Meeting Note - Jan 18

Where we are: We are developing an idea based on [1], where authors offer a computationally tractable violation detection algorithm following their notion of serializability for EC (eventually consistent) applications. Now, we want to not just *detect* such violations, but also prescribe (incremental) applicaiton repairs by *inspecting* and *classifying* such violations.

What's new: I first tried to tackle the violation inspection and classification problem by collecting a large number of examples and manually inspecting them. However, I relized that I cannot make further progress until I clearly define what the system model is. So, I did that (following [2], but incorporating my own ideas on what should be included in the model) and I will be talking about this model.

References

1. Lucas Brutschy, Dimitar Dimitrov, Peter Müller, and Martin Vechev. 2017. Serializability for eventual consistency: criterion, analysis, and applications. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017). ACM, New York, NY, USA, 458-472. DOI: https://doi.org/10.1145/3009837.3009895
2. Sebastian Burckhardt, Alexey Gotsman, Hongseok Yang, and Marek Zawirski. 2014. Replicated data types: specification, verification, optimality. In Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14). ACM, New York, NY, USA, 271-284. DOI: https://doi.org/10.1145/2535838.2535848