Exploring the Configuration Jungle for Invariant Checking in nuXmv

Bernhard Luedtke
Software Technologies Research Group
Otto-Friedrich-Universität Bamberg

Automatic Verification for the SWTbahn

- Students upload models for train engines and interlockers
  - SCChart [vHDM+14] and BahnDSL[1] models
- How to prevent upload and use of “unsafe” models?
  - 1. Compile model to SMV
  - 2. Verify safety properties, e.g., train does not exceed speed limit
  - 3. Permit model if verification succeeds, otherwise reply with counterexample(s)
- Using nuXmv [CCD+14] as the model checker

1: BahnDSL: A Domain-Specific Language for Configuring and Modelling Model Railways. https://github.com/trinnguyen/bahndsl

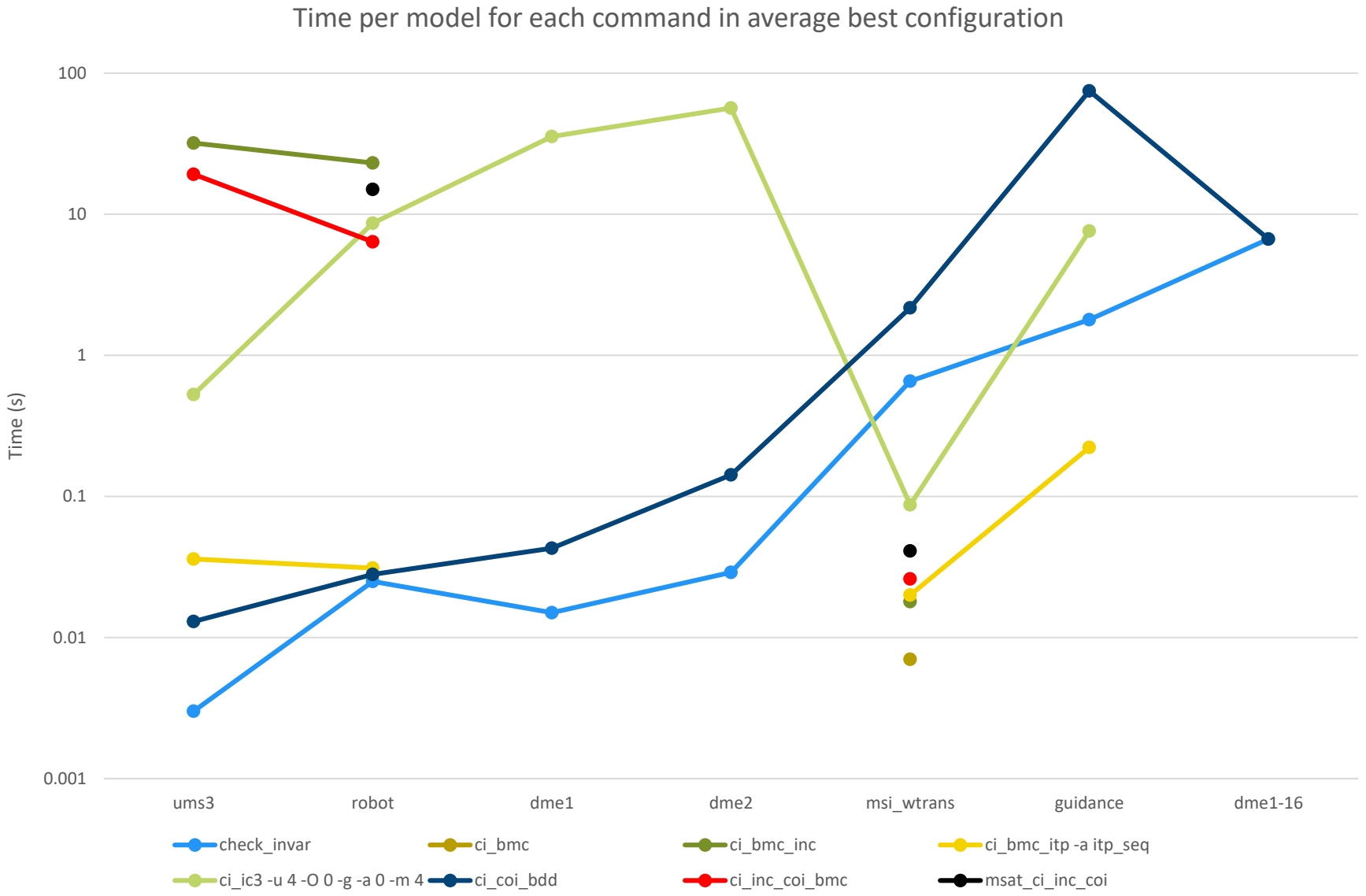
nuXmv Usage Scenario

- Our situation is thus:
  - We do not know the models ahead of time
  - We want the verification process to be automatic and fast
- What invariant checking commands should we use, and how should they be configured?
- *Short-term interest:*
  - Improve model railway ease-of-use
- **Long-term interest:**
  - Can we *auto-configure* nuXmv based on model characteristics? (cf. Storm decision-tree based *automatic* engine [HJK+21])

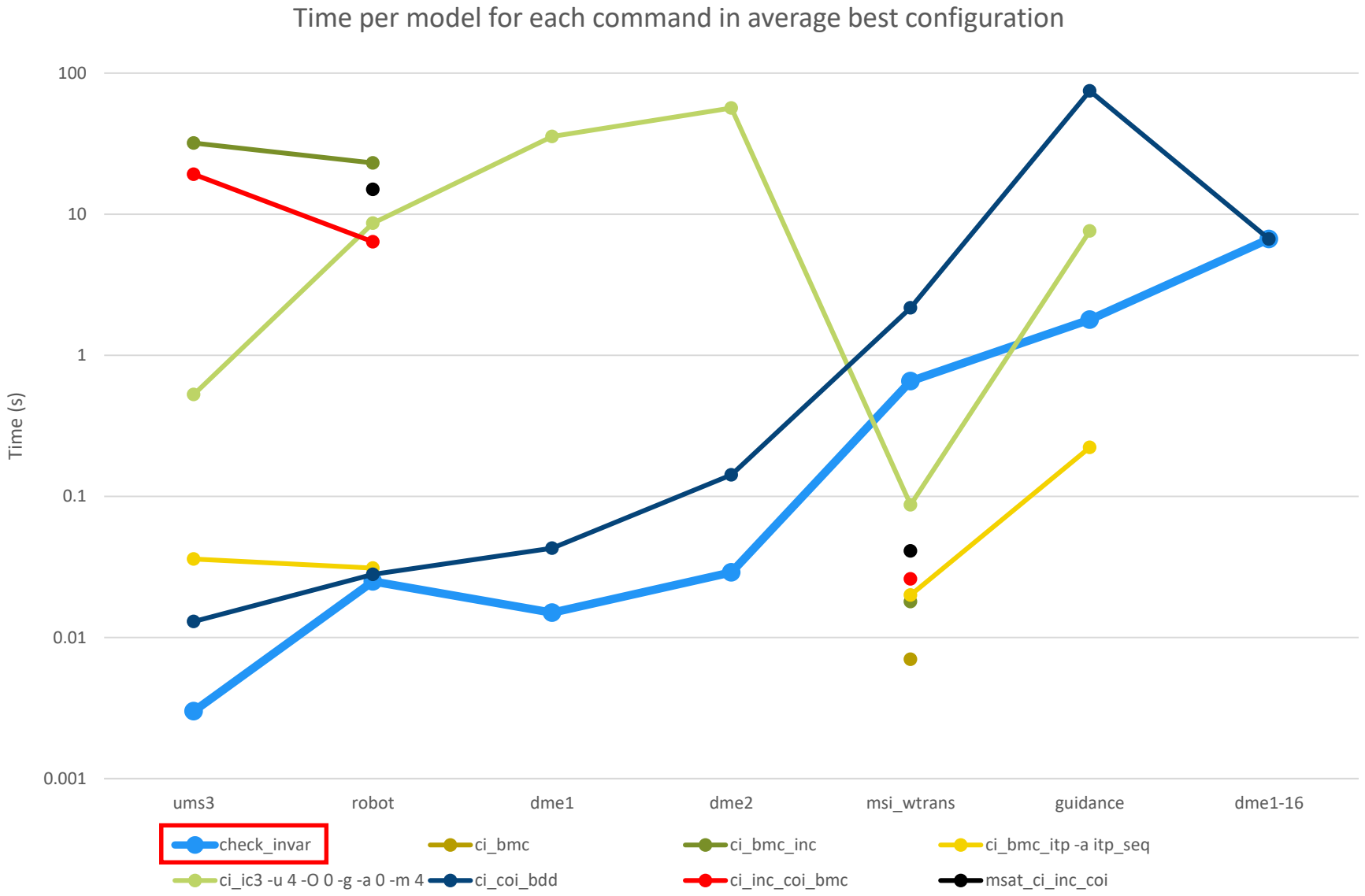
Benchmarking Invariant Checking

- 8 commands, ~100 configurations
  - BDD: `check_invar`, `ci_inc_coi_bdd`
  - BMC: `ci_bmc`, `ci_bmc_inc`, `ci_bmc_itp`, `ci_inc_coi_bmc`
  - Other: `ci_ic3`, `msat_ci_inc_coi`
- 7 Models
  - Models from NuSMV [CCG+02] and nuXmv [CCD+14] distribution and academic work (mostly [FTBW+21])
- *Why so few models?*
  - Few considered models use invariants
  - Some models are “too easy” to verify

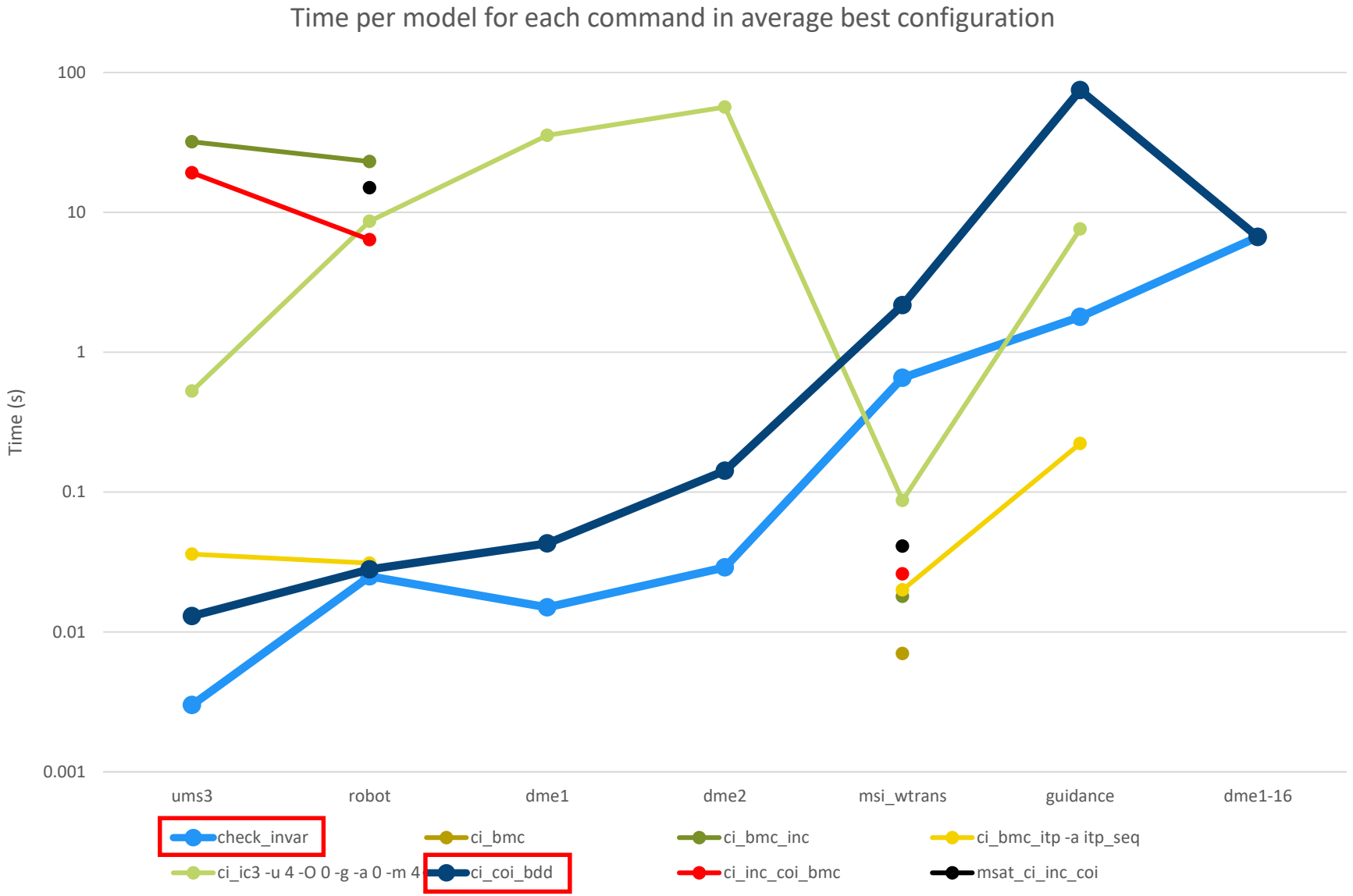
Benchmark Results



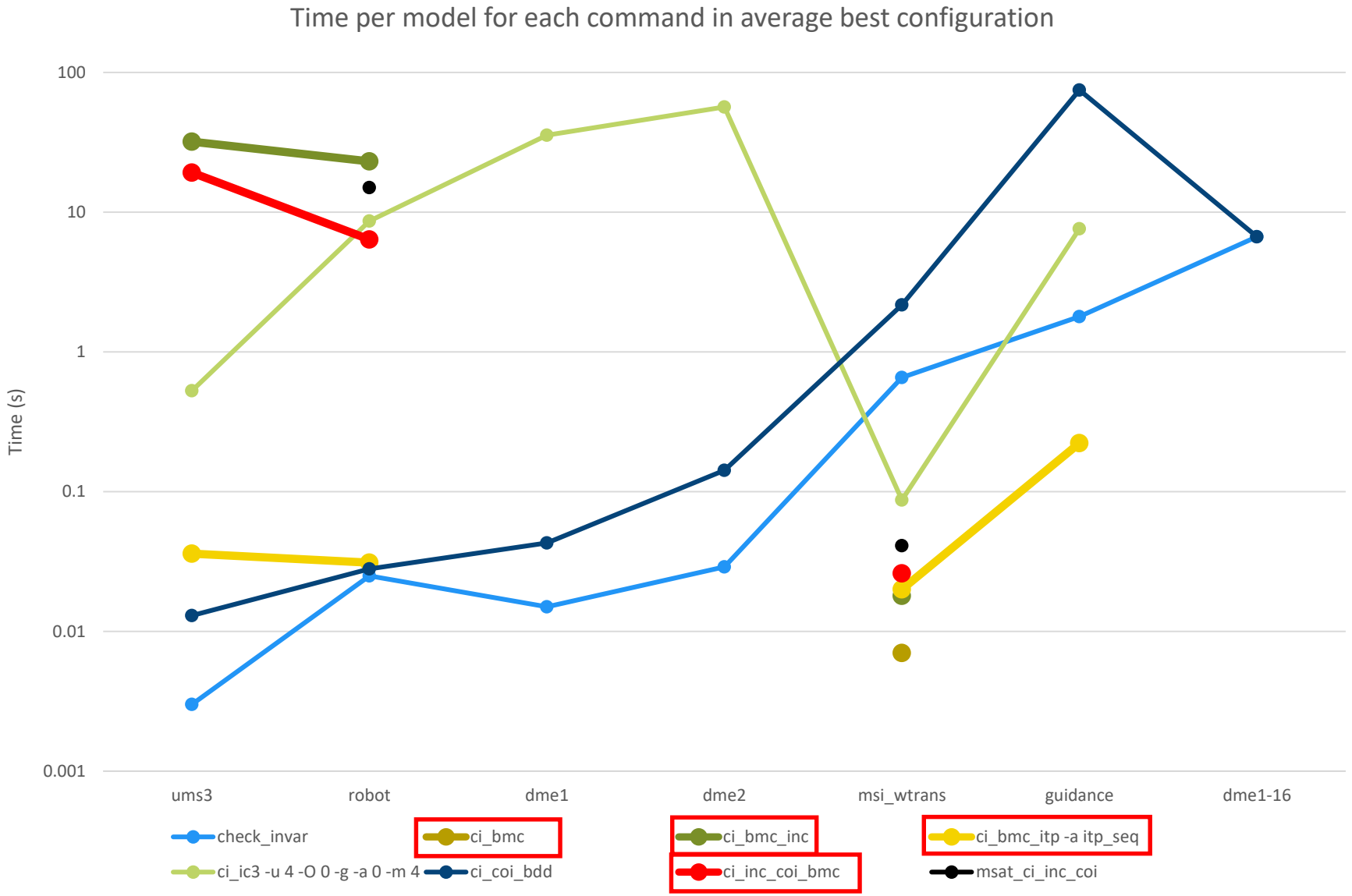
Benchmark Results



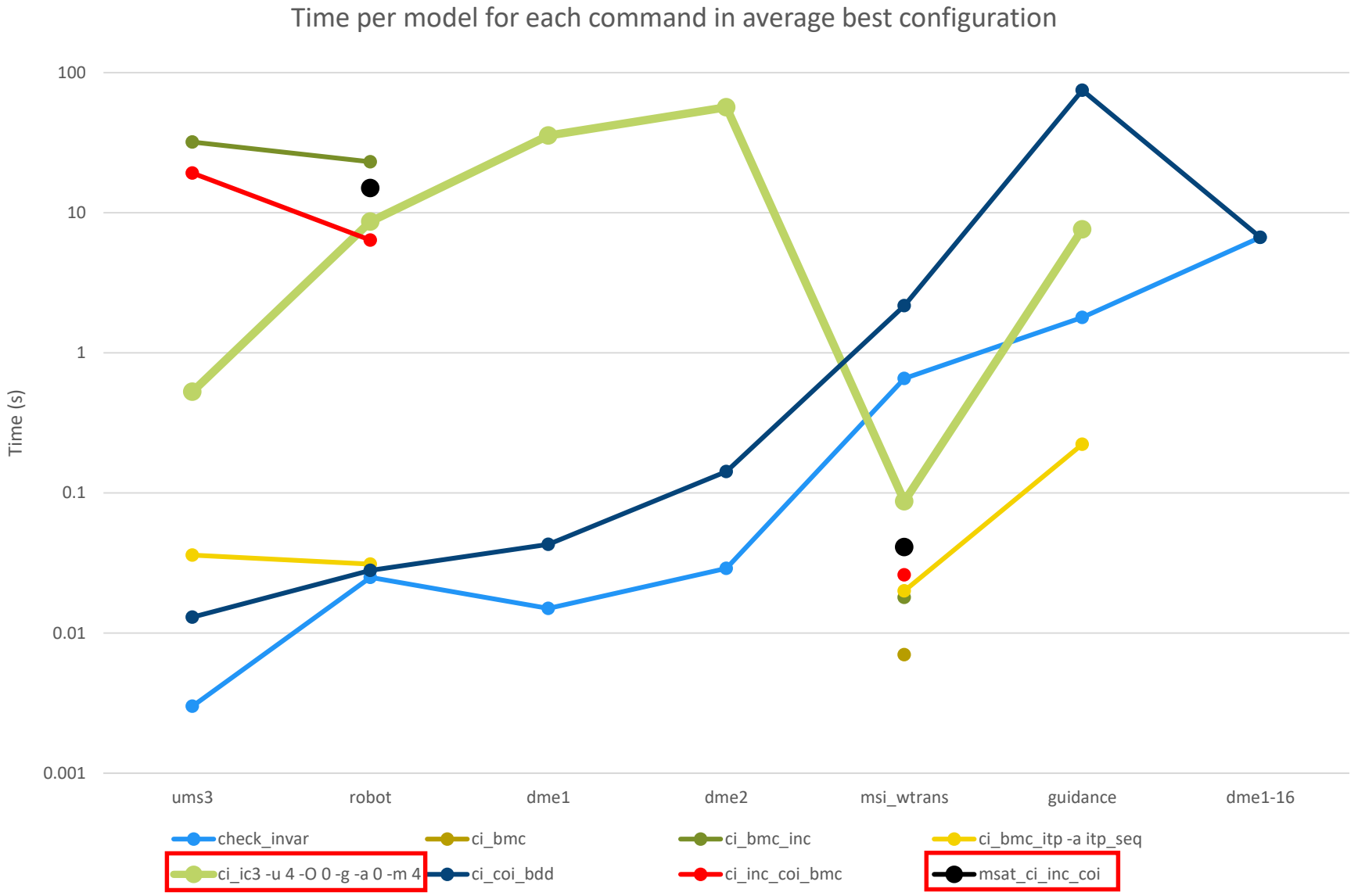
Benchmark Results



Benchmark Results



Benchmark Results



Conclusions and Future Work

- The “default” performs extremely well!
- Too few models to extrapolate to general case
- If you have a model you want me to include, please contact me at bernhard.luedkte@uni-bamberg.de

Future Work:

- Investigate the *automatic* engine of Storm
- Investigate possible correlations with model characteristics
- Integrate models from other sources, e.g., HWMCC [BvDH+17]

Sources

- [BvDH+17] A. Biere, T. van Dijk, and K. Heljanko. Hardware model checking competition 2017. In 2017 Formal Methods in Computer Aided Design (FMCAD), pp. 9-9. IEEE, 2017.
- [CCD+14] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuxmv symbolic model checker. In Computer Aided Verification (CAV 2014), volume 8559 of Lecture Notes in Computer Science, pages 334–342. Springer, 2014.
- [CCG+02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings 14, pp. 359-364. Springer, 2002.
- [FTBW+21] A. Fellner, M. Tabaei Befrouei, and G. Weissenbacher. Mutation testing with hyperproperties. Software and Systems Modeling 20, no. 2 (2021): 405-427.
- [HJK+21] C. Hensel, S. Junges, J.-P. Katoen, T. Quatmann, and M. Volk. The probabilistic model checker Storm. International Journal on Software Tools for Technology Transfer (2021): 1-22.
- [vHDM+14] R. von Hanxleden, B. Duderstadt, C. Motika, S. Smyth, M. Mendler, J. Aguado, S. Mercer, and O. O'Brien. SCCharts: Sequentially Constructive Statecharts for safety-critical applications. In Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 372-383. 2014.