Enriching Software Verification with Analyses and Applications from Hardware

Nian-Ze Lee

LMU Munich, Germany

2022-10-06 @ CPAchecker Workshop

Who Am I?

- PhD at National Taiwan University in July 2021
  - Formal methods for electronic design automation
  - Visiting student at SoSy-Lab during 2020
- PostDoc at SoSy-Lab since October 2021

My Scientific Curiosities

- Hardware and software verification share many in common
  - Modeling: state-transition system
  - Approach: satisfiability, interpolation, etc
- New methods for SW verification
  - Adopt algorithms for HW verification
  - Represent programs as circuits and use HW verifiers
- New applications for SW verification
  - Represent circuits as programs

New Methods for Software Verification

Adopting HW-Verification Algorithms for SW

- Two new reachability analyses added to CPAchecker
  - IMC: based on *Interpolation and SAT-Based Model Checking*, K. L. McMillan, CAV 2003 [5]
  - ISMC: based on *Interpolation-Sequence Based Model Checking*, Y. Vizel and O. Grumberg, FMCAD 2009 [9]

Interpolation-Based Model Checking

- State-transition system: $I(s), T(s, s'), P(s)$

Interpolation-Based Model Checking

- State-transition system: $I(s), T(s, s'), P(s)$
- $I(s_0)T(s_0, s_1)\ T(s_1, s_2) \ldots T(s_{k-1}, s_k)\neg P(s_k)$

Interpolation-Based Model Checking

- State-transition system: $I(s), T(s, s'), P(s)$
- $\underbrace{I(s_0)T(s_0, s_1)}_{A(s_0,s_1)} \underbrace{T(s_1, s_2) \ldots T(s_{k-1}, s_k) \neg P(s_k)}_{B(s_1,s_2,\ldots,s_k)}$
- Interpolant $C_1(s_1)$: 1-step overapproximation

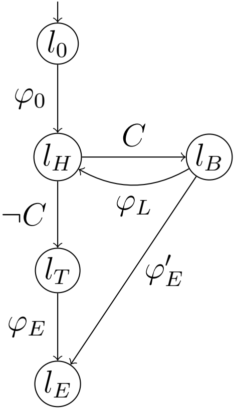
Interpolation-Based Model Checking

- State-transition system: $I(s), T(s, s'), P(s)$
- $\underbrace{I(s_0)T(s_0, s_1)}_{A(s_0,s_1)}\ \underbrace{T(s_1, s_2) \ldots T(s_{k-1}, s_k) \neg P(s_k)}_{B(s_1,s_2,\ldots,s_k)}$
- Interpolant $C_1(s_1)$: 1-step overapproximation
- $\underbrace{C_1(s_0)T(s_0, s_1)}_{A'(s_0,s_1)}\ \underbrace{T(s_1, s_2) \ldots T(s_{k-1}, s_k) \neg P(s_k)}_{B'(s_1,s_2,\ldots,s_k)}$
  - Interpolant $C_2(s_1)$: 2-step overapproximation
  - Repeat until $\bigvee C_i$ becomes a fixed point
  - Increment k if query becomes satisfiable

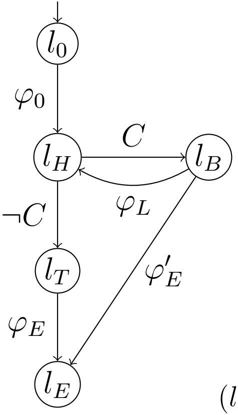
Interpolation-Sequence-Based Model Checking

- $\underbrace{I(s_0)T(s_0,s_1)\ldots}_{A(s_0,s_1,\ldots,s_i)}\underbrace{T(s_i,s_{i+1})\ldots T(s_{k-1},s_k)\neg P(s_k)}_{B(s_i,s_{i+1},\ldots,s_k)}$
- Interpolation-sequence $C_1, C_2, \ldots, C_k$
  - Interpolant C_i: i-step overapproximation
  - Repeat until $\bigvee C_i$ becomes a fixed point
- Similar to Impact [6] (with stop^{join})

Single-Loop CFA summarized by LBE

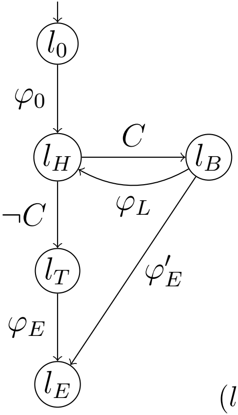


Single-Loop CFA summarized by LBE



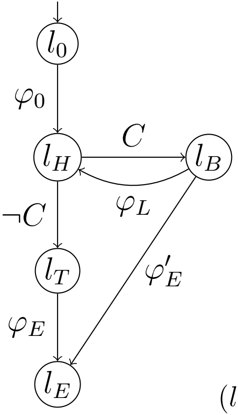
$$\begin{tikzcd}
(l_0, true) \arrow[d] \\
(l_H, \varphi_0|_{i=0}) \arrow[d] \\
(l_H, (C \land \varphi_L)|_{i=1}) \arrow[d] \\
(l_H, (C \land \varphi_L)|_{i=2}) \arrow[d, dotted] \\
(l_H, (C \land \varphi_L)|_{i=k}) \arrow[d] \\
(l_E, ((\neg C \land \varphi_E) \lor (C \land \varphi'_E))|_{i=k})
\end{tikzcd}$$

Single-Loop CFA summarized by LBE



$$\begin{tikzcd}
(l_0, true) \arrow[d] & \\
(l_H, \varphi_0|_{i=0}) \arrow[d] & : I(s_0) \\
(l_H, (C \wedge \varphi_L)|_{i=1}) \arrow[d] & : T(s_0, s_1) \\
(l_H, (C \wedge \varphi_L)|_{i=2}) \arrow[d, dotted] & : T(s_1, s_2) \\
(l_H, (C \wedge \varphi_L)|_{i=k}) \arrow[d] & : T(s_{k-1}, s_k) \\
(l_E, ((\neg C \wedge \varphi_E) \vee (C \wedge \varphi'_E))|_{i=k}) & : \neg P(s_k)
\end{tikzcd}$$

Single-Loop CFA summarized by LBE



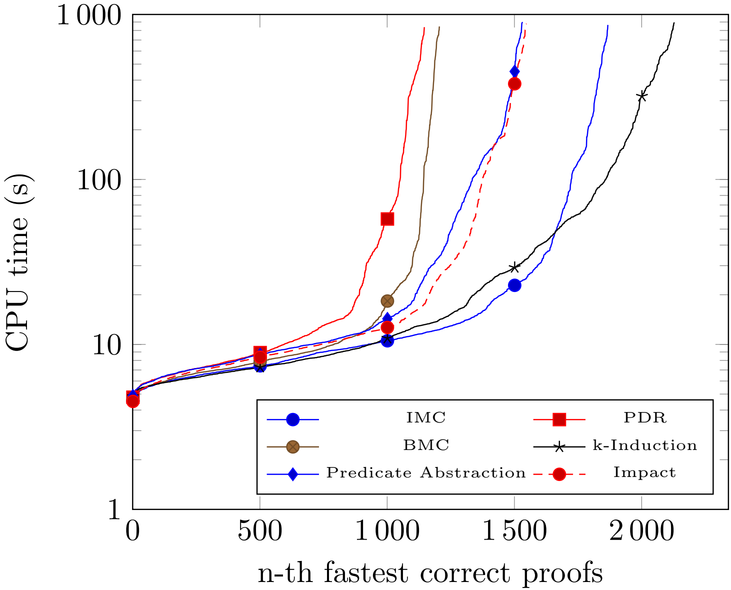
$$\begin{tikzcd}
(l_0, \mathit{true}) \arrow[d] & \\
(l_H, \varphi_0|_{i=0}) \arrow[d] & : {\color{red} I(s_0)} \\
(l_H, (C \land \varphi_L)|_{i=1}) \arrow[d] & : {\color{red} T(s_0, s_1)} \\
(l_H, (C \land \varphi_L)|_{i=2}) \arrow[d, dotted] & : {\color{red} T(s_1, s_2)} \\
(l_H, (C \land \varphi_L)|_{i=k}) \arrow[d] & : {\color{red} T(s_{k-1}, s_k)} \\
(l_E, ((\neg C \land \varphi_E) \lor (C \land \varphi'_E))|_{i=k}) & : {\color{red} \neg P(s_k)}
\end{tikzcd}$$

Solution for multi-loop programs:
standard transformation to single loop

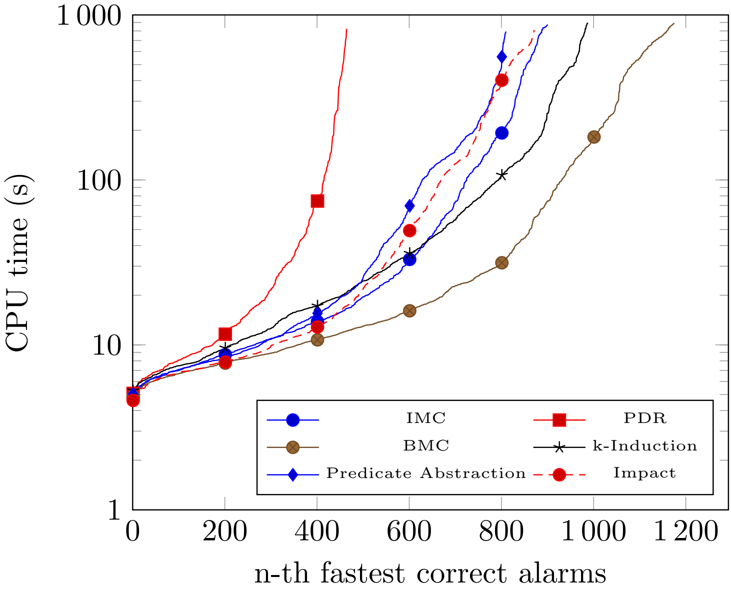
Experimental Setup

- CPAchecker revision 40806
- Interpolants provided by MathSAT 5
- Compared algorithms
  - IMC
  - PDR
  - BMC
  - k-Induction
  - Predicate abstraction
  - Impact
- Subset of *ReachSafety* from SV-COMP '22 [1]
  - Safe: 4234 tasks
  - Unsafe: 1793 tasks

Quantile Plot: Safe Tasks



Quantile Plot: Unsafe Tasks



New Applications for Software Verification

Verifying Hardware with Software Model Checkers

- Hardware systems usually described as a *sequential circuit*
  - Sequential elements (registers, storing state variables)
  - Combinational circuitry (property and next-state logic)
- Operation
  - Initialize state variables (reset)
  - Evaluate property and compute next states

Can be modeled as a single-loop program

Hardware Description Languages

- Verilog
- VHDL
- . . .

Which frontend language should be supported?

The Btor2 [8] Language

- Word-level sequential circuits
  - *Cf.* Bit-level AIGER format
- Bit-vector and array
- Used in the Hardware Model Checking Competitions

Example

```
1 sort bitvec 3
2 zero 1
3 state 1
4 init 1 3 2
5 one 1
6 add 1 3 5
7 next 1 3 6
8 ones 1
9 sort bitvec 1
10 eq 9 3 8
11 bad 10
```

```
extern void abort(void);
void main() {
  typedef unsigned char SORT_1;
  typedef unsigned char SORT_9;
  const SORT_1 var_2 = 0;
  const SORT_1 var_5 = 1;
  const SORT_1 var_8 = 0b111;
  SORT_1 state_3 = var_2;
  for (;;) {
    SORT_9 var_10 = state_3 == var_8;
    SORT_9 bad_11 = var_10;
    if (bad_11) {
      ERROR: abort();
    }
    SORT_1 var_6 = state_3 + var_5;
    var_6 = var_6 & 0b111;
    state_3 = var_6;
  }
}
```

BTOR2C: A Converter from Btor2 to C

- Implemented in C
- Supports all Btor2 constructs
- Converted more than a thousand tasks from HWMCC (which will be submitted to SV-COMP this year)

Verilog vs. Btor2

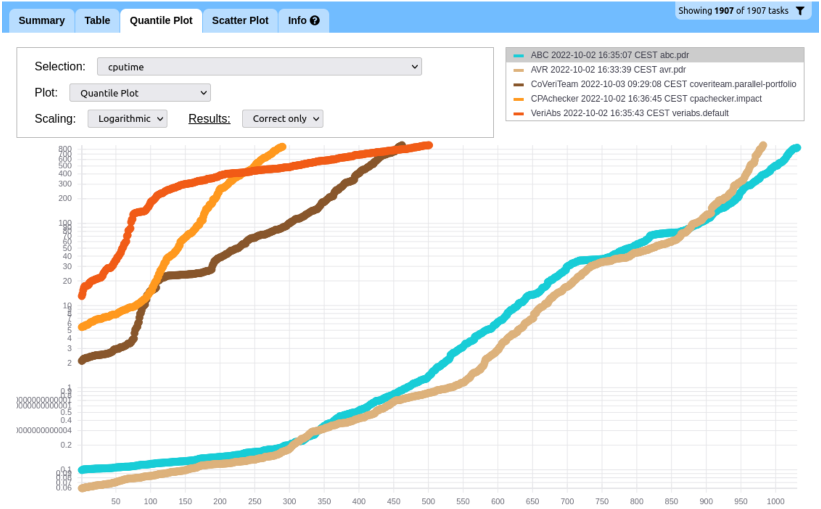
- Previous works [4, 7] use Verilog as frontend
- Benefits of Btor2
  - Simple; suitable for verification (IR)
  - Many tasks and tools from HWMCC

In practice, Yosys [10] can translate Verilog to Btor2

Experimental Setup

- Converted 1907 tasks from HWMCC and prior work [7]
  - 1593 tasks with only bit-vectors
  - 314 tasks with both arrays and bit-vectors
  - 575 unsafe; 1332 safe
- Verifiers
  - Bit-level model checker: ABC [2]
  - Word-level model checker: AVR [3]
  - Software verifiers
    - CPAchecker
    - CoVeriTeam (parallel: CPA-Seq, Esbmc, Symbiotic)
    - VeriAbs

Comparing SW and HW verifiers



Showing **1907** of 1907 tasks

Selection: cputime

Plot: Quantile Plot

Scaling: Logarithmic

Results: Correct only

- ABC 2022-10-02 16:35:07 CEST abc.pdr
- AVR 2022-10-02 16:33:39 CEST avr.pdr
- CoVeriTeam 2022-10-03 09:29:08 CEST coveriteam.parallel-portfolio
- CPAchecker 2022-10-02 16:36:45 CEST cpachecker.impact
- VeriAbs 2022-10-02 16:35:43 CEST veriabs.default

Observations

- Underlying solvers: SAT vs. SMT
- Algorithms: IC3/PDR vs. other algorithms
- Task representation: transition relation vs. arbitrary CFA

Conclusion

- What have been done?
  - New algorithms IMC and ISMC in CPAchecker
- What is ongoing?
  - Verifying hardware with CPAchecker via Btor2C
- What will be done?
  - More new approaches (especially, IC3/PDR)
  - Verifying programs with HW verifiers

References I

Beyer, D.: Progress on software verification: SV-COMP 2022. In: Proc. TACAS (2). pp. 375–402. LNCS 13244, Springer (2022). https://doi.org/10.1007/978-3-030-99527-0_20

Brayton, R.K., Mishchenko, A.: ABC: an academic industrial-strength verification tool. In: Proc. CAV. pp. 24–40. LNCS 6174, Springer (2010). https://doi.org/10.1007/978-3-642-14295-6_5

Goel, A., Sakallah, K.A.: AVR: abstractly verifying reachability. In: Proc. TACAS. pp. 413–422. LNCS 12078, Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_23

Greaves, D.J.: A Verilog to C compiler. In: Proc. RSP. pp. 122–127. IEEE Computer Society (2000). https://doi.org/10.1109/IWRSP.2000.855208

McMillan, K.L.: Interpolation and SAT-based model checking. In: Proc. CAV. pp. 1–13. LNCS 2725, Springer (2003). https://doi.org/10.1007/978-3-540-45069-6_1

McMillan, K.L.: Lazy abstraction with interpolants. In: Proc. CAV. pp. 123–136. LNCS 4144, Springer (2006). https://doi.org/10.1007/11817963_14

Mukherjee, R., Tautschnig, M., Kroening, D.: v2c - A Verilog to C translator. In: Proc. TACAS. pp. 580–586. LNCS 9636, Springer (2016). https://doi.org/10.1007/978-3-662-49674-9_38

References II

Niemetz, A., Preiner, M., Wolf, C., Biere, A.: Btor2, BtorMC and Boolector 3.0. In: Proc. CAV. pp. 587–595. LNCS 10981, Springer (2018). https://doi.org/10.1007/978-3-319-96145-3_32

Vizel, Y., Grumberg, O.: Interpolation-sequence based model checking. In: Proc. FMCAD. pp. 1–8. IEEE (2009). https://doi.org/10.1109/FMCAD.2009.5351148

Wolf, C.: Yosys open synthesis suite. http://www.clifford.at/yosys/