Pono: a Flexible and Extensible SMT-based Model Checker

Makai Mann, Ahmed Irfan, Florian Lonsing, Yahan Yang,
Hongce Zhang, Kris Brown, Aarti Gupta, Clark Barrett
Pono

• Definition: Right, correct, moral

• Solver-agnostic SMT-based model checker
Motivation

• SMT focused model checker
  • Supports SMT-LIB-like interface through Smt-Switch C++ API

• Use Cases
  • Push-button verification
  • Expert verification
  • Model checking development
Push-button techniques

- Competitive standard algorithms
- Bounded Model Checking (BMC)
  - + simple path check option
- K-Induction
- Interpolant-based model checking
- IC3 (suite of variants)
Flexibility

- User-guided invariant finding (de facto standard)
- Limitations of a black box
  - Translation step very important
  - Not always easily reducible to invariant checking
- Integrated verification — not a new idea, but hard to do right in practice
- Solution: flexible API for solving diverse problems
Prover engines

BMC
KInduction
InterpolantMC

IC3Base
IC3
IC3IA

CEG Prophecy

LEGEND

Directory
Base Class
Class
Flexibility: Prover

• Abstract **Prover** class interface
  • initialize
  • check_until(int k) – check property with resource limit
  • prove – no resource limit
  • witness – trace for counterexample
  • invar – inductive invariant (not supported by all algorithms)

• Inherited and implemented for each model checking algorithm
Flexibility: Inductive Datatypes

• Smt-switch - inductive datatypes with CVC4 backend

• Leverage generality of SMT interface

• Encode generalized algebraic theories (GATs) problem
  • Is there a sequence of rewrites that proves two terms are equal

• Undecidable, but BMC can check boundedly
Extensibility

• Flexibility for users

• Extensibility for developers

• Infrastructure
  • Open-source and simple
  • Serve as a research platform for experts
Extensibility: CEGAR

• Template-based class to extend model checking algorithm
  • cegar_abstract
  • cegar_refine

• CEGAR
  • Abstract such that a proof is valid, but a counterexample might not be
  • Loop: attempt to **prove** with abstraction, **refine** spurious counterexamples

• Two major options:
  • Restart underlying prover model checker
  • Incremental refinement (underlying prover not restarted)
Extensibility: IC3 Variants

• IC3/PDR leading technique for SAT/SMT based model checking
  • Lots of variations and optimizations
  • Lifted to SMT level in different ways

• IC3Base
  • IC3Formula: conjunction or disjunction of terms
  • inductive_generalization
  • predecessor_generalization
  • [optional] abstract
  • [optional] refine
Extensibility: IC3 Variants

- Implemented
  - IC3 (Boolean)
  - IC3Bits (split bit-vectors into bits)
  - ModelBasedIC3 (equalities with model values)
  - IC3 via Implicit Predicate Abstraction
  - IC3 with Syntax-Guided Abstraction – equality domain for hardware (BV only)
  - SyGuS-PDR – SyGuS-based inductive generalization for hardware (BV only)
Demo

- New algorithms: IC3-style variants
- New CEGAR loops: operator abstraction
Evaluation

• Compare against state-of-the-art tools on three categories
  • Infinite-state arrays
  • Linear integer and real arithmetic
  • Hardware

• 1 hour timeout, 16Gb memory, portfolios with individual processes
Evaluation: infinite-state arrays

- Constrained Horn Clause (CHC) benchmarks
  - Most required quantified invariants
- Implemented CEGAR algorithm Counterexample-Guided Prophecy
  - With IC3IA as the underlying prover
- Results on Freqhorn Array benchmarks
  - 81 total - all safe

<table>
<thead>
<tr>
<th></th>
<th>Pono</th>
<th>prophic3</th>
<th>prophic3-SA</th>
<th>freqhorn</th>
<th>nuXmv</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>71 (16s)</td>
<td>71 (20s)</td>
<td>66 (31s)</td>
<td>69 (6s)</td>
<td>4 (51s)</td>
</tr>
</tbody>
</table>
Evaluation: linear arithmetic

- Lustre benchmarks (integer) and SystemC benchmarks (real)

<table>
<thead>
<tr>
<th>result</th>
<th>SystemC (43 total)</th>
<th>Lustre (951 total)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pono</td>
<td>nuXmv</td>
</tr>
<tr>
<td>safe</td>
<td>18 (673s)</td>
<td>21 (571s)</td>
</tr>
<tr>
<td>unsafe</td>
<td>14 (325s)</td>
<td>15 (479s)</td>
</tr>
<tr>
<td>total</td>
<td>32 (521s)</td>
<td>36 (533s)</td>
</tr>
</tbody>
</table>
Evaluation: hardware

- Bit-vector only and Bit-vector + Arrays
- Compared to winners of last two competitions
  - + reference implementation of SyGuS-PDR
- Results on HWMCC2020 benchmarks

<table>
<thead>
<tr>
<th>result</th>
<th>BV (324 total)</th>
<th>BV + Array (315 total)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pono</td>
<td>AVR</td>
</tr>
<tr>
<td>safe</td>
<td>183 (283s)</td>
<td><strong>215</strong> (115s)</td>
</tr>
<tr>
<td>unsafe</td>
<td>47 (314s)</td>
<td>47 (219s)</td>
</tr>
<tr>
<td>total</td>
<td>230 (289s)</td>
<td><strong>262</strong> (133s)</td>
</tr>
</tbody>
</table>
Use Cases

• Research

• Academic/Industrial Verification Tool

• Teaching
Conclusion

• Solver-agnostic SMT-based model checker

• Use Cases
  • Push-button verification – performance
  • Expert verification – flexibility
  • Model checking development – extensibility
    • Controlled experiments