Brief Announcement: Automating and Mechanising Cutoff Proofs for Parameterized Verification of Distributed Protocols

Shreesha G. Bhat
Indian Institute of Technology Madras, India

Kartik Nagar
Indian Institute of Technology Madras, India

Abstract

We propose a framework to automate and mechanize simulation-based proofs of cutoffs for parameterized verification of distributed protocols. We propose a strategy to derive the simulation relation given the cutoff instance and encode the correctness of the simulation relation as a formula in first-order logic. We have successfully applied our approach on a number of distributed protocols.

2012 ACM Subject Classification Software and its engineering → Software verification

Keywords and phrases Formal Methods, Automated Verification, Distributed Protocols

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.48

Related Version *Full Version*: `https://github.com/shreesha00/DISC2021-Brief-Announcement-Automating-and-Mechanising-Cutoff-Proofs-for-Parameterized-Verification`

Funding This work was supported by the Young Research Fellow Program at IITM.

1 Introduction

The problem of parameterized verification [1] of distributed protocols asks whether a protocol satisfies its specification for all values of the parameter. Here, the parameter is typically the number of nodes involved in the protocol. Cutoff based approaches for parameterized verification rely on the following observation: If the protocol can break its specification for some value of the parameter, it is guaranteed to break the specification for a value $\leq k$, where k is also called the cutoff. Small cutoffs can then enable fully automated verification, for example by exhaustively model checking all instances of the protocol of size $\leq k$.

In the recent past, there has been a lot of interest in automated and mechanised verification of distributed protocols [11, 4, 12, 7, 9, 10]. Most of these approaches rely on constructing and proving some form of inductive invariant. While previous works have also attempted to use cut-off based approaches for verification [3, 6, 8, 1], they have mostly been limited to either a restricted class of protocols [6] or a restricted class of specifications [8]. In this work, we develop a methodology for mechanising simulation-based proofs of cutoffs by observing that the simulation relation can be generated using a direct correspondence between the nodes of an arbitrarily large system and the cutoff instance. We have successfully applied the proposed approach on a variety of distributed protocols.

2 Proposed Technique with Example

Model. We consider distributed protocols modelled in RML [11] where the system state is represented by a set of relations. The communication model between nodes is assumed to be asynchronous message passing. A set of actions are defined with guards and each step of the protocol involves non-deterministically firing one of these actions in an atomic fashion. The


© Shreesha G. Bhat and Kartik Nagar;
licensed under Creative Commons License CC-BY 4.0

specification for the protocol is given as a safety property. The parameterized verification problem then asks whether for all instances of the protocol, does the specification hold at every step. We are interested in a cutoff on the number of nodes. As an example protocol, we consider Leader Election in a Ring as given in [11]. The system consists of a finite number of nodes in a ring setting. Each node has a unique ID, and there is a total order on the IDs. There are two actions, (1) $\texttt{generate}(N, ID(N), NG(N))$: Node N sends a message $ID(N)$ to its neighbour $NG(N)$ and (2) $\texttt{handle\_message}(N, m, NG(N))$: Node N takes a pending message m in its pending queue (denoted by `pnd`) and forwards it to its neighbour only if $m > ID(N)$, else if $m = ID(N)$, N is elected as a leader (denoted by `leader`). The specification for the protocol is that there is at most one leader.

Inputs. We assume that the protocol designer provides the proposed framework with, (1) The protocol description, (2) A cutoff instance, (3) A mapping from nodes of any arbitrary system to nodes of the cutoff instance given by sim, (4) Two functions Ω and τ as described below. Let C be the cutoff instance and $\mathbb{M}_C$ the set of nodes in C. Consider an arbitrary instance L of the protocol where $\mathbb{M}_L$ is the set of nodes in L such that $|\mathbb{M}_L| > |\mathbb{M}_C|$. The mapping function sim has the following meaning, for each node $N_L \in \mathbb{M}_L$, N_L is simulated by $sim(N_L) \in \mathbb{M}_C$. The key intuition here is that a node $N_C \in \mathbb{M}_C$ effectively maintains the state components relevant to the violation of the safety property for all nodes $N_L \in \mathbb{M}_L$ such that $sim(N_L) = N_C$. To show that $|\mathbb{M}_C|$ is the cutoff, we will show that for some sequence of actions which leads to the first violation of the specification in any instance L of size $|\mathbb{M}_L| > |\mathbb{M}_C|$, there also exists a sequence of actions in the cutoff system of size $|\mathbb{M}_C|$ which leads to a violation. Specifically, for the example of Leader Election, for any arbitrary size system L with at least two nodes where nodes L_A and L_B (IDs A and B) are elected as leaders, we consider a cutoff system of size 2 with nodes C_A and C_B with the same IDs. The sim function is such that nodes in the portion of the ring in-between L_A and L_B (in the direction of communication) including L_B are simulated by C_B and the rest of the nodes in L are simulated by C_A.

Simulation Relation. Our observation is that a generic form of the simulation relation can be given in-terms of the sim function. Let σ_L and σ_C denote the states of two instances L and C respectively. We can in general view σ_X as a function from nodes of the instance to state components. The simulation R between states maintains the property that all effects of actions that can contribute to a violation and are present in the state of a node in L must be present in the state of its simulating node in C. This can be mathematically stated as follows:

$$(\sigma_L, \sigma_C) \in R \Leftrightarrow \forall N \in \mathbb{M}_L.\, \Omega(\sigma_L(N)) \subseteq \sigma_C(sim(N))$$

The relation uses the function Ω, which filters out those state components (i.e. effects of actions) which do not contribute in any way to the violation of the specification. With respect to our example of Leader Election, the above general strategy translates to the following,

$$\begin{aligned}(\sigma_L, \sigma_C) \in R \iff \forall N \in \mathbb{M}_L.\, &(\texttt{leader}_L(L_A) \rightarrow \texttt{leader}_C(C_A)) \wedge (\texttt{leader}_L(L_B) \rightarrow \texttt{leader}_C(C_B)) \\ &\wedge (\neg\texttt{leader}_L(L_A) \wedge \texttt{pnd}_L(A, N) \rightarrow \texttt{pnd}_C(A, sim(N))) \\ &\wedge (\neg\texttt{leader}_L(L_B) \wedge \texttt{pnd}_L(B, N) \rightarrow \texttt{pnd}_C(B, sim(N)))\end{aligned}$$

Lock Step. The lock-step describes the action(s) taken in C for every action taken in L. The generic strategy behind the lock-step is that actions involving any two nodes L_1 and L_2 in L are translated to actions involving $sim(L_1)$ and $sim(L_2)$ in C. Note that this might

result in some steps where $sim(L_1) = sim(L_2)$, which represents a *stuttering step* where L transitions according to the action but C stays in the same state. Stuttering steps can also occur when L performs some action that C cannot perform. Similarly, a single action in L might need to be simulated by more than one action in C. This behaviour can be encapsulated in a function τ. In general, action a in L is translated to $\tau(a)$ in C, where $\tau(a)$ can be a sequence of zero or more actions where zero actions represents a stuttering step. In our example of leader election, the lockstep relation τ is defined as follows:

$$\tau(\texttt{generate}_L(L_A, A, NG(L_A))) = \texttt{generate}_C(C_A, A, C_B)$$
$$\tau(\texttt{handle\_message}_L(L_A, A, NG(L_A))) = \texttt{handle\_message}_C(C_A, A, C_B)$$
$$\tau(\texttt{handle\_message}_L(L_A, B, NG(L_A))) = \texttt{handle\_message}_C(C_A, B, C_B)\texttt{generate}_C(C_B, B, C_A).$$

Note that the second action is required to maintain the simulation relation. Apart from these, the symmetric versions with A and B interchanged are also included in the lockstep.

FOL Encoding & Theorem. To prove that the simulation relation holds at each step, we show that it is an inductive invariant of the combined instances L and C. Given FOL encoding of the states and actions of the protocol, we construct the following FOL formula to check the correctness of the simulation relation:

$$(\sigma_L, \sigma_C) \in R \wedge a(\sigma_L, \sigma'_L) \wedge \tau(a)(\sigma_C, \sigma'_C) \wedge (\sigma'_L, \sigma'_C) \notin R \tag{1}$$

Here, a can be any of the actions possible in L according to the protocol description, and we use the notation $a(\sigma_X, \sigma'_X)$ to denote the change in state after the action.

We construct a FOL encoding consisting of the protocol states, actions and the simulation relation along with the formula 1. If the resulting formula is `UNSAT`, then the simulation relation holds at every step. Note that we are only interested in the first violation of the specification in any arbitrary instance, because once a single violation occurs, the specification is broken and the protocol is incorrect, therefore, if Φ denotes the specification, we also conjunct $\Phi(\sigma_L)$ and $\Phi(\sigma_C)$ to the above formula. We also need to show that violations would be preserved by the simulation relation:

$$\neg\Phi(\sigma_L) \wedge R(\sigma_L, \sigma_C) \wedge \Phi(\sigma_C) \tag{2}$$

▶ **Theorem 1.** *If the formulae 1 and 2 are unsatisfiable, and if the cutoff instance C does not violate the specification Φ, then no instance of the protocol violates Φ.*

3 Experiments and Future Work

The tool implementing the ideas described in the paper is still a work-in-progress. However, we have some positive preliminary results: we have been able to verify the leader election protocol and the significantly more complicated sharded key-value store protocol [4]. We use Z3 [2] as a back-end SMT solver, and in both cases, the verification time is in the order of a few seconds. In addition, we have manually checked the correctness of our approach on a variety of other protocols: Lock Service [13], Learning Switch [11], Distributed Lock Service [5]. As part of future work, we plan to finish the tool and apply our method on more complex protocols. In addition, we also want to leverage our observation regarding the relation between the cutoff instance and violations of the specification to automatically synthesize the cutoff instance.

To conclude, in this work, we have proposed an approach to significantly simplify and automate cut-off based proofs for verification of distributed protocols. Our experience is that cutoff-based proofs can be applied to a large number of distributed protocols, and we hope that this work would pave the way for more widespread application of this proof technique.

References

1. Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.
2. Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
3. E. Allen Emerson and Kedar S. Namjoshi. Reasoning about rings. In *POPL*, pages 85–94. ACM Press, 1995.
4. Yotam M. Y. Feldman, James R. Wilcox, Sharon Shoham, and Mooly Sagiv. Inferring inductive invariants from phase structures. In *CAV (2)*, volume 11562 of *Lecture Notes in Computer Science*, pages 405–425. Springer, 2019.
5. Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch, Bryan Parno, Michael L. Roberts, Srinath T. V. Setty, and Brian Zill. Ironfleet: proving practical distributed systems correct. In *SOSP*, pages 1–17. ACM, 2015.
6. Nouraldin Jaber, Swen Jacobs, Christopher Wagner, Milind Kulkarni, and Roopsha Samanta. Parameterized verification of systems with global synchronization and guards. In *CAV (1)*, volume 12224 of *Lecture Notes in Computer Science*, pages 299–323. Springer, 2020.
7. Haojun Ma, Aman Goel, Jean-Baptiste Jeannin, Manos Kapritsos, Baris Kasikci, and Karem A. Sakallah. I4: incremental inference of inductive invariants for verification of distributed protocols. In *SOSP*, pages 370–384. ACM, 2019.
8. Ognjen Maric, Christoph Sprenger, and David A. Basin. Cutoff bounds for consensus algorithms. In *CAV (2)*, volume 10427 of *Lecture Notes in Computer Science*, pages 217–237. Springer, 2017.
9. Kenneth L. McMillan and Oded Padon. Ivy: A multi-modal verification tool for distributed algorithms. In *CAV (2)*, volume 12225 of *Lecture Notes in Computer Science*, pages 190–202. Springer, 2020.
10. Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. Paxos made EPR: decidable reasoning about distributed protocols. *Proc. ACM Program. Lang.*, 1(OOPSLA):108:1–108:31, 2017.
11. Oded Padon, Kenneth L. McMillan, Aurojit Panda, Mooly Sagiv, and Sharon Shoham. Ivy: safety verification by interactive generalization. In *PLDI*, pages 614–630. ACM, 2016.
12. Marcelo Taube, Giuliano Losa, Kenneth L. McMillan, Oded Padon, Mooly Sagiv, Sharon Shoham, James R. Wilcox, and Doug Woos. Modularity for decidability of deductive verification with applications to distributed systems. In *PLDI*, pages 662–677. ACM, 2018.
13. James R. Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Thomas E. Anderson. Verdi: a framework for implementing and formally verifying distributed systems. In *PLDI*, pages 357–368. ACM, 2015.