cybersecurity centre

Academic Centre of Excellence

EPSRC

UNIVERSITY OF Southampton

ESBMC: Model-checking C programs

Bernd Fischer, Omar Inverso, Jeremy Morse, Denis Nicole and Gennaro Parlato

Cyber Security Centre

University of Southampton

Model checking

The C programming language continues to be popular for the programming of a wide variety of desktop, embedded and server systems. It is well-known that programming errors in C can create dangerous vulnerabilities, such as potential buffer overflows, in critical software systems. Careful coding standards, checking and extensive testing can help reduce these and related vulnerabilities.

Testing, however, can only attempt to explore a representative sample of program behaviours in a few situations; there is thus a need for a more effective approach to tool support.

Recent advances in algorithms and in processor performance have made it possible to perform an exhaustive analysis of all the possible behaviours of a moderate-sized C program.

ESBMC is a state-of-the-art bounded model checker for C which has a world leading ability to investigate multi-threaded (concurrent) C programs.

ESBMC for sequential C

ESBMC uses a sequence of transformations to turn a C program into a set of mathematical constraints:

- Library components are bound to the program text.
- The C control flow is converted into a simplified GOTO form.
- The GOTO program is converted into a *single static assignment* form. At this stage we have moved from a representation of C *variables* into one which associates a new (constrained) algebraic unknown with each execution of an assignment.
- We introduce additional constraints between the unknowns that can only be satisfied if there is a program error.
- A *satisfiability modulo theory* (SMT) solver is invoked to investigate whether all the constraints can be satisfied at once. Any success is a program error.
- If the SMT solver succeeds (i.e. the program fails) we use its successful assignment of values to unknowns to deduce the erroneous path through the program.

An example, the program:

```
#include <stdlib.h>
int
main()
{
        unsigned int size = nondet_int();
        char *hunk = malloc(size);

        unsigned int pos = nondet_int();
        hunk[pos] = 0;
        return 0;
}
```

generates the report:

```
Violated property:
  file test.c line 10 function main
  dereference failure: dynamic object upper bound
  !(__ESBMC_is_dynamic[POINTER_OBJECT(hunk + pos)] && DYNAMIC_SIZE(hunk +
pos) <= POINTER_OFFSET(hunk) + pos)

VERIFICATION FAILED
```

For further information, publications, and downloads, see:

http://www.esbmc.org/

ESBMC is a collaboration between the University of Southampton and the Federal University of Amazonas, Brazil.

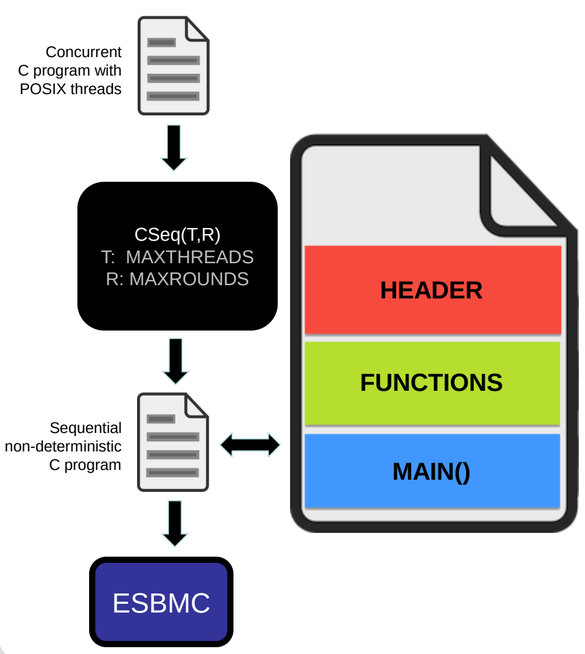
ESBMC for multi-threaded C

Many of the most intractable C errors are *race conditions* that appear during executions of multithreaded C code. ESBMC is especially capable in dealing with these cases. It adopts an *explicit state* approach by constructing all necessary interleavings of thread execution and building an SMT expression for each. This large collection of SMT challenges can then, if necessary, be evaluated on the Universty's *Iridis 3* supercomputer. If any are satisfiable, the program is erroneous and an explicit failing example interleave is displayed.

Sequentialisation: Cseq-ESBMC

We are also investigating a second technique for model-checking multithreaded C Programs:

- Code-to-code translation (from concurrent to sequential C)
- Symbolic modeling of thread interleaving (using non-determinism + arrays)
- ESBMC for bounded model-checking on the sequentialised version



FUNCTIONS

- convert global variable read or write expressions:
  `z = 10;` → `z[ROUND] = 10;`
- non-deterministic context-switches for each statement:
  `<s>;` → `cs(); <s>;`

HEADER

- extra functions for non-det context switch etc.
- extra variables to simulate rounds etc.
- convert global variables into arrays:
  `int x;` → `int x[MAXROUNDS];`
  (the index of the array simulates the round)

MAIN()

- initial values guessing for global variables
- original code for thread creations etc.
- thread-wrapping code:
  check that for each global variable its value at the end of round **(i)** is equal to its value at the beginning of round **(i+1)**
- checks: reachability, deadlock, …

Validating specifications

By default, ESBMC checks for violations of an ISO/ANSI C programming standard: for language errors. Of course, a program may be correct C but may still do the wrong thing. It is therefore possible to include additional specifications for the program in the form of C **`assert`** calls or as *linear temporal logic* specifications. These too are coded into appropriate SMT constraints.

Improvement by competition

ESBMC is a C language tool. It relies on SMT solvers developed by others; currently the best performance is achieved using Microsoft's SMT solver *Z3*.

The field of C model checking research is now big enough to support annual competitions; perhaps the best known is that held in conjunction with the *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS).

The team is proud to report that ESBMC v1.17 won the Gold Medal in the "SystemC" and "Concurrency" categories and the Bronze Medal in the overall ranking of the first International Competition on Software Verification at TACAS 2012.

The development of ESBMC was supported by EPSRC under the NOTOS project, grant EP/E012973/1.

cybersecurity southampton