Automating Trace Validation with PGo

Finn Hackett and Ivan Beschastnikh
University of British Columbia
{fhackett,bestchai}@cs.ubc.ca

Abstract

TLA+ allows us to model concurrent and distributed systems in simplified, abstract ways. Then, we can analyze these models [3, 9–11] in order to predict and fix design issues that would be impractical to find in implementation code. While useful at the design level, a key problem with this approach is that there is no inherent guarantee that a system implementation matches its abstract model. We can use a tool like PGo [7] to generate the implementation from an MPCal model, but compiled models can also have bugs [6]. To cross-check implementations against models, we can use trace validation [2, 4, 8]: collect structured logs of implementation behavior, adapt them into a format that can be compared with the original TLA+ model, and match the two via model checking.

Trace validation has shown promising results and gained industry adoption, but the technique can be inconvenient to put into practice. Given an existing TLA+ specification and a system implementation, significant boilerplate is necessary to connect the two. In this talk we will describe our work on reducing this effort by using code generation with the PGo compiler. Some of our methodology is suitable for all TLA+ specifications, and some leverages specific features of the MPCal modeling language.

This talk gives background on trace validation and specification compilation using PGo, then it introduces the family of automations we are developing in order to make trace validation more accessible and convenient.

Talk Outline

Total talk time: 35 minutes speaking, 5 minutes for questions.

[6 min] Intro

- Speaker intro
- Motivation for system analysis
- Difficulty due to abstraction: can't tell if implementation matches design.
- Known techniques: compilers (PGo[1], Erla+ [5]), model-driven development, trace validation.

[1 min] Structure overview: explain trace validation and its current methodology, contextualize how PGo works, illustrate current and future ways we can simplify the process.

[5 min] Overview of Trace Validation

[1] https://github.com/distCompiler/pgo

- Validation as a refinement relationship. TLC can be used to compare two models, check if there is a consistent relationship between one model's behaviors and the others'
- We can extract an explorable TLA+ model from implementation tracing data, illustrated.
- With the right setup, this is deployed right now in 2 industry projects that we know of (etcd[2], CCF[3]).

[4 min] Introducing PGo, MPCal, and having a compiler that can read TLA+

- MPCal as a superset of PlusCal
- Parts of PGo relevant to trace validation
- You can iterate and pattern match over spec components. Super powered TLA+ metaprogramming!

[3 min] Automatic tracing of MPCal specifications

- Introspecting the life cycle of Go code compiled from MPCal
- Auto-inserting vector clock instrumentation

[3 min] Generating trace validation boilerplate with MPCal

- Inferring state variables and actions from MPCal
- Inferring behavior from the logged statements (at scale)

[5 min] Demo: tracing a test system, generating TLA+, running TLC

[3 min] Generating trace validation boilerplate for plain TLA+

- Reading metadata from the TLA+
- What if the log is incomplete?

[4 min] Intended evaluation, where we are so far

- Validating PGo's prototype systems (and how to get good coverage [1])
- Integration with existing tracing setups
- Ability to find types of planted bug
- An anecdotal "real" bug: over-constrained model of TCP connections.

[1 min] Conclusion: wrap-up, summary, and aspirations going forward.

[2] https://github.com/etcd-io/raft/tree/main/tla
[3] https://github.com/microsoft/CCF/tree/main/tla

Bibliography

[1] Sebastian Burckhardt, Pravesh Kothari, Madanlal Musuvathi, and Santosh Nagarakatte. 2010. A randomized scheduler with probabilistic guarantees of finding bugs. *ACM SIGARCH Comput. Arch. N.* 38, 1 (2010), 167–178. https://doi.org/10.1145/1735970.1736040

[2] Amaury Chamayou, Benjamin Loillier, Eddy Ashton, Eric Dai, Heidi Howard, Horatiu Cirstea, Jian Zhou, Joshua Zhang, Markus Kuppe, Stephan Merz, and Vincent Li. 2024. Validating System Executions with the TLA$^+$ Tools. In *TLA+Conf*.

[3] Kaustuv Chaudhuri, Damien Doligez, Leslie Lamport, and Stephan Merz. 2010. Verifying Safety Properties with the TLA+ Proof System. In (*Lecture Notes in Computer Science*), 2010. 142–148. https://doi.org/10.1007/978-3-642-14203-1_12

[4] Horatiu Cirstea, Markus A Kuppe, Benjamin Loillier, and Stephan Merz. 2024. Validating Traces of Distributed Programs Against TLA+ Specifications. In *SEFM 2024*, 2024. 126–143. https://doi.org/10.1007/978-3-031-77382-2_8

[5] Kiko Fernandez-Reyes, Adriana Laura Voinea, Marian Hristov, and Annette Bieniusa. 2024. Erla+: Translating TLA+ Models into Executable Actor-Based Implementations. In (*Proceedings of the 23rd ACM SIGPLAN International Workshop on Erlang*), 2024. 13–23. https://doi.org/10.1145/3677995.3678190

[6] Pedro Fonseca, Kaiyuan Zhang, Xi Wang, and Arvind Krishnamurthy. 2017. An Empirical Study on the Correctness of Formally Verified Distributed Systems. *Proc Twelfth European Conf Comput Syst* (2017), 328–343. https://doi.org/10.1145/3064176.3064183

[7] Finn Hackett, Shayan Hosseini, Renato Costa, Matthew Do, and Ivan Beschastnikh. 2023. Compiling Distributed System Models with PGo. *Proc. 28th ACM Int. Conf. Arch. Support Program. Lang. Oper. Syst., Vol. 2* (2023), 159–175. https://doi.org/10.1145/3575693.3575695

[8] Heidi Howard, Markus A Kuppe, Edward Ashton, Amaury Chamayou, and Natacha Crooks. 2024. Smart Casual Verification of CCF's Distributed Consensus and Consistency Protocols. *arXiv* (2024). https://doi.org/10.48550/arxiv.2406.17455

[9] Igor Konnov, Jure Kukovec, and Thanh-Hai Tran. 2019. TLA+ model checking made symbolic. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 1–30. https://doi.org/10.1145/3360549

[10] Markus Alexander Kuppe, Leslie Lamport, and Daniel Ricketts. 2019. The TLA+ Toolbox. *Electron. Proc. Theor. Comput. Sci.* 310, (2019), 50–62. https://doi.org/10.4204/eptcs.310.6

[11] Yuan Yu, Panagiotis Manolios, and Leslie Lamport. 1999. Model Checking TLA+ Specifications. In (Lecture Notes in Computer Science), 1999. 54–66. https://doi.org/10.1007/3-540-48153-2_6