Multi-Grained Specifications for Distributed System Model Checking and Verification (EuroSys 2025)

William Schultz

April 18, 2025

Multi-Grained Specifications for Distributed System Model Checking and Verification [OST$^+$25] describes an approach for specifying and verifying ZooKeeper. They write abstract TLA+ models of the ZooKeeper protocol, while also checking conformance between the model and code. To tackle state space explosion of models, they decompose the specs into modules which are then written at different "granularities", and composed together for checking properties.

Multi-Grained Specifications

The basic idea of multi-grained specifications is based around the ability to decompose an overall system specification into separate logical modulues, which can independently be specified at different "granularities" (e.g. levels of atomicity), and then composed together in a semantics preserving way. This enables efficient model checking by reliance on *interaction preservation* idea, which was introduced in prior work [GCZ$^+$22]. The idea is that, for a given module, its behavior can be defined in terms of its *interactions* with other modules. Its "internal" behavior is irrelevant to the operation of the overall, composed system, so can be abstracted away as long as high level correctness properties don't depend on any of this internal state.

In TLA+, they define global state of a system with *variables* and states are updated with *actions.* They define *dependency variables* of an action as the variables in the enabling condition of the action. A *module* is defined as a set of actions, and the dependency variables of a module are then defined as the union of dependency variables for all actions in the module.

They claim that coarsening of a module preserves interaction if

(1) All dependency variables of the target module, as well as all interaction variables, remain unchanged after the coarsening.

(2) All the updates of the dependency variables and interaction variables remain unchanged after the coarsening.

So, overall, if you can describe a specification S as composed of n modules as

$$S = \bigcup_{1 \leq i \leq n} M_i$$

and $\widetilde{M_i}$ represents the coarsened version of module M_i, then if the above hold, then S_i is denoted as the specification by coarsening S with every other module except M_i i.e.

$$S_i = \left(\bigcup_{j \neq i} \widetilde{M_j} \right) \cup M_i$$

They claim the main theorem, the **Interaction Preservation Theorem**, which says essentially that:

Theorem 1. *Given* $S = \bigcup_{1 \leq i \leq n} M_i$ *and* $S_i = \left(\bigcup_{j \neq i} \widetilde{M_j} \right) \cup M_i$*, we have* $T_S \overset{M_i}{\sim} T_{S_i}$.

where $T_S \overset{M_i}{\sim} T_{S_i}$ is defined as trace T_S and T_{S_i} being equal when projected to M_i

Conformance Checking and Tooling

They built a tool called REMIX, which includes a conformance checking framework for checking ZooKeeper implementation against their specifications. The conformance checker explores random traces of the model-level state space under some budet, and then runs these model traces checking for discrepancies between implementation and model. As they describe it, apparently developers still need to manually provide a mapping from each model-level action to the events in the code that represent the beginning and end of the corresponding code-level action. They use some Java specific instrumentation tools (AspectJ) to achieve all of this, but it seems the manual burden of defining these conformance mappings is still high/nontrivial

Spec	Election	Discovery	Log Replication	
			Synchronization	Broadcast
SysSpec	Baseline	Baseline	Baseline	Baseline
mSpec-1	Coarsened		Baseline	Baseline
mSpec-2	Coarsened		Fine-grained (atom.)	Baseline
mSpec-3	Coarsened		Fine-grained (atom.+ concur.)	Fine-grained (concur.)
mSpec-4	Baseline	Baseline	Fine-grained (atom.+ concur.)	Fine-grained (concur.)

Table 1. Mixed-grained specifications for verifying log replication, composed from multi-grained specifications. "SysSpec" refers to the system specification that passes conformance checking (used as the baseline).

Evaluation Results

Basically, they find a bunch of bugs. They also note the significant efficiency gains from checking multi-grained specifications. They note in some cases over 1000x speedups in exploration costs, as illustrated in Table 5.

Spec	Time	Depth	# States	# Violation	# Vio. Inv.
Baseline	>24h	26	2,271,335,268	0	None
mSpec-1	12m20s	56	17,586,953	0	None
mSpec-2	15m55s	62	24,211,064	1,404	I-8
mSpec-3	5m10s	21	1,727,234	>10,000	I-10, I-11, I-12
mSpec-4	>24h	26	2,478,453,900	35	I-10, I-11, I-12

(b) Running to completion

Table 5. Verification efficiency of specifications with different granularities. The configuration is three servers, two transactions, two crashes, and two partitions. "Depth" refers to the number of state transitions; "States" refers to the *distinct* states explored (and reported) by TLC.

Thoughts/Questions

- Are their interaction-preserving coarsenings guaranteed to be formally correct in any way, or are they just eyeballing it? For bug-finding, eyeballing is arguably ok, since you're not as worried about completeness, and if you find a candidate bug, you can check it manually for veracity. But for completeness, I feel sketchy about their approach being formally correct.

- Conformance checking still seems laborious, as with many other papers on the topic. Not really sure if there's much to take away from that aspect i.e. it seems there was a lot of manual effort involved and is not clear it would be be easy to do in general for many systems. Multi-grained specification aspect seems the main value here.

References

[GCZ+22] Xiaosong Gu, Wei Cao, Yicong Zhu, Xuan Song, Yu Huang, and Xiaoxing Ma. Compositional model checking of consensus protocols specified in tla+ via interaction-preserving abstraction, 2022.

[OST+25] Lingzhi Ouyang, Xudong Sun, Ruize Tang, Yu Huang, Madhav Jivrajani, Xiaoxing Ma, and Tianyin Xu. Multi-grained specifications for distributed system model checking and verification. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 379–395, New York, NY, USA, 2025. Association for Computing Machinery.