On the Self-Stabilization of Dijkstra's Asynchronous Token Circulation

Asma Khoualdia
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Sami Cherif
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Stéphane Devismes
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Léo Robert
MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

Abstract

Dijkstra's token ring algorithm is a fundamental example of a self-stabilizing algorithm for solving mutual exclusion in an asynchronous distributed system arranged as a rooted directed ring. This paper studies the self-stabilization of this algorithm using an approach based on propositional satisfiability. We propose a logical modeling framework for the asynchronous executions of the algorithm, as well as the mechanisms for detecting convergence or divergence. Furthermore, we introduce an offset-based symmetry-breaking technique applied to the initial configurations to reduce redundant explorations of equivalent execution scenarios. In addition, we extended the study to restricted daemon assumptions to assess open challenges.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Constraint and logic programming; Theory of computation → Logic and verification

Keywords and phrases Self-stabilization, Token Circulation, Asynchronism, Satisfiability

1 Introduction

The notion of self-stabilization, introduced by Dijkstra in 1974 [3], refers to a distributed system's ability to return autonomously to a legitimate configuration from any initial state, ensuring fault tolerance. Dijkstra proposes three self-stabilizing token circulation algorithms operating in rooted directed rings. The first requires K states per process, where $K \geq n$ with n denoting the size of the ring. the K-state algorithm was shown to be self-stabilizing under a distributed unfair daemon in [1]. More recently, we have initiated the rigorous analysis of self-stabilizing algorithms via propositional satisfiability in [5]. However, this study is restricted to synchronous distributed systems. We propose a logical framework for modeling asynchronous executions of the algorithm, capturing state updates, convergence and divergence. We also study offset-based symmetry breaking and extend the analysis to synchronous and central unfair daemons, providing insights into unresolved divergence cases for non-synchronous and non-central executions.

2 Dijkstra's Token Ring

We consider a distributed system composed of $n \geq 3$ interconnected processes in a rooted directed ring. In general, the network is modeled as a directed graph. Here, the network is a directed ring, a process has unique successor and predecessor in the ring. Moreover, the ring being rooted, that is, a single process is distinguished as the *root* of the ring, the other processes are anonymous and therefore indistinguishable. We formalize the fact that a process holds a token by a predicate which depends on the local state of the process and

its predecessor. Conversely, divergence is exhibited by a sequence of configurations that can never converge. In the algorithm, a process is enabled in a given configuration when it holds the token. The system is asynchronous: it is controlled by an unfair distributed daemon which, at each step, activates an arbitrary non-empty subset of enabled processes, which hold a token. When activated, a process modifies its variable based on the rules given by the algorithms. We are interested in the self-stabilizing properties of the K-state token circulation algorithm over a ring of arbitrary size.

3 Formal Modeling of Dijkstra's Token Ring Algorithm

We model the behavior of Dijkstra's K-state algorithm on a network represented by a ring graph. To guarantee valid execution, it is necessary to maintain processes uniqueness and guarantee update semantics. For token possession, a process holding the token is enabled, and it can execute its action if chosen by the daemon. We focus on handling asynchronism as well, which leads to different possible executions starting from the same configuration. State updates depend on its activation and authorization. A process can be in one of these situations: (i) It is not enabled and retains its current state; (ii) It is enabled but not activated and maintains its state unchanged; (iii) It is both enabled and activated, so it updates its value. We focus on detecting: convergence, which guarantees that a legitimate configuration is reached from any initial configuration in a finite number of steps; secondly, divergence, which allows us to identify executions that cannot lead to a legitimate state. We aim at reducing the number of initial configurations by introducing the notion of offset-equivalence. Moreover, we study K-state algorithm under restricted daemon assumptions. More specifically, we investigate the following situations: (i) whether there exist *non-synchronous* executions that diverge when $K = n - 1$; and (ii) whether there exists *non-central* executions that diverge when $K < n - 1$.

4 Experimental Evaluation

We analyzed for ring sizes n ranging from 3 to 10 nodes, and varying the number of states K from 2 to 15. Implemented in Python with the PySAT library [4], the instances were solved using CaDiCaL solver [2]. The number of configurations is set to $t_f = \frac{3n(n-1)}{2}$. We also note that the Cardinality Network encoding was used to rewrite cardinality constraints in CNF form. We generated instances for convergence (CNV) and divergence (DIV) corresponding to the initial model (INI), then added the Offset-Elimination (OE) constraint on initial configurations, yielding 448 instances in total. We manage to exhibit convergence for all values of K when $n = 7$ and we demonstrated divergence for all the rings studied when $K \leq n - 1$. Overall, the results show that OE achieves a considerable improvement in solving time, convergence and divergence detection. Finally, we studied restricted daemon assumptions and used these observations to derive formal results on self-stabilizing behavior.

5 Conclusion

We present a formal approach based on propositional satisfiability to analyze the self-stabilizing behavior of the asynchronous K-state algorithm of Dijkstra. Furthermore, we introduced offset-based symmetry breaking on initial configurations and we extended the study to restricted daemon assumptions to address open challenges. As future work, it would be interesting to deepen the analysis of the non-synchronous case. It would also be relevant to extend our work to the two other token circulation algorithms of Dijkstra [3, 6].

References

1 Karine Altisen, Stéphane Devismes, Swan Dubois, and Franck Petit. *Introduction to Distributed Self-Stabilizing Algorithms*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2019. doi:10.2200/S00908ED1V01Y201903DCT015.

2 Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. Cadical 2.0. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024. doi:10.1007/978-3-031-65627-9_7.

3 Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974. doi:10.1145/361179.361202.

4 Alexey Ignatiev, António Morgado, and João Marques-Silva. Pysat: A python toolkit for prototyping with SAT oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018. doi:10.1007/978-3-319-94144-8_26.

5 Asma Khoualdia, Sami Cherif, Stéphane Devismes, and Léo Robert. Analyzing self-stabilization of synchronous unison via propositional satisfiability. In Maria Garcia de la Banda, editor, *31st International Conference on Principles and Practice of Constraint Programming, CP 2025, Glasgow, Scotland, August 10-15, 2025*, volume 340 of *LIPIcs*, pages 19:1–19:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.CP.2025.19, doi:10.4230/LIPICS.CP.2025.19.

6 Masahiro Kimoto, Tatsuhiro Tsuchiya, and Tohru Kikuno. On the time complexity of dijkstra's three-state mutual exclusion algorithm. *IEICE Trans. Inf. Syst.*, 92-D(8):1570–1573, 2009. URL: https://doi.org/10.1587/transinf.E92.D.1570, doi:10.1587/TRANSINF.E92.D.1570.