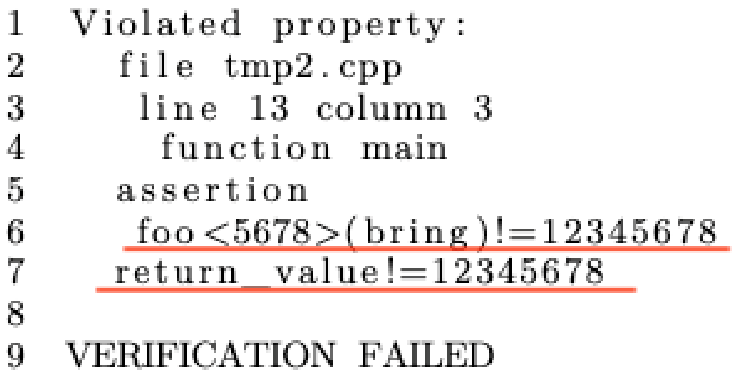
```
thunk::Penguin::doit(Bird*):
   int return_value;
   return_value =
     Penguin::doit(
        (Penguin*)this)
   RETURN: return_value
   END_FUNCTION

Penguin::doit(Penguin*):
   RETURN: 42
   END_FUNCTION
```

Redirecting the call to the overriding function.

ESBMC v7.3 in Action: Template



```
#include <cassert>
template <int N> struct X
{
  template <int M>
  friend int foo(X const &)
  {
    return N * 10000 + M;
  }
};
X<1234> bring;

int main() {
  assert(
   foo<5678> (bring)
    !=12345678);
}
```



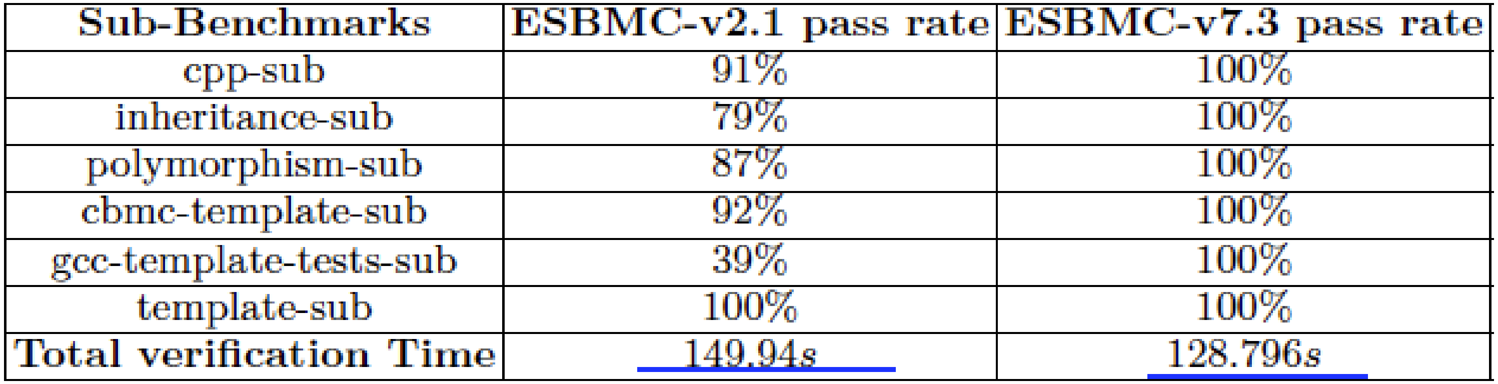
```
Violated property:
  file tmp2.cpp
   line 13 column 3
    function main
  assertion
   foo<5678>(bring)!=12345678
  return_value!=12345678

VERIFICATION FAILED
```

- Template test case from Bug 10158 on GCC Bugzilla, containing nested template with non-type parameters.
- Return value should be 12345678. ESBMC v7.3 successfully detected the assertion failure.
    - ESBMC v2.1 failed due to “CONVERSION ERROR”. CBMC 5.88.1 aborted during type-checking. Cppcheck v2.11.1 did not give any verification verdict.

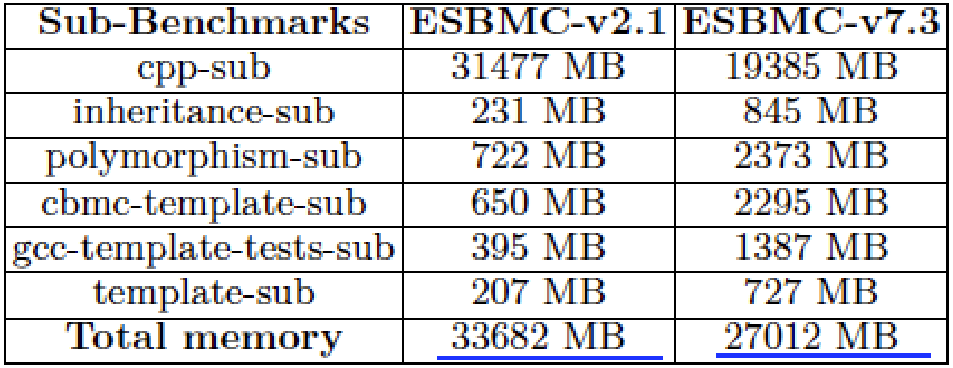
ESBMC v7.3

- Motivations
- ESBMC v7.3 Architectural Overview
- ESBMC v7.3 in Action
- **Evaluation**
- Future Work
- Download and Installation of ESBMC v7.3

Evaluation – v2.1 vs v7.3



Sub-Benchmarks	ESBMC-v2.1 pass rate	ESBMC-v7.3 pass rate
cpp-sub	91%	100%
inheritance-sub	79%	100%
polymorphism-sub	87%	100%
cbmc-template-sub	92%	100%
gcc-template-tests-sub	39%	100%
template-sub	100%	100%
Total verification Time	149.94s	128.796s



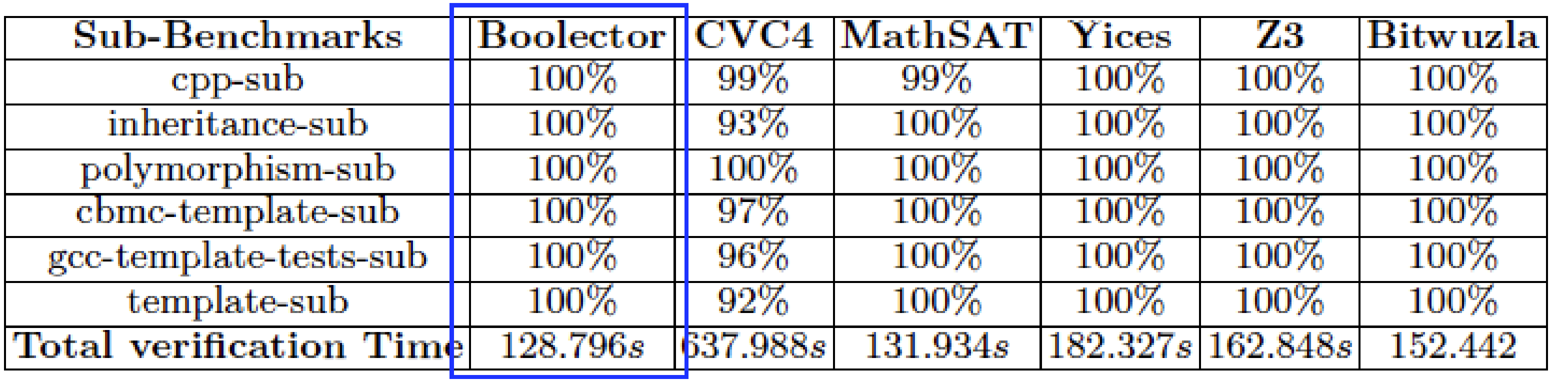
Sub-Benchmarks	ESBMC-v2.1	ESBMC-v7.3
cpp-sub	31477 MB	19385 MB
inheritance-sub	231 MB	845 MB
polymorphism-sub	722 MB	2373 MB
cbmc-template-sub	650 MB	2295 MB
gcc-template-tests-sub	395 MB	1387 MB
template-sub	207 MB	727 MB
Total memory	33682 MB	27012 MB

- ESBMC v7.3 improved the pass rate in all benchmark categories, giving more correct verification results.

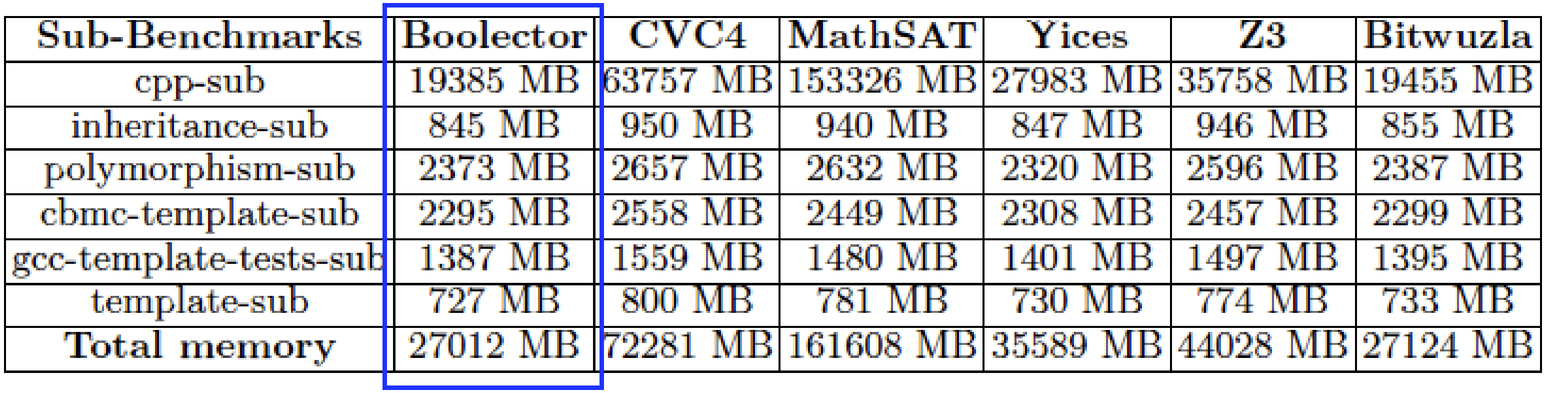
- ESBMC v7.3 can provide more accurate results faster and uses less memory.

ESBMC v7.3

Evaluation – Different backend solvers



Sub-Benchmarks	Boolector	CVC4	MathSAT	Yices	Z3	Bitwuzla
cpp-sub	100%	99%	99%	100%	100%	100%
inheritance-sub	100%	93%	100%	100%	100%	100%
polymorphism-sub	100%	100%	100%	100%	100%	100%
cbmc-template-sub	100%	97%	100%	100%	100%	100%
gcc-template-tests-sub	100%	96%	100%	100%	100%	100%
template-sub	100%	92%	100%	100%	100%	100%
Total verification Time	128.796s	637.988s	131.934s	182.327s	162.848s	152.442



Sub-Benchmarks	Boolector	CVC4	MathSAT	Yices	Z3	Bitwuzla
cpp-sub	19385 MB	63757 MB	153326 MB	27983 MB	35758 MB	19455 MB
inheritance-sub	845 MB	950 MB	940 MB	847 MB	946 MB	855 MB
polymorphism-sub	2373 MB	2657 MB	2632 MB	2320 MB	2596 MB	2387 MB
cbmc-template-sub	2295 MB	2558 MB	2449 MB	2308 MB	2457 MB	2299 MB
gcc-template-tests-sub	1387 MB	1559 MB	1480 MB	1401 MB	1497 MB	1395 MB
template-sub	727 MB	800 MB	781 MB	730 MB	774 MB	733 MB
Total memory	27012 MB	72281 MB	161608 MB	35589 MB	44028 MB	27124 MB

- ESBMC supports multiple SMT solvers in the backend.

- ESBMC v7.3 with Boolector is the fastest configuration that consumes the minimum amount of memory to verify all benchmarks.

- Bitwuzla comes near the Boolector configuration.

ESBMC v7.3

- Motivations
- ESBMC v7.3 Architectural Overview
- ESBMC v7.3 in Action
- Evaluation
- **Future Work**
- Download and Installation of ESBMC v7.3

Future work

- The operational models (OMs) require regular review and updates to align with the C++ standard used in the input program.
  - Accurate OMs are essential, as any approximation may lead to incorrect encoding and invalidate the verification results.

- Enhance the new frontend coverage and reduce the number of OMs we need to maintain, supporting more C++ libraries.

- Verify the C++ interpreter in OpenJDK Morello (CHERI capability enhanced C++ code).

- Contribute to benchmarks for SV-COMP.

- Motivations
- ESBMC v7.3 Architectural Overview
- ESBMC v7.3 in Action
- Evaluation
- Future Work
- Download and Installation of ESBMC v7.3

ESBMC v7.3

Download and Installation ESBMC

ESBMC Release v7.3

https://github.com/esbmc/esbmc/releases/tag/v7.3

ESBMC Publications for C++ verification:

- Ramalho, Mikhail, et al. "SMT-based bounded model checking of C++ programs." *2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*. IEEE, 2013.
- Monteiro, Felipe R., Mikhail R. Gadelha, and Lucas C. Cordeiro. "Model checking C++ programs." *Software Testing, Verification and Reliability* 32.1 (2022): e1793.
- Song, Kunjian, et al. "ESBMC v7. 3: Model Checking C++ Programs using Clang AST." *26th Brazilian Symposium on Formal Methods*. Springer London, 2023.

Thank you!