Serializability in Programmable Networking Services

ANONYMOUS AUTHOR(S)

Add abstract here

1 Introduction

continue..

2 Problem Definition

The problem..

3 Example: NetKAT + Global Variables + Yields

NetKAT...

4 Formulations

The..

5 Proofs

The theorems..

6 Implementation

The implementation..

7 Related Work

Serializability first introduced by Eswaran et al. [17]. It is the first to put forth serializability as a correctness condition for concurrent transaction execution. The paper also covers conflict serializability. Papadimitriou [35] proved that even deciding the history of a single interleaving is serializable is NP-hard.

conflict serializability is enforced during runtime in with with pessimistic locking approaches (e.g, 2-Phase locking [6]), or with optimistic locking approaches, e.g., Optimistic Concurrency Control (OCC) [10, 25]

[TODO] start

Alur et al. [2]... cover conflict serializability (not "regular" serializability, which is what we do). Furthermore, a main caveat is that they focus on a bounded number of transactions

continued by [7]. In the followup paper (Boujjani et al.) - they also cover conflict serializability, but find a stronger result than Alur, based on unbounded transactions. They find an interesting result that although you can have an infinite conflict graph (when having infinite transactions), then you can still decide conflict serializability in the unbounded case by finding a cycle in the graph when it's non (conflict) serializable, and the cycle length surprisingly does not depend on the number of transactions, which is pretty cool. Another point is that they define a VASS (=Petri Net) that represents the interleaving, and their definition for it is similar to our PN. They then modify it to include a conflict cycle. The most relevant part to us in this paper is that it's on an unbounded number of transactions and also, that they represent Int(S) with a VASS that is similar to us. Still, it's not our notion of serializability (and indeed, they have EXPTIME complexity, while we're probably Ackermann complete?).

Another line of work leverages the highly expressive *Logic of Temporal Actions* (TLA) [29]. These work encode serializability in TLA+ [21, 38],...model checkers (such as TLC and Apalache) [23, 40]. TLA+ can indeed encode an infinite number of transactions. For example, here is the TLA+ spec for encoding serializability. However, for doing model checking on a TLA spec (with the TLC model checker) — the model checker takes a .cfg file as additional input, in in the .cfg you explicitly specify all of the sets in the model, and these have to be finite. You can see this here where the model checking file needs to encode in advance the number of transactions (see attached figure)

[TODO] end

[TODO] go over the paper and its citing papers Me: 1992 paper today, they seem to model a concurrent execution with petri nets but they don't ask if all executions are serializable which is our subject matter

Some work relaxes the (strong) consistency notion of serializability and allows weaker consistency notions. For example, Rastogi et al. [36] introduce *predicate-wise serializability* (PDSR) — a relaxation of serializability in which

transactions might not be atomic, but are still required to maintain some desired database consistency predicate Furthermore, other relaxations focus on weaker consistency models. One such model is causal consistency, which was put forth by Lamport [28], en extended to shared memory systems as *causal memory* [1]. (include causal + consistency, designed in COPS [30]). The have been a plethora of works on model checking systems that adhere to causal consistency, and hence the complexity of such procedures [8, 26, 41]. We also note that some work combine various consistency notions. These include the recent work by Brutschy et al. [9], who put form a method to statically detect non-serializable executions on top of causally-consistent databases.

Our work also builds upon both theoretical literature, as well as practical results, pertaining to Petri Nets [14, 34, 37]. Firstly, our undecidability result is based on a classic result by Hack [18, 19], showing that, given two Petri Nets, it is undecidable to answer whether they have equivalent reachability sets. Hack based his result on the work of Rabin (which was never published). These undecidability results follow from a series of reductions, originating from Hilbert's 10th problem, i.e., deciding if a Diophantine polynomial has an integer root (a problem that was proved undecidable by Matijasévič [31]). Later, Jančar [22] proved this result by demonstrating that Petri Nets can simulate 2-counter Minsky Machines [33], which are universally computable and hence undecidable. Moreover, Jančar strengthened the original result and proved that reachability equivalence is undecidable even for Petri Nets with five unbounded places [22].

Our decision procedure itself is based on an algorithm for deciding whether a given marking is reachable, for a Petri Net. Mayr [32] was the first to put forth an algorithm for this problem given a (potentially, unbounded) Petri Net (note that for a bounded case this is straightforward, as you can enumerate all reachable markings.) Mayr's reachability algorithm was later improved and simplified by Kosaraju [24], and then again by Lambert [27]. Very recently, this problem was also proven to be Ackermann complete [11], implying that, although decidable, it is practically infeasible to solve on large nets. Furthermore, these theoretical algorithms have inspired various tools, such as K-Reach [13], DICER [39], MARCIE [20], and others. Specifically, our tool employs SMPT [4], a state-of-the-art Petri Net reachability tool, which employs an SMT-based approach [3, 5]. SMPT curtails the search space by reducing the reachability problem to a satisfiability query (that is subsequently dispatched to the Z3 solver [12]) and inferring invariants on the net's structure. We refer the reader to a survey by Esparza and Nielsen [15] (recently republished in [16]) for a comprehensive summary on additional decidability results pertaining to Petri Nets.

8 Discussion

8.1 Conclusion

To conclude..

8.2 Future Work

Next..

References

[1] M. Ahamad, G. Neiger, J. Burns, P. Kohli, and P. Hutto. Causal Memory: Definitions, Implementation, and Programming. Distributed Computing, 9(1):37–49, 1995.

[2] R. Alur, K. McMillan, and D. Peled. Model-Checking of Correctness Conditions for Concurrent Objects. In Proc. 11th Annual IEEE Symposium on Logic in Computer Science, pages 219–228, 1996.

[3] N. Amat, B. Berthomieu, and S. Dal Zilio. On the Combination of Polyhedral Abstraction and SMT-Based Model Checking for Petri Nets. In Proc. 42nd Int. Conf. on Applications and Theory of Petri Nets and Concurrency (PETRI NETS), pages 164–185, 2021.

[4] N. Amat and S. Dal Zilio. SMPT: A Testbed for Reachability Methods in Generalized Petri Nets. In Proc. 25th Int. Symposium on Formal Methods (FM), pages 445–453, 2023.

[5] N. Amat, S. Dal Zilio, and T. Hujsa. Property Directed Reachability for Generalized Petri Nets. In Proc. 28th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS),, pages 505–523, 2022.

[6] P. Bernstein, V. Hadzilacos, and N. Goodman. Concurrency Control and Recovery in Database Systems, volume 370. Addison-Wesley Reading, 1987.

[7] A. Bouajjani, M. Emmi, C. Enea, and J. Hamza. Verifying Concurrent Programs Against Sequential Specifications. In Proc. 22nd European Symposium on Programming (ESOP), pages 290–309, 2013.

[8] A. Bouajjani, C. Enea, R. Guerraoui, and J. Hamza. On Verifying Causal Consistency. In Proc. 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL), pages 626–638, 2017.

[9] L. Brutschy, D. Dimitrov, P. Müller, and M. Vechev. Static Serializability Analysis for Causal Consistency. In Proc. 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), pages 90–104, 2018.

[10] B. Burke and R. Monson-Haefel. Enterprise JavaBeans 3.0. O'Reilly Media, Inc., 2006. Optimistic Locking.

[11] W. Czerwiński and Ł. Orlikowski. Reachability in Vector Addition Systems is Ackermann-Complete. In Proc. 62nd Annual Symposium on Foundations of Computer Science (FOCS), pages 1229–1240. IEEE, 2021.

[12] L. De Moura and N. Bjørner. Z3: An Efficient SMT Solver. In Proc. 14th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pages 337–340, 2008.

[13] A. Dixon and R. Lazić. KReach: A Tool for Reachability in Petri Nets. In Proc. 26th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pages 405–412, 2020.

[14] J. Esparza. Decidability and Complexity of Petri Net Problems — An Introduction. In Advanced Course on Petri Nets, volume 1491 of Lecture Notes in Computer Science, pages 374–428. Springer, 1996.

[15] J. Esparza and M. Nielsen. Decidability Issues for Petri Nets — A Survey. Journal of Information Processing and Cybernetics, 30(3):143–160, 1994.

[16] J. Esparza and M. Nielsen. Decidability Issues for Petri Nets — A Survey, 2024. Technical Report. http://arxiv.org/abs/2411.01592.

[17] K. Eswaran, J. Gray, R. Lorie, and I. Traiger. The Notions of Consistency and Predicate Locks in a Database System. Communications of the ACM, 19(11):624–633, 1976.

[18] M. Hack. Decidability Questions for Petri Nets. PhD thesis, Massachusetts Institute of Technology, 1976.

[19] M. Hack. The Equality Problem for Vector Addition Systems is Undecidable. Theoretical Computer Science, 2(1):77–95, 1976.

[20] M. Heiner, C. Rohr, and M. Schwarick. MARCIE — Model Checking and Reachability Analysis Done Efficiently. In Proc. 34th Int. Conf. on Applications and Theory of Petri Nets and Concurrency (PETRI NETS), pages 389–399, 2013.

[21] L. Hochstein. Serializability and TLA+, Oct 2024.

[22] P. Jančar. Undecidability of Bisimilarity for Petri Nets and Some Related Problems. Theoretical Computer Science, 148(2):281–301, 1995.

[23] I. Konnov, J. Kukovec, and T.-H. Tran. TLA+ Model Checking Made Symbolic. In Proc. Int. Conf. on Object-Oriented Programming Systems, Languages, and Applications Proceedings of the ACM on Programming Languages (OOPSLA).

[24] S. Kosaraju. Decidability of Reachability in Vector Addition Systems. In Proc. 14th Annual ACM Symposium on Theory of Computing (STOC), pages 267–281, 1982.

[25] H.-T. Kung and J. T. Robinson. On Optimistic Methods for Concurrency Control. ACM Transactions on Database Systems (TODS), 6(2):213–226, 1981.

[26] O. Lahav and U. Boker. Decidable Verification Under a Causally Consistent Shared Memory. In Proc. 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), pages 211–226, 2020.

[27] J.-L. Lambert. A Structure to Decide Reachability in Petri Nets. Theoretical Computer Science, 99(1):79–104, 1992.

[28] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7):558–565, 1978.

[29] L. Lamport. The Temporal Logic of Actions. Number 3, pages 872–923, 1994.

[30] W. Lloyd, M. Freedman, M. Kaminsky, and D. Andersen. Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS. In Proc. 23rd ACM Symposium on Operating Systems Principles (SOSP), pages 401–416, 2011.

[31] J. Matijasévič. Enumerable Sets are Diophantine. Soviet Math. Dokl., 11(2):354–357, 1970.

[32] E. Mayr. An Algorithm for the General Petri Net Reachability Problem. In Proc. 13th Annual ACM Symposium on Theory of Computing (STOC), pages 238–246, 1981.

[33] M. Minsky. Computation: Finite and Infinite Machines. Prentice Hall.

[34] T. Murata. Petri Nets: Properties, Analysis and Applications. Proc. of the IEEE, 77(4):541–580, 1989.

[35] C. Papadimitriou. The Serializability of Concurrent Database Updates. Journal of the ACM (JACM), 26(4):631–653, 1979.

[36] R. Rastogi, S. Mehrotra, Y. Breitbart, H. Korth, and A. Silberschatz. On Correctness of Non-Serializable Executions. In Proc. 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Aystems (PODS), pages 97–108, 1993.

[37] W. Reisig. Petri Nets: An Introduction, volume 4. Springer Science & Business Media, 2012.

[38] T. Soethout, T. van der Storm, and J. J. Vinju. Automated Validation of State-Based Client-Centric Isolation with TLA^+. In Proc. 18th Int. Conf. Software Engineering and Formal Methods (SEFM), pages 43–57, 2020.

[39] D. Xiang, F. Zhao, and Y. Liu. DICER 2.0: A New Model Checker for Data-Flow Errors of Concurrent Software Systems. Mathematics, 9(9):966, 2021.

[40] Y. Yu, P. Manolios, and L. Lamport. Model Checking TLA+ Specifications. In Proc. Int. Conf. Correct Hardware Design and Verification Methods (CHARME), pages 54–66, 1999.

[41] R. Zennou, R. Biswas, A. Bouajjani, C. Enea, and M. Erradi. Checking Causal Consistency of Distributed Databases. In Proc. Int. Conf. on Networked Systems (NETYS), pages 35–51, 2019.