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

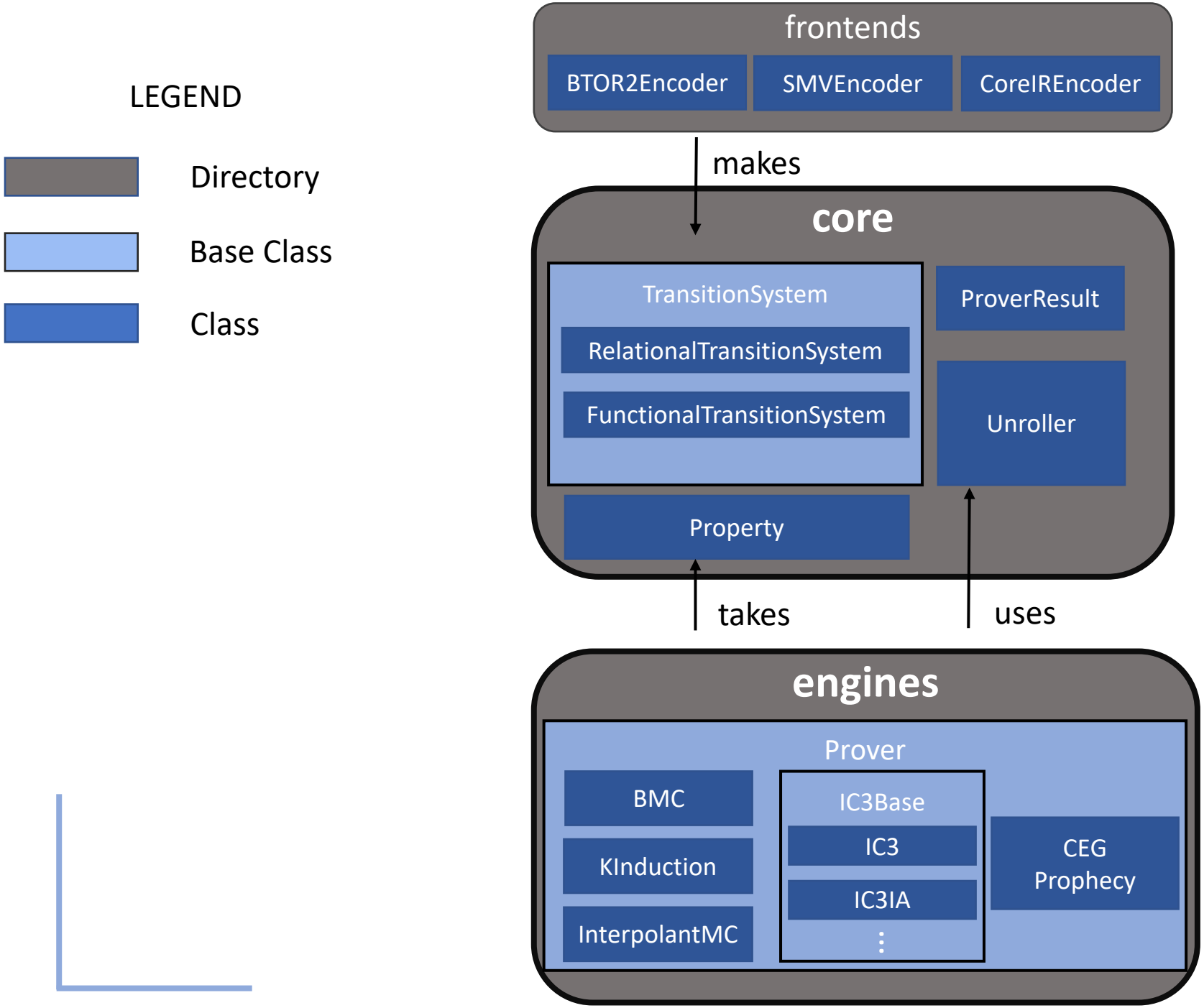
Push-button techniques

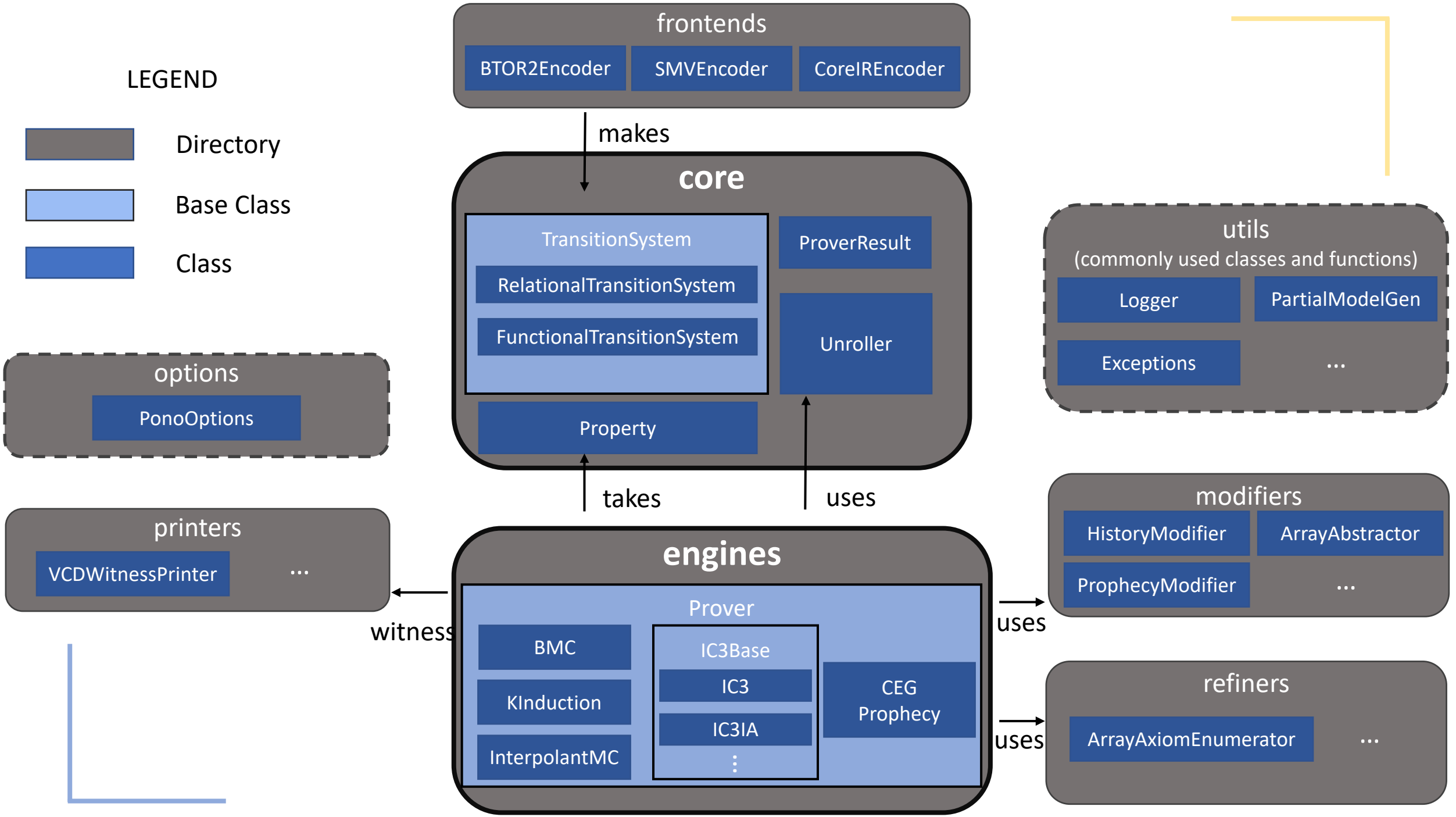
- Competitive standard algorithms

- 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


LEGEND
Directory
Base Class
Class
frontends
BTOR2Encoder
SMVEncoder
CoreIREncoder
makes
core
TransitionSystem
RelationalTransitionSystem
FunctionalTransitionSystem
ProverResult
Unroller
Property
takes
uses
engines
Prover
BMC
KInduction
InterpolantMC
IC3Base
IC3
IC3IA
CEG
Prophecy


LEGEND
Directory
Base Class
Class
frontends
BTOR2Encoder
SMVEncoder
CoreIREncoder
makes
core
TransitionSystem
RelationalTransitionSystem
FunctionalTransitionSystem
ProverResult
Unroller
Property
utils
(commonly used classes and functions)
Logger
PartialModelGen
Exceptions
...
options
PonoOptions
takes
uses
printers
VCDWitnessPrinter
...
engines
Prover
BMC
KInduction
InterpolantMC
IC3Base
IC3
IC3IA
CEG
Prophecy
witness
uses
modifiers
HistoryModifier
ArrayAbstractor
ProphecyModifier
...
uses
refiners
ArrayAxiomEnumerator
...

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

```
namespace pono {

Bmc::Bmc(const Property & p, SmtSolver & solver) : super(p, solver)
{
  initialize();
}

Bmc::Bmc(const PonoOptions & opt, const Property & p, smt::SmtSolver & solver)
    : super(opt, p, solver)
{
  initialize();
}

Bmc::~Bmc() {}

void Bmc::initialize()
{
  super::initialize();
  solver_->assert_formula(unroller_.at_time(ts_.init(), 0));
}

ProverResult Bmc::check_until(int k)
{
  for (int i = 0; i <= k; ++i) {
    if (!step(i)) {
      return ProverResult::FALSE;
    }
  }
  return ProverResult::UNKNOWN;
}

bool Bmc::step(int i)
{
  if (i <= reached_k_) {
    return true;
  }

  bool res = true;
  if (i > 0) {
    solver_->assert_formula(unroller_.at_time(ts_.trans(), i - 1));
  }

  solver_->push();
  logger.log(1, "Checking bmc at bound: {}", i);
  solver_->assert_formula(unroller_.at_time(bad_, i));
  Result r = solver_->check_sat();
  if (r.is_sat()) {
    res = false;
  } else {
    solver_->pop();
  }

  ++reached_k_;

  return res;
}
```

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

	Pono	prophic3	prophic3-SA	freqhorn	nuXmv
solved	**71** (16s)	**71** (20s)	66 (31s)	69 (6s)	4 (51s)

Evaluation: linear arithmetic

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

result	SystemC (43 total)		Lustre (951 total)		
	Pono	nuXmv	Pono	nuXmv	kind2
safe	18 (673s)	**21** (571s)	**521** (10s)	517 (8s)	508 (2s)
unsafe	14 (325s)	**15** (479s)	412 (5s)	412 (1s)	410 (0.2s)
total	32 (521s)	**36** (533s)	**933** (8s)	929 (5s)	918 (1s)

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

result	BV (324 total)				BV + Array (315 total)		
	Pono	AVR	CoSA2	sygus-apdr	Pono	AVR	CoSA2
safe	183 (283s)	**215** (115s)	98 (283s)	115 (545s)	252 (224s)	**274** (63s)	209 (299s)
unsafe	47 (314s)	47 (219s)	41 (232s)	15 (279s)	19 (207s)	19 (352s)	19 (204s)
total	230 (289s)	**262** (133s)	139 (268s)	130 (514s)	271 (223s)	**293** (82s)	228 (291s)

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