CPV: A Circuit-Based Program Verifier

Po-Chun Chien[1] and Nian-Ze Lee[2,1]

[1]LMU Munich · [2]National Taiwan University

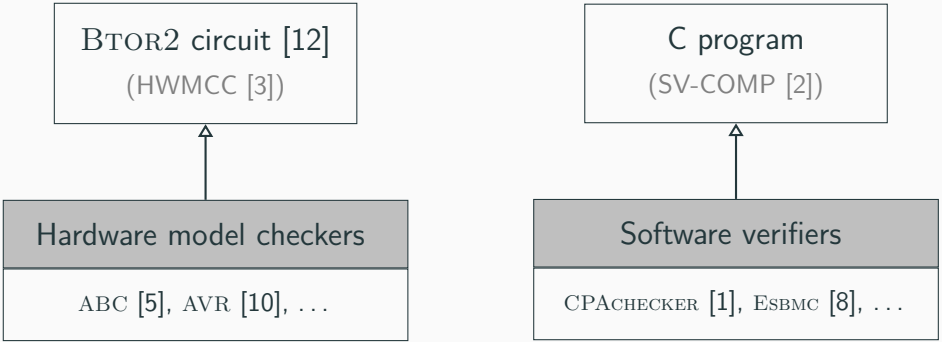
SV-COMP 2025 @ Hamilton, Canada

Motivation

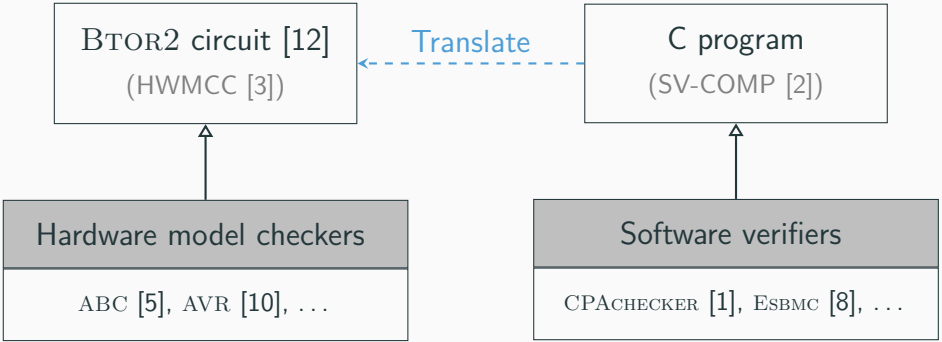
BTOR2 circuit [12]
(HWMCC [3])

C program
(SV-COMP [2])

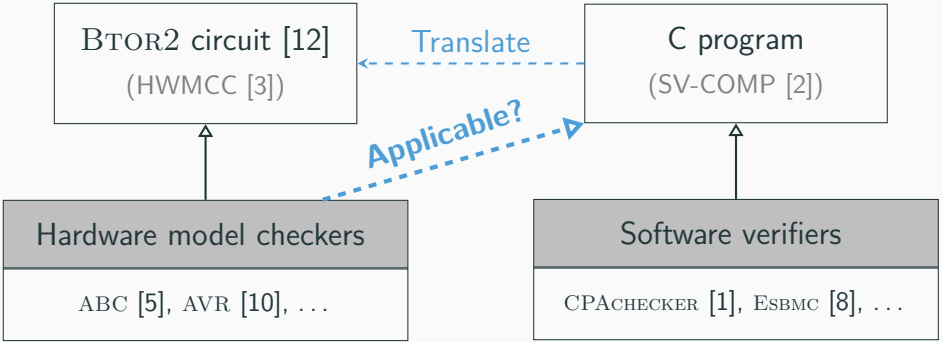
Motivation


BTOR2 circuit [12]
(HWMCC [3])
C program
(SV-COMP [2])
Hardware model checkers
ABC [5], AVR [10], . . .
Software verifiers
CPAchecker [1], Esbmc [8], . . .

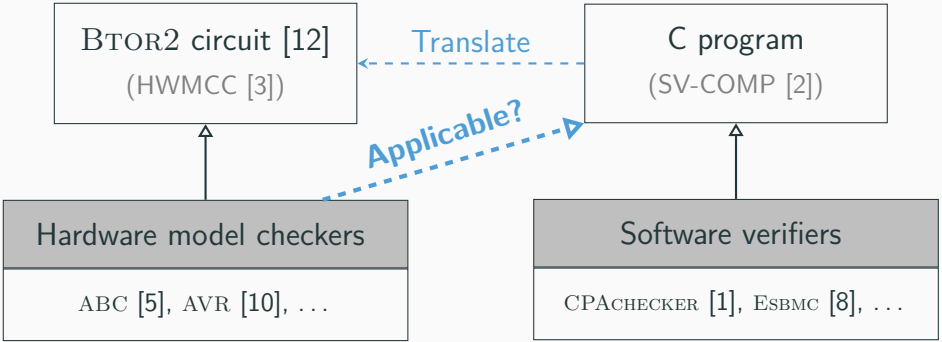
Motivation


BTOR2 circuit [12]
(HWMCC [3])
Translate
C program
(SV-COMP [2])
Hardware model checkers
ABC [5], AVR [10], . . .
Software verifiers
CPAchecker [1], Esbmc [8], . . .

Motivation

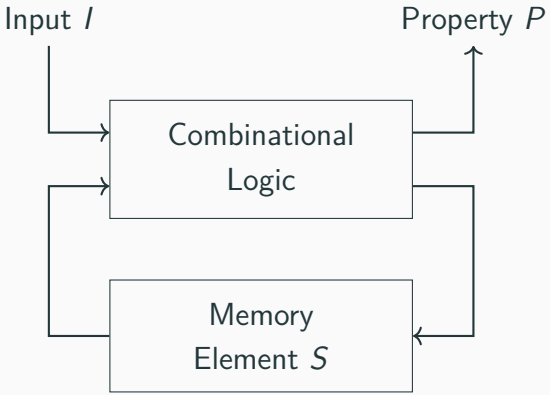

BTOR2 circuit [12]
(HWMCC [3])
Translate
C program
(SV-COMP [2])
Applicable?
Hardware model checkers
ABC [5], AVR [10], . . .
Software verifiers
CPAchecker [1], Esbmc [8], . . .

Motivation



Could circuits serve as IR for program verification?

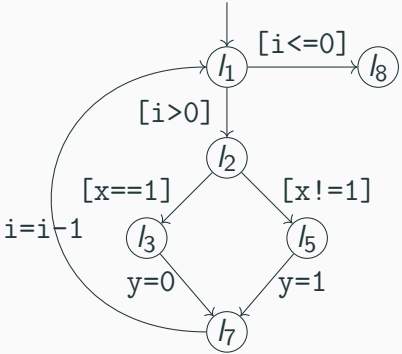
Hardware Model: Sequential Circuit



- State transition:
  $S' \leftarrow T_{func}(S, I)$
- Property:
  $P(S)$ or $P(S, I)$

Large-Block Encoding

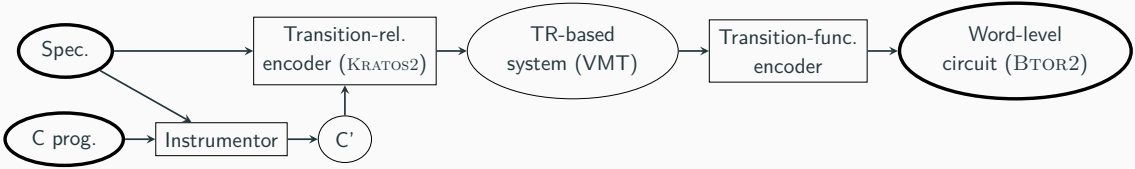
C Program → Transition Relation



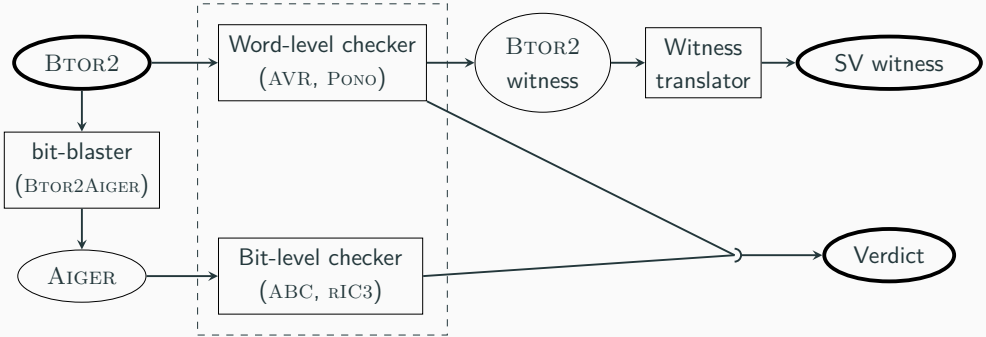
Each loop-free section as a block:

$$
\begin{aligned}
TR = (&pc = l_1 \wedge pc' = l_1 \wedge i > 0 \\
&\wedge i' = i - 1 \wedge y' = ite(x = 1, 0, 1) \ldots) \\
\vee\, (&pc = l_1 \wedge pc' = l_8 \wedge i \leq 0 \wedge i' = i \ldots) \\
\vee\, &\ldots
\end{aligned}
$$

System Architecure: Frontend



System Architecure: Backend

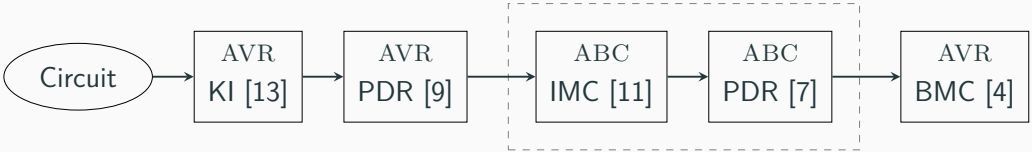


Managed by COVERITEAM

Strategy for SV-COMP

A sequential portfolio consisting of

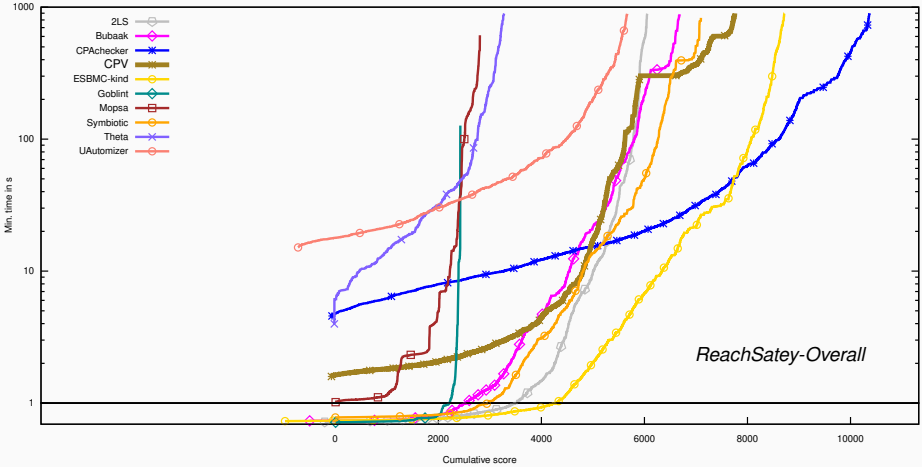
- Different encodings: functional and relational
- Different model-checking engines:



if BTOR2-to-AIGER translation succeeds

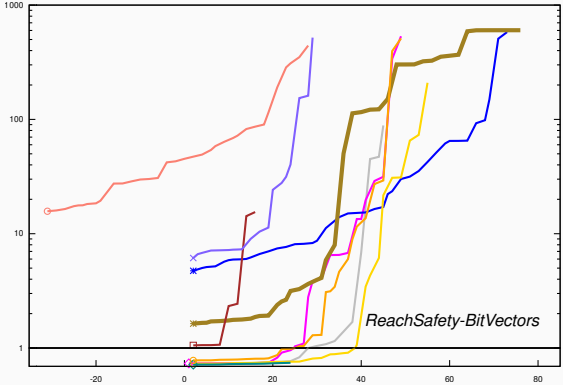
Results in SV-COMP

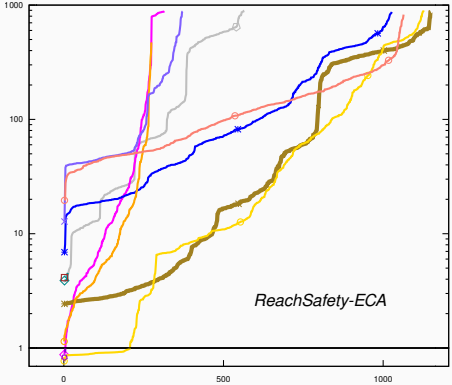
3rd place in *ReachSafety* category (2×, 2×, and 2× in subcategories)



Results in SV-COMP

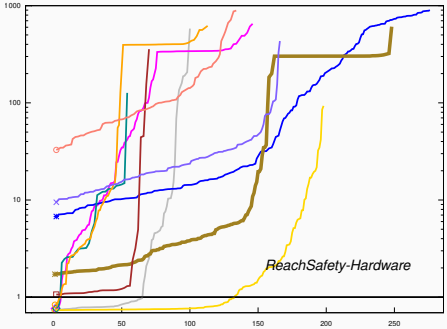
🥇 **1st** place in *ReachSafety-BitVectors* and *-ECA*

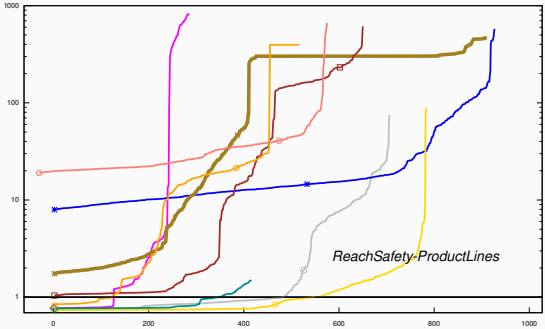




Results in SV-COMP

- **2nd** place in *ReachSafety-Hardware* and *-ProductLines*





Conclusion

- Our verifier CPV [6]
  - encodes programs into circuits and
  - employs hardware model checkers as backend.
- Pretty good performance in SV-COMP!
- Ongoing development:
  - Support termination analysis
  - Integrate more backends from HWMCC

gitlab.com/sosy-lab/software/cpv

References i

[1] Beyer, D., Keremoglu, M.E.: CPAchecker: A tool for configurable software verification. In: Proc. CAV. pp. 184–190. LNCS 6806, Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_16

[2] Beyer, D., Strejček, J.: Improvements in software verification and witness validation: SV-COMP 2025. In: Proc. TACAS (3). pp. 151–186. LNCS 15698, Springer (2025). https://doi.org/10.1007/978-3-031-90660-2_9

[3] Biere, A., Froleyks, N., Preiner, M.: Hardware model checking competition 2024. In: Proc. FMCAD. pp. 7–7. TU Wien Academic Press (2024). https://doi.org/10.34727/2024/isbn.978-3-85448-065-5_6

[4] Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Advances in Computers **58**, 117–148 (2003). https://doi.org/10.1016/S0065-2458(03)58003-2

[5] Brayton, R., 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

References ii

[6] Chien, P.C., Lee, N.Z.: CPV: A circuit-based program verifier (competition contribution). In: Proc. TACAS (3). pp. 365–370. LNCS 14572, Springer (2024). https://doi.org/10.1007/978-3-031-57256-2_22

[7] Eén, N., Mishchenko, A., Brayton, R.K.: Efficient implementation of property directed reachability. In: Proc. FMCAD. pp. 125–134. FMCAD Inc. (2011). https://dl.acm.org/doi/10.5555/2157654.2157675

[8] Gadelha, M.R., Monteiro, F.R., Morse, J., Cordeiro, L.C., Fischer, B., Nicole, D.A.: ESBMC 5.0: An industrial-strength C model checker. In: Proc. ASE. pp. 888–891. ACM (2018). https://doi.org/10.1145/3238147.3240481

[9] Goel, A., Sakallah, K.: Model checking of Verilog RTL using IC3 with syntax-guided abstraction. In: Proc. NFM. pp. 166–185. Springer (2019). https://doi.org/10.1007/978-3-030-20652-9_11

[10] Goel, A., Sakallah, K.: 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

References iii

[11] 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

[12] 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

[13] Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-solver. In: Proc. FMCAD, pp. 127–144. LNCS 1954, Springer (2000). https://doi.org/10.1007/3-540-40922-X_8