# CAV 2022 Protocol Conversion

Ian Dardik

April 8, 2022

# 1 Converting MLDR From TLA+ To Ivy

Converting the *MongoLoglessDynamicRaft* protocol from TLA+ to Ivy was a nontrivial exercise. In this appendix we describe the conversion process as well as the challenges we faced.

## 1.1 Protocol Conversion

The *MongoLoglessDynamicRaft* protocol that is introduced in [1] is encoded as a TLA+ specification. The specification encodes the STS using TLA+ constructs; the sort *Server* is encoded as a CONSTANT, each state variable is encoded as a VARIABLE, and the initial constraint and transition relation are each encoded as a TLA+ *operator*. In addition, the TLA+ spec also includes several auxiliary operators that make the spec more readable. We can convert by mapping the sort *Server* to Ivy's *type* construct, the state variables to Ivy *functions*, the initial constraint and transition relation to Ivy *actions*, and the auxiliary operators to relations. We describe each mapping in a separate section below, except for the sort *Server* because it maps directly to an Ivy *type*.

### 1.1.1 State Variable Mapping

The state variables are mapped to functions based on the *TypeOK* property over the MLDR state variables in TLA+:

| $TypeOK ==$ | Ivy translation: |
|---|---|
| $\land\ currentTerm \in [Server \to \mathbb{N}]$ | function $currentTerm(S : server) : nat$ |
| $\land\ state \in [Server \to \{Secondary, Primary\}]$ | function $state(S : server) : state\_type$ |
| $\land\ config \in [Server \to \text{SUBSET } Server]$ | function $config(S : server) : conf$ |
| $\land\ configVersion \in [Server \to \mathbb{N}]$ | function $config\_version(S : server) : nat$ |
| $\land\ configTerm \in [Server \to \mathbb{N}]$ | function $config\_term(S : server) : nat$ |

This translation implies that we also need three more types: *nat*, *state_type*, and *conf*, which we encode as Ivy types.

### 1.1.2 Auxiliary Operator Mapping

blah blah

## 1.2 Inductive Invariant Conversion

## 1.3 Challenges

In this section we cover the challenges we faced during conversion.

### 1.3.1 EPR

Fitting MLDR into EPR was especially tough because of the quorum overlap property that must be encoded into MLDR. They key here is to separate the type for configs and quorums, even though their types are both subsets of *Server*.

### 1.3.2 Testing Our Ivy Spec

Once we completed the conversion process, it was not clear how we might test our spec since our inductive invariant puts our spec outside of EPR. One option would be to develop a new inductive invariant within EPR using Ivy, but this is a costly process and it is not clear whether we are guaranteed to find such an inductive invariant. Instead we gained intuition by proving simple properties, and by confirming our inductive invariant "works" in IC3PO for sufficiently large sort sizes.

# References

[1] William Schultz, Siyuan Zhou, Ian Dardik, and Stavros Tripakis. Design and Analysis of a Logless Dynamic Reconfiguration Protocol. In Quentin Bramas, Vincent Gramoli, and Alessia Milani, editors, *25th International Conference on Principles of Distributed Systems (OPODIS 2021)*, volume 217 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.