SMT-Based Context-Bounded Model Checking for Embedded Systems

Lucas Cordeiro

Joint work with
Jeremy Morse, Denis Nicole, Bernd Fischer,
Eddie Lima, and João Edgar Chaves

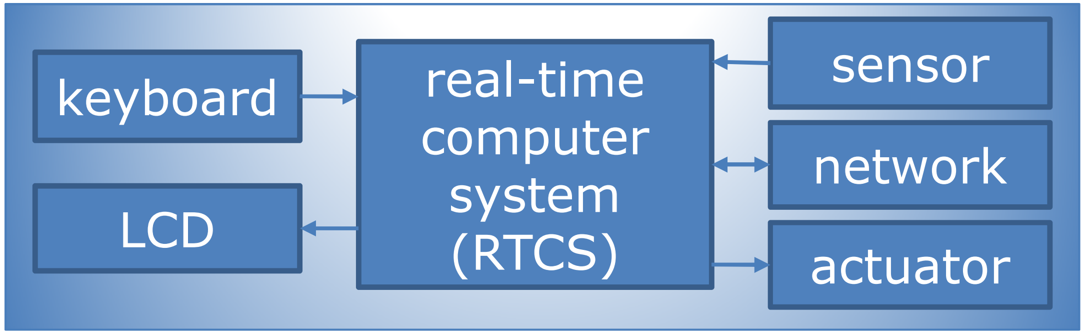
UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

UNIVERSIDADE FEDERAL DO AMAZONAS
IN UNIVERSA SCIENTIA VERITAS
UFAM

Embedded systems are ubiquitous but their verification becomes more difficult.

- **embedded system** is part of a well-specified larger system (**intelligent product**)

RTCS usually implemented in µC, DSP, and FPGA



keyboard

LCD

real-time computer system (RTCS)

sensor

network

actuator

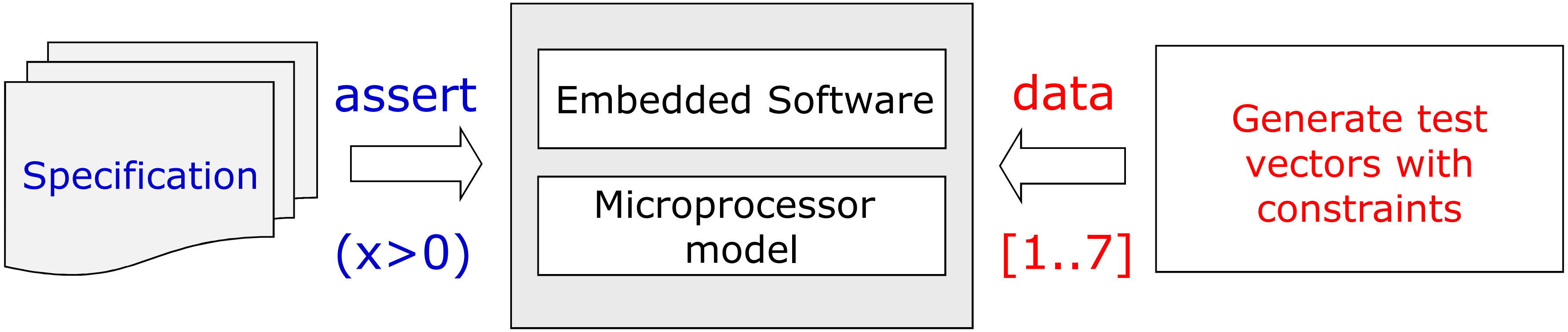
mass production

multi-core processors
limited amount of energy

software in read-only memory

Verification Challenges

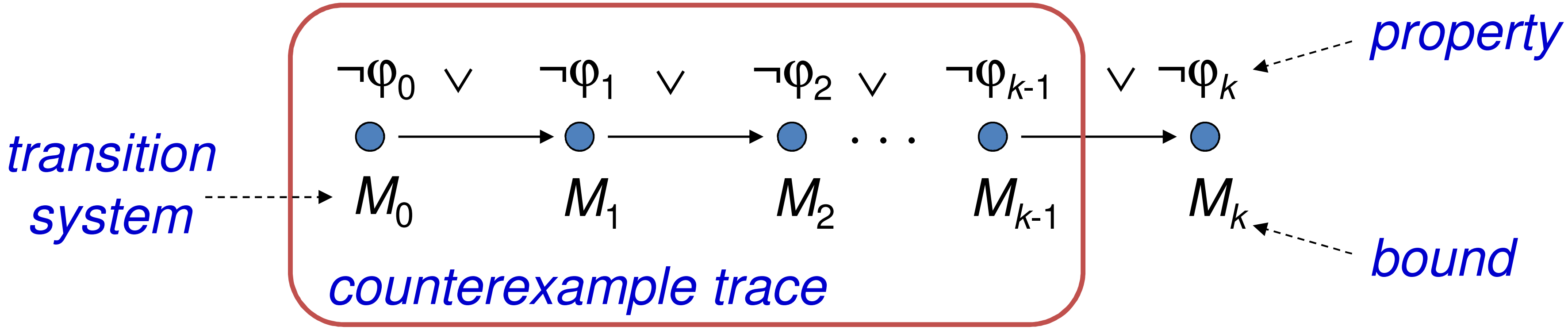
- **verification methodologies** for embedded systems



- verification of embedded systems raises **additional challenges**
  - handle concurrent software
  - meet time and energy constraints
  - evaluate implementation choices (design space exploration)
  - support legacy designs (usually written in low-level languages)
- improve **coverage** and reduce **verification time**

Bounded Model Checking (BMC)

Basic Idea: check negation of given property up to a given depth


$\neg\varphi_0 \vee \neg\varphi_1 \vee \neg\varphi_2 \vee \neg\varphi_{k-1} \vee \neg\varphi_k$
M_0
M_1
M_2
M_{k-1}
M_k
property
transition system
counterexample trace
bound

- transition system *M* unrolled *k* times
  - for programs: loops, arrays, …
- translated into verification condition ψ such that

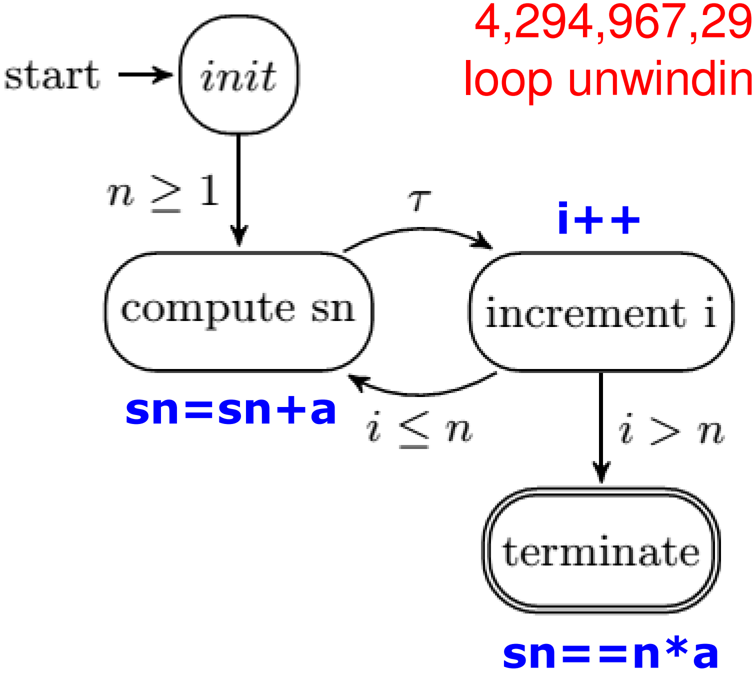
  ψ satisfiable iff φ has counterexample of max. depth *k*
- has been applied successfully to verify (embedded) software
  - main criticism is related to **completeness**

Difficulties in proving the correctness of programs with loops in BMC

- BMC tools typically fail to verify programs that contain **bounded and unbounded loops**
  - they can prove correctness only if an upper bound of *k* is known (**unwinding assertion**)

$$S_n = \sum_{i=1}^{n} a = na, n \geq 1$$

the loop will be unfolded 2^{n-1} times (in the worst case, 2^{32-1} times on 32 bits integer)



4,294,967,295 loop unwindings

Research Problem (RP)

- **(RP1)** provide suitable encoding into the Satisfiability Modulo Theories (SMT) by extending background theories (e.g., FP)
  - how to reason accurately about **heap-manipulating programs**?
- **(RP2)** exploit SMT techniques to leverage bounded model checking of concurrent software
  - how to exploit **unsat cores** to remove **redundant behaviour**?
- **(RP3)** prove correctness and timeliness (incl. energy) of embedded systems considering hardware constraints
  - how to check **system robustness** w.r.t. **implementation aspects**?
- **(RP4)** incorporate knowledge about system purpose and features to detect system-level and behaviour failures
  - how to model **target applications** or **system behaviour**?

Achievements (1)

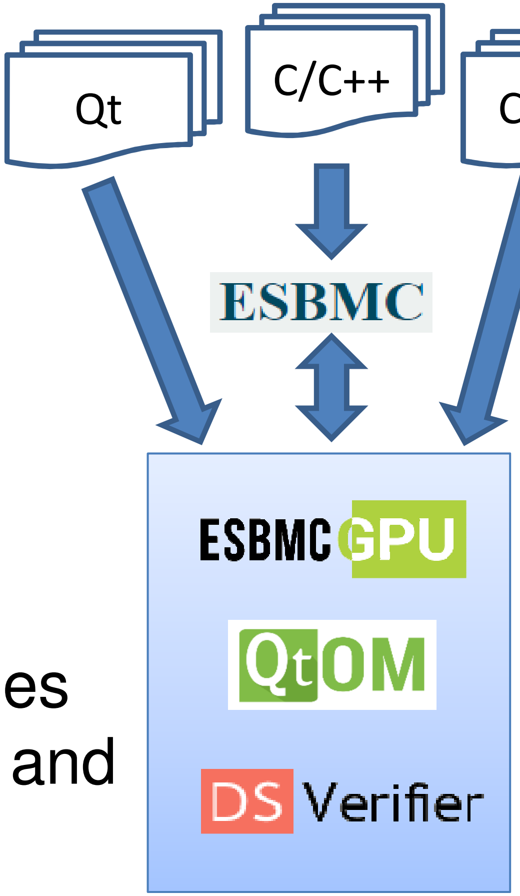
- **(RP1)** proposed the first SMT-based BMC for full C programs (ASE’09,TSE’12)
  - in addition to support C++98 (ECBS’13), CUDA (SAC’16), and Qt-based consumer electronics applications (SPIN’16)
  - memory management test-case generation of C programs using BMC (SEFM’15, TACAS’16)
    - ➢ coverage and verification time are still limited, especially for programs that contain **floating-point arithmetic** and **dynamic memory allocation**
- **(RP2)** proposed SMT-based context-BMC to verify deadlock, data races, lock acquisition ordering, and atomicity violations in multi-threaded software (ICSE’11)
  - considers **monotonic partial-order reduction** and **state-hashing techniques** to prune the state-space exploration
    - ➢ recent advances lead to BMC of multi-threaded C programs via **Lazy Sequentialization**

Achievements (2)

- **(RP3)** proposed a verification approach for (embedded) software using *k*-induction and invariants (TACAS'13, STTT'15, SBESC'15)
  - main challenge is to **compute** and **strengthen loop invariants** to prove program correctness and timeliness
    - ➢ exploiting the combination of different **invariant generation** algorithms to ensure **system robustness** w.r.t. **implementation aspects**
- **(RP4)** proposed SMT-based context-BMC to verify overflow, limit cycle, time constraints, stability, and minimum phase in digital systems (IECON'14, SPIN'15, DAES'16)
  - specify **system-level properties** using LTL (SEFM'11,SoSyM'15)
  - understand **programming bugs** using **counterexamples** (IFM'12)
  - **fault localization** in multi-threaded C programs (SBESC'15)
    - ➢ verify **cyber-physical systems** (computation, control, and communication)

Automated Software and Systems Verification Laboratory

- **ESBMC** is a BMC tool for embedded C/C++98 software based on SMT solvers (future release includes **clang**)
  - **ESBMC-GPU** checks concurrency errors in C/C++98/CUDA programs
  - **ESBMC-QtOM** checks C++ programs based on Qt cross-platform framework
  - **DSVerifier** checks low-level properties related to digital systems (closed-loop control systems)
- **ESBMC** is also able to prove properties for any given depth using *k*-induction and (inductive) invariants



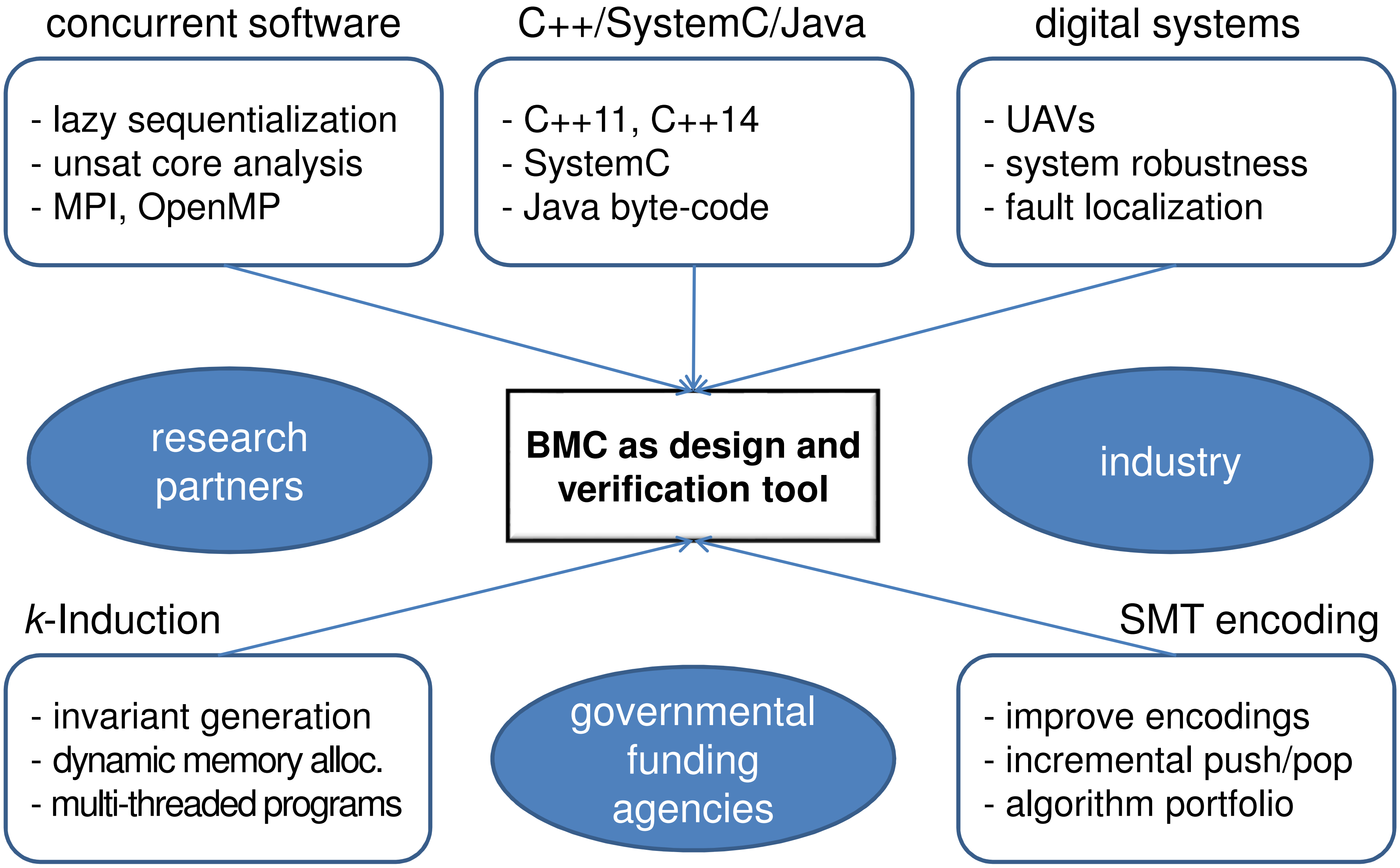
esbmc.org

esbmc-gpu.org

esbmc.org/qtom

dsverifier.org

Research & Development Plan



concurrent software

- lazy sequentialization
- unsat core analysis
- MPI, OpenMP

C++/SystemC/Java

- C++11, C++14
- SystemC
- Java byte-code

digital systems

- UAVs
- system robustness
- fault localization

research partners

BMC as design and verification tool

industry

k-Induction

- invariant generation
- dynamic memory alloc.
- multi-threaded programs

governmental funding agencies

SMT encoding

- improve encodings
- incremental push/pop
- algorithm portfolio