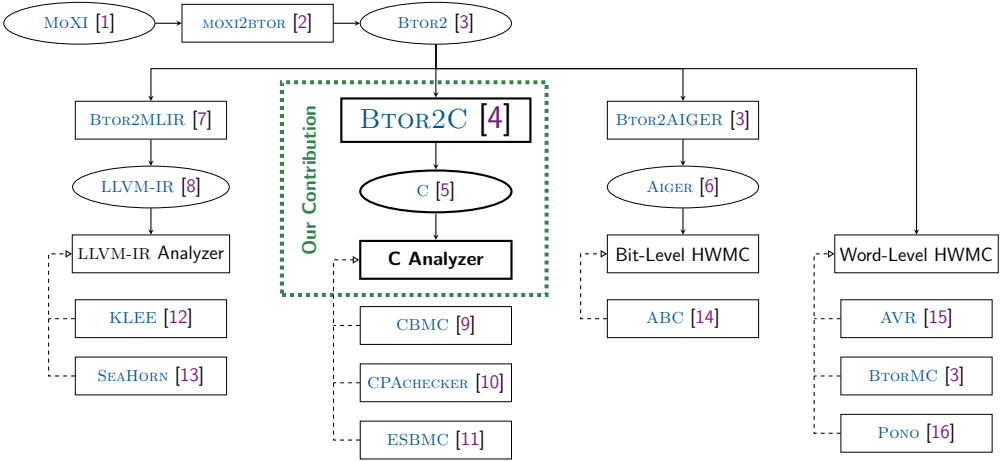
Extending the MoXI Eco System

Dirk Beyer, Po-Chun Chien, and Nian-Ze Lee
LMU Munich, Germany

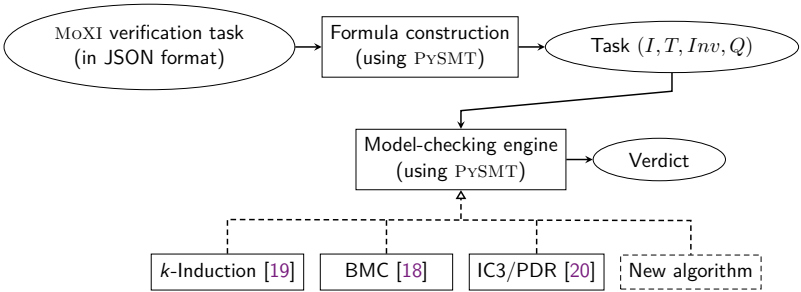
July 23, 2024, at OSSyM @CAV 2024

(i) Btor2C: Enabling C Verifiers to Analyze MoXI Models [4]



There are **26 verifiers for C available** for *ReachSafety-Hardware* tasks (see SV-COMP 2024 [17]). $\Rightarrow$ 26 new MoXI model checkers via translation

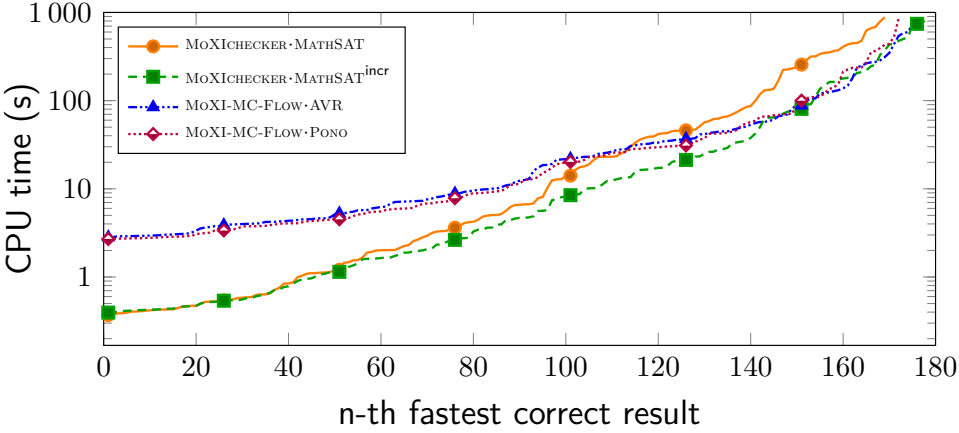
(ii) MoXIchecker: An Extensible Model Checker for MoXI [22]



- verifies MoXI models directly (no translation step involved)
- implemented in Python, based on PySMT [21]
- extensible, easy to add new algorithms and theories

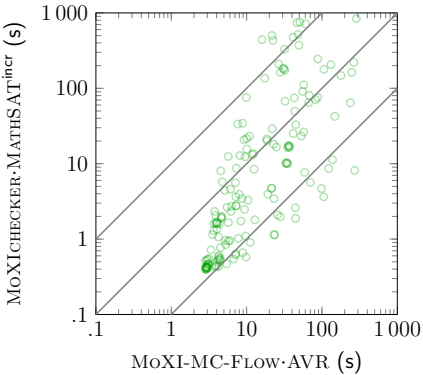
MoXIchecker: Performance

Comparison with MoXI-MC-Flow on 382 QF_BV tasks (also incremental solving)

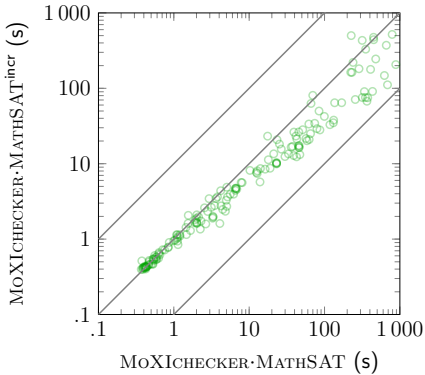


⇒ Direct implementation can be faster (think of CI [23])

MoXIchecker: Response Time and Extensibility



Efficiency of
MoXIchecker vs. MoXI-MC-Flow
⇒ Response Time



Effect of incremental SMT solving in
MoXIchecker
⇒ Extensibility

MoXIchecker: Effectiveness and Extensibility (Solvers, Algs)

Evaluation on 382 QF_BV tasks:

Tool	MoXIchecker				MoXI-MC-Flow	
Backend	MathSAT	MathSAT$^{\text{incr}}$	Z3	Z3$^{\text{incr}}$	AVR	Pono
Correct results	169	**178**	169	174	175	172
Proofs	44	46	**47**	**47**	45	45
Alarms	125	**132**	122	127	130	127

$\Rightarrow$ competitive to MoXI-MC-Flow
$\Rightarrow$ can serve as a base line
$\Rightarrow$ ideal for teaching due to slim architecture

MoXIchecker: Extensibility (Theories of integers, reals)

Task	Theory	Verdict	MoXIchecker	MoXI-MC-Flow
FibonacciSequence	QF_LIA	safe	safe	unsafe
IntIncrement	QF_LIA	unsafe	unsafe	safe
IntCounter	QF_LIA	safe	safe	timeout
IntMultiply	QF_NIA	safe	safe	unsafe
BoundedLinearGrowth	QF_LRA	safe	safe	unsupported
DoubleDelay2	QF_LRA	unsafe	unsafe	unsupported
OscillatingRatio	QF_NRA	safe	safe	unsupported
SafeNonlinearGrowth	QF_NRA	safe	safe	unsupported
NonlinearGrowth	QF_NRA	unsafe	unsafe	unsupported

$\Rightarrow$ extensible compared to MoXI-MC-Flow

$\Rightarrow$ sound compared to MoXI-MC-Flow's approximation

(iii) MoXI Competition

Should we create a competition of MoXI model checkers?
There are already ≥ 2 model checkers for MoXI.

$\Rightarrow$ drives standardization

$\Rightarrow$ maintains benchmark sets

Conclusion

- BTOR2C [4]: enables transformation MoXI → C
  ⇒ we can use 26 C verifiers for MoXI
  Status: not yet evaluated
- MoXIchecker [22]
  ⇒ extensible framework for direct MoXI model checking
  Status: software released and paper published
- MoXI-Comp
  ⇒ annual comparative evaluation of the MoXI state of the art
  Status: just proposed

MoXIchecker @GitLab

MoXIchecker @arXiv

https://gitlab.com/sosy-lab/software/moxichecker

References I

[1] Rozier, K.Y., Dureja, R., Irfan, A., Johannsen, C., Nukala, K., Shankar, N., Tinelli, C., Vardi, M.Y.: MoXI: An intermediate language for symbolic model checking. In: Proc. SPIN. LNCS , Springer (2024)

[2] Johannsen, C., Nukala, K., Dureja, R., Irfan, A., Shankar, N., Tinelli, C., Vardi, M.Y., Rozier, K.Y.: Symbolic model-checking intermediate-language tool suite. In: Proc. CAV. LNCS , Springer (2024)

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

[4] Beyer, D., Chien, P.C., Lee, N.Z.: Bridging hardware and software analysis with Btor2C: A word-level-circuit-to-C translator. In: Proc. TACAS (2). pp. 152–172. LNCS 13994, Springer (2023). doi:10.1007/978-3-031-30820-8_12

[5] ISO/IEC JTC 1/SC 22: ISO/IEC 9899-2018: Information technology — Programming Languages — C. International Organization for Standardization (2018), `https://www.iso.org/standard/74528.html`

[6] Biere, A.: The AIGER And-Inverter Graph (AIG) format version 20071012. Tech. Rep. 07/1, Institute for Formal Models and Verification, Johannes Kepler University (2007). doi:10.35011/fmvtr.2007-1

[7] Tafese, J., Garcia-Contreras, I., Gurfinkel, A.: Btor2MLIR: A format and toolchain for hardware verification. In: Proc. FMCAD. pp. 55–63. IEEE (2023). doi:10.34727/2023/ISBN.978-3-85448-060-0_13

[8] Lattner, C., Adve, V.S.: LLVM: A compilation framework for lifelong program analysis and transformation. In: Proc. CGO. pp. 75–88. IEEE (2004). doi:10.1109/CGO.2004.1281665

References II

[9] Clarke, E.M., Kröning, D., Lerda, F.: A tool for checking ANSI-C programs. In: Proc. TACAS. pp. 168–176. LNCS 2988, Springer (2004). doi:10.1007/978-3-540-24730-2_15

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

[11] 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). doi:10.1145/3238147.3240481

[12] Cadar, C., Dunbar, D., Engler, D.R.: Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: Proc. OSDI. pp. 209–224. USENIX Association (2008), https://dl.acm.org/doi/10.5555/1855741.1855756

[13] Gurfinkel, A., Kahsai, T., Komuravelli, A., Navas, J.A.: The SeaHorn verification framework. In: Proc. CAV. pp. 343–361. LNCS 9206, Springer (2015). doi:10.1007/978-3-319-21690-4_20

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

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

References III

[16] Mann, M., Irfan, A., Lonsing, F., Yang, Y., Zhang, H., Brown, K., Gupta, A., Barrett, C.W.: Pono: A flexible and extensible SMT-based model checker. In: Proc. CAV. pp. 461–474. LNCS 12760, Springer (2021). doi:10.1007/978-3-030-81688-9_22

[17] Beyer, D.: State of the art in software verification and witness validation: SV-COMP 2024. In: Proc. TACAS (3). pp. 299–329. LNCS 14572, Springer (2024). doi:10.1007/978-3-031-57256-2_15

[18] Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: Proc. TACAS. pp. 193–207. LNCS 1579, Springer (1999). doi:10.1007/3-540-49059-0_14

[19] 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). doi:10.1007/3-540-40922-X_8

[20] Bradley, A.R.: SAT-based model checking without unrolling. In: Proc. VMCAI. pp. 70–87. LNCS 6538, Springer (2011). doi:10.1007/978-3-642-18275-4_7

[21] Gario, M., Micheli, A.: PySMT: A solver-agnostic library for fast prototyping of SMT-based algorithms. In: Proc. SMT (2015)

[22] Beyer, D., Chien, P.C., Lee, N.Z.: MoXIchecker: An extensible model checker for MoXI. arXiv/CoRR **2407**(15551) (July 2024). doi:10.48550/arXiv.2407.15551

References IV

[23] Chong, N., Cook, B., Eidelman, J., Kallas, K., Khazem, K., Monteiro, F.R., Schwartz-Narbonne, D., Tasiran, S., Tautschnig, M., Tuttle, M.R.: Code-level model checking in the software development workflow at Amazon Web Services. Softw. Pract. Exp. **51**(4), 772–797 (2021). doi:10.1002/spe.2949