Distributed Systems in Lean 4 (DSL)

Spring 2025

*Last Updated: 2025-05-22*

Motivations

In my CSE 232 class there are a lot of protocols and components. We argue about their properties using informal proofs but it’d be nice to have a Formally Verified proof that if I deliver using Uniform Reliable Broadcast, every other correct process will also deliver.

The idea is to develop some kind of theory that lets me prove safety and liveness properties about distributed components. You’d think that most of this can be done with some kind of linear temporal logic:

S always holds ≈ ◻S

Eventually S holds ≈ ◇◻S

Unfortunately, probabilistic results prove somewhat problematic:

Sending infinitely often on a fair loss link will deliver infinitely often.

To prove probabilistic safety and liveness properties, I’ll need a different interpretation of ◇, and for that, I’ll need to develop some practical application of Linear Temporal Logic that includes limits.

Scope and Goals

Stage 0

A theory for distributed components is written and is used to prove the following properties:

|  |  |
| --- | --- |
| * Fair Loss Link   + FL1. Fair Loss   + FL2. Finite Duplication   + FL3. No Creation * Stubborn Link   + SL1. Stubborn Delivery   + SL2. No Creation * Perfect Link   + PL1. Validity   + PL2. No Duplication   + PL3. No Creation * Perfect Failure Detector (P)   + P1. Strong Completeness   + P2. Strong Accuracy | * Best Effort Broadcast   + BEB1.   + BEB2.   + BEB3. * Reliable Broadcast   + RB1 = BEB1   + RB2 = BEB2   + RB3 = BEB3   + RB4. Agreement * Uniform Reliable Broadcast   + URB1 = BEB1   + URB2 = BEB2   + URB3 = BEB3   + URB4. Agreement |

This theory does not necessarily need to be the best or even good. It just needs to do its job. In some sense, I will be reinventing the wheel to get a feel for what making a wheel even feels like.

**Non-Goals:**

* Partially Synchronous Systems
* Asynchronous Systems
* Domain Specific Language
* Meta-theory proofs
* Elegance

Stage 1

Anything in here probably isn’t getting done at school since time is so limited:

* Blog posts and other meta-content about this project
* Reading through TLC and adopting its syntax instead of my amalgamation
* Writing a more condensed guide to distributed systems.
* Possibly linking CSE 232 course slides to their associated proofs in Lean
* A Lean-embedded DSL for distributed components

Research Materials

* [Prof. Mohsen Lesani’s CSE 232 Lecture Slides](https://mohsenlesani.github.io/slugcse232/)
* Introduction to Reliable and Secure Distributed Programming, 2nd ed.
* [@kmil/cse290q-25sp](https://github.com/kmill/cse290q-25sp) Spring 2025 CSE 290Q repository on GitHub.
* [Mathematics in Lean](https://leanprover-community.github.io/mathematics_in_lean/)
* [Glimpse of Lean](https://github.com/PatrickMassot/GlimpseOfLean)

Prior Art:

* <https://github.com/kenmcmil/ivy>
* Veil (CAV 25)
* Basically half the papers Lesani has done
  + Lesani TLC (ICFP'20)
  + Lesani Chapar (POPL 16)

Work Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Week | Starting | Ending | Tasks |
| 8 | 2025-05-19 | 2025-05-25 | * This document * All exercises in the CSE 290Q repo * Mathematics in Lean Ch 1, 2, 3, 4, 5, 6 |
| 9 | 2025-05-26 | 2025-06-01 | * Mathematics in Lean Ch 7, 8(?), 11 * Begin development of theory * Formalize:   + Fair Loss Link   + Stubborn Link   + Perfect Link   + Best Effort Broadcast   + Reliable Broadcast |
| 10 | 2025-06-02 | 2025-06-05 | TODO |
| Finals | 2025-06-10 | 2025-06-15 | * Deliverables Met * Project Report Document |

Accountability:

I will be committing regularly to several GitHub repositories:

* [@falfiya/dsl](https://github.com/falfiya/dsl), which contains this document and all related project files
* [@falfiya/lean-with-math](https://github.com/falfiya/lean-with-math): My fork of Mathematics in Lean
* @falfiya/adventures-in-lean [missing link]: My fork(?) of the [Spring 2025 CSE 290Q repository](https://github.com/kmill/cse290q-25sp).

Progress

Week 8

In Week 8, I hope to gain more familiarity with the syntax of Lean by finishing all exercises in the CSE 290Q repository and by starting Mathematics in Lean (MIL). Not yet sure about how Glimpse of Lean is going to fit into my schedule, but I know that I need to get through Chapter 11 of MIL to finally see how to do limits. Apparently, they have something to do with a “filter”. See [Filter.Tendsto](https://leanprover-community.github.io/mathlib4_docs/Mathlib/Order/Filter/Defs.html#Filter.Tendsto).