


Analyzing Self-stabilization of Synchronous Unison via Propositional Satisfiability

Asma Khouldia ✉


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

Synchronous unison is a classical clock synchronization problem in distributed computing, and especially in self-stabilization. This paper explores the self-stabilization of a synchronous unison algorithm proposed by Arora *et al.* using a propositional satisfiability-based approach. We give a logical formulation of the algorithm. This formulation includes the uniqueness of clock values at each node, the updates of clocks based on the minimum clock value in the neighborhood, and the detection of convergence or divergence. To optimize the models, additional constraints are introduced to reduce redundant cases of initial configurations to be analyzed. Our approach not only verifies the algorithm’s behaviour but also offers insights into enhancing its robustness and applicability to broader distributed systems.

Keywords and phrases Self-stabilization, Synchronous Unison, Satisfiability

1 Introduction

The notion of self-stabilization, introduced by Dijkstra in 1974 [10], refers to a distributed system’s ability to return autonomously to a legitimate configuration from any initial state, ensuring fault tolerance. A key problem is *Synchronous Unison*, where processes must synchronize local clocks modulo a period m . Arora *et al.* [3] proposed a self-stabilizing algorithm for $m \geq \max(2, 2\mathcal{D} - 1)$, with \mathcal{D} the network diameter, though the tightness of this bound remains open [2]. This work, accepted at CP 2025, introduces a formal SAT-based approach [4, 7] that encodes clock update rules of synchronous unison as CNF constraints, enabling systematic verification of stabilization or its absence across different topologies, paving the way for optimized analysis of other distributed algorithms.

2 Synchronous Unison

We study the *synchronous unison* problem [3], where n processes communicate over a connected graph $G = (V, E)$. Each process p maintains a clock variable $p.c \in \{0, \dots, m - 1\}$, where m is a global *period*. In each synchronous round, enabled processes update their clock to $(\min\{q.c : q \in N(p) \cup \{p\}\} + 1) \bmod m$. The global state at time i is a configuration $\gamma_i \in \Gamma_{n,m}$, with $\Gamma_{n,m} = \{0, \dots, m - 1\}^n$. The algorithm aims to reach a *legitimate* configuration where all clocks are equal. It is said to *converge* if this state is eventually reached from any initial configuration, and *diverge* if some executions remain forever unsynchronized [8, 11]. Convergence is guaranteed for all connected graphs of diameter \mathcal{D} when $m \geq 2\mathcal{D} - 1$, with stabilization in at most $3\mathcal{D} - 2$ rounds [2]. To analyze self-stabilization behaviour for a given m , we must examine up to m^n configurations but we reduce this bound to a reasonable number in order to avoid combinatorial explosion, following the principles of Bounded Model

44 Checking (BMC) [5]. We encode this dynamic behavior as a SAT instance using Boolean
 45 variables and CNF constraints in order to take full advantage of modern SAT solvers, which
 46 are able to efficiently solve complex instances with a high number of variables and clauses
 47 [6, 13].

48 **3 Formal Modeling of Synchronous Unison through SAT**

49 We model synchronous unison on a network $G = (V, E)$ of n processes with clock values
 50 in $M = \{0, \dots, m - 1\}$ over t_f time steps $T = \{0, \dots, t_f - 1\}$. Variables $h_{p,t,v}$ encode
 51 whether process p has clock v at time t , ensuring clock uniqueness per process via cardinality
 52 constraints. Clock updates are defined so each process sets its next clock to one plus the
 53 minimum clock in its closed neighborhood, modulo m . Convergence is expressed by requiring
 54 that no final configuration has all clocks equal, while divergence is modeled by detecting cycles
 55 where an illegitimate initial configuration repeats later in the execution. Additional variables
 56 c_t and $s_{p,t,v}$ capture cycles and clock similarities between configurations. The constraints are
 57 transformed into CNF, with complexity mainly dependent on the graph's maximal degree,
 58 number of nodes, period and number of analyzed configurations. This formalization enables
 59 reasoning about convergence and divergence properties of the synchronous unison algorithm
 60 through SAT solving.

61 **4 Experimental Evaluation**

62 We analyzed classical network topologies—chains, rings, and stars—defined by node count
 63 n , period m , diameter \mathcal{D} , and maximum degree d_{\max} . Using Python and PySAT with the
 64 Cadical solver on an Intel Core i7, we generated 4968 instances: $n = 3\text{--}20$, $m = 2\text{--}20$ for chains
 65 and rings, and $n = 3\text{--}10$, $m = 2\text{--}10$ for stars. Models included initial convergence/divergence
 66 (INI_{CNV} , INI_{DIV}) and elimination constraints—rotation elimination (RE), lexicographic
 67 order (LO), and convergence variants (IC_P , IC_X)—plus combinations. We set the number
 68 of analyzed configurations to $t_f = 3\mathcal{D}$. Convergence occurred for $m \geq \max\{2, 2\mathcal{D} - 1\}$,
 69 though some instances converged earlier. Stars had tight bounds: divergence at $m = 2$,
 70 convergence at $m \geq 3$. The model solved most instances but slowed for larger graphs,
 71 especially for stars due to high topology degree. Best results were achieved though constraint
 72 combinations: $ER + IC_X$ for rings and $LO + IC_P$ for stars. These refinements enhance
 73 scalability, particularly in high-degree or centralized networks.

74 **5 Conclusion**

75 We present a SAT-based approach to analyze the self-stabilizing synchronous unison al-
 76 gorithm [3], detecting convergence and divergence in chains, rings, and stars. Encoding states
 77 as logical constraints enables to detect convergence or prove divergence. Model Efficiency
 78 was improved by eliminating redundant initial configurations. Divergence was observed and
 79 proven to occur in stars for period $m = 2$. Future work targets other graph types, tighter
 80 bounds, and symmetry-based cycle detection as well as extending insights to asynchronous
 81 models [8, 9] and dynamic topologies [1, 12].

References

- 1 Karine Altisen, Stéphane Devismes, Anaïs Durand, Colette Johnen, and Franck Petit. Self-stabilizing systems in spite of high dynamics. *Theor. Comput. Sci.*, 964:113966, 2023. URL: <https://doi.org/10.1016/j.tcs.2023.113966>, doi:10.1016/J.TCS.2023.113966.
- 2 Karine Altisen, Stéphane Devismes, Swan Dubois, and Franck Petit. *Introduction to Distributed Self-Stabilizing Algorithms*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2019.
- 3 Anish Arora, Shlomi Dolev, and Mohamed G. Gouda. Maintaining digital clocks in step. *Parallel Processing Letters*, 1:11–18, 1991. doi:10.1007/bfb0022438.
- 4 Armin Biere. Handbook of satisfiability. In *Frontiers in Artificial Intelligence and Applications*, pages 75–98. IOS Press, 2009. doi:10.3233/978-1-58603-929-5-75.
- 5 Armin Biere. Bounded model checking. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 739–764. IOS Press, 2021. doi:10.3233/FAIA201002.
- 6 Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of Satisfiability: Second Edition*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. URL: <https://books.google.fr/books?id=dUAvEAAAQBAJ>.
- 7 Stephen A. Cook. The complexity of theorem-proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971. doi:10.1145/800157.805047.
- 8 Jean-Michel Couvreur, Nissim Francez, and Mohamed G. Gouda. Asynchronous unison (extended abstract). In *The 12th International Conference on Distributed Computing Systems (ICDCS)*, pages 486–493. IEEE Computer Society, 1992. doi:10.1109/ICDCS.1992.235005.
- 9 Stéphane Devismes, David Ilcinkas, Colette Johnen, and Frédéric Mazoit. Being efficient in time, space, and workload: a self-stabilizing unison and its consequences. In Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen, editors, *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, March 4-7, 2025, Jena, Germany*, volume 327 of *LIPICs*, pages 30:1–30:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: <https://doi.org/10.4230/LIPICs.STACS.2025.30>, doi:10.4230/LIPICs.STACS.2025.30.
- 10 Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974. doi:10.1145/361179.361202.
- 11 Danny Dolev and Dahlia Malkhi. Consensus: Perspectives and challenges. In *Proceedings of the Tenth International Workshop on Distributed Algorithms (WDAG)*, pages 1–12, 1995. doi:10.1007/3-540-60220-8_1.
- 12 Shlomi Dolev and Ted Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago Journal of Theoretical Computer Science*, 1995.
- 13 João P. Marques-Silva and Kareem A. Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.