Verifying replicated data structures: causal consistency

Internship proposal (2016 - 2017)

Constantin Enea (`www.irif.fr/~cenea`)

Institut de Recherche en Informatique Fondamentale (IRIF), Paris

cenea@irif.fr

Research context. One of the most complex aspects in modern software is related to the issue of efficient handling of data. Indeed, the continuous quest of performances leads to the design of sophisticated data structures and algorithms for efficient storage, organization, and manipulation of data. This leads also to the development of complex software and hardware systems, with more and more parallelization and distribution. Therefore, data structures running over parallel and distributed infrastructures become ubiquitous in real life. For instance, the data structures used in cloud computing infrastructures are essential to a wide range of services, from social networks and games to collaborative spaces and online shops. Key-value stores like Amazon Simple Storage Service, Google App Engine Datastore, and Yahoo PNUTS are largely used in the services provided by the companies developing them. Furthermore, researchers are also proposing new technologies, as well as algorithms and implementations to improve distributed data structures. For instance Shapiro et al. have recently introduced the framework of Conflict-Free Replicated Data Types (CRDTs) where it is possible to define various data structures like counters, sets, maps, graphs, and lists, in a scalable way. This framework represents a nice success story since it is now developed in an industrial context by Basho technologies.

The implementations of these data structures, called *Replicated Data Structures* (RDSs), are very complex and difficult to get right. The main reason being the weak guarantees provided by the underlying infrastructure: in order to cope with possible network partitions and faults, data structures are geo-replicated in several data centers across the world, which can be simultaneously accessed by clients to update or query the data structure. Geo-replication renders the maintenance of a data structure extremely complex: for instance, data centers must communicate to know all the updates submitted in the system and also to agree on the order in which to execute them [7].

Ideally, one would desire that RDSs tolerate network partitions and provide immediate availability (each request is followed by an immediate response) while also ensuring strong consistency. This last requirement roughly guarantees that the outcome of a set of concurrent requests to the data structure is the same as what could be obtained by executing these requests atomically in a certain order. Unfortunately, the famous CAP theorem (*Consistency, Availability, Partition tolerance*)shows that such a data structure cannot be conceived. Consequently, modern geo-replicated data structures provide weaker forms of consistency, e.g. eventual consistency and causal consistency [3, 5].

A popular consistency criterion for RDSs is *causal consistency* [5]. Roughly, an RDS satisfies this criterion if operations that potentially are causally related are seen by every site in the network in the same order. Concurrent operations (i.e. ones that are not causally related) may be seen in different orders by different sites.

Objectives. This internship is intended to focus on developing automated formal verification techniques for causal consistency. During the internship, the student should become familiar with the state of the art on the verification of correctness criteria for distributed/concurrent data structures. Possible research questions that could be addressed during this internship are:

- Decidability/complexity results for the verification of causal consistency for different classes of RDSs. In general, causal consistency is a criterion stronger than eventual consistency but still weaker than strong consistency. While the decidability/complexity status of these other criteria is known [1, 3], the design of algorithmic verification techniques for causal consistency is (almost) unexplored in the literature.

- Defining automated bug-detection techniques in the spirit of bounded model checking [4]. The goal is to define algorithmic techniques for analyzing a representative subset of the RDS's behaviors. This subset of behaviours should be defined by a suitable bounding parameter in the spirit of context-bounding, which is used in the verification of shared-memory concurrent programs [6], or interval length bounding, which is used in the verification of concurrent data structures running over shared-memory architectures [2].

Skills. During the internship, we will rely on knowledge of concepts in the area of automated formal verification, algorithmic reasoning, automata theory, concurrent programming.

Propects. This is a broad topic that could be the starting point of a Ph.D.

References

[1] Ahmed Bouajjani, Constantin Enea, Michael Emmi, and Jad Hamza. Verifying concurrent programs against sequential specifications. In *Proc. of 22nd European Symposium on Programming (ESOP'13)*, Lecture Notes in Computer Science. Springer, 2013.

[2] Ahmed Bouajjani, Constantin Enea, Michael Emmi, and Jad Hamza. Tractable refinement checking for concurrent objects. In *Proc. of the 42nd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'15)*. ACM, 2015.

[3] Ahmed Bouajjani, Constantin Enea, and Jad Hamza. Verifying eventual consistency of optimistic replication systems. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14*, pages 285–296, 2014.

[4] Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.

[5] Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Don't settle for eventual: Scalable causal consistency for wide-area storage with cops. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 401–416, New York, NY, USA, 2011. ACM.

[6] Shaz Qadeer and Jakob Rehof. Context-bounded model checking of concurrent software. In *Proc. of the 11th Intern. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05*, pages 93–107, 2005.

[7] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.