ESBMC 5.0

Mikhail R. Gadelha, Felipe R. Monteiro, Jeremy Morse,
Lucas Cordeiro, Bernd Fischer, Denis Nicole

University of Southampton School of Electronics and Computer Science

University of Bristol

UFAM

The University of Manchester

Universiteit Stellenbosch University

MOTIVATION

Motivation

USS Yorktown

- Battleship built in 1946 and automated in 1996 (27 dual-core 200MHz processors and Windows NT).

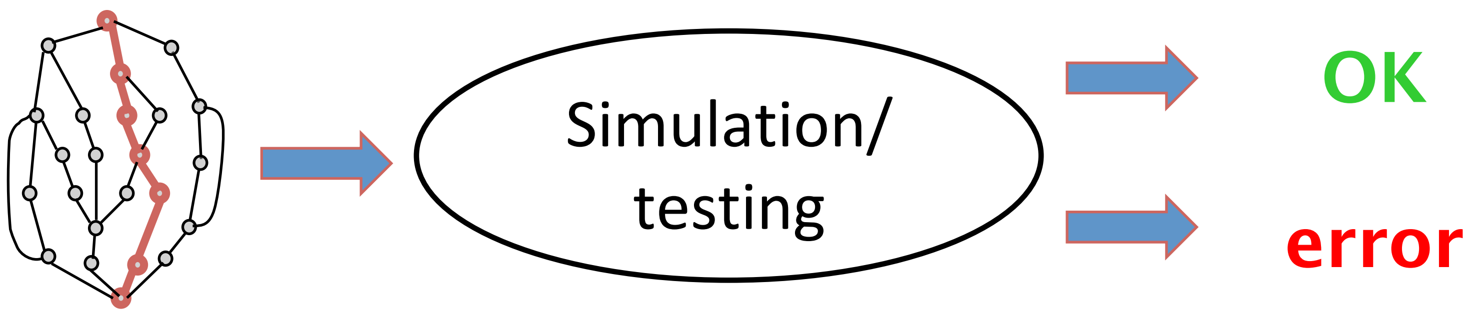
Motivation

USS Yorktown

- Battleship built in 1946 and automated in 1996 (27 dual-core 200MHz processors and Windows NT).
- **Failure due to a division by zero:** It had to be towed back to its naval base.

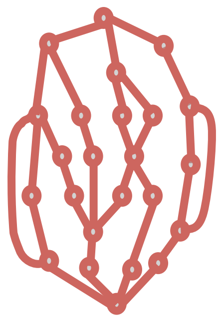
SOFTWARE VERIFICATION TECHNIQUES

Model Checking vs Testing/ Simulation



- Checks only some of the system executions.
- May miss errors.
- Can be less expensive than model checking.

Model Checking vs Testing/ Simulation



Model Checking

OK

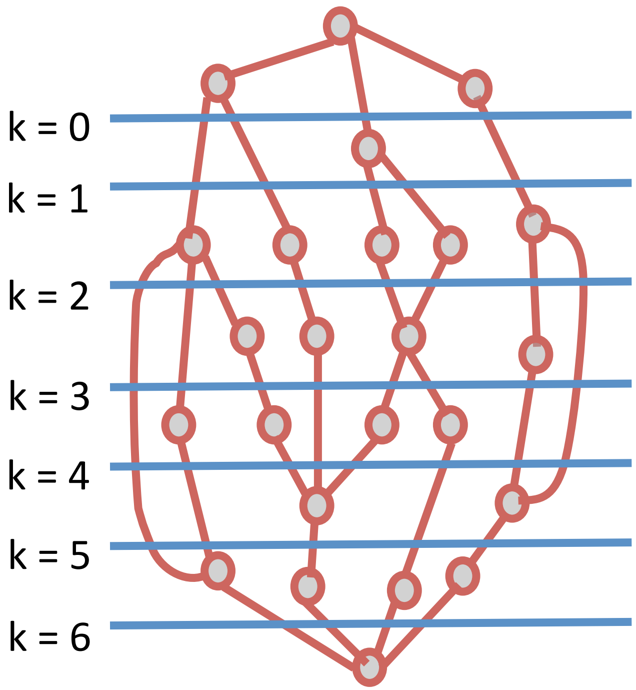
Error trace

```
Line 5: …
Line 12: …
…
Line 41:…
```

Specification (e.g, LTL)

- Exhaustively explores all executions.
  - Can be bounded to limit number of iterations, context-switch, etc.
- Report errors as traces.
- Can be extremely resource-hungry.

Bounded Model checking



- Bounded model checkers “slice” the state space in depth.
- It’s aimed to find bugs and (naïvely) can only prove correctness if all states are reachable within the bound.

ESBMC 5.0

ESBMC 5.0

- ESBMC, the *Efficient SMT-Based Context-Bounded Model Checker* was originally developed at Southampton by Lucas Cordeiro under the supervision of Bernd Fischer.

- Jeremy Morse further developed ESBMC during his PhD.

- Development is now led from Southampton by Mikhail Gadelha.

- Turned 10 years in 2018!

ESBMC 5.0

- SMT-based BMC of single- and multi-threaded C/C++ programs.
- exploits SMT solvers and their background theories:
  - optimized encodings for pointers, bit operations, unions, arithmetic over- and underflow, and floating-points,
  - support for Boolector, Z3, MathSAT, CVC4 and Yices.
- supports verifying multi-threaded software that uses pthreads threading library:
  - *lazy exploration of the reachability tree.*

Supported Properties

- built-in properties:
  - arithmetic under- and overflow,
  - pointer safety,
  - array bounds,
  - division by zero,
  - memory leaks,
  - atomicity and order violations,
  - deadlock,
  - data race.

K-INDUCTION

K-induction: we can sometimes analyse to unbounded depths

- In general, there is no way to deduce depths:
  - halting problem,
  - lots of current work on deducing invariants.
- For simple loops, they can sometimes be guessed.
- Interval analysis often speed up the analysis considerably.

K-induction: the proof falls into three parts

1. Base case: naïve BMC, tries to find bugs.

2. Forward condition: checks the completeness threshold (if all loops were completely unrolled).

3. Inductive step: over-approximate loops so all states can be checked without unrolling them completely (sometime it helps to unroll a few times to strengthen invariants).

FLOATING-POINTS

Floating-points: can it fail?

```
int main()
{
  float x;
  float y = x;
  assert (x == y);
  return 0;
}
```

Floating-points: can it fail?

```
int main()
{
   float x = NaN;
   float y = x;
   assert (x == y);
   return 0;
}
```

Floating-point Encoding

- ESBMC encodes floating-point arithmetic using:
  - **bitvectors**, which extends the floating-point arithmetic support to all solvers that are currently integrated.
  - the **SMT theory of floating-points**, available only in Z3 and MathSAT.

PYTHON API

Python API

- ESBMC now includes a **Python API** that reduces the difficulty of prototyping new features and makes the tool internals accessible to a wider audience.

- The verification process can be intercepted and modified: we currently use the process to call Matlab and generate the transfer functions of digital systems.

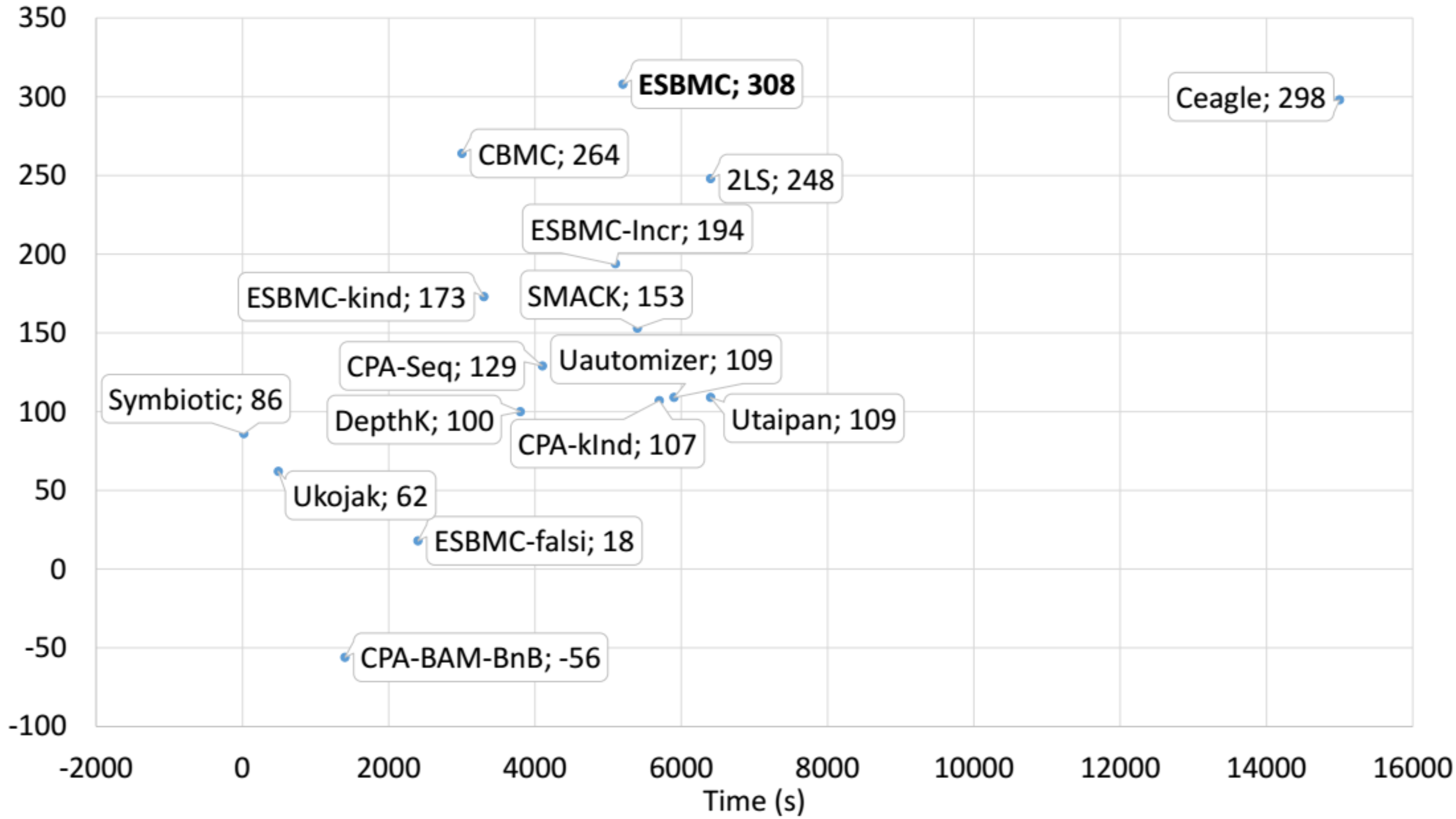
Experimental Evaluation

- Our evaluation consists of 9523 benchmarks from SV-COMP'18, checking a range of properties:
  - Reachability in single- and multi-threaded programs,
  - Memory safety,
  - Overflow,
  - Termination.

Experimental Evaluation

- ESBMC ranked third in the overall category, with a 5476 score.
- The k-induction algorithm reported 4301 correct results, the best result among tools that used *k*-induction in the competition.
- 92% of witnesses being correctly validated.

Experimental Evaluation (floating-points)



Thank you

www.esbmc.org

https://github.com/esbmc/esbmc

mikhail.ramalho@gmail.com

rms.felipe@gmail.com