Title

Advances in Discrete Chemical Computation: Algorithms, Lower Bounds, and Software for Population Protocols

Author

Severson, Eric

Publication Date

2021

Advances in Discrete Chemical Computation: Algorithms, Lower Bounds, and Software for Population Protocols

By

ERIC E. SEVERSON
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Professor David Doty, Chair

Professor Jesús A. De Loera

Professor David Soloveichik

Committee in Charge

2021

Contents

Advances in Discrete Chemical Computation:

Algorithms, Lower Bounds, and Software for Population Protocols

Abstract

The **population protocols** model [19] describes a population of n finite-state computational agents, whose states change through successive pairwise interactions. Crucially, the agents have no control over the interaction schedule, and agents in the same state are indistinguishable. One motivating example are **chemical reaction networks**, viewing the agents as molecules undergoing pairwise collisions and changing state via reactions. Since this model is mathematically fundamental, equivalent processes have been independently studied in many fields. This work contributes new software tools for simulating population protocols, studies the time and space complexity of fundamental tasks, and explores various additional constraints on the model that ensure reliability, low bandwidth, and composability.

A standard choice of dynamics is to pick a uniform random pair of agents to interact at each step. The set of states and update rule then give a discrete Markov process. In the first part of this work, we introduce `ppsim`, a software package for simulating such processes. Complex protocols can be succinctly described in Python code, and are simulated by an algorithm [41] which is asymptotically faster than the standard algorithm which repeatedly samples interacting pairs of agents and updates their state.

Chemical reaction networks (CRNs) are typically described by continuous time dynamics, where the Gillespie algorithm [103] is able to sample trajectories from the chemical master equation. A CRN with only unimolecular (1 input, 1 output) or bimolecular (2 input, 2 output) reactions, with arbitrary rate constants, can be faithfully represented by a continuous time population protocol. In this way, `ppsim` is able to exactly sample the same chemical master equation, while achieving asymptotic gains in running time that can exactly simulate hundreds of billions of reactions in seconds.

What differentiates population protocols from other models that capture the same dynamics is that, coming from the field of distributed computing, it studies which computational tasks the population can perform. Typically, there is some global property of the initial configuration, and the task is to converge to a configuration where each agent locally has knowledge of this property.

The correct output must be reached with probability 1 and then be **stable**, unable to be changed by future interactions. A prototypical example is the **majority** problem. There, the initial configuration has two states A and B, and the agents must determine whether there are more A, more B, or a tie.

A recent line of work has established progressively more efficient protocols for solving the majority problem. In the second part of this work, we describe and analyze a majority protocol using $O(\log n)$ states and taking expected time $O(\log n)$ ($O(n \log n)$ pairwise interactions). Existing lower bounds [6] for protocols with natural constraints show that this protocol is asymptotically optimal in both state space and convergence time.

The other most studied problem in population protocols is **leader election**. There, the population must reach a configuration with a unique agent in a designated leader state. Typically, the entire population starts in some fixed initial state, but the additional **self-stabilizing** constraint requires the protocol to correctly converge starting from **any** possible initial configuration. In the third part of this work, we develop multiple new protocols for solving **self-stabilizing leader election**, exhibiting a trade-off in state space and convergence time.

In the original model, the agents' state set was fixed and independent of the population size n. Most recent results, including the work above, allow the number of states to scale with the population size. The standard transition rule assumes that when two agents interact, each observes the entire state of the other agent. In the fourth part of this work, we initiate the study of **message complexity** for population protocols, where the state of an agent is divided into an externally-visible **message** and an internal component, and only the message can be observed by the other agent in an interaction. The focus is on constant-size message complexity, so only the number of internal states can grow with the population size. This models a regime where communication bandwidth between agents is more limited than internal storage. Without restricting convergence time, we obtain an exact characterization of the computational power, based on the number of internal states. We also introduce multiple novel $O(\text{polylog}(n))$ expected time protocols, solving problems of leader election, general purpose broadcast, and population size counting.

The final part of this work considers the more general model of **discrete chemical reaction networks**, which now allow reactions such as $X \to 2Y$ which change the size of the population. This reaction can be viewed as computing the function $f(x) = 2x$, treating species X as input and

Y as output. An additional **output oblivious** constraint, where the output species cannot be the input to any further reactions, enables these CRNs to be easily composed. We exactly characterize which integer-valued functions $f : \mathbb{N}^d \to \mathbb{N}$ can be computed by these output oblivious CRNs.

Acknowledgments

I'm deeply grateful to my advisor, Dave Doty, for help in many ways. For getting me interested in the field and providing stimulating problems that blossomed into enough material for this dissertation. For being an active participant in all this research, always available for long discussions about interesting ideas, both in front of the whiteboard and on Zoom. For the continued generous financial support I've had during these last few years. And for all additional skills I've learned along the way: how to write better papers, how to give better presentations, and how to write better software, letting me more properly now think of myself as a computer scientist.

Also to all the many coauthors that were part of this work: Mahsa Eftekhari, David Haley, James Aspnes, Talley Amir, Leszek Gąsieniec, Grzegorz Stachowiak, Przemyslaw Uznański, Janna Burman, Ho-Lin Chen, Hsueh-Ping Chen, Thomas Nowak, and Chuan Xu. I view math as a social endeavor. Having people to share ideas with along the way was vital sustaining my interest in research, and a large reason I was able to complete a PhD.

To the community in the math department at UC Davis: all the staff, professors, and classmates who made the first years of the program filled with active learning and stimulating ideas. I developed so much of my mathematical maturity during my first year of graduate school, where actively working with my classmates was key to that growth. And I felt a part of the program in ways I never did in undergrad.

And above all, to my wife Jillian. For coming along to Davis and building a life here, as well as a thriving business. For becoming my lifelong partner on top of being my best friend, and caring about me more than I knew somebody could. Your love and support has been instrumental to all the ways I have been able to grow as a person.

The financial support for the work in this dissertation came from NSF award 1900931 and CAREER award 1844976.

CHAPTER 1

Introduction

Nature is filled with examples where simple local rules, applied across a large scale, give rise to emergent global behavior. One simple abstract model that captures this phenomenon is **population protocols**, where a large population of agents undergo random pairwise interactions which change their state. These local state changes lead to global stochastic dynamics, which are determined by the set of states and the **transition function** that describes how each pair of interacting states will update in an interaction. These rules define a protocol, and protocols are studied which solve computational tasks, where often each agent will locally obtain information about some global property of the system.

A primary application of this model is as a simple description of **chemical reaction networks (CRNs)**, where the agents are molecules randomly colliding in solution and the transition function describes the allowable chemical reactions. Recent advances in DNA nanotechnology have implemented artificial CRNs using the physical primitive of nucleic-acid strand displacement cascades [56, 66, 152, 153]. This lets CRNs be viewed as a programming language, which motivates the work in population protocols about the computational power of these models.

The population protocols model is mathematically fundamental, and has thus been independently studied across a range of fields. It was originally motivated by wireless sensor networks [19], but has been independently studied for social networks: modelling the propagation of trust [77], diseases [27], and rumors [73]. It also arises in the study of game theory [4, 48, 49], and to model gene regulatory networks [50] or animal populations [165]. Fundamental processes in population protocols are equivalent to well-studied distributed computing tasks such as gossip algorithms [147] and load balancing [39, 130]. Generalizing beyond uniform random pairs of interacting agents to more complicated interaction graphs gives the models of interacting particle systems [124] or stochastic cellular automata [164], both widely studied in physics and probability theory.

Population protocols describe discrete CRNs where every reaction has 2 inputs (reactants) and 2 outputs (products). The more general CRN model allows arbitrary number of reactants and products in each reaction, which can now change the size of the population. Still, fundamental results about which functions CRNs can compute [63] were based on theoretical results from population protocols [20]. These discrete CRNs are equivalent to multiple other mathetmatical models: vector addition systems [119], Petri nets [139], and commutative semi-Thue systems (or, when all transitions are reversible, "commutative semigroups") [59,126]. These other models have focused more on the question of reachability. The computational complexity of determining if a configuration in a CRN is reachable had been a long-standing open question, and was recently resolved as being Ackermann-complete [72,123] (i.e., the problem is decidable, but not even primitive recursive).

In this work, we advance the theoretical understanding of the computational power of population protocols and chemical reaction networks. In Chapter 2, we introduce `ppsim`, a Python software package built to easily encode and efficiently simulate complicated population protocols. For any chemical reaction network with unimolecular (1 input, 1 output) and bimolecular (2 input, 2 output) reactions (and arbitrary rate constants), the `ppsim` simulation exactly samples the chemical master equation [104] with asymptotic gains in run-time over standard approaches [103]. Chapter 3 focuses on the well-studied **majority** problem, where the population must reach consensus on which of two opinions was in the initial majority. There, we provide the first protocol which is asymptotically optimal in both state space and run-time. Then in Chapter 4, we consider the problem of **self-stabilizing leader election**, where starting from any adversarial initial configuration, the population must stabilize to a configuration with a unique leader agent. There was one existing protocol [54] which solved this problem in its original setting, and we provide multiple more efficient protocols that exhibit a trade-off in state space and run-time. Next in Chapter 5, we introduce the idea of **message complexity** for population protocols, which enables the distinction between communication complexity and internal storage, both of which increase in larger-state protocols that are more common in the recent literature. Here, we focus on protocols with limited communication bandwidth but larger internal storage. In this regime, we find an exact computational characterization as well as multiple novel time-efficient protocols. Finally, Chapter 6 studies the more general model of discrete CRNs, which compute functions $f : \mathbb{N}^d \to \mathbb{N}$ using counts of designated input and output species, and are able compute exactly the **semilinear functions** [63]. We explore the additional **output-oblivious** constraint, which forbids the output species from being a reactant in any further reactions, and enables straightforward composition of CRNs. We exactly characterize which subclass of semilinear functions are computable by this restricted class of modular CRNs.

1.1. Simple Example Protocols

As a warm-up to the population protocol model, we start by analyzing a few examples of important protocols that use a very small number of states.

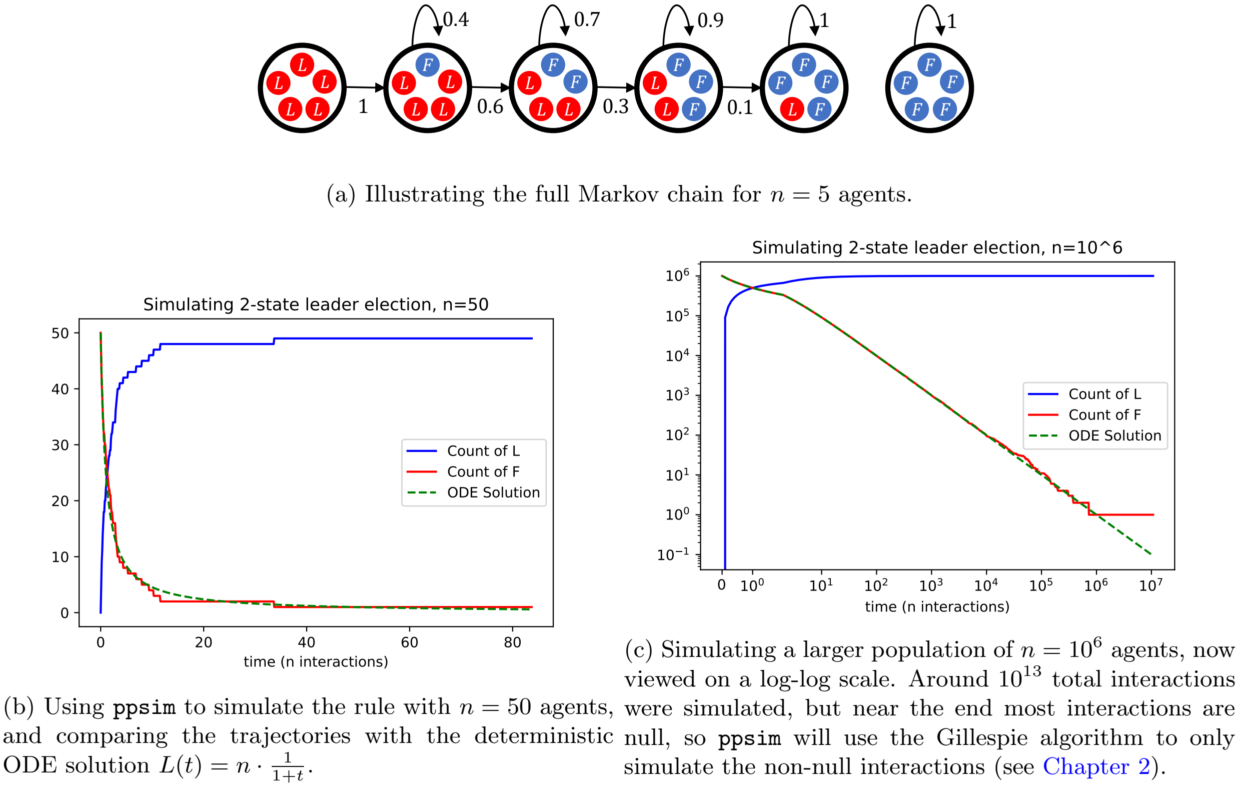
1.1.1. Leader Election. As a first example, consider the protocol with only two states $\{L, F\}$ and the single non-null transition

$$L, L \to L, F.$$

We start in an initial configuration with all n agents in state L. The dynamics then correspond to choosing a uniform random (ordered) pair of agents to interact at each step. If both are in state L, then the latter agent changes to state F, and otherwise there is no change. This gives rise to a discrete-time, discrete-state Markov process. Because of the symmetry where each pair of agents is equally likely to interact, the dynamics of a configuration only depend on the counts of agents in each state. We thus have a Markov chain, whose state space are the $n+1$ configurations corresponding to $\#L = 0, 1, \dots, n$, see Fig. 1.1a.

This simple 2-state rule solves the task of **leader election**, by the standard technique of "fratricide" where whenever two leader agents in state L meet, one becomes a follower in state F. Starting from the $\{nL\}$ configuration, the system will converge to the $\{1L, (n-1)F\}$ configuration with a unique leader agent. Crucially, this configuration is **stable**, where the leader status of all agents can no longer change. In this case, it has the even stronger property of being **silent**, where the only possible future transitions are null. In other words, by being silent, the Markov chain has reached a unique terminal state. But it was only necessary to be stable, reaching a strongly connected component of configurations that all had a single agent in a subset of possible leader states. Note also that this protocol does **not** solve the strictly harder problem of self-stabilizing leader election that is studied in Chapter 4, because of the possible initial configuration $\{nF\}$.

This protocol has optimal space complexity, using only 2 states (1 bit of memory). To think about time complexity, we analyze the convergence time of the Markov chain, which for this simple case we can do in full detail. From each configuration $\{kL, (n-k)F\}$, the probability of the non-null



(a) Illustrating the full Markov chain for $n = 5$ agents.

(b) Using `ppsim` to simulate the rule with $n = 50$ agents, and comparing the trajectories with the deterministic ODE solution $L(t) = n \cdot \frac{1}{1+t}$.

(c) Simulating a larger population of $n = 10^6$ agents, now viewed on a log-log scale. Around 10^{13} total interactions were simulated, but near the end most interactions are null, so `ppsim` will use the Gillespie algorithm to only simulate the non-null interactions (see Chapter 2).

FIGURE 1.1. Simulating the 2-state leader election protocol, defined in `ppsim` as `rule = {('L','L'):('L','F')}`.

transition between two leaders is $\frac{\binom{k}{2}}{\binom{n}{2}} = \frac{k(k-1)}{n(n-1)}$. Then the number of interaction steps it takes for this transition to decrement the count of leaders is a geometric random variable G_k, with $\mathbb{E}[G_k] = \frac{n(n-1)}{k(k-1)}$. The total number of steps to converge $T = G_n + \ldots + G_2$ is then the sum of independent geometrics, with expected value

$$\mathbb{E}[T] = \sum_{k=2}^{n} \frac{n(n-1)}{k(k-1)} = n(n-1) \sum_{k=2}^{n} \left(\frac{1}{k-1} - \frac{1}{k} \right) = n(n-1)\left(1 - \frac{1}{n}\right) = (n-1)^2.$$

In discussing time complexity, the natural time scale is that of **parallel time**, with $\Theta(n)$ pairwise interactions per time step, so time is proportional to the number of interactions per agent. Thus this 2-state leader election takes $\Theta(n)$ time to stabilize. It was later shown that any leader election protocol, when the number of states is constant (independent of population size n), must take linear time to stabilize, so this simple protocol is optimal in that sense. See Section 1.3 for more discussion.

This notion of time also lines up with other common modelling choices, such as a “mean-field” approximation using differential equations. This would be equivalent to using the standard law of mass action [95] to model this CRN $L, L \to L, F$. Letting $l = \frac{\#L}{n}$ denote the concentration of leaders, we would have the differential equation $\frac{dl}{dt} = -l^2$ with initial value $l(0) = 1$ and solution $l(t) = \frac{1}{1+t}$. When all counts are large, this is a good approximation to the discrete stochastic process, see Fig. 1.1b and Fig. 1.1c. This convergence of the discrete stochastic model to the deterministic continuous model in the limit of large counts can be made rigorous [121, 166, 168]. Unfortunately, these results do not give powerful black box methods for reasoning about population protocol dynamics. The error bound between the differential equation and stochastic system increases exponentially with time, and the results only hold if all states are present in large count, which is not the case in most protocols of interest. For this 2-state process, the ideal tool to get large deviation bounds on the likely behavior would be tail bounds on the sum of independent geometrics [115].

1.1.2. Epidemic (Rumor Spreading) Process. The next two state rule is essentially the reverse reaction. We will now call the states $\{I, S\}$ and have a single non-null transition [1]

$$I, S \to I, I.$$

The names come from viewing this as a simple SI epidemic model, where susceptible agents in state S are infected by agents in state I. As a CRN, this describes an autocatalytic reaction. This is also the natural way to broadcast information, where the informed state I spreads through the whole population. For this reason, this is an extremely common primitive in more complicated protocols, whenever there is a Boolean property $\mathtt{i} \in \{\mathsf{True}, \mathsf{False}\}$ of the agents' states which gets updated as $a.\mathtt{i}, b.\mathtt{i} \leftarrow (a.\mathtt{i} \text{ or } b.\mathtt{i})$ in an interaction between agents a and b.

This protocol by itself can also be viewed as solving the **detection** problem, determining whether some initial agent is in state I. Recent works [9, 93] have found protocols for detection with additional desirable properties, being self-stabilizing and robust to faulty reactions.

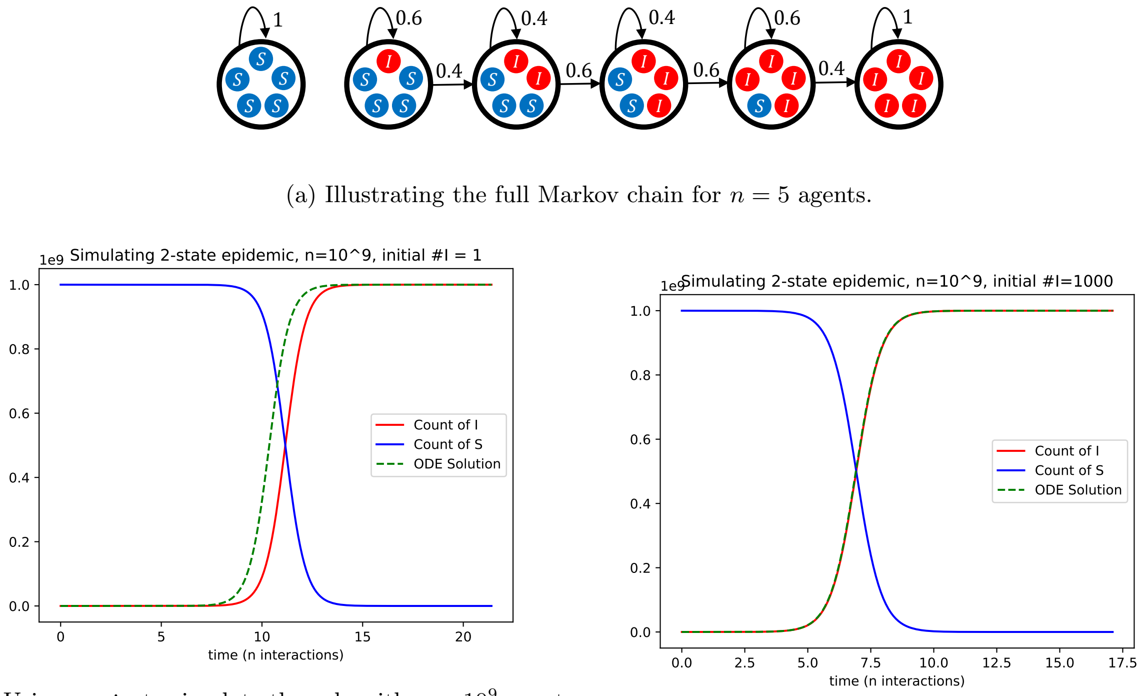
We can analyze this 2-state protocol in a very similar way to Section 1.1.1. Now the Markov chain (see Fig. 1.2a) will reach the terminal $\{nI\}$ configuration as long as there are any infected

[1] Here, the order of interacting agents does not matter, giving a **two-way epidemic**. Some works in the literature consider the similar **one-way epidemic**, where the interaction order matters. This gives an identical process, just progressing at half the rate.

agents. From each configuration $\{kI, (n-k)S\}$, the number of interactions until the next infection is a geometric random variable G_k with probability $\frac{k(n-k)}{\binom{n}{2}}$. Starting from a single infected agent, the total number of steps to converge, $T = G_1 + \ldots + G_{n-1}$ is again the sum of independent geometrics. The expected value is now

$$\mathbb{E}[T] = \sum_{k=1}^{n-1} \frac{n(n-1)}{2k(n-k)} = \frac{n-1}{2} \sum_{k=1}^{n-1} \left(\frac{1}{k} + \frac{1}{n-k} \right) = (n-1) \sum_{i=1}^{n-1} \frac{1}{i} \sim n \ln n.$$

Thus it takes expected $\ln n$ parallel time for the epidemic to finish.



(a) Illustrating the full Markov chain for $n = 5$ agents.

(b) Using `ppsim` to simulate the rule with $n = 10^9$ agents starting from $\#I = 1$, compared with the deterministic ODE solution $I(t) = n \cdot \frac{1}{1+(n-1)e^{-2t}}$.

(c) Now starting from $\#I = 1000 = n^{1/3}$, and comparing to the ODE solution $I(t) = n \cdot \frac{1}{1+(n^{2/3}-1)e^{-2t}}$.

FIGURE 1.2. Simulating the 2-state epidemic protocol, defined in `ppsim` as `rule = {('I','S'):('I','I')}`.

$\Theta(\log n)$ time is the optimal time for efficient computation in population protocols, because this is also how long it takes for each agent to have an interaction. If we picked one agent at each interaction (or were waiting for each agent to be the first **initiator** agent in the interacting pair), then waiting for all n agents to be chosen would exactly be the well-known **coupon collector** process. This gives an expected number of interactions $n\sum_{i=1}^{n}\frac{1}{i} \sim n\ln n$. Since two agents are picked at each interaction[2] , it will take an expected $\sim \frac{1}{2}n\ln n$ interactions for all agents to interact.

A common strategy for more complicated protocols is to try to organize the population into synchronous rounds, when they can all perform the same part of an algorithm. These synchronization sub-protocols are called **phase clocks**, where each phase lasts $\Theta(\log n)$ time. This is necessary for the entire population to interact during the phase, and it is also sufficient to broadcast messages by epidemic that will reach the entire population. The epidemic process itself is the basis for the original leader-driven phase clock [**21**], see Section 1.3 for more discussion.

The mean field approximation to the epidemic process gives the well-known logistic curve. With concentration i of infected agents, the probability of choosing an infected-susceptible pair gives the differential equation $\frac{di}{dt} = 2i(1-i)$. This has general solution $i(t) = \frac{1}{1+e^{-2t+C}}$. Fitting the initial condition $i(0) = \frac{1}{n}$ corresponding to $\#I = 1$, as in Fig. 1.2b, we see a random offset from the deterministic curve, coming from the variance in how quickly the epidemic initially spreads. [3]

Deviation bounds on the time for the epidemic process are used multiple times in this work. A detailed analysis in [**134**] found very precise upper bounds on the time for the epidemic to complete. These were simplified in Lemma 4.2.7 in order to be easier to apply, while still as sharp as possible. As an application, we study a related **roll call** process. There, each agent has a unique piece of information (its name), and in each interaction the agents share the set of all names they have heard about (the roster). While it would take a very large number of states (exponential in n) to realize, this process shows when it is information-theoretically possible for every agent to have complete information about initial global configuration. Viewing the spread of each name as an epidemic, we are able to get bounds on the time for the roll call process to complete, and show the expected time is $\sim \frac{3}{2}n\ln n$.

We also use time bounds for sections of the epidemic process, which come from tail bounds on the sum of independent geometric random variables [**115**]. The time for an epidemic to spread from a fraction a to a fraction b of the population (where $0 < a < b < 1$ are independent of population size n) has extremely low variance, as shown in Lemma 3.3.5. Fig. 1.2c shows an example, where an

[2] Since each successive pair of agents must be different, this is slightly faster than choosing two independent coupons at each step, but the difference from the exact coupon collector process is asymptotically negligible.

[3] It follows from large deviation theory of the coupon collector process [**112**] that the random offset has a Gumbel distribution.

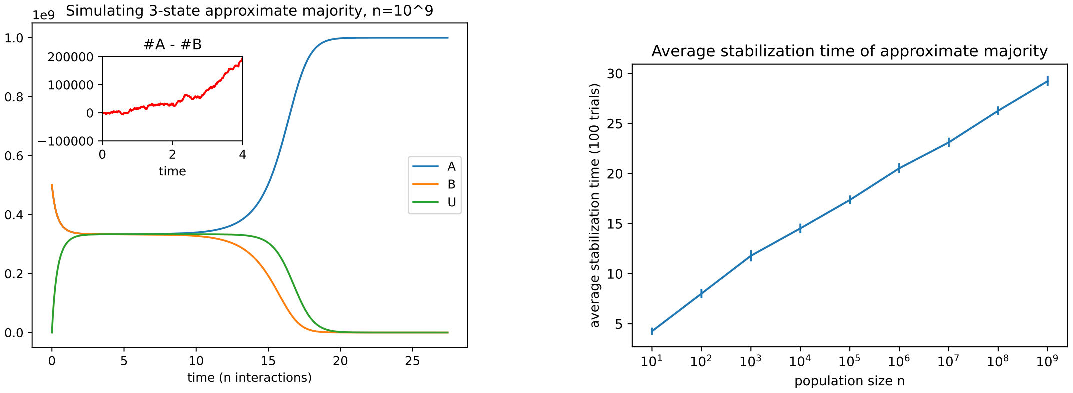
epidemic starting from a fraction $a = 10^{-6}$ of the population very follows a deterministic trajectory. In Lemma 5.4.4, we consider a more general setting where only a subset of agents are susceptible.

1.1.3. Approximate Majority. One of the most well-studied 3-state protocols can be described the rules

$$A, B \to U, U \qquad A, U \to A, A \qquad B, U \to B, B.$$

Intuitively, states A and B can be thought of as “opinionated”, whose opinions cancel to become the undecided U state. The U agents then adopt the opinion of the an opinionated agent. The agents will reach a consensus all-A or all-B configuration, and this consensus output is very likely to be the initial majority opinion.

As a result of this useful behavior and small number of states, equivalent rules have been found to exist in real biological networks [57]. Yet despite only having 3 states, rigorous proofs about these dynamics are surprisingly nontrivial. It was shown in [22] that the configuration will stabilize in $O(\log n)$ time, and if the initial majority has an advantage of at least $\omega(\sqrt{n}\log n)$ agents, then this opinion will become the consensus with high probability. A later work [69] simplified the arguments and showed high-probability correctness required a smaller gap of $\Omega(\sqrt{n \log n})$ agents.



(a) Using `ppsim` to simulate the rule with $n = 10^9$ agents starting from equal counts of A and B. The inset shows the difference between counts performing a random walk until A attains a significant majority.

(b) For a range of values of n, 100 trials were simulated starting from $\#A = \#B = n/2$. The average time until stabilizing is shown here, on a log-linear plot to clearly show the $\Theta(\log n)$ time stabilization.

FIGURE 1.3. Simulating the 3-state approximate majority protocol, defined in `ppsim` as `rule = {('A','B'):('U','U'), ('A','U'):('A','A'), ('B','U'):('B':'B')}`.

Fig. 1.3a shows a typical example of the approximate majority dynamics, starting from an equal counts of half a billion A and B agents. The corresponding system of ODEs has a saddle point at

$(\frac{\#A}{n}, \frac{\#B}{n}, \frac{\#U}{n}) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. We see the discrete process quickly approach this point, while at a finer scale the difference between counts of A and B is essentially performing a random walk. The most involved part of the rigorous analysis is to show an anti-concentration result, that these random effects push the system to then converge to the stable fixed points at either all-A or all-B. Because of these random effects, if the initial gap between opinions is significantly small, either opinion is equally likely.

Fig. 1.3b shows data generated by `ppsim`, which supports the result that even from an initial split case of half A and half B, the dynamics reach consensus in $O(\log n)$ time. Note to get data that conclusively supports the $O(\log n)$ time stabilization requires n to scale across a wide range of orders of magnitude, making it useful that `ppsim` can handle such large population sizes. See Fig. 2.2 for a comparison of how long `ppsim` takes to simulate this rule on large population sizes, compared to existing CRN simulators.

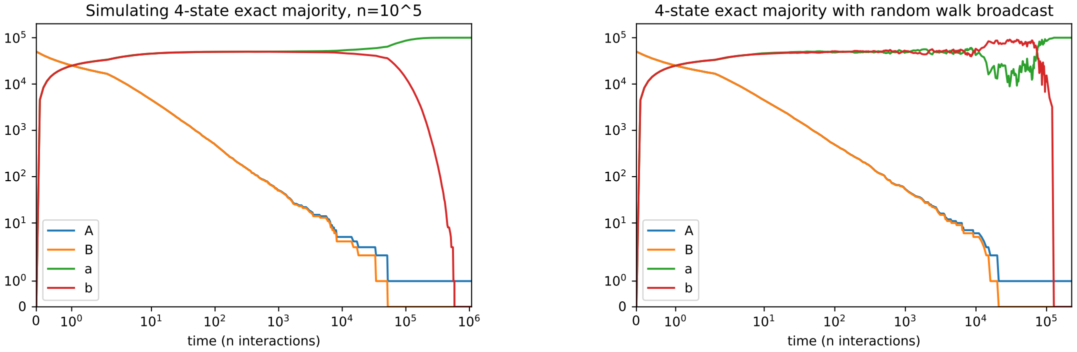
1.1.4. Exact Majority. The 3-state protocol above does not solve the exact majority problem, because the agents are not guaranteed to reach the correct answer of which state was in the initial majority. It is provably necessary to use at least 4 states [127], but with an additional state the exact majority problem can be solved. The basic rules are

$$A, B \to a, b \qquad A, b \to A, a \qquad B, a \to B, b.$$

Intuitively, when states A and B meet, they become the "passive" states a, b, which hold the same output opinion. The "active" A and B agents then convert the passive agents to their opinion. These rules now preserve $\#A - \#B$ as an invariant, which is key to correctness. Eventually, the only remaining active agents have the majority opinion, and they convert all the passive agents and stable consensus is reached.

If the initial gap between opinions is a constant fraction $\#A - \#B = \frac{c}{n}$, then all active minority agents will be eliminated by the first "cancel" reaction in $\Theta(\log n)$ time (the precise time shown in Lemma 3.3.6) and then all passive agents will be converted to the correct opinion in $\Theta(\log n)$ time (see Lemma 3.3.7).

The worst case for convergence time happens when the initial counts are almost equal[4]. For example, if the initial gap $\#A - \#B = 1$, then the final cancel reaction (with $\#A = 2, \#B = 1$) will take $\Theta(n)$ time (the sum of the times for all cancel reactions also takes $\Theta(n)$ time, with an analysis similar to Section 1.1.1). Waiting for the one A agent to interact with and convert all passive agents looks like the coupon collector process, and takes $\Theta(n \log n)$ time. See Fig. 1.4a for a simulation with this initial gap $\#A - \#B = 1$.



(a) Using **ppsim** to simulate the rule with $n = 10^5 + 1$ agents starting with $\#A - \#B = 1$. The final remaining active A agent must directly convert all remaining passive b agents to a.

(b) Starting from the same initial condition with additional random walk transitions between passive a and b states. Now with a single remaining A agent, the population more quickly stabilizes to all a.

FIGURE 1.4. Simulating the 4-state exact majority protocol, defined in **ppsim** as `rule = {('A','B'):('a','b'), ('A','b'):('A','a'), ('B','a'):('B':'b')}`. Random walk transitions are added as `rule | {('a','b'):('b','a'), ('b','a'):('b','b')}`.

This convergence time can be dropped to $\Theta(n)$ by the "random-walk broadcast" technique from [21]. This adds the additional pair of ordered transitions

$$(a, b) \to (a, a) \qquad (b, a) \to (b, b).$$

Whenever an a meets a b agent, they both adopt one the same state based on the order of the interaction. This rule makes the count of a and b agents perform a symmetric random walk. The remaining active A agent forces this random walk to reach the all-a configuration, which is shown in [21] to take $\Theta(n)$ time. See Fig. 1.4b for a simulation that shows the effect of these additional random walk transitions.

[4] Note this 4-state rule does have not well-defined output if the initial configuration is a tie. There is a related 6-state rule which additional recognizes this tie case, used as Phase 10 for our the majority protocol in Chapter 3.

1.1.5. Parity. Another property of the initial configuration which can be exactly computed is parity, determining if the population size is odd or even. The following protocol uses 4 states $\{L_0, L_1, F_0, F_1\}$. All agents start in state L_1, and the subscript "output bit" for each agent will stabilize to $n \mod 2$:

$$L_i, L_j \to L_{(i+j \mod 2)}, F_{(i+j \mod 2)} \qquad L_i, F_j \to L_i, F_i \qquad (F_i, F_j) \to (F_i, F_i).$$

This does simple leader election, while preserving the parity of the sum of all leader output bits. Thus the ultimate final leader will have the correct output bit (in $\Theta(n)$ time), which is then broadcast to the entire population by the same $\Theta(n)$ time random-walk broadcast from [21].

The majority problem is a representative special case of a **threshold** predicate, which asks whether a particular weighted sum of input states $\sum_{i=1}^{k} w_i X_i > c$ exceeds a constant c. (For majority, $w_1 = 1, w_2 = -1, c = 0$.) This parity problem is a special case of a **mod** predicate, which asks whether a particular weighted sum of input states $\sum_{i=1}^{k} w_i X_i \equiv b \mod m$ (For parity, $w_1 = 1, b = 0, m = 2$). There are constructions which generalizing these last two protocols, and show that any threshold or mod predicate can be stably computed in $O(n)$ time [19, 21]. Taking arbitrary Boolean combinations of threshold and mod predicates gives the class of **semilinear** predicates, which precisely characterize the computational power of population protocols with a constant number of states (independent of population size n). See Section 1.3 for more discussion.

1.2. Formal Model Definitions and Notation

1.2.1. Defining a protocol. A **population protocol** is a pair $\mathcal{P} = (Q, \delta)$, where Q is a set of **states**, and $\delta : Q \times Q \to Q \times Q$ is the **transition function** which describes how agents update their states in an interaction. In general, the transition is **asymmetric**, applied to the ordered pair of an **initiator** agent and **responder** agent. Most transitions in this work are symmetric, so we only explicitly label the initiator and responder for asymmetric transitions where the interaction order is relevant.[5] Furthermore, any asymmetric protocol can be simulated by a symmetric protocol [49], so this distinction does not add fundamental power to the model.

We additionally make use of **random transition functions** that instead output a discrete distribution over pairs of output states. This also does not add fundamental computational power

[5] The same convention is used for specifying protocols in `ppsim`, via the default value of parameter `transition_order = 'symmetric'`.

to the model, because standard "synthetic coin" techniques [5] are able to exploit the randomness of the scheduler to give the agents access to additional random bits, but rigorous analysis becomes more cumbersome than assuming purely random transitions. `ppsim` is able to directly simulate randomized transitions, which is used crucially to faithfully simulate CRNs with varying rate constants.

A **transition** (a.k.a., **reaction**) is written $r_1, r_2 \to p_1, p_2$. For simple protocols, such as the examples in Section 1.1, we directly list all transitions. For every pair of states r_1, r_2 without an explicitly listed transition $r_1, r_2 \to p_1, p_2$, there is an implicit **null** transition $r_1, r_2 \to r_1, r_2$ in which the agents interact but do not change state.

For more complicated protocols in later chapters, we instead rely on pseudocode to implicitly describe the state set Q and transition function δ. We describe states of agents by several **fields**, using fixed-width font to refer to a field such as `field`. Constant values are displayed in a sans serif front such as Yes/No. If an agent has several fields each from a certain set, then that agent's potential set of states is the cross product of all the sets for each field, i.e., adding a field from a set of size k multiplies the number of states by k. A special type of field is called a **role**, used in some of our protocols to optimize space usage. A role is used to **partition** the state space: different roles correspond to different sets of fields, so switching roles amounts to deleting the fields from the previous role. Thus the total number of states is obtained by **adding** the number of states in each role. When two agents a and b interact, we describe the update of each of them using pseudocode, where we refer to `field` of agent $i \in \{a, b\}$ as i.`field`.

1.2.2. Defining a population. A **population** $\mathcal{A}$ is a set of n agents. A configuration $\mathbf{c} : \mathcal{A} \to Q$ is formally a mapping from agents to states. More generally, there could be a **interaction graph** which describes possible ordered pairs of agents which can interact. In this work, we exclusively use the common assumption of the complete interaction graph and **uniform random scheduler** where all $n(n-1)$ ordered pairs of agents are equally likely to be chosen. This **well-mixed** assumption is a modelling choice that is also the basis for standard kinetics of chemical reactions, described in Section 1.2.5. As a result, the agents are indistinguishable and the dynamics of a configuration only depend on the counts of states. Thus we can equivalently define a configuration $\mathbf{c}$ as a multiset over Q. This multiset (as an array of counts) is used internally by `ppsim`, and is crucial for the simulation algorithm.

This uniform scheduler then gives a discrete Markov chain, where at each step a uniform random ordered pair of agents is chosen, and a transition (possibly chosen at random if δ outputs a distribution of pairs of states) is applied. Each interaction uses $\frac{1}{n}$ units of **parallel time**, which we also henceforth refer to as simply **time**. There is a related continuous-time variant [96], where each agent has a rate-1 Poisson clock, upon which it interacts (as the initiator) with a randomly chosen responder agent. In this case, each ordered pair is still equally likely to be chosen next, and the expected time until the next interaction is $\frac{1}{n}$, so up to a re-scaling of time, which by straightforward Chernoff bounds is negligible, these two models are equivalent. `ppsim` can use either time model.

1.2.3. Stable computation. There are many modes of computation considered in population protocols: computing integer-valued functions [32,63,86] where the number of agents in a particular state is the output, Boolean-valued predicates [20,21] where each agent outputs a Boolean value as a function of its state and the goal is for all agents eventually to have the correct output, problems such as leader election [5, 10, 40, 42, 88, 99, 100], and generalizations of predicate computation, where each agent individually outputs a value from a larger range, such as reporting the population size [43, 83, 85]. The computational task and possible input configurations then define a notion of **correct output configurations**. For example, in leader election these are configurations with exactly one agent in a subset $L \subset Q$ of designated leader states. For local computation tasks such as majority or population size counting, there is a mapping $\phi : Q \to O$ of states to output values, and all states in a correct output configuration must have the same correct output value.

To **stably compute** the given task, the population must reach and maintain correct output configurations with probability 1. Stable computation can be equivalently defined solely based on **reachability**, without referencing the precise random dynamics. The possible transitions in the Markov chain on configurations give a directed graph on configurations (this graph describes the **reachability relation**), and this graph only depends on which transitions are possible, not their exact probabilities. A configuration is **stable** if it belongs to a strongly connected component of correct output configurations, i.e., no incorrect configuration is reachable. **Stable computation** then requires that from all reachable configurations, it is possible to reach a stable configuration. Stable computation is **rate-independent**: it does not depend on the exact stochastic dynamics of a kinetic model and is thus a more robust property.

Results on time complexity, however, do depend on the exact dynamics coming from the uniform random scheduler. Given the associated stochastic process, we can define the **stabilization time** as the hitting time required to reach a stable configuration (ie. reach the strongly connected component of correct configurations). The **convergence time** of the stochastic process is the time after which all future configurations in the sequence are correct output configurations. Once a protocol has stabilized, it has also converged, but it is possible to converge before stabilizing if there are still reachable configurations with an incorrect output. As long as the reachability graph is finite, a protocol converges from a configuration $\mathbf{c}$ with probability $p \in [0, 1]$ if and only if it stabilizes from $\mathbf{c}$ with probability p. All existing time lower bounds [5, 6, 32, 88, 156] are lower bounds on stabilization time. At the time of writing, there are still no lower bounds on convergence time for fundamental tasks in this model. And there are examples [36, 120] which show some of these lower bounds do not also hold for convergence time.

1.2.4. Uniformity. A **uniform** protocol [62, 83, 85] is a single set of transitions that can be used in any population size. The original population protocols model [19] was uniform because it had a finite state set Q and transition function δ, independent of the population size n. More recent works have allowed the size of the state set $|Q|$ to grow with n. Many of them [5, 6, 10, 13, 35, 42, 44, 53, 100, 132, 133, 158] are **nonuniform**, where both (Q, δ) depend on the population size n. A uniform protocol whose states grow with the population size n now technically has an infinite state set Q. One formalization is to have a Turing machine that computes the transition function δ [62, 83, 85]. The state bound for such protocols can now be described as a bound on the total number of reachable states, based on an initial configuration of fixed size n, such as in [62]. For most uniform protocols, however, the number of reachable states can be unbounded. Still there is a high probability bound on the number of states that will be reached in a particular execution (for instance, if each agent has an integer counter that will sample a geometric random variable, this will use at most $O(\log n)$ states with high probability).

1.2.5. Discrete chemical reaction networks. In a population protocol, each reaction (transition) consists of two **reactants** (input states) and two **products** (output states). This is a special case of a discrete CRN, which can more generally have an arbitrary multiset of reactant states (a.k.a. **species**) and product states, such as in the reaction $A, B \to 3C$. A configuration $\mathbf{c}$ is still a multiset of states / species, or equivalently a vector of their counts. The same reachability relation from population protocols now generalizes, where a reaction is applicable from a configuration $\mathbf{c}$ if all reactant species are present in sufficient count, which then gives a new configuration by subtracting counts of reactants and adding counts of products. Using this notion of reachability, the same definition of stable computation applies to more general discrete CRNs.

The standard kinetic model for discrete CRNs is a continuous time Markov chain coming from the chemical master equation [104], which is simulated by the Gillespie algorithm [103]. Each reaction has an associated **rate constant** k. Given a fixed volume $v \in \mathbb{R}^+$, the **propensity** of a unimolecular reaction $r : X \overset{k}{\to} \dots$ is $\rho(r) = k \cdot \#X$, where $\#X$ is the count of X. The propensity of a bimolecular reaction $r : X + Y \overset{k}{\to} \dots$ is $\rho(r) = k \cdot \frac{\#X \cdot \#Y}{v}$ if $X \neq Y$ and $k \cdot \frac{\#X \cdot (\#X - 1)}{2v}$ otherwise. The Gillespie algorithm calculates the sum of the propensities of all reactions: $\rho = \sum_r \rho(r)$. The time until the next reaction is sampled as an exponential random variable T with rate ρ, and a reaction r_{next} is chosen with probability $\rho(r_{\text{next}})/\rho$ to be applied.

1.2.6. Notation. $\mathbb{N}$ denotes the set of nonnegative integers. The variable n is exclusively used to refer to the population size. We write $\log n$ to denote $\log_2 n$, and $\ln n$ to denote the natural logarithm $\log_e n$. $H_k = \sum_{i=1}^{k} \frac{1}{i}$ denotes the kth harmonic number, with $H_k \sim \ln k$, where $f(k) \sim g(k)$ denotes that $\lim\limits_{k \to \infty} \frac{f(k)}{g(k)} = 1$. We additional use the standard "big-oh" asymptotic notations $o(n)$, $O(n)$, $\Theta(n)$, $\Omega(n)$, $\omega(n)$. We omit floors or ceilings (which are asymptotically negligible) when writing quantities such as $\ln n$ to describe a quantity that should be integer-valued.

We say event E happens **with high probability** if $\mathbb{P}[\neg E] = O(n^{-c})$, where c is a constant that depends on our choice of parameters in the protocol, where c can be made arbitrarily large by changing the parameters. In other words, the probability of failure can be made an arbitrarily small polynomial. For concreteness, we will write a particular polynomial probability such as $O(n^{-2})$, but in each case we could tune some parameter (say, increasing the time complexity by a constant factor) to increase the polynomial's exponent. We say event E happens **with very high probability** if $\mathbb{P}[\neg E] = O(n^{-\omega(1)})$, i.e., if its probability of failure is smaller than any polynomial probability.

1.3. Population Protocols Literature Review

In this section, we review the key foundational results that have been shown in the population protocols model. See also [26, 128] for surveys on early results, and [11, 94] for surveys on more recent algorithmic techniques.

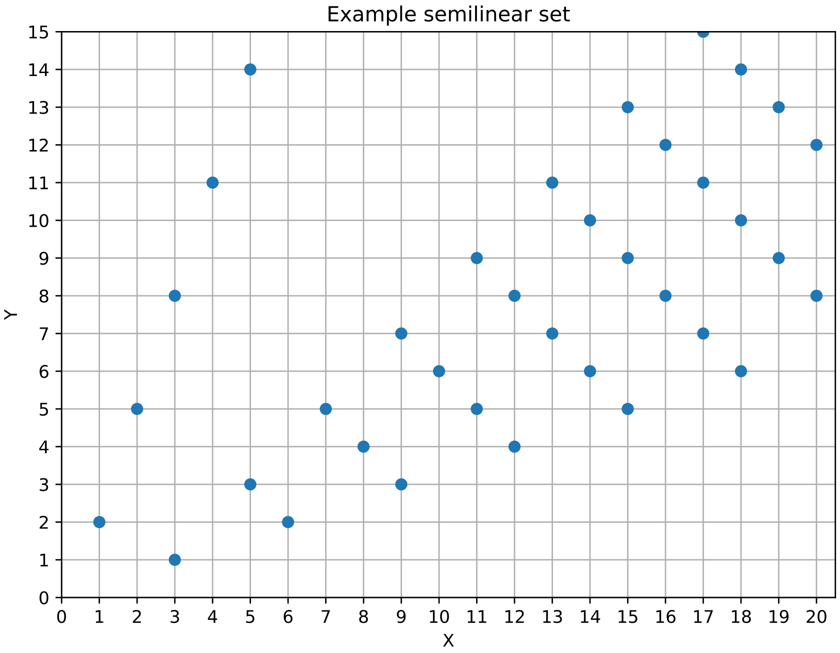


FIGURE 1.5. An example semilinear set $S \subset \mathbb{N}^2$.

1.3.1. Computational power of population protocols. Population protocols were originally defined in [19]. These first works did not focus on issues of time complexity with the uniform scheduler, instead focusing on the the power of stable computation, and exclusively considered constant-state protocols (independent of n). There it was shown that the model could stably compute all semilinear predicates (see Section 1.1.5). A follow-up work [20] then showed any stably computable predicate must also be semilinear.

For example, Fig. 1.5 shows an arbitrary semilinear set, so there is a population protocol which stably computes whether the input configuration (with input states X, Y) is in S. A semilinear set has multiple equivalent definitions. One is as the union of linear sets of the form $L = \{\mathbf{b} + \mathbf{p_1}\mathbb{N} + \ldots + \mathbf{p_k}\mathbb{N}\}$, so the example $S = \{(1,2) + (1,3)\mathbb{N}\} \cup \{(3,1) + (2,2)\mathbb{N} + (3,1)\mathbb{N}\}$. Semilinear sets are also defined as a Boolean combination of threshold and mod sets, as in Section 1.1.5 and Definition 6.2.5. For example, the second linear set that comprises S is the interaction of the threshold sets $\{X - Y \geq 2\}$, $\{X - 3Y \leq 0\}$ and the mod set $\{X + Y \equiv 0 \mod 4\}$. Semilinear sets are also exactly the sets which are definable in Presburger arithmetic [109].

It was also shown the semilinear predicates can still be computed under the model of **stabilizing inputs** [17], where the input field for each agent could repeatedly change before finally stabilizing. This allows for a natural form of composition, since this input could be an output field from an upstream protocol, which will initially be incorrect but eventually stabilize. A stronger notion of **self-stabilization** [79] lets the entire memory of an agent change. These transient faults are modelled by starting the population in an arbitrary configuration, from which they must stabilize to some desired behavior. In the original model, with constant states and a complete interaction graph, it was shown fundamental tasks like **self-stabilizing leader election** were impossible [24]. Our work in Chapter 4 circumvents this by allowing a larger number of states, see Section 4.1.2 for other problem relaxations.

Some of these relaxations considered special interaction graphs, such as rings. An argument from the original paper [19] shows that the complete interaction graph is the hardest case, because it can be simulated in any other connected interaction graph. The agents switch their states in each interaction, which effectively creates a population of "virtual agents" which are getting passed like tokens across the graph. The later focus on time complexity has standardized the uniform random scheduler with a complete interaction graph, motivated by a well-mixed assumption in the modelling of chemical kinetics. Questions about dynamics on arbitrary interaction graphs have been much less studied in general, but the recent work of [12] has initiated a study of fast protocols for general interaction graph dynamics.

These semilinear results also applied to more general discrete CRNs. There, stable computation has focused on functions $f : \mathbb{N}^d \to \mathbb{N}^k$, represented by counts of input and output states (species). Discrete CRNs can stably compute exactly the **semilinear functions**, whose graph is a semilinear set. They can also be defined as piecewise affine functions, see Definition 6.2.6. One natural mode of composition for this discrete CRN function computation is to be **output oblivious**, never using the output states as future inputs, which allows them to be freely used by downstream computation.

Unlike stabilizing inputs for population protocols, however, this creates a proper subclass of the semilinear functions, which we precisely classify in Chapter 6.

1.3.2. Fast population protocols. The work [21] initiated the study of efficient population protocols, with time measured by the uniform random scheduler. They established that all semilinear predicates can be computed in $O(n)$ time. Moreover, they showed more efficient computation was possible starting from a leader. A key technical contribution was the **leader-driven phase clock**. The basic rule is

$$L_i, F_i \to L_{i+1}, F_i \qquad \underset{j<i}{L_i, F_j \to L_i, F_i} \qquad F_i, F_j \to F_{\max(i,j)}, F_{\max(i,j)}.$$

Intuitively, there are leader and follower agents, which have a "minute" field (the subscript). The leader increments the minute when it sees the same minute, and the other rules increase the minute by epidemic. This epidemic must bring most of the population up to the same minute before the leader is likely to see an agent in the same minute, which keeps the population synchronized. This clock can then use a finite state set $\{L_0, \dots, L_{m-1}, F_0, \dots, F_{m-1}\}$, with the minutes wrapping around mod m.[6] Results from [21] show the population will stay synchronized[7] within a constant number of consecutive minutes for polynomial amount of time, with high probability. So for a sufficiently large constant value m of total minutes, we can partition them into "hours" (consecutive blocks of minutes) and the entire population will be synchronized in the same hour.

This original phase clock was used in [21] to simulate a register machine, correct with high probability, whose operations take $O(\operatorname{polylog} n)$ time. For semilinear predicates, this fast computation can be combined with a **stable backup** (the $O(n)$ time stable protocol). The initial leader can then set a timer which takes $\omega(n)$ time[8] , after which all agents will use switch to the results of the slow backup. Note that this technique yields $O(\operatorname{polylog} n)$ **convergence time**, because with high probability the original fast computation is correct, and by the time the agents switch to the slow backup, that is also correct. However, it has $\Omega(n)$ **stabilization time**, because until the slow

[6] Now to determine $\max(i,j)$, we use the non-transitive "cyclic order" where $i \leq j$ means $(j - i \mod m) < \frac{m}{2}$.

[7] This requires the population to be initialized in the same minute, however. The rule is not self-stabilizing, and cannot recover from configurations where all minutes are simultaneously present.

[8] One method is to have the leader mark a single other agent, and wait to encounter that agent multiple times. Note the termination lemma of [83] implies that such a timer is not possible without a leader. With constant states, all states with the signal that the timer has finished will be produced in constant time. The initial leader is necessary to delay this timer signal.

backup has stabilized, it is possible for the agents to switch early to using this backup and become incorrect.

1.3.3. Lower bounds. The work of [21] left open the question of more efficiently electing a single leader. It was then shown in [88] that any stable leader election protocol (in the original model with constant states) must take $\Omega(n)$ time to stabilize, so the simple protocol of Section 1.1.1 is asymptotically optimal. The arguments were later generalized [32] to show a wide variety of problems have $\Omega(n)$ lower bounds on stabilization time, including most semilinear predicates such as majority and parity, as well as functions like $f(x) = \lfloor \frac{x}{2} \rfloor$ computed by the discrete CRN $2X \to Y$. These techniques were also extended in [5] to show the time lower bound for leader election still holds if the number of states is $\leq \frac{1}{2} \log\log n$.

The bound of $\Theta(\log\log n)$ states comes from generalizing the **density lemma**, the key first step in the lower bound argument which requires showing all reachable states are present in large count. The argument starts in an initial configuration where all input states $\in Q_0$ have a large count, $\geq cn$ for some constant c. We then consider all transitions between states in Q_0, letting the set Q_1 be all new states produced this way. The probability of each of these transitions is at least c^2, and it follows that each new state in Q_1 appears in at least count $\approx c^2 n$. We then repeat this argument, where the repeated squaring shows that all states at level Q_k are produced in count $\approx c^{2^k} n$. The state bound then implies $k \leq \frac{1}{2} \log\log n$, and all states become present in sufficiently large count for the remainder of the proof.

These same ideas were also used to show **impossibility of termination** in uniform, leaderless population protocols [83]. There, the uniform protocol has a potentially infinite number of states, but some subset of states are terminated (ie. have some flag `terminated` = `True`, which corresponds to choosing an output value or proceeding to a later part of the algorithm). As above, all states in the initial configuration must be present in large count (so there can not be an initial leader). In that case, the terminated states are produced in constant $O(1)$ time and in large $\Omega(n)$ count. Thus, the only way the delay the production of a termination signal is to either have a nonuniform protocol where the agents can agree on some value related to the population size n, or to start with a state in small count (ie. a leader or more generally a **junta** of leaders). This argument is very useful for showing that potential ideas for fast and stable algorithms cannot work.

1.3.4. Protocols for leader election. With a larger number of states, it does become possible to solve stable leader election in sublinear time. The first provably correct protocol of [10] used $\mathrm{polylog}(n)$ time and $\mathrm{polylog}(n)$ states. The idea of their "leader-minon" protocol can be described by transitions

$$\underset{x=\max(i,j)}{L_i, L_j \to L_{x+1}, F_{x+1}} \qquad \underset{j\leq i}{L_i, F_j \to L_{i+1}, F_{i+1}} \qquad \underset{j>i}{L_i, F_j \to F_j, F_j} \qquad \underset{x=\max(i,j)}{F_i, F_j \to F_x, F_x}.$$

Agents all start in state L_0, as leaders with value 0. In addition to doing simple leader election, leaders increment their value at each interaction. The maximum value then spreads among all followers, and leaders drop out if they see a larger value, knowing for certain that there exists another leader.

As written, this protocol is uniform, but agents will use an unbounded number of states. The analysis from [10] showed that there will be a unique leader with high probability within time $O(\log^3 n)$, when the maximum counter value is still $O(\log^3 n)$. This it suffices to add a bound $m = \Theta(\log^3 n)$ to all counters to bound the state space. This protocol is now nonuniform, since the transitions depend on n, although in the weakest possible way, since the nonuniformity is just a bound on the reachable states of what would otherwise be a uniform protocol. A follow-up protocol from [42] added synthetic coins to this leader-minion logic, now only needing $O(\log n)$ states and $O(\log^2 n)$ time.

Following the $\Omega(\log\log n)$ state lower bound, a breakthrough result [99] showed that using only $O(\log\log n)$ states, stable leader election was possible in time $O(\log^2 n)$. These super-constant states were used for stable **junta election**, selecting a subset of $O(\sqrt{n})$ junta agents. The idea behind junta election is for every agent to start "active" with $\texttt{level} = 0$. If an active agent interacts with another active agent at their level or higher, it increments its level and remains active. Otherwise, the active agent becomes inactive. The maximum level achieved will be $O(\log\log n)$, because the fraction of active agents at each level decays by the same repeated squaring process. Letting a_i denote the fraction of agents who ever reach level i or above, then each of those agents must have met another agent at level i or above in their first interaction after reaching level i. Thus the expected fraction to reach level $i+1$ is $(a_i)^2$. This gives the same doubly-exponential decay that was seen in the $\Omega(\log\log n)$ state lower bound. The inactive agents will then spread the maximum level by epidemic, and the agents that themselves reached the maximum level (ie. never heard above a larger level than the one they reached) will be in the junta. The number of agents that reach the maximum level is $O(\sqrt{n})$, because if $\omega(\sqrt{n})$ agents are at the same level, it is likely that a pair of them will interact and increase to a new higher level.

This $O(\sqrt{n})$ sized junta is sufficient to drive the leader-driven phase clock from [21], which only actually needed the number of leaders to be $O(n^\epsilon)$ for some fixed $\epsilon < 1$. Whenever the agents increment their level, they reset this phase clock. Thus the clock corresponding to the maximum level will only ever have a small $O(\sqrt{n})$ sized junta and start with all agents at minute 0, so it will stay synchronized with high probability.

Using the synchronization from this phase clock, the protocol will then reduce the size of the junta to a single remaining leader. In each cycle of the clock, all junta agents flip a coin and broadcast the result of the flip. If an agent flips tails and sees that another agent flips heads, it drops out. This causes half the junta to drop out in each clock cycle (which takes $O(\log n)$ time). Thus we get down to a single leader in $O(\log n)$ cycles which takes $O(\log^2 n)$ time.

Crucially, the protocol must always stabilize to a unique leader, in expected time $O(\log^2 n)$. It is possible, with low probability, for the phase clock to get out of sync, and for the last remaining leader to flip tails, but drop by out by meeting a delayed agent that has not interacted since the previous round and is still broadcasting that the leader flipped heads. To handle this, the agents will run the simple 2-state leader election as a stable backup, and use this value of the true output. If all junta agents drop out, then the phase clocks will stop advancing, and the protocol will stabilize when the slow backup stabilizes. To stabilize with high probability in $O(\log^2 n)$ time, the protocol of [99] uses a second junta-driven phase clock, whose interactions are only executed once per cycle of the first junta-driven phase clock. Since the first clock takes $O(\log n)$ time per cycle, the second clock will take $O(\log^2 n)$ time per cycle. Then when the second clock finishes, there will be a unique leader with high probability, which sends out a termination signal which tells the population to stop using the slow backup as their output value.

The protocol from [99] is uniform. There is no bound on `level`, so it uses unbounded states, but with high probability will only use $O(\log \log n)$ states because this is the largest level that is reached. The later result of [40] improved the expected stabilization time to the optimal value $O(\log n)$[9],

[9] The coupon-collector argument that it takes $O(\log n)$ time for all agents to interact is not technically enough for a lower bound here, since in principle the initial state could be a non-leader and a unique leader could be elected before all agents have finished interacting. A more involved proof in [156] shows this is not possible, and that even with unbounded states, leader election needs $\Omega(\log n)$ time to stabilize.

while still using $O(\log\log n)$ states. This time and space optimal protocol is crucially nonuniform. The nonuniform variant of junta election uses a maximum level value $m = \Theta(\log\log n)$, and defines the junta to be all agents whose level reaches the maximum value. This allows the agents to "forget" their level value after the junta election step in order to use less space. Later phases are then able to also use $\Theta(\log\log n)$ states, for example to count how many rounds of coin-flip elimination is required to bring the count of leaders from $O(\log n)$ down to $O(1)$.

If the conditions from the $\Omega(n)$ time lower bound on stabilization time for leader election are relaxed, it seems that sublinear time does become possible using $O(1)$ states. The methods from [120] shows an $O(1)$-space and expected $O(\log^2 n)$-time protocol leader election protocol, that has a nonzero probability of error. They also give a protocol with is correct with probability 1, and takes sublinear (e.g., $\sqrt{n}$) expected **convergence time**, although it provably must take longer to then stabilize.

1.3.5. Protocols for majority. The other most studied problem in population protocols is majority. The approximate majority protocol, discussed in Section 1.1.3 and analyzed in [22, 69], has optimal $O(\log n)$ time, but is not correct with probability 1, and is only even correct with high probability when the gap between initial opinions is $\Omega(\sqrt{n \log n})$.

The exact majority problem was solved in the original paper [19] as a special case of a semilinear predicate, but can also be solved by a 4-state protocol [89, 127], which when optimized takes $O(n)$ time, as discussed in Section 1.1.4. Variants of this protocol, also able to distinguish the case of a tie, were discussed in [47].

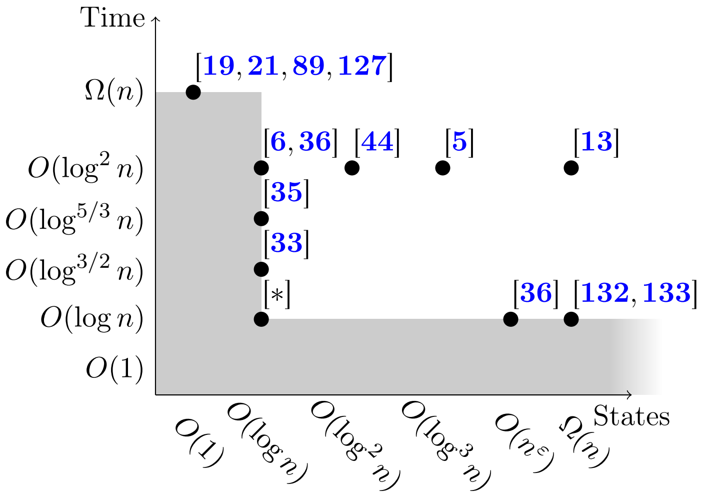
The time lower bounds discussed above showed that $o(n)$-time majority must use $\Omega(\log\log n)$ states. For the special case of majority, these techniques were adapted to show there must be at least $\Omega(\log n)$ states to stabilize in sublinear time.[10] Table 1.1 summarizes the state space and run-time of existing stable majority protocols.

The simplest rule which is able to efficiently solve exact majority is **discrete averaging**, where the agents' states are integers which update as

$$i, j \to \left\lfloor \frac{i+j}{2} \right\rfloor, \left\lceil \frac{i+j}{2} \right\rceil.$$

[10] Technically the result is for stabilizing in $o(n/\operatorname{polylog} n)$ time, and the techniques require the protocol to satisfy two additional conditions (satisfied by all known stable majority protocols, including ours) **monotonicity** and **output dominance**. These concepts are discussed in Section 3.8.

Table 1.1. Summary of known work on the stable exact majority problem in population protocols, including Chapter 3 [∗]. Gray regions are provably impossible: $o(\log\log n)$ state, $o(n)$ time unconditionally [5], $o(\log n)$ state, $O(n^{1-\varepsilon})$ time for monotone, output-dominant protocols [6], and $o(\log n)$ time unconditionally.



This process was shown [39, 135] to converge in $O(\log n)$ time to a configuration where all agents have at most 3 consecutive integer values. If we then have a nonuniform protocol where the initial state A maps to a value $m \geq 2n$, and the initial state B maps to the value $-m$, then if the majority state is A (resp. B), the population average will be ≥ 2 (resp. ≤ -2) which can be stably detected by all agents having the same sign. Also, this detects ties, which are the unique situation that stabilizes to a configuration containing a majority of agents in state 0. This approach was used in [132, 133], and while time optimal, requires a very large number $\Omega(n)$ states.

Most other efficient majority protocols use alternating phases of **cancelling** (two biased agents with opposite opinions both become unbiased, preserving the difference between the majority and minority counts) and **splitting** (a.k.a. **doubling**): a biased agent converts an unbiased agent to its opinion; if all biased agents that didn't cancel can successfully split in that phase, then the count difference doubles. The goal is to increase the count difference until it is n; i.e., all agents have the majority opinion. If each of these phases is an $O(\log n)$ time round of a phase clock, this will give an $O(\log^2 n)$ time protocol. This phase clock could come from the junta-driven phase clock from [99], which was the approach of [36].

Another phase clock scheme, introduced for the majority problem in [6], is based on the **power of two choices** load balancing scheme [130]. The basic idea is to update the minutes by the rule

$$i, j \xrightarrow[i \leq j]{} i + 1, j$$

ie. the smaller[11] of the two minutes increments. Analysis from [137] shows all minutes remain concentrated in an interval of $O(\log n)$, so with $\Omega(\log n)$ total minutes, they can be partitioned into disjoint hours, and all agents will stay synchronized in the same hour with high probability.

Other results dropped the stabilization time to $O(\log^{5/3} n)$ [35] and then $O(\log^{3/2} n)$ [33], by compressing the lengths of the some of these phases. Our optimal $O(\log n)$ time result in Chapter 3 could then be interpreted as compressing the length of all phases down to $O(1)$ time. Our synchronization scheme uses a new phase clock (see Fig. 3.2 and Section 3.5.1) with a similar distribution to the power of two choices phase clock. Our $O(1)$ time hours are now not long enough to ensure perfect synchronization, so substantial analysis is required to show the effects of the loose synchronization provided by this clock.

As with leader election, by relaxing the stabilization requirements, the $\Omega(\log n)$ state lower bound no longer holds, and majority can be solved efficiently with a smaller number of states. Results in [36, 120] show that with fewer states, it is still possible to have $O(\operatorname{polylog} n)$ time if the protocol is only correct with high probability. Also, these fast error prone protocols can be composed with slow backups to get probability 1 correctness, with sublinear convergence time, although still $\Omega(n)$ stabilization time as required by the lower bound of [6].

1.3.6. Other problems.

One generalization of the majority problem is **plurality consensus**, where there are now a larger number of initial opinions, which was studied in [98], and more widely studied in related synchronous gossip models [37, 101].

Applications to engineered chemical reaction networks have motivated additional definitions of **leak reactions** [9], which are spurious events that change the state of an agent, modelling side effects of chemical implementation [162]. The only states which are not affected by leaks are **catalysts**, which can only appear in catalytic reactions such as $A, B \to A, C$ where state A does not change. The majority problem has been considered for comparing two inputs which are catalytic states. It was shown in [16] that the 3-state approximate majority protocol can be adapted to these catalytic inputs, and no fast $O(\log n)$ time protocol can reliably distinguish catalytic inputs if the difference in their counts is $o(\sqrt{n})$. The work of [14] considered catalytic inputs with very

[11] Again, when minutes wrap around mod m, we use the non-transitive "cyclic order" where $i \leq j$ means $(j - i \mod m) < \frac{m}{2}$.

small count, and showed how to amplify their count to detect the difference. In addition to being leak-robust, this work is also self-stabilizing, able to recover from changes in this input counts.

Detection The basic rule for self-stabilization signal amplification was developed in [9], where it was used in the context of the **detection** problem. This problem of detecting the small presence of a signal has clear biological application, and the protocol of [9] has desirable properties of being both leak-robust and self-stabilizing. Moreover, this self-stabilizing rule has proven to be applicable in other contexts, such as generalizing to majority in [14] above and being used in our self-stabilizing subprotocol Propagate-Reset, a key part of our efficient solutions to self-stabilizing leader election in Chapter 4.[12]

The detection problem was also considered in the work of [93]. There, the signal to be detected was used to drive a modified version of the rock-paper-scissors oscillator (the basic 3-state oscillator is shown in Fig. 2.3). The oscillatory dynamics will persist in the presence of a small signal, but otherwise die out, leading to a solution to the detection problem. Moreover, this oscillator was used in [120] to be the basis for a phase clock which gives efficient constant-state protocols.

Size counting Once the number of states grows with the population size, one new problem that becomes possible is population size counting. A simple protocol for **exact size counting** is given by

$$L_i, L_j \to L_{i+j}, F_{i+j} \qquad \underset{m=\max(i,j)}{F_i, F_j \to F_m, F_m}.$$

All agents start in state L_1, which intuitively can be thought of as leaders holding one token. The tokens all get given to remaining leaders during the simple leader election process of Section 1.1.1, which will eventually leave one leader in state L_n. The maximum number of tokens spreads by epidemic, so the rest of the population stabilizes in state F_n. This process takes $\Theta(n)$ time, since the epidemic is negligible compared to the leader election.

Exact size counting necessarily requires $\Omega(n)$ states, but one smaller state variant is **approximate size counting**, requiring only an estimate of n. One simple approach to do this is to restrict

[12] Note that self-stabilizing leader election as considered in Chapter 4 has the stricter requirement of stable computation, where the eventually output must be correct and never change. The detection protocols do not satisfy such strict conditions on the output. A similar notion of **loose stabilization** was applied to leader election [157], which only requires the output to be correct for a sufficiently long period of time. This gets around impossibility results, and can be done with fewer states. See Section 4.1.2 for more discussion of variants of self-stabilizing leader election.

the count of allowable tokens to perfect powers of two, with new rules

$$L_{2^i}, L_{2^i} \to L_{2^{i+1}}, F_{2^{i+1}} \qquad\qquad \underset{m=\max(i,j)}{F_{2^i}, F_{2^j} \to F_{2^m}, F_{2^m}}.$$

This will stabilize to a configuration with one agent in state L_{2^i} for each i giving the position of a 1 in the binary expansion of n, and all other agents in state F_{2^k}, where $k = \lfloor \log_2 n \rfloor$. For more detailed analysis, see Section 3.7.2.

Recent work has considered more time efficient protocols for both exact and approximate size counting [43, 83, 85], where one of the most important building blocks to faster protocols is the discrete averaging process. Another well-studied variant of counting uses an initial leader (also called "base-station"). Given an initialized leader, it is possible to solve self-stabilizing counting, where the initial states of all other agents are arbitrary [25, 29, 31, 114].

For approximate size counting, notice that this function $f : \mathbb{N} \to \mathbb{N}$ given by $f(n) = \lfloor \log_2 n \rfloor$ is no longer semilinear. The simple approximate size counting protocol shows more functions can be computed using a non-constant state bound of $O(\log n)$ states.[13] The work of [62] formalized stable computation with uniform, super-constant state protocols.[14] They showed this threshold $O(\log n)$ states is tight: if the number of reachable states grows with n, it must grow at least as $\Omega(\log n)$, so the only smaller reachable state bounds are constant, computing only semilinear predicates. They also show in [62] that with $\Omega(n)$ states, it is possible to assign unique identifiers, and then simulate a space-bounded Turing machine. See Section 5.3 for more discussion. Our work in Chapter 5 then gives a similar analysis for the case of large internal states but constant size message complexity.

The approximate size counting protocol can be viewed in a separate framework, as instead computing the semilinear threshold predicate $n \geq m$ using $O(\log m)$ states. This connects to a different question about space complexity: what is the most space-efficient way to compute a given semilinear predicate? [45, 46, 71] Now space is not scaling with the population size, it is instead scaling with the complexity of the semilinear predicate. Unfortunately, it is not very clear what a canonical choice of complexity is for semilinear predicates: there can be a doubly exponential difference between the size of a formula in Presburger arithmetic and its corresponding expansion as threshold and mod predicates [109]. Thus a reasonable case study is to focus on just these

[13] Here the state bound is in the strict sense of **reachable** states, where there will be only $O(\log n)$ reachable states from an initial configuration of n agents in state L_{2^0}.

[14] They give a formalization of uniform protocols via linear space Turing machines, see Chapter 5 for more discussion.

threshold predicates, to ask what the most space efficient protocol is for a given threshold predicate. It was shown in [46], using an older construction from vector-additional systems [126], that starting from an additional leader, for some values of m, there are protocols that stably compute $n \geq m$ using only $O(\log \log m)$ states. In other words, with an additional leader, there is a protocol using exponentially fewer states. The recent work of [71] has established some of the first lower bounds on the minimum number of states needed in both the leader and leaderless settings.

1.4. Thesis Contributions

1.4.1. ppsim: A Software Package for Efficiently Simulating Population Protocols.

In Chapter 2, we introduce `ppsim` [141], a software package for efficiently simulating population protocols. In a recent breakthrough, Berenbrink, Hammer, Kaaser, Meyer, Penschuck, and Tran [41] discovered a population protocol simulation algorithm quadratically faster than the naïve algorithm, simulating $\Theta(\sqrt{n})$ reactions in **constant** time (independently of n, though the time scales with the number of species), while preserving the **exact** stochastic dynamics. `ppsim` implements this algorithm, with a tightly optimized Cython implementation that can exactly simulate hundreds of billions of reactions in seconds. It dynamically switches to the CRN Gillespie algorithm for efficiency gains when the number of applicable reactions in a configuration becomes small. As a Python library, `ppsim` also includes many useful tools for data visualization in Jupyter notebooks, allowing robust visualization of time dynamics such as histogram plots at time snapshots and averaging repeated trials.

Finally, we give a framework that takes any CRN with only bimolecular (2 reactant, 2 product) or unimolecular (1 reactant, 1 product) reactions, with arbitrary rate constants, and compiles it into a continuous-time population protocol. This lets `ppsim` exactly sample from the chemical master equation (unlike approximate heuristics such as τ-leaping), while achieving asymptotic gains in running time. In linked Jupyter notebooks, we demonstrate the efficacy of the tool on some protocols of interest in molecular programming, including the approximate majority CRN and CRN models of DNA strand displacement reactions.

1.4.2. A Time and Space Optimal Stable Population Protocol Solving Exact Majority.

In Chapter 3, we describe and analyze a protocol that solves exact majority using $O(\log n)$ states and optimal expected time $O(\log n)$. The number of states $O(\log n)$ is known to be optimal for the class of polylogarithmic time stable protocols that are “output dominant” and “monotone” [6]. These are two natural constraints satisfied by our protocol, making it simultaneously time- and state-optimal for that class. We introduce a key technique called a “fixed resolution clock” to achieve partial synchronization.

Our protocol is nonuniform: the transition function has the value $\lceil \log n \rceil$ encoded in it. We show that the protocol can be modified to be uniform, while increasing the state complexity to $\Theta(\log n \log \log n)$.

1.4.3. Time-Optimal Self-Stabilizing Leader Election in Population Protocols. In Chapter 4, we study **self-stabilizing leader election**, where the population must converge on a single leader agent from **any** possible initial configuration. We initiate the study of time complexity of population protocols solving this problem in its original setting: with probability 1, in a complete interaction graph. The only previously known protocol [54] runs in expected parallel time $\Theta(n^2)$ and uses an optimal count of n states. The existing protocol has the additional property that it becomes silent, i.e., the agents’ states eventually stop changing.

Observing that any silent protocol solving self-stabilizing leader election requires $\Omega(n)$ expected parallel time, we introduce a silent protocol that uses optimal $O(n)$ parallel time and states. Without any silence constraints, we show that it is possible to solve self-stabilizing leader election in asymptotically optimal expected parallel time of $O(\log n)$, but using at least exponential states (a quasipolynomial number of bits). Like the original protocol from [54], all of our protocols work by solving the more difficult **ranking** problem: assigning agents the ranks $1, \dots, n$.

1.4.4. Message Complexity of Population Protocols. The standard population protocol model assumes that when two agents interact, each observes the entire state of the other agent. In Chapter 5, we initiate the study of **message complexity** for population protocols, where the state of an agent is divided into an externally-visible **message** and an internal component, where only the message can be observed by the other agent in an interaction.

We consider the case of $O(1)$ message complexity. When time is unrestricted, we obtain an exact characterization of the stably computable predicates based on the number of internal states $s(n)$: If $s(n) = o(n)$ then the protocol computes semilinear predicates (unlike the original model, which can compute non-semilinear predicates with $s(n) = O(\log n)$), and otherwise it computes a predicate decidable by a nondeterministic $O(n \log s(n))$-space-bounded Turing machine. We then introduce novel $O(\text{polylog}(n))$ expected time protocols for junta/leader election and general purpose broadcast correct with high probability, and approximate and exact population size counting correct with probability 1. Finally, we show that the main constraint on the power of bounded-message-size protocols is the size of the internal states: with unbounded internal states, any computable function can be computed with probability 1 in the limit by a protocol that uses only **1-bit** messages.

1.4.5. Composable Computation in Discrete Chemical Reaction Networks. In Chapter 6, we study the composability of discrete CRNs that stably compute integer-valued functions $f : \mathbb{N}^d \to \mathbb{N}$. We consider **output-oblivious** CRNs in which the output species is never a reactant (input) to any reaction. The class of output-oblivious CRNs is fundamental, appearing in earlier studies of CRN computation, because it is precisely the class of CRNs that can be composed by simply renaming the output of the upstream CRN to match the input of the downstream CRN.

Our main theorem precisely characterizes the functions f that are stably computable by output-oblivious CRNs with an initial leader. The key necessary condition is that for sufficiently large inputs, f is the minimum of a finite number of nondecreasing **quilt-affine** functions. (An affine function is linear with a constant offset; a **quilt-affine** function is linear with a periodic offset).

This work left an open conjecture about classifying the functions computable without an initial leader. A more recent follow-up work from Hashemi, Chugg, and Condon [**111**] built off this theory to exactly characterize this leaderless case.

CHAPTER 2

ppsim: A Software Package for Efficiently Simulating Population Protocols

This chapter is joint work with David Doty. It was originally published as [87].

2.1. Introduction

The tool `ppsim` was built to define, simulate, and visualize population protocols. Its development came after most of the results in later chapters, and was largely motivated by wishing this tool had existed for that earlier research. The basic dynamics of population protocols are simple enough that it is straightforward to write an ad-hoc simulation of a particular protocol: creating an array of n agents and successively sampling a random pair and updating their state. A recent algorithmic breakthrough [41] has shown that a much more efficient simulation is possible, able to update $O(\sqrt{n})$ agents in parallel, while preserving the exact dynamics. This is especially useful for simulating cutting-edge population protocols: they use a relatively small number of states, and have $\mathrm{polylog}(n)$ running time, so the number of interactions to simulate is $n\,\mathrm{polylog}(n)$. And the claims we want to prove about such protocols are often detailed statements about runtime. For example, the protocol Propagate-Reset used in Chapter 4 has a simpler analysis that it takes $O(\log^2 n)$ time, but a sharper analysis shows the true time is $O(\log n)$. Distinguishing between these two values in simulation requires simulating population sizes n that scale across a wide range of orders of magnitude. And for large population sizes such as $n = 10^{12}$, this quadratic speedup is crucial.

Beyond implementing a faster simulator, `ppsim` was built to encourage and enable easier collaboration and sharing of results. It is a package in Python, an easily accessible language that is part of a powerful open source ecosystem for scientific programming. It is designed to run easily in a Jupyter notebook, where code cells can unambiguously describe the exact logic of the protocol. An optimized Cython backend then handles the work of simulating the protocol rules. The simulation data is then available for immediate visualization, creating figures to be used in papers such as this, as well as animations.

`ppsim` also has additional applications beyond the theory of population protocols. This model is also a simple description of chemical reaction networks. For CRNs, the standard continuous-time kinetic model (chemical master equation) is simulated by the Gillespie algorithm (see Section 1.2.5). For any CRN that has only bimolecular (2-input, 2-output) or unimolecular (1-input, 1-output) reactions, with arbitrary rate constants, there is a continuous time population protocol with equivalent dynamics (proven in Theorem 2.6.1). Using these ideas, `ppsim` is able to take descriptions of such CRNs and faithfully simulate the exact dynamics from the chemical master equation. Its asymptotic gains in efficiency mean that in many situations (very large total number of molecules n compared to the number of species / reactions) it is considerably faster than existing CRN simulators based on the Gillespie algorithm (see Fig. 2.2).

2.2. The algorithm

One speedup heuristic for population protocol simulation is to sample the number of each interaction that would result from a random matching of size m, and update species counts in a single step. This, is an inexact approximation: unlike the true process, it prevents any molecule from participating in more than one of the next m interactions, adding a negative dependency which can bias the overall dynamics. The algorithm implemented by `ppsim`, due to Berenbrink, Hammer, Kaaser, Meyer, Penschuck, and Tran [**41**], builds on this heuristic. Conditioned on the event that no molecule is picked twice during the next m interactions, these interacting pairs are a random disjoint matching of the molecules. Define the random variable C as the number of interactions until the same molecule is picked twice. Their basic algorithm samples this collision length C according to its exact distribution, then updates counts in batch assuming all pairs of interacting molecules are disjoint until this collision, and finally simulates the interaction involving the collision. By the Birthday Paradox, $\mathbb{E}C \approx \sqrt{n}$ in a population of n molecules, giving a quadratic factor speedup over the naïve algorithm. The time to update a batch scales quadratically with q, the total number of states. The "multibatch" variant, used by `ppsim`, samples multiple successive collisions to process an even larger batch, and uses $O\left(|Q|\sqrt{\frac{\log n}{n}}\right)$ time per simulated interaction. See [**41**] for details.

Comparing this multibatch simulation strategy to the CRN Gillespie algorithm, there is an important difference: the Gillespie algorithm only simulates non-null interactions (reactions), so spends time proportional to the number of non-null interactions that take place. Consider the simple 2-state leader election from Section 1.1.1, when $\#L = 2$ and $\#F = n - 2$. The multibatch interaction is able to simulate $O(\sqrt{n})$ interactions in one step, but it is very likely these interactions are all null $L, F \to L, F$ or $F, F \to F, F$. A smarter approach for this exact situation is used by the Gillespie algorithm, sampling the time until the next non-null interaction. So here the number of interactions to advance by can be sampled as a geometric random variable with expected value $\binom{n}{2}$, which now simulates $O(n^2)$ interactions in a single step. To better handle cases like this, `ppsim` dynamically switches to a discrete-time Gillespie algorithm when the number of null interactions is sufficiently large. As a bonus, this gives `ppsim` the ability to detect when a configuration is silent, because it calculates that the probability of a non-null interaction is 0. Thus, by default `ppsim` will simulate a protocol until the configuration becomes silent, and gives a fast an easy way to sample this silence time, such as was done in Fig. 2.1b. See documentation [**141**] for implementation details.

During the simulation of 2-state leader election on $n = 10^6$ agents from Fig. 1.1c, `ppsim` is able to use the Gillespie algorithm for the tail when the count of leaders is sufficiently small, and thus very quickly simulate out to time 10^7 (ie. 10^{13} simulated interactions). For another example, for the simple 4-state exact majority simulated in Fig. 1.4a, once the number of active A and B states is $O(1)$, then the probability of a non-null interaction is $O(\frac{1}{n})$, so the Gillespie algorithm is able to simulate $O(n)$ interactions at once. Adding the random-walk variant in Fig. 1.4b means all a, b pairs now have a non-null interaction, so the Gillespie algorithm is no longer efficient even when the count of A and B agents is small. As a result, this random walk variant is considerably slower to simulate all the way until silence time.

Other CRN simulation algorithms. Variants of the Gillespie algorithm reduce the time to apply a single reaction from $O(|R|)$ to $O(\log |R|)$ [**102**] or $O(1)$ [**149**], where $|R|$ is the number of types of reactions. However, the time to apply n reactions still scales with n. A common speedup heuristic for simulating $\omega(1)$ reactions in $O(1)$ time is **τ-leaping** [**55**, **105**, **106**, **143**, **150**], which "leaps" ahead by time τ, by assuming reaction propensities will not change and updating counts in a single batch step by sampling according these propensities. Such methods necessarily approximate the kinetics inexactly, though it is possible in some cases to prove bounds on the approximation accuracy [**150**]. Linear noise approximation (LNA) [**58**] can be used to approximate the discrete kinetics, by adding stochastic noise to an ODE approximation.

2.3. Usage of the ppsim tool

We direct the reader to [141] for detailed installation, usage instructions, and examples. Here we highlight basic usage examples for specifying protocols.

There are three ways one can specify a population protocol, each best suited for different contexts. The most direct specification of a protocol directly encodes the mapping of input state pairs to output state pairs using a Python (the following is the well-studied **approximate majority** protocol, which has been studied theoretically [22, 69] and implemented experimentally with DNA [66]):

```python
a,b,u = 'A','B','U'
approx_majority = {(a,b):(u,u), (a,u):(a,a), (b,u):(b,b)}
```

More complex protocols with many possible species are often specified in pseudocode instead of listing all possible reactions. `ppsim` supports this by allowing the **transition function** mapping input states to output states to be computed by a Python function. The following allows species to be integers and computes an integer average of the two reactants:

```python
def discrete_averaging(s: int, r: int):
    return math.floor((s+r)/2), math.ceil((s+r)/2)
```

States and transition rules are converted to integer arrays for internal Cython methods, so there is no efficiency loss for the ease of representing protocol rules, since a Python function defining the transition function is not called during the simulation: producible states are enumerated before starting the simulation.

For complicated protocols, an advantage of `ppsim` over standard CRN simulators is the ability to represent species/states as Python objects with different fields (as they are often represented in pseudocode), and to plot counts of agents based on their field values.[1]

Finally, protocols can be specified using CRN-like notation for CRNs with reactions that are bimolecular (2-input, 2-output) or unimolecular (1-input, 1-output), with arbitrary rate constants. For instance, this code specifies the CRN

$$A + B \underset{4}{\overset{0.5}{\rightleftharpoons}} 2C, \qquad C \xrightarrow{5} D$$

[1] Download and run `https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/majority.ipynb` to visualize such large state protocols.

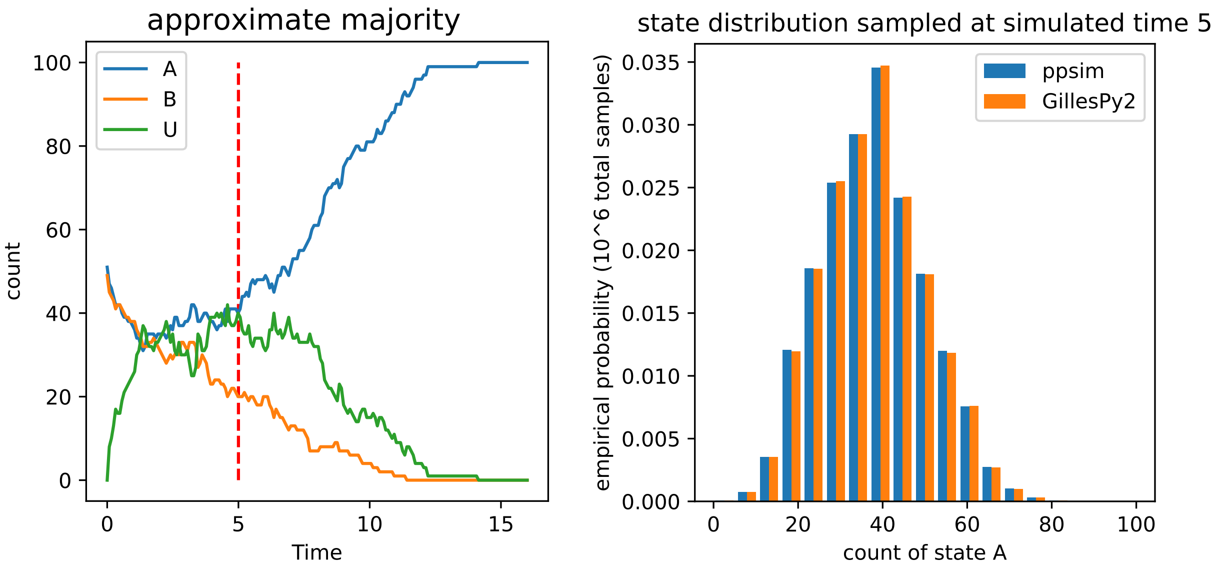
```
a,b,c,d = species('A B C D')
crn = [(a+b | 2*c).k(0.5).r(4), (c >> d).k(5)]
```

This will then get compiled into a continuous time population protocol that samples the same distribution as Gillespie.

Any of the three specifications (`dict`, Python function, or list of CRN reactions) can be passed to the `Simulation` constructor. The `Simulation` can be run to generate a history of sampled configurations.

```
init_config = {a: 51, b: 49}
sim = Simulation(init_config, approx_majority)
sim.run(16, 0.1)   # 160 samples up to time 16
sim.history.plot() # Pandas dataframe with counts
```

This would produce the plot shown in Fig 2.1a. When the input is a CRN, `ppsim` defaults to continuous time and produces the exact same distributions as the Gillespie algorithm. Fig 2.1b shows a test against the package GillesPy2 [**107**] to confirm they sample the same distribution.



(a) Plot of sim.history.

(b) Comparison with Gillespie algorithm.

FIGURE 2.1. Time 5 (dotted line in Fig 2.1a) was sampled 10^6 times with `ppsim` and GillesPy2 to verify they both sample the same chemical master equation distribution (Fig 2.1b).

2.4. Speed comparison with other CRN simulators

We ran speed comparisons of ppsim against both GillesPy2 [107] and StochKit2 [145], the latter being the fastest option we found for Gillespie simulation. Fig 2.2 shows that ppsim is able to reach significantly larger population sizes. Other tests shown in an example notebook[2] show how each package scales with the number of species and reactions.

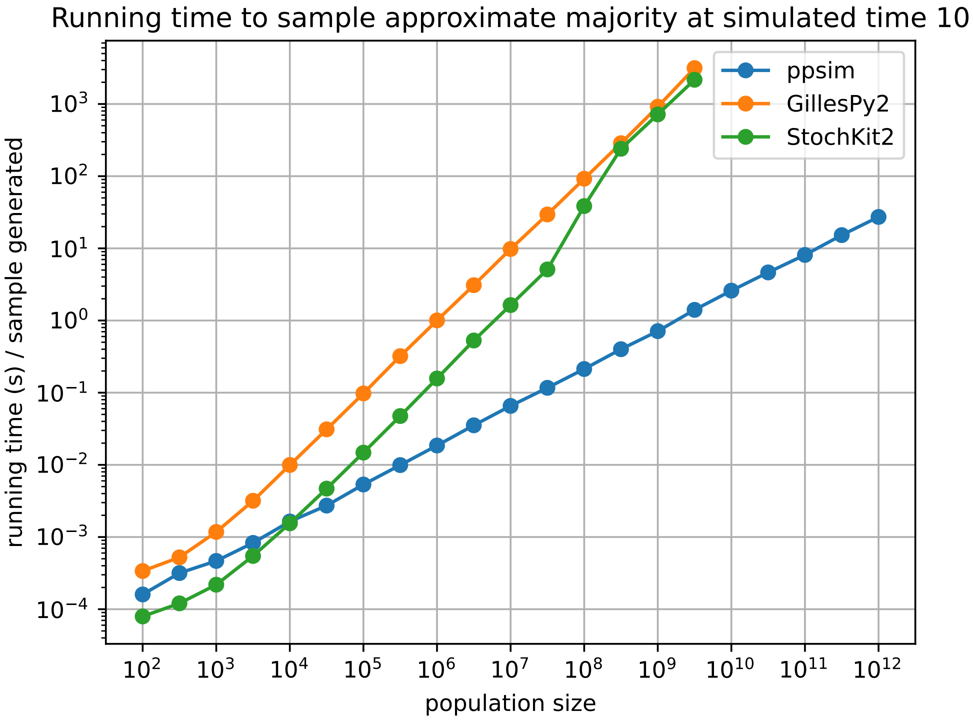
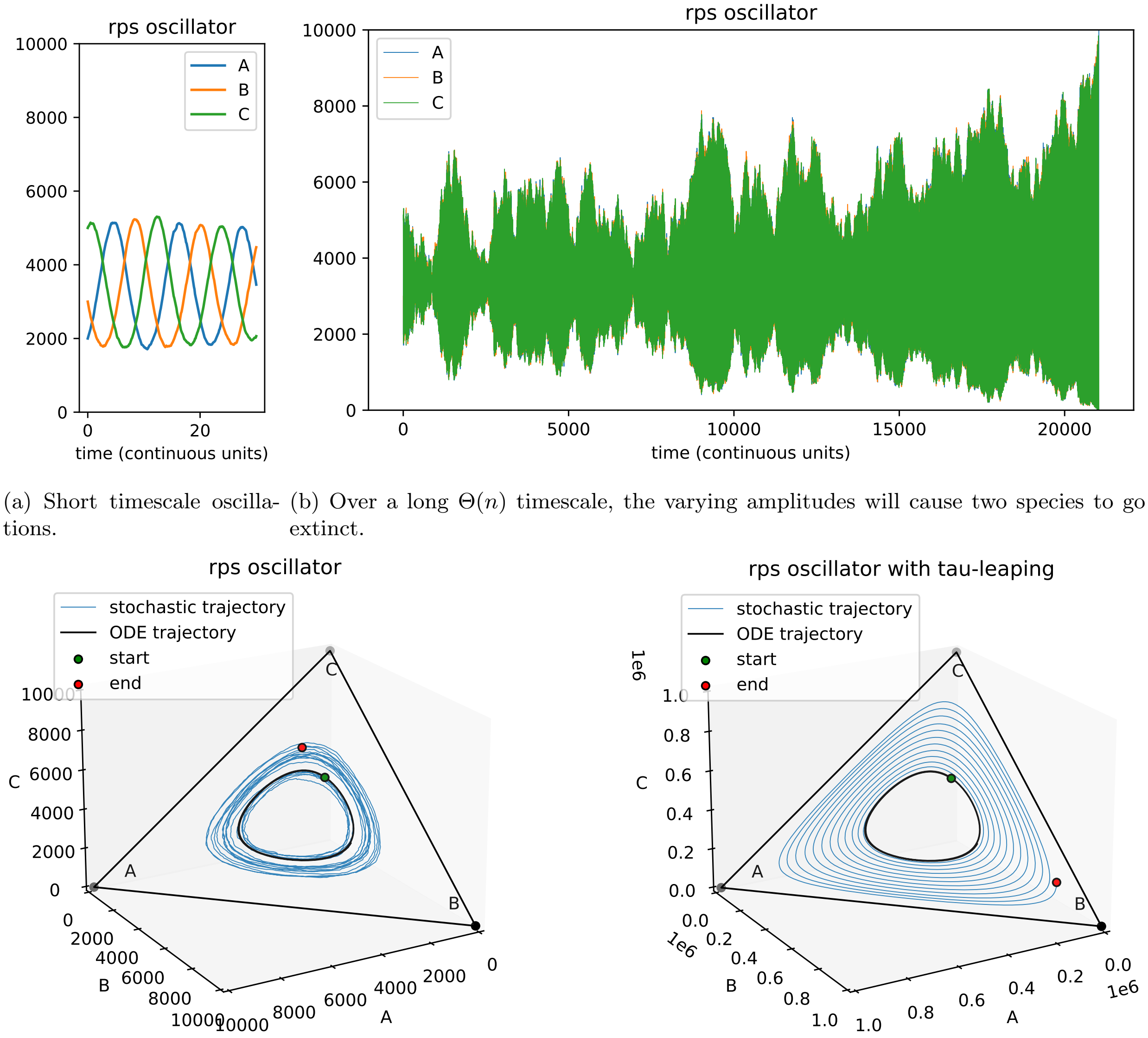


Figure 2.2. Comparing runtime with population size n shows $O(n)$ scaling for Gillespie (slope 1 on log-log plot) versus $O(\sqrt{n})$ scaling for ppsim (slope 1/2).

2.5. Issues with other speedup methods

It is reasonable to conjecture that exact stochastic simulation of large-count systems is unnecessary, since Gillespie is fast enough on small-count systems, and faster ODE approximation is "reasonably accurate" for large-count systems. However, there are example large count systems with stochastic effects not observed in ODE simulation, and where τ-leaping introduces systematic inaccuracies that disrupt the fundamental qualitative behavior of the system, demonstrating the need for exact stochastic simulation. A simple such example is the 3-state rock-paper-scissors oscillator:

[2] https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/crn.ipynb shows further plots and explanations.



(a) Short timescale oscillations.

(b) Over a long $\Theta(n)$ timescale, the varying amplitudes will cause two species to go extinct.

(c) Dynamics from Figs 2.3a, 2.3b in phase space. The ODE solution has a neutrally stable orbit.

(d) τ-leaping adds a consistent outward drift that will lead to extinction on a much shorter timescale.

FIGURE 2.3. The rock-paper-scissors oscillator has qualitative dynamics missed by both ODE simulation (never goes extinct) and τ-leaping (too quickly goes extinct).

$B + A \to 2B,\ C + B \to 2C,\ A + C \to 2A$. Figure 2.3 compares exact simulation of this CRN to τ-leaping and ODEs.

The population protocol literature furnishes more examples, with problems such as leader election [**10**,**40**,**42**,**44**,**53**,**88**,**94**,**99**,**100**,**156**,**159**] and single-molecule detection [**9**,**93**],[3] that crucially

[3] Download and run https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/rps_oscillator.ipynb to see visualizations of the generalized 7-state rps oscillator used for single-molecule detection in [**93**].

use small counts in a very large population, a regime not modelled correctly by ODEs. See also [122] for examples of CRNs with qualitative stochastic behavior not captured by ODEs, yet that behavior appears only in population sizes too large to simulate with Gillespie.

2.6. Full specification of compilation of CRN to population protocol

It is possible to specify 1-reactant/1-product reactions such as $A \to B$, which are compiled into 2-reactant/2-product reactions $A + C \to B + C$ for every species C, with reaction rates adjusted appropriately. The full transformation is described in the proof of Theorem 2.6.1. Here, we give an example of the transformation on the CRN

$$\begin{aligned} 2A &\underset{2}{\overset{3}{\rightleftharpoons}} B + C \\ C &\xrightarrow{1} D \end{aligned}$$

First, each reversible reaction is turned into two irreversible reactions:

$$\begin{aligned} 2A &\xrightarrow{3} B + C \\ B + C &\xrightarrow{2} 2A \\ C &\xrightarrow{1} D \end{aligned}$$

For each non-symmetric bimolecular reaction (with two unequal reactants), add its "swapped" reaction reversing the order of reactants and the order of products. From now on we write reactions using ordered pair notation (e.g., $(A, A) \to (B, C)$ instead of $2A \to B + C$).

$$\begin{aligned} (A, A) &\xrightarrow{3} (B, C) \\ (B, C) &\xrightarrow{2} (A, A) \\ (C, B) &\xrightarrow{2} (A, A) \\ C &\xrightarrow{1} D \end{aligned}$$

Each (originally) bimolecular reaction (not the result of converting a unimolecular reaction below) has its rates multiplied by the corrective factor $(n-1)/(2 \cdot v)$, where n is the population size and v is the volume. We choose $n = v = 10$ for this example, so $(n-1)/(2 \cdot v) = 0.45$. (See proof of Theorem 2.6.1 for explanation of correction factor.)

$$(A, A) \overset{1.35}{\rightarrow} (B, C)$$
$$(B, C) \overset{0.9}{\rightarrow} (A, A)$$
$$(C, B) \overset{0.9}{\rightarrow} (A, A)$$
$$C \overset{1}{\rightarrow} D$$

Each unimolecular reaction is converted to several bimolecular reactions with all other species in the CRN.

$$(A, A) \overset{1.35}{\rightarrow} (B, C)$$
$$(B, C) \overset{0.9}{\rightarrow} (A, A)$$
$$(C, B) \overset{0.9}{\rightarrow} (A, A)$$
$$(C, A) \overset{1}{\rightarrow} (D, A)$$
$$(C, B) \overset{1}{\rightarrow} (D, B)$$
$$(C, C) \overset{1}{\rightarrow} (D, C)$$
$$(C, D) \overset{1}{\rightarrow} (D, D)$$

Finally, for each ordered pair of input states (x, y), sum the rates of all reactions that have ordered reactants (x, y), and we let m be the maximum value of this sum over all ordered pairs of reactants. In this example, the pair (C, B) has rates 0.9 and 1 for its two reactions, whose sum achieves the maximum $m = 1.9$. Divide rates by m to convert them to probabilities.

This gives us the final randomized transitions of the population protocol. Below, whenever the probabilities for a given input state pair (x, y) sum to a value $p < 1$, implicitly the transition on input (x, y) is null (i.e., outputs (x, y)) with probability $1 - p$.

$(A, A) : (B, C)$ with probability 1.35/1.9

$(B, C) : (A, A)$ with probability 0.9/1.9

$(C, B) : \{(A, A)$ with probability 0.9/1.9, $\quad (D, B)$ with probability $1/1.9\}$

$(C, A) : (D, A)$ with probability 1/1.9

$(C, C) : (D, C)$ with probability 1/1.9

$(C, D) : (D, D)$ with probability 1/1.9

Time is now scaled by $m = 1.9$. Thus in one unit of time, there should be an expected $1.9 \cdot n$ interactions. In order to simulate t units of time, we choose a Poisson random variable with mean $1.9 \cdot n \cdot t$ to get the number of interactions to simulate.

The following theorem shows that the above transformation results in a population protocol whose continuous time dynamics exactly sample from the same distribution as the Gillespie stochastic model applied to the original CRN.

THEOREM 2.6.1. *Let $\mathcal{C}$ be a CRN consisting of only unimolecular reactions $r_i : X_{i_1} \xrightarrow{k_i} X_{i_2}$ and bimolecular reactions $r_j : X_{j_1} + X_{j_2} \xrightarrow{k_j} X_{j_3} + X_{j_4}$.*

Then for any initial configuration $\mathcal{I} = \{a_1 X_1, \ldots, a_s X_s\}$ and fixed volume $v \in \mathbb{R}^+$, there exists an equivalent continuous time population protocol $\mathcal{P}$ with time scaling constant m. For any time $t \in \mathbb{R}^+$, the distribution over all possible configurations sampled by the Gillespie algorithm at time t is the same distribution as configurations of $\mathcal{P}$ at time $m \cdot t$.

PROOF. The continuous time population protocol $\mathcal{P}$ will use the same state set $\{X_1, \ldots, X_s\}$ and initial configuration $\mathcal{I}$. In the population protocol dynamics, each agent has a rate 1 Poisson process for the event where they interact with a randomly chosen agent. Thus for each ordered pair of agents, that pair meets in that order as a rate $\frac{1}{n-1}$ Poisson process.

After converting all reactions, we will be left with a set of ordered transitions with rates, of the form $(a, b) \xrightarrow{k} (c, d)$. An ordinary population protocol transition should correspond to rate $k = 1$, so for each ordered pair of agents (v_1, v_2) in states (a, b), this transition should happen as a Poisson process with rate $\frac{k}{n-1}$. We must handle the fact that these rates could exceed 1, and also there could be multiple ordered transitions starting from the same pair (a, b). Define m to be the maximum over all ordered pairs (a, b) of the sum of the rates of any ordered transitions with pair (a, b) on the left. The population protocol transition rule for a pair (a, b) is then a randomized rule, where each ordered reaction $(a, b) \xrightarrow{k} (c, d)$ happens with probability $\frac{k}{m}$ (and otherwise the transition is null). Because we are also scaling time by this factor m, it follows that the rate of this ordered transition between a single pair of agents (v_1, v_2) in states (a, b) will be $\frac{m}{n-1} \cdot \frac{k}{m} = \frac{k}{n-1}$, as desired.

Next we show how each unimolecular reaction $r_i : X_{i_1} \xrightarrow{k_i} X_{i_2}$ is converted to a set of ordered transitions with rates. For each $j = 1, \ldots, s$, we add the ordered transition $(X_{i_1}, X_j) \xrightarrow{k_i} (X_{i_2}, X_j)$. In other words, the first agent in the pair v_1, independent of the state of the other agent, will change state from X_{i_1} to X_{i_2}. This agent v_1 gets chosen as the first agent in the pair as a Poisson process with rate m, and this unimolecular transition will happen (independent of the state of the other agent) with probability $\frac{k_i}{m}$. Thus each agent in state X_{i_1} changes to state X_{i_2} as a rate k_i Poisson process, which is exactly the model simulated by the Gillespie algorithm.

Finally, we show how each bimolecular reaction $r_j : X_{j_1} + X_{j_2} \xrightarrow{k_j} X_{j_3} + X_{j_4}$ is converted to a set of ordered transitions with rates. In the Gillespie model, for each unordered pair $\{v_1, v_2\}$ of agents in states X_{j_1} and X_{j_2}, the time until this reaction happens is an exponential random variable with rate $\frac{k_j}{v}$, where $v \in \mathbb{R}^+$ is the volume. In our protocol $\mathcal{P}$, the time when this unordered pair of agents will interact is an exponential random variable with rate $\frac{2m}{n-1}$. Thus, we multiply each rate by the conversion factor $\frac{n-1}{2v}$ to get $k'_j = k_j \cdot \frac{n-1}{2v}$. Then we add the ordered transition $(X_{j_1}, X_{j_2}) \xrightarrow{k'_j} (X_{j_3}, X_{j_4})$, and if $X_{j_1} \neq X_{j_2}$, also the reverse ordered transition $(X_{j_2}, X_{j_1}) \xrightarrow{k'_j} (X_{j_4}, X_{j_3})$. As a result, each unordered pair $\{v_1, v_2\}$ will interact with rate $\frac{2m}{n-1}$, then do this transition with probability $\frac{k'_j}{m}$. This gives the reaction a total rate of $k'_j \cdot \frac{2}{n-1} = \frac{k_j}{v}$, as desired.

□

2.7. Conclusion

Unfortunately, the algorithm of Berenbrink et al. [41] implemented by `ppsim` seems inherently suited to population protocols, not more general CRNs. For instance, reversible dimerization reactions $A + B \rightleftharpoons C$ (used, for example, in [153] to model toehold occlusion reactions in DNA systems) seem beyond the reach of the batching technique of [41]. Although such reactions can be **approximated** by $A + B \rightleftharpoons C + F$ for some anonymous "fuel" species F, the count of F influences the rate of the reverse reaction $F + C \to A + B$, with a different rate than $C \to A + B$.

Another area for improvement is the handling of null reactions. There could be a way to more deeply intertwine the logic of the Gillespie and batching algorithms, to gain the simultaneous benefits of each, skipping the null reactions while simulating many non-null reactions in batch.

CHAPTER 3

A Time and Space Optimal Stable Population Protocol Solving Exact Majority

This chapter is joint work with David Doty, Mahsa Eftekhari, Leszek Gąsieniec, Grzegorz Stachowiak, and Przemyslaw Uznański. It was originally published as [**84**].

3.1. Introduction

We show a stable population protocol solving the exact majority problem in optimal $O(\log n)$ time (in expectation and with high probability) that uses $O(\log n)$ states. Our protocol is both monotone and output dominant (see Section 3.8 or [**6**] for discussion of these definitions), so by the $\Omega(\log n)$ state lower bound of [**6**], our protocol is both time and space optimal for the class of monotone, output-dominant stable protocols. We define the slightly generalized problem that requires recognizing when there is a tie, so the range of outputs is $\{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$.

A high-level overview of the algorithm is given in Sections 3.2.1 and 3.2.2, with a full formal description given in Section 3.2.4. Like most known majority protocols using more than constant space (the only exceptions being in [**36**]), our protocol is **nonuniform**: agents have an estimate of the value $\lceil \log n \rceil$ embedded in the transition function and state space. Section 3.7 describes how to modify our main protocol to make it uniform, retaining the $O(\log n)$ time bound, but increasing the state complexity to $O(\log n \log \log n)$ in expectation and with high probability. That section discusses challenges in creating a uniform $O(\log n)$ state protocol.

3.2. Nonuniform majority algorithm description

The goal of Sections 3.2 through 3.6 is to show the following theorem:

Theorem 3.2.1. *There is a nonuniform population protocol* Nonuniform Majority*, using $O(\log n)$ states, that stably computes majority in $O(\log n)$ stabilization time, both in expectation and with high probability.*

3.2.1. High-level overview of algorithm. In this overview we use "pseudo-transitions" such as $A, B \to \mathcal{O}, \mathcal{O}$ to describe agents updating a portion of their states, while ignoring other parts of the state space.

Each agent initially has a `bias`: $+1$ for opinion A and -1 for opinion B, so the population-wide sum $g = \sum_v v.\texttt{bias}$ gives the **gap** between opinions. The majority problem is equivalent to determining $\text{sign}(g)$. Transitions redistribute biases among agents but, to ensure correctness, maintain the population-wide g as an invariant. Biases change through **cancel reactions** $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ and **split reactions** $\pm\frac{1}{2^i}, 0 \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}}$, down to a minimum $\pm\frac{1}{2^L}$. The constant $L = \lceil \log_2(n) \rceil$ ensures $\Theta(\log n)$ possible states.

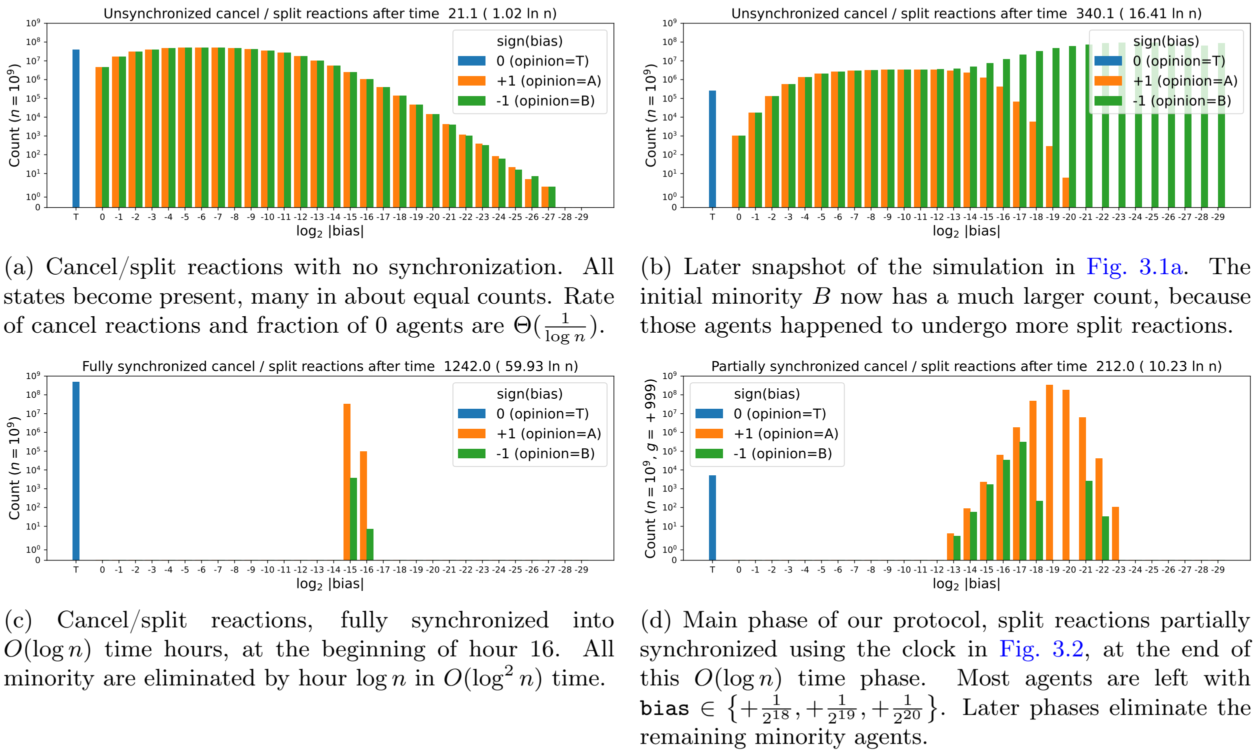
The cancel and split reactions average the power-of-2 bias between both agents, but only when the average is also a power of 2, or 0. If we had averaging reactions between all pairs of biases (also allowing, e.g., $\frac{1}{2}, \frac{1}{4} \to \frac{3}{8}, \frac{3}{8}$), then all biases would converge to $\frac{g}{n}$, but this would use too many states.[1] With our limited set $\{0, \pm\frac{1}{2}, \pm\frac{1}{4}, \dots, \pm\frac{1}{2^L}\}$ of possible biases, allowing all cancel and split reactions simultaneously does not work. Most biases become present simultaneously, which slows the rate of cancel reactions, and the count of unbiased 0 agents is reduced, which slows the rate of split reactions, see Fig. 3.1a. Also, there is a non-negligible probability for the initial minority opinion to reach a much greater count, if those agents happen to do more split reactions, see Fig. 3.1b. Thus using only the count of positive versus negative biases will not work to solve majority even with high probability.

To solve this problem, we partially synchronize the unbiased agents with a field `hour`, adding $\log n$ states $0_0, 0_1, 0_2, \dots, 0_L$. The new split reactions

$$\pm\frac{1}{2^i}, 0_h \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}} \qquad \text{if } h > i$$

will wait until $\texttt{hour} \geq h$ before doing splits down to $\texttt{bias} = \pm\frac{1}{2^h}$. We could use existing phase clocks to perfectly synchronize `hour`, by making each hour use $\Theta(\log n)$ time, enough time for all opinionated agents to split. Then WHP all agents would be in states $\{0_h, +\frac{1}{2^h}, -\frac{1}{2^h}\}$ by the end of hour h, see Fig. 3.1c. The invariant $g = \sum_v v.\texttt{bias}$ implies that all minority opinions would be eliminated by hour $\lceil \log_2 \frac{1}{g} \rceil \leq L$. This would give an $O(\log n)$-state, $O(\log^2 n)$-time majority algorithm, essentially equivalent to [6, 36].

[1] This was effectively the approach used for majority in [133], for an $O(n)$ state, $O(\log n)$ time protocol.


Unsynchronized cancel / split reactions after time 21.1 (1.02 ln n)

Unsynchronized cancel / split reactions after time 340.1 (16.41 ln n)

Fully synchronized cancel / split reactions after time 1242.0 (59.93 ln n)

Partially synchronized cancel / split reactions after time 212.0 (10.23 ln n)

(a) Cancel/split reactions with no synchronization. All states become present, many in about equal counts. Rate of cancel reactions and fraction of 0 agents are $\Theta(\frac{1}{\log n})$.

(b) Later snapshot of the simulation in Fig. 3.1a. The initial minority B now has a much larger count, because those agents happened to undergo more split reactions.

(c) Cancel/split reactions, fully synchronized into $O(\log n)$ time hours, at the beginning of hour 16. All minority are eliminated by hour $\log n$ in $O(\log^2 n)$ time.

(d) Main phase of our protocol, split reactions partially synchronized using the clock in Fig. 3.2, at the end of this $O(\log n)$ time phase. Most agents are left with `bias` $\in \left\{+\frac{1}{2^{18}}, +\frac{1}{2^{19}}, +\frac{1}{2^{20}}\right\}$. Later phases eliminate the remaining minority agents.

FIGURE 3.1. Cancel / split reactions with no synchronization (3.1a,3.1b), perfect synchronization (3.1c), and partial synchronization (3.1d) via the fixed-resolution phase clock of our main protocol. Plots generated from [1].

The main idea of our algorithm is to use these rules with a faster clock using only $O(1)$ time per hour. The `hour` field of unbiased agents is synchronized to a separate subpopulation of clock agents, who use a field `minute`, with k consecutive minutes per hour. Minutes advance by **drip reactions** $C_i, C_i \to C_i, C_{i+1}$, and catch up by **epidemic reactions** $C_i, C_j \to C_{\max(i,j)}, C_{\max(i,j)}$. See Fig. 3.2 for an illustration of the clock `minute` and `hour` dynamics.

Since $O(1)$ time per hour is not sufficient to bring all agents up to the current hour before advancing to the next, we now have only a large constant fraction of agents, rather than all agents, synchronized in the current hour. Still, we prove this looser synchronization keeps the values of `hour` and `bias` relatively concentrated, so by the end of this phase, we reach a configuration as shown in Fig. 3.1d. Most agents have the majority opinion (WLOG positive), with three consecutive biases $+\frac{1}{2^l}, +\frac{1}{2^{l+1}}, +\frac{1}{2^{l+2}}$.

Detecting ties. This algorithm gives an elegant way to detect a tie with high probability. In this case, $g = 0$, and with high probability, all agents will finish the phase with `bias` $\in \left\{0, \pm\frac{1}{2^L}\right\}$.

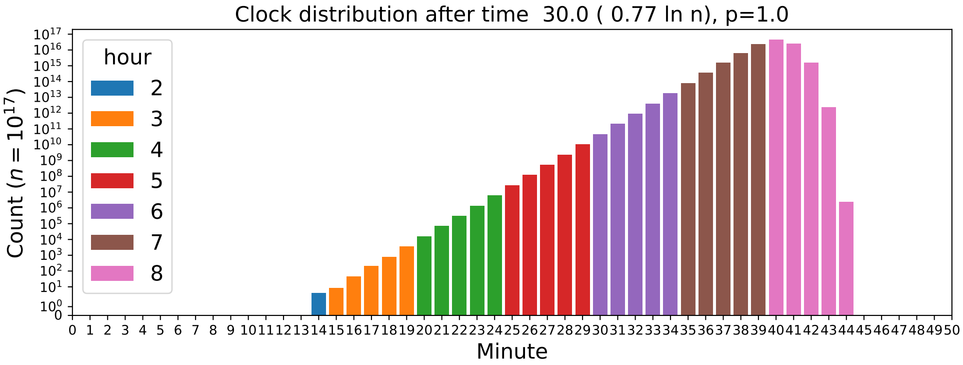


FIGURE 3.2. Clock rules of our protocol, showing a travelling wave distribution over minutes, on a larger population of size $n = 10^{17}$ to emphasize the distribution. The distribution's back tail decays exponentially, and its front tail decays doubly exponentially. A large constant fraction of agents are in the same two consecutive `hour`'s (here 7 and 8). Plot generated from [1].

Checking this condition **stably** detects a tie (i.e., with probability 1, if this condition is true, then there is a tie), because for any nonzero value of g, there must be some agent with $|\texttt{bias}| > \frac{1}{2^L}$.

Cleanup Phases. We must next eliminate all minority opinions, while still relying on the invariant $g = \sum_v v.\texttt{bias}$ to ensure correctness. Note that is it possible with low probability to have a greater count of minority opinions (with smaller values of `bias`), so only relying on counts of positive and negative biases would give possibilities of error.

We first remove any minority agents with large bias, by using an additional subpopulation of Reserve agents that enable additional split reactions for large values of $|\texttt{bias}| > \frac{1}{2^l}$. Then after cancel reactions with the bulk of majority agents, the only minority agents left must have $|\texttt{bias}| < \frac{1}{2^{l+2}}$.

To then remove minority agents with small bias, we allow agents with larger bias to "consume" agents with smaller bias, such as an interaction between agents $+\frac{1}{4}$ and $-\frac{1}{256}$. Here the positive agent can be thought to hold the entire bias $+\frac{1}{4} - \frac{1}{256} = +\frac{63}{256}$, but since this value is not in the allowable states, it can only store that its bias is in the range $+\frac{1}{8} \leq \texttt{bias} \leq +\frac{1}{4}$. Without knowing its exact `bias`, this agent cannot participate in future averaging interactions. However, we show there are enough available majority agents to eliminate all remaining minority via these **consumption reactions**. Thus with high probability, all minority agents are eliminated.

A final phase checks for the presence of both positive and negative `bias`, and if one has been completely eliminated, it stabilizes to the correct output. In the case where both are present, this is a detectable error, where we can move to a slow, correct backup that using the original inputs. Due to the low probability of this case, it contributes negligibly to the total expected time.

3.2.2. Intuitive description of each phase. Our full protocol is broken up into 11 consecutive phases. We describe each phase intuitively before presenting full pseudocode in Section 3.2.4. Note that some further separation of phases was done to create more straightforward proofs of correctness, so simplicity of the proofs was optimized over simplicity of the full protocol pseudocode. It is likely possible to have simpler logic that still solves majority via the same strategy.

Phase 0: "Population splitting" [11] divides agents into roles used in subsequent phases: Main, Reserve, Clock. In timed phases (those not marked as **Untimed** or **Fixed-resolution clock**, including the current phase), Clock agents count from $\Theta(\log n)$ to 0 to cause the switch to the next phase after $\Theta(\log n)$ time.

"Standard" population splitting uses reactions such as $x, x \to r_1, r_2$ to divide agents into two roles r_1, r_2. This takes $\Theta(n)$ time to converge, which can be decreased to $\Theta(\log n)$ time via $r_1, x \to r_1, r_2$ and $r_2, x \to r_2, r_1$, while maintaining that $\#r_1$ and $\#r_2$ are both $n/2 \pm \sqrt{n}$ whp. However, since all agents initially have an opinion, but Clock and Reserve agents do not hold an opinion, agents that adopt role Clock or Reserve must first pass off their opinion to a Main agent.

From each interacting pair of unassigned agents, one will take the Main role and hold the opinions of both agents, interpreting A as $+1$ and B as -1. This Main agent will then be allowed to take at most one other opinion (in an additional reaction that enables rapid convergence of the population splitting), and holding 3 opinions can end up with a bias in the range $\{-3, -2, -1, 0, +1, +2, +3\}$.

Phase 1: Agents do "integer averaging" [10] of biases in the set $\{-3, \dots, +3\}$ via reactions $i, j \to \lfloor \frac{i+j}{2} \rfloor, \lceil \frac{i+j}{2} \rceil$. Although taking $\Theta(n)$ time to converge in some cases, this process is known to result in three consecutive values in $O(\log n)$ time [135]. If those three values are detected to be $\{-1, 0, +1\}$ in the next phase, the algorithm continues.

Phase 2: (**Untimed**) Agents propagate the set of opinions (signs of biases) remaining after Phase 1 to detect if only one opinion remains. If so, we have converged on a majority consensus, and the algorithm halts here (see Fig. 3.6). At this point, this is essentially the exact majority protocol of [127], which takes $O(\log n)$ time with an initial gap $\Omega(n)$, but longer for sublinear gaps (e.g., $\Omega(n)$ time for a gap of 1). Thus, if agents proceed beyond this phase (i.e., if both opinions A and B remain at this point), we will use later that the gap was smaller than $0.025 \cdot \#\mathsf{Main}$. With low probability we have opposite signs but absolute value > 1, in which case we proceed directly to a slow stable backup protocol in Phase 10.

Phase 3: (**Fixed-resolution clock**) The key goal at this phase is to use cancel and split reactions to average the bias across the population to give almost all agents the majority opinion. Biased agents hold a field $\texttt{exponent} \in \{-L, \ldots, -1, 0\}$, describing the magnitude $|\texttt{bias}| = 2^{\texttt{exponent}}$, a quantity we call the agent's **mass**. Cancel reactions eliminate opposite biases $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ with the same $\texttt{exponent}$; cancel reactions strictly reduce total mass. Split reactions $\pm\frac{1}{2^i}, 0 \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}}$ give half of the bias to an unbiased agent, decrementing the $\texttt{exponent}$; split reactions preserve the total mass. The unbiased $\mathcal{O}$ agents, with $\texttt{role} = \mathsf{Main}$, $\texttt{opinion} = \texttt{bias} = 0$, act as the fuel for split reactions.

We want to obtain tighter synchronization in the exponents than Fig. 3.1a, approximating the ideal synchronized behavior of the $O(\log^2 n)$ time algorithm of Fig. 3.1c while using only $O(\log n)$ time. To achieve this, the Clock agents run a "fixed resolution" clock that keeps them roughly synchronized (though not perfectly; see Fig. 3.2) as they count their "minutes" from 0 up to $L' = kL$, using $O(1)$ time per minute. This is done via "drip" reactions $C_i, C_i \to C_i, C_{i+1}$ (when minute i gets sufficiently populated, pairs of C_i agents meet with sufficient likelihood to increment the minute) and $C_j, C_i \to C_j, C_j$ for $i < j$ (new higher minute propagates by epidemic).[2] If randomized transitions are allowed, by lowering the probability p of the drip reaction, the clock rate can be slowed by a constant

[2] This clock is similar to the power-of-two-choices leaderless phase clock of [6], where the agent with smaller (or equal) minute increments their clock ($C_j, C_i \to C_j, C_{i+1}$ for $i \leq j$), but increasing the smaller minute by only 1. Similarly to our clock, the maximum minute can increase only with both agents at the same minute. A similar process was analyzed in [34], and in fact was shown to have the key properties needed for our clock to work—an exponentially-decaying back tail and a double-exponentially-decaying front tail—so it seems likely that a power-of-two-choices clock could also work with our protocol.
The randomized variant of our clock with drip probability p is also similar to the "junta-driven" phase clock of [99], but with a linear number $2pn$ of agents in the junta, using $O(1)$ time per minute, rather than the $O(n^\epsilon)$-size junta of [99], which uses $O(\log n)$ time per minute. There, smaller minutes are brought up by epidemic, and only an agent in the junta seeing another agent at the same minute will increment. The epidemic reaction is exactly the same in both rules. The probability p of a drip reaction can be interpreted as the probability that one of the agents is in the junta. For similar rate of $O(1)$ time per minute phase clock construction see also Dudek and Kosowski work [93].

factor. Although we prove a few lemmas about this generalized clock, and some of our simulation plots in Section 3.2.3 use $p < 1$, our proofs work even for $p = 1$, i.e., a deterministic transition function.

Now the $\mathcal{O}$ agents will use $\Theta(\log n)$ states to store an "hour", coupled to the C clock agents via $C_{\lfloor i/k \rfloor}, \mathcal{O}_j \to C_{\lfloor i/k \rfloor}, \mathcal{O}_i$ if $\lfloor i/k \rfloor > j$, i.e., every consecutive k Clock minutes corresponds to one Main hour, and clock agents drag $\mathcal{O}$ agents up to the current hour (see Fig. 3.3). Our proofs require $k = 45$ minutes per hour when $p = 1$, but smaller values of k work in simulation. For example, the simulation in Fig. 3.1d showing intended behavior of this phase used only $k = 3$ minutes per hour with $p = 1$.

This clock synchronizes the $\texttt{exponent}$s because agents with $\texttt{exponent} = -i$ can only split down to $\texttt{exponent} = -(i+1)$ with an $\mathcal{O}$ agent that has $\texttt{hour} \geq i + 1$.

This prevents the biased agents from doing too many splits too quickly. As a result, during hour i, most of the biased agents have $|\texttt{bias}| = \frac{1}{2^i}$, so the cancel reactions $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ happen at a high rate, providing many $\mathcal{O}$ agents as "fuel" for future split reactions. We tune the constants of the clock to ensure hour i lasts long enough to bring most biased agents down to $\texttt{exponent} = -i$ via split reactions and then let a good fraction do cancel reactions (see Fig. 3.4c).

The key property at the conclusion of this phase is that unless there is a tie, WHP most majority agents end up in three consecutive exponents $-l, -(l+1), -(l+2)$, with a negligible mass of any other Main agent (majority agents at lower/higher exponents, minority agents at any exponent, or $\mathcal{O}$ agents).[3] Phases 5-7 use this fact to quickly push the rest of the population to a configuration where **all** minority agents have exponents strictly below $-(l+2)$; Phase 8 then eliminates these minority agents quickly.

Phase 4: (**Untimed**) The special case of a tie is detected by the fact that, since the total bias remains the initial gap g, if all biased agents have minimal exponent $-L$, g has magnitude less than 1:

$$|g| = \left| \sum_{a.\texttt{role}=\mathsf{Main}} a.\texttt{bias} \right| \leq \sum_{a.\texttt{role}=\mathsf{Main}} |a.\texttt{bias}| \leq \sum_{a.\texttt{role}=\mathsf{Main}} \frac{1}{2^L} < \frac{n}{2^{\lceil \log_2(n) \rceil}} \leq 1.$$

[3] l is defined such that if all biased agents were at exponent $-l$, the gap would be between $0.4 \cdot \#\mathsf{Main}$ and $0.8 \cdot \#\mathsf{Main}$.

The initial gap g is integer valued, so $|g| < 1 \implies g = 0$. Thus this condition implies there is a tie with probability 1; the converse that a tie forces all biased agents to exponent $-L$ holds with high probability. If only exponent $-L$ is detected, the algorithm halts here with all agents reporting output T (see Fig. 3.7). Otherwise, the algorithm proceeds to the next phase.

Phase 5, Phase 6: Using the key property of Phase 3, these phases WHP pull all biased agents above exponent $-l$ down to exponent $-l$ or below using the Reserve R agents. The R's activate themselves in Phase 5 by sampling the exponent of the first biased agent they meet. This ensures WHP that sufficiently many Reserve agents exist with exponents $-l, -(l+1), -(l+2)$ (distributed similarly to the agents with those exponents). Then in Phase 6, they act as fuel for splits, via $R_i, \pm\frac{1}{2^j} \to \pm\frac{1}{2^{j+1}}, \pm\frac{1}{2^{j+1}}$ when $|i| > |j|$. The reserve agents, unlike the $\mathcal{O}$ agents in Phase 3, do not change their exponent in response to interactions with Clock agents. Thus sufficiently many reserve agents will remain to allow the small number of biased agents above exponent $-l$ to split down to exponent $-l$ or below.[4]

Phase 7: This phase allows more general reactions to distribute the dyadic biases, allowing reactions between agents up to two exponents apart, to eliminate the opinion with smaller exponent: $\frac{1}{2^i}, -\frac{1}{2^{i+1}} \to \frac{1}{2^{i+1}}, 0$ and $\frac{1}{2^i}, -\frac{1}{2^{i+2}} \to \frac{1}{2^{i+1}}, \frac{1}{2^{i+2}}$ (and the equivalent with positive/negative biases swapped). Since all agents have exponent $-l$ or below, and many more majority agents exist at exponents $-l, -(l+1), -(l+2)$ than the total number of minority agents anywhere, these (together with standard cancel reactions $\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$) rapidly eliminate all minority agents at exponents $-l, -(l+1), -(l+2)$, while maintaining $\Omega(n)$ majority agents at exponents $\geq -(l+2)$ and $< 0.01n$ total minority agents, now all at exponents $\leq -(l+3)$.

[4] The reason we do this in two separate phases is to ensure that the Reserve agents have close to the same distribution of exponents that the Main agents have at the end of Phase 3. If Reserve agents allowed split reactions in the same phase that they sample the `exponent` of Main agents, then the splits would disrupt the distribution of the Main agents before all Reserve agents have finished sampling. Thus would possibly give the Reserve agents a significantly different distribution among levels than the Main agents had at the start. While this may possibly work anyway, we find it is more straightforward to prove if the Reserve agents have a close distribution over `exponent` values to that of the Main agents.

Phase 8: This phase eliminates the last minority agents, while ensuring that if any error occurred in previous phases, some majority agents remain, to allow detecting the error by the presence of both opinions.[5]

The biased agents add a Boolean field `full`, initially False, and **consumption** reactions that allow an agent at a larger exponent i to consume (set to mass 0 by setting it to be $\mathcal{O}$) an agent at an arbitrarily smaller exponent $j < i$. Now the remaining agent represents some non-power-of-two mass $m = 2^i - 2^j$, which it lacks sufficient memory to track exactly. Thus setting the flag `full` = True corresponds to the agent having an uncertain mass m in the range $2^{i-1} \le m < 2^i$. Because of this uncertainty, full agents are not allowed to consume other smaller levels. However, there are more than enough high-exponent majority agents by this phase to consume all remaining lower exponent minority agents.

Crucially, agents that have consumed another agent and set `full` = True may themselves then be consumed by a third agent (with `full` = False) at an even larger exponent. This is needed because a minority agent at exponent $i \le -(l+3)$ may consume a (rare) majority agent at exponent $j < i$, but the minority agent itself can be consumed by another majority agent with exponent $k > i$.

Phase 9: (**Untimed**) This is identical to Phase 2: it detects whether both biased opinions A and B remain. If not (the likely case), the algorithm halts, otherwise we proceed to the next phase.

Phase 10: (**Untimed**) Agents execute a simple, slow stable majority protocol [47], similar to that of [89, 127] but also handling ties. This takes $\Theta(n \log n)$ time, but the probability that an earlier error forces us to this phase is $O(1/n^2)$, so it contributes negligibly to the total expected time.

3.2.3. Simulation of full algorithm. In this section we show simulation results, where the complete pseudocode of Section 3.2.4 was translated into Java code available on GitHub [2]. In these simulations, we stop the protocol once all agents reach Phase 9. For the low probability case that agents switch to the Phase 10, the simulator prints an error indicating the agents should switch

[5] A naïve idea to reach a consensus at this phase is to allow cancel reactions $\frac{1}{2^i}, -\frac{1}{2^j} \to 0, 0$ between arbitrary pairs of exponents with opposite opinions. However, this has a positive probability of erroneously eliminating the majority. This is because the majority, while it necessarily has larger mass than the minority at this point, could have smaller **count**. For example, we could have 16 A's with `exponent` $= -2$ and 32 B's with `exponent` $= -5$, so A's have mass $16 \cdot 2^{-2} = 4$ and B's have smaller mass $32 \cdot 2^{-5} = 1$, but larger count than A.

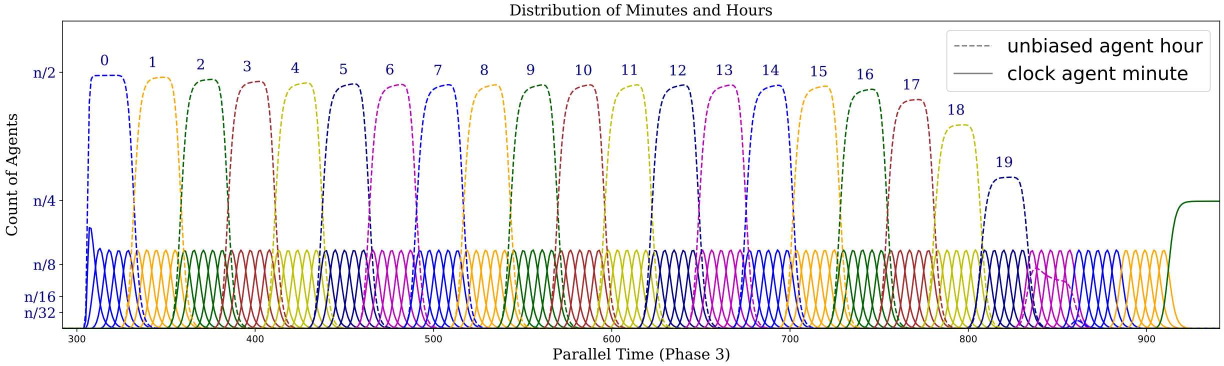
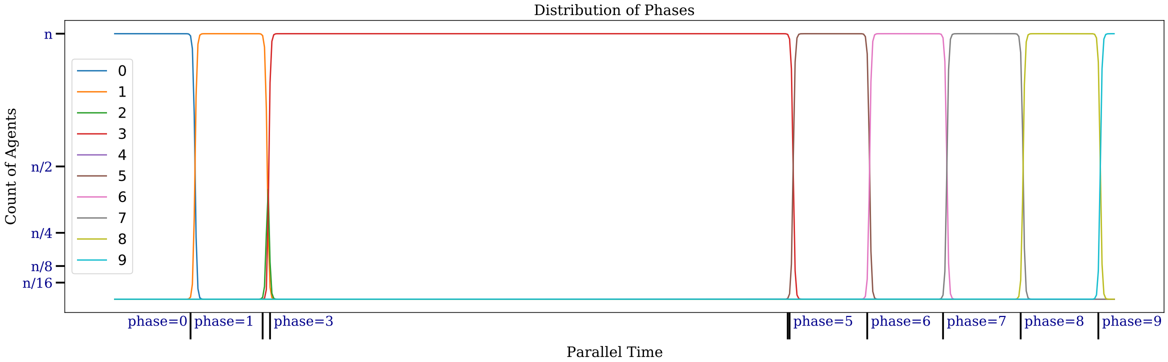


FIGURE 3.3. Fixed-resolution phase clock used in Phase 3, with $n \approx 2^{23}$. All Clock agents set `minute` $= 0$ and count up to $k \cdot L$ with special rules defined in Phase 3. The solid curves show the distributions of the counts at each value of `minute` in the Clock agents. Blocks of $k = 5$ consecutive minutes correspond to a `hour` in the unbiased $\mathcal{O}$ agents with `role` $=$ Main and `bias` $= 0$. The dashed curves show the distribution of the counts of each value of `hour`, with the value written on top. We show every k minutes with one color equal to its hour color. Near the end of Phase 3, the $\mathcal{O}$ count falls to 0 as the majority count takes over. This plot used the value $k = 5$ to emphasize the clock behavior. Later plots will all use the weaker constant $k = 2$. In later plots we also omit the `minute` distribution and only show the `hour` from the $\mathcal{O}$ agents.

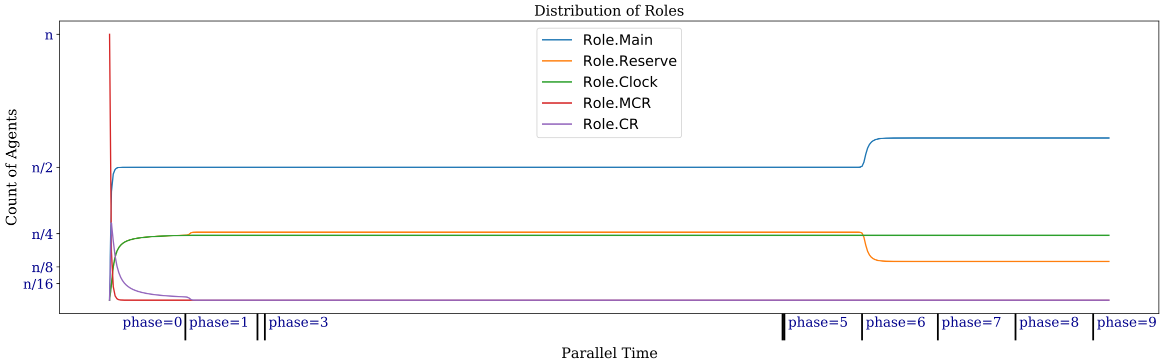
to slow back up, but as expected this was not observed in our simulations. For all our plots, we collect data from simulations with $n \approx 2^{23}$, p (drip probability) $= 0.1$. The first simulation in Fig. 3.3 shows the relationship between `minute` of Clock agents and `hour` of Main agents. Here we used $k = 5$ minutes per hour to show clearly the relationship and the discrete nature of the hours.

All remaining simulations used the even weaker value $k = 2$, to help see enough low probability behavior that the logic enforcing probability-1 correctness in later phases is necessary. We show 3 simulations corresponding to the 3 different types of initial gap. Figs. 3.4 and 3.5 show constant initial gap $g = +2$. This is our "typical" case, where the simulation eventually stabilizes to the correct output in Phase 9. Fig. 3.6 shows linear initial gap $g \approx \frac{n}{10}$, which stabilizes in Phase 2 after quickly cancelling all minority agents. Fig. 3.7 shows initial tie $g = 0$, which stabilizes in Phase 4 after all biased agents reach the minimum `exponent` $= -L = -23$.

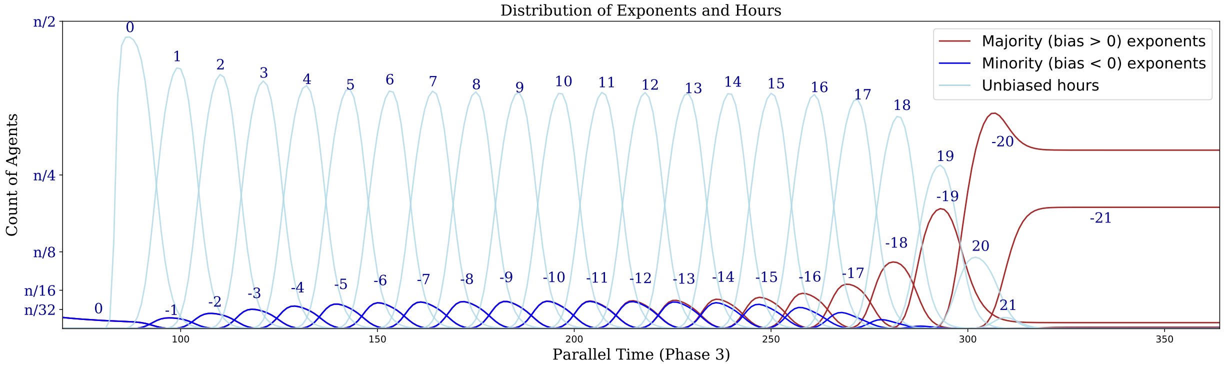
All three simulations show various snapshots of configurations of the biased agents. These particular snapshots are at special times marked in Figs. 3.4d, 3.6a and 3.7a. In all cases, an animation is available at GitHub [3], showing the full evolution of these distributions over all recorded time steps from the simulation.



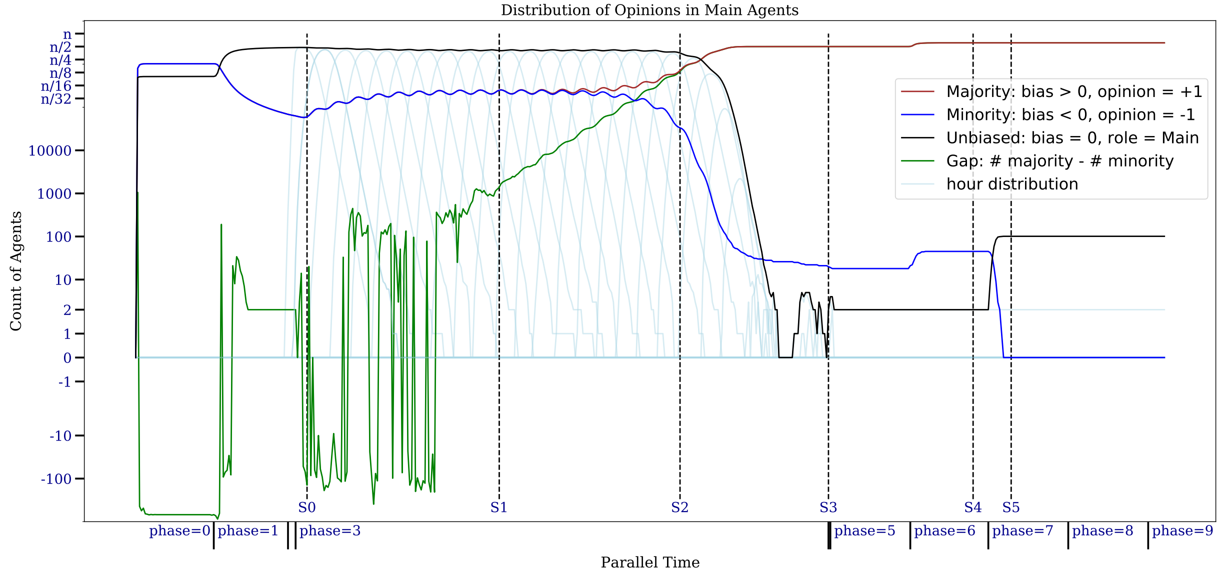
(a) The phase distribution, giving counts of the agents with each value of phase. We show these markers on the horizontal axis in later plots. We have set max counter $= 5\log_2(n)$, and removed the counting requirements for Clock agents in Phase 0. This makes all timed phases 0, 1, 5, 6, 7, 8 take around the same parallel time $2.5\log_2(n)$. The fixed-resolution clock in Phase 3 uses $O(\log n)$ time with a larger constant. Phase 2 and Phase 4 are untimed, so they end almost immediately and are not labeled above.



(b) The role distribution. All agents start in role $\mathsf{Role}_{\mathsf{MCR}}$. By the end of Phase 0 almost all agents decide on a role. They enter Phase 1 with $\approx \frac{n}{2}$, $\approx \frac{n}{4}$, and $\approx \frac{n}{4}$ agents in the respective roles Main, Clock, and Reserve. An agent's role remain the same in the following phases except Phase 6, where split reactions convert Reserve agents into Main agents.

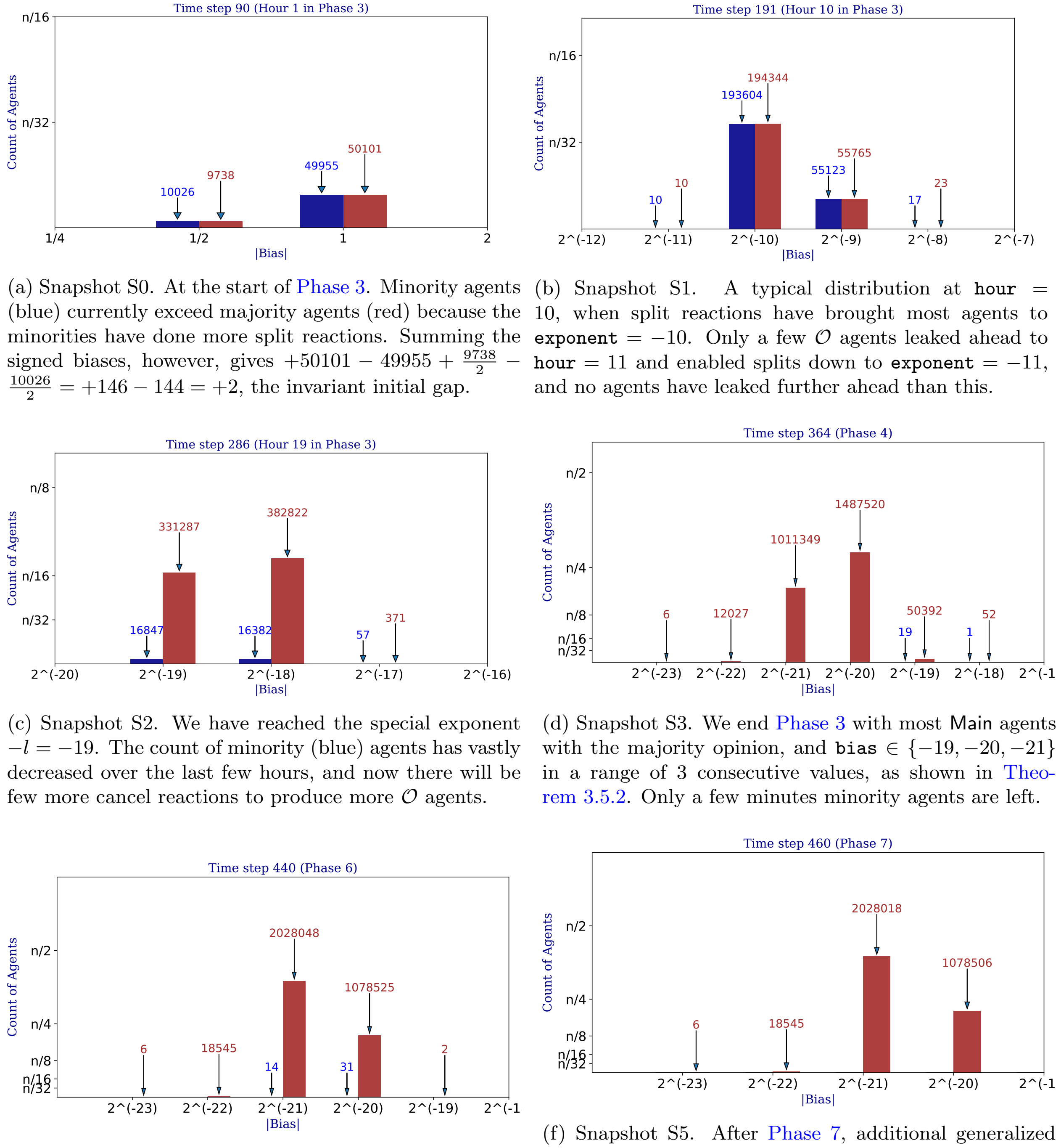


(c) The distribution of exponent and hour in biased and unbiased Main agents. This plot only shows the time during Phase 3, the only time hour is used. The values for exponent can decrease from 0 to $-L$. As described in Phase 3, during hour h, only agents with exponent $> -h$ are allowed to split and decrease their exponent by one. Thus, the changes of exponent are synchronized with the the changes in the hour values. In this simulation, the Phase 3 stopped with majority of agents having 3 consecutive values $(-19, -20, -21)$ in their exponent. The hour values are shown behind the next plot in Fig. 3.4d, which shows the totals of these 3 types of Main agents.



(d) The distribution of `opinion` over all Main agents, with count shown on a log scale. The green line shows the difference in count between majority and minority agents. Special snapshots at times marked S0, S1, S2, S3, S4, S5 are shown in Fig. 3.5. All Main agents are assigned in Phase 0, with **bias** $\in \{0, \pm1, \pm2, \pm3\}$, and all initial biases represented held by this Main subpopulation. Here the gap depends randomly on the distribution of $|$**bias**$| \in \{1, 2, 3\}$. Then Phase 1 brings all **bias** $\in \{-1, 0, +1\}$, so the gap returns to exactly the true initial gap of 2, and the biased counts decrease polynomially like $\frac{1}{t}$ from cancel reactions. In the first part of Phase 3 the gap oscillates randomly about 0, (S0 in Fig. 3.5a). Once we reach a high enough hour / low enough exponent, the doubling trend takes over and the gap undergoes constant exponential growth (S1 in Fig. 3.5b). Finally, this becomes visible as a separation between counts of majority and minority agents (S2 in Fig. 3.5c). Phase 3 ends with a small but nonzero count of minority agents and the count of unbiased $\mathcal{O}$ agents brought near 0 (S3 in Fig. 3.5d). Then during Phase 6, this minority count is amplified slightly by more split reactions bringing the minority exponents down (S4 in Fig. 3.5e). During Phase 7, additional cancel reactions bring the minority count to 0 (S5 in Fig. 3.5f). Since minority agents are gone, Phase 8 has no effect, and the protocol stabilizes to the correct majority output in Phase 9.

FIGURE 3.4. The case of constant initial gap $g = 2$, from simulation with $n = 5122666 \approx 2^{23}$, $p = 0.1$ (drip probability), $k = 2$ (number of minutes per hour). The horizontal axis is in units of parallel time, with the ranges corresponding to each phase marked. All agents converge to the correct majority output in Phase 9.



(a) Snapshot S0. At the start of Phase 3. Minority agents (blue) currently exceed majority agents (red) because the minorities have done more split reactions. Summing the signed biases, however, gives $+50101 - 49955 + \frac{9738}{2} - \frac{10026}{2} = +146 - 144 = +2$, the invariant initial gap.

(b) Snapshot S1. A typical distribution at `hour` $= 10$, when split reactions have brought most agents to `exponent` $= -10$. Only a few $\mathcal{O}$ agents leaked ahead to `hour` $= 11$ and enabled splits down to `exponent` $= -11$, and no agents have leaked further ahead than this.

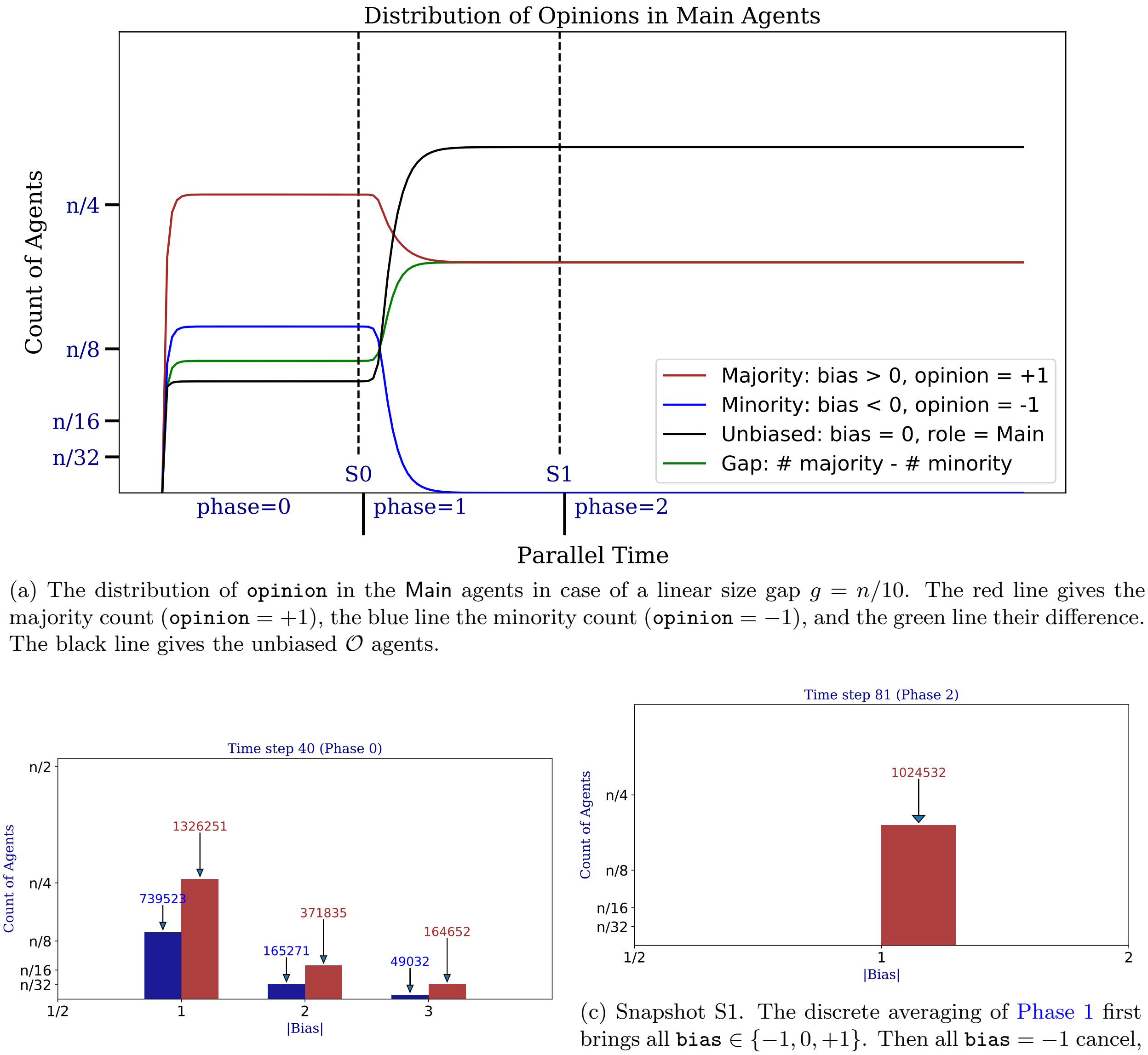
(c) Snapshot S2. We have reached the special exponent $-l = -19$. The count of minority (blue) agents has vastly decreased over the last few hours, and now there will be few more cancel reactions to produce more $\mathcal{O}$ agents.

(d) Snapshot S3. We end Phase 3 with most Main agents with the majority opinion, and `bias` $\in \{-19, -20, -21\}$ in a range of 3 consecutive values, as shown in Theorem 3.5.2. Only a few minutes minority agents are left.

(e) Snapshot S4. After Phase 6, where Reserve agents with `sample` $\in \{-19, -20, -21\}$ enabled additional split reactions that brought all minority agents down.

(f) Snapshot S5. After Phase 7, additional generalized cancel reactions eliminate all minority agents at the higher `exponent` values. There are no minority agents left, and we will stabilize to the correct output in Phase 9.

FIGURE 3.5. Snapshots S0, S1, S2, S3, S4, S5 from Fig. 3.4d, showing the distribution of `bias` among agents with $|$`bias`$| > 0$. The red and blue bars give the count of A (majority, with `bias` > 0) and B (minority, with `bias` < 0) agents respectively. The exact counts are written above the bars, and everywhere not explicitly written the count of is 0. See the link to animations of these plots at the end of this section.

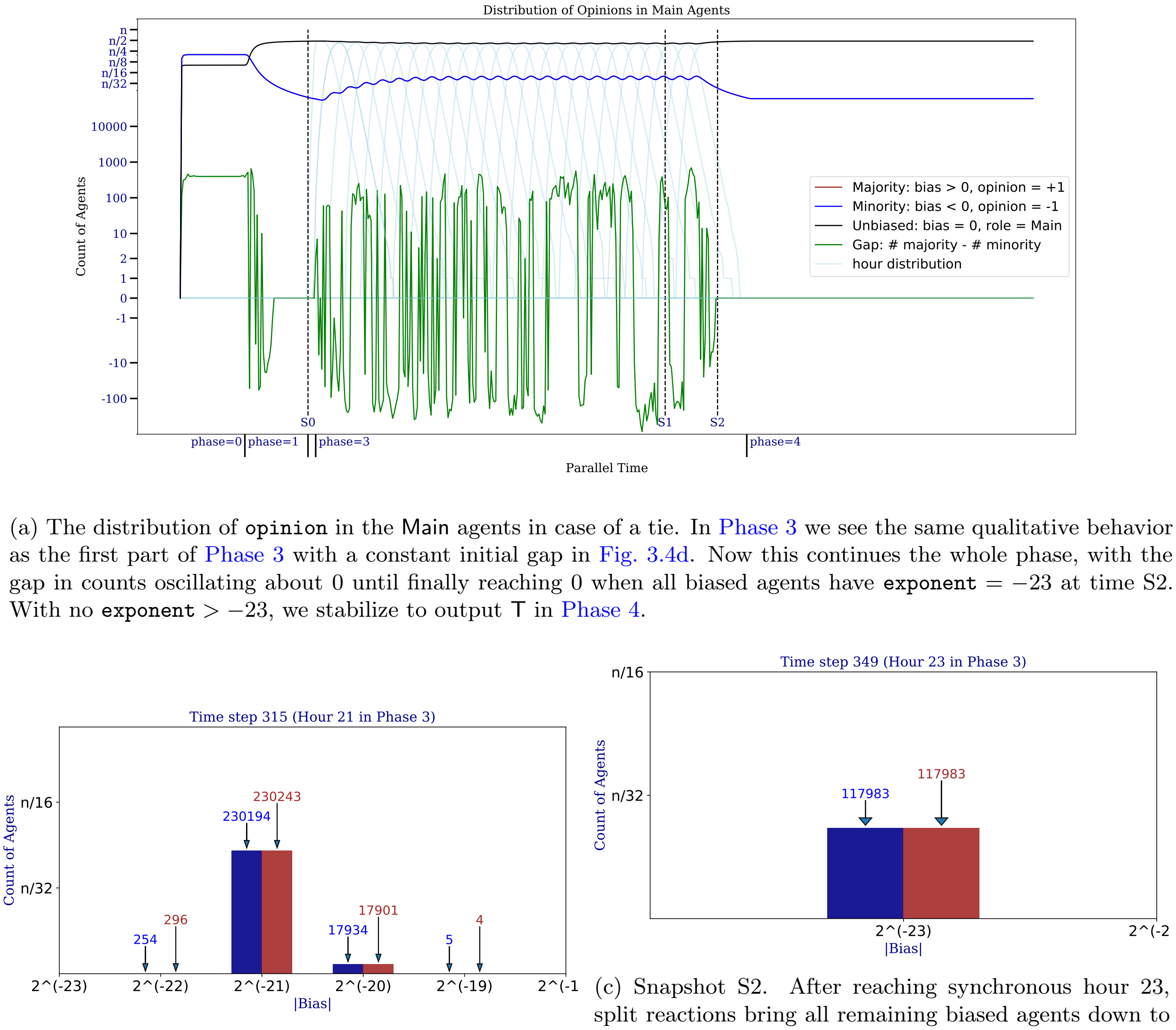


(a) The distribution of `opinion` in the Main agents in case of a linear size gap $g = n/10$. The red line gives the majority count ($\texttt{opinion} = +1$), the blue line the minority count ($\texttt{opinion} = -1$), and the green line their difference. The black line gives the unbiased $\mathcal{O}$ agents.

(b) Snapshot S0. At the end of Phase 0, the biased agents have $|\texttt{bias}| \in \{1, 2, 3\}$.

(c) Snapshot S1. The discrete averaging of Phase 1 first brings all $\texttt{bias} \in \{-1, 0, +1\}$. Then all $\texttt{bias} = -1$ cancel, leaving $\texttt{bias} = +1$ as the only nonzero bias. With the minority opinion eliminated, we converge in Phase 2.

FIGURE 3.6. The case of linear size gap $g = n/10$, again with $n = 5122666 \approx 2^{23}$, $p = 0.1$, $k = 2$. With large initial gap, the simulation converges in Phase 2. Fig. 3.6a shows the counts of each opinion among the population of Main agents. Two special times S0 and S1, and the configurations of biased agents at these snapshots are shown in Figs. 3.6b and 3.6c. See link to animations of these plots at the end of this section.



(a) The distribution of `opinion` in the Main agents in case of a tie. In Phase 3 we see the same qualitative behavior as the first part of Phase 3 with a constant initial gap in Fig. 3.4d. Now this continues the whole phase, with the gap in counts oscillating about 0 until finally reaching 0 when all biased agents have **`exponent`** $= -23$ at time S2. With no **`exponent`** > -23, we stabilize to output T in Phase 4.

(b) Snapshot S1. We see similar distributions as in Fig. 3.5b all the way until the end of Phase 3.

(c) Snapshot S2. After reaching synchronous hour 23, split reactions bring all remaining biased agents down to **`exponent`** $= -23$, which is only possible with an initial tie.

FIGURE 3.7. The case of a tie, with initial gap $g = 0$, again with $n = 5122666 \approx 2^{23}$, $p = 0.1$, $k = 2$. The simulation converges in Phase 2. Fig. 3.6a shows the counts of each opinion among the population of Main agents. Three special times S1, S2, and the configurations of biased agents at these snapshots are shown in Figures 3.7b, 3.7c.

3.2.4. Algorithm pseudocode. In this section we give a full formal description of the main algorithm.

Every agent starts with a read-only field `input` $\in \{\mathsf{A}, \mathsf{B}\}$, a field `output` $\in \{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$ corresponding to outputs that the majority is A, B, or a tie. The protocol is broken up into 11 consecutive phases, marked by the additional field `phase` $= 0 \in \{0, \dots, 10\}$. The phase updates via the epidemic reaction $u.$`phase`$, v.$`phase` $\leftarrow \max(u.$`phase`$, v.$`phase`$)$. Some fields are only used in particular phases, to ensure the total state space is $\Theta(\log n)$. Such fields and the initial behavior of an agent upon entering a phase are described in the **Init** section above each phase. Whenever an agent increments their `phase`, they execute **Init** for the new phase (and sequentially for any phases in between if they happen to increment `phase` by more than 1). We refer to the "end of phase i" to mean the time when the first agent sets `phase` $\leftarrow i + 1$. Note that the agents actually enter each new phase by epidemic, so there is technically no well-defined "beginning of phase i". However, an arbitrarily large constant fraction of agents will have the correct `phase` within a constant amount of time. After this point, interactions between two agents that are out of phase become negligible and otherwise both interacting agents will now have the correct `phase`. Thus we will formally start our arguments for each phase assuming each agent is in the current phase.

Each timed phase i each requires setting agents to count from $c_i \ln n$ down to 0, where the minimum required value of c_i depends on the phase. These constants can be derived from the technical analysis in Sections 3.4-3.6 but for brevity we avoid giving them concrete values in the pseudocode. Our simulations (Section 3.2.3) that used the same small constant $5 \log_2(n)$ for all counters seem to work, but the proofs require larger constants to ensure the necessary behavior within each phase can complete with high probability $1 - O(1/n^2)$. By increasing these constants c_i (along with changing the phase clock constants p, k; see Phase 3 and Theorem 3.5.9), we could also push high probability bound to $1 - O(1/n^c)$ for any desired constant c. For concreteness, use $1-O(1/n^2)$ for most high probability guarantees, since this is large enough to take appropriate union bounds and ensure the extra time from low probability failures does not contribute meaningfully to the total $O(\log n)$ time bound.

Phase 0 is a timed phase that splits the population into three subpopulations: Main to compute majority, Clock to time the phases and the movement through exponents in Phase 3, and Reserve to aid in cleanup during Phase 6. An agent can only move into role Clock or Reserve by "donating" its

Protocol 3.1 Nonuniform Majority$_L(u,v)$. For population sizes n with $L = \lceil \log n \rceil$.
Init: $\texttt{phase} \leftarrow 0 \in \{0, 1, \ldots, 10\}$ and execute **Init** for Phase 0.

```
if $i.\texttt{phase} < j.\texttt{phase}$ where $\{i, j\} = \{u, v\}$ then
    for $p = \{i.\texttt{phase} + 1, \ldots, j.\texttt{phase}\}$ do
        execute Init for Phase $p$ on agent $i$
    $i.\texttt{phase} \leftarrow j.\texttt{phase}$
execute Phase $u.\texttt{phase}(u, v)$
```

opinion to a Main agent, who can collect up to two other opinions in addition to their own, leading to a **bias** of up to ±3. After this phase, the populations of the three roles are near the expected one quarter Main, one quarter Clock, and one half Reserve. Lemma 3.4.2 shows that all initial opinions have been given to assigned Main agents and these subpopulations are near their expected fractions, both with high probability.

Phase 0 Initialize-Roles. Agent u interacting with agent v.
Init $\texttt{role} \leftarrow \mathsf{Role}_{\mathsf{MCR}} \in \{\mathsf{Main}, \mathsf{Clock}, \mathsf{Reserve}, \mathsf{Role}_{\mathsf{MCR}}, \mathsf{Role}_{\mathsf{CR}}\}$
$\texttt{assigned} \leftarrow \mathsf{False} \in \{\mathsf{True}, \mathsf{False}\}$
if $\texttt{input} = \mathsf{A}$, $\texttt{bias} \leftarrow +1 \in \{-3, -2, -1, 0, +1, +2, +3\}$
if $\texttt{input} = \mathsf{B}$, $\texttt{bias} \leftarrow -1 \in \{-3, -2, -1, 0, +1, +2, +3\}$
we always maintain the invariant $\texttt{opinion} = \mathrm{sign}(\texttt{bias}) \in \{-1, 0, +1\}$
if $\texttt{role} = \mathsf{Clock}$, $\texttt{counter} \leftarrow c_0 \ln n \in \{0, \ldots, c_0 \ln n\}$ only used in the current phase

```
if $u.\texttt{role} = v.\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$ then          ▷ Allocate $\approx \frac{1}{2}$Main agents
    $u.\texttt{role} \leftarrow \mathsf{Main}$; $u.\texttt{bias} \leftarrow u.\texttt{bias} + v.\texttt{bias}$
    $v.\texttt{bias} \leftarrow 0$; $v.\texttt{role} \leftarrow \mathsf{Role}_{\mathsf{CR}}$          ▷ $v$ won't use its bias subsequently
if $i.\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$, $j.\texttt{role} = \mathsf{Main}$, $j.\texttt{assigned} = \mathsf{False}$ where $\{i, j\} = \{u, v\}$ then
    $j.\texttt{assigned} \leftarrow \mathsf{True}$; $j.\texttt{bias} \leftarrow j.\texttt{bias} + i.\texttt{bias}$   ▷ Main agents can assign 1 non-Main agent
    $i.\texttt{bias} \leftarrow 0$; $i.\texttt{role} \leftarrow \mathsf{Role}_{\mathsf{CR}}$          ▷ $i$ won't use its bias subsequently
if $i.\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$, $j.\texttt{role} \neq \mathsf{Main}, \mathsf{Role}_{\mathsf{MCR}}$, $j.\texttt{assigned} = \mathsf{False}$ where $\{i, j\} = \{u, v\}$ then
    $j.\texttt{assigned} \leftarrow \mathsf{True}$          ▷ non-Main agents can assign 1 Main agent
    $i.\texttt{role} \leftarrow \mathsf{Main}$
if $u.\texttt{role} = v.\texttt{role} = \mathsf{Role}_{\mathsf{CR}}$ then          ▷ Allocate $\approx \frac{1}{4}$Clock agents, $\approx \frac{1}{4}$Reserve agents
    $u.\texttt{role} \leftarrow \mathsf{Clock}$; $u.\texttt{counter} \leftarrow \Theta(\log n)$
    $v.\texttt{role} \leftarrow \mathsf{Reserve}$
if $u.\texttt{role} = v.\texttt{role} = \mathsf{Clock}$ then          ▷ time the phase once we have at least 2 Clock agents
    execute Standard Counter Subroutine$(u)$, execute Standard Counter Subroutine$(v)$
```

Protocol 3.2 Standard Counter Subroutine(c). Agent c with field $\texttt{counter}$.

```
$c.\texttt{counter} \leftarrow c.\texttt{counter} - 1$
if $c.\texttt{counter} = 0$ then
    $c.\texttt{phase} \leftarrow c.\texttt{phase} + 1$          ▷ move to next phase
```

Phase 1 is a timed phase that averages the biases in Main agents; with high probability at the end of the phase the bias fields have three consecutive values, shown in Lemma 3.4.3.

```
Phase 1 DISCRETE-AVERAGING. Agent $u$ interacting with agent $v$.
Init if role = $\mathsf{Role}_{\mathsf{MCR}}$, phase ← 10 (error, skip to stable backup)
if role = $\mathsf{Role}_{\mathsf{CR}}$, role ← Reserve
if role = Clock, counter ← $c_1 \ln n \in \{0, \ldots, c_1 \ln n\}$ only used in the current phase
if $u$.role = $v$.role = Main then
    $u$.bias ← $\lfloor \frac{u.\mathtt{bias}+v.\mathtt{bias}}{2} \rfloor$; $v$.bias ← $\lceil \frac{u.\mathtt{bias}+v.\mathtt{bias}}{2} \rceil$
for $c \in \{u, v\}$ with $c$.role = Clock do
    execute STANDARD COUNTER SUBROUTINE(c)
```

Phase 2 (an untimed phase) checks to see if the entire minority population was eliminated in Phase 1 by checking whether both positive and negative biases still exist. If not, the minority opinion is gone, and the protocol will stabilize here to the correct output. Otherwise, we proceed to the next phase; note this phase is untimed and proceeds immediately upon detection of conflicting opinions. Lemma 3.4.3 shows that starting from a large initial gap, we will stabilize here with high probability.

```
Phase 2 OUTPUT-THE-CONSENSUS. Agent $u$ interacting with agent $v$.
Init opinions ← {opinion} ⊆ $\{-1, 0, +1\}$
$u$.opinions, $v$.opinions ← $u$.opinions ∪ $v$.opinions        ▷ union opinions
if $\{-1, +1\}$ ⊆ opinions then
    $u$.phase, $v$.phase ← $u$.phase + 1        ▷ no consensus, move to next phase
else if $+1 \in$ opinions then
    $u$.output, $v$.output ← A        ▷ current consensus is A
else if $-1 \in$ opinions then
    $u$.output, $v$.output ← B        ▷ current consensus is B
else if opinions = $\{0\}$ then
    $u$.output, $v$.output ← T        ▷ current consensus is T
```

Phase 3 is where the bulk of the work gets done. It assumes a starting condition where all bias $\in \{-1, 0, +1\}$ (ie. like the initial condition, but allowing some cancelling to have already happened). So the **Init** checks if any $|\mathtt{bias}| > 1$, which can only happen with low probability, and we consider this an error and simply proceed immediately to Phase 10. The biased agents (with non-zero opinion) have an additional field exponent $\in \{-L, \ldots, 0\}$, initially 0, where $L = \lceil \log_2(n) \rceil$, corresponding to holding $2^{\mathtt{exponent}}$ units of mass. The unbiased $\mathcal{O}$ agents have an additional field hour $= L \in \{0, \ldots, L\}$, and will only participate in split reactions with $-\mathtt{exponent} > \mathtt{hour}$. The field hour is set by the Clock agents, who have a field $\texttt{minute} = 0 \in \{0, \ldots, kL\}$, which counts up as Phase 3 proceeds. Intuitively, there are k minutes in an hour, so $\texttt{hour} = \lceil \frac{\texttt{minute}}{k} \rceil$ ranges from 0 to L. Fig. 3.3 shows how nonconsecutive minutes in the Clock agents may overlap significantly, but non-consecutive hours in the $\mathcal{O}$ agents have negligible overlap. Once a Clock agent has minute at its maximum value kL, they initialize a new field $\texttt{counter} = \Theta(\log n) \in \{0, \ldots, \Theta(\log n)\}$ to wait for the end of the phase (waiting for any remaining $\mathcal{O}$ agents to reach $\texttt{hour} = L$ and the distribution to settle).

See the overview in Section 3.2.2 for an intuitive description of this phase. Theorem 3.5.1 gives the main result of Phase 3 in the case of an initial tie, where all remaining biased agents are at the minimal $\texttt{exponent} = -L$. In the other case, Theorem 3.5.2 gives the main result that most of the population settle on the majority output with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$.

Phase 3 SYNCHRONIZED-RATIONAL-AVERAGING. Agent u interacting with agent v.

Init if $|\texttt{bias}| > 1$, $\texttt{phase} \gets 10$ (error, skip to stable backup)
if $\texttt{role} = \textsf{Main}$ and $\texttt{opinion} \in \{-1, +1\}$, $\texttt{exponent} \gets 0 \in \{-L, \ldots, -1, 0\}$, and we define $\texttt{bias} = \texttt{opinion} \cdot 2^{\texttt{exponent}}$
if $\texttt{role} = \textsf{Main}$ and $\texttt{opinion} = 0$, $\texttt{hour} \gets 0 \in \{0, \ldots, L\}$
if $\texttt{role} = \textsf{Clock}$, $\texttt{minute} \gets 0 \in \{0, \ldots, kL\}$ and $\texttt{counter} \gets c_3 \ln n \in \{0, \ldots, c_3 \ln n\}$

```
if $u.\texttt{role} = v.\texttt{role} = \textsf{Clock}$ then
    if $u.\texttt{minute} = v.\texttt{minute} < kL$ then
        $u.\texttt{minute} \gets \texttt{minute} + 1$                          ▷ clock drip reaction
    else if $u.\texttt{minute} \neq v.\texttt{minute}$ then
        $u.\texttt{minute}, v.\texttt{minute} \gets \max(u.\texttt{minute}, v.\texttt{minute})$     ▷ clock epidemic reaction
    else if $u.\texttt{minute} = v.\texttt{minute} = kL$ then        ▷ count only when both clocks finished
        execute STANDARD COUNTER SUBROUTINE(u), STANDARD COUNTER SUBROUTINE(v)
if $m.\texttt{role} = \textsf{Main}, m.\texttt{opinion} = 0$ and $c.\texttt{role} = \textsf{Clock}$ where $\{m, c\} = \{u, v\}$ then
    $m.\texttt{hour} \gets \max\left(m.\texttt{hour}, \lfloor \frac{c.\texttt{minute}}{k} \rfloor\right)$                ▷ clock update reaction
else if $u.\texttt{role} = v.\texttt{role} = \textsf{Main}$ then
    if $\{u.\texttt{opinion}, v.\texttt{opinion}\} = \{-1, +1\}$ and $u.\texttt{exponent} = v.\texttt{exponent} = -h$ then
        $u.\texttt{opinion}, v.\texttt{opinion} \gets 0$; $u.\texttt{hour}, v.\texttt{hour} \gets h$          ▷ cancel reaction
    if $t.\texttt{opinion} = 0$, $i.\texttt{opinion} \in \{-1, +1\}$ and $|t.\texttt{hour}| > |i.\texttt{exponent}|$, where $\{t, i\} = \{u, v\}$ then
        $t.\texttt{opinion} \gets i.\texttt{opinion}$                                 ▷ split reaction
        $i.\texttt{exponent}, t.\texttt{exponent} \gets i.\texttt{exponent} - 1$      ▷ sets $i.\texttt{bias}, t.\texttt{bias} \gets i.\texttt{bias}/2$
```

In Phase 4 (an untimed phase), the population checks if all Main agents have reached the minimum $\texttt{exponent} = -L$, which only happens in the case of a tie. If so, the population will stabilize to the tie output. Otherwise, any Main agent with `exponent` above L can trigger the move to the next phase.

Phase 4 OUTPUT-TIE. Agent u interacting with agent v.
Init `output` ← T

```
if $|m.\texttt{bias}| > 2^{-L}$ where $m \in \{u, v\}$ then      ▷ stable if all bias $\in \{-\frac{1}{2^L}, 0, +\frac{1}{2^L}\}$
    $u.\texttt{phase}, v.\texttt{phase} \leftarrow u.\texttt{phase} + 1$      ▷ end of this phase
```

If there was not a tie detected in Phase 4, then most agents should have the majority opinion, with `exponent` $\in \{-l, -(l+1), -(l+2)\}$ in a small range. The next goal is to bring all agents with `exponent` $> -l$ down to `exponent` $\leq -l$. This will be accomplished by the Reserve agents, whose goal is to let exponents above l do split reactions.

The Reserve agents do this across two consecutive phases. In Phase 5, the Reserve agents become active, and will set `sample` $= \perp \in \{\perp, 0, \ldots, L\}$ to the exponent of the first biased agent they meet, which is likely in $\{-l, -(l+1), -(l+2)\}$. The Clock agents now only hold a field `counter` $= \Theta(\log n) \in \{0, \ldots, \Theta(\log n)\}$ to act as a simple timer for how long to wait until moving to the next phase. This allows the Reserve agents to adopt a distribution of `exponent` values approximately equal to that of the Main agents. This behavior is proven in Lemma 3.6.1 and Lemma 3.6.2.

Phase 5 RESERVES-SAMPLE-EXPONENT. Agent u interacting with agent v.
Init if `role` = Reserve, `sample` $\leftarrow \perp \in \{\perp, -L, \ldots, -1, 0\}$
if `role` = Clock, `counter` $\leftarrow c_5 \ln n \in \{0, \ldots, c_5 \ln n\}$

```
if $r.\texttt{role}$ = Reserve and $m.\texttt{role}$ = Main, $m.\texttt{opinion} \in \{-1, +1\}$ where $\{r, m\} = \{u, v\}$ then
    if $r.\texttt{sample} = \perp$ then      ▷ sample exponent of biased agent $m$
        $r.\texttt{sample} \leftarrow m.\texttt{exponent}$
for $c \in \{u, v\}$ with $c.\texttt{role}$ = Clock do
    execute STANDARD COUNTER SUBROUTINE(c)
```

In Phase 6, the Reserve agents can help facilitate more split reactions, with any agent at a exponent above their sampled exponent. Because they have approximately the same distribution across all exponents as Main agents, particular exponents $-l, -(l+1), -(l+2)$, this allows them to bring all agents above `exponent` $= -l$ down to $-l$ or below. Again, the Clock agents keep a `counter` $\in \{0, \ldots, c_6 \ln n\}$. Lemma 3.6.2 proves that Phase 6 works as intended, bringing all agents down to `exponent` $\leq -l$ with high probability.

Phase 6 Reserve-Splits. Agent u interacting with agent v.
Init if $\texttt{role} = \textsf{Clock}$, $\texttt{counter} \leftarrow c_6 \ln n \in \{0, \ldots, c_6 \ln n\}$

```
if $r.\texttt{role} = \textsf{Reserve}$ and $m.\texttt{role} = \textsf{Main}, m.\texttt{opinion} \in \{-1, +1\}$ where $\{r, m\} = \{u, v\}$ then
    if $r.\texttt{sample} \neq \perp$ and $r.\texttt{sample} < m.\texttt{exponent}$ then
        $r.\texttt{role} \leftarrow \textsf{Main}$; $r.\texttt{opinion} \leftarrow m.\texttt{opinion}$        ▷ split reaction
        $r.\texttt{exponent}, m.\texttt{exponent} \leftarrow m.\texttt{exponent} - 1$        ▷ sets $r.\texttt{bias}, m.\texttt{bias} \leftarrow m.\texttt{bias}/2$
for $c \in \{u, v\}$ with $c.\texttt{role} = \textsf{Clock}$ do
    execute Standard Counter Subroutine(c)
```

Now that all agents are $\texttt{exponent} \leq -l$, the goal of Phase 7 is to eliminate any minority agents with exponents $-l, -(l+1), -(l+2)$. This is done by letting the biased agents do generalized cancel reactions that allow their difference in exponents to be up to 2, while still preserving the mass invariant. Again, the Clock agents keep a $\texttt{counter} \in \{0, \ldots, c_7 \ln n\}$. Lemma 3.6.5 shows that by the end of this phase, any remaining minority agents must have $\texttt{exponent} < -(l+2)$, with high probability.

Phase 7 High-exponent-Minority-Elimination. Agent u interacting with agent v
Init if $\texttt{role} = \textsf{Clock}$, $\texttt{counter} \leftarrow c_7 \ln n \in \{0, \ldots, c_7 \ln n\}$

```
if $u.\texttt{role} = v.\texttt{role} = \textsf{Main}$ and $\{u.\texttt{opinion}, v.\texttt{opinion}\} = \{-1, +1\}$ then
    if $u.\texttt{exponent} = v.\texttt{exponent}$ then
        $u.\texttt{opinion}, v.\texttt{opinion} \leftarrow 0$        ▷ cancel reaction
    else if $i.\texttt{exponent} = j.\texttt{exponent} + 1$ where $\{i, j\} = \{u, v\}$ then
        $i.\texttt{exponent} \leftarrow i.\texttt{exponent} - 1$        ▷ gap-1 cancel reaction
        $j.\texttt{opinion} \leftarrow 0$        ▷ example bias update: $+\frac{1}{4}, -\frac{1}{8} \to +\frac{1}{8}, 0$
    else if $i.\texttt{exponent} = j.\texttt{exponent} + 2$ where $\{i, j\} = \{u, v\}$ then
        $j.\texttt{opinion} \leftarrow i.\texttt{opinion}$        ▷ gap-2 cancel reaction
        $i.\texttt{exponent} \leftarrow i.\texttt{exponent} - 1$        ▷ example bias update: $+\frac{1}{4}, -\frac{1}{16} \to +\frac{1}{8}, +\frac{1}{16}$
        $j.\texttt{exponent} \leftarrow i.\texttt{exponent} - 2$
for $c \in \{u, v\}$ with $c.\texttt{role} = \textsf{Clock}$ do
    execute Standard Counter Subroutine(c)
```

Now that all minority agents occupy exponents below $-(l+2)$, yet a large number of majority agents remain at exponents $-l, -(l+1), -(l+2)$, in Phase 8, the algorithm eliminates the last remaining minority opinions at any exponent. It allows opposite-opinion agents of **any** two exponents to react and eliminate the smaller-exponent opinion. The larger exponent $\frac{1}{2^i}$ "absorbs" the smaller $-\frac{1}{2^j}$, setting the smaller to mass 0; the larger now represents mass $\frac{1}{2^i} - \frac{1}{2^j}$, which it lacks the memory to track exactly, so it cannot absorb any further agents (though it can itself be absorbed by $-\frac{1}{2^m}$ for $m > i$). Lemma 3.6.6 shows that Phase 8 eliminates any remaining minority agents with high probability.

Phase 8 Low-exponent-Minority-Elimination. Agent u interacting with agent v
Init if role $=$ Clock, counter $\leftarrow c_8 \ln n \in \{0,\ldots,c_8 \ln n\}$

```
if $u$.role = $v$.role = Main and $\{u$.bias, $v$.bias$\} = \{-1,+1\}$ then
    if $i$.exponent > $j$.exponent and $i$.full = False where $\{i,j\} = \{u,v\}$ then
        $i$.full ← True                                          ▷ consumption reaction
        $j$.opinion ← 0
for $c \in \{u,v\}$ with $c$.role = Clock do
    execute Standard Counter Subroutine(c)
```

Phase 9 (an untimed phase) acts exactly as Phase 2, to check that agents have reached consensus.

Phase 9 Output-the-Consensus. Exact repeat of Phase 2.

In Phase 10, the agents give up on the fast algorithm, having determined in Phase 9 that it failed to reach consensus, or detected an earlier error with role or bias assignment. Instead they rely instead on a slow stable backup protocol. This is a 6-state protocol that stable decides between the three cases of majority A, B, and tie T. They only use the fields output $=$ input $\in \{\mathsf{A},\mathsf{B},\mathsf{T}\}$, and the initial field active $=$ True $\in \{$True, False$\}$. Lemma 3.6.7 proves this 6-state protocol stably computes majority in $O(n \log n)$ time.

Phase 10 Stable-Backup. Agent u interacting with agent v. Similar to 6-state algorithm from [47], slight modification of 4-state algorithm from [89, 127].
Init output $\leftarrow$ input, active $\leftarrow$ True

```
if $u$.active = $v$.active = True then
    if $\{u$.output, $v$.output$\} = \{A, B\}$ then
        $u$.output, $v$.output ← T                                    ▷ cancel reaction
    else if $i$.output ∈ {A, B} and $t$.output = T where $\{i,t\} = \{u,v\}$ then
        $t$.output ← $i$.output, $t$.active ← False               ▷ biased converts unbiased
if $a$.active = True and $p$.active = False where $\{a,p\} = \{u,v\}$ then
    $p$.output ← $a$.output                                      ▷ active converts passive
```

3.3. Useful time bounds

This section introduces various probability bounds which will be used repeatedly in later analysis.

We will use the following standard multiplicative Chernoff bound:

Theorem 3.3.1. *Let $X = X_1 + \ldots + X_k$ be the sum of independent $\{0,1\}$-valued random variables, with $\mu = \mathbb{E}[X]$. Then for any $\delta > 0$, $\mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right)$.*

We use the standard Azuma inequality for supermartingales:

THEOREM 3.3.2. *Let $X_0, X_1, X_2, \dots$ be a supermartingale such that, for all $i \in \mathbb{N}$, $|X_{i+1} - X_i| \leq c_i$. Then for all $n \in \mathbb{N}$ and $\epsilon > 0$, $\mathbb{P}[(X_n - X_0) - \mathbb{E}[X_n - X_0] \geq \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=0}^{n} c_i^2}\right)$.*

In application we will consider potential functions ϕ that decay exponentially, with $\mathbb{E}[\phi_{j+1}] \leq (1-\epsilon)\phi_j$. In this case, we will take the logarithm $\Phi = \ln(\phi)$, which we will show is a supermartingale upon which we can apply Azuma's Inequality to conclude that ϕ achieves a requisite amount of exponential decay.

We will also use the following two concentration bounds on heterogeneous sums of geometric random variables, due to Janson [115, Theorems 2.1, 3.1]:

THEOREM 3.3.3. *Let X be sum of k independent geometric random variables with success probabilities $p_1, ..., p_k$, let $\mu = \mathbb{E}[X] = \sum_{i=1}^{k} \frac{1}{p_i}$, and let $p^* = \min_{1 \leq i \leq k} p_i$. For all $\lambda \geq 1$, $\mathbb{P}[X \geq \lambda\mu] \leq e^{-p^*\mu(\lambda - 1 - \ln\lambda)}$. For all $\lambda \leq 1$, $\mathbb{P}[X \leq \lambda\mu] \leq e^{-p^*\mu(\lambda-1-\ln\lambda)}$.*

The expression $\lambda - 1 - \ln\lambda$ is hard to work with if we are not fixing exact constant values for λ. The following Corollary gives asymptotic approximation to the error bound:

Corollary 3.3.4. *Let X be sum of k independent geometric random variables with success probabilities $p_1, ..., p_k$, let $\mu = \mathbb{E}[X] = \sum_{i=1}^{k} \frac{1}{p_i}$, and let $p^* = \min_{1 \leq i \leq k} p_i$. For any $0 < \epsilon < 1$, we have $(1-\epsilon)\mu \leq X \leq (1+\epsilon)\mu$ with probability $1 - \exp\left(-\Theta(\epsilon^2 p^* \mu)\right)$.*

PROOF. Setting $\lambda = 1 + a$ in Theorem 3.3.3, where $a = \epsilon$ in the upper bound and $a = -\epsilon$ in the lower bound, by Taylor series approximation of $\ln(1+a)$, $\lambda - 1 - \ln(\lambda) = a - \ln(1+a) \geq a - (a - \frac{a^2}{2} + \frac{a^3}{3}) = a^2(\frac{1}{2} - \frac{a}{3}) = \Theta(\epsilon^2)$. The stated inequality then follows from Theorem 3.3.3. □

The following lemmas give applications of Theorem 3.3.3 and Corollary 3.3.4 to common processes that we will repeatedly analyze. When the fractions we consider are distinct and independent of n, Corollary 3.3.4 applies to give tight time bounds with very high probability. We also consider cases where we run a process to completion and bring a count to 0. Here we must use Theorem 3.3.3 and only get a high probability bound with times at most a constant factor above the mean.

First we consider the epidemic process, $i, s \to i, i$, moving from a constant fraction infected to another constant fraction infected.

Lemma 3.3.5. *Let* $0 < a < b < 1$. *Consider the epidemic process starting from a count of* $a \cdot n$ *infected agents. The expected parallel time* t *until there is a count* $b \cdot n$ *of infected agents is*

$$\mathbb{E}[t] = \frac{\ln\!\left(b - \frac{1}{n}\right) - \ln\!\left(1 - b + \frac{1}{n}\right) - \ln(a) + \ln(1-a)}{2} \sim \frac{\ln(b) - \ln(1-b) - \ln(a) + \ln(1-a)}{2}.$$

Let $c = \min(a, 1 - b + \frac{1}{n})$ *and* $0 < \epsilon < 1$. *Then* $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ *with probability at least* $1 - \exp\!\left[-\Theta(\epsilon^2 \mathbb{E}[t] n c)\right]$.

Proof. When there are i infected agents, the probability the next reaction infects another agent and increases this count is $\frac{i(n-i)}{\binom{n}{2}}$. The number of interactions T for the count to increase from $a \cdot n$ to $b \cdot n$ is a sum of geometric random variables, with expected value

$$\begin{aligned}\mathbb{E}[T] &= \sum_{i=a\cdot n}^{b\cdot n - 1} \frac{\binom{n}{2}}{i(n-i)} \sim \frac{1}{2} \sum_{i=a\cdot n}^{b\cdot n-1} \frac{1}{\frac{i}{n}(1-\frac{i}{n})} \sim \frac{n}{2} \int_{x=a}^{b-\frac{1}{n}} \frac{dx}{x(1-x)} \\ &= \frac{n}{2}[\ln(x) - \ln(1-x)]_a^{b-\frac{1}{n}} = n \cdot \frac{\ln\!\left(b - \frac{1}{n}\right) - \ln\!\left(1 - b + \frac{1}{n}\right) - \ln(a) + \ln(1-a)}{2},\end{aligned}$$

giving the stated value of $\mathbb{E}[t]$ after converting from interactions to parallel time. The minimum probability $p^* = \Theta(\min(a, 1 - b + \frac{1}{n}))$, so using Corollary 3.3.4 we have Then $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\!\left[-\Theta(\epsilon^2 \mathbb{E}[T] p^*)\right]$. □

Note that if $a, (1-b), \epsilon$ are all constants independent of n, then the bound above is with very high probability. If we wanted to consider the complete epidemic process (which starts with $a = 1/n$ and ends with $(1-b) = 1/n$), we would have $\mathbb{E}[t] = \Theta(\log n)$ and minimum probability $p^* = \Theta(\frac{1}{n})$. Then the probability bound would become $1 - \exp[-\Theta(\log n)] = 1 - n^{-\Theta(1)}$, which is high probability, but requires carefully considering the constants to get the exact polynomial bound. In our proofs, we only use the very high probability case. For tight bounds on a full epidemic with precise constants, see [53, 134].

Next we consider the cancel reactions $a, b \to 0, 0$, which are key to the majority protocol.

Lemma 3.3.6. *Consider two disjoint subpopulations* A *and* B *of initial sizes* $|A| = a \cdot n$ *and* $|B| = b \cdot n$, *where* $0 < b < a < 1$. *An interaction between an agent in* A *and an agent in* B *is a* ***cancel reaction*** *which removes both agents from their subpopulations. Thus after* i *cancel reactions we have* $|A| = a \cdot n - i$ *and* $|B| = b \cdot n - i$.

The expected parallel time t until $d \cdot n$ cancel reactions occur, where $d < b$ is

$$\mathbb{E}[t] = \frac{\ln(b) - \ln(a) - \ln\left(b - d + \frac{1}{n}\right) + \ln\left(a - d + \frac{1}{n}\right)}{2(a-b)}.$$

Let $c = (a - d + \frac{1}{n})(b - d + \frac{1}{n})$ and $0 < \epsilon < 1$. Then $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\left[-\Theta(\epsilon^2 \mathbb{E}[t] nc)\right]$.

The parallel time t until all $b \cdot n$ cancel reactions occur has $\mathbb{E}[t] \sim \frac{\ln n}{2(a-b)}$ and satisfies $t \leq \frac{5 \ln n}{2(a-b)}$ with high probability $1 - O(1/n^2)$.

Again, the first bound is with very high probability if all constants a, b, d are independent of n.

Proof. After i cancel reactions, the probability of the next cancel reaction is $p \sim \frac{2|A| \cdot |B|}{n^2} = 2(a - \frac{i}{n})(b - \frac{i}{n})$, and the number of interactions until this cancel reaction is a geometric random variable with mean p. The number of interactions T for $d \cdot n$ cancel reactions to occur is a sum of geometrics with mean

$$\mathbb{E}[T] = \sum_{i=0}^{d \cdot n - 1} \frac{1}{2(a - \frac{i}{n})(b - \frac{i}{n})} \sim n \int_{x=0}^{d - \frac{1}{n}} \frac{dx}{2(a-x)(b-x)} = \frac{n}{2(a-b)} \int_{x=0}^{d-\frac{1}{n}} \left(\frac{1}{b-x} - \frac{1}{a-x} \right) dx$$
$$= \frac{n}{2(a-b)} \Big[-\ln(b-x) + \ln(a-x) \Big]_0^{d - \frac{1}{n}} = n \cdot \frac{\ln(b) - \ln(a) - \ln\left(b - d + \frac{1}{n}\right) + \ln\left(a - d + \frac{1}{n}\right)}{2(a-b)}.$$

Translating to parallel time and using Corollary 3.3.4 gives the first result, where minimum probability $p^* = \Theta(c)$.

In the second case where $d = b$ and we are waiting for all cancel reactions to occur, then

$$\mathbb{E}[t] \sim \frac{\ln(b) - \ln(a) + \ln(n) + \ln(a-b)}{2(a-b)} = \frac{\ln n}{2(a-b)},$$

and $\mu = \mathbb{E}[T] \sim \frac{n \ln n}{2(a-b)}$. Now the minimum geometric probability is when $i = bn - 1$ and $p^* \sim \frac{2(a-b)}{n}$. Choosing $\lambda = 5$ so that $\lambda - 1 - \ln \lambda > 2$, by Theorem 3.3.3 we have

$$\mathbb{P}\left[t \geq \frac{5 \ln n}{2(a-b)}\right] \leq \mathbb{P}[T \geq \lambda \mu] \leq e^{-p^* \mu (\lambda - 1 - \ln \lambda)} = \exp\left(-\frac{2(a-b)}{n} \cdot \frac{n \ln n}{2(a-b)} \cdot 2 \right) = n^{-2}.$$

Thus $t \leq \frac{5 \ln n}{2a}$ with high probability.

Note we get the same result for this second case also by assuming a fixed minimal fraction $a - b$ in $|A|$ is always there to cancel the agents from $|B|$, and using the next Lemma 3.3.7. □

Now we consider a "one-sided" cancel process, where reactions $a, b \to a, 0$ change only the b agents into a different state. This process happens for example when Clock agents change the `hour` of $\mathcal{O}$ agents.

Lemma 3.3.7. *Let $0 < a, b_1, b_2, \epsilon < 1$ be constants with $b_1 > b_2$. Consider a subpopulation A maintaing its size above $a \cdot n$, and B initially of size $b_1 \cdot n$. Any interaction between an agent in A and in B is* ***meaningful****, and forces the agent in B to leave its subpopulation.*

The expected parallel time t until the subpopulation B reaches size $b_2 \cdot n$ is

$$\mathbb{E}[t] = \frac{\ln(b_1) - \ln(b_2)}{2a},$$

and satisfies $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\left[-\Theta(\epsilon^2 \mathbb{E}[t] n a b_2)\right]$.

The parallel time t until the subpopulation B reaches size 0 has $\mathbb{E}[t] \sim \frac{\ln n}{2a}$ and satisfies $t \leq \frac{5 \ln n}{2a}$ with high probability $1 - O(1/n^2)$.

Again, the first bound is with very high probability if constants a, b_2 are independent of n.

Proof. When $|B| = i$, then the probability of a meaningful interaction $p \sim \frac{2ia}{n}$. Then the number of interactions before the next meaningful interaction is a geometric with probability p, and the total number T of interaction is a sum of geometrics with mean

$$\mathbb{E}[T] = \sum_{i=b_2 n+1}^{b_1 n} \frac{n}{2ia} = \frac{n}{2a}\left(\sum_{i=1}^{b_1 n} \frac{1}{i} - \sum_{i=1}^{b_2 n+1} \frac{1}{i}\right) \sim \frac{n}{2a}(\ln(b_1 n) - \ln(b_2 n + 1)) \sim n \frac{\ln(b_1) - \ln\left(b_2 + \frac{1}{n}\right)}{2a}.$$

Translating to parallel time and using Corollary 3.3.4 gives the first result, where minimum probability $p^* = \Theta(a \cdot b_2)$.

In the case where $b_2 = 0$, we have $\mu = \mathbb{E}[T] \sim \frac{n \ln n}{2a}$ and $\mathbb{E}[t] \sim \frac{\ln(n)}{2a}$. Now the minimum geometric probability $p^* = \frac{2a}{n}$. Choosing $\lambda = 5$ so that $\lambda - 1 - \ln \lambda > 2$, by Theorem 3.3.3 we have

$$\mathbb{P}\left[t \geq \frac{5 \ln n}{2a}\right] \leq \mathbb{P}[T \geq \lambda \mu] \leq e^{-p^* \mu (\lambda - 1 - \ln \lambda)} = \exp\left(-\frac{2a}{n} \cdot \frac{n \ln n}{2a} \cdot 2\right) = n^{-2}.$$

Thus $t \leq \frac{5 \ln n}{2a}$ with high probability. □

3.4. Analysis of initial phases

Define $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{R}$ to be the sub-populations of agents with roles Main, Clock, $\mathsf{Reserve}$ when they first set `phase` $= 1$. Let $|\mathcal{M}| = m \cdot n, |\mathcal{C}| = c \cdot n, |\mathcal{R}| = r \cdot n$, where $m + c + r = 1$.

The population splitting of Phase 0 will set the fractions $m \approx \frac{1}{2}$, $c \approx \frac{1}{4}$, and $r \approx \frac{1}{4}$. The rules also ensure deterministic bounds on these fractions once all $\mathsf{Role}_{\mathsf{MCR}}$ agents have been assigned. The probability 1 guarantees on subpoplation sizes using this method will be key for a later uniform adaptation of the protocol. We first prove the behavior of the rules used for this initial top level split in Phase 0 between the roles Main and $\mathsf{Role}_{\mathsf{CR}}$.

Lemma 3.4.1. *Consider the reactions*

$$U, U \to S_f, M_f$$

$$S_f, U \to S_t, M_f$$

$$M_f, U \to M_t, S_f$$

starting with n U agents. Let $u = \#U$, $s = \#S_f + \#S_t$, and $m = \#M_f + \#M_t$. This converges to $u = 0$ in expected time at most $2.5 \ln n$ and in $12.5 \ln n$ time with high probability $1 - O(1/n^2)$. Once $u = 0$, $\frac{n}{3} \leq s, m \leq \frac{2n}{3}$ with probability 1, and for any $\epsilon > 0$, $\frac{n}{2}(1-\epsilon) \leq s, m \leq \frac{n}{2}(1+\epsilon)$ with very high probability.

Proof. Let $s_f = \#S_f, s_t = \#S_t, m_f = \#M_f, m_t = \#M_t$.

First we consider the interactions that happen until $u = 2n/3$. Note that while $u \geq 2n/3$, the probability of the first reaction is $\sim \left(\frac{u}{n}\right)^2 \geq \frac{4}{9}$ and the probability of the other two reactions is $\sim 2\left(\frac{u}{n}\right)\left(\frac{m_f+s_f}{n}\right) \leq 2 \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{9}$. Thus until $u = 2n/3$, each non-null interaction is the top reaction with at least probability $1/2$. Then by standard Chernoff Bounds at least a fraction $\frac{1}{2} - \epsilon$ of these reactions are the top reaction, which create a count $s_f + m_f \geq \frac{2}{9}(1-\epsilon)n > \frac{n}{5}$.

Now notice that this count $s_f + m_f$ can never decrease, so we have $s_f + m_f > \frac{n}{5}$ for all future interactions. Now we can use the rate of the second and third reactions to bound the completion time. The probability of decreasing u is at least $2\left(\frac{u}{n}\right)\left(\frac{1}{5}\right)$, so the number of interactions it takes to decrement u is stochastically dominated by a geometric random variable with probability $p = \frac{2u}{5n}$. Then the number of interactions for u to decrease from $\frac{2n}{3}$ down to 0 is dominated by a sum T of geometric random variables with mean

$$\mathbb{E}[T] = \sum_{u=1}^{2n/3} \frac{5n}{2u} = \frac{5n}{2} \sum_{u=1}^{2n/3} \frac{1}{u} \sim \frac{5n}{2} \ln(2n/3) \sim \frac{5}{2} n \ln n.$$

Now we will apply the upper bound of Theorem 3.3.3, using $\mu = \frac{5}{2}n \ln n$, $p^* = \frac{2}{5n}$ (when $u = 1$) and $\lambda = 5$, where $\lambda - 1 - \ln(\lambda) > 2$, so we get

$$\mathbb{P}[T \geq \lambda\mu] \leq \exp(-p^*\mu(\lambda - 1 - \ln\lambda)) \leq \exp\left(-\frac{2}{5n} \cdot \frac{5}{2} n \ln n \cdot 2\right) = n^{-2}.$$

Thus with high probability $1 - O(1/n^2)$, this process converges in at most $\frac{\lambda\mu}{n} = 12.5 \ln n$ parallel time.

Observe that the reactions all preserve the following invariant: $s_f + 2s_t = m_f + 2m_t$. The probability-1 count bounds follow by observing that when $u = 0$ (i.e., we have converged) we have $s_f + s_t + m_f + m_t = n$. Maximizing $s = s_f + s_t$ and minimizing $m = m_f + m + t$ is achieved by setting $s_f = 2n/3, s_t = 0, m_f = 0, m_t = n/3$, and symmetrically for maximizing m.

Assume we have $s > m$ at some point in the execution, so $s_f + s_t > m_f + m_t$. Subtracting the invariant equation gives $-s_t > -m_t$, which implies $s_t < m_t$. Together with the first inequality this implies that $s_f > m_f$. Thus the rate of second reaction is higher than the rate of the third reaction, so it is more likely that the next reaction changing the value $s - m$ decreases it.

By symmetry the opposite happens when $m > s$, so we conclude that the absolute value $|m - s|$ is more likely to decrease than increase. Thus we can stochastically dominate $|m - s|$ by a sum of independent coin flips. The high probability count bounds follow by standard Chernoff bounds. □

Now we can prove bounds on the sizes $|\mathcal{M}| = mn, |\mathcal{C}| = cn, |\mathcal{R}| = rn$ of Main, Clock, and Reserve agents from the population splitting of Phase 0.

Lemma 3.4.2. *For any $\epsilon > 0$, with high probability $1 - O(1/n^2)$, by the end of Phase 0, $|\mathsf{Role}_{\mathsf{MCR}}| = 0$, $\frac{n}{2}(1 - \epsilon) \leq |\mathcal{M}| \leq \frac{n}{2}(1 + \epsilon)$ and $|\mathcal{C}|, |\mathcal{R}| \geq \frac{n}{4}(1 - \epsilon)$. If $|\mathsf{Role}_{\mathsf{MCR}}| = 0$ and Phase 1 initializes without error, then with probability 1, $\frac{1}{3}n \leq |\mathcal{M}| \leq \frac{2}{3}n$, $\frac{1}{6}n \leq |\mathcal{R}| \leq \frac{2}{3}n$, and $2 \leq |\mathcal{C}| \leq \frac{1}{3}n$.*

Proof. The top level of splitting of $\mathsf{Role}_{\mathsf{MCR}}$ into $\mathsf{Role}_{\mathsf{CR}}$ and Main is equivalent to the reactions of Lemma 3.4.1, with $U = \mathsf{Role}_{\mathsf{MCR}}$, $M =$ Main, $S = \mathsf{Role}_{\mathsf{CR}}$, and the f and t subscripts representing the Boolean value of the field `assigned`. Lemma 3.4.1 gives the stated bounds on Main, if there were no further splitting of $\mathsf{Role}_{\mathsf{CR}}$.

Lemma 3.4.1 gives that with high probability, all $\mathsf{Role}_{\mathsf{MCR}}$ are converted to $\mathsf{Role}_{\mathsf{CR}}$ and Main in $12.5 \ln n$ time; we begin the analysis at that point, letting s be the number of $\mathsf{Role}_{\mathsf{CR}}$ agents produced, noting $n/3 \le s \le 2n/3$ with probability 1, and for any ϵ' $\frac{n}{2}(1-\epsilon') \le s \le \frac{n}{2}(1+\epsilon')$ with high probability.

The splitting of $\mathsf{Role}_{\mathsf{CR}}$ into Clock and $\mathsf{Reserve}$ follows a simpler process that we analyze here. Let $r = |\mathcal{R}|$ and $c = |\mathcal{C}|$ at the end of Phase 0. To see the high probability bounds on r and c, we model the splitting of $\mathsf{Role}_{\mathsf{CR}}$ by the reaction $U, U \to R, C$ during Phase 0 and $U \to R$ at the end of Phase 0, since all un-split $\mathsf{Role}_{\mathsf{CR}}$ agents become $\mathsf{Reserve}$ agents upon leaving Phase 0.

The reaction $U, U \to R, C$ reduces the count of U from its initial value s $(= n/2 \pm \epsilon' n/2$ whp) to $\epsilon' s$, with the number of interactions between each reaction when $\#U = l$ governed by a geometric random variable with success probability $O(l^2/n^2)$. Applying Corollary 3.3.4 with $k = s - \epsilon' s$, $p_i = O((i + \epsilon' s)^2/n^2)$ for $i \in \{1, \ldots, k\}$, the reaction $U, U \to R, C$ takes $O(1)$ time to reduce the count of U from its initial value m $(= n/2 \pm \epsilon' n/2$ whp) to $\epsilon' m$ with very high probability. This implies that after $O(1)$ time, r and c are both at least

$$\begin{aligned} s/2 - \epsilon' s = s(1/2 - \epsilon') &\ge (n/2 - \epsilon' n/2)(1/2 - \epsilon') \\ &= n/4 - \epsilon' n/2 - \epsilon' n/4 + (\epsilon')^2 n/2 \\ &> n/4 - \epsilon' n = n/4(1 - 4\epsilon') \end{aligned}$$

with very high probability. Choosing $\epsilon = \epsilon'/4$ gives the high probability bounds on r and c. Thus we require $12.5 \ln n + O(1) \le 13 \ln n$ time, and for appropriate choice of `counter` constant c_0, this happens before the first Clock agent advances to the next phase with high probability.

We now argue the probability-1 bounds. The bound $r \le 2n/3$ follows from $|\mathcal{M}| \ge n/3$. The bound $r \ge n/6$ follows from $|\mathcal{M}| \le 2n/3$, so $\mathsf{Role}_{\mathsf{CR}} \ge n/3$ if no $U, U \to R, C$ splits happen, and the fact that at least half of $\mathsf{Role}_{\mathsf{CR}}$ get converted to $\mathsf{Reserve}$: exactly half by $U, U \to R, C$ and the rest by $U \to R$.

Although the reactions $U, U \to R, C$ and $U \to R$ can produce only a single C, there must be at least two Clock agents for Standard Counter Subroutine to count at all and end Phase 0, so if Phase 1 initializes, $c \ge 2$. The bound $c \le n/3$ follows from the fact that c is maximized when $|\mathcal{M}| = n/3$ and no $U \to R$ reactions happen, i.e., all $2n/3$ $\mathsf{Role}_{\mathsf{CR}}$ agents are converted via $U, U \to R, C$, leading to $c = n/3$. □

We now reason about Phase 1, which has different behavior based on the initial gap g.

Lemma 3.4.3. *If the initial gap $|g| \geq 0.025|\mathcal{M}|$, then we stabilize to the correct output in Phase 2. If $|g| < 0.025|\mathcal{M}|$, then at the end of Phase 1, all agents have* $\texttt{bias} \in \{-1, 0, +1\}$*, and the total count of biased agents is at most $0.03|\mathcal{M}|$. Both happen with high probability $1 - O(1/n^2)$.*

Proof. In the case where all agents enter Phase 1, none are still in $\mathsf{Role}_{\mathsf{MCR}}$, so every non-Main agent has given their $\texttt{bias}$ to a Main agent via Line 2 or Line 5 of Phase 0. Thus the initial gap $g = \sum_{m.\texttt{role}=\mathsf{Main}} m.\texttt{bias}$.

Let $\mu = \lfloor \frac{g}{|\mathcal{M}|} \rceil$ be the average $\texttt{bias}$ among all Main agents, rounded to the nearest integer. By [**135**], we will converge to have all $\texttt{bias} \in \{\mu-1, \mu, \mu+1\}$, in $O(\log n)$ time with high probability $1 - O(1/n^2)$. We use Corollary 1 of [**135**], where the constant $K = O(\sqrt{n})$ and $\delta = \frac{1}{n^2}$. This gives that with probability $1 - \delta$, all $\texttt{bias} \in \{\mu - 1, \mu, \mu + 1\}$ after a number of interactions

$$t \geq (n-1)(2\ln\big(K + \sqrt{n}\big) - \ln(\delta) - \ln(2)) \sim n(2\ln\big(\sqrt{n}\big) + \ln\big(n^2\big)) = 3n\ln n.$$

Thus after time $3\ln n$, all $\texttt{bias} \in \{\mu - 1, \mu, \mu + 1\}$ with high probability $1 - 1/n^2$.

If $|g| > 0.5|\mathcal{M}|$, then $|\mu| \geq 1$, so all remaining biased agents have the majority opinion, and we will stabilize in Phase 2 to the correct majority output.

If $|g| \leq 0.5|\mathcal{M}|$, then $\mu = 0$, so now all $\texttt{bias} \in \{-1, 0, +1\}$. We will use Lemma 3.3.6, with the sets of biased agents $A = \{a : a.\texttt{bias} = +1\}$ and $B = \{b : b.\texttt{bias} = -1\}$, which have initial sizes $|A| = a \cdot n$ and $|B| = b \cdot n$.

In the first case where $0.025|\mathcal{M}| \leq |g| \leq 0.5|\mathcal{M}|$, we have $a - b \geq 0.025m$ (assuming WLOG that A is the majority). Then by Lemma 3.3.6, with high probability $1 - O(1/n^2)$, the count of B becomes 0 in at most time $\frac{5\ln n}{2(a-b)} = \ln n \frac{5}{2 \cdot 0.025m} = \frac{100}{m}\ln n \leq 201\ln n$. With all minority agents eliminated, we will again stabilize in Phase 2 with the correct output.

In the second case where $|g| < 0.025|\mathcal{M}|$, we can use Lemma 3.3.6 with constant $d = b - 0.0025m$. Then even with maximal gap $a - b = 0.025m$, with very high probability in constant time we bring the counts down to $b = 0.0025m$ and $a = 0.0275m$. Thus the total count of biased agents is at most $(0.0025m + 0.0275m)n = 0.03|\mathcal{M}|$.

Since all the above arguments take at most $O(\log n)$ time, for appropriate choice of counter constant c_1, the given behavior happens before the first Clock agent advances to the next phase with high probability. □

3.5. Analysis of main averaging Phase 3

The longest part in the proof is analyzing the behavior of the main averaging Phase 3. The results of this section culminate in the following two theorems, one for the case of an initial tie, and the other for an initial biased distribution.

In the case of an initial tie, we will show that all biased agents have minimal $\texttt{exponent} = -L$ by the end of the phase.

THEOREM 3.5.1. *If the initial configuration was a tie with gap 0, then by the end of Phase 3, all biased agents have* $\texttt{exponent} = -L$*, with high probability* $1 - O(1/n^2)$.

Note that the where all biased agents have $\texttt{exponent} = -L$ gives a stable configuration in the next Phase 4, with all agents having $\texttt{output} = \mathsf{T}$. Thus from Theorem 3.5.1, we conclude that from an intial tie, the protocol will stabilize in Phase 4 with high probability. Conversely, we have already observed that this configuration can only be reached in the case of a tie, since the sum of all biased agents would bound the magnitude of the initial gap $|g| < 1$. Thus in the other case of a majority initial distribution, the agents will proceed through to Phase 5 with probability 1.

In this majority case, we will show that a large majority of the Main agents have $\texttt{opinion}$ set to the majority opinion and $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$ is in a consecutive range of 3 possible exponents, where the value $-l$ depends on the initial distribution. In addition, we show the upper tail above this exponent $-l$ is very small.

THEOREM 3.5.2. *Assume the initial gap* $|g| < 0.025|\mathcal{M}|$. *Let the exponent* $-l = \lfloor \log_2(\frac{g}{0.4|\mathcal{M}|}) \rfloor$. *Let* $i = sign(g)$ *be the majority opinion and* M *be the set of all agents with* $\texttt{role} = \mathsf{Main}$, $\texttt{opinion} = i$, $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$. *Then at the end of Phase 3,* $|M| \geq 0.92|\mathcal{M}|$ *with high probability* $1 - O(1/n^2)$.

In addition, the total mass above exponent $-l$ *is* $\mu_{(>-l)} = \sum\limits_{a.\texttt{exponent} > -l} |a.\texttt{bias}| \leq 0.002|\mathcal{M}|2^{-l}$, *and the total minority mass is* $\beta_{-} = \sum\limits_{a.\texttt{opinion} = -i} |a.\texttt{bias}| \leq 0.004|\mathcal{M}|2^{-l}$.

Note the assumption of small initial gap g that we get from Phase 1 is just for convenience in the proof, to reason uniformly about the base case behavior at hour $h = 0$ for our inductive argument. The rules of Phase 3 would also work as intended with even larger initial gaps, just requiring a variant of the later analysis to acknowledge that the initial `exponent` $= 0$ is quite close to the final range `exponent` $\in \{-l, -(l+1), -(l+2)\}$.

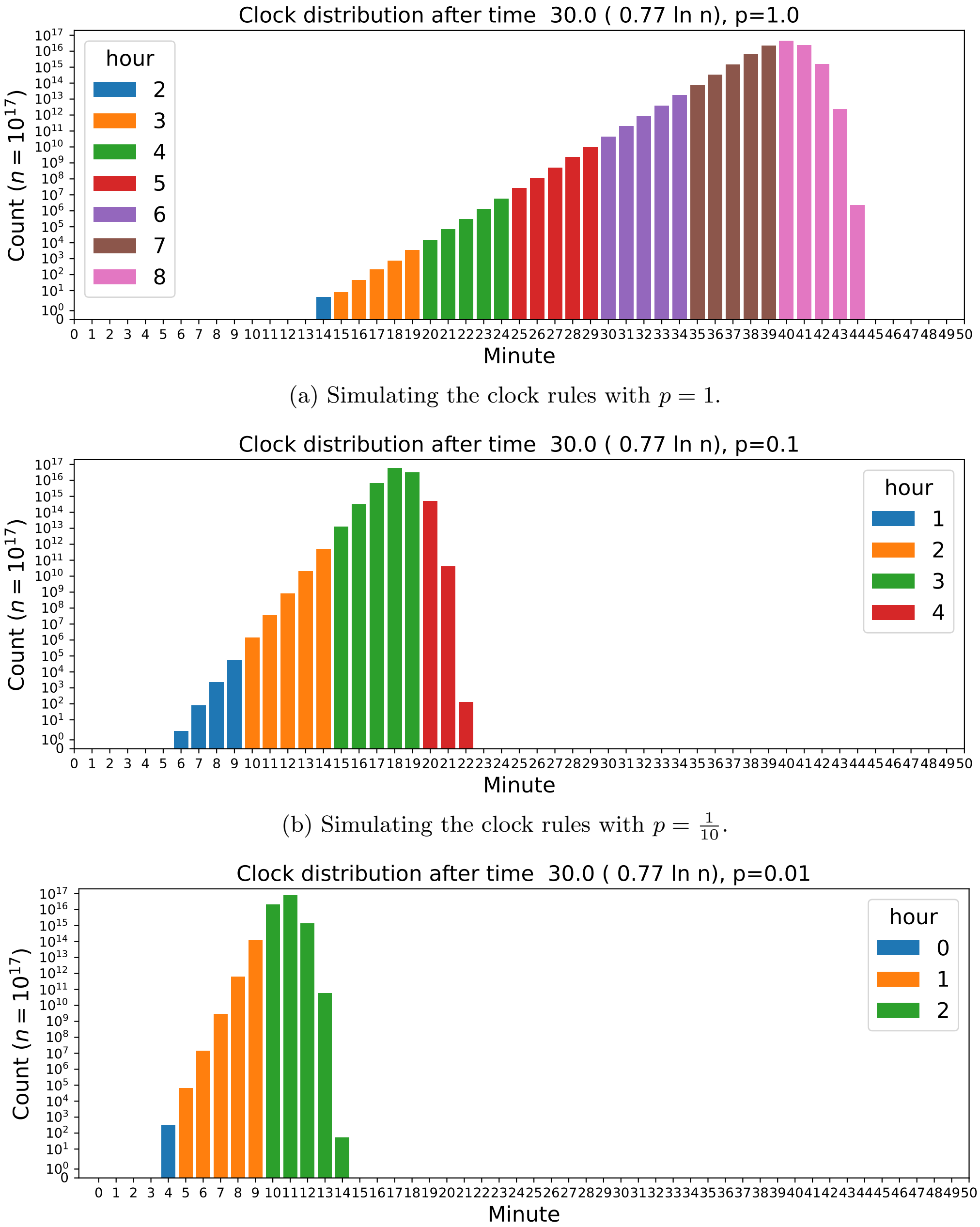
3.5.1. Clock synchronization theorems. We will first consider the behavior of the Clock agents in Phase 3. The goal is to show their `minute` fields remain tightly concentrated while moving from 0 up to kL, summarized in Theorem 3.5.9. See Fig. 3.8 for simulations of the clock distribution. The back tail behind the peak of the `minute` distribution decays exponentially, since each agent is brought ahead by epidemic at a constant rate and thus their counts each decay exponentially. The front part of the distribution decays much more rapidly. With a fraction f of agents at `minute` $= i$, the rate of the drip reaction is proportional to pf^2. This repeated squaring leads to the concentrations at the leading minutes decaying doubly exponentially. The rapid decay is key to showing that very few Clock agents can get very far ahead of the rest.

While our algorithm runs for only $O(\log n)$ minutes, the clock behavior we require can be made to continue for $O(n^c)$ minutes for arbitrary c. Our proofs rely on induction on minutes, and all results in this section hold with very high probability. Thus, we can take a union bound over polynomially many minutes and still keep the very high probability guarantees. If the clock runs for a superpolynomial number of minutes, the results of this section no longer hold.

We start by consider an entire population running the clock transitions (lines 1-5 of Phase 3), so $|\mathcal{C}| = n$. The following lemmas describe the behavior of just this clock protocol. In our actual protocol, the clock agents are a subpopulation $|\mathcal{C}| = c \cdot n$, and these clock transitions only happen when two clock agents meet with probability $\binom{|\mathcal{C}|}{2} / \binom{n}{2} \sim \frac{1}{c^2}$. This more general situation is handled in later theorems applying to our exact protocol.

Definitions of values used in subsequent lemma statements. Throughout this section, we reference the following quantities. For each minute i and parallel time t, define $c_{\geq i}(t) = |\{c : c.\texttt{minute} \geq i\}| / |\mathcal{C}|$ to be the fraction of clock agents at minute i or beyond at time t. Then define $t^{+}_{\geq i} = \min\{t : c_{\geq i}(t) > 0\}$, $t^{0.1}_{\geq i} = \min\{t : c_{\geq i}(t) \geq 0.1\}$ and $t^{0.9}_{\geq i} = \min\{t : c_{\geq i}(t) \geq 0.9\}$ to be the first times where this fraction becomes positive, hits 0.1 and hits 0.9.

We first show that the $c_{\geq i+1}$ is significantly smaller than $c_{\geq i}$ while both are still increasing, so the front of the clock distribution decays very rapidly. In our argument, we consider three types of reactions that change the counts at minutes i or above:



(a) Simulating the clock rules with $p = 1$.

(b) Simulating the clock rules with $p = \frac{1}{10}$.

(c) Simulating the clock rules with $p = \frac{1}{100}$.

FIGURE 3.8. Simulating the clock rules on a large population of size $n = 10^{17}$, with $k = 5$ minutes per hour and multiple values of p. The `minute` distribution gets tighter for smaller values of p. To make the `hour` distribution tighter, we can also simply make k larger. The full evolution of these distributions can be seen in the example notebook [1], along with the code that ran the large simulation to generate the data.

(i) Drip reactions $i, i \to i, (i+1)$ (Line 3 in Phase 3)

(ii) Epidemic reactions $j, k \to j, j$, for any minutes j, k with $k \leq i < j$ (Line 5 in Phase 3)

(iii) Epidemic reactions $j, k \to j, j$, for any minutes j, k with $k < i \leq j$ (Line 5 in Phase 3)

Note we ignore the drip reactions $(i-1), (i-1) \to (i-1), i$, which would only help the argument, but we do not have guarantees on the count at minute $i-1$.

We will try to show the relationship pictured in Fig. 3.8, that $c_{\geq (i+1)}(t) \approx c_{\geq i}(t)^2$. One challenge here is that this relationship will no longer hold with high probability for small values of $c_{\geq i}$. For example, if $c_{\geq i}(t) = n^{-1/2-\epsilon}$, the desired relationship would require $c_{\geq i+1}(t) < \frac{1}{n}$, meaning the count above minute i is 0. A drip reaction at minute i could happen with non-negligible probability $\approx n^{-2\epsilon}$.

To handle this difficulty, we define the set of **early drip agents** $d_{\geq i+1}(t)$, the set of agents that moved above minute i via a drip reaction at a time $\leq t$ when $c_{\geq i}(t) < n^{-0.45}$, or that were brought above minute i via an epidemic reaction with another early drip agent in $d_{\geq i+1}$. Note that the latter group of agents can move to minute $\geq i+1$ **after** time $t^{>0}_{\geq i+1}$. Thus, this set represents the effect of any drip reactions that happen before $c_{\geq i}(t)$ grows large enough for our large deviation bounds to work. We then define $d_{\geq i+1}(t) = |d_{\geq i+1}(t)|/|\mathcal{C}|$ to be the fraction of early drip agents. The set of agents above minute i that comprise the fraction $c_{\geq (i+1)}$ is then partitioned into the early drip agents that comprise $d_{\geq i+1}$, and the rest. By first ignoring these early drip agents $d_{\geq i+1}$, we can show that the rest of $c_{\geq (i+1)}$ stays small compared to $c_{\geq i}$.

Lemma 3.5.3. *With very high probability, if $n^{-0.45} \leq c_{\geq i}(t) \leq 0.1$, then $c_{\geq i+1}(t) \leq 0.9pc_{\geq i}(t)^2 + d_{\geq i+1}(t)$.*

Proof. The proof will proceed by induction on time t. As a base case, for all t such that $c_{\geq i}(t) < n^{-0.45}$, the statement holds simply by definition of $d_{\geq i+1}(t)$, which is equal to $c_{\geq i+1}(t)$.

For the inductive step, to show the relationship $c_{\geq (i+1)}(t) < 0.9pc_{\geq i}(t)^2 + d_{\geq i+1}(t)$ holds at time t, we will use the inductive hypothesis that $c_{\geq (i+1)}(t-0.1) < 0.9pc_{\geq i}(t-0.1)^2 + d_{\geq i+1}(t-0.1)$. Let $x(t) = c_{\geq i}(t)$ and $y(t) = c_{\geq i+1}(t) - d_{\geq i+1}(t)$, so we need to show $y(t) < 0.9px(t)^2$.

We first bound how much $x(t)$ grows by epidemic reactions, in order to show $x(t-0.1) < 0.84x(t)$. Using Lemma 3.3.5, the expected amount of time for an epidemic to grow from fraction $0.84x$ to x

is

$$\frac{1}{2}\left[\ln(x) - \ln(0.84x) + \ln\left(\frac{1-0.84x}{1-x}\right)\right] \leq \frac{1}{2}\left[-\ln(0.84) + \ln\left(\frac{1-0.84\cdot 0.1}{1-0.1}\right)\right] < 0.096,$$

where we used the fact that $\frac{1-0.84x}{1-x}$ is nondecreasing and $x(t) \leq 0.1$.

The minimum probability $p^* = \Theta(x(t)) = \Omega(n^{-0.45})$, and the expected number of interactions $\mu = \Theta(n)$. So the application of Theorem 3.3.3 will give probability $1 - e^{-\Omega(n^{0.55})}$ for the epidemic to have grown enough within time 0.1. Thus $x(t-0.1) < 0.84x(t)$ with very high probability.

Next we bound how much $y(t)$ grows, by both epidemic reactions and drip reactions. The probability of a drip reaction is at most $px(t)^2$, so the expected number of drip reactions in time 0.1 is $0.1px(t)^2$. A standard Chernoff bound then gives that there are at most $0.11px(t)^2$ drip reactions with very high probability. We assume in the worst case all these drip reactions happen at time $t-0.1$, and then y grows by epidemic starting from $z = y(t-0.1) + 0.11px(t)^2$.

By Lemma 3.3.5, the expected amount of time for an epidemic to grow from fraction z to $1.23z$ is

$$\frac{1}{2}\left[\ln(1.23z) - \ln(z) + \ln\left(\frac{1-z}{1-1.23z}\right)\right] \geq \frac{1}{2}\ln(1.23) > 0.103.$$

The minimum probability $p^* = \Omega(x(t)^2) = \Omega(n^{-0.9})$, and the expected number of interactions $\mu = \Theta(n)$. So the application of Theorem 3.3.3 will give probability $1 - e^{-\Omega(n^{0.1})}$. Thus with very high probability

$$\begin{aligned} y(t) &\leq 1.23[y(t-0.1) + 0.11px(t)^2] \\ &\leq 1.23[0.9px(t-0.1)^2 + 0.11px(t)^2] \\ &\leq 1.23[0.9p(0.84x(t))^2 + 0.11px(t)^2] < 0.9px^2. \end{aligned}$$ □

In order to bound $c_{\geq(i+1)}(t)$ as a function of $c_{\geq i}(t)$ only (Theorem 3.5.5), we need to bound how large the set $D_{\geq i+1}$ of early drip agents can be. The strategy will be to show there is not enough time for the set $D_{\geq i+1}$ to grow very large starting from, $t^{+}_{\geq i+1}$, the first time any agent appears at minute $\geq i+1$ (i.e., the first drip reaction into minute $i+1$), because there are only $O(\log\log n)$ minutes in the front tail (see Fig. 3.8), so the time between $t^{+}_{\geq i+1}$ and $t^{0.1}_{\geq i+1}$ is $O(\log\log n)$.

We will first need an upper bound on how long it takes the clock to move from one minute to the next.

Lemma 3.5.4. $t^{0.1}_{\geq i+1} - t^{0.1}_{\geq i} \leq 2.11 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$ *with very high probability.*

Proof. First we argue that $c_{\geq i+1}(t^{0.1}_i + \frac{1}{2}) > 0.0045p$. If not, the count at minute i, $c_{\geq i}(t) - c_{\geq i+1}(t) > 0.1 - 0.0045 = 0.0955$ for all $t^{0.1}_i < t < t^{0.1}_i + 0.5$. Then, the probability of a drip reaction is at least $0.0955^2 p > 0.0091p$. By standard Chernoff bounds, we then have that in time $\frac{1}{2}$, there are at least $0.0045p$ drip reactions with very high probability. Thus $c_{\geq i+1}(t^{0.1}_i + \frac{1}{2}) > 0.0045p$ from just those drip reactions alone.

Now we argue that the amount of time it takes for epidemic reactions to bring $c_{\geq(i+1)}$ up to 0.1. By Lemma 3.3.5, the expected amount of time for an epidemic to grow from fraction $0.0045p$ to 0.1 is

$$\begin{aligned}\frac{1}{2}[\ln(0.1) - \ln(0.0045p) + \ln(1-0.0045p) - \ln(0.9)] &< \frac{\ln(0.1) - \ln(0.0045) - \ln(0.9)}{2} - \frac{\ln p}{2} \\ &< 1.603 - \frac{\ln p}{2}.\end{aligned}$$

As long as $p = \Theta(1)$, then the minimum probability $p^* = \Theta(p)$ is constant, and by Lemma 3.3.5, the epidemic takes at most time $1.61 - \frac{\ln p}{2}$ with very high probability.

In total, we then get that $t^{0.1}_{i+1} - t^{0.1}_i \leq 0.5 + 1.61 - \frac{\ln p}{2} = 2.11 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$ with very high probability. □

Proving that the set $D_{\geq i+1}$ remains small will let us prove the main theorem about the front tail of the clock distribution:

Theorem 3.5.5. *With very high probability, if $n^{-0.4} \leq c_{\geq i}(t) \leq 0.1$, then $c_{\geq(i+1)}(t) < pc_{\geq i}(t)^2$.*

Proof. The proof will proceed by induction on the minute i, where the base case is vacuous because $c_{\geq 0}(0) = 1$, so $c_{\geq 0}(t) > 0.1$ for all times t.

The inductive hypothesis will use two claims. The first is that the time $t^{0.1}_{\geq i} - t^{+}_{\geq i} = O(\log\log n)$ for minute i. The second is that $d_{\geq i+1}(t^{0.1}_{\geq i}) = O(n^{-0.85})$. Note that using this second claim along with Lemma 3.5.3 proves the Theorem statement at minute i: when $n^{-0.4} \leq c_{\geq i}(t) \leq 0.1$, by Lemma 3.5.3 we have

$$c_{\geq(i+1)}(t) < 0.9pc_{\geq i}(t)^2 + d_{\geq i+1}(t) \leq pc_{\geq i}(t)^2,$$

because $c_{\geq i}(t)^2 \geq n^{-0.8}$, so the $d_{\geq i+1}$ term is negligible for sufficiently large n.

We will prove the first claim in two parts. First we argue that $t^{0.1}_{\geq i} - t^{+}_{\geq i} = O(\log\log n)$, because the width of the front tail is at most $2\log\log n$. We will show that at $t^{+}_{\geq i}$, when the first agent arrives at $\texttt{minute} = i$, we already have $c_{\geq j}(t^{+}_{\geq i}) \geq 0.1$, where $j = i - 2\log\log n$ (for $i < 2\log\log n$, we just have $j = 0$ and there is nothing to show because the width of the front tail can be at most i). First we move $\log\log n$ levels back to $k = i - \log\log n$, to show $c_{\geq k}(t^{+}_{\geq i}) \geq n^{-0.4}$.

Assume, for the sake of contradiction that $c_{\geq k}(t^{+}_{\geq i}) < n^{-0.4}$. Between $t^{+}_{\geq k}$ and $t^{n^{-0.4}}_{\geq k}$, consider the number of drips that happen from levels $k+1$ and above. By the inductive hypothesis, we have $c_{\geq k+1}(t) \leq p c_{\geq k}(t)^2 < n^{-0.8}$ during this whole time. Thus in any interaction of this period the probability of a drip above level $k+1$ is at most $p \cdot (n^{-0.8})^2 \leq n^{-1.6}$. The interval length is $t^{n^{-0.4}}_{\geq k} - t^{+}_{\geq k} = O(\log\log n)$ by the inductive hypothesis, so the probability of having at least $\log\log n$ drips during this interval is at most

$$\binom{O(n\log\log n)}{\log\log n}(n^{-1.6})^{\log\log n} \leq \left(\frac{O(n\log\log n)}{n^{1.6}}\right)^{\log\log n} = n^{-\omega(1)}.$$

This implies $c_{\geq i}(t^{n^{-0.4}}_{\geq k}) = 0$ with very high probability.

Thus with very high probability, we already have $c_{\geq k}(t^{+}_{\geq i}) \geq n^{-0.4}$. Then we can iterate the inductive hypothesis $c_l(t) \leq p(c_{l-1}(t))^2$ for the $\log\log n$ minutes $l = k, k-1, \ldots, j$, which implies $c_{\geq j}(t^{+}_{\geq i}) \geq 0.1$. Now that we have shown the width of the front tail is at most $2\log\log n$, we use Lemma 3.5.4, which shows that each minute takes $O(1)$ time, so it takes $O(\log\log n)$ time between $t^{+}_{\geq i}$ when the first agent gets to $\texttt{minute} = i$ and $t^{0.1}_{\geq i}$, when the fraction at $\texttt{minute} \geq i$ reaches 0.1.

Now we prove the second claim, arguing that in this $O(\log\log n)$ time, $d_{\geq i+1}$ can grow to at most $O(n^{-0.85})$. By definition of $d_{\geq i+1}$, the drip reactions that increase $d_{\geq i+1}$ happen with probability at most $p(n^{-0.45})^2 = pn^{-0.9}$. By a standard Chernoff bound, the number of drip reactions in $O(\log\log n)$ time is $O(\log\log n \cdot n^{-0.9}) = O(n^{-0.89})$. Then we assume in the worst case this maximal number of drip reactions happen, and then $d_{\geq i+1}$ can grow by epidemic. By Lemma 3.3.5, the time for an epidemic to grow from fraction $O(n^{-0.89})$ to $\Omega(n^{-0.85})$ is $\Omega(\log n) > O(\log\log n)$ with very high probability. Thus, with very high probability, we still have $d_{\geq i+1} = O(n^{-0.85})$ within $O(\log\log n)$ time. □

Now the relationship proven in Theorem 3.5.5 implies that $c_{\geq (i+1)}(t^{0.1}_{\geq i}) \leq 0.01p$. We can now use this bound to get lower bounds on the time required to move from one minute to the next.

Lemma 3.5.6. *With very high probability,* $t_{i+1}^{0.1} - t_i^{0.1} \geq \frac{1}{2}\ln\left(1 + \frac{2}{9p}\right) - 0.01$.

Proof. We start at time $t_{\geq i}^{0.1}$, where by Theorem 3.5.5 we have $c_{\geq i+1}(t_{\geq i}^{0.1}) \leq 0.01p$ with very high probability.

Define $x = x(t) = c_{\geq i}(t)$ and $y = y(t) = c_{\geq i+1}(t)$. The number of interactions for $Y = yn$ to increase by 1 is a geometric random variable with mean $\frac{1}{\mathbb{P}[(i)]+\mathbb{P}[(ii)]}$, where the drip reaction (i) has probability $\mathbb{P}[(i)] \sim p(x-y)^2 \leq p(1-y)^2$ and the epidemic reaction (ii) has probability $\mathbb{P}[(ii)] \sim 2y(1-y)$. Assuming in the worst case that $y(t_i^{0.1}) = 0.01p$, then the number of interactions $T = (t_{i+1}^{0.1} - t_i^{0.1})n$ for Y to increase from $0.1pn$ to $0.1n$ is a sum of independent geometric random variables with mean

$$
\begin{aligned}
\mathbb{E}[T] &\geq \sum_{Y=0.01pn}^{0.1n-1} \frac{1}{p(1-Y/n)^2 + 2(Y/n)(1-Y/n)} \sim n\int_{0.01p}^{0.1} \frac{dy}{p(1-y)^2 + 2y(1-y)} \\
&= n\int_{0.01p}^{0.1} \frac{dy}{(1-y)(p+(2-p)y)} = n\int_{0.01p}^{0.1} \frac{1/2}{1-y} + \frac{1-p/2}{p+(2-p)y}dy \\
&= n\left[-\frac{1}{2}\ln(1-y) + \frac{1}{2}\ln\left(p+(2-p)y\right)\right]_{0.01p}^{0.1} \\
&= \frac{n}{2}[-\ln(0.9) + \ln(1-0.01p) + \ln(p+0.1(2-p)) - \ln(p+0.01p(2-p))] \\
&= \frac{n}{2}\left[-\ln(0.9) + \ln\left(\frac{1-0.01p}{p-0.01p^2+0.02p}\right) + \ln(0.9p+0.2)\right] \\
&\geq \frac{n}{2}\left[-\ln(0.9) + \ln\left(\frac{0.9p+0.2}{1.02p}\right)\right] \geq \frac{n}{2}\left[-0.02 + \ln\left(1 + \frac{2.04}{9p}\right)\right]
\end{aligned}
$$

Note that the probability in the geometric includes the term $p(1-y)^2 \geq 0.81p$ since $y \leq 0.1$, thus the minimum geometric probability $p^* \geq 0.81p$ is bounded by a constant. Also the mean $\mu = \Theta(n)$ so by Corollary 3.3.4, $\mathbb{P}[T \leq n(-0.01 + \frac{1}{2}\ln\left(1+\frac{2}{9p}\right)] \leq \exp(-\Theta(n))$, so the time $t_{i+1}^{0.1} - t_i^{0.1} = T/n \geq \frac{1}{2}\ln\left(1+\frac{2}{9p}\right) - 0.01$ with very high probability. □

The worst case upper bound for the dripping probability, $p(1-y)^2$, used in the above Lemma, is weakest when $p = 1$. We now give a special case lower bound that is stronger in the deterministic case with $p = 1$.

Lemma 3.5.7. *With very high probability,* $t_{i+1}^{0.1} - t_i^{0.1} \geq 0.45$.

Proof. We assume in the worst case that $p = 1$. Then by Theorem 3.5.5, we have $c_{\geq i+1}(t_i^{0.1}) \leq 0.01$. This initial fraction will grow by epidemic to at most $0.01 \cdot e^{2\cdot 0.45}(1+\epsilon)$ with very high probability by Lemma 3.3.5. We must also consider all agents that later make it to minute $i+1$ by a drip reaction, and how much they grow by epidemic.

By time $t_i^{0.1}+s$, the fraction $c_{\geq i}$ could have increased to at most $0.1+s$, since at most 1 agent can increase its minute in each interaction. Then the probability of a drip reaction at this time is at most $(0.1+s)^2$. By standard Chernoff bounds, there will be at most $(1+\epsilon)(0.1+s)^2 n^{0.5}$ drip reactions in the next $n^{0.5}$ interactions with very high probability. Then by Lemma 3.3.5, these agents can grow by epidemic by at most a factor $(1+\epsilon)e^{2\cdot(0.45-s)}$ by time $t_i^{0.1}+0.45$, with very high probability. Summing over consecutive groups of $n^{0.5}$ interactions at time $i\cdot\frac{n^{0.5}}{n}$ and using the union bound, we get a total bound

$$c_{\geq i+1}(t_i^{0.1}+0.45) \leq 0.01\cdot e^{2\cdot 0.45}(1+\epsilon) + \sum_{i=0}^{0.45n^{0.5}} (1+\epsilon)(0.1+n^{-0.5}i)^2 n^{0.5}\cdot e^{2\cdot(0.45-n^{-0.5}i)}$$

$$\sim (1+\epsilon)\cdot e^{0.9}\left[0.01+\int_0^{0.45}(1+s)^2 e^{-2s}ds\right] \leq 0.45.$$

□

We now summarize all bounds on the length of a clock minute in a single theorem:

THEOREM 3.5.8. *Let* $t_{\geq i+1}^{0.1} - t_{\geq i}^{0.1}$ *be the time between when a fraction* 0.1 *of agents have* `minute` $\geq i$ *and when a fraction* 0.1 *of agents have* `minute` $\geq i+1$. *Then with very high probability,*

$$\max\left(0.45, \frac{1}{2}\ln\left(1+\frac{2}{9p}\right)-0.01\right) \leq t_{\geq i+1}^{0.1} - t_{\geq i}^{0.1} \leq 2.11 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$$

These bounds are shown in Fig. 3.9, along with sampled minute times from simulation. This suggests the actual time per minute is roughly $0.75+\frac{1}{2}\ln\left(\frac{1}{p}\right)$.

We can now build from these theorems to get bounds on the values of `hour`. For a clock agent a, define $a.\texttt{hour} = \lfloor\frac{a.\texttt{minute}}{k}\rfloor$. Define $\text{start}_h = \min\left(t : |\{a : a(t).\texttt{hour} \geq h\}| \geq 0.9|\mathcal{C}|\right)$ be the first time when the fraction of clock agents at hour h or beyond reaches 0.9 and $\text{end}_h = \min\left(t : |\{a : a(t).\texttt{hour} > h\}| \leq 0.001|\mathcal{C}|\right)$ be the first time when the fraction of clock agents beyond hour h reaches 0.001. Define the **synchronous hour h** to be the parallel time interval $[\text{start}_h, \text{end}_h]$, i.e. when fewer than 0.1% are in any hour beyond h and at least (90 - 0.1) = 89.9% are in hour h. Note that if $\text{end}_h < \text{start}_h$ then this interval is empty, but we show this happens with low probability. We choose the threshold $0.001|\mathcal{C}|$ to be a sufficiently small constant for later proofs.

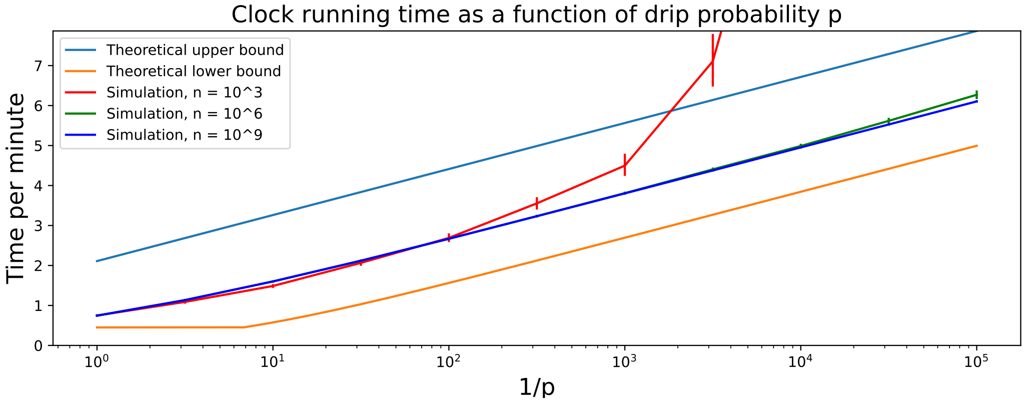


FIGURE 3.9. The upper and lower bounds from Theorem 3.5.8 for the time of one clock minute, along with samples from simulation. For each value of n, 100 minute times were sampled, taking $t_{i+1}^{0.1} - t_i^{0.1}$ for $i = 9, \dots, 18$ over 10 independent trials. All our proofs assume p is constant, and for any fixed value of p, will only hold for sufficiently large n. The case $n = 10^3$ shows that when $p = O(1/n)$, the bounds no longer hold. This is to be expected because the expected number of drips becomes too small for large deviation bounds to still hold.

Recall $c = |\mathcal{C}|/n$ is the fraction of clock agents and k is the number of minutes per hour.

THEOREM 3.5.9. *Consider a fraction c of agents running the clock protocol, with $p = 1$. Then for every synchronous hour h, its length* $\mathrm{end}_h - \mathrm{start}_h \geq \frac{1}{c^2}[0.45k - 3.1]$, *with very high probability. The time between consecutive synchronous hours* $\mathrm{start}_{h+1} - \mathrm{start}_h \leq \frac{1}{c^2}[2.11k + 2.2]$ *with very high probability.*

PROOF. Note that the previous lemmas assumed $|\mathcal{C}| = n$, so the entire population was running the clock algorithm. In reality, we have a fraction $c = |\mathcal{C}|/n$ of clock agents. The reactions we considered only happen between two clock agents, which interact with probability $\sim c^2$. Thus we can simply multiply the bounds from our lemmas by the factor $\frac{1}{c^2}$ to account for the number of regular interactions for the requisite number of clock interactions to happen, which is very close to its mean with very high probability by standard Chernoff bounds.

Because our definition references time $t_{\geq i}^{0.9}$ when the fraction reaches 0.9, we will first bound the time it takes an epidemic to grow from 0.1 to 0.9. By Lemma 3.3.5, this takes parallel time t, where

$$\mathbb{E}[t] \sim \frac{\ln(0.9) - \ln(1 - 0.9) - \ln(0.1) + \ln(1 - 0.1)}{2} = \ln(9) < 2.2.$$

Since $\ln(9)(1 + \epsilon) < 2.2$, this completes within parallel time 2.2 with very high probability.

Since $c.\mathtt{hour} = h \iff hk \leq c.\mathtt{minute} < (h+1)k$, the times $\text{start}_h = t_{hk}^{0.9}$ and $\text{end}_h = t_{hk}^{0.001}$. For the upper bound, by Lemma 3.5.4 we have $t_{i+1}^{0.1} - t_i^{0.1} \leq 2.11/c^2$ with very high probability. We start at $t_{hk}^{0.9} \geq t_{hk}^{0.1}$, and sum over the k minutes in hour h, then add at most time $\frac{2.2}{c^2}$ between $t_{(h+1)k}^{0.1}$ and $t_{(h+1)k}^{0.1}$. This gives that $\text{start}_{h+1} - \text{start}_h \leq \frac{1}{c^2}[2.11k + 2.2]$ with very high probability.

For the lower bound, by Theorem 3.5.5, at time $t_{(h+1)k-2}^{0.1}$, we have $c_{\geq (h+1)k} \leq (0.1^2)^2 = 10^{-4} < 0.001$, so $t_{(h+1)k-2}^{0.1} < \text{end}_h$. Then $\text{start}_h \leq t_{hk}^{0.1} + \frac{2.2}{c^2}$. Using Lemma 3.5.7, we have $t_{i+1}^{0.1} - t_i^{0.1} \leq 0.45/c^2$ with very high probability, for each $i = hk, \ldots, hk+k-3$. All together, this gives $\text{end}_h - \text{start}_h \geq \frac{1}{c^2}[0.45(k-2) - 2.2] = \frac{1}{c^2}[0.45k - 3.1]$. □

We will set $k = 45$ to give us sufficiently long synchronous hours for later proofs. Since every hour takes constant time with very high probability, all L hours will finish within $O(\log n)$ time.

We finally show one more lemma concerning how the Clock agents affect the Main agents. The Clock agents will change the hour of the $\mathcal{O}$ agents via Line 9 of Phase 3. There are a small fraction $0.001|\mathcal{C}|$ of Clock agents that might be running too fast and thus have a larger hour than the synchronized hour. We must now show these agents are only able to affect a small fraction of the Main agents. Intuitively, the Clock agents with $\mathtt{hour} > h$ bring up the hour of both $\mathcal{O}$ agents and other Clock agents. The following lemma will show they don't affect too many Main agents before also bringing in a large number of Clock agents.

We will redefine $c_{>h} = c_{>h}(t) = |\{c : c.\mathtt{hour} > h\}|/|\mathcal{C}|$ to be the fraction of Clock agents beyond hour h, and similarly $m_{>h} = m_{>h}(t) = |\{m : m.\mathtt{hour} > h\}|/|\mathcal{M}|$ to be the fraction of Main agents beyond hour h.

Lemma 3.5.10. *For all times $t \leq \text{end}_h$, we have $m_{>h}(t) \leq 1.2c_{>h}(\text{end}_h) = 0.0012$ with very high probability.*

Proof. We have $c_{>h}(\text{end}_h) = 0.001$ by the definition of synchronous hour h. Thus it suffices to show that $m_{>h}(t) \leq 1.2c_{>h}(\text{end}_h)$.

Recall $c = |\mathcal{C}|/n$ and $m = |\mathcal{M}|/n$, so $c \cdot c_{>h}$ and $m \cdot m_{>h}$ are rescaled to be fractions of the whole population.

We will assume in the worst case that every Main agent can participate in the clock update reaction

$$C_h, M_j \to C_h, M_h \text{ where } h > j,$$

so the probability of clock update reaction is $2c \cdot c_{>h} \cdot m(1 - m_{>h})$. Among the Clock agents, we will now only consider the epidemic reactions $C_h, C_j \to C_h, C_h$ between agents in different hours $h > j$, so the probability of the clock epidemic reaction is $2c^2 \cdot c_{>h}(1 - c_{>h})$.

We use the potential $\Phi(t) = m_{>h}(t) - 1.1 \cdot c_{>h}(t)$. Note that initially $\Phi(0) = 0$ since both terms are 0. The desired inequality is $\Phi(\mathrm{end}_h) \leq 0.1 c_{>h}(\mathrm{end}_h) = 0.0001$, and we will show this holds with very high probability by Azuma's Inequality because Φ is a supermartingale. The clock update reaction increases Φ by $\frac{1}{|\mathcal{M}|} = \frac{1}{mn}$. The clock epidemic reaction decreases Φ by $\frac{1.1}{|\mathcal{C}|} = \frac{1.1}{cn}$. This gives an expected change

$$\begin{aligned}
\mathbb{E}[\Phi(t + 1/n) - \Phi(t)] &= \frac{1}{mn}[2c \cdot c_{>h} \cdot m(1 - m_{>h})] - \frac{1.1}{cn}\left[2c^2 \cdot c_{>h}(1 - c_{>h})\right] \\
&= \frac{2c \cdot c_{>h}}{n}[(1 - m_{>h}) - 1.1(1 - c_{>h})] \\
&\leq \frac{2c \cdot c_{\geq i}}{n}[1 - 1.1(0.999)] < 0.
\end{aligned}$$

Thus the sequence $\left(\Phi_j = \Phi(\frac{j}{n})\right)_{j=0}^{n \cdot \mathrm{end}_h}$ is a supermartingale, with bounded differences $|\Phi_{j+1} - \Phi_j| \leq \max\left(\frac{1}{mn}, \frac{r}{cn}\right) = O(\frac{1}{n})$. So we can apply Azuma's Inequality (Theorem 3.3.2) to conclude

$$\mathbb{P}[\Phi_{n \cdot \mathrm{end}_h} \geq \delta] \leq \exp\left(-\frac{\delta^2}{2\sum_{j=0}^{n \cdot \mathrm{end}_h} O(\frac{1}{n})^2}\right) = \exp\bigl(-O(n)\delta^2\bigr),$$

since by Theorem 3.5.9, we have time $\mathrm{end}_h = O(1)$ with very high probability. Thus we can choose $\delta = 0.1 c_{>h}(\mathrm{end}_h) = 0.0001$ to conclude that $m_{>h}(\mathrm{end}_h) \leq 1.2 c_{>h}(\mathrm{end}_h)$ with very high probability $1 - \exp(-\Omega(n)) = 1 - O(n^{-\omega(1)})$.

The lemma statement for times $t \leq \mathrm{end}_h$ simply follows from the monotonicity of the value $m_{>h}$, since agents only decrease their `hour` field. □

3.5.2. Phase 3 exponent dynamics. We now analyze the behavior of the Main agents in Phase 3. We will show their `exponent` fields stay roughly synchronized with the `hour` of the Clock agents, decreasing from 0 toward $-L$. We first use the above results on the clock to conclude that during synchronous hour h, the tail of either $\mathcal{O}$ agents with `hour` $> h$ or biased agents with `exponent` $< -h$ is sufficiently small.

Lemma 3.5.11. *Until time* end_h*, the count* $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > h\} \cup \{b : b.\texttt{exponent} < -h\}| \leq 0.0024|\mathcal{M}|$ *with very high probability.*

Proof. By Lemma 3.5.10, the count of $\mathcal{O}$ agents that have been pulled up by a Clock agent to $\texttt{hour} > h$ is at most $0.0012|\mathcal{M}|$ with very high probability. Each of these agents could participate in a split reaction that results in two biased agents with $\texttt{exponent} < -h$, increasing the total count of $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > h\} \cup \{b : b.\texttt{exponent} < -h\}|$ by 1. Thus this total count can be at most twice as large: $\leq 0.0024|\mathcal{M}|$. □

Our main argument will proceed by induction on synchronous hours. During each synchronous hour h, we will first show that at least a constant fraction $0.77|\mathcal{M}|$ of agents have $\texttt{bias} = \mathsf{T}$ with $\texttt{hour} = h$. Then we will show that split reactions bring most biased agents down to $\texttt{exponent} = -h$. Finally we will show that cancel reactions will reduce the count of biased agents, leaving sufficiently many $\mathcal{O}$ agents for the next step of the induction.

Recall that the initial gap

$$g = (|\{a : a.\texttt{input} = \mathsf{A}\}| - |\{b : b.\texttt{input} = \mathsf{B}\}|) = \sum_{m.\texttt{role}=\mathsf{Main}} m.\texttt{bias}$$

is both the difference between the counts of A and B agents in the initial configuration and the invariant value of the total bias. We define $\beta_+ = \beta_+(t) = \sum_{a.\texttt{opinion}=+1} |a.\texttt{bias}|$ and $\beta_- = \beta_-(t) = \sum_{b.\texttt{opinion}=-1} |b.\texttt{bias}|$ to be the total unsigned bias of the positive A agents at time t and negative B agents at time t. Then the net bias $g = \beta_+ - \beta_-$ is the invariant.

We also define $\mu = \mu(t) = \beta_+ + \beta_-$ to be the total unsigned bias, which we interpret as "mass". Note that split reactions preserve μ, while cancel reactions strictly decrease μ. The initial configuration has mass $\mu(0) = n$, and if we eliminate all of the minority opinion then we will get $\beta_- = 0$ and $\mu = g$. Thus decreasing the mass shows progress toward reaching consensus. Also, if every biased agent decreased their exponent by 1, this would exactly cut the μ in half. We will show an upper bound on μ that cuts in half after each synchronous hour, which implies that on average all biased agents are moving down one exponent.

We also define $\mu_{(>-i)}(t) = \sum_{m.\texttt{exponent}>-i} |m.\texttt{bias}|$ as the total mass of all biased agents above exponent $-i$. Note that a bound $\mu_{(>-i)} \leq x2^{-i+1}$ gives a bound on total count $|\{a : a.\texttt{exponent} > -i\}| \leq x$, since even if all agents above exponent $-i+1$ split down to exponent $-i+1$, they would have count at most x. Also note that $\mu(t)$ and $\mu_{(>-i)}(t)$ are nonincreasing in t, since the mass above a given exponent can never increase.

This inductive argument will stop working once we reach a low enough exponent that the gap has been sufficiently amplified. We define $g_i = g \cdot 2^i$, which we call the relative gap to hour i / exponent $-i$, because if all agents had $\texttt{exponent} = -i$, then a gap $g_i = |\{a : a.\texttt{opinion} = +1\}| - |\{b : b.\texttt{opinion} = -1\}|$ would maintain the invariant $g = \sum_v v.\texttt{bias} = \sum_{a.\texttt{opinion}=+1} \frac{1}{2^i} - \sum_{b.\texttt{opinion}=-1} \frac{1}{2^i} = \frac{g_i}{2^i}$. Note that $g_0 = g$, and the relative gap doubles as we increment the hour and decrement the exponent. So if the Main agents have roughly synchronous exponents, there will be some minimal exponent where the relative gap has grown to exceed the number of Main agents $|\mathcal{M}|$, and there are not enough agents available to maintain the invariant using lower exponents.

We now formalize this idea of a minimal exponent. In the case where there is an initial majority, $g_i \neq 0$, and because we assume the majority is A, we have $g_i > 0$. Define the minimal exponent $-l = \lfloor \log_2(\frac{g}{0.4|\mathcal{M}|}) \rfloor$ to be the unique exponent corresponding to relative gap $0.4|\mathcal{M}| \leq g_l < 0.8|\mathcal{M}|$. For larger exponents $-i \geq -l + 5 = -(l-5)$, we have $g_i \leq g_{l-5} < 0.025|\mathcal{M}|$. Thus for hours $0, 1, \ldots, l-5$ and exponents $0, -1, \ldots, -l+5$, the gap is still very small. The small gap makes the inductive argument stronger, and we will use this small gap to show a high rate of cancelling keeps shrinking the mass μ in half each hour and keeps the count of $\mathcal{O}$ agents large, above a constant fraction $0.77|\mathcal{M}|$.

For the last few hours $l-4, \ldots, l$ and exponents $-l+4, \ldots, -l$, the doubling gap weakens the argument. Thus we have separate bounds for each of these hours. These weaker bounds acknowledge the fact that the majority count is starting to increase while the count of $\mathcal{O}$ agents is starting to decrease.

Theorem 3.5.12. *For all synchronous hours $h = 0, \ldots, l$, during times $[\text{start}_h + \frac{2}{c}, \text{end}_h]$, the count of $\mathcal{O}$ agents at $\texttt{hour} = h$, $|\mathcal{O}_h| \geq \tau_h|\mathcal{M}|$. Then by time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$, the mass $\mu_{(>-h)}$ above exponent $-h$ satisfies $\mu_{(>-h)} \leq 0.001 \cdot 2^{-h+1}$. Then by time $\text{start}_h + \frac{2}{c} + \frac{47}{m} \leq \text{end}_h$, the total mass $\mu(\text{end}_h) \leq \rho_h|\mathcal{M}|2^{-h}$.*

The constant $\rho_h = 0.1$ for $h \leq l-5$. Then we have $\rho_{l-4} = 0.104$, $\rho_{l-3} = 0.13$, $\rho_{l-2} = 0.212$, $\rho_{l-1} = 0.408$, and $\rho_l = 0.808$.

The constant $\tau_h \geq 0.97 - 2\rho_{(h-1)}$, so $\tau_h \geq 0.77$ for $h \leq l-4$, and the minimum value $\tau_l \geq 0.15$.

Note that in the case of a tie, l is undefined since we always have gap $g_i = 0$. Here the stronger inductive argument will hold for all hours and exponents. In the tie case, we will technically define $l = L + 5$ so the stronger $h \leq l - 5$ bounds apply to all hours.

We will prove the three sequential claims via three separate lemmas. The first argument of Lemma 3.5.13, where the clock brings a large fraction of $\mathcal{O}$ agents up to hour h, will need parallel time $\frac{2}{c}$. The second argument of Lemma 3.5.15, where the $\mathcal{O}$ agents reduce the mass above exponent $-h$ by split reactions, will need parallel time $\frac{41}{m}$. The third argument of Lemma 3.5.16, where cancel reactions at exponent $-h$ reduce the total mass, will need parallel time $\frac{6}{m}$. Thus the total time we need in a synchronous hour is

$$\frac{2}{c} + \frac{47}{m} \leq \frac{49}{c} \leq \frac{17}{c^2} \leq \frac{0.45 \cdot 45 - 3.1}{c^2},$$

where we use that $c < \frac{1}{3} < m$ by Lemma 3.4.2. Thus by Theorem 3.5.9, since we have constant $k = 45$ minutes per hour, each synchronous hour is long enough with very high probability.

The argument proceeds by induction, with each lemma using the previous lemmas and the inductive hypotheses at previous hours. The base case comes from Lemma 3.4.3, where we have that the initial gap $|g| < 0.025m$, so $l - 5 \geq 0$, and the starting mass is at most $0.03|\mathcal{M}|$. This mass bound gives the base case for Lemma 3.5.16, whereas since $h = 0$ is the minimum possible hour, the base cases for Lemma 3.5.13 and Lemma 3.5.15 are trivial.

Lemma 3.5.13. *For each hour $h = 0, \ldots, l$, during the whole synchronous hour $[\text{start}_h, \text{end}_h]$, the count of $\mathcal{O}$ agents $|\mathcal{O}| \geq (0.9976 - 2\rho_{(h-1)})|\mathcal{M}|$. Then during times $[\text{start}_h + \frac{2}{c}, \text{end}_h]$, the count of $\mathcal{O}$ agents at $\texttt{hour} = h$, $|\mathcal{O}_h| \geq (0.97 - 2\rho_{(h-1)})|\mathcal{M}|$ with very high probability.*

Proof. We show the first claim, that during synchronous hour h, the count $|\mathcal{O}| \geq (0.9976 - 2\rho_{(h-1)})|\mathcal{M}|$ by process of elimination. By Lemma 3.5.16, by time $\text{end}_{h-1} < \text{start}_h$, the total mass $\mu \leq \rho_{(h-1)}|\mathcal{M}|2^{-h+1}$, which implies the total count $\{a : a.\texttt{exponent} \geq -h\} \leq 2\rho_{(h-1)}|\mathcal{M}|$ in all future configurations, since total mass is nonincreasing. Then by Lemma 3.5.11, the count $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > h\} \cup \{b : b.\texttt{exponent} < -h\}| \leq 0.0024|\mathcal{M}|$ until time end_h. This leaves $|\mathcal{M}| - 0.0024|\mathcal{M}| - 2\rho_{(h-1)}|\mathcal{M}| = (0.9976 - 2\rho_{(h-1)})|\mathcal{M}|$ agents who must be $\mathcal{O}$ agents with $\texttt{hour} \geq h$ until time end_h.

By definition of time start_h, there are at least $0.9|\mathcal{C}|$ Clock agents with $\texttt{hour} \geq h$ that will bring these $\mathcal{O}$ agents up to $\texttt{hour} = h$. We need to bring all but $0.0276|\mathcal{M}|$ of these agents up to $\texttt{hour} = h$ to achieve the desired bound $|\mathcal{O}_h| \geq (0.97 - 2\rho_{(h-1)})|\mathcal{M}|$. We can apply Lemma 3.3.7, with $a = 0.9c$, $b_1 = (0.9976 - 2\rho_{(h-1)})m < (0.9976 - 2 \cdot 0.1)m$, and $b_2 = 0.0276m$ to conclude this happens after parallel time t, where $t < (1+\epsilon)\mathbb{E}[t]$ with very high probability. The expected time

$$\mathbb{E}[t] = \frac{\ln(b_1) - \ln(b_2)}{2a} \leq \frac{\ln(0.7976m) - \ln(0.0276m)}{2 \cdot 0.9c} \leq \frac{1.89}{c}.$$

Thus for small constant $\epsilon > 0$, we have $t < (1+\epsilon)\frac{1.89}{c} < \frac{2}{c}$ with very high probability.

□

In order to reason about the total mass $\mu_{(>-h)}$ above exponent $-h$, we will define the potential function $\phi_{(>-h)}$, where for a biased agent a with $a.\texttt{exponent} = -i \geq -h+1$ at time t, $\phi_{(>-h)}(a,t) = 4^{-i+h-1}$. The global potential

$$\phi_{(>-h)}(t) = \sum_{a.\texttt{exponent}>-h} \phi(a,t) \geq \sum_{a.\texttt{exponent}>-h} 4^{-h+1+h-1} = |\{a : a.\texttt{exponent} > -h\}|.$$

Since $\phi_{(>-h)}$ upper bounds the count above exponent h, we can bound the mass $\mu_{(>-h)}(t) \leq 2^{-h+1}\phi_{(>-h)}(t)$. Also note that unlike the mass, $\phi_{(>-h)}(t)$ strictly decreases via split reactions since $4^{-i} > 4^{-i-1} + 4^{-i-1}$. This will let us show that $\phi_{(>-h)}$ exponentially decays to 0 when there are a constant fraction of $\mathcal{O}$ agents to do these splits.

We first show that by hour h exponents significantly far above $-h$ are empty. Letting $q = \lfloor \frac{\ln n}{3} \rfloor$, we will show the potential $\phi_{(>-h+q)}$ hits 0 by hour h.

Lemma 3.5.14. *For each hour $h = 0, \ldots, l$, by time $\text{start}_h + \frac{2}{c}$, the maximum level among all biased agents is at most $-h+q$ with high probability $1 - O(1/n^{12})$.*

Proof. The statement is vacuous for hours $h < q$, so we must only consider hours $h \geq q$. We use the potential $\phi_{(>-h+q)}$, and start the argument at time $t_{\text{start}} = \text{start}_{(h-q)} + \frac{2}{c} + \frac{41}{m}$ where inductively by Lemma 3.5.15 $\phi_{(>-h+q)}(t_{\text{start}}) \leq 0.001|\mathcal{M}|$. We must show that by time $t_{\text{end}} = \text{start}_h + \frac{2}{c}$, we have $\phi_{(>-h+q)}(t_{\text{end}}) = 0$.

By Lemma 3.5.13, the count of $\mathcal{O}$ agents with $\texttt{hour} \geq h-q$, $|\mathcal{O}_{(\geq h-q)}| \geq 0.77|\mathcal{M}|$ during all synchronous hours after $\text{start}_{(h-q)} + \frac{2}{c}$ and up through synchronous hour $l - 5 \geq h - 5$. Thus the interval we consider consists of at least $q-5$ synchronous hours. By Theorem 3.5.9, each synchronous hour takes at least time $[1.48(45-2)-2.2]/c^2 \geq 17/c^2 \geq 51/m$, since $c < \frac{1}{3} < m$ by Lemma 3.4.2. Thus the total time for this argument is at least $t_{\text{end}} - t_{\text{start}} \geq \frac{51(q-5)}{m}$ time.

For each of the $\frac{51(q-5)}{m}n$ interactions in this interval, we consider the expected change to $\phi_{(>-h+q)}$. For each biased agent a with $a.\texttt{exponent} = -i > -h+q$, a split reaction will change the potential by $\Delta\phi_a \leq 4^{h-1}(2\cdot 4^{-i-1} - 4^{-i}) = 4^{h-1}(-\frac{1}{2}4^{-i}) = -\frac{1}{2}\phi_a$. Then in each interaction at parallel time t, the expected change in the potential

$$\begin{aligned}\mathbb{E}[\phi_{(>-h+q)}(t+1/n) - \phi_{(>-h+q)}(t)] &\leq \sum_{a.\texttt{exponent}>(-h+q)} \mathbb{P}[a \text{ splits}]\cdot\Delta\phi_a \\ &\leq \sum_{a.\texttt{exponent}>(-h+q)} \frac{2\cdot 0.77m}{n}\cdot -\frac{1}{2}\phi_a = -\frac{0.77m}{n}\phi_{(>-h+q)}(t).\end{aligned}$$

Then we have $\mathbb{E}[\phi_{(>-h+q)}(t+1/n)|\phi_{(>-h+q)}(t)] \leq (1-\frac{0.77m}{n})\phi_{(>-h+q)}(t)$.

We now recursively apply this bound to all $\frac{51(q-5)}{m}n$ interactions beginning at time t_{start}:

$$\begin{aligned}\mathbb{E}\big[\phi_{(>-h+q)}(t_{\text{end}})|\phi_{(>-h+q)}(t_{\text{start}})\big] &\leq \left(1-\frac{0.77m}{n}\right)^{\frac{51(q-5)}{m}n}\phi_{(>-h+q)}(t_{\text{start}}) \\ \mathbb{E}\big[\phi_{(>-h+q)}(t_{\text{end}})\big] &\leq \exp(-0.77\cdot 51\ln n/3)\cdot\exp(51\cdot 5\cdot 0.77)\cdot 0.001|\mathcal{M}| \\ &\leq n^{-13}\exp(197)\cdot 0.001mn \\ &= O\big(n^{1-13}\big) = O\big(n^{-12}\big).\end{aligned}$$

Finally, since $\phi_{(>-h+q)}$ takes nonnegative integer values, we can apply Markov's Inequality to conclude $\mathbb{P}[\phi_{(>-h+q)}(t_{\text{end}}) > 0] = O(1/n^{12})$. □

Now we can reason about the potential $\phi_{(>-h)}$ during hour h, which will decrease by a large constant factor. The upper bound on $\phi_{(>-h)}$ gives the a bound on the mass of the upper tail $\mu_{(>-h)}$.

Lemma 3.5.15. *For each hour $h = 0,\ldots,l$, by time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$, the potential $\phi_{(>-h)} \leq 0.001|\mathcal{M}|$ with very high probability. This implies the mass above exponent $-h$ is $\mu_{(>-h)} \leq 0.001|\mathcal{M}|2^{-h+1}$.*

Proof. By Lemma 3.5.13, after time $\text{start}_h + \frac{2}{c}$, we have a count $|\mathcal{O}_h| \geq \tau_h|\mathcal{M}|$, where the weakest bound is at hour l, where $\tau_l = 0.15$.

Inductively, we have $\phi_{(>-h+1)}(\text{start}_h) \leq 0.001|\mathcal{M}|$ by time $\text{start}_{(h-1)} + \frac{2}{c} + \frac{41}{m}$. By Lemma 3.5.16, by time $\text{end}_{(h-1)}$, the total mass $\mu \leq \rho_{(h-1)}|\mathcal{M}|2^{-h+1} \leq \rho_{(l-1)}|\mathcal{M}|2^{-h+1}$. Thus there are at most

$\rho_{(l-1)}|\mathcal{M}| = 0.408|\mathcal{M}|$ biased agents with exponent $-h+1$, which lets us bound the potential $\phi_{(>-h)}(\text{start}_h + \frac{2}{c}) \leq (0.408 + 4 \cdot 0.001)|\mathcal{M}| = 0.412|\mathcal{M}|$. Thus we must drop the potential by the constant factor 412.

To show that ϕ drops by a constant factor, we will use $\Phi(t) = \ln\big(\phi_{(>-h)}(t)\big)$, which will be a supermartingale. If agent a at with exponent $-i$ splits, with $\phi_a = 4^{-i+h-1}$, this changes the potential ϕ by $\Delta\phi_a = -\frac{1}{2}\phi_a$. The potential Φ then changes by

$$\Delta\Phi_a = \ln\big(\phi_{(>-h)} + \Delta\phi_a\big) - \ln\big(\phi_{(>-h)}\big) = \ln\left(1 + \frac{-\frac{1}{2}\phi_a}{\phi_{(>-h)}}\right) \leq -\frac{\phi_a}{2\phi_{(>-h)}}.$$

The expected change in Φ is then

$$\mathbb{E}[\Delta\Phi] \leq \sum_{a.\texttt{exponent}>-h} \mathbb{P}[a \text{ splits}] \cdot \Delta\Phi_a \leq \sum_{a.\texttt{exponent}>-h} \frac{2\tau_h m}{n} \cdot -\frac{\phi_a}{2\phi_{(>-h)}} = -\frac{0.15m}{n}.$$

We define the supermartingale $(\Phi_j)_{j=0}^{\frac{41}{m}n}$, where $\Phi_j = \Phi(\text{start}_h + \frac{2}{c} + \frac{j}{n})$. Then the desired inequality is $\phi_{>(h+1)}(\text{start}_h + \frac{2}{c} + \frac{41}{m}) \leq 0.001|\mathcal{M}| \leq \phi_{>(h+1)}(\text{start}_h + \frac{2}{c})/412$, so we need to show $\Phi_{\frac{41}{m}n} - \Phi_0 \leq \ln\big(\frac{1}{412}\big)$. The expected value

$$\mathbb{E}[\Phi_{\frac{41}{m}n} - \Phi_0] \leq -\frac{0.15m}{n} \cdot \frac{41}{m}n = -6.15 \leq \ln\left(\frac{1}{412}\right) - 0.12.$$

To apply Azuma's Inequality, we will need a bound on the difference $|\Phi_{j+1} - \Phi_j| \leq \max\left|\frac{\phi_a}{2\phi_{(>-h)}}\right|$. During the interval we consider, $\phi_{(>-h)} \geq 0.001|\mathcal{M}|$, since after that the desired inequality will hold. By Lemma 3.5.14, by time $\text{start}_h + \frac{2}{c}$, the maximum exponent in the population is at most $-h+q$, where $q = \lfloor\frac{\ln n}{3}\rfloor$, so $\phi_a \leq 4^{(-h+q)+h-1} = 4^{q-1}$. Using the fact that $4^q = e^{\ln 4 \ln n/3} = O(n^{0.47})$, we can bound the largest change in Φ as

$$|\Phi_{j+1} - \Phi_j| \leq \frac{4^{q-1}}{0.002|\mathcal{M}|} = O\left(\frac{4^q}{n}\right) = O(n^{0.47}/n) = O(1/n^{0.53}).$$

Now by Azuma's Inequality (Theorem 3.3.2) we have

$$\mathbb{P}\big[(\Phi_{\frac{41}{m}n} - \Phi_0) - \mathbb{E}[\Phi_{\frac{41}{m}n} - \Phi_0] \geq 0.12\big] \leq \exp\left(-\frac{0.12^2}{2\sum_{j=0}^{\frac{41}{m}n}(O(n^{-0.53}))^2}\right) = \exp\big(-\Theta(n^{0.06})\big).$$

Thus $\phi_{(>-h)}$ will drop by at least the constant factor 412 with very high probability, giving $\phi_{(>-h)}(\text{start}_h + \frac{2}{c} + \frac{41}{m}) \leq 0.001|\mathcal{M}|$. □

Lemma 3.5.16. *For each hour $h = 0, \ldots, l$, by time* $\mathrm{start}_h + \frac{2}{c} + \frac{47}{m} \leq \mathrm{end}_h$*, the total mass* $\mu \leq \rho_h|\mathcal{M}|2^{-h}$ *with very high probability. The constant $\rho_h = 0.1$ for all $h \leq l+5$, then the last few constants $\rho_{l-4} = 0.104$, $\rho_{l-3} = 0.13$, $\rho_{l-2} = 0.212$, $\rho_{l-1} = 0.408$, and $\rho_l = 0.808$.*

Proof. We start the argument at time $\mathrm{start}_h + \frac{2}{c} + \frac{41}{m}$. Let $A = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -h\}$ and $B = \{b : b.\texttt{opinion} = -1, b.\texttt{exponent} = -h\}$. We will show that by time end_h, large number of cancel reactions happen between the agents in A and B to reduce the total mass. Inductively, by end_{h-1} we have total mass $\mu \leq \rho_{(h-1)}|\mathcal{M}|2^{-h+1}$. We assume in the worst case (since we need the total mass to be small) that we start with the maximum possible amount of mass $\mu = \rho_{(h-1)}|\mathcal{M}|2^{-h+1}$. We use the upper bound on the gap $g_h \leq \alpha|\mathcal{M}|$, where $\alpha = 0.8 \cdot 2^{h-l}$ and assume in the worst case (since larger gaps reduce the rate of cancelling reactions) the largest possible gap $\beta_+ - \beta_- = g_h 2^{-h} = \alpha|\mathcal{M}|2^{-h}$. This gives majority mass $\beta_+ = (\rho_{(h-1)} + \frac{\alpha}{2})|\mathcal{M}|2^{-h}$ and minority mass $\beta_- = (\rho_{(h-1)} - \frac{\alpha}{2})|\mathcal{M}|2^{-h}$.

Since the minority is the limiting reactant in the cancelling reactions, we assume in the worst case the smallest possible minority count at exponent $-h$, where all mass outside of exponent $-h$ is the minority. Then the majority count at exponent $-h$ is $|A| = (\rho_{(h-1)} + \frac{\alpha}{2})|\mathcal{M}|$. By Lemma 3.5.15, the mass above exponent $-h$ is $\mu_{(>-h)}(\mathrm{start}_h + t_2) \leq 0.001|\mathcal{M}|2^{-h+1}$. By Lemma 3.5.11, the maximum count below exponent $-h$ is $0.0024|\mathcal{M}|$, so the mass $\mu_{(<-h)} \leq 0.0024|\mathcal{M}|2^{-h-1}$. This leaves a minority count at exponent $-h$ of $|B| = (\rho_{(h-1)} - \frac{\alpha}{2} - 0.002 - 0.0012)|\mathcal{M}|$.

Now we will apply Lemma 3.3.6 with $a = (\rho_{(h-1)} + \frac{\alpha}{2})m$ and $b = (\rho_{(h-1)} - \frac{\alpha}{2} - 0.0032)m$. First we consider the cases where $h \leq l+5$, where $\rho_h = \rho_{(h-1)} = 0.1$ There the bound on the gap $\alpha = 0.8 \cdot 2^{(l-5)-l} = 0.025$. This gives $a = 0.1125m$ and $b = 0.0843m$. By Lemma 3.3.6, the time t to cancel a fraction $d = 0.05m$ from both a and b has mean

$$\begin{aligned}\mathbb{E}[t] &\sim \frac{\ln(b) - \ln(a) - \ln(b-d) + \ln(a-d)}{2(a-b)} \\ &= \frac{\ln(0.0843) - \ln(0.1125) - \ln(0.0343) + \ln(0.0625)}{2(0.0282m)} < \frac{5.53}{m},\end{aligned}$$

so $\mathbb{E}[t](1+\epsilon) < \frac{6}{m}$ and $t < \frac{6}{m}$ with very high probability. Cancelling this fraction d reduces the total mass by $2dn2^{-h} = 0.1|\mathcal{M}| \cdot 2^{-h} = \rho_{(h-1)}|\mathcal{M}|2^{-h+1} - \rho_h|\mathcal{M}|2^{-h}$. Then the total mass is at most $\rho_h|\mathcal{M}|2^{-h}$ by time $\mathrm{start}_h + \frac{2}{c} + \frac{47}{m}$.

For the remaining levels $h = l-4, l-3, l-2, l-1, l$, we will repeat the above argument and calculation, but now the bound will change so $\rho_h \neq \rho_{(h-1)}$. First our bound on the gap α will double as we move down each level. This causes the worst case fractions a and b to be spread further apart. Then d, the largest fraction which will cancel from both sides within $\frac{6}{m}$ time with high probability, will be smaller. This gives the new mass bound ρ_h, since from the equation $\rho_{(h-1)}|\mathcal{M}|2^{-h+1} - \rho_h|\mathcal{M}|2^{-h} = 2dn2^{-h}$, we have $d = (\rho_{(h-1)} - \frac{\rho_h}{2})m$ and $\rho_h = 2(\rho_{(h-1)} - \frac{d}{m})$. This relative total mass bound ρ_h will be growing larger as h decreases. This is to be expected, since we do in fact see the count of biased agents growing as they reach the final exponents before the count of $\mathcal{O}$ agents runs out (see Fig. 3.4c and Fig. 3.4d, where the value $-l = -19$).

The following table shows the constants used in the computations at each of the steps $h = l-5, l-4, \ldots, l$. The top row $h = l-5$ corresponds to the exact calculations above, then the following rows are the constants used in the same argument for lower levels.

$h = l-5$	$\alpha = 0.025$	$a = 0.1125m$	$b = 0.0843m$	$d = 0.05m$	$\mathbb{E}[t] < 5.53/m$	$\rho_{(l-5)} = 0.1$
$h = l-4$	$\alpha = 0.05$	$a = 0.125m$	$b = 0.0718m$	$d = 0.048m$	$\mathbb{E}[t] < 5.83/m$	$\rho_{(l-4)} = 0.104$
$h = l-3$	$\alpha = 0.1$	$a = 0.154m$	$b = 0.0508m$	$d = 0.039m$	$\mathbb{E}[t] < 5.66/m$	$\rho_{(l-3)} = 0.13$
$h = l-2$	$\alpha = 0.2$	$a = 0.23m$	$b = 0.0268m$	$d = 0.024m$	$\mathbb{E}[t] < 5.29/m$	$\rho_{(l-2)} = 0.212$
$h = l-1$	$\alpha = 0.4$	$a = 0.412m$	$b = 0.0088m$	$d = 0.008m$	$\mathbb{E}[t] < 2.95/m$	$\rho_{(l-1)} = 0.408$
$h = l$	$\alpha = 0.8$	$a = 0.808m$	$b = 0.0048m$	$d = 0.004m$	$\mathbb{E}[t] < 1.11/m$	$\rho_l = 0.808$

Note that for the last couple hours, the worst case for b is very small, so the amount of cancelling that happens is negligible, and the relative mass bound essentially doubles. We will see later that the minority count does in fact sharply decrease, so it is accurate that the rate of cancellation drops to zero, and the count of biased majority agents is roughly doubling for these last couple rounds. □

3.5.3. End of Phase 3. Now that we have proven Theorem 3.5.12, we will finish analyzing separately the cases of an initial tie and an initial majority opinion.

In the case of a tie, the stronger bounds from Theorem 3.5.12 hold all through the final hour $h = L$. Thus at hour L we still have a constant fraction $0.77|\mathcal{M}|$ of $\mathcal{O}$ agents. We now show that in the remaining $O(\log n)$ time in Phase 3, while the `Clock` agents with `hour` $= h$ decrement their counter in Line 6, these $\mathcal{O}$ agents can keep doing split reactions to bring all remaining biased agents down to exponent $-L$. This lets us now prove Theorem 3.5.1, the main result of the section for the case of a tie.

Theorem 3.5.1. *If the initial configuration was a tie with gap 0, then by the end of Phase 3, all biased agents have* `exponent` $= -L$*, with high probability* $1 - O(1/n^2)$.

Proof. We start by using Theorem 3.5.12, where at hour L, the total mass $\mu(\mathrm{end}_L) \leq 0.1|\mathcal{M}|2^{-L}$. Thus the count of biased agents is at most $0.1|\mathcal{M}|$. We also have at least $0.77|\mathcal{M}|$ $\mathcal{O}$ agents with `hour` $= L$ (this count is actually slightly higher, as we could show almost all of the $0.9|\mathcal{M}|$ $\mathcal{O}$ agents reach `hour` $= L$ quickly by the same argument as Lemma 3.5.13, but this bound will suffice).

We use the potential $\phi_{(>-L)}$, where by Lemma 3.5.15, we have $\phi_{(>-L)}(\mathrm{end}_0) \leq 0.001|\mathcal{M}|$. The goal is to now show that $\phi_{(>-L)} = 0$ within the $O(\log n)$ time it takes for any `counter` to hit 0 and trigger the move to the next phase. This proof proceeds identically to the proof of Lemma 3.5.14, where we had shown that $\mathbb{E}[\phi_{(>-L)}(t+1/n)|\phi_{(>-L)}(t)] \leq (1-\frac{0.77m}{n})\phi_{(>-L)}(t)$. We again recursively bound the expected value of $\phi_{(>-L)}$ after an additional $16\ln n$ time.

$$\mathbb{E}[\phi_{(>-L)}(\mathrm{end}_L + 16\ln n)|\phi_{(>-L)}(\mathrm{end}_0)] \leq \left(1 - \frac{0.77m}{n}\right)^{16n\ln n} \phi_{(>-L)}(\mathrm{end}_0)$$

$$\mathbb{E}[\phi_{(>-L)}(\mathrm{end}_L + 16\ln n)] \leq \exp(-0.77 \cdot 0.25 \cdot 16\ln n) \cdot 0.001|\mathcal{M}|$$

$$\leq n^{-3.08} 0.001mn = O(1/n^2).$$

Again we conclude by Markov's Inequality that $\mathbb{P}[\phi_{(>-L)}(\mathrm{end}_L + 16\ln n) > 0] \leq O(1/n^2)$.

Finally we must argue that Phase 3 lasts at least until time $\mathrm{end}_L + 16\ln n$. We will bound the probability that a Clock agent can decrement its `counter` down to 0 before time $\mathrm{end}_L + 16\ln n$. Note the if statement on Line 6 will only decrement the counter when both Clock agents have reached the final minute. Even if in the worst case an agent was at the final minute at the beginning of the phase, until time end_{L-1} the count of other Clock agents with `minute` $= kL$ is at most $0.1p|\mathcal{C}| = 0.001|\mathcal{C}|$. By Theorem 3.5.9, the time between consecutive hours is at most $\frac{3.86k}{c^2} < \frac{290}{c}$ using $k = 45$ and the bound $c > 0.2$ with very high probability from Lemma 3.4.2. Then the number of interactions until end_{L-1} is at most $\frac{290}{c}Ln$. In each interaction, the probability of the Clock agent decrementing its counter is at most $2 \cdot 0.001c \cdot \frac{1}{n}$, so the expected number of decrements $\mathbb{E}[X] \leq 290L \cdot 0.002 \leq 0.58(\log_2(n)) \leq 0.84\ln n$. Then by Chernoff bounds in Theorem 3.3.1, with

$\mu = 0.84 \ln n, \delta = 5$, we have

$$\mathbb{P}[X \geq 5.04 \ln n] \leq \mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2+\delta}\right) = \exp(-3 \ln n) = n^{-3}.$$

Then after time end_{L-1}, we assume that all Clock agents have `minute` $= kL$. In this case, the probability a given Clock agent decrements its counter is at most $\frac{2c}{n}$. Thus in $16n \ln n$ interactions, the expected number of decrements $\mathbb{E}[X] \leq 32c \ln n \leq 11 \ln n$, using $c < \frac{1}{3}$ from Lemma 3.4.2. We again use Theorem 3.3.1, with $\mu = 11 \ln n, \delta = 0.9$, to conclude

$$\mathbb{P}[X \geq 20.9 \ln n] \leq \mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2+\delta}\right) \leq \exp(-3.07 \ln n) \leq n^{-3}.$$

Then by union bound, Clock agents decrements their counter at most $5.04 \ln n + 20.9 \ln n < 26 \ln n$ times with high probability $1 - O(1/n^2)$. In Phase 3, we initialize `counter` $\leftarrow c_3 \ln n$, so setting this counter constant $c_3 = 26$, we conclude that no agent moves to the next phase before time $\text{end}_L + 16 \ln n$ with high probability. This gives enough time to conclude that all biased agents have `exponent` $= -L$ by the end of Phase 3. □

Now we consider the case where there is an initial majority, which we have assumed without loss of generality was A. Here the argument of Theorem 3.5.12 stops working at exponent $-l$, because of the substantial relative gap $0.4|\mathcal{M}| \leq g_l < 0.8|\mathcal{M}|$. Next we will need to show that the count of $\mathcal{O}$ gets brought to almost 0 by split reactions with the majority. In order to show that the count of $\mathcal{O}$ stays small however, we will need some way to bound future cancel reactions. To do this, we will now show that during the last few hours $l-4, \ldots, l$, the minority mass β_- drops to a negligible amount. We will then use this tight bound to show the majority consumes most remaining $\mathcal{O}$ agents, and finally that constraints on the majority mass β_+ will force the majority count to remain above 99% at exponents $-l, -(l+1), -(l+2)$ for all reachable configurations in the rest of Phase 3.

We first show inductively that the minority mass becomes negligible during the last few hours $l-5, \ldots, l$. We already showed a bound on the total mass in Lemma 3.5.16, which used an upper bound on the gap. If we want the minority mass to be small, now the worst case is the smallest possible gap. Notice that just using the final mass bound $\mu < \rho_l |\mathcal{M}| 2^{-l} = 0.808 |\mathcal{M}| 2^{-l}$ in the worst case with the smallest possible gap $g_l = 0.4$ at exponent $-l$, would imply $\beta^+ \approx 0.6 |\mathcal{M}| 2^{-l}$ and $\beta^- \approx 0.2 |\mathcal{M}| 2^{-l}$. To get a much tighter upper bound on minority mass, we will make a similar inductive argument to Lemma 3.5.16, now using the lower bound for each gap g_h, to show enough minority mass cancels at exponent $-h$ during each hour h.

Lemma 3.5.17. *For each hour $h = l-5, \dots, l$, by time end_h, the minority mass $\beta_- \le \xi_h |\mathcal{M}| 2^{-h}$ with very high probability. The constants $\xi_{(l-5)} = 0.04375$, $\xi_{(l-4)} = 0.0375$, $\xi_{(l-3)} = 0.0267$, $\xi_{(l-2)} = 0.0145$, $\xi_{(l-1)} = 0.0056$, and $\xi_l = 0.004$.*

Proof. For the base case $h = l-5$, by Lemma 3.5.16, the total mass $\mu(\mathrm{end}_{(l-5)}) \le 0.1|\mathcal{M}|2^{-l+5}$. The relative gap $0.0125|\mathcal{M}| \le g_{(l-5)} < 0.025|\mathcal{M}|$, so now in the worst case (for minority mass being small), we assume the smallest gap $g_{(l-5)} = 0.0125|\mathcal{M}|$ and largest total mass $\mu(\mathrm{end}_{(l-5)}) = \beta_+ + \beta_- = 0.1|\mathcal{M}|2^{-l+5}$. Since the invariant gap $g = \beta_+ - \beta_- = g_{(l-5)}2^{-l+5}$, we have $\beta_+ = 0.05625|\mathcal{M}|2^{-l+5}$ and $\beta_- = 0.04375|\mathcal{M}|2^{-l+5}$, giving an upper bound constant $\xi_{(l-5)} = 0.04375$.

Now we outline the inductive step, for each level $h = l-4, \dots, l$. We will show the constants for the first step $h = l-4$ here, and then list constants for the remaining steps in a table below. Let $A = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -h\}$ and $B = \{b : b.\texttt{opinion} = -1, b.\texttt{exponent} = -h\}$.

We start the argument at time $\mathrm{start}_h + \frac{2}{c} + \frac{41}{m}$, and with the previous bound $\beta_-(\mathrm{end}_{(h-1)}) = \xi_{(h-1)}|\mathcal{M}|2^{-h+1} = 0.0875|\mathcal{M}|2^{-h}$. We will then assume in the worst case the smallest possible gap $g_h = \alpha|\mathcal{M}| = 0.4 \cdot 2^{h-l}|\mathcal{M}| = 0.025|\mathcal{M}|$. This gives $\beta_+ = (2\xi_{(h-1)} + \alpha)|\mathcal{M}|2^{-h}$. We then assume in the worst case that all mass allowed outside exponent $-h$ belongs to the minority (so doesn't reduce by cancelling), giving majority count $|A| = (2\xi_{(h-1)} + \alpha)|\mathcal{M}|$. We also assume the maximum amount of mass outside exponent $-h$. By Lemma 3.5.11, before time end_h, the count below exponent $-h$ is at most $0.0024|\mathcal{M}|$, so the mass is at most $0.0024|\mathcal{M}|2^{-h-1}$. By Lemma 3.5.15, after time $\mathrm{start}_h + \frac{2}{c} + \frac{41}{m}$, the mass above exponent $-h$ is $\mu_{(>-h)} \le 0.001 \cdot 2^{-h+1}$. This leftover remaining mass at exponent $-h$ gives minority count $|B| = (2\xi_{(h-1)} - 0.0012 - 0.002)|\mathcal{M}|$.

Now we will apply Lemma 3.3.6 with fractions $a = (2\xi_{(h-1)} + \alpha)m = 0.1125m$ and $b = (2\xi_{(h-1)} - 0.0036)m = 0.0843m$. These match the first fractions a, b used in the proof of Lemma 3.5.16, and again we use that the time t to cancel a fraction $d = 0.05m$ from both a and b has mean

$$\begin{aligned}\mathbb{E}[t] &\sim \frac{\ln(b) - \ln(a) - \ln(b-d) + \ln(a-d)}{2(a-b)} \\ &= \frac{\ln(0.0843) - \ln(0.1125) - \ln(0.0343) + \ln(0.0625)}{2(0.0282m)} < \frac{5.53}{m},\end{aligned}$$

so $\mathbb{E}[t](1+\epsilon) < \frac{6}{m}$ and $t < \frac{6}{m}$ with very high probability. Cancelling this fraction d reduces the minority mass by $dn2^{-h} = 0.05|\mathcal{M}| \cdot 2^{-h}$, giving a new minority mass $\beta_- = (2\xi_{(h-1)} - \frac{d}{m})|\mathcal{M}|2^{-h} = 0.0375|\mathcal{M}|2^{-h}$. Thus for $\xi_h = 0.0375$, the minority mass is at most $\xi_h|\mathcal{M}|2^{-h}$ by time $\text{start}_h + \frac{2}{c} + \frac{47}{m}$.

For the remaining levels $h = l-4, l-3, l-2, l-1, l$, we will repeat the above argument and calculation. The following table shows the constants used in the computations at each of the steps $h = l-4, l-3, \ldots, l$. The top row $h = l-4$ corresponds to the exact calculations above, then the following rows are the constants used in the same argument for lower levels.

$h = l-4$	$\alpha = 0.025$	$a = 0.1125m$	$b = 0.0843m$	$d = 0.05m$	$\mathbb{E}[t] < 5.53/m$	$\xi_{(l-4)} = 0.0375$
$h = l-3$	$\alpha = 0.05$	$a = 0.125m$	$b = 0.0718m$	$d = 0.048m$	$\mathbb{E}[t] < 5.83/m$	$\xi_{(l-3)} = 0.0267$
$h = l-2$	$\alpha = 0.1$	$a = 0.154m$	$b = 0.0508m$	$d = 0.039m$	$\mathbb{E}[t] < 5.66/m$	$\xi_{(l-2)} = 0.0145$
$h = l-1$	$\alpha = 0.2$	$a = 0.23m$	$b = 0.0268m$	$d = 0.024m$	$\mathbb{E}[t] < 5.29/m$	$\xi_{(l-1)} = 0.0056$
$h = l$	$\alpha = 0.4$	$a = 0.412m$	$b = 0.0088m$	$d = 0.008m$	$\mathbb{E}[t] < 2.95/m$	$\xi_l = 0.004$

Note that the structure of the proof yields the same sequence of values a, b as the proof of Lemma 3.5.16, so we have repeated the exact same applications of Lemma 3.3.6. This time, however, we are using the cancelled fraction d to show the minority mass becomes very small. □

Now Lemma 3.5.17 gives that minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ by time end_l. The rest of the argument will not try to reason about probabilistic guarantees on what the distribution looks like. Instead, we will make claims purely about reachability, and show that we get to a configuration where invariants force the count of majority agents to remain high in any reachable configuration using the transitions of Phase 3.

Lemma 3.5.18. *By the time* $\text{end}_{(l+2)}$*, the count of majority agents with* `exponent` $\in \{-l, -(l+1), -(l+2)\}$ *is at least* $0.96|\mathcal{M}|$*, with very high probability. Then in all reachable configurations where all agents are still in Phase3, the count of majority agents with* `exponent` $\in \{-l, -(l+1), -(l+2)\}$ *is at least* $0.92|\mathcal{M}|$.

Proof. We argue by process of elimination. First we apply Lemma 3.5.11 to conclude that until time $\text{end}_{(l+2)}$, the count $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > l+2\} \cup \{b : b.\texttt{exponent} < -(l+2)\}| \leq 0.0024|\mathcal{M}|$ with very high probability.

Now Lemma 3.5.17 gives that minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ after time end_l. Then we can bound the maximum minority count at exponent $-(l+2)$ and above by $0.016|\mathcal{M}|$. In addition, these minority agents could eliminate more majority agents by cancel reactions. The count of agents they could cancel with is at most $0.016|\mathcal{M}|$.

This leaves at least $|\mathcal{M}| - 0.0024|\mathcal{M}| - 0.016|\mathcal{M} - |0.016|\mathcal{M}| = 0.9656|\mathcal{M}|$ agents that are either majority or $\mathcal{O}$ agents at time end_l. We will next show that in hour $l+2$, most of any remaining $\mathcal{O}$ agents set $\texttt{hour} = l+2$ and then are consumed in split reactions with majority agents with $\texttt{exponent} > -(l+2)$.

The majority mass $\beta_+ \geq g = g_l 2^{-l} \geq 0.4|\mathcal{M}|2^{-l}$. By Lemma 3.5.15, at most $0.001|\mathcal{M}|2^{-l+1} = 0.002|\mathcal{M}|2^{-l}$ of β_+ can come from exponents above $-l$. Thus we need mass $0.398|\mathcal{M}|2^{-l}$ from majority agents at exponent $-l$ and below. Note that if the count $|\{a : a.\texttt{opinion} = +1, a.\texttt{exponent} \in \{-l, -(l+1)\}\}| \leq 0.19|\mathcal{M}|$, then the maximum majority mass could achieve would be $0.81|\mathcal{M}|$ with $\texttt{exponent} = -l-2$ and $0.19|\mathcal{M}|$ with $\texttt{exponent} = -l$, which gives $(\frac{0.81}{4} + 0.19)|\mathcal{M}|2^{-l} < 0.398|\mathcal{M}|2^{-l}$. This implies we must have a count of at least $0.19|\mathcal{M}|$ of majority agents at exponents $-(l+1)$ and $-l$ in all reachable configurations.

Next we show that all but at most $0.0036|\mathcal{M}|$ of these $\mathcal{O}$ agents gets brought up to $\texttt{hour} = l+2$. We start at time start_{l+2}, when the count $|\mathcal{O}| \leq (0.9656 - 0.398)\mathcal{M} = 0.5676|\mathcal{M}|$, and the count of clock agents at $\texttt{hour} = l+2$ is at least $0.9|\mathcal{C}|$. Thus we can apply Lemma 3.3.7, with $a = 0.9c$, $b_1 = 0.5676m$, $b_2 = 0.0056m$ to conclude that at most $0.0056|\mathcal{M}|$ of $\mathcal{O}$ agents are left at the wrong hour after parallel time $t_1 < (1+\epsilon)\mathbb{E}[t_1]$, where the expected time

$$\mathbb{E}[t_1] = \frac{\ln(b_1) - \ln(b_2)}{2a} = \frac{\ln(0.5676m) - \ln(0.0036m)}{1.8c} \leq \frac{2.82}{c}.$$

Thus with very high probability $t_1 < \frac{3}{c}$.

Next the at least $0.19|\mathcal{M}|$ majority agents at exponents $-(l+1)$ and $-l$ will eliminate these $\mathcal{O}$ agents by split reactions. We show that at most $0.001|\mathcal{M}|$ of $\mathcal{O}_h$ agents are left. We again apply Lemma 3.3.7, with $a = 0.19m$, $b_1 = 0.5676m$, $b_2 = 0.01m$ to conclude this takes at most parallel time $t_2 < (1+\epsilon)\mathbb{E}[t_2]$, where

$$\mathbb{E}[t_2] = \frac{\ln(b_1) - \ln(b_2)}{2a} = \frac{\ln(0.5676m) - \ln(0.001m)}{0.38m} \leq \frac{16.69}{m}.$$

Thus with very high probability $t_2 < \frac{17}{m}$.

Account for these counts $0.0032|\mathcal{M}|$ and $0.001|\mathcal{M}|$ of leftover $\mathcal{O}$ agents, along with the maximal count $0.001|\mathcal{M}|$ of biased agents above exponent $-l$, we are left with $0.9656|\mathcal{M}| - 0.0032|\mathcal{M}| - 0.001|\mathcal{M}| - 0.001|\mathcal{M}| = 0.96|\mathcal{M}|$ agents that must be majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$.

Now we argue that no reachable configurations can bring this count below $0.92|\mathcal{M}|$. We have already accounted for the maximum number of minority agents that can do cancel reactions at exponents $-l, -(l+1), -(l+2)$. Thus we only have to argue that no amount of split reactions can bring this count down by $0.04|\mathcal{M}|$. Note that reducing the count by $0.04|\mathcal{M}|$ will reduce the mass at these exponents by at least $0.04|\mathcal{M}|2^{-(l+2)}$. It will take at least $0.08|\mathcal{M}|$ agents below exponent $-l-2$ to account for the same mass, for a total of $0.96|\mathcal{M}| - 0.04|\mathcal{M}| + 0.08|\mathcal{M}| = |\mathcal{M}|$. Thus it is not possible to move more than $0.04|\mathcal{M}|$ majority agents below exponent $-(l+2)$ by split reactions, because there are not enough Main agents to account for the mass. □

We can now use the results of Lemma 3.5.18, Lemma 3.5.15, and Lemma 3.5.17, including the high probability requirements from all previous lemmas, to prove Theorem 3.5.2, the main result for Phase 3 in the non-tie case:

THEOREM 3.5.2. *Assume the initial gap* $|g| < 0.025|\mathcal{M}|$. *Let the exponent* $-l = \lfloor \log_2(\frac{g}{0.4|\mathcal{M}|}) \rfloor$. *Let* $i = sign(g)$ *be the majority opinion and* M *be the set of all agents with* $\texttt{role} = \textsf{Main}$, $\texttt{opinion} = i$, $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$. *Then at the end of Phase 3,* $|M| \geq 0.92|\mathcal{M}|$ *with high probability* $1 - O(1/n^2)$.

In addition, the total mass above exponent $-l$ *is* $\mu_{(>-l)} = \sum_{a.\texttt{exponent} > -l} |a.\texttt{bias}| \leq 0.002|\mathcal{M}|2^{-l}$, *and the total minority mass is* $\beta_- = \sum_{a.\texttt{opinion} = -i} |a.\texttt{bias}| \leq 0.004|\mathcal{M}|2^{-l}$.

3.6. Analysis of final phases

3.6.1. Reserve agent Phases 5 and 6.

We now consider the behavior of the Reserve agents in Phase 5 and Phase 6. We first show that with high probability they are also able to set their $\texttt{sample}$ field during Phase 5.

Lemma 3.6.1. *By the end of Phase 5, all* Reserve *agents have* $\texttt{sample} \neq \perp$, *with high probability* $1 - O(1/n^2)$.

Proof. By Theorem 3.5.2, we have at least $0.92|\mathcal{M}|$ majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$ by the end of Phase 3. Now we can analyze the process of Reserve agents sampling the $\texttt{exponent}$ of the first biased agent they encounter in Phase 5 with Lemma 3.3.7, where subpopulations $A = \{a : a.\texttt{role} = \mathsf{Main}, a.\texttt{bias} = \pm 1\}$ with and $B = \mathcal{R}$, comprising fractions $a \geq 0.92m$ and $b_1 = r$. Then by Lemma 3.3.7, all Reserve agents set their $\texttt{sample}$ within time t, where $t \leq \frac{5\ln n}{2 \cdot 0.92m}$ with high probability $1 - O(1/n^2)$. Thus for appropriate choice of counter constant c_5, this will happen before the first Clock agent advances to the next phase. □

We next show that the Reserve agents are able to bring the exponents of all biased agents down to at most $-l$ during Phase 6.

Lemma 3.6.2. *By the end of Phase 6, all biased agents have* $\texttt{exponent} \leq -l$*, with high probability* $1 - O(1/n^2)$.

Proof. Theorem 3.5.2, the mass above exponent $-l$ is $\mu_{(>-l)} \leq 0.001|\mathcal{M}|2^{-l+1}$. We must show that all of this mass moves down to at least exponent $-l$ via split reactions with the Reserve agents (Line 3 in Phase 6). We consider the following two cases based on whether the size of $A_{-l} = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -l\}$.

In the first case, $|A| > 0.19|\mathcal{M}|$ and we will show all agents get brought down to $\texttt{exponent} \leq -l$. Because of the sampling process in Phase 5, the count of $R_{-l} = \{r : r.\texttt{role} = \mathsf{Reserve}, r.\texttt{sample} = -l\}$ has size at least $|R_{-l}| \geq 0.18|\mathcal{R}|$ with very high probability, by standard Chernoff bounds. If all the agents above exponent $-l$ split down to exponent $-l$, they would have count at most $0.002|\mathcal{M}|$, potentially all coming out of the initial count of R_{-l}. Thus for the entirety of Phase 6, we have

$$|R_{-l}| \geq 0.18|\mathcal{R}| - 0.002|\mathcal{M}| = n(0.18r - 0.002m) \geq n(0.18 \cdot 0.24 - 0.002) \geq 0.04n,$$

using the very high probability bound $r > 0.24$ from Lemma 3.4.2.

In the second case $|A| \leq 0.19|\mathcal{M}|$. Now we cannot make guarantees on the size $|R_{-l}|$, so we will instead reason about $A_{-(l+1)} = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -(l+1)\}$ and $R_{-(l+1)} = \{r : r.\texttt{role} = \mathsf{Reserve}, r.\texttt{sample} = -(l+1)\}$.

We first show that $|A_{-(l+1)}| > 0.59|\mathcal{M}|$. Recall by definition of l and g_l that the majority mass $\beta_+ \geq g_l 2^{-l} \geq 0.4|\mathcal{M}|2^{-l}$. At most $0.001|\mathcal{M}|2^{-l+1}$ can come from exponents above $-l$ and at most $0.19|\mathcal{M}|2^{-l}$ comes from exponent $-l$. Thus there must be at least mass $(0.398 - 0.19)|\mathcal{M}|2^{-l}$ from

$A_{-(l+1)}$ and the exponents below. If $|A_{-(l+1)}| = 0.59|\mathcal{M}|$, then even putting all $0.41|\mathcal{M}|$ remaining Main agents at $\texttt{exponent} = -(l+2)$ would only give mass $0.59|\mathcal{M}|2^{-(l+1)} + 0.41|\mathcal{M}|2^{-(l+2)} = 0.3975|\mathcal{M}|2^{-l}$. Thus we require $|A_{-(l+1)}| > 0.59|\mathcal{M}|$. Again by standard Chernoff bounds from the sampling process of Phase 5, this implies the initial size of $|R_{-(l+1)}| \geq 0.58|\mathcal{R}|$.

If all the agents above exponent $-l$ split down to exponent $-(l+1)$, they would have count at most $0.004|\mathcal{M}|$, potentially all coming out of the initial count of R_{-l}. In addition, we can lose count $|A| \leq 0.19|\mathcal{M}|$ from R_{-l} from additional split reactions. This implies that for the entirety of Phase 6, we have count at least

$$\begin{aligned} |R_{-(l+1)}| &\geq 0.58|\mathcal{R}| - 0.004|\mathcal{M}| - 0.19|\mathcal{M}| = n(0.58r - 0.004m - 0.19m) \\ &\geq n(0.58 \cdot 0.24 - (0.004 + 0.19) \cdot 0.51) \geq 0.04n, \end{aligned}$$

using the very high probability bounds $r > 0.24$ and $m < 0.51$ from Lemma 3.4.2.

Thus in both cases we maintain a count of at least $0.04n$ Reserve agents at level $-l$ or $-(l+1)$. For an agent a with $a.\texttt{exponent} > -l$, the probability that in a given interaction agent a does a split reaction (Line 3 in Phase 6) with an agent in $R_{-l} \cup R_{-(l+1)}$ is at least $\frac{2 \cdot 0.04n \cdot 1}{n^2} = \frac{0.08}{n}$. Now we proceed as in the proofs of Lemma 3.5.14 and Theorem 3.5.1, analyzing the potential $\phi_{(>-l)}$ to show it hits 0 in $O(\log n)$ time.

Recall that for each biased agent a with $a.\texttt{exponent} = -i > -l$ and local potential $\phi_a = 4^{-i+l-1}$, a split reaction changes the potential by $\Delta\phi_a \leq 4^{l-1}(2 \cdot 4^{-i-1} - 4^{-i}) = 4^{l-1}(-\frac{1}{2}4^{-i}) = -\frac{1}{2}\phi_a$. Then in each interaction at parallel time t, the expected change in the potential

$$\begin{aligned} \mathbb{E}[\phi_{(>-l)}(t + 1/n) - \phi_{(>-l)}(t)] &\leq \sum_{a.\texttt{exponent}>-l} \mathbb{P}[a \text{ splits}] \cdot \Delta\phi_a \\ &\leq \sum_{a.\texttt{exponent}>-l} \frac{0.08}{n} \cdot -\frac{1}{2}\phi_a = -\frac{0.04}{n}\phi_{(>-l)}(t). \end{aligned}$$

Thus we have $\mathbb{E}[\phi_{(>-l)}(t + 1/n) | \phi_{(>-l)}(t)] \leq (1 - \frac{0.04}{n})\phi_{(>-l)}(t)$.

When we begin the argument at time t_{start} at the beginning of Phase 6, we start with $\phi_{(>-l)}(t_{\text{start}}) \leq 0.001|\mathcal{M}|$ from the final iteration of Lemma 3.5.15. We will wait until time $t_{\text{end}} = t_{\text{start}} + 75 \ln n$, and now recursively bound the potential after these $75n \ln n$ interactions:

$$\mathbb{E}\left[\phi_{(>-l)}(t_{\text{end}})|\phi_{(>-l)}(t_{\text{start}})\right] \leq \left(1-\frac{0.04}{n}\right)^{75n\ln n}\phi_{(>-l)}(t_{\text{start}})$$

$$\mathbb{E}\left[\phi_{(>-h+q)}(t_{\text{end}})\right] \leq \exp(-0.04\cdot 75\ln n)\cdot 0.001|\mathcal{M}|$$

$$\leq n^{-3}\cdot 0.001mn = O\left(n^{-2}\right)$$

Finally, since $\phi_{(>-l)}$ takes nonnegative integer values, we can apply Markov's Inequality to conclude $\mathbb{P}[\phi_{(>-l)}(t_{\text{end}}) > 0] = O(1/n^2)$. So after $75\ln n$ time in Phase 6, all biased agents have $\texttt{exponent} \leq -l$, with high probability. For appropriate choice of counter constant c_6, this happens before any Clock agent advances to the next phase. □

Recall M is the set of all majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$, where $|M| \geq 0.92|\mathcal{M}|$ at the end of Phase 3 by Theorem 3.5.2. We finally observe that after Phase 6, M is still large.

Lemma 3.6.3. *At the end of Phase 6, $|M| \geq 0.87|\mathcal{M}|$ with very high probability.*

Proof. By Theorem 3.5.2, $|M| \geq 0.92|\mathcal{M}|$ at the end of Phase 3, so the count of all other main agents is at most $0.08|\mathcal{M}|$. By standard Chernoff bounds on the sampling process in Phase 5, the count

$$|\{r : r.\texttt{role} = \textsf{Reserve}, r.\texttt{sample} \notin \{-l, -(l+1), -(l+2)\}\}| \leq 0.09|\mathcal{R}| \leq 0.05|\mathcal{M}|.$$

Split reactions that consume these Reserve agents are the only way to bring agents out of the set M. Thus at the end of Phase 6, the count is still at least

$$|M| \geq 0.92|\mathcal{M}| - 0.09|\mathcal{R}| \geq 0.87|\mathcal{M}|,$$

using $|\mathcal{R}| < \frac{5}{9}|\mathcal{M}|$ with very high probability by Lemma 3.4.2. □

3.6.2. Minority elimination Phases 7 and 8. Next we argue that in Phase 7, the agents in M are able to eliminate all minority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$. Note that these minority agents are able to do a cancel reaction with any agent in M, since by design Phase 7 allows reactions between agents with an exponent gap of at most 2. We first argue that $|M|$ must stay large through the entirety of Phase 7:

Lemma 3.6.4. *At the end of Phase 7,* $|M| \geq 0.8|\mathcal{M}|$.

Proof. We use the bound on minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ from Theorem 3.5.2. This implies the minority count at exponent $-(l+4)$ and above is at most $0.064|\mathcal{M}|$, since a $0.064|\mathcal{M}| \cdot 2^{-(l+4)} \geq \beta$. Note that cancelling with these minority agents is the only way for a majority agent in M to change its state in Phase 7. Using the previous bound from Lemma 3.6.3, the count of M can decrease at most to $|M| = 0.87|\mathcal{M}| - 0.064|\mathcal{M}| \geq 0.8|\mathcal{M}|$. □

Now we argue that these agents in M eliminate all high exponent minority agents.

Lemma 3.6.5. *At the end of Phase 7, all minority agents have* `exponent` $< -(l+2)$*, with high probability* $1 - O(1/n^2)$.

Proof. From Lemma 3.6.2, all minority agents already have `exponent` $\leq -l$. Let $B_{-l}, B_{-(l+1)}$, $B_{-(l+2)}$ be the sets of minority agents with `exponent` $= -l, -(l+1), -(l+2)$, respectively. We argue successively that $|B_{-l}| = 0$, then $|B_{-(l+1)}|$, then $|B_{-(l+2)}| = 0$.

The initial bound on minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ from Theorem 3.5.2 implies that initially $|B_{-l}| \leq 0.004|\mathcal{M}|$. Note that an interaction with any agent in M will bring remove an agent from B_{-l} via one of the Phase 7 cancel reactions. Using the bound $|M| \geq 0.8|\mathcal{M}|$ during the entire phase from Lemma 3.6.4, we can use Lemma 3.3.7 to bound the time for $|B_{-l}| = 0$. We have constants $a = 0.8m$ and $b_1 = 0.004m$. Then by Lemma 3.6.4, with high probability $1 - O(1/n^2)$, the time t_1 to eliminate the count of B_{-l} is at most $t_1 \leq \frac{5 \ln n}{2(0.8m - 0.004m)} \leq 6.41 \ln n$, using the bound $m \geq 0.49$ from Lemma 3.4.2.

Next we wait to eliminate the count of $B_{-(l+1)}$, by a similar argument. Initially the count is at most $|B_{-(l+2)}| \leq 0.008|\mathcal{M}|$, and this will all cancel in at most time t_2. By Lemma 3.6.4, with high probability, $t_2 \leq \frac{5 \ln n}{2(0.8m - 0.008m)} \leq 6.45 \ln n$. The same argument for $B_{-(l+2)}$, initially $|B_{-(l+2)}| \leq 0.016|\mathcal{M}|$, gives $t_3 \leq \frac{5 \ln n}{2(0.8m - 0.016m)} \leq 6.51 \ln n$.

Thus the entire process requires time at most $t_1 + t_2 + t_3 < 20 \ln n$. So for appropriate choice of counter constant c_7, this will happen before the first Clock agent advances to the next phase. □

Now we will prove that the remaining $0.8|\mathcal{M}|$ majority agents in M are able to eliminate all remaining minority agents in Phase 8.

Lemma 3.6.6. *At the end of Phase 8, there are no more minority agents, with high probability* $1 - O(1/n^2)$.

Proof. From Lemma 3.6.4 and Lemma 3.6.5, by the end of Phase 7, we have $|M| \geq 0.8|\mathcal{M}|$ and all minority agents have $\texttt{exponent} < -(l+2)$. We assume in the worst case all $0.2|\mathcal{M}|$ other Main agents have the minority opinion. Note by the consumption reaction in Phase 8, every minority agent will be eliminated by an agent in M that still has $\texttt{full} = \mathsf{False}$. Thus we can apply Lemma 3.3.6, with $a = 0.8m$ and $b = 0.2m$, to conclude that all remaining minority agents are eliminated in time t, where $t \leq \frac{5 \ln n}{2(0.8m - 0.2m)} \leq 8.5 \ln n$, using $m \geq 0.495$ with very high probability from Lemma 3.4.2. So for appropriate choice of counter constant c_8, this will happen before the first Clock agent advances to the next phase. □

3.6.3. Fast Stabilization. We next give a time bound for the stable backup:

Lemma 3.6.7. *The 6-state protocol in Phase 10 stably computes majority in* $O(n \log n)$ *stabilization time, both in expectation and with high probability.*

Proof. Note that agents in the initial state with $\texttt{active} = \mathsf{True}$ and $\texttt{output} \in \{\mathsf{A}, \mathsf{B}\}$ can only change their state by a cancel reaction in Line 3 with another active agent with the opposite opinion.

First we consider the case where one opinion, without loss of generality A, is the majority. We first wait for all active B agents to meet an active A agent and changed their state to active T via Line 3. This is modelled by the "cancel reaction" process described in Lemma 3.3.6, taking expected $O(n)$ time and $O(n \log n)$ time with high probability. At this point, there are no active B agents and at least one active A agent remaining. We next wait for this active A agent to interact with all remaining active T agents, which will then become passive via Line 5. This takes $O(n \log n)$ time in expectation and with high probability by a standard coupon-collector argument, after which point there are only active A agents and passive agents. Finally, we wait for these active A agents to convert all passive agents to $\texttt{output} = \mathsf{A}$ via Line 7, taking another $O(n \log n)$ time by the same coupon-collector argument.

Next we consider the case where the input distribution is a tie. We first wait for all $n/2$ pairs of active A and B agents to cancel via Line 3, taking $O(n)$ time in expectation and $O(n \log n)$ time with probability. Consider the last such interaction. At this point, those two agents have become active T agents, and there are no active A or B agents left. Thus after interacting with one of the T agents, all passive agents will output T via Line 7. This takes $O(n \log n)$ time in expectation and with high probability by the same coupon-collector argument. □

We are now able to combine all the results from previous sections to prove Theorem 3.2.1, giving guarantees on the behavior of the protocol:

THEOREM 3.2.1. *There is a nonuniform population protocol* NONUNIFORM MAJORITY*, using $O(\log n)$ states, that stably computes majority in $O(\log n)$ stabilization time, both in expectation and with high probability.*

PROOF. We first argue that with high probability $1 - O(1/n^2)$, the protocol NONUNIFORM MAJORITY stabilizes to the correct output in $O(\log n)$ time. We consider three cases based on the size of the initial gap $|g|$. For the majority cases where $|g| > 0$, we assume without loss of generality $g > 0$, so A is the majority.

If $|g| \geq 0.025|\mathcal{M}|$?, then by Lemma 3.4.3, we stabilize in Phase 2 to the correct output with high probability $1 - O(1/n^2)$.

If $0 < |g| < 0.025|\mathcal{M}|$, then by Theorem 3.5.2, with high probability $1 - O(1/n^2)$, we end Phase 3 with at least 92% of Main agents in the set M, with the majority opinion and `exponent` $\in \{-l, -(l+1), -(l+2)\}$. Then after Phase 5 and Phase 6, all biased agents have `exponent` $\leq -l$ with high probability $1 - O(1/n^2)$ by Lemma 3.6.2. Then after Phase 7 and Phase 7, there are no more minority agents with high probability $1 - O(1/n^2)$ by Lemma 3.6.6. Thus in Phase 9, the majority opinion $+1$ will spread to all agents by epidemic, and we reach a stable correct configuration where all agents have `opinions` $= \{0, +1\}$ and `output` $= \mathsf{A}$.

If $g = 0$, then by Theorem 3.5.1, with high probability $1 - O(1/n^2)$, we end Phase 3 with all biased agents at `exponent` $= -L$. Thus in Phase 4, we reach a stable correct configuration where all agents have `output` $= \mathsf{T}$, and there are no agents with `exponent` $> -L$ to increment the phase.

Next we justify that NONUNIFORM MAJORITY stably computes majority, since it is always possible to reach a stable correct configuration. By Lemma 3.4.2, we must produce at least 2 Clock agents by the end of Phase 0. Thus we must eventually advance through all timed phases 0, 1, 3, 5, 6, 7, 8 using the `counter` field. Also all agents have left the initial role $\mathsf{Role}_{\mathsf{MCR}}$, otherwise the **Init** of Phase 1 will trigger all agents to move to Phase 10 by epidemic. Thus the sum of bias across all Main agents is the initial gap g, and this key invariant is maintained through all phases.

Using this invariant, if the agents stabilize in Phase 2, they have a consensus on their output, the sign of their bias, and this consensus must be the sign of the sum of the biases, giving the sign of the initial gap. If the agents stabilize in Phase 4, all $|\texttt{bias}| \leq \frac{1}{2^L} \leq \frac{1}{n}$. This implies the gap $|g| \leq |\mathcal{M}| \cdot \frac{1}{n} < 1$, so the gap $g = 0$ and the agents are correctly outputting T. Finally, note that in Phase 8, when the agents set the field $\texttt{full} = \mathsf{True}$, they can no longer actually store in memory the true bias they are holding. For example an agent with $\texttt{opinion} = +1, \texttt{exponent} = -i, \texttt{full} = \mathsf{True}$, is actually representing the interval $\frac{1}{2^{i+1}} \leq \texttt{bias} < \frac{1}{2^i}$. Then we still have the guarantee that if we reach stable consensus in Phase 9, then this consensus is the sign of the sum of the biases and is thus correct. The final case is that we do not stabilize here, and then move to Phase 10, where they agents eventually stabilize to the correct output.

We finally justify that the expected stabilization time is $O(n \log n)$. By Lemma 3.6.7, the stable backup Phase 10 will stabilize in expected $O(n \log n)$ time. Note that the high probability guarantees are all at least $1 - O(1/n^2)$, so the time for the stable backup contributes at most $O\left(\frac{n \log n}{n^2}\right) = o(1)$ to the total expected time. Now by Lemma 3.4.2, with very high probability we have at least $|\mathcal{C}| \geq 0.24n$ Clock agents. In this case, the time upper bounds of Theorem 3.5.9 and upper bounds on the counter times using standard Chernoff bound, imply that every timed phase lasts expected $O(\log n)$ time. If we do not stabilize in the untimed phases, then we also pass through by epidemic expected $O(\log n)$ time. Thus we either stabilize or reach the backup Phase 10 in expected $O(\log n)$ time. There is a final very low probability case that $|\mathsf{Clock}| = o(n)$, but we must at least have $|\mathsf{Clock}| = 2$. Even in this worst case, the time upper bounds of Theorem 3.5.9 and all counter rounds are at most polynomial in n, whereas the low probability of such a small Clock population is smaller than any polynomial. Thus this event adds a negligible contribution, and we conclude the total expected stabilization time is $O(\log n)$. □

3.7. Uniform, stable protocols for majority using more states

The algorithm described in Section 3.2 is **nonuniform**: the set of transitions used for a population of size n depends on the value $\lceil \log n \rceil$. A uniform protocol [**62**, **83**, **85**] is a single set of transitions that can be used in any population size. Since it is "uniformly specified", the transition function is formally defined by a linear-space[6] Turing machine, where the space bound is the maximum space needed to read and write the input and output states. The original model [19] used $O(1)$ states and transitions for all n and so was automatically uniform, but many recent $\omega(1)$ state protocols are nonuniform. With the exception of the uniform variant in [36], all $\omega(1)$ state stable majority protocols are nonuniform [5, 6, 13, 33, 35, 44]. The uniform variant in [36] has a tradeoff parameter s that, when set to $O(1)$ to minimize the states, uses $O(\log n \log\log n)$ states and $O(\log^2 n)$ time.

In this section we show that there is a way to make Nonuniform Majority in Section 3.2 uniform, retaining the $O(\log n)$ time bound, but increasing expected states to $\Theta(\log n \log\log n)$.[7]

3.7.1. Main idea of $O(\log n \log\log n)$ state uniform majority (not handling ties). Since Nonuniform Majority uses the hard-coded value $L = \lceil \log n \rceil$, to make the algorithm uniform, we require a way to estimate $\log n$ and store it in a field L (called `logn` below) stored in each agent. For correctness and speed, it is only required that `logn` be within a constant multiplicative factor of $\log n$.

Gąsieniec and Stachowiak [99] show a uniform $O(\log\log n)$ state, $O(\log n)$ time protocol (both bounds in expectation and with high probability) that computes and stores in each agent a value $\ell \in \mathbb{N}^+$ that, with high probability, is within additive constant $O(1)$ of $\lceil \log\log n \rceil$ (in particular, whp $\ell \geq \lfloor \log\log n \rfloor - 3$ [36, Lemma 8]), so $2^\ell = \Theta(\log n)$. (This is the so-called **junta election** protocol used as a subroutine for a subsequent leader election protocol.) Furthermore, agents approach this estimate from below, propagating the maximum estimate by epidemic $\ell', \ell \to \ell, \ell$ if $\ell' < \ell$. This gives an elegant way to compose the size estimation with a subsequent nonuniform protocol $\mathcal{P}$ that requires the estimate: agents store their estimate `logn` of $\log n$ and use it in $\mathcal{P}$. Whenever an agent's estimate `logn` updates—always getting larger—it simply resets $\mathcal{P}$, i.e., sets the entire state of $\mathcal{P}$ to

[6] That is, the Turing machine is allowed to use a bit more than the space necessary simply to read and write the input and output states, but not significantly more (constant-factor). This allows it to do simple operations, such as integer multiplication, that require more than constant space, without "cheating" by allowing the internal memory usage of the Turing machine to vastly exceed that required to represent the states.

[7] We say "expected" because this protocol has a non-zero probability of using an arbitrarily large number of states. The number of states will be $O(\log n \log\log n)$ in expectation and with high probability.

its initial state. We can then reason as though all agents actually started with their final convergent value of `logn`.[8]

To make our protocol uniform, but remove its correctness in the case of a tie, as we explain below, all agents conduct this size estimation, stored in the field `logn`, in parallel with the majority protocol $\mathcal{P}$ of Section 3.2. Each agent resets $\mathcal{P}$ to its initial state whenever `logn` updates. This gives the stated $O(\log n)$ time bound and $O(\log n \log \log n)$ state bound. Note that in Phase 0, agents count from `counter` $= \Theta(\log n)$ down to 0. It is sufficient to set the constant in the Θ sufficiently large that all agents with high probability receive by epidemic the convergent final value of `logn` significantly before any agent with the same convergent estimate counts down to 0.

Acknowledging that, with small probability, the estimate of $\log n$ could be too low for Phase 4 to be correct, we simply remove Phase 4 and do not attempt to detect ties. So if we permit undefined behavior in the case of a tie (as many existing fast majority protocols do), then this modification of the algorithm otherwise retains stably correct, $O(\log n)$ time behavior, while increasing the state complexity to $O(\log n \log \log n)$.

3.7.2. How to stably compute ties. With low but positive probability, the estimate of $\log n$ could be too small. For most phases of the algorithm, this would merely amplify the probability of error events (e.g., Phase 1 doesn't last long enough for agents to converge on biases $\{-1, 0, +1\}$) that later phases are designed to handle. However, the correctness of Phase 4 (which detects ties) requires agents to have split through at least $\log n$ exponents in Phase 3. Since the population-wide bias doubles each time the whole population splits down one exponent, the only way for the whole population to split through $\log n$ exponents is for there to be a tie (i.e., the population-wide bias is 0, so can double unboundedly many times). In this one part of the algorithm, for correctness we require the estimate to be at least $\log n$ with probability 1. (It can be much greater than $\log n$ without affecting correctness; an overestimate merely slows down the algorithm.)

To correct this error, we will introduce a stable backup size estimate, to be done in Phase 4. Note that there are only a constant number of states with `phase` $= 4$: Clock agents do not store a counter in this phase, and Main agents that stay in this phase must have `bias` $\in \left\{0, \pm\frac{1}{2^L}\right\}$. Thus

[8] One might hope for a stronger form of composition, in which the size estimation **terminates**, i.e., sets an initially False Boolean flag to True only if the size estimation has converged, in order to simply prohibit the downstream protocol $\mathcal{P}$ from starting with an incorrect estimate of $\log n$. However, when both states A and B are initially $\Omega(n)$, this turns out to be impossible; $\Omega(n)$ agents will necessarily set the flag to True in $O(1)$ time, long before the $O(\log n)$ time required for the size estimation to converge [**83**, Theorem 4.1].

we can use an additional $\Theta(\log n)$ states for the agents that are currently in $\texttt{phase} = 4$ to stably estimate the population size. If they detect that their estimate of L was too small, they simply go to the stable backup Phase 10.

Stable computation of $\lfloor \log n \rfloor$. The stable computation of $\log n$ has all agents start in state L_0, where the subscript represents the agent's estimate of $\lfloor \log n \rfloor$. We have the following transitions: for each $i \in \mathbb{N}$, $L_i, L_i \to L_{i+1}, F_{i+1}$ and, for each $0 \leq j < i$, $F_i, F_j \to F_i, F_i$. Among the agents in state L_i, half make it to state L_{i+1}, reaching a maximum of L_k at $k = \lfloor \log_2 n \rfloor$.[9] All remaining F agents receive the maximum value k by epidemic. A very similar protocol was analyzed in [43, Lemma 12]. We give a quick analysis below for the sake of self-containment.

Time analysis of stable computation of $\lfloor \log n \rfloor$. The expected time is $\Theta(n \log n)$. For the upper bound, for each i, if j is the count of L_i agents, then the probability of a $L_i, L_i \to \dots$ reaction is $O(j^2/n^2)$, so for any possible starting count k of L_i agents, it takes expected time at most $O\left(\frac{1}{n}\sum_{j=1}^{k} \frac{n^2}{j^2}\right) = O(n)$ for the count of L_i agents to get from k to 0 or 1. Assuming in the worst case that no $L_{i+1}, L_{i+1} \to \dots$ reaction happens until all $L_i, L_i \to \dots$ reactions complete, then summing over $\lfloor \log n \rfloor$ values of i gives the $O(n \log n)$ time upper bound to converge on one L_k agent, where $k = \lfloor \log n \rfloor$. It takes additional $O(\log n)$ time for the F agents to propagate k by epidemic.

For the lower bound, observe that for each value of i, the **last** reaction $L_i, L_i \to \dots$ occurs when the count of L_i is either 2 (and will increase at most once more), or 3 (and will never increase again). This takes expected time $\Theta(n)$. Since the count of L_i will increase at most once more, at that time (just before the last $L_i, L_i \to \dots$ reaction), the count of L_{i-1} is at most 3, otherwise **two** more L_i's could be produced, and this would not be the last $L_i, L_i \to \dots$ reaction. Thus, to produce the L_i needed for this last $L_i, L_i \to \dots$ reaction, the previous $L_{i-1}, L_{i-1} \to L_i, F_i$ reaction occurs when the count of L_{i-1} is at most 5, also taking $\Theta(n)$ time. Thus the final reaction at each level takes time $\Theta(n)$ and depends on a reaction at the previous level that also takes time $\Theta(n)$, so summing across $\lfloor \log n \rfloor$ levels gives $\Omega(n \log n)$ completion time.

It is shown in [43, Lemma 12] that the time is $O(n \log^2 n)$ with high probability, i.e., slower than the expected time by a $\log n$ factor. Since the probability is $O(1/n^2)$ that we need to rely on this slow backup, even this larger time bound contributes negligibly to our total expected time.

[9] More generally, the unique stable configuration encodes the full population size n in binary in the following distributed way: for each position i of a 1 in the binary expansion of n, there is one agent L_i. Thus, these remaining agents lack the space to participate in propagating the value $k = \lfloor \log n \rfloor$ by epidemic, but there are $\Omega(n)$ followers F to complete the epidemic quickly.

3.7.3. Challenges in creating $O(\log n)$ state uniform algorithm. It is worth discussing some ideas for adjusting the uniform protocol described above to attempt to reduce its space complexity to $O(\log n)$ states. The primary challenge is to enable the population size n to be estimated without storing the estimate in any agent that participates in the main algorithm (i.e., the agent has role Main, Clock, or Reserve). If agents in the main algorithm do not store the size, then by [83, Theorem 4.1] they will provably go haywire initially, with agents in every phase, totally unsynchronized, and require the size estimating agents to reset them after having converged on a size estimate that is $\Omega(\log n)$.

The following method would let the Size agents reset main algorithm agents, without actually having to store an estimate of $\log n$ in the algorithm agents, but it only works with high probability. The size estimating agents could start a junta-driven clock as in [99], which is reset whenever they update their size estimate. Then, as long as there are $\Omega(n)$ Size agents, they could for a phase timed to last $\Theta(\log n)$ time, reset the algorithm agents by **direct** communication (instead of by epidemic). This could put the algorithm agents in a quiescent state where they do not interact with each other, but merely wait for the Size agents to exit the resetting phase, indicating that the algorithm agents are able to start interacting again. Since there are $\Omega(n)$ size-estimating agents, each non-size-estimating agent will encounter at least one of them with high probability in $O(\log n)$ time.

The problem is that the reset signal is not guaranteed to reach every algorithm agent. There is some small chance that a Main agent with a bias different from its `input` does not encounter a Size agent in the resetting phase, so is never reset. The algorithm from that point on could reach an incorrect result when the agent interacts with properly reset agents, since the sum of biases across the population has changed. In our algorithm, by "labeling" each reset with the value `logn`, we ensure that no matter what states the algorithm agents find themselves in during the initial chaos before size computation converges, every one of them is guaranteed to be reset one last time with the same value of `logn`. The high-probability resetting described above seems like a strategy that could work to create a high probability uniform protocol using $O(\log n)$ time and states, though we have not thoroughly explored the possibility.

But it seems difficult to achieve probability-1 correctness using the technique of "reset the whole majority algorithm whenever the size estimate updates," without multiplying the state complexity by the number of possible values of `logn`. Since we did not need $\lfloor \log n \rfloor$ exactly, but only a value that is $\Theta(\log n)$, we paid only $\Theta(\log \log n)$ multiplicative overhead for the size estimate, but it's not straightforward to see how to avoid this using the resetting technique. Of course, one could imagine that the savings could come from reducing the state complexity of the main majority-computing agents in the nonuniform algorithm. However, reducing the nonuniform algorithm's state complexity to below the $\Omega(\log n)$ lower bound of [6] would provably require the algorithm to be not monotonic or not output dominant. (See Section 3.8 for a discussion of those concepts.) Another possible approach is to intertwine more carefully the logic of the majority algorithm with the size estimation, to more gracefully handle size estimate updates without needing to reset the entire majority algorithm.

3.8. Conclusion

There are two major open problems remaining concerning the majority problem for population protocols.

3.8.1. Uniform $O(\log n)$-time, $O(\log n)$-state majority protocol. Our main $O(\log n)$ state protocol, described in Section 3.2, is **nonuniform**: all agents have the value $\lceil \log n \rceil$ encoded in their transition function. The uniform version of our protocol described in Section 3.7 uses $O(\log n \log \log n)$ states. It remains open to find a uniform protocol that uses $O(\log n)$ time and states.

3.8.2. Unconditional $\Omega(\log n)$ state lower bound for stable majority protocols. The lower bound of $\Omega(\log n)$ states for (roughly) sublinear time majority protocols shown by Alistair, Aspnes, and Gelashvili [6] applies only to stable protocols satisfying two conditions: **monotonicity** and **output dominance**.

Recall that a **uniform** protocol is one where a single set of transitions works for all population sizes; nonuniform protocols typically violate this by having an estimate of the population size (e.g., the integer $\lceil \log n \rceil$) embedded in the transition function. Monotonicity is a much weaker form of uniformity satisfied by nearly all known nonuniform protocols. While allowing different transitions as the population size grows, monotonicity requires that the transitions used for population size n must also be correct for all smaller population sizes $n' < n$ (i.e., an **overestimate** of the size cannot hurt), and furthermore that the transitions be no slower on populations of size n' than on populations of size n (though the transitions designed for size n may be slower on size n' than the transitions intended for size n'). Typically the nonuniform estimate of $\log n$ is used by the protocol to synchronize phases taking time $\approx T = \Theta(\log n)$, by having each agent individually count from T down to 0 (a so-called "leaderless phase clock", our Standard Counter Subroutine). $\Theta(\log n)$ is the time required for an epidemic to reach the whole population and communicate some message to all agents before the phase ends. Most errors in such protocols are the result of some agent not receiving a message before a phase ends, i.e., the clock runs atypically faster than the epidemic. If the size estimate is significantly **larger** than $\log n$, this slows the protocol down, but only increases the probability that all agents receive intended epidemic messages before a phase ends; thus such protocols are monotone.

In our nonuniform protocol Nonuniform Majority, which gives each agent the value $L = \lceil \log n \rceil$, giving a different value of L does not disrupt the stability (only the speed) of the protocol, with one exception: Phase 3 must go for at least $\log n$ exponents, or else (i.e., if L is an underestimate) Phase 4 could incorrectly report a tie. If we consider running Phase 3 on a smaller population size n' than the size n for which it was designed, $L = \lceil \log n \rceil$ of $\lceil \log n' \rceil$ will be an **overestimate**, which does not disrupt correctness. Furthermore, since Clock agents are counting to L in each case, they are just as fast on population size n' as on size n. This implies that Phase 3, thus the whole protocol Nonuniform Majority, is monotone.

Output dominance references the concept of a **stable** configuration $\mathbf{c}$, in which all agents have a consensus opinion that cannot change in any configuration subsequently reachable from $\mathbf{c}$. A protocol is **output dominant** if in any stable configuration $\mathbf{c}$, adding more agents with states already present in $\mathbf{c}$ maintains the property that every reachable stable configuration has the same output (though it may disrupt the stability of $\mathbf{c}$). This condition holds for all known stable majority protocols, including that described in this paper, because they obey the stronger condition that adding states already present in $\mathbf{c}$ does not even disrupt its stability. Such protocols are based on the idea that two agents with the same opinion cannot create the opposite opinion, so stabilization happens exactly when all agents first converge on a consensus output.

To see that our protocol is output dominant, define a configuration to be **silent** if no transition is applicable (i.e., all pairs of agents have a null interaction); clearly a silent configuration is also stable. Although the definition of stable computation allows non-null transitions to continue happening in a stable configuration, many existing stable protocols have the stronger property that they reach a silent configuration with probability 1, including our protocol. It is straightforward to see that any silent configuration has the property required for output dominance, since if no pair of states in a configuration can interact nontrivially, their counts can be increased while maintaining this property. (One must rule out the special case of a state with count 1 that can interact with another copy of itself, which does not occur in our protocol's stable configurations.)

Monotonicity can be seen as a natural condition that all "reasonable" non-uniform protocols must satisfy, but output dominance arose as an artifact that was required for the lower bound proof strategy of [6]. It remains open to prove an unconditional (i.e., removing the condition of output dominance) lower bound of $\Omega(\log n)$ states for **any** stable monotone majority protocol taking polylogarithmic time, or to show a stable polylogarithmic time monotone majority protocol using $o(\log n)$ states, which necessarily violates output dominance. If the unconditional lower bound holds, then our protocol is simultaneously optimal for both time and states. Otherwise, it may be possible to use $o(\log n)$ states to stably compute majority in polylogarithmic stabilization time with a non-output-dominant protocol. In this case, there may be an algorithm simultaneously optimal for both time and states, or there may be a tradeoff.

CHAPTER 4

Time-Optimal Self-Stabilizing Leader Election in Population Protocols

This chapter is joint work with Janna Burman, Ho-Lin Chen, Hsueh-Ping Chen, David Doty, Thomas Nowak, and Chuan Xu. It was originally published as [**53**].

4.1. Introduction

4.1.1. Reliable leader election. This chapter studies leader election in the context of **reliability**. What if agents are prone to memory or communication errors? What if errors cannot be directly detected, so agents cannot be re-initialized in response?

We adopt the approach of self-stabilization [**24**, **79**]. A protocol is called **self-stabilizing** if it stabilizes with probability 1 from an **arbitrary** configuration[1] (resulting from any number of transient faults). Non-self-stabilizing (a.k.a., **initialized**) leader election is easily solvable using only two states $\{L, F\}$, see Section 1.1.1. However, this protocol fails (as do nearly all other published leader election protocols) in the self-stabilizing setting from an all-F configuration, see Fig. 1.1a. Thus, any self-stabilizing leader election (SSLE) protocol must be able not only to reduce multiple potential leaders to one, but also to create new leaders. A particular challenge here is a careful verification of leader absence, to avoid creating excess leaders forever.

Because of this challenge, in any SSLE protocol, agents must know the **exact** population size n, and the number of states must be at least n [**54**] (Theorem 4.2.1 in the preliminaries section). Despite the original assumption of constant space, population protocols with linear space (merely $O(\log n)$ bits of memory) may be useful in practice, similarly to distributed algorithms in other models (message passing, radio networks, etc.). One may now imagine such memory-equipped devices communicating in a way as agents do in population protocols [**117**,**140**]. Think of a group of mobile devices (like sensors, drones or smart phones) operating in different types of rescue, military

[1] For a self-stabilizing protocol, it is equivalent to consider probability 1 and fixed probability $p > 0$ of correctness; See Section 4.2.

or other monitoring operations (of traffic, pollution, agriculture, wild-life, etc.). Such networks may be expected to operate in harsh inaccessible environments, while being highly reliable and efficient. This requires an efficient "strong" fault-tolerance in form of automatic recovery provided by self-stabilization. Moreover, even if one considers only protocols with polylog(n) states interesting, it remains an interesting fact that such protocols cannot solve SSLE.

Finally, self-stabilizing algorithms are easier to compose [80, 81]. Composition is in general difficult for population protocols [61,146], since they lack a mechanism to detect when one computation has finished before beginning another. However, a self-stabilizing protocol S can be composed with a prior computation P, which may have set the states of S in some unknown way before P stabilized, c.f. [24, Section 4], [15, Theorem 3.5].

4.1.2. Problem variants. To circumvent the necessary dependence on population size n, previous work has considered relaxations of the original problem. One approach, which requires agents only to know an upper bound on n, is to relax the requirement of self-stabilization: **loose-stabilization** requires only that a unique leader persists for a long time after a stabilization, but not forever [155,157,160]. Other papers study leader election in more general and powerful models than population protocols, which allow extra computational ability not subject to the limitations of the standard model. One such model assumes an external entity, called an **oracle**, giving clues to agents about the existence of leaders [28, 97]. Other generalized models include **mediated population protocols** [131], allowing additional shared memory for every pair of agents, and the k-interaction model [169], where agents interact in groups of size 2 to k.

While this paper considers only the complete graph (the most difficult case), other work considers protocols that assume a particular non-complete graph topology. In rings and regular graphs with constant degree, SSLE is feasible even with only a constant state space [24,64,65,170]. In another recent related work [161], the authors study the feasibility requirements of SSLE in arbitrary graphs, as well as the problem of **ranking** that we also study (see below). They show how to adapt protocols in [28, 54] into protocols for an arbitrary (and unknown) connected graph topology (without any oracles, but knowing n).

4.1.3. Contribution. We initiate the study of the limits of time efficiency or the time/space trade-offs for SSLE in the standard population protocol model, in the complete interaction graph.

Table 4.1. Overview of time and space (number of states) complexities of self-stabilizing leader election protocols (which all also solve ranking). For the silent protocols, the silence time also obeys the stated upper bound. Times are measured as parallel time until stabilization both in expectation and with high probability (WHP is defined as probability $1 - O(1/n)$, but implies a guarantee for any $1 - O(1/n^c)$, see Section 4.2). Entries marked with * are asymptotically optimal in their class (silent/non-silent); see Observation 4.2.6. The final two rows really describe the same protocol Sublinear-Time-SSR; it is parameterized by the positive integer H; setting $H = \Theta(\log n)$ gives the time-optimal $O(\log n)$ time protocol.

protocol	**expected time**	**WHP time**	**states**	**silent**
Silent-n-state-SSR [**54**]	$\Theta(n^2)$	$\Theta(n^2)$	* n	yes
Optimal-Silent-SSR (Sec. 4.4)	* $\Theta(n)$	* $\Theta(n \log n)$	* $O(n)$	yes
Sublinear-Time-SSR (Sec. 4.5)	* $\Theta(\log n)$	* $\Theta(\log n)$	$\exp\big(O(n^{\log n} \cdot \log n)\big)$	no
Sublinear-Time-SSR (Sec. 4.5)	$\Theta(H \cdot n^{\frac{1}{H+1}})$	$\Theta(\log n \cdot n^{\frac{1}{H+1}})$	$\Theta(n^{\Theta(n^H)} \log n)$	no

The most related protocol, of Cai, Izumi, and Wada [**54**] (Silent-n-state-SSR, Protocol 4.1), given for complete graphs, uses exactly n states and $\Theta(n^2)$ expected parallel time (see Theorem 4.2.4), exponentially slower than the polylog(n)-time non-self-stabilizing existing solutions [**40**, **99**, **100**, **120**, **159**]. Our main results are two faster protocols, each making a different time/space tradeoff.

Our protocols, along with that of [**54**], are summarized in Table 4.1. These main results are later proven as Theorem 4.4.3 and Theorem 4.5.7. Both expected time and high-probability time are analyzed. Any **silent** protocol (one guaranteed to reach a configuration where no agent subsequently changes states) must use $\Omega(n)$ parallel time in expectation (Observation 4.2.6). This lower bound has helped to guide our search for sublinear-time protocols, since it rules out ideas that, if they worked, would be silent. Thus Optimal-Silent-SSR is time- and space-optimal for the class of silent protocols.

Sublinear-Time-SSR is actually a family of sublinear time protocols that, depending on a parameter H that can be set to an integer between 1 and $\Theta(\log n)$, causes the algorithm's running time to lie somewhere in $O(\sqrt{n})$ and $O(\log n)$, while using more states the larger H is; setting $H = \Theta(\log n)$ gives the time-optimal $O(\log n)$ time protocol. However, even with $H = 1$, it requires exponential states. It remains open to find a sublinear-time SSLE protocol that uses sub-exponential states. We note that any protocol solving SSLE requires $\Omega(\log n)$ time: from any configuration where all n agents are leaders, by a coupon collector argument, it takes $\Omega(\log n)$ time for $n - 1$ of them to interact and become followers. (This argument uses the self-stabilizing assumption that "all-leaders" is a valid initial configuration; otherwise, for **initialized** leader election, it requires considerably more care to prove an $\Omega(\log n)$ time lower bound [**159**].)

For some intuition behind the parameterized running times for Sublinear-Time-SSR, the protocol works by detecting "name collisions" between agents, communicated via paths of length $H+1$. For example, $H=0$ corresponds to the simple linear-time algorithm that relies on two agents s, a with the same name directly interacting, i.e., the path $s \to a$. $H=1$ means that s first interacts with a third agent b, who then interacts with a, i.e., the path $s \to b \to a$. To analyze the time for this process to detect a name collision, consider the following "bounded epidemic" protocol. The "source" agent s that starts the epidemic is in state 0, and all others are in state ∞, and they interact by $i, j \to i, i+1$ whenever $i < j$. The time τ_k is the first time some target agent a has state $\leq k$. In other words, this agent has heard the epidemic via a path from the source of length at most k. We have $\mathbb{E}[\tau_1] = O(n)$, since a must meet s directly. An iterative process can then show $\mathbb{E}[\tau_2] = O(\sqrt{n})$, and more generally $\mathbb{E}[\tau_k] = O(kn^{1/k})$. τ_n is the hitting time for the standard epidemic process, since the path from any agent to the source can be at most n. However, with high probability, the epidemic process will reach each agent via a path of length $O(\log n)$, so it follows that $\tau_k = O(\log n)$ if $k = \Omega(\log n)$, so setting $H = \Theta(\log n)$ will detect this name collisions in $O(\log n)$ time. Bounds on τ_k are given as Lemma 4.2.10 and Lemma 4.2.11.

Ranking. All protocols in the table solve a more difficult problem than leader election: **ranking** the agents by assigning them the IDs $1, \ldots, n$. Ranking is helpful for SSLE because it gives a deterministic way to detect the absence of a state (such as the leader state). If any rank is absent, the pigeonhole principle ensures multiple agents have the same rank, reducing the task of absence detection to that of collision detection.

Collision detection is accomplished easily in $O(n)$ time by waiting for the colliding agents to meet, which is done by Optimal-Silent-SSR. Achieving stable collision detection in optimal $O(\log n)$ time is key to our fast protocol Sublinear-Time-SSR. This collision detection problem is interesting in its own right, see Conclusion.

Ranking is similar to the **naming** problem of assigning each agent a unique "name" (ID) [52, 129], but is strictly stronger since each agent furthermore knows the order of its name relative to those of other agents. Naming is related to leader election: if each agent can determine whether its name is "smallest" in the population, then the unique agent with the smallest name becomes the leader. However, it may not be straightforward to determine whether some agent exists with a smaller name; much of the logic in the faster ranking algorithm Sublinear-Time-SSR is devoted to propagating the set of names of other agents while determining whether the adversary has planted "ghost" names in this set that do not actually belong to any agent. On the other hand, any ranking algorithm automatically solves both the naming and leader election problems: ranks are unique names, and the agent with rank 1 can be assigned as the leader. (Observation 4.2.5 shows that the converse does not hold.)

4.2. Preliminaries

If a self-stabilizing protocol stabilizes with high probability, then we can make this high probability bound $1 - O(1/n^c)$ for any constant c. This is because in the low probability of an error, we can repeat the argument, using the current configuration as the initial configuration. Each of these potential repetitions gives a new "epoch", where the Markovian property of the model ensures the events of stabilizing in each epoch are independent. Thus the protocol will stabilize after at most c of these "epochs" with probability $1 - O(1/n^c)$. By the same argument, if a self-stabilizing protocol can stabilize with any positive probability $p > 0$, it will eventually stabilize with probability 1.

Leader election and ranking. The two tasks we study in this paper are self-stabilizing **leader election** (SSLE) and **ranking** (SSR). For both, the self-stabilizing requirement states that from any configuration, a stably correct configuration must be reached with probability 1. For leader election, each agent has a field `leader` with potential values $\{\mathsf{Yes}, \mathsf{No}\}$, and a **correct** configuration is defined where exactly one agent a has $a.\texttt{leader} = \mathsf{Yes}$.[2] For ranking, each agent has a field `rank` with potential values $\{1, \ldots, n\}$, and a **correct** configuration is defined as one where, for each $r \in \{1, \ldots, n\}$, exactly one agent a has $a.\texttt{rank} = r$. As noted in Sec. 6.1, any protocol solving SSR also solves SSLE by assigning `leader` to Yes if and only if $\texttt{rank} = 1$; for brevity we omit the `leader` bit from our protocols and focus solely on the ranking problem. Observation 4.2.5 shows that the converse does not hold.

SSLE Protocol from [54]. Protocol 4.1 shows the original SSLE protocol from [54]. We display it here to introduce our pseudocode style and make it clear that this protocol is also solving ranking.[3]

[2] We do not stipulate the stricter requirement that one agent stays the leader, rather than letting the $\texttt{leader} = \mathsf{Yes}$ bit swap among agents, but we claim these problems are equivalent due to the complete interaction graph. A protocol solving SSLE can also "immobilize" the unique $\texttt{leader} = \mathsf{Yes}$ bit by replacing any transition $(x, y) \to (w, z)$, where $x.\texttt{leader} = z.\texttt{leader} = \mathsf{Yes}$ and $y.\texttt{leader} = w.\texttt{leader} = \mathsf{No}$, with $(x, y) \to (z, w)$.

[3] Their state set $\{0, \ldots, n-1\}$ from [54] is clearly equivalent to our formal definition of a $\texttt{rank} \in \{1, \ldots, n\}$, but simplifies the modular arithmetic.

The convergence proofs in [54] did not consider our definition of parallel time via the uniform random scheduler. Thus we also include proofs that Silent-n-state-SSR stabilizes in $\Theta(n^2)$ time, in expectation and WHP (see Theorem 4.2.4). It is straightforward to argue an $\Omega(n^2)$ time lower bound from a configuration with 2 agents at $\texttt{rank} = 0$, 0 agents at $\texttt{rank} = n-1$, and 1 agent at every other $\texttt{rank}$. This requires $n-1$ consecutive "bottleneck" transitions, each moving an agent up by one rank starting at 0. Each takes expected time $\Theta(n)$ since two specific agents (the two with the same rank) must interact directly. Our arguments for a $O(n^2)$ time upper bound give a separate proof of correctness from that in [54], reasoning about a barrier rank that is never crossed.

Protocol 4.1 Silent-n-state-SSR, for initiator a interacting with responder b
Fields: $\texttt{rank} \in \{0, \ldots, n-1\}$

```
if $a.\texttt{rank} = b.\texttt{rank}$ then
    $b.\texttt{rank} \leftarrow (b.\texttt{rank} + 1) \bmod n$
```

Cai, Izumi, and Wada [54] show that the state complexity of this protocol is optimal. A protocol is **strongly nonuniform** if, for any $n_1 < n_2$, a different set of transitions is used for populations of size n_1 and those of size n_2 (intuitively, the agents hardcode the exact value n).

Theorem 4.2.1 ([54]). *Any population protocol solving SSLE has $\geq n$ states and is strongly nonuniform.*

It is worth seeing why any SSLE protocol must be strongly nonuniform. Suppose the same transitions are used in population sizes $n_1 < n_2$. By identifying in a single-leader population of size n_2 any subpopulation of size n_1 that does not contain the leader, sufficiently many interactions strictly within the subpopulation must eventually produce a second leader. Thus the full population cannot be stable. These conflicting requirements to both produce a new leader from a leaderless configuration, but also make sure the single-leader configuration is stable, is the key new challenge of leader election in the self-stabilizing setting. Protocols solving SSLE circumvent this error by using knowledge of the exact population size n.

We analyze the time complexity of the Silent-n-state-SSR protocol. It crucially relies on the fact that every (initial) configuration guarantees the existence of a "barrier rank" that is never exceeded by an interaction, disallowing indefinite cycles. More formally, denote by $m_i(C)$ the number of agents with rank i in configuration C. We will show that, starting from a configuration C_0, there exists some k such that

$$\forall r \in \{0,\dots,n-1\}\colon \quad \sum_{d=0}^{r} m_{(k-d) \bmod n}(C) \le r+1 \tag{4.1}$$

for all configurations C that are reachable from C_0. Then k is a barrier rank as it guarantees $m_k(C) \le 1$ during the whole execution.

Lemma 4.2.2. *For every configuration C of* Silent-n-state-SSR*, there exists some $k \in \{0,\dots,n-1\}$ such that (4.1) holds in C.*

Proof. Define $S_i = \sum_{j=0}^{i} \big(m_j(C) - 1\big)$ for $i \in \{0,\dots,n-1\}$. Note that $S_{n-1} = 0$ since $\sum_{j=1}^{n-1} m_j(C) = n$. Choose $k \in \{0,\dots,n-1\}$ such that S_k is minimal. For $r \le k$ we have

$$\sum_{d=0}^{r} m_{(k-d) \bmod n}(C) = \sum_{j=k-r}^{k} m_j(C) = (r+1) + \sum_{j=k-r}^{k} \big(m_j(C) - 1\big) = (r+1) + (S_k - S_{k-r+1}) \le r+1$$

since $S_k \le S_{k-r+1}$. For $r > k$ we have

$$\sum_{d=0}^{r} m_{(k-d) \bmod n}(C) = \sum_{j=0}^{k} m_j(C) + \sum_{j=n-r+k}^{n-1} m_j(C) = (r+1) + (S_k + S_{n-1} - S_{n-r+k-1}) \le r+1$$

since $S_{n-1} = 0$ and $S_k \le S_{n-r+k-1}$. □

Lemma 4.2.3. *Let $k \in \{0,\dots,n-1\}$. If (4.1) holds for k in configuration C, then (4.1) holds for k in all configurations reachable from C.*

Proof. Without loss of generality we assume $k = n-1$ by cyclic permutation of the ranks. It suffices to show the lemma's statement for a direct successor configuration C' of C. Let i be the rank of the initiator and j that of the responder in the interaction leading from C to C'. If $i \neq j$, then $m_\ell(C') = m_\ell(C)$ for all $\ell \in \{0,\dots,n-1\}$ and thus the statement follows from the hypothesis on C. So assume $i = j$ in the rest of the proof. Then $i < n-1$ since $m_{n-1}(C) \le 1$ using (4.1) with $r = 0$. Thus $m_i(C') = m_i(C) - 1$ and $m_{i+1}(C') = m_{i+1}(C) + 1$. This means that all sums except for $r = n - i - 2$ in (4.1) remain constant when passing from C to C'.

To bound the sum for $r = n-i-2$, we prove that we actually have $\sum_{d=0}^{r} m_{k-d}(C) \le r$, which then implies $\sum_{d=0}^{r} m_{k-d}(C') = 1 + \sum_{d=0}^{r} m_{k-d}(C) \le r+1$ as required. In fact, if $\sum_{d=0}^{r} m_{k-d}(C) =$

$r+1$, then

$$m_i(C) = \sum_{d=0}^{r+1} m_{k-d}(C) - \sum_{d=0}^{r} m_{k-d}(C) \leq (r+2) - (r+1) = 1 \ ,$$

which is a contradiction to the fact that there are two different agents with rank i in configuration C that can interact. □

Theorem 4.2.4. *The* Silent-n-state-SSR *protocol solves self-stabilizing ranking. The silence time from the worst-case initial configuration is* $\Theta(n^2)$ *in expectation and with probability* $1 - \exp(-\Theta(n))$.

Proof. We first prove the time lower bound. We let the protocol start in initial configuration C_0 with $m_1(C_0) = 2$, $m_{n-1}(C_0) = 0$, and $m_i(C_0) = 1$ for all $i \in \{1, \ldots, n-2\}$. If we define $T_{-1}, T_0, T_1, \ldots, T_{n-1}$ by $T_{-1} = 0$ and $T_s = \min\{t \geq T_{s-1} \mid m_s(t) = 1\}$, then T_i is the first time that two agents with rank $i-1$ interact. As there is at most one rank with more than one agent, the difference $T_i - T_{i-1}$ is equal to the number of interactions until the two agents with rank i interact, which is a geometric random variable with probability of success $p = 1/\binom{n}{2}$. Then $\mathbb{E}\, T_{n-1} = \sum_{i=0}^{n-1} \mathbb{E}(T_i - T_{i-1}) = (n-1)\binom{n}{2} = \Theta(n^3)$. As T_{n-1} is the sum of independent geometric random variables, we will also use Theorem 3.1 from [**115**], where $\mu = \mathbb{E}\, T_{n-1}$ and $\lambda = 1/2$, to show the lower tail bound

$$\mathbb{P}[T_{n-1} \leq \lambda\mu] \leq \exp\left(-p\mu(\lambda - 1 - \ln\lambda)\right) = \exp(-\Theta(n)).$$

Thus the convergence time from C_0 is $\Omega(n^2)$ ($\Omega(n^3)$ interactions) in expectation and with probability $1 - \exp(-\Theta(n))$. This concludes the proof of the time lower bound.

We now turn to the time upper bound. Now let the initial configuration C_0 be arbitrary. By Lemmas 4.2.2 and 4.2.3 we get the existence of some $k \in \{0, \ldots, n-1\}$ such that (4.1) holds for all $t \in \mathbb{N}_0$. In particular $m_k(t) \leq 1$, which means that rank k is indeed a barrier for every execution starting in C_0. Without loss of generality, by cyclic permutation of the ranks, we assume $k = n-1$. We inductively define $T_{-1}, T_0, T_1, \ldots, T_{n-1}$ by $T_{-1} = 0$ and $T_s = \min\{t \geq T_{s-1} \mid m_s(t) = 1\}$. Then $m_s(t) = 1$ for all $t \geq T_s$.

Now $T_s - T_{s-1}$ is the number of interactions for rank collisions to reduce the count m_s to 1. This is stochastically dominated by the convergence of the classic "fratricide" leader election ($L + L \to L + F$, starting from all L). Letting F be the number of interactions for the fratricide process to converge to a single L, we have F is the sum of independent (non-identical) geometric random variables, where $\mathbb{E}\, F = \Theta(n^2)$ and the minimum parameter $p_* = 1/\binom{n}{2}$. We then have T_{n-1} stochastically dominated by a sum $S = \sum_{i=1}^{n-1} F_i$ of $n-1$ independent copies of F (which is a sum of $\Theta(n^2)$ geometric random variables). Then $\mathbb{E}\, T_{n-1} \leq \mathbb{E}\, S = (n-1)\mathbb{E}\, F = O(n^3)$. Also, we can use Theorem 2.1 of [115] to show an upper tail bound (where $\mu = \mathbb{E}\, S$ and $\lambda = 3/2$)

$$\mathbb{P}[S \geq \lambda\mu] \leq \exp\left(-p_* \mu(\lambda - 1 - \ln\lambda)\right) = \exp(-\Theta(n)).$$

Thus from any initial configuration C_0, the convergence time is $O(n^2)$ ($O(n^3)$ interactions) in expectation and with probability $1 - \exp(-\Theta(n))$. □

Observation 4.2.5. *There is a silent SSLE protocol whose states cannot be assigned ranks such that it also solves the SSR problem.*

Proof. The following protocol solves silent SSLE for a population size $n = 3$. (Note the construction from [54] in Protocol 4.1 is strictly better protocol, the purpose of this construction is just to show an example solving silent SSLE without solving ranking).

The state set is $S = \{l\} \cup F$, where $F = \{f_0, f_1, f_2, f_3, f_4\}$. There will be exactly 5 silent configurations of the three agents: $\{l, f_0, f_1\}, \{l, f_1, f_2\}, \{l, f_2, f_3\}, \{l, f_3, f_4\}, \{l, f_4, f_0\}$. (In other words, a leader l and two distinct followers f_i, f_j with $|i - j| \equiv 1 \mod 5$).

This can be easily accomplished by adding transitions from (s, s) (for all states $s \in S$) and from (f_i, f_j) (for all $f_i, f_j \in F$ with $|i - j| \not\equiv 1 \mod 5$) to a uniform random pair of states $(a, b) \in S \times S$. It is easily observed that starting from any configuration of 3 agents, this protocol must stabilize to one of the 5 silent configurations above, and thus solves SSLE.

However, there is no way to consistently assign the ranks $1, 2, 3$ to the states in the silent configurations. If WLOG we denote l to be rank 1, then we must assign ranks 2 or 3 to each state in F. But since $|F|$ is odd, every such assignment places two states f_i, f_j in the same rank where $|i - j| \equiv 1 \mod 5$. Since $\{l, f_i, f_j\}$ is a silent configuration that is incorrectly ranked, we have a contradiction. □

Note that Observation 4.2.5 does not rule out a more sophisticated way to transform any SSLE protocol to a SSR protocol. It merely rules out the most simple approach: assigning ranks to the existing states, without otherwise changing the protocol.

Silent protocols. A configuration C is **silent** if no transition is applicable to it (put another way, every pair of states present in C has only a null transition that does not alter the configuration). A self-stabilizing protocol is **silent** if, with probability 1, it reaches a silent configuration from every configuration. Since convergence time $\leq$ stabilization time $\leq$ silence time, the following bound applies to all three.

Observation 4.2.6. *Any silent SSLE protocol has $\Omega(n)$ expected convergence time and for any $\alpha > 0$, probability $\geq \frac{1}{2}n^{-3\alpha}$ to require $\geq \alpha n \ln n$ convergence time.*

For example, letting $\alpha = 1/3$, with probability $\geq \frac{1}{2n}$ the protocol requires $\geq \frac{1}{3}n \ln n$ time.

Proof. Let C be a silent configuration with a single agent in a leader state ℓ. Let C' be the configuration obtained by picking an arbitrary non-leader agent in C and setting its state also to ℓ. Since C is silent and the states in C' are a subset of those in C, no state in C' other than ℓ can interact nontrivially with ℓ. So the two ℓ's in C' must interact to reduce the count of ℓ. The number of interactions for this to happen is geometric with $\mathbb{P}[\text{success}] = 1/\binom{n}{2} = \frac{2}{n(n-1)} < 3/n^2$, so expected time $\geq n/3$ and for any $\alpha > 0$, at least $\alpha n^2 \ln n$ interactions ($\alpha n \ln n$ time) are required with probability at least

$$\left(1 - 3/n^2\right)^{\alpha n^2 \ln n} \geq \frac{1}{2}e^{-3\alpha \ln n} = \frac{1}{2}n^{-3\alpha}. \qquad \square$$

4.2.1. Probabilistic Tools. We consider the epidemic process from Section 1.1.2), arising whenever agents have a variable `infected` $\in \{\mathsf{True}, \mathsf{False}\}$ updating as

$$a.\texttt{infected}, b.\texttt{infected} \leftarrow (a.\texttt{infected} \ \vee \ b.\texttt{infected}).$$

Mocquard, Sericola, Robert, and Anceaume [**134**] gave an in-depth analysis of the two-way epidemic process. This analysis gives upper bounds for many processes in our protocols. In any process where some field value is propagated this way, in addition to other transitions or initial conditions with more than one infected agent, which may speed up the propagation but cannot hinder it, we denote that process a **superepidemic**. The number of interactions X to spread to the whole population is clearly stochastically dominated by the two-way epidemic.[4] Consequently, we state the results below for normal epidemics, but use them to reason about superepidemics.

[4] We note that this sort of process, which is stochastically dominated by a "pure epidemic", is generally the sort of process studied in most population protocols papers that use the term **epidemic**.

The next lemma uses results of [134] to prove a simplified upper tail bound.

Lemma 4.2.7 ([134]). *Starting from a population of size n with a single infected agent, let T_n be the number of interactions until $a.\mathtt{infected} = \mathsf{True}$ for all $a \in \mathcal{A}$. Then $\mathbb{E}[T_n] = (n-1)H_{n-1} \sim n \ln n$, and for $n \geq 8$ and $\delta \geq 0$,*

$$\mathbb{P}[T_n > (1+\delta)\mathbb{E}[T_n]] \leq 2.5\ln(n) \cdot n^{-2\delta}.$$

Proof. From [134] we have $\mathbb{E}[T_n] = (n-1)H_{n-1} \sim n \ln n$. Also from [134], for any $n \geq 3$ and $c \geq 1$, we have large deviation bound

$$\begin{aligned}
\mathbb{P}[T_n > c\mathbb{E}[T_n]] \leq f(c,n) &= \left(1 + \frac{2c(n-1)H_{n-1}(n-2)^2}{n}\right) \times \left(1 - \frac{2}{n}\right)^{c(n-1)H_{n-1}-2} \\
&\leq \left(\frac{1}{(1-2/n)^2} + \frac{2c(n-1)H_{n-1}(n-2)^2}{n(1-2/n)^2}\right)\left(e^{-\frac{2}{n}}\right)^{c(n-1)H_{n-1}} \\
&= \left(\frac{1}{(1-2/n)^2} - 2cnH_{n-1} + 2cn^2H_{n-1}\right)\exp\left(-2c\frac{n-1}{n}H_{n-1}\right) \\
&\leq \left(\frac{1}{(1-2/3)^2} - 2 \cdot 3 \cdot H_{3-1} + 2cn^2H_{n-1}\right)\exp\left(-2c\frac{n-1}{n}H_{n-1}\right) \\
&= \left(0 + 2cn^2H_{n-1}\right)\exp\left(-2c\frac{n-1}{n}H_{n-1}\right).
\end{aligned}$$

Now observe that $\frac{n-1}{n}H_{n-1} > \ln n + 0.189$ for all $n \geq 8$. Then

$$\begin{aligned}
\mathbb{P}[T_n > c\mathbb{E}[T_n]] &\leq 2cn^2H_{n-1}e^{-2c(\ln n + 0.189)} \\
&= 2H_{n-1}ce^{-0.378c}n^{2-2c}.
\end{aligned}$$

Now we observe that $H_{n-1} < 1.25\ln n$ for all $n \geq 8$ and $ce^{-0.378c} < 1$ for all $c \geq 1$. These inequalities give

$$\mathbb{P}[T_n > c\mathbb{E}[T_n]] \leq 2 \cdot 1.25\ln n \cdot n^{2-2c} = 2.5\ln n \cdot n^{-2\delta}$$

taking $c = 1 + \delta$. □

Corollary 4.2.8. *Define T_n as in Lemma 4.2.7. Then $\mathbb{E}[T_n] < 1.2n\ln n$ and $\mathbb{P}[T_n > 3n\ln n] < \frac{1}{n^2}$.*

Proof. Observe that $\mathbb{E}[T_n] = (n-1)H_{n-1} < 1.2n \ln n$ for all $n \geq 2$. Then $\mathbb{E}[T_n] < 1.2n \ln n$. Also $3n \ln n > 2.5\mathbb{E}[T_n]$, so by the upper tail bound of Lemma 4.2.7, we have

$$\mathbb{P}[T_n > 3n \ln n] \leq \mathbb{P}[T_n > (1+1.5)\mathbb{E}[T_n]] \leq 2.5 \ln n \cdot n^{-3} \leq n^{-2}$$

since $n > 2.5 \ln n$ for all $n \geq 2$. □

We now consider a variation called the **roll call process**, where every agent starts with a `roster` containing a single entry: their unique ID. The agents update with $a.$`roster` $\leftarrow (a.$`roster`$\cup$ $b.$`roster`$)$. Let R_n be the number of interactions to reach the terminal configuration where `a.set` contains all n IDs for every $a \in \mathcal{A}$.

Again, we will consider processes that are stochastically dominated by R_n. We can view the roll call process as the spreading of n epidemics in parallel. Note that the roll call process as described takes exponential states, but it also gives an upper bound for any constant number of epidemics spreading in parallel. We find that asymptotically R_n is 1.5 times larger than T_n. This result about the expected value was shown independently in [**51**, **60**, **110**, **136**]. The results we give here use a different technique which also gives our required large deviation bounds on the time for the roll call process.

Lemma 4.2.9. *Let R_n be the number of interactions for the roll call process to complete. Then $\mathbb{E}[R_n] \sim 1.5n \ln n$. Also $\mathbb{P}[R_n > 3n \ln n] < \frac{1}{n}$.*

Proof. Notice that in the roll call process, each individual ID spreads as a two-way epidemic. Thus we have n epidemic processes happening in parallel; however they are not independent.

We start by observing a lower bound for $\mathbb{E}[R_n]$.

First it is necessary for every agent to have an interaction. Let $\mathbb{E}_1$ be the expected number of interactions for every agent to interact. This is a coupon collector process where we select two agents (coupons) at each step. It follows from a standard coupon collector analysis that $\mathbb{E}_1 \sim \frac{1}{2} n \ln n$.

It is then necessary for the last agent to be picked to spread their ID to the whole population. Let $\mathbb{E}_2$ be the expected number of interactions for this ID to spread to the whole population, starting from this agent's first interaction. This is a standard epidemic process (starting with two infected agents, which is an asymptotically negligible difference), so by Lemma 4.2.7 $\mathbb{E}_2 \sim n \ln n$ interactions.

Then $\mathbb{E}[R_n] \geq \mathbb{E}_1 + \mathbb{E}_2 \sim 1.5n \ln n$. (Note that the entire process may still be incomplete by this point.)

Now we can get an upper tail bound on R_n by considering it as the maximum of n (non independent) epidemic processes. Taking the union bound with Lemma 4.2.7 gives

$$\mathbb{P}[R_n > (1+\delta)\mathbb{E}[T_n]] \leq n \cdot 2.5 \ln n \cdot n^{-2\delta}$$

and then taking $\delta = \frac{1}{2} + u$ for $u > 0$ we have

$$\mathbb{P}[R_n > (1.5+u)\mathbb{E}[T_n]] \leq 2.5 \ln n \cdot n^{-2u}$$

Now since $R_n \geq 0$ we can compute $\mathbb{E}[R_n]$ as

$$\begin{aligned}
\mathbb{E}[R_n] &= \int_{t=0}^{\infty} \mathbb{P}[R_n > t] \; dt \\
&\leq \int_{t=0}^{1.5\mathbb{E}[T_n]} 1 \; dt + \int_{1.5\mathbb{E}[T_n]}^{\infty} \mathbb{P}[R_n > t] \; dt \\
&= 1.5\mathbb{E}[T_n] + \frac{1}{\mathbb{E}[T_n]} \int_0^{\infty} \mathbb{P}[R_n > (1.5+u)\mathbb{E}[T_n]] \; du \\
&\leq 1.5\mathbb{E}[T_n] + \frac{1}{\mathbb{E}[T_n]} \int_0^{\infty} 2.5 \ln n \cdot n^{-2u} \; du \\
&= 1.5\mathbb{E}[T_n] + \frac{2.5}{\mathbb{E}[T_n]} \cdot -\frac{1}{2} n^{-2u} \Big|_0^{\infty} \\
&= 1.5\mathbb{E}[T_n] + \frac{1.25}{\mathbb{E}[T_n]} \sim 1.5n \ln n
\end{aligned}$$

Thus we have $\mathbb{E}[R_n] \sim 1.5n \ln n$.

The observation that $\mathbb{P}[R_n > 3n \ln n] < \frac{1}{n}$ then follows immediately from the same union bound and Corollary 4.2.8. □

We next consider another variation called the **bounded epidemic process**. Here some source agent s has $s.\texttt{level} = 0$, and when agents a, b interact, a updates as $a.\texttt{level} \leftarrow \min(a.\texttt{level}, b.\texttt{level} + 1)$ (and symmetrically for b). Let a be a fixed target agent. We define the time τ_k to be the first time that $a.\texttt{level} \leq k$. Intuitively, τ_k is the time at which a hears information from the source s through a path of length at most k. For example, τ_1 is the time until a and s interact directly, and τ_2 is the time until a interacts with some agent who has already interacted with s.

Lemma 4.2.10. *For any constant $k = O(1)$, $\mathbb{E}[\tau_k] \leq kn^{1/k}$ and $\tau_k \leq n^{1/k} \cdot \frac{c}{2}\ln n$ with high probability $1 - 1/n^c$.*

Proof. For each $i = 0, 1, \dots, k-1$, define L_i to be the first time when the count of agents with $\texttt{level} \leq i$ is at least $n^{i/k}$. Note $L_0 = 0$, since at time 0 (and all future times) we have a single source agent at $\texttt{level} = 0$. We will now show inductively that $\mathbb{E}[L_i] \leq kn^{1/k}$ and $L_i = O(kn^{1/k})$.

After time L_i, we have at least $n^{i/k}$ agents with $\texttt{level} \leq i$. We next need to wait until time L_{i+1}, when at least $n^{(i+1)/k}$ agents have $\texttt{level} \leq i+1$. Let $x(t)$ be the number of agents with $\texttt{level} \leq i+1$ at time t, so $x(L_i) \geq n^{(i+1)/k}$. Then the probability x increases in the next interaction is at least

$$p_x = \frac{\#(\texttt{level} \leq i) \cdot \#(\texttt{level} > i+1)}{\binom{n}{2}} \geq \frac{2n^{i/k}(n-x)}{n^2} \geq \frac{n^{i/k}}{n},$$

since $x \leq n^{i+1}/k \leq n^{(k-1)/k} < \frac{n}{2}$. Now the number of interactions $T = n(L_{i+1} - L_i)$ until time L_{i+1} is bounded by a sum of independent geometric random variables with probability p_x, as x ranges from $n^{i/k}$ to $n^{(i+1)/k}$. This gives

$$\mathbb{E}[T] \leq \sum_{x=n^{i/k}}^{n^{(i+1)/k}} \frac{n}{n^{i/k}} \leq n^{(i+1)/k} \cdot \frac{n}{n^{i/k}} = n^{1+1/k}.$$

Moving from interactions to parallel time, we get $\mathbb{E}[L_{i+1} - L_i] \leq n^{1/k}$, so $\mathbb{E}[L_{i+1}] \leq (i+1)n^{1/k}$ as desired.

We can also get a high probability bound on T, using the Chernoff bound variant for independent geometric random variables from [115], which will show that $L_i = O(kn^{1/k})$ with high probability. This will end up being negligible compared to the $O(\log n \cdot n^{1/k})$ high probability bound we need for the last step, so we omit the details here.

After time L_{k-1}, we have a count $n^{(k-1)/k}$ agents with $\texttt{level} \leq k-1$, and now must wait for the target agent a to meet one of these agents, which will ensure $a.\texttt{level} \leq k$. The number T of required interactions is a single geometric random variable with probability $p \geq \frac{n^{(k-1)/k} \cdot 1}{\binom{n}{2}} \geq \frac{2}{n^{1+1/k}}$. Thus $\mathbb{E}[T] \leq \frac{1}{2}n^{1+1/k}$, giving less than $n^{1/k}$ time, so $\mathbb{E}[\tau_k] \leq kn^{1/k}$. For the high probability bound,

$$\mathbb{P}[T \geq \frac{c}{2}\ln n n^{1+1/k}] \leq (1-p)^{c \ln n \cdot n^{1+1/k}} \leq \exp\left(-\frac{2}{n^{1+1/k}} \cdot \frac{c}{2} \ln n \cdot n^{1+1/k}\right) = n^{-c}.$$

Since this bound is asymptotically larger than the high probability bounds for the inductive argument on earlier levels, we have $\tau_k \leq \frac{c}{2}n^{1/k} \ln n$ with high probability $1 - 1/n^c$. □

The following lemma is similar to Lemma 4.2.10 but for $k = \Theta(\log n)$ rather than $k = O(1)$.

Lemma 4.2.11. *For $k = 3\log_2 n$, $\tau_k \leq 3\ln n$, in expectation and with probability* $1 - O(1/n^2)$.

Proof. Consider the standard epidemic process starting at the source s. By Corollary 4.2.8, it takes at most $3\ln n$ time to complete with high probability $1 - O(1/n^2)$. We consider this epidemic process as generating a random tree in the population as follows. The source s is the root, labeled 1. Then each other agent is a vertex whose parent is the agent who infected them. Furthermore, we can label each vertex by the order in which that agent was infected. This process will exactly create a uniform random recursive tree, as discussed in [**91**]. By Theorem 3 in [**90**], this tree has expected height $\mathbb{E}[H] = e\ln n$, and there is a high probability bound $\mathbb{P}[|H - E[H]| \geq \eta] \leq e^{-c\eta}$ for some constant c, which implies that $H = O(\log n)$ with high probability $1 - 1/n^a$ for any desired constant a.

We then set k to be this bound on the height H. It follows that when agent a gets infected in the full epidemic process, a also has $\texttt{level} \leq k$, since the whole tree has height $\leq k$.

It is also possible to analyze the depth of agent a in this random tree more directly. In the worst case, a has label n (it is the last to be infected). Then for a vertex with label i, their parent's label is uniform in the range $\{1, \ldots, i-1\}$, since at the time this agent gets infected, who infected them has uniform probability over all currently infected agents. Thus we get a recursive process, where $T_0 \leq n$, and $T_i = \text{Uniform}(\{1, \ldots, T_{i-1} - 1\})$, giving a sequence of random variables. We have $\mathbb{E}[T_i | T_{i-1}] = \frac{1+(T_{i-1}-1)}{2} = \frac{T_{i-1}}{2}$ when $T_{i-1} > 1$, and the depth of a is $d = \min\{i : T_i = 1\}$.

To get a high probability bound on d, we will formally define $T_i = \frac{1}{2}T_{i-1}$ for all $i > d$. This way, the recurrence $\mathbb{E}[T_i | T_{i-1}] = \frac{T_{i-1}}{2}$ holds for all i. Unwrapping the recurrence, we get $\mathbb{E}[T_i] = \frac{1}{2^i}T_0 \leq \frac{n}{2^i}$. Then setting $i = 3\log_2 n$ gives $\mathbb{E}[T_{3\log_2 n}] \leq \frac{n}{n^3} = \frac{1}{n^2}$, and by Markov's Inequality, $\mathbb{P}[T_{3\log_2 n} \geq 1] \leq \frac{1}{n^2}$. Thus choosing $k = 3\log_2 n$ ensures a has depth $\leq k$ in the epidemic tree with probability $1 - 1/n^2$. Then using the union bound with the event the epidemic takes longer than $3\ln n$ time, we have $\tau_k \leq 3\ln n$ with probability $1 - O(1/n^2)$. □

4.3. Resetting subprotocol

Propagate-Reset (Protocol 4.2) is used as a subroutine in both of our protocols Optimal-Silent-SSR (Sec. 4.4) and Sublinear-Time-SSR (Sec. 4.5). Intuitively, it provides a way for agents (upon detecting an error that indicates the starting configuration was "illegal" in some way) to "reset" quickly, after which they may be analyzed as though they began from the reset state. For that, the protocol RESET has to be defined for use by PROPAGATE-RESET. We assume that RESET changes the `role` variable to something different from Resetting. Crucially, after the reset, agents have no information about whether a reset has happened and do not attempt any synchronization to ensure they only reset once, lest the adversary simply sets every agent to believe it has already reset, preventing the necessary reset from ever occurring.[5]

We now define some terms used in the analysis of PROPAGATE-RESET, and their intuition:

If $a.\mathtt{role} \neq \mathsf{Resetting}$, then we say a is **computing** (it is executing the outside protocol). Otherwise, for $a.\mathtt{role} = \mathsf{Resetting}$, we use three terms. If $a.\mathtt{resetcount} = R_{\max}$, we say a is **triggered** (it has just detected an error and initiated this global reset). If $a.\mathtt{resetcount} > 0$ we say a is **propagating** (intuitively this property of positivity spreads by epidemic to restart the whole population; we also consider triggered agents to be propagating). If $a.\mathtt{resetcount} = 0$, we say a is **dormant** (it is waiting for a delay to allow the entire population to become dormant before they start waking up, this prevents an agent from waking up multiple times during one reset).

Likewise, we will refer to a configuration as **fully** / **partially** propagating (resp. dormant, computing, triggered) if all / some agents are propagating (resp. dormant, computing, triggered).

A configuration C is **awakening** if it is the first partially computing configuration reachable from a fully dormant configuration. Protocols that use PROPAGATE-RESET will start their analysis by reasoning about an awakening configuration, which formalizes the idea of having gone through a "clean reset". In an awakening configuration, all agents are dormant except one agent who has just executed RESET. Computing agents will awaken dormant agents by epidemic, so within $O(\log n)$ time, all agents will have executed RESET once and then be back to executing the main algorithm.

[5] This is unlike in standard population protocol techniques in which "phase information" is carried in agents indicating whether they are encountering an agent "before" or "after" a new phase starts.

Protocol 4.2 Propagate-Reset(a,b), for Resetting agent a interacting with agent b.
Fields: If role $=$ Resetting, resetcount $\in \{0, 1, \ldots, R_{\max}\}$ and when resetcount $= 0$ an additional field delaytimer $\in \{0, 1, \ldots, D_{\max}\}$.

```
1: if $a$.resetcount $> 0$ and $b$.role $\neq$ Resetting then          ▷ bring $b$ into Resetting role
2:     $b \leftarrow$ Resetting,   $b$.resetcount $\leftarrow 0$,   $b$.delaytimer $\leftarrow D_{\max}$
3: if $b$.role $=$ Resetting then                                      ▷ change resetcount
4:     $a$.resetcount, $b$.resetcount $\leftarrow \max(a$.resetcount $- 1, b$.resetcount $- 1, 0)$
5: for $i \in \{a, b\}$ with $i$.role $=$ Resetting and $i$.resetcount $= 0$ do     ▷ dormant agents
6:     if $i$.resetcount just became $0$ then                          ▷ initialize delaytimer
7:         $i$.delaytimer $\leftarrow D_{\max}$
8:     else
9:         $i$.delaytimer $\leftarrow i$.delaytimer $- 1$
10:    if $i$.delaytimer $= 0$ or $b$.role $\neq$ Resetting then        ▷ awaken by epidemic
11:        execute Reset$(i)$                      ▷ Reset subroutine provided by main protocol
```

We require $R_{\max} = \Omega(\log n)$, and for our protocol will choose the concrete value $R_{\max} = 60 \ln n$. We also require $D_{\max} = \Omega(R_{\max})$. For our $O(\log n)$ time protocol Sublinear-Time-SSR, we have $D_{\max} = \Theta(\log n)$. In Optimal-Silent-SSR, we set $D_{\max} = \Theta(n)$, to give enough time for the dormant agents to do a slow leader election so they finish reset with a unique leader.

Propagate-Reset begins by some agent becoming triggered (resetcount $= R_{\max}$). Although introduced for a different purpose, Propagate-Reset is essentially equivalent to a subprotocol used in [9], so we adopt their time analysis to prove it completes in $O(\log n)$ time. Briefly, from a partially triggered configuration, the propagating condition (resetcount > 0) spreads by epidemic (in $O(\log n)$ time) (Lemma 4.3.2). Once the configuration is fully propagating, it becomes fully dormant in $O(\log n)$ time (Lemma 4.3.3). From the fully dormant configuration, we reach an awakening configuration within $O(\log n)$ time when the first agent executes Reset (Theorem 4.3.4). Then the instruction to execute Reset spreads by epidemic (in $O(\log n)$ time).

4.3.1. Proofs for Propagate-Reset.

We first observe (by lines 2 and 4 of Propagate-Reset) that we can analyze the resetcount field a using the definition from [160] of a **propagating variable** (that updates as $a, b \leftarrow \max(a - 1, b - 1, 0)$):

Observation 4.3.1. *If we define the* resetcount *field for all agents by letting* a.resetcount $= 0$ *for any computing agent* a *(*a.role $\neq$ Resetting*), then in any interaction between* $a, b \in \mathcal{A}$*, their* resetcount *fields both become* $\max(a$.resetcount $- 1, b$.resetcount $- 1, 0)$.

Lemma 4.3.2. *Using the specific value $R_{\max} = 60 \ln n$, starting from a partially triggered configuration, we reach a fully propagating configuration after at most $4 \ln n$ time with probability at least $1 - O(1/n)$.*

PROOF. Noting that `resetcount` is a propagating variable [160], we can use the same proof as [160, Corollary 8]. The constant $R_{\max} = 60 \ln n$ matches the constant used in their result. □

Although we set $R_{\max} = 60 \ln n$, the following lemma holds for arbitrary positive $R_{\max}$.

Lemma 4.3.3. *Let $R_{\max} \in \mathbb{N}^+$ and $D_{\max} = \Omega(\log n + R_{\max})$. Starting from a fully propagating configuration, we reach a fully dormant configuration after $O(\log n + R_{\max})$ time with high probability $1 - O(1/n)$.*

An equivalent process was analyzed in [9], and this proof follows Lemma 1 from [9].

PROOF. We first assume no agents are computing, so no further agents will become triggered, and the `resetcount` field will only change as noted in Observation 4.3.1. We define a local potential $\Phi_t(a) = 3^{a.\texttt{resetcount}}$ for agent a in C_t, except $\Phi_t(a) = 0$ if $a.\texttt{resetcount} = 0$. We then define a global potential $\Phi_t = \sum_{a \in \mathcal{A}} \Phi_t(a)$. We assume as a worst case that every agent starts with `resetcount` $= R_{\max}$, so $\Phi_0 \leq n3^{R_{\max}}$. The goal will now be to show this global potential drops quickly to 0 (corresponding to a fully dormant configuration). We will show that $\Phi_t = 0$ with high probability $1 - O(1/n)$, where $t = O(n(\log n + R_{\max}))$.

Now from Observation 4.3.1, if agents a and b interact in the t^{th} interaction, we can further observe that

$$\Phi_{t+1}(a) + \Phi_{t+1}(b) \leq 2/3 \cdot (\Phi_t(a) + \Phi_t(b))$$

(with equality when $\Phi_t(a) = 0$ or $\Phi_t(b) = 0$). We will thus have the change in global potential

$$\Phi_{t+1} - \Phi_t \leq -1/3 \cdot (\Phi_t(a) + \Phi_t(b))$$

Then conditioning on the configuration C_t, we can compute

$$\begin{aligned}\mathbb{E}[\Phi_{t+1} - \Phi_t | C_t] &\leq \sum_{\{a,b\} \in \mathcal{A}} \mathbb{P}[a, b \text{ interact in } t^{\text{th}} \text{ interaction}] \cdot (-1/3) \cdot (\Phi_t(a) + \Phi_t(b)) \\ &= \frac{1}{\binom{n}{2}} \cdot (-1/3) \sum_{a \in \mathcal{A}} (n-1)\Phi_t(a) = -\frac{2}{3n}\Phi_t\end{aligned}$$

and thus $\mathbb{E}[\Phi_{t+1}|C_t] \leq (1-\frac{2}{3n})\Phi_t$, so $\mathbb{E}[\Phi_{t+1}] = \mathbb{E}[\mathbb{E}[\Phi_{t+1}|C_t]] \leq (1-\frac{2}{3n})\mathbb{E}[\Phi_t]$. Then by induction we have

$$\mathbb{E}[\Phi_t] \leq \left(1-\frac{2}{3n}\right)^t \Phi_0 \leq \exp\left(-\frac{2}{3n}t\right) n3^{R_{\max}}$$

Thus we have $\mathbb{E}[\Phi_t] \leq \frac{1}{n}$ for $t = \frac{3}{2}n\ln\bigl(n^2 \cdot 3^{R_{\max}}\bigr) = \frac{3}{2}n(2\ln n + R_{\max} \cdot \ln 3) = O(n(\log n + R_{\max}))$, so by Markov's inequality we have $\mathbb{P}[\Phi_t = 0] \geq 1 - \frac{1}{n}$.

We assumed that no agents are computing during these t interactions. For the first agent to become computing, `delaytimer` must hit 0 starting from $D_{\max}$ (after that agent has become dormant). Choosing $D_{\max} = \Omega(\log n + R_{\max})$, then no agent will have more than $D_{\max}$ interactions during these t interactions, with high probability $1 - O(1/n)$ by standard Chernoff bounds. Thus all agents will become dormant before the first agent becomes computing with high probability. □

We now combine the previous lemmas to describe the behavior of Propagate-Reset when initialized by a triggered agent. Recall that an awakening configuration is the first partially computing configuration reached from a fully dormant configuration, so the first agent has set `delaytimer` $= 0$ and executed Reset.

Theorem 4.3.4. *Let* $R_{\max} = 60\ln n$ *and* $D_{\max} = \Omega(\log n + R_{\max})$*. Starting from a partially-triggered configuration, we reach an awakening configuration in at most* $O(D_{\max})$ *parallel time with probability at least* $1 - O(1/n)$*.*

Proof. By Lemmas 4.3.2 and 4.3.3, we reach a fully dormant configuration after $O(n(\log n + R_{\max}))$ interactions.

Next, after an additional $n\frac{D_{\max}}{2}$ interactions (which each include 2 of the n agents), by Pigeonhole Principle some agent must have participated in $D_{\max}$ interactions. This agent had `delaytimer` at most $D_{\max}$ in the fully dormant configuration, so by lines 9-11 of Propagate-Reset that agent has executed Reset. When the first such agent executes Reset, we reach an awakening configuration. □

Theorem 4.3.4 describes the intended behavior of Propagate-Reset. This will let protocols start their analysis from an awakening state, and show that given this "clean reset" the protocol will then stabilize. The remaining step in the analysis will be to show from any configuration, we quickly become triggered and enter Propagate-Reset (or simply stabilize). It is possible that this initial configuration includes some agents in arbitrary Resetting states. The following Corollary will let us only have to reason about starting from a fully computing configuration (so we can essentially assume WLOG that our initial configuration is fully computing, since we will quickly leave the Resetting states).

Corollary 4.3.5. *Starting from any configuration, we reach either an awakening configuration or a fully computing configuration after $O(\log n + D_{\max})$ parallel time with probability at least $1 - O(1/n)$.*

Proof. If at any point we enter a partially-triggered configuration, then the result follows from Theorem 4.3.4. Now we will show that if we do not enter a partially-triggered configuration, we must reach some fully computing configuration.

We consider a non-fully-computing configuration that is also not triggered (some agents are in Resetting state, but none have `resetcount` $= R_{\max}$). In this case, following the proof of Lemma 4.3.3, every agent will become dormant (or computing) after $O(n(\log n + R_{\max}))$ interactions with probability $1 - O(1/n)$. Then as in the proof of Theorem 4.3.4, some agent will become computing after $O(nD_{\max})$ interactions. Once some agent is computing, the process for the whole population to become computing is a superepidemic, so by Corollary 4.2.8, this takes at most $3n \ln n$ with high probability $1 - O(1/n^2)$. □

4.4. Linear-time, linear-state, silent protocol

In this section, we present a silent self-stabilizing ranking protocol, Optimal-Silent-SSR, which achieves asymptotically optimal $O(n)$ time and state complexity. Like Silent-n-state-SSR, there will be a unique stable and silent configuration where every agent has a unique rank, but now a rank collision will trigger our Propagate-Reset, causing the entire population to reset. The key idea behind Optimal-Silent-SSR is to add a large delay $D_{\max} = \Theta(n)$ in the Propagate-Reset, which will ensure that the entire population is dormant for long enough to do a simple slow leader election via $L, L \to L, F$, where all agents set themselves to L upon entering the Resetting role. Thus after the population has undergone a reset, we have a unique leader with high probability. After this reset, we do a linear-time leader-driven ranking, where the ranks correspond to nodes in a full binary tree rooted at the leader. In this ranking algorithm, each agent that has been assigned a rank (starting with the leader) assigns ranks directly to 0, 1, or 2 other agents (depending on its number of children in the tree).

In more detail, each agent can be classified into three roles: Settled, Unsettled, and Resetting. A Settled agent has the field `rank` $\in \{1, 2, ..., n\}$. On the other hand, an Unsettled agent has no rank, and it waits for the assignment of a rank from Settled agents.

We use the subprotocol Propagate-Reset described in Section 4.3 to reset each agent when detecting errors. For Optimal-Silent-SSR, the resetting process is triggered under two different situations. 1) Two Settled agents have an identical rank. The rank conflict can be detected when the two agents interact. 2) An Unsettled agent does not get its rank after $\Theta(n)$ interactions.

During the dormant phase of Propagate-Reset, lasting for $\Theta(n)$ time in this protocol, we do slow leader election via $L, L \rightarrow L, F$. Upon awakening (calling Reset), the (likely unique) leader L is Settled with `rank` $= 1$ and followers F are Unsettled. Thus, after resetting, with high probability there will be exactly one Settled agent with `rank` $= 1$, and all the other agents are Unsettled. The Settled agent will act as a leader to assign ranks to all Unsettled agents in the following way. At this point the protocol executes an initialized ranking algorithm, similar to others in the renaming literature [7, 8]. Intuitively, a full binary tree forms within the population. Each Settled agent recruits at most two Unsettled agents, assigning them ranks based on its own to guarantee uniqueness. The children of rank i are $2i$ and $2i + 1$; in other words if an agent's rank has binary expansion s, its childrens' ranks have binary expansions $s0$ and $s1$. Since each agent knows the exact population size, each knows whether its rank corresponds to a node with 0, 1, or 2 children in the full binary tree with n nodes. See Figure 4.1 for an example. This process clearly terminates when all agents are recruited and become settled into different ranks.

Optimal-Silent-SSR takes linear time by the following high-level argument: If there is a rank collision, this is detected in $O(n)$ time. If any agent remains Unsettled without a rank, this is detected via counting up to `errorcount` in $O(n)$ time. Either of these triggers a call to Propagate-Reset. Upon exiting and reaching a fully computing configuration, taking $O(n)$ time by Corollary 4.3.5, we have the leader-driven ranking protocol analyzed in Lemma 4.4.1. There is a constant probability the slow leader election fails (i.e., we end up with multiple leaders), but the expected number of times we must repeat this process before getting a unique leader is constant. The fact that this ranking protocol is $O(n)$ time follows by analyzing each level of the binary tree created across the population: each level takes time proportional to the number of nodes in the level, whence the time is proportional to the size of the tree, i.e., $O(n)$.

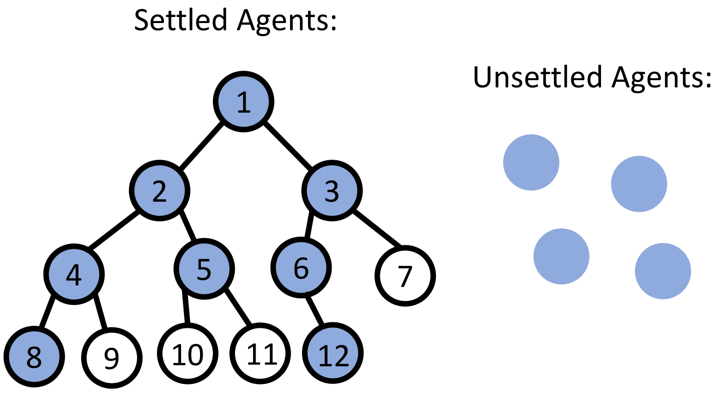


FIGURE 4.1. An example of the rank assignment in OPTIMAL-SILENT-SSR with $n = 12$ agents. There are 8 settled agents on the left (blue circles), with ranks given by the numbers. There are 4 ranks in the binary tree left to be filled by the unsettled agents, when they interact with the settled agents with ranks 3,4 or 5. Lemma 4.4.1 shows this process completes in expected $\Theta(n)$ time.

Protocol 4.3 OPTIMAL-SILENT-SSR, for initiator a interacting with responder b

Fields: $\texttt{role} \in \{\text{Settled}, \text{Unsettled}, \text{Resetting}\}$

If $\texttt{role} = \text{Settled}$, $\texttt{rank} \in \{1, ..., n\}$, $\texttt{children} \in \{0, 1, 2\}$

If $\texttt{role} = \text{Unsettled}$, $\texttt{errorcount} \in \{0, 1, ..., E_{\max} = \Theta(n)\}$

If $\texttt{role} = \text{Resetting}$, $\texttt{leader} \in \{L, F\}$, $\texttt{resetcount} \in \{1, \ldots, R_{\max}\}$, $\texttt{delaytimer} \in \{0, 1, \ldots, D_{\max} = \Theta(n)\}$

```
if $a.\texttt{role} = \text{Resetting}$ or $b.\texttt{role} = \text{Resetting}$ then
    execute PROPAGATE-RESET(a,b)
    if $a.\texttt{leader} = L$ and $b.\texttt{leader} = L$ then
        $b.\texttt{leader} \leftarrow F$
if $a.\texttt{role} = b.\texttt{role} = \text{Settled}$ and $a.\texttt{rank} = b.\texttt{rank}$ then
    $a.\texttt{role}, b.\texttt{role} \leftarrow \text{Resetting}$,   $a.\texttt{resetcount}, b.\texttt{resetcount} \leftarrow R_{\max}$
    $a.\texttt{leader}, b.\texttt{leader} \leftarrow L$
for $(i, j) \in \{(a, b), (b, a)\}$ do
    if $i.\texttt{role} = \text{Settled}$, $j.\texttt{role} = \text{Unsettled}$, $i.\texttt{children} < 2$, and $2 \cdot i.\texttt{rank} + i.\texttt{children} < n$ then
        $j.\texttt{role} \leftarrow \text{Settled}$,   $j.\texttt{children} \leftarrow 0$
        $j.\texttt{rank} \leftarrow 2 \cdot i.\texttt{rank} + i.\texttt{children}$        ▷ $j$ becomes a child node of $i$
        $i.\texttt{children} \leftarrow i.\texttt{children} + 1$
for $i \in \{a, b\}$ do
    if $i.\texttt{role} = \text{Unsettled}$ then
        $i.\texttt{errorcount} \leftarrow \max(i.\texttt{errorcount} - 1, 0)$
        if $i.\texttt{errorcount} = 0$ then
            $a.\texttt{role}, b.\texttt{role} \leftarrow \text{Resetting}$,   $a.\texttt{resetcount}, b.\texttt{resetcount} \leftarrow R_{\max}$
            $a.\texttt{leader}, b.\texttt{leader} \leftarrow L$
```

Protocol 4.4 Reset(a) for Optimal-Silent-SSR, for agent a.
(Called in line 11 of Propagate-Reset.)

```
if $a$.leader $= L$ then
    $a$.role $\leftarrow$ Settled,    $a$.rank $\leftarrow 1$,    $a$.children $\leftarrow 0$
if $a$.leader $= F$ then
    $a$.role $\leftarrow$ Unsettled,    $a$.errorcount $\leftarrow E_{\max} = \Theta(n)$
```

4.4.1. Proofs for Optimal-Silent-SSR.

We first consider the **binary-tree rank assignment** process, which is given by lines 8 to 12 of Optimal-Silent-SSR, starting from a configuration with one agent a with $a.\texttt{role} = \mathsf{Settled}, a.\texttt{leader} = L$, $a.\texttt{children} = 0$ and all other agents b with $b.\texttt{role} = \mathsf{Unsettled}$ (we ignore the field errorcount in this analysis). In other words, we are starting from a single leader a, who builds a binary tree, rooted at a that assigns roles to the entire population.

Lemma 4.4.1. *The binary-tree rank assignment process takes expected $O(n)$ time, starting from a single leader and $n-1$ Unsettled agents.*

Proof. We consider the time taken to assign all nodes at level d of the tree, assuming all nodes at level $d-1$ have been assigned, showing that this time is $O(2^d)$ (i.e., proportional to the number 2^d of nodes at level d). Thus, even if we consider the stochastically dominating process in which no agent is assigned at level $> d$ until all agents at level d have been assigned, the time to complete the tree is $O\left(\sum_{d=1}^{\log n} 2^d\right) = O(n)$.

Assume that level $d-1$ is completely assigned, and let i be the number i of nodes still unassigned at level d, with $0 < i \leq 2^d$. We will now estimate the probability that the next interaction assigns a new node at level d. The count of agents at level $d-1$ with $\texttt{children} < 2$ is at least $\frac{i}{2}$. To estimate the number of Unsettled agents, for each of the i unassigned nodes at level d, we consider the eventually subtree rooted at that node. Each such subtree contains

$$\sum_{j=0}^{\log n-d} 2^j = 2 \cdot 2^{\log n-d} - 1 \leq 2^{\log n-d}$$

nodes. Since the root of the subtree has not been assigned yet, none of the nodes in the subtree have been assigned either. Thus the number of Unsettled agents is at least $i \cdot n \cdot 2^{-d}$. The probability of an Unsettled agent meeting an agent at level $d-1$ with `children` < 2 is at least

$$\frac{(i/2)(i \cdot n \cdot 2^{-d})}{\binom{n}{2}} \sim \frac{i^2}{n2^d}.$$

The total number of interactions to assign all nodes at level d is stochastically dominated by a sum $T = \sum_{i=1}^{n} T_i$ of independent geometric random variables, where T_i is a geometric random variable with success probability $\frac{i^2}{n2^d}$. We can then compute

$$\mathbb{E}[T] = \sum_{i=1}^{n} \frac{n2^d}{i^2} \leq n2^d \sum_{i=1}^{\infty} \frac{1}{i^2} = O(n2^d).$$

Since T is counting interactions, the expected parallel time to assign all nodes at level d is $O(2^d)$, as desired. □

Lemma 4.4.2. *An awakening configuration has a single leader with constant probability.*

Proof. We first analyze the simple leader election process executed in line 4 of Optimal-Silent-SSR. The process $L, L \to L, F$ completes in number of interactions $\sum_{i=2}^{n} T_i$, where T_i is a random variable representing the number of interactions to get from i leaders down to $i-1$. This is geometric with probability of success $\frac{\binom{i}{2}}{\binom{n}{2}} = \frac{i(i-1)}{n(n-1)}$. Thus the expected number of interactions is

$$\sum_{i=2}^{n} \frac{n(n-1)}{i(i-1)} = (n-1)n \sum_{i=2}^{n} \frac{1}{i-1} - \frac{1}{i} = n(n-1)\left[1 - \frac{1}{n}\right] \sim n^2,$$

so the expected parallel time is $\sim n$.

For an awakening configuration to have a single leader, we need this leader election process to finish before the first agent sets `delaytimer` $= 0$, which takes $O(n)$ interactions, and will take $O(n)$ parallel time with high probability by standard Chernoff bounds. By Markov's inequality, for any $\alpha > 0$, the probability that the leader election takes longer than αn time is a most $1/\alpha$. Thus we have a constant probability of successfully electing a single leader before the first dormant agent counts up to $D_{\max}$, where the precise constant depends on the choice of $D_{\max} = \Theta(n)$. □

The required proofs for Optimal-Silent-SSR are given in Section 4.4.1 and yield the following main results:

Theorem 4.4.3. Optimal-Silent-SSR *is a silent protocol that solves self-stabilizing ranking with $O(n)$ states and $O(n)$ expected parallel time.*

Proof. By Corollary 4.3.5, we only have to consider initial configurations that are fully computing or awakening. First consider any fully computing configuration, and argue until we either reach a partially triggered configuration or the unique stable configuration. After expected $O(n)$ time, any Unsettled agents have had enough interactions to count down to errorcount $= 0$. By standard Chernoff bounds, this is true after $O(n \log n)$ time with high probability. Thus we have either reached a partially triggered configuration, or have only Settled agents left. Now if this configuration is not the unique silent, stable configuration, there is a rank collision between two Settled agents, which again will be detected in expected $O(n)$ time and $O(n \log n)$ time with high probability. After reaching a partially triggered configuration, by Theorem 4.3.4, we will then be in an awakening configuration after $O(n)$ time with high probability.

It now remains to analyze the process starting from an awakening configuration. Define an epoch to be the sequence of configurations, starting from an awakening configuration, that ends at either the unique stable configuration or another partially triggered configuration. If it ends in the unique stable configuration, we call the epoch successful. We will now show that an epoch is successful with constant probability, the expected time for an epoch is $O(n)$.

For an epoch to be successful, we first require the awakening configuration to have a single leader, which is true with constant probability by Lemma 4.4.2. Then given a single leader in an awakening configuration, we can analyze the process of the leader assigning ranks to the rest of the population with the analysis of the binary-tree rank assignment process in Lemma 4.4.1, which shows that it takes expected $O(n)$ time. By Markov's inequality, it will also take $O(n)$ time with constant probability, which will be before any of the Unsettled agents have had enough interactions to reach errorcount $= 0$ (again by standard Chernoff bounds on the number of interactions for each agent, where the exact probability depends on the initial value $E_{\max} = \Theta(n)$). As a result, with constant probability we finish the rank assignment, and reach the unique stable configuration with all Settled agents, having unique ranks.

The expected $O(n)$ time for an epoch follows immediately from Lemma 4.4.1 and Theorem 4.3.4. Now the number of required epochs is a geometric random variable with constant expectation, so the total expected time is still $O(n)$.

In analyzing the state set, note that each role using $O(n)$ states. The Settled agents have $O(n)$ states for rank, the Unsettled agents have $O(n)$ states for errorcount, and the Resetting agents have $O(R_{\max} + D_{\max}) = O(n)$ states. The total state set is the sum of each of these disjoint roles, giving $O(n)$ total states. □

Corollary 4.4.4. Optimal-Silent-SSR *takes* $O(n \log n)$ *time with high probability.*

Proof. By Theorem 4.4.3, Optimal-Silent-SSR takes expected $O(n)$ time, so by Markov's inequality it takes $O(n)$ time with constant probability. Now consider epochs of fixed length $O(n)$ time, where an epoch is successful if we reach the unique stable configuration by the end of the epoch. If the epoch is not successful, then we can consider the final configuration as the initial configuration, because the protocol is self-stabilizing. Thus the analysis of Theorem 4.4.3 holds for each epoch. The total number of epochs required is then a geometric random variable with constant success probability, which is $O(\log n)$ with high probability ($1 - O(1/n^c)$ for any desired c). Since these epochs were defined to use a fixed amount of $O(n)$ time, the high probability bound is $O(n \log n)$ time. □

4.5. Logarithmic-time protocol

In this section, we show a protocol solving SSR, and thus SSLE, in optimal $O(\log n)$ expected time, using a "quasi-exponential" number of states: $\exp\big(O(n^{\log n} \cdot \log n)\big)$. Observation 4.2.6 shows that to achieve sublinear time, the protocol necessarily must be non-silent: agents change states forever.

4.5.1. Overview. Intuitively, Sublinear-Time-SSR works as follows. Each agent has a field `name`, a bitstring of length $3 \log_2 n$. The n^3 possible values ensure that if all agents pick a `name` randomly, with high probability, there are no collisions. The set of all `name` values in the population is propagated by epidemic in $O(\log n)$ time in a field called `roster` (which has an exponential number of possible values). Agents update their `rank` (a write-only output field) only when their `roster` field has size n; in this case the agent's `rank` is its `name`'s lexicographical order in the set `roster`.

One source of error is that we can start in a configuration with a "**ghost name**": a name that is in the `roster` set of some agent, but that is not the `name` of any agent. If there are no collisions among actual name's, this error is easy to handle: eventually we will have $|\texttt{roster}| > n$, indicating that there is a ghost name, triggering Propagate-Reset.[6]

The main challenge is then to detect name collisions. Sublinear-Time-SSR calls a subroutine Detect-Name-Collision that detects whether two agents have the same name. If so, we call the same subroutine Propagate-Reset used in Optimal-Silent-SSR, now with $D_{\max} = \Theta(\log n)$ rather than $\Theta(n)$ as in Optimal-Silent-SSR. Upon awakening from Propagate-Reset, agents pick a new name randomly. They use their dormant time, while still in role Resetting, but with $\texttt{resetcount} = 0$ while counting delaytimer down to 0, to generate random bits to pick a new random name.

The bulk of the analysis is in devising an $O(\log n)$ time protocol implementing Detect-Name-Collision. The rest of the protocol outside of Detect-Name-Collision is silent: once the protocol stabilizes, no name or rank field changes. Indeed, we can implement a silent protocol on top of this scheme if we are content with $\Theta(n)$ time: Detect-Name-Collision can be implemented with the simple rule that checks whether the name fields of the two interacting agents are equal, i.e., direct collision detection. The challenge, therefore, is in implementing Detect-Name-Collision in sublinear time by **indirectly** detecting collisions, without requiring agents with the same name to meet directly. Any method of doing this will necessarily be non-silent.

The protocol Detect-Name-Collision is parameterized to give a tradeoff between stabilizing time and state complexity. For instance, there is a $O(\sqrt{n})$ time protocol that uses a data structure with kn bits for a parameter k, i.e., 2^{kn} possible values. Of course, all of the schemes use at least exponential states, since the field roster has $\approx n^{3n}$ possible values. However, the faster schemes will use even more states than this, and their analysis is more complex. This is discussed in more detail in Section 4.5.3.

The proofs in Section 4.5.2 shows this top-level protocol works as intended, once we have correct and efficient collision detection from Detect-Name-Collision.

[6] Eventually we will introduce a tree data structure that also has all the names. However, it is necessary to keep a separate set roster of names for the following reason. The set roster is propagated in time $O(\log n)$, whereas in slower variants of our algorithm, the tree takes too long to collect the names. For example, in the $O(\sqrt{n})$ time (and uses less memory) variant, the tree takes time $\Omega(n)$ to populate with all names.

Protocol 4.5 Sublinear-Time-SSR, for agent a interacting with agent b.
Fields: $\texttt{role} \in \{\text{Collecting}, \text{Resetting}\}$, $\texttt{name} \in \{0,1\}^{\leq 3\log_2 n}$
If $\texttt{role} = \text{Collecting}$, $\texttt{rank} \in \{1,\dots,n\}$, $\texttt{roster} \subseteq \{0,1\}^{\leq 3\log_2 n}$, $|\texttt{roster}| \leq n$, other fields from Detect-Name-Collision
If $\texttt{role} = \text{Resetting}$, $\texttt{resetcount} \in \{1,\dots,R_{\max}\}$, $\texttt{delaytimer} \in \{0,1,\dots,D_{\max} = \Theta(\log n)\}$

```
if $a.\texttt{role} = b.\texttt{role} = \text{Collecting}$ then
    if Detect-Name-Collision($a$,$b$) or $|a.\texttt{roster} \cup b.\texttt{roster}| > n$ then
        $a.\texttt{role}, b.\texttt{role} \leftarrow \text{Resetting}$,    $a.\texttt{resetcount}, b.\texttt{resetcount} \leftarrow R_{\max}$
    else
        $a.\texttt{roster}, b.\texttt{roster} \leftarrow a.\texttt{roster} \cup b.\texttt{roster}$
        if $|a.\texttt{roster} \cup b.\texttt{roster}| = n$ then          ▷ do not set rank until all names collected
            for $i \in \{a,b\}$ do
                $i.\texttt{rank} \leftarrow$ lexicographic order of $i.\texttt{name}$ in $\texttt{roster}$
else                                                                      ▷ some agent is Resetting
    execute Propagate-Reset($a$,$b$)
    for $i \in \{a,b\}$ such that $|i.\texttt{resetcount}| > 0$ do
        $i.\texttt{name} \leftarrow \varepsilon$                          ▷ clear names while propagating the reset signal
    for $i \in \{a,b\}$ such that $i.\texttt{resetcount} = 0$ and $|i.\texttt{name}| < 3\log_2 n$ do
        append a random bit to $i.\texttt{name}$                          ▷ can be derandomized, see Section 4.6
```

Protocol 4.6 Reset(a) for Sublinear-Time-SSR, for agent a.
(Called in line 11 of Propagate-Reset.)

```
$\texttt{role} \leftarrow \text{Collecting}$
$\texttt{roster} \leftarrow \{\texttt{name}\}$
```

4.5.2. Proofs for Sublinear-Time-SSR.

We will first prove a series of Lemmas about the behavior of Propagate-Reset, which will show that if Detect-Name-Collision works as intended to detect collisions in $O(T_H)$ time, then Sublinear-Time-SSR will solve self-stabilizing ranking in $O(T_H)$ time.

We call a configuration **non-colliding** if all agents have distinct names ($a.\texttt{name} \neq b.\texttt{name}$ for all $a, b \in \mathcal{A}$) and $|a.\texttt{name}| = 3\log_2 n$ for all $a \in \mathcal{A}$. This ensures any Resetting agents have generated enough random bits to have picked a new name, and all new names are unique. We first reason about awakening configurations, which we show are non-colliding with high probability.

Lemma 4.5.1. *With high probability $1 - O(1/n)$, from a partially triggered configuration, we reach an awakening, non-colliding configuration.*

Proof. By Theorem 4.3.4, we reach an awakening configuration after $O(D_{\max})$ time. Also, for $\Theta(D_{\max})$ time, all agents will be dormant (with $\texttt{resetcount} = 0$ and decrementing $\texttt{delaytimer}$).

By standard Chernoff bounds for appropriate choice of constant $D_{\max} = \Theta(\log n)$, all agents will be dormant for long enough to generate all $3\log_2 n$ bits of a new name, with high probability.

In this case, each agent has a uniform random name out of all $O(n^3)$ possible bit strings. The probability of any arbitrary pair of agents choosing the same name is $O(1/n^3)$, by union bound over all pairs of agents, the probability of name collision is $O(1/n)$. Thus with high probability $1 - O(1/n)$, the awakening configuration we reach is non-colliding. □

Now we can appeal to the roll call process to show that from an awakening, non colliding configuration, we quickly get to unique ranks.

Lemma 4.5.2. *From an awakening, non-colliding configuration, we reach a configuration with unique ranks in $O(\log n)$ time with high probability $1 - O(1/n)$.*

PROOF. The agents update their field `roster` by taking unions, so the process of all agents getting a complete `roster` is exactly the roll call process. By Lemma 4.2.9, this finishes within $3\ln n$ parallel time with high probability $1 - O(1/n)$. Once every agent has all n unique names in their `roster`, they choose unique ranks by the lexicographic order of their name, in line 8 of SUBLINEAR-TIME-SSR. □

Note that if we were using simple direct interactions to detect name collisions, then this unique rank configuration would be silent and stable. For the actual algorithm, we must later prove a safety condition, that DETECT-NAME-COLLISION will never falsely detect a collision in this stable configuration, shown as Lemma 4.5.4.

There are now two different errors that we must quickly detect. One is name collisions, which we will handle in the next section with DETECT-NAME-COLLISION. The other is “ghost names”. We say a configuration is **ghostly** if some `roster` set contains a ghost name, i.e., if

$$\bigcup_{a \in \mathcal{A}} a.\texttt{roster} \not\subseteq \{b.\texttt{name} \mid b \in \mathcal{A}\} \ .$$

If the configuration is non-colliding, then it is easy to detect ghost names, because the full set of names will be larger than n. Thus we can again appeal to the roll call process to show we quickly detect an error in this case.

Lemma 4.5.3. *Starting from a ghostly and non-colliding configuration, we reach a partially triggered configuration after $O(\log n)$ time with high probability $1 - O(1/n)$.*

Proof. The agents update their field `roster` by taking unions, so the process of all agents getting a complete `roster` is exactly the roll call process. By Lemma 4.2.9, this finishes within $3 \ln n$ parallel time with high probability $1 - O(1/n)$. Because the configuration is ghostly and noncolliding, the total number of distinct names in the rosters is $> n$. Once some agent has `roster` $> n$, they will become triggered by line 3 of Sublinear-Time-SSR. □

4.5.3. Fast Collision Detection. In Sublinear-Time-SSR, both Propagate-Reset and filling all agents' `roster` take $O(\log n)$ time, so the time bottleneck is waiting to detect a name collision. If we simply wait for two agents with the same name to meet to detect a collision, this will take $\Theta(n)$ time in the worst case, which would give a $\Theta(n)$ time silent algorithm.

The goal of Detect-Name-Collision is to detect these names collisions in sublinear time. Because of the lower bound of Observation 4.2.6, this protocol must not be silent. Detect-Name-Collision will have to satisfy two conditions. In order to allow $O(\log n)$ time convergence, from any configuration with a name collision, some agent must detect this collision in $O(\log n)$ time to initiate Propagate-Reset. Second, to ensure the eventual ranked configuration is stable, it must satisfy a safety condition where from a configuration with unique names, no agent will ever think there is a name collision.[7]

As a warm-up to the full $O(\log n)$-time protocol of Detect-Name-Collision, consider the following simpler $O(\sqrt{n})$-time protocol. Each agent keeps a dictionary keyed by names of other agents they have encountered in the population. Whenever agents a and b meet, they generate a random shared value `sync` $\in \{1, \ldots, k\}$, which a stores in its dictionary keyed by the name of b, and b stores in its dictionary keyed by the name of a. If the two agents disagree on this `sync` value at the beginning of an interaction, they declare a name collision.

From a configuration with two agents a and a' who share the same name, within $O(\sqrt{n})$ time, some agent b will interact with both a and a' (assume b first interacts with a, then a'). With probability $1 - \frac{1}{k}$, the sync value that b generates with a will disagree with the sync value that a'

[7] The initial configuration could have unique names, but with auxiliary data adversarially planted to mislead agents into believing there is a name collision, triggering a reset. So the actual safety condition is more subtle and involves unique-name configurations reachable only after a reset.

has with b. Thus when b then meets a', it is able to detect a name collision. Also note that from a configuration with unique names, an invariant is maintained that all pairs of agents agree on their corresponding `sync` values, giving the required safety property.

The actual protocol Detect-Name-Collision is a generalization of this idea. The agents now store a more complicated data structure: a tree whose nodes are labelled by names. See Figure 4.2 for an example. The root is labelled by the agent's own name, and every root-to-leaf path is **simply labelled**, meaning that each node on the path contains unique names (it is permitted for the same name to appear on multiple nodes in the tree, but neither of these nodes can be an ancestor of the other). Each edge is labelled by a `sync` value. The intuition is that these paths correspond to **histories**: chains of interactions between agents, where the `sync` values on the edges were generated by the interaction between that pair of agents. For instance, if a has a path $a \xrightarrow{3} b \xrightarrow{5} c \xrightarrow{7} d$, the interpretation is that when a last met b, a and b generated sync value 3, and in that interaction, b told a that when b last met c, b and c generated sync value 5, and in that interaction, c told b that when c last met d, c and d generated sync value 7. In particular, it could be that c and d have interacted again, generating a different sync value than 7, before a and b interact, but b has not heard about that interaction. See Fig. 4.2 for an example showing how this information is built up.

Protocol 4.7 Detect-Name-Collision(a,b) for Sublinear-Time-SSR, for agent a, b.
Fields: `tree`: depth H, root labelled `name`, other nodes have `node.name` $\in$ `roster`. Edges have `edge.sync` $\in \{1, \ldots, S_{\max} = \Theta(n^2)\}$ and `edge.timer` $\in \{0, \ldots, T_H\}$. The parameter $T_H = \Theta(H \cdot n^{1/(H+1)})$ for $H = O(1)$ and $T_H = \Theta(\log n)$ for $H = \Theta(\log n)$ (we need $T_H = \Theta(\tau_{H+1})$ as in Lemmas 4.2.10 and 4.2.11).

```
for $(i, j) \in \{(a, b), (b, a)\}$ do
    for every path $(i.e_1, \ldots, i.e_p)$ in $i$.tree with $i.e_1$.timer, ..., $i.e_p$.timer $> 0$ and last node $v$
    with $v$.name $= j$.name do                         ▷ All of $i$'s histories about $j$ that aren't outdated
        if Check-Path-Consistency$(j, (i.e_1, \ldots, i.e_p))$ = Inconsistent then
            Return True                                                          ▷ collision detected
$x \leftarrow$ chosen uniformly at random from $\{1, \ldots, S_{\max}\}$                   ▷ Choose new sync value
for $(i, j) \in \{(a, b), (b, a)\}$ do                              ▷ Update trees to share new information
    if $i$.tree has node $v$ at depth 1 with $v$.name $= j$.name then
        Remove the subtree rooted at $v$ from $i$.tree
    Add $j$.tree (to depth $H - 1$) as a subtree of $i$.tree via new edge $e$ from the root
    $e$.sync $\leftarrow x$, $e$.timer $\leftarrow T_H$
for $i \in \{a, b\}$ do                                                   ▷ Keep the trees simply labelled
    remove from $i$.tree all subtrees with root labelled with $i$.name
for each edge $e$ in $a$.tree and $b$.tree do
    $e$.timer $\leftarrow$ max($e$.timer $- 1$,0)
Return False                                                                ▷ no collision detected
```

The $O(\sqrt{n})$ time algorithm above can be thought of as a tree of depth 1, where each agent stores only the names and `sync` values of the agents it has directly interacted with. The general algorithm has a tree of depth H, which allows agents to hear about other agents' `sync` values through longer chains of interactions. In line 3 of Detect-Name-Collision, each agent checks any paths ending at the name of the other agent (the additional fields `edge.timer` are a technicality to handle certain adversarial initial conditions, see Lemma 4.5.5). Intuitively, they require the other agent to show information that is logically consistent with this path, formalized in the conditions of Check-Path-Consistency. To detect a name collision between agents a and a', it will now suffice for some agent b to have heard about agent a before meeting a'. With constant probability, the duplicate agent a' will not have any `sync` values that are logically consistent with this path, and b will declare a collision. Allowing longer paths decreases the time it takes for this information to travel between a and a'. Because the paths that spread information in the epidemic process have length at most $O(\log n)$ with high probability (see Lemma 4.2.11), once we take $H = O(\log n)$, in the $O(\log n)$ time it would take for an epidemic starting at a to reach a', some agent will detect a collision in this way.

Protocol 4.8 Check-Path-Consistency(j,P) for Detect-Name-Collision, for agent j verifying path $P = (i.e_1, \dots, i.e_p)$

```
$q \leftarrow \min\{q' \mid \exists(j.e_p, \dots, j.e_{q'})$ in $j$.tree$\}$        ▷ $(j.e_p, \dots, j.e_q)$ is a root-to-leaf path
for edge $j.e \in (j.e_p, \dots, j.e_q)$ do
    if $j.e$.sync $= i.e$.sync then
        Return True
Return Inconsistent
```

4.5.4. Proofs for Detect-Name-Collision.

We first argue the safety condition, that configurations with unique names will not return false collisions. We start by arguing about awakening, non-colliding configuration, where every agent will execute Reset and initialize a `tree` only containing the root with their unique value of `name`. Here every value `node.name` in an agent's tree will refer to the name of some unique agent. We will sometimes abuse notation and use a to refer to both an agent and the unique name held by that agent.

We now show that from this initialized setting, no collisions are ever detected, ie. there are no "false positives". See Figure 4.2 for an example of why collisions are not detected started from an awakening configuration.

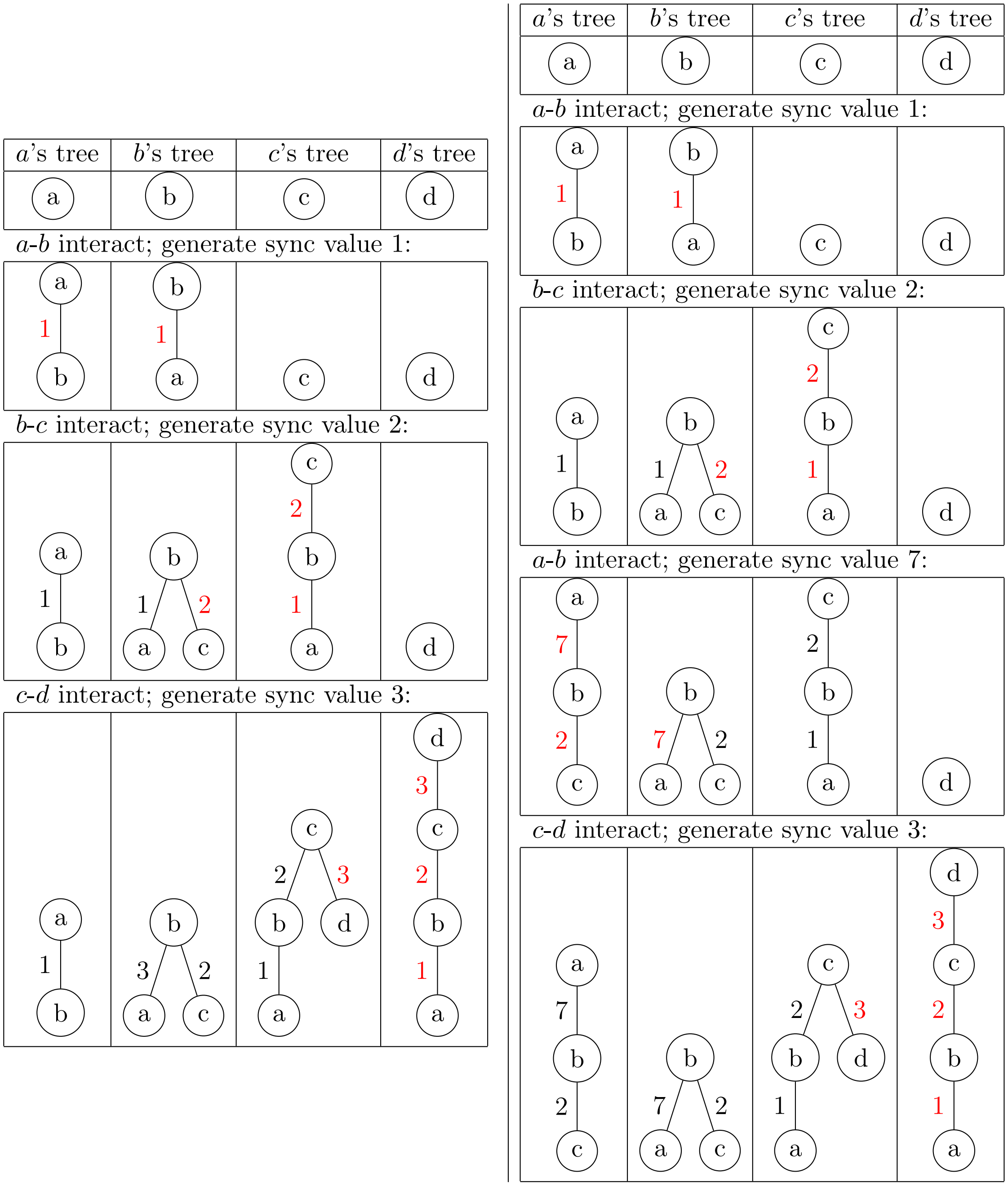

a's tree
b's tree
c's tree
d's tree
a-b interact; generate sync value 1:
b-c interact; generate sync value 2:
c-d interact; generate sync value 3:
a's tree
b's tree
c's tree
d's tree
a-b interact; generate sync value 1:
b-c interact; generate sync value 2:
a-b interact; generate sync value 7:
c-d interact; generate sync value 3:

FIGURE 4.2. Example executions building up trees in agents, starting from a "clean" configuration with singleton trees. Red sync values are those that are newly generated or communicated in the preceding interaction. As an example of how agents check for consistency, when a and d interact, before updating their trees, d checks any paths p that end with a (here there's just one, $d \xrightarrow{3} c \xrightarrow{2} b \xrightarrow{1} a$) against a's corresponding path, which is a's longest reversed suffix of p. In the example on the left, a's reverse suffix is $a \xrightarrow{1} b$, with just a single edge that matches the final sync value in this path p, so CHECK-PATH-CONSISTENCY will return True after checking the first edge. In the example on the right, a's reverse suffix is $a \xrightarrow{7} b \xrightarrow{2} c$. The first edge $a \xrightarrow{7} b$ does not match d's tree, because agents a and b generated the new sync value 7 in a later interaction. However, in that interaction, a added the edge $b \xrightarrow{2} c$, hearing about the b-c interaction with sync value 2 that matches the path in d's tree. Now CHECK-PATH-CONSISTENCY will return True after checking the second edge.

Lemma 4.5.4 (Safety after a correct reset)**.** *An awakening, non-colliding configuration is* ***safe****:* DETECT-NAME-COLLISION *will return* False *in all future reachable configurations.*

PROOF. Observe that because the configuration is non-colliding, every name used to label the nodes in all trees uniquely correspond to one agent in the population. Also, because the configuration is awakening, every agent will at some point start with an empty tree. Thus every edge $e = (i, j)$ in the tree of any agent corresponds to some interaction between agents i and j, who randomly generated the sync value $x = e.\texttt{sync}$.

Now consider any time we run CHECK-PATH-CONSISTENCY(j, P), for a simply labelled path $P = (i.e_1, \dots, i.e_p)$ in the tree of agent i, checking this path against agent j. We prove this returns True, arguing by induction on the length p of the path.

Base case for paths of length $p = 1$: $i.e_1$ corresponds to the last interaction between i and j, where they agreed on a sync value $i.e_1.\texttt{sync} = j.e_1.\texttt{sync}$. Thus j will still have a matching edge $j.e_1$ in its tree, with the same sync value, causing it to return True.

Induction step for paths of length $p \geq 2$: Let $i.e_1 = (i, a)$, where a is the unique agent with the other name on this edge, and $i.e_p = (b, j)$, where b is the unique agent with the other name on this edge. (Note that for $p = 2$, $a = b$.) Consider which of the two pairs (i, a) and (b, j) interacted most recently.

In the first case, assume (i, a) had the more recent interaction than (b, j), at some time T_{ia} before the current time T_{ij}. The path P in $i.\texttt{tree}$ goes from the root to node a, so a has a path $P' = (a.e_2, \dots, a.e_p)$ ending at j that is a length-$(p-1)$ suffix of P, which gets copied to $i.\texttt{tree}$ in this interaction (line 9 in DETECT-NAME-COLLISION). Now consider a hypothetical interaction scheduled instead between a and j at time T_{ia}, when a has the path P' in $a.\texttt{tree}$. By the induction hypothesis, in this hypothetical (a, j) interaction, CHECK-PATH-CONSISTENCY(j, P') would return True, so j has some reverse suffix $(j.e_p, \dots, j.e_q)$ with a matching edge $j.e$ where $j.e.\texttt{sync} = a.e.\texttt{sync} = i.e.\texttt{sync}$. Also, because agents b and j do not interact between times T_{ia} and T_{ij}, this path will not change in $j.\texttt{tree}$. Thus CHECK-PATH-CONSISTENCY(j, P) will also return True at the time T_{ij}.

The second case is mostly symmetric. Assume (b, j) was more recent than (i, a), at some time T_{bj} before the current time T_{ij}. Now consider a hypothetical interaction scheduled instead between i and b at time T_{bj}. i has the prefix $P' = (i.e_1, \dots, i.e_{p-1})$ ending at b, and CHECK-PATH-CONSISTENCY(b, P')

would return True, so b has some reverse suffix $S' = (b.e_{p-1}, \dots, b.e_q)$ with a matching edge $b.e$ where $b.e.\texttt{sync} = i.e.\texttt{sync}$. Also, because agents i and a do not interact between times T_{bj} and T_{ij}, the path P' will not change in $i.\texttt{tree}$. In the actual interaction between b and j at time T, this reverse suffix S' gets copied to $j.\texttt{tree}$, so j has a suffix $S = (j.e_p, j.e_{p-1}, \dots, j.e_q)$, which does not change because j has no further interactions with b. Thus Check-Path-Consistency(j, P) will also return True at the current time, via the edge e in this suffix S. □

In a non-self-stabilizing setting, where we could assume the agents are initialized with empty trees, Lemma 4.5.4 would have been sufficient. However, there is the possibility of adversarial initial conditions that do not have any collisions, but where agents are initialized with inconsistent trees that could falsely detect a collision in a later interaction. It could take up to linear time for the pair of agents with inconsistent data to meet.

In order to circumvent this issue, we add the field `edge.timer` to each edge. From such an adversarial initial condition, once `edge.timer` has counted down to 0 for each edge coming from the initial condition and not a true interaction, then by the check on line 2 of Detect-Name-Collision, no paths from the initial data will ever get checked. Note that paths whose timers have hit 0 can still be used as verification in Check-Path-Consistency, which is essential to ensure correctness, but only if the reverse path still has positive timers.

The following lemma shows that starting from any non-colliding configuration, once all initial timers have run out, either we will have detected a collision, or we will have reached a situation where Lemma 4.5.4 can apply to give safety in all reachable configurations. It reasons about $O(T_H)$ time, where T_H is the parameter giving the maximum value of the `edge.timer`.

Lemma 4.5.5 (Safety from all configurations)**.** *Starting from any non-colliding configuration C, after $O(T_H)$ time, with probability $1 - O(1/n)$, either some agent has become triggered, or we reach a configuration C_0 that is **safe**: Detect-Name-Collision will return* False *in all future reachable configurations.*

Proof. First we argue that after $O(T_H)$ time, with high probability, all timers that correspond to edges from the initial configuration have reached 0. Define E as the set of all edges in the trees of any agents in the initial configuration. For each agent a, define $m(a) = \max\{e.\texttt{timer} : e \in E \cap a.\texttt{tree}\}$, i.e. the largest timer value corresponding to an edge that agent has that came from the initial configuration, so is possibly corrupted initial data. Note that because agents share edges in their interaction, and also all edge timers decrement, these counts m update via $m(a), m(b) \leftarrow \max(m(a)-1, m(b)-1, 0)$. So m is a "propagating" variable of the exact type analyzed in PROPAGATE-RESET. By Lemma 4.3.3, setting $R_{\max} = T_H$, we have $\max_a m(a) = 0$ with high probability in time $O(T_H)$.

Note that before these timers have all run out, it is possible for some agent to return True from DETECT-NAME-COLLISION, and thus become triggered. Also, if the configuration is ghostly, then some agent will become triggered in $O(\log n)$ time with probability $1 - O(1/n)$ by Lemma 4.5.3 (since the configuration is non-colliding), so we can now assume there are no ghost names. But if no agent has become triggered, then after $O(T_H)$ time we will reach a configuration C_0 where every remaining edge that was originally present in C has `edge.timer` $= 0$.

Now we wish to show that C_0 is safe. Consider a coupled configuration C_{empty}, which is identical to C_0, except every tree only contains the root, i.e., is in the state just after executing RESET. Let D' be the configuration where we follow the exact same execution that reached D starting from configuration C_0. We start by arguing that C_{empty} is safe, then use this to argue that C_0 is safe.

Note that C_{empty} is reachable from an awakening and non-colliding configuration (if all agents immediately executed RESET before future interactions). Thus we can apply Lemma 4.5.4 to show DETECT-NAME-COLLISION will return False from every interaction possible from D', i.e., C_{empty} is safe. It remains to argue that C_0 is safe.

Now consider some interaction between agents i and j in configuration D, where they then instantiate CHECK-PATH-CONSISTENCY on some path P in i.`tree`. In this case, every edge in P has `edge.timer` > 0, so these all correspond to interactions that happened in the execution sequence from C_0. Because CHECK-PATH-CONSISTENCY returns True in the coupled configuration D', there must be some matching reverse path in j.`tree`. This same path will also be present in D, so CHECK-PATH-CONSISTENCY also returns True, whence C_0 is safe. □

Lemma 4.5.6 (Fast Detection). *If the configuration is colliding, then* DETECT-NAME-COLLISION *returns* True *for some pair of agents in expected $O(T_H)$ time. When $H = \Theta(\log n)$, it also takes $O(T_H)$ with high probability $1 - O(1/n)$.*

Proof. Let a, a' be two agents with a.name $= a'$.name. Consider the bounded epidemic process with agent a as the source and a' as the target, and time τ_{H+1} as defined in Lemma 4.2.10. At exactly time τ_{H+1}, there will first be a sequence of interacting agents $P = (a, x_1, x_2, \ldots, x_h, a')$ with $h \leq H$, where a interacted with x_1, then x_1 interacted with x_2, etc. Let $b = x_h$ be the agent that interacted with a'. In the case where $H = O(1)$, Lemma 4.2.10 gives $\mathbb{E}[\tau_{H+1}] = O(Hn^{1/(H+1)}) = O(T_H)$, so $\tau_{H+1} = O(Hn^{1/(H+1)})$ with constant probability by Markov's inequality. In the case where $H = \Theta(\log n)$, we use Lemma 4.2.11 to instead say $\tau_{H+1} = O(\log n) = O(T_H)$ with high probability $1 - O(1/n^2)$.

Let b be the agent that interacts with a' on this path. Then there is a sequence of interacting agents $Q = (a, x_1, \ldots, x_{h-1}, b)$, ie. the prefix of P without the final agent a'. There are $h \leq H$ agents before b, which is at most the maximum depth of b.tree. Because of this history of interactions, b.tree will contain a node labelled a at depth h, whose edges have sync values corresponding to the various interactions in Q, i.e., b's tree has a root-to-leaf path $Q_r = (b, x_{h-1}, \ldots, x_1, a)$ that is the reverse of Q.

We will now justify that all edges in Q_r have edge.timer > 0, which will follow because we have only waited $O(T_H)$ time. As the sequence of interacting agents P grows from a to a', we consider the number of interactions from the agent at the current front of the sequence. This will give the number of times the edges on the eventual path Q_r in b.tree have decremented edge.timer. By standard Chernoff bounds, there will be at most $O(T_H)$ of these interactions within $O(T_H)$ parallel time with very high probability $1 - \exp(-\Theta(n))$. Since the initial value of edge.timer $= T_H$, it follows that all edges for path P in b.tree still have positive timers. Thus by line 2 in Detect-Name-Collision, in the interaction between b and a', they will instantiate Check-Path-Consistency with this path Q_r in b.tree.

a' has not interacted with any of the agents $a, x_1, \ldots, x_{h-1}$, so it has not had any interactions where it would learn any sync values of the path. Thus for each edge, the probability of a' having a matching sync value is at most $\frac{1}{S_{\max}} = O\left(\frac{1}{n^2}\right)$. Then taking the union bound over the whole path of length at most $O(\log n)$, a' does not have any matching sync values with high probability $1 - O(\log n/n^2)$. So with high probability, Check-Path-Consistency returns Inconsistent, and Detect-Name-Collision returns True as desired.

In the case where $H = O(1)$, the probability of all required events was at least some constant. Thus, we repeat the argument until Detect-Name-Collision returns True, an expected constant number of times, to conclude that Detect-Name-Collision returns True in expected $O(T_H)$ time. In the case where $H = \Theta(\log n)$, we have stronger high probability $1 - O(1/n)$ for all required events, so we can in addition conclude that Detect-Name-Collision returns True in $O(T_H)$ time with high probability $1 - O(1/n)$. □

Section 4.5.4 proves that Detect-Name-Collision works in $O(T_H)$ time, and also satisfies required safety conditions that ensure there are no "false positives" where collisions are detected from configurations with unique names. These results will let us prove the main theorem about the behavior of Sublinear-Time-SSR:

Theorem 4.5.7. Sublinear-Time-SSR *uses* $\exp\big(O(n^H)\log n\big)$ *states.*

When $H = O(1)$*,* Sublinear-Time-SSR *solves self-stabilizing ranking in expected* $O(H \cdot n^{1/(H+1)})$ *time, and* $O(H \cdot \log n \cdot n^{1/(H+1)})$ *time with high probability* $1 - O(1/n)$*.*

When $H = \Theta(\log n)$*,* Sublinear-Time-SSR *solves self-stabilizing ranking in time* $O(\log n)$*, in expectation and with high probability* $1 - O(1/n)$*.*

Proof. We first count the number of states by counting the number of bits each agent must store. The main memory cost comes from the field `tree`, which has depth H, and each node can have at most n children, so will have $O(n^H)$ nodes. Each node uses $O(\log n)$ bits for `node.name`, and each edge uses $O(\log n)$ bits for `edge.sync` and `edge.timer`. Thus the tree uses $O(n^H \log n)$ bits. Compared to this, all other fields in the protocol are negligible, so it follows that the protocol uses $O(n^H \log n)$ bits, or $\exp\big(O(n^H)\log n\big)$ states.

We now argue that Sublinear-Time-SSR solves self-stabilizing ranking. By Corollary 4.3.5, we only have to consider initial configurations that are fully computing or awakening. We first consider fully computing configurations that are colliding. Here by Lemma 4.5.6, the configuration becomes triggered in $O(T_H)$ time. When $H = O(1)$, this is in expectation (and thus with constant probability by Markov's inequality). When $H = \Theta(\log n)$, this is with high probability $1 - O(1/n)$. We next consider fully computing configurations which are non-colliding. By Lemma 4.5.5, after $O(T_H)$ time we reach either a partially triggered configuration, or a safe configuration. In the case where we reach a safe configuration, there were no ghost names, so every agent has $|\texttt{roster}| = n$

and will have unique ranks based on the lexicographic ordering of `roster`. Thus we have a stable ranked configuration.

We finally argue about awakening configurations. By Lemma 4.5.1, this awakening configuration is non-colliding with high probability $1 - O(1/n)$. In this case, by Lemma 4.5.2, we reach a configuration with unique ranks in $O(\log n)$ time, which is again a stable ranked configuration, since by Lemma 4.5.4 this configuration is also safe.

For the time bounds, when $H = \Theta(\log n)$, we have $T_H = O(\log n)$, and all probability bounds were high probability $1 - O(1/n)$, so we use $O(\log n)$ time in expectation and with high probability.

When $H = O(1)$, we have $T_H = O(H \cdot n^{1/(H+1)})$, and we stabilize in $O(T_H)$ time with constant probability. Then we can consider "epochs" of $O(T_H)$ time, where we declare an epoch successful if it stabilizes. Each epoch has a constant probability of being successful, so the expected number of required epochs is constant, giving an expected $O(H \cdot n^{1/(H+1)})$. With high probability $1 - O(1/n)$, the required number of epochs is $O(\log n)$, giving time $O(H \cdot \log n \cdot n^{1/(H+1)})$ with high probability. □

4.6. Derandomization of Protocols

Note that our model as defined allowed random transitions. However, this was simply for ease of presentation, and the randomness can be simulated through standard "synthetic coin" techniques that exploit the randomness of the scheduler.

We only used randomness in the Reset for Sublinear-Time-SSR, to generate a name uniformly from the set $\{0,1\}^{3\log^2 n}$. We show one approach for how an agent can collect $O(\log n)$ random bits to generate this name.

This approach was inspired by a similar technique from [**158**], but substitutes their "space multiplexing" (splitting the population into two approximately equal-size subpopulations A and F, which is not clear how to implement in a self-stabilizing manner) with "time-multiplexing". On each interaction the agent switches between two roles: "normal algorithm" role (Alg), and "coin flip" role (Flip). When an agent needs a random bit, it waits until it is role Alg and its partner is role Flip. If Alg is the initiator, this represents heads, and if Alg is the responder, this represents tails. This decouples any dependence of the coin flips on each other or on the state of the agent being interacted with. It also incurs an expected slowdown of factor only 1/4 per bit (since each agent requiring a random bit waits expected 4 interactions until it is in role Alg and the other is in role Flip). Thus, by the Chernoff bound, with high probability, the actual slowdown over all $O(\log n)$ bits is at most of factor 1/8.

Thus the agents can become inactive during Reset for the $O(\log n)$ interactions it takes to generate enough random bits to create a new name.

Our constructions here relied on initiator / responder asymmetry as the source of randomness. It would also be possible to avoid this capability and use a symmetric synthetic coin technique as described in [**5**]. In that case, our protocols would be almost entirely symmetric, with one exception: the slow leader election $L, L \rightarrow L, F$ required as part of Reset used in Optimal-Silent-SSR. This line itself could use the symmetric synthetic-coin to be simulated with symmetric transitions, leading to entirely symmetric protocols.

4.7. Conclusion and Perspectives

For the first time, we addressed time-space trade-offs of self-stabilizing leader election and ranking in population protocols over complete graphs. We emphasize that solving these problems, while ensuring such a strong form of fault-tolerance, necessitates linear states and strong nonuniformity (Theorem 4.2.1). Other forms of "strong" fault-tolerance, such as Byzantine-tolerance [**108**] or loosely-stabilizing leader election with **exponential holding time** (a period of time where a unique leader persists after stabilization) [**113**,**160**], similarly necessitate $\Omega(n)$ states. By contrast, a sublinear number of states suffices for many non-fault-tolerant protocols (cf. [**11**]) and weaker forms of tolerance, such as loosely-stabilizing leader election with **polynomial** holding time [**160**] or tolerance to a **constant** number of crashes and transient faults [**75**].[8]

To conclude, we propose several perspectives.

Time/space tradeoffs. It is open to find a subexponential-state sublinear-time self-stabilizing ranking protocol. Observation 4.2.6 states that any sublinear time SSR protocol is not silent. Sublinear-Time-SSR is non-silent because it perpetually passes around information about agents' recent interactions with each other, as a way to detect name collisions without requiring the agents with equal names to meet directly. Even when limiting the tree of interactions to depth 1, this results in an exponential number of states, since each agent must maintain a value to associate

[8] Recall that self-stabilization tolerates any number of transient faults.

to every other agent in the population. Thus, a subexponential-state protocol (if based upon fast collision detection) would somehow need to embed enough information in each agent to enable fast collision detection, while somehow allowing the agent to forget "most" of the information about its interactions. Furthermore, our strategy of using the set `roster` of all names to go from unique names to unique ranks fundamentally requires exponential states.

Ranking vs. leader election. Ranking implies leader election ("automatically"), but the converse does not hold. In the **initialized** case where we can specify an initial state for each agent, it is possible to elect a leader without ranking, using the single transition $\ell, \ell \to \ell, f$ (using too few states for the ranking problem even to be definable). Though any **self-stabilizing** protocol for leader election must use at least n states [**54**] (Theorem 4.2.1 here), it is not the case that any SSLE protocol implicitly solves the ranking problem. (See Observation 4.2.5.) It would be interesting to discover an SSLE algorithm that is more efficient than our examples because it does not also solve ranking.

Initialized ranking. In the other direction, consider the ranking problem in a non-self-stabilizing setting. Without the constraint of self-stabilization, there is no longer the issue of ghost names. Compared to self-stabilization, it may be easier to find an **initialized** ranking protocol that still uses polylogarithmic time, but only polynomial states.

Initialized collision detection. The core difficulty of Sublinear-Time-SSR is **collision detection**: discovering that two agents have been assigned the same name without waiting $\Theta(n)$ time for them to interact directly. It would be interesting to study this problem in the (non-self-stabilizing) setting where an adversary assigns read-only names to each agent, but the read/write memory can be initialized to the same state for each agent. Can a name collision be detected in sublinear time and sub-exponential states?

CHAPTER 5

Message Complexity of Population Protocols

This chapter is joint work with Talley Amir, James Aspnes, David Doty and Mahsa Eftekhari. It was originally published as [15].

5.1. Introduction

The original population protocols model [19] limited the agents to $O(1)$ states, independent of the population size n. This limited the computational power (to only semilinear predicates [20]) and the time efficiency possible in performing fundamental tasks (such as the linear-time lower bound for leader election [88]). Recent work has generalized to $\omega(1)$ states, increasing with n, finding more efficient algorithms for fundamental tasks (e.g., [5, 6, 10, 13, 40, 99, 100, 120, 158]). Is the improved performance a consequence of higher communication throughput or higher local storage capacity?

The original model supposes that agents can view the entirety of the other's local state upon interacting with another agent, which we call an **open** protocol. We introduce a new variant of this model that draws a distinction between the state of the agent and the segment of the state that is externally visible to its interacting partner, called the **message**. This variant generalizes previous work in the context of consensus that examines the particular case of **binary signaling** [18, 138], where the message is limited to a single bit. We study the computational power of population protocols that have $O(1)$ message complexity and varying local state complexity, ranging from $O(1)$ to unbounded.

5.1.1. Motivation. The population protocol framework was conceived to model passively mobile ad hoc sensor networks. In this setting the amount of communication bandwidth can be a tighter constraint than the local computation performed by a sensor. These two constraints—bandwidth efficiency and energy efficiency—are viewed as distinct in the networking literature. In some scenarios it makes more sense to optimize for one or the other, or to strike a balance [76, 125, 163]. The restriction to $O(1)$ messages but $\omega(1)$ internal states is germane when the communication in an interaction is more costly than the accompanying local computation.

Synthetic chemistry is another domain in which population protocols are an appropriate abstract model of computation. They are a subclass of chemical reaction networks, which are known to have similar computational power [63, 151]. Using a physical primitive known as **DNA strand displacement** [171], **every** chemical reaction network with $O(1)$ species (**states** in the language of population protocols) can be theoretically implemented by a set of DNA complexes [152], justifying the use of chemical reactions as an implementable programming language. Using this approach, nontrivial chemical systems have been synthesized in the wet lab, resulting in pure DNA implementations of a chemical oscillator [153] and the "approximate majority" population protocol [22, 66]. Some theoretical [142] and experimental [167] systems are able to assemble unbounded-length heteropolymers such as DNA in an algorithmic way. For such systems, reactions may best be modeled as allowing arbitrarily many states (exponential in the polymer length) but only $O(1)$ messages modeling the smaller "locally visible region" near one or both ends of the polymer.

Finally, our model of $\omega(1)$ internal states and $O(1)$ external messages is a natural mathematical intermediate between the original $O(1)$ states/messages model and the more recent $\omega(1)$ states/messages model. Because population protocols with superconstant states and messages are provably more powerful [62], it is intrinsically interesting to determine how powerful this new intermediate model is.

5.1.2. Our Contribution.

We introduce this new variant of population protocols and show three main results:

We first (Section 5.3) completely resolve the question of the computational power of $O(1)$ messages, with Theorem 5.3.2. In the positive direction, with poly(n) states, we give a simulation of $\Omega(1)$-bit messages (Theorems 5.3.3 and 5.3.5). Corollary 5.3.9 is an asymptotically sharp negative result: $O(1)$-message, $o(n)$-state protocols compute only semilinear predicates.

Secondly (Section 5.5), we focus on time-efficient computation. We develop novel $O(\log^2 n)$-time algorithms for junta election (the key primitive to leader election) and exact population size counting (naturally suited to this model, where $O(n)$ local states and $O(1)$ messages are the minimal power to make this problem solvable). The counting protocol can specialize with fewer states to estimate the size (count $\log n$), and also generalize with more states to count the entire input configuration (so any predicate can be locally computed).

Problem solved	**Pr[error]**	**Time**	**States**	$\lvert M\rvert$	**Leader**
Simulate $s(n)$-state open protocol (Corollary 5.3.4)	0	$O(t_P n^2 \log s(n))$	$O(s(n)^2)$	$O(1)$	Yes
Junta election (Theorem 5.5.1)	> 0	$O(\log^2 n)$	$O(\log^2 n)$	1-bit	No
Compute n (Theorem 5.5.7)	> 0	$O(\log^2 n)$	$O(n \log^2 n)$	$O(1)$	Yes
Compute $\log n$ (Corollary 5.5.8)	> 0	$O(\log^2 n)$	$O(\log n)$	$O(1)$	Yes
Stably compute n (Corollary 5.5.9)	0	$O(\log^2 n)$	$O(n^4 \log^4 n)$	$O(1)$	Yes
Leaderlessly compute n (Corollary 5.5.10)	> 0	$O(\log^2 n)$	$O(n \text{ polylog}(n))$	$O(1)$	No
Leaderlessly compute $\log n$ (Corollary 5.5.10)	> 0	$O(\log^2 n)$	$O(\text{polylog}(n))$	$O(1)$	No
Compute d-input predicate (Corollary 5.5.11)	> 0	$O(d \log^2 n)$	$O(n^d \log^2 n)$	$O(1)$	Yes
TM simulation (Theorem 5.6.5)	0	unbounded	unbounded	1-bit	No

TABLE 5.1. Summary of positive results: Above, the event of "not error" means that the answer is correct **and** the stated time and state bounds hold, unless the error probability is 0, in which case it refers only to the output being correct. In that case, the time and state bounds are in expectation, but still hold with high probability: in all cases, the probability of error is $O(1/n)$. (It should be straightforward to extend to $O(1/n^k)$ for any k, but for simplicity in proof statements we fix $k = 1$.) Note that when the probability of computing the correct output is 1 (i.e. the protocol stabilizes), the Time column denotes time to convergence. State complexities are accurate with high probability. $|M|$ is the number of messages, either a constant larger than 1, or exactly 2 (1-bit). Compute $\log n$ means computing either $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$. In the first row t_P is the expected convergence time for P.

Thirdly (Section 5.6), we explore the extreme limits of the model where message complexity is limited to 1 bit. We construct a 1-bit broadcast primitive, showing it is powerful enough to simulate a Turing Machine with probability 1 correctness using unbounded local memory.

5.1.3. Comparison to existing work and new techniques required. Most protocols using $\omega(1)$ states [**5**, **6**, **10**, **35**, **36**, **42**, **43**, **44**, **53**, **83**, **99**, **100**, **132**, **133**, **158**] crucially use $\omega(1)$-size messages. Key transitions in such protocols involve comparing two integers/ids of size $\omega(1)$ in a single step, which is not possible with $O(1)$-size messages. Sending a superconstant-size message over multiple interactions is not efficient (though it is a trick we employ for unbounded time results such as Theorem 5.3.3), since there is not enough time for the two agents to wait for another interaction (which takes $\Theta(n)$ expected time), nor is there any way to distinguish each other in future interactions. We introduce new techniques that rely on timing of internal counters to get around this limitation.

Protocol 5.2, Junta-Election, is our primary fast leaderless protocol, used to make other leader-driven protocols leaderless. It elects a **junta**, a group of $O(\sqrt{n})$ agents, in $O(\log^2 n)$ time. As with many other existing protocols [40,99,100,158], this is used to drive a junta-driven **phase clock** [21] that allows agents to synchronize in a downstream computation. The cited protocols have agents choose an integer "level" ℓ, propagating by epidemic the maximum level ($\Theta(\log\log n)$ [40, 99, 100] as in our case, or $\Theta(\log n)$ [158]). Agents who chose the maximum level are in the junta. Lacking the ability to communicate the levels in 1-bit messages, we rely on **timing** of internal counters of agents to detect whether a higher level exists: Agents with level ℓ count up to $\approx 4^\ell$, (roughly) telling all other agents to continue counting up, and stop at $\approx 4^\ell$, unless another agent (with high probability with a higher level) tells them to continue counting. The actual details are more involved and require intricate choice of timing and analysis to conclude that all agents with high probability stop at the same counter value.

We push the technique of communication via timing further, showing that only **1-bit** messages suffice to elect a leader, broadcast arbitrary messages, and simulate a Turing Machine.

All of our protocols are **uniform** (requiring no estimate of n), in contrast to several existing $\omega(1)$-state protocols [5,6,10,13,35,42,44,53,100,132,133,158]. Many of our protocols could be simplified greatly by allowing nonuniformity. Briefly, an estimate of $\log n$ within a constant factor allows agents to run a **leaderless phase clock** in which they count up to $c \cdot \log n$ (for some large constant c), which aids in synchronizing agents in rounds $r = 0, 1, 2, \ldots$ based on the number of times they have counted from 0 up to $c \log n$; the lack of such synchronization is a major challenge in devising correct, efficient $O(1)$-message protocols.

5.2. Model

We consider a refinement of the basic model defined in Section 1.2. Now the state of an agent is explicitly divided into an internal component that is not visible to other agents, and an external component that is. The internal component of the state is drawn from the state space I and the external component, or **message**, is drawn from a message space M. The set of states Q is the Cartesian product $I \times M$. The transition function δ is modified to enforce the restriction that an agent in an interaction cannot observe the internal state of the other agent: δ is now a function from $Q \times M \times \{\mathsf{initiator}, \mathsf{responder}\}$ to Q. When an agent in state $q_1 = \langle i_1, m_1 \rangle$ initiates an interaction with an agent in state $q_2 = \langle i_2, m_2 \rangle$, the new states of the agents are given by $q_1' = \langle i_1', m_1' \rangle = \delta(q_1, m_2, \mathsf{initiator})$ and $q_2' = \langle i_2', m_2' \rangle = \delta(q_2, m_1, \mathsf{responder})$.

The set of producible states $Q(n)$ and the set of producible messages $M(n)$ can both depend on n. The function $s : \mathbb{N} \to \mathbb{N}$ defined as $s(n) = |Q(n)|$ is the **state complexity** [1] of a population protocol. The function $n \mapsto |M(n)|$ is the **message complexity**. If $|I| = 1$ and each agent's state is merely defined by its message (the original model [19] and its superconstant state generalization), we say the protocol is **open**, so $|Q(n)| = |M(n)|$ for all n. We will mostly be interested in population protocols with modest state complexity (at most polynomial in n, and often only polylogarithmic in n) and constant message complexity. Given two functions $s, m : \mathbb{N} \to \mathbb{N}$, a **$s(n)$-state, $m(n)$-message population protocol** is one with state complexity s and message complexity m. Note that the complexity bounds we discuss are **worst-case**: $s(n)$ is the most number of states that can be produced in any population of size n under any execution.

We will also place high probability bounds on the state complexity (such as Protocol 5.2 where each agent generates a geometric random variable, which may take on any positive integer value). These are not statements about the set of producible states, so our impossibility results (Theorem 5.3.8) on state and message complexity do not apply.

Some of our protocols utilize an initial leader. Formally, a population protocol is **leader-driven** if its states have a Boolean field `leader` $\in \{\mathsf{L}, \mathsf{F}\}$ (i.e. the state set $Q = \{\mathsf{L}, \mathsf{F}\} \times Q'$), such that in every valid initial configuration, exactly one agent has `leader` $= \mathsf{L}$. For this chapter only, we will slightly modify the definition of **parallel time**, to now say one unit of parallel time is $n/2$ interactions. This scaling ensures in time t, the expected number of interactions that an agent participates in is also t. Our protocols heavily involve counters where agents count all their interactions, so it is convenient for the analysis to have time match up to expected interactions in this way.

[1] This definition of state complexity abstracts away the space used for the local computation of δ as counted in [62]. It can be thought of as a simpler information-theoretic measure of how many different memory configurations agents can be in before and after—but not during—their transitions. Also, the space overhead to compute δ will always use $O(\log |Q|)$ bits, so the asymptotic size $\Theta(\log |Q|)$ of the state space in bits will be unchanged. Because the results of [62] are all asymptotic statements about the number of bits of memory, they can apply directly.

5.3. Computability with unrestricted time

In this section we study $s(n)$-state, $O(1)$-message protocols, when the time is not restricted. Theorem 5.3.2 is our main result in this section, which completely characterizes the power of such protocols in terms of the number of bits required to store the states.

Let $\mathsf{CMPP}(f(n))$ be the set of all predicates stably computed by an $s(n)$-state, $O(1)$-message population protocol, where $s(n) = 2^{O(f(n))}$ (using $O(f(n))$ bits of memory) [2]. Let $\mathsf{SNSPACE}(g(n))$ be the set of all predicates $\phi : \mathbb{N}^d \to \{0,1\}$ decidable by a nondeterministic $O(g(n))$-space-bounded Turing machine, when inputs are given in unary.[3] The results of [**62**] considered a similar complexity class $\mathsf{PMSPACE}(f(n))$ of stably computable predicates using $O(f(n))$ bits of memory and $O(f(n))$ bit messages.[4] Let SL be the set of all **semilinear** predicates (see Section 1.3.1). Their main result is the following characterization:

Theorem 5.3.1 ([**62**]). *Let $f : \mathbb{N} \to \mathbb{N}$. If $f(n) = o(\log\log n)$, then* $\mathsf{PMSPACE}(f(n)) = \mathsf{SL}$*. If $f(n) = \Omega(\log n)$, then* $\mathsf{PMSPACE}(f(n)) = \mathsf{SNSPACE}(n \cdot f(n))$.

Since the memory is expressed in Theorem 5.3.1 as number of **bits** (exponentially smaller than number of **states**), the multiplicative constants hidden in the $O()$ notation become polynomial-factor terms in number of states. Theorem 5.3.2 is a similar dichotomy theorem for $O(1)$-message population protocols, which is sharper in that it holds for **all** values of $f(n)$.

Theorem 5.3.2. *Let $f : \mathbb{N} \to \mathbb{N}$. If $f(n) = o(\log n)$, then* $\mathsf{CMPP}(f(n)) = \mathsf{SL}$*, otherwise* $\mathsf{CMPP}(f(n)) = \mathsf{SNSPACE}(n \cdot f(n))$.

Proof. First note that $2^{O(f(n))}$-state $O(1)$-message population protocols are a special case of the Passively Mobile Machines from [**62**] with space bound $f(n)$ (since we assume the space overhead to compute δ is $O(f(n))$ bits[5]). Thus $\mathsf{CMPP}(f(n)) \subseteq \mathsf{PMSPACE}(f(n))$.

[2] Our model formally requires the local computation of δ to take $O(f(n))$ bits, so here $f(n)$ is the full memory bound on local computation, as in [**62**].

[3] In [**62**] these are called **symmetric** predicates on the assumption that the d counts in $\mathbf{i} \in \mathbb{N}^d$ are presented to the Turing machine as a $\|\mathbf{i}\|$-length string of symbols from an input alphabet Σ with $|\Sigma| = d$, with the same answer on all permutations of the string.

[4] In fact, to obtain their positive result for large space bounds, they do not need fully open protocols. Their simulation of nondeterministic $nf(n)$-space-bounded Turing machines just requires $O(\log n)$ bit messages to exchange unique IDs, even if $f(n) = \omega(\log n)$.

[5] Note even if our definitions were more powerful and the space overhead to compute δ was as large as $O(nf(n))$ bits, we could still make the argument of Theorem 5 of [**62**] to conclude an $O(nf(n))$ nondeterministic Turing Machine can simulate an $2^{O(f(n))}$-state $O(1)$-message population protocols, and Theorem 5.3.2 would still hold.

When $f(n) = \Omega(\log n)$, we will show via Theorem 5.3.5 and Theorem 5.3.3, that $2^{O(f(n))}$-state $O(1)$-message population protocols can simulate a open protocols, with a polynomial state overhead (ie. a constant overhead in $f(n)$ which does not change the definition of the complexity classes). The ability to simulate large messages then implies $\mathsf{PMSPACE}(f(n)) \subseteq \mathsf{CMPP}(f(n))$, and then using Theorem 5.3.1 we have $\mathsf{CMPP}(f(n)) = \mathsf{PMSPACE}(f(n)) = \mathsf{SNSPACE}(n \cdot f(n))$.

Finally, when $f(n) = o(\log n)$, we have $s(n) = 2^{O(f(n))} = o(n)$ and we will show via Corollary 5.3.9 that $\mathsf{CMPP}(f(n)) = \mathsf{SL}$. Note that our necessary condition $s(n) = o(n)$ in Corollary 5.3.9 is actually even sharper than $\log(s(n)) = o(\log n)$. □

5.3.1. Leader-driven $O(s(n)^2)$-state, $O(1)$-message protocols can simulate open $s(n)$-state protocols. In this section we show that $O(s(n)^2)$-state, $O(1)$-message, leader-driven protocols can simulate $s(n)$-state open protocols (whether leader-driven or not). Thus, allowing a leader and ignoring quadratic differences in state complexity (see discussion of quadratic blowup below), there is no difference whatsoever between the computational power of $O(1)$-message protocols and open protocols. Theorem 5.3.3 proves the general case of $m(n)$-message protocols, and Corollary 5.3.4 is the special case of **open** protocols, where $s(n) = m(n)$. The simulation incurs a time slowdown of factor $n^2 \log m(n)$, where n is the population size and $m(n) \leq s(n)$ is the message complexity of the simulated protocol, so it helps port computability results from the open protocol model, but not sublinear time results.

Quadratic state blowup in Theorem 5.3.3. The quadratic state blowup of Theorem 5.3.3 is an artifact of definitional choice, in a sense, owing to each agent a needing to write down the state of another agent b, bit by bit over many interactions, before a can execute the transition δ. However, the model from [62] explicitly counts the space required to store the other agent's message against the total space required, so there is no space blowup in that case.

Intuitively, the construction of Theorem 5.3.3 chooses two agents to "mark" as $\mathsf{initiator}$ and $\mathsf{responder}$, which then successively pass a bit string as they interact, until they have transmitted the full message of size $\log m(n)$ bits. Crucially, starting with a leader allows only one simulated transition to be taking place at a time.

Simulation of a population protocol by another. Formal definitions of simulation in population protocols exist [118, 148] (for the strictly more general model of chemical reaction networks), but such definitions are complex and have to cover many corner cases when applied to arbitrary systems. Since we study just a single simulation construction in Theorem 5.3.3, we avoid a completely formal definition in this paper. Let P, S be population protocols and $\mathbf{c}_P, \mathbf{c}_S$ be configurations of P and S, respectively. Intuitively, we say that S from $\mathbf{c}_S$ **simulates** P from $\mathbf{c}_P$ if, for every execution $\mathcal{E}_P$ of P starting at $\mathbf{c}_P$, there is an execution $\mathcal{E}_S$ of S starting at $\mathbf{c}_S$ that "looks like" $\mathcal{E}_P$, and furthermore every **fair** execution $\mathcal{E}_S$ of S starting at $\mathbf{c}_S$ "looks like" some fair execution $\mathcal{E}_P$ of P starting at $\mathbf{c}_P$.

Here, "looks like" is a tricky concept that can be formalized in a few ways. Intuitively, we imagine that the states of P are projections of the states of S, i.e., each state of S is a pair (p, e), where p is a state of P and e is extra "overhead" information that S requires for the simulation. Furthermore, if we project states from $\mathcal{E}_S$ onto only the first state element p for each agent, and we remove those transitions that appear null from the point of view of P (i.e., the p portion of the state does not change in any agent), and we similarly remove null transitions from $\mathcal{E}_P$, then the resulting executions $\mathcal{E}'_S$ and $\mathcal{E}'_P$ are identical (i.e., go through the exact same sequence of configurations of P).

THEOREM 5.3.3. *For every $s(n)$-state, $m(n)$-message protocol P, there is a leader-driven, $O(s(n) \cdot m(n))$-state, $O(1)$-message*[6] *protocol S that simulates P, and each interaction of P takes expected $O(n^2 \log m(n))$ interactions of S to simulate.*

PROOF. Let M_P be the messages of the simulated protocol P, and $\delta_P : Q_P \times M_P \times \{\mathsf{initiator}, \mathsf{responder}\} \to Q_P$ be its transition function. Intuitively, we will simulate δ_P by marking two agents to exchange bit strings over $O(\log |Q_P|)$ interactions, so each learns the message of the other and can locally compute δ_P.

We now define $Q = I \times M$, the state set of the simulating protocol S. The internal state I contains two fields:

(1) a value $p \in Q_P$ representing the state of this agent in the simulated protocol P

(2) a value $m_o \in M_P$ representing a message of the "other" agent. It is easiest to think of the messages in M_P as binary strings in $\{0, 1\}^*$, because this field will be built up bit-by-bit

[6] The message bound is an absolute constant that does not depend on P. By inspecting the messages as defined in the proof, it is at most $2 \cdot 2 \cdot 3 \cdot 3 = 36$ total messages, though some combinations of fields never appear together, so can be reduced somewhat.

in interactions to learn the other agent's full message. Thus, λ (the empty string) will represent having no information about any other agent's message.

S is leader-driven, so there is a field $\texttt{leader} \in \{L, F\}$ within the message state M. M also contains a field $\texttt{token} \in \{\mathsf{True}, \mathsf{False}\}$, a field $\texttt{mark} \in \{r, i, u\}$ (responder,initiator,unmarked), and a field $\texttt{bit} \in \{0, 1, \texttt{end}\}$.

To represent an initial configuration $\mathbf{c}_P$ of P, we define the initial configuration $\mathbf{c}_S$ (with $\|\mathbf{c}_S\| = \|\mathbf{c}_P\|$) of S as follows. Each agent in S has its field p representing a state in Q_P in the obvious way, λ for p_o, $\texttt{token} = \mathsf{False}$ and $\texttt{mark} = u$ (unmarked). Because S is leader-driven, exactly one agent starts with $\texttt{leader} = L$.

We now describe the transition function δ of S, at a high level. All non-null interactions are between an agent with $\texttt{token} = \mathsf{True}$. The leader L, on its first interaction, immediately becomes a follower F, and the other agent sets $\texttt{token} = \mathsf{True}$.

If the initiator agent has $\texttt{token} = \mathsf{True}$ with mark u, then it marks itself as i and the other agent with r; otherwise it marks itself as r and the other agent with i. The other agent now knows it is a receiver/initiator in the simulated transition. All agents have null transitions now, except for two marked agents. (They could be picked in the opposite order on subsequent transitions and still carry out the following protocol; the initiator-responder distinction in S only matters for the very first transition of S simulating a transition of P.)

Now, the responder and initiator communicate their messages from M_P one bit at a time, storing the other agent's message by appending the received bits to the field p_o, using the field $\texttt{bit}$, sending the value $\texttt{end}$ to indicate their message string has ended. Once both agents have received the other's full message, they can compute δ_P to change their simulated state p. Finally, they both set their mark value to u, and the agent with $\texttt{token} = \mathsf{True}$ sets $\texttt{token} = \mathsf{False}$ and $\texttt{leader} = L$. It is now ready to pass the token to the next agent it sees to simulate another transition of P.

Note that there is one form of asymmetry in the sense that no agent can have the token twice in a row; hence the probabilities of transitions S simulates are different from the original transition probabilities in P. Still, at each new step in the simulation (when an agent who had the token sets $\texttt{leader} = L$ and then passes off the token), every possible transition can be simulated (since the new token recipient can pick the old token holder to mark for the next interaction as well, giving them either r or i). After this nondeterministic choice, the protocol S stably simulates the transition is has committed to by the assignment of `mark`.

Since all possible transitions can be chosen at each step, and the transition will be stably executed (in expected $O(n^2 \log m(n))$ interactions for the i and r mark to meet enough to pass the whole message), S faithfully simulates P. □

We note that execution probabilities are not preserved by this simulation. The agent with the token in the current simulated interaction is half as likely to be chosen in the next simulated interaction as the rest of the population. Section 5.3 focuses on probability-1 results, which are robust to this change. By passing the initial token a larger number of times, the agent-pair probabilities are closer, but not equal, to uniform. We leave open whether there is a simulation as in Theorem 5.3.3 that exactly preserves execution probabilities.

The next corollary applies to **open** protocols, where each agent's message is its full state.

Corollary 5.3.4. *For every $s(n)$-state, open population protocol P, there is a leader-driven, $O(s(n)^2)$-state, $O(1)$-message population protocol S that simulates P, and each interaction of P takes expected $O(n^2 \log s(n))$ interactions of S to simulate.*

It is known that $\Omega(\log n)$-state open protocols have computational power beyond that of $O(1)$-state protocols (limited to semilinear predicates [**20**] and functions [**63**]), and Corollary 5.3.4 grants this same computational power to leader-driven $O(1)$-message protocols. Theorem 5.3.8 in subsection 5.3.4 shows that Corollary 5.3.4 crucially depends on the assumption of an initial leader in the simulating protocol, by demonstrating that **leaderless** $O(1)$-message, $o(n)$-state protocols are no more powerful than $O(1)$-state open protocols.

5.3.2. Leader election can be composed with leader-driven, $s(n)$-state, $O(1)$-message protocols using $O(n^3 \log n)$ state overhead. Leader election is possible in linear time with 1-bit messages by "fratricide": $\ell, \ell \to \ell, f$. A downstream leader-driven protocol P will not work unaltered if composed with this leader election, because the presence of multiple leaders prior to convergence causes incorrect transitions of P. A straightforward fix using $O(n)$ messages involves exact size counting via transitions $\ell_i, \ell_j \to \ell_{i+j}, f_{i+j}$ (requiring $\Omega(n)$ messages) and each transition between agents with respective values n and $i < n$ resets the latter agent to its initial state in P. At the moment the last agent is reset with value n, the protocol at that point faithfully executes a **tail** of an execution of P from $\mathbf{i}$, i.e., an execution starting at a configuration $\mathbf{c}$ reachable from $\mathbf{i}$. Thus if P is correct with probability 1, the composed protocol is also correct with probability 1. Theorem 5.3.5 shows how to achieve a similar "composition by resetting" strategy using only $O(1)$ messages.

Protocol 5.1 STABLY-COMPOSABLE-LEADER-ELECTION(Agent v seeing message m).

P is the downstream protocol with state set Q_P, message set M_P, and transition function $\delta_P : Q_P \times M_P \times \{\mathsf{initiator}, \mathsf{responder}\} \to Q_P$.

P is **leader-driven**, with a field $\mathtt{leader}_P \in \{\mathsf{L}, \mathsf{F}\}$ and possible input states Σ_P.

State set Q of composed protocol S is $Q_S \times Q_P$, where $Q_S = \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \Sigma_P \times M_S$ is the **overhead** of the composition. The fields are named $\mathtt{c}_0 \in \mathbb{N}$, $\mathtt{c}_1 \in \mathbb{N}$, $\mathtt{count} \in \mathbb{N}$, $\mathtt{i}_P \in \Sigma_P$ (representing the input symbol for protocol P), and M_S has fields $\mathtt{role} \in \{\mathsf{L}, \mathsf{F}\}$, $\mathtt{phase} \in \{0, 1\}$ and $\mathtt{signal} \in \{\mathsf{restart}, \mathsf{go}\}$.

initial state of agent v: $\mathtt{c}_0 = \mathtt{c}_1 = \mathtt{count} = 0$, $\mathtt{i}_P = \mathtt{q}_P =$ initial input state of v in P, $\mathtt{role} = \mathsf{L}$, $\mathtt{phase} = 0$, $\mathtt{signal} = \mathsf{restart}$

```
if $v.\mathtt{signal} = \mathsf{go}$ and $m.\mathtt{signal} = \mathsf{go}$ then
    $v.\mathtt{q}_P \leftarrow \delta_P(v.\mathtt{q}_P, m, \mathsf{initiator}$ if $v$ is initiator, else $\mathsf{responder})$
if $v.\mathtt{role} = \mathsf{L}$ and $m.\mathtt{role} = \mathsf{L}$ then
    if $v$ is responder then
        $v.\mathtt{role} \leftarrow \mathsf{F}$                         ▷ fratricide leader election
    else
        reinitialize $v$                              ▷ surviving leader goes back to initial state
else if $v.\mathtt{role} = \mathsf{L}$ and $m.\mathtt{role} = \mathsf{F}$ then           ▷ base-station counting from [25]
    if $b = v.\mathtt{phase} = m.\mathtt{phase}$ then
        $v.\mathtt{count} \leftarrow 0$
        $v.\mathtt{c}_{1-b} \leftarrow v.\mathtt{c}_{1-b} + 1$
        if $v.\mathtt{c}_b = 0$ then
            $v.\mathtt{signal} \leftarrow \mathsf{restart}$              ▷ population estimate $\mathtt{c}_0 + \mathtt{c}_1$ has increased
            $v.\mathtt{q}_P \leftarrow v.\mathtt{i}_P$
            $v.\mathtt{leader}_P \leftarrow \mathsf{L}$
        else if $v.\mathtt{c}_b > 0$ then
            $v.\mathtt{c}_b \leftarrow v.\mathtt{c}_b - 1$
            if $v.\mathtt{c}_b = 0$ then
                $v.\mathtt{signal} \leftarrow \mathsf{go}$              ▷ all counted agents have been restarted
    else if $v.\mathtt{count} \geq 6c_{1-b} \ln c_{1-b} + 1$ then
        $v.\mathtt{count} \leftarrow 0$
        $v.\mathtt{phase} \leftarrow 1 - v.\mathtt{phase}$
    else if $c_{v.\mathtt{phase}} = 0$ then
        $v.\mathtt{count} \leftarrow v.\mathtt{count} + 1$
else if $v.\mathtt{role} = \mathsf{F}$ and $m.\mathtt{role} = \mathsf{L}$ then
    $v.\mathtt{phase} \leftarrow 1 - m.\mathtt{phase}$
    $v.\mathtt{signal} \leftarrow m.\mathtt{signal}$
    if $v.\mathtt{signal} = \mathsf{restart}$ then
        $v.\mathtt{q}_P \leftarrow v.\mathtt{i}_P$
        $v.\mathtt{leader}_P \leftarrow \mathsf{F}$
```

Theorem 5.3.5. *For any leader-driven, $s(n)$-state, $O(1)$-message protocol P, there is a leaderless, $O(s(n)n^3 \log n)$-state, $O(1)$-message protocol S (Protocol 5.1) that, after $O(n \log n)$ expected time, executes a tail of an execution of P.*

Proof sketch. Briefly, we elect a leader in $O(n)$ time by fratricide. The leader counts the followers using the $O(n \log n)$-time counting protocol of [25], which is self-stabilizing under the assumption (proven necessary in [30]) of a "base-station": a leader that is initialized (starts in a pre-determined state), though other agents can start in arbitrary states. That paper does not use the term "self-stabilizing" explicitly, but the assumptions we claimed are stated in their introduction. In our case, the remaining leader resets to the initial state whenever it kills another leader, so after the last such transition, the unique leader is initialized, and other agents are in arbitrary states, exactly the setting handled in [25]. Whenever the leader's population count increases, it tries to reset each follower. Once the leader has counted the whole population and knows n, its count will never change, and it will reset every follower to its initial state of P by direct interaction (using its knowledge of n to ensure all $n-1$ followers are reset), at which point a tail of P executes. □

Proof. First observe that the field `role` updates via the standard "fratricide" leader election. Thus there exists some first time t_1, (with $E(t_1) = O(n)$), where there is a unique agent $a.\texttt{role} = L$. By line 7, a has reinitialized with $c_0 = c_1 = \texttt{count} = \texttt{phase} = 0$.

Now agent a acts as the base-station in the self-stabilizing counting protocol of [25], communicating with the other agents via the field `phase`. In each phase $b \in \{0, 1\}$, a counts the other agents it sees in $\texttt{phase} = b$, moving them into phase $1 - b$, decrementing its counter c_b (if possible) and incrementing c_{1-b}. By the results of [25], the count $c_0 + c_1$ increases monotonically, and stabilizes at a maximum value of $n - 1$ in $O(n \log n)$ expected time.

Let t_2 be the first time $c_0 + c_1 = n - 1$, with $a.\texttt{phase} = b$. Then c_{1-b} just incremented to $n - 1$, and $c_b = 0$ and failed to decrement. The if condition in line 12 was true, so $a.\texttt{signal} = \textsf{restart}$. In all future interactions, c_0 and c_1 accurately count the number of follower agents in each phase, so the condition of line 12 will never be met again.

Now consider the next time t_3 when a changes to phase $1 - b$ and brings the count $c_{1-b} = 0$. By [25] this will also take an expected $O(n \log n)$ time. Then, since time t_2, a has interacted with all agents, who now have $v.\texttt{signal} = \textsf{restart}$. By line 19, $a.\texttt{signal} = \textsf{go}$ for the first time since t_2. Then every agent v who interacts with a will have $v.\texttt{signal} = \textsf{go}$ for all future interactions.

Let $\mathbf{i}$ denote the configuration in protocol P when every agent has their original input state $\mathtt{i}_P$, alongside $a.\mathtt{leader}_P = L$ and $v.\mathtt{leader}_P = F$ for all $v \neq a$. Now observe that when each agent sets $v.\mathtt{signal} \leftarrow \mathtt{go}$ for the last time, they have the same configuration as in $\mathbf{i}$. They only execute transitions in P via line 2 with other agents with $\mathtt{signal} = \mathtt{go}$.

Let t_4 be the next time when every agent has $\mathtt{signal} = \mathtt{go}$, and $\mathbf{c}$ the configuration within P at t_4. Then the only transitions that have made $\mathbf{c}$ different from $\mathbf{i}$ were between two agents with $\mathtt{signal} = \mathtt{go}$, who began from an initialized state. Thus $\mathbf{c}$ is reachable from the correctly initialized configuration $\mathbf{i}$. Finally, all future transitions execute δ_P on both agents, so the composed protocol now exactly implements P, i.e., executes an execution of P from $\mathbf{c}$, i.e., a tail of an execution from $\mathbf{i}$.

Note that we assume that (as is the case in most population protocols, even with $\omega(1)$ states), that only a $O(1)$-size subset Σ of states appear in valid initial configurations, thus $\mathtt{i}_P$ in Protocol 5.1 gives at most a constant-factor overhead to the simulation. If instead agents could start with more states, then the factor would be $|\Sigma_P|$. □

Theorem 5.3.5 depends crucially on using $\geq n$ states, since Theorem 5.3.8 shows leaderless, $O(1)$-message, $o(n)$-state protocols are no more powerful than $O(1)$-state open protocols.

5.3.3. Deterministic Broadcast. The construction used in Protocol 5.1 can be modified to also give the leader the ability to stably broadcast a message to the entire population. After the last restart, the leader agent a counts the entire population by moving them between phases. We can view these phases now as deterministically synchronized rounds (each expected time $O(n \log n)$ [**25**]). Add a field $\mathtt{bit} \in \{0, 1\}$ to the message. The leader a can then communicate a bit string to the population by sending one bit during each round. This lets the entire population stably compute the population size n, by having the leader send n as a bit string in $O(\log n)$ rounds (stabilizing in expected $O(n \log^2 n)$ time). It uses $O(\log n)$ state overhead to store the number of bits it has broadcast, so $O(n^3 \log^2 n)$ states total. We can thus conclude:

Corollary 5.3.6. *There is an $O(n^3 \log^2 n)$-state, $O(1)$-message protocol that stably computes the population size n (storing in every agents state), in expected $O(n \log^2 n)$ time.*

Building on the ideas used in Corollary 5.3.6, we can have the leader assign unique IDs to the agents, for example marking a new unmarked agent in each synchronized round. On top of this deterministic broadcast primitive, we could set up a nondeterministic Turing Machine simulation equivalent to the construction in [**62**]. This gives a direct constructive proof of Theorem 5.3.2, rather than relying on the simulation arguments via Corollary 5.3.4 and Theorem 5.3.5.

5.3.4. Leaderless $o(n)$-state, $O(1)$-message protocols compute only semilinear predicates. Theorem 5.3.8 is broad and does not apply to a particular "mode of computation" (e.g., deciding predicates [**19**, **20**], computing functions [**32**, **63**, **86**], leader election [**40**, **94**]). It does, however, assume a problem-specific notion of **valid** initial configurations.[7] We say a protocol is **additive** if the set of valid initial configurations is closed under addition. This rules out, for instance, protocols with an initial leader. Indeed, Corollary 5.3.9 is false if an initial leader is allowed, by applying Theorem 5.3.3 to let a leader-driven $O(1)$-message protocol simulate any $o(n)$-state open protocol that stably computes a non-semilinear predicate/function.[8]

A lower bound result in [**62**] shows that with an absolute space bound [9] of $o(\log n)$ states, their model is limited to only stably computing the semilinear predicates.[10] The core of their argument bounds the number of reachable memory states.

Theorem 5.3.7 ([**62**]). *Let $s : \mathbb{N} \to \mathbb{N}$ and consider an additive, $s(n)$-state, open population protocol. Then either $s(n) = O(1)$ or $s(n) = \Omega(\log n)$.*

As a corollary, if $s(n) = o(\log n)$, then $s(n)$ is in fact constant, reducing to the original $O(1)$-state model, which can only stably compute semilinear predicates [**20**]. We use a similar proof technique to show an exponentially stronger result in the model of $O(1)$ messages.

Theorem 5.3.8. *Let $s : \mathbb{N} \to \mathbb{N}$ and consider an additive, $s(n)$-state, $O(1)$-message population protocol. Then either $s(n) = O(1)$ or $s(n) = \Omega(n)$.*

Proof sketch. A fixed population $\mathbf{i}_c$ suffices to produce any of the $O(1)$ messages. Consider a population $\mathbf{i}_n$ of size n. If $s(n) \neq O(1)$, then for some state b not producible from $\mathbf{i}_n$, b **is**

[7] For example, for leader election, all agents have the same initial state. For computation of predicates [**19**] or functions [**32**, **63**], all agents represent "input" from a constant alphabet, with possibly an extra leader.

[8] For example, transitions $(i;\ell),(i;\ell) \to (i+1;\ell),(i+1;f)$ and $(j;\ell),(i;f) \to (j;\ell),(j;f)$, which starting from all agents in state $(1,\ell)$, give each agent the value $\lfloor \log n \rfloor$.

[9] The notion of a state bound has a few different meanings. By "absolute", we mean that $s(n)$ is the most number of states producible from any valid initial configuration of size n. Some uniform protocols (those without preprogrammed knowledge of n) have a space bound $s(n)$ that is only probabilistic, so memory usage can (with low probability) grow arbitrarily large in a fixed population; for example, see [**43**, **83**, **99**, **100**] or Protocol 5.2.

[10] Theorem 14 of [**62**] states "$o(\log \log n)$" bits, which implies $o(\log n)$ states, though the converse does not hold. However, inspecting their proof reveals that the result holds up to $\log(n) - 1$ states.

producible by sending some message m to a state a producible from $\mathbf{i}_n$ (though a and m cannot appear **simultaneously** in a configuration reachable from $\mathbf{i}_n$). By combining $\mathbf{i}_n$ with $\mathbf{i}_c$, we have a population of size $n + O(1)$ that can produce b. Thus the number of producible states grows at least linearly with n. □

Proof. If $s(n) = O(1)$ we are done, so assume $s(n)$ grows without bound. Let $M(n)$ (respectively, $S(n)$) be the set of all messages (respectively, states) producible from a valid initial configuration of size n. Note $|M(n)| = O(1)$ and $|S(n)| = s(n)$. It suffices to show that for some constant $\epsilon > 0$ depending on the protocol, there are infinitely many n such that $|S(n)| \geq \epsilon n$.

Let c be the smallest population size n such that $M(n)$ is the set of all messages M. We do not require all messages to be producible **simultaneously**, only that for each $m \in M(n)$, there is a valid initial configuration $\mathbf{i}_m$ such that m can be produced from $\mathbf{i}_m$. Let $\epsilon = 1/c$. Inductively assume for some $n \in \mathbb{N}^+$ that $|S(n)| \geq \epsilon n$. Let $n' = n + c = n + 1/\epsilon$. It suffices to show that $|S(n')| \geq |S(n)| + 1 = \epsilon n + 1 = \epsilon n'$, i.e., a new state not in $S(n)$ is producible from some valid initial configuration of size n'.

There is some state $b \notin S(n)$ producible by an interaction of an agent in state $a \in S(n)$ with some message $m \in M$. Let $\mathbf{i}_n$ be a valid initial configuration of size n from which a is producible, and let $\mathbf{i}_c$ be a valid initial configuration of size c from which m is producible. Define $\mathbf{i}_{n'} = \mathbf{i}_n + \mathbf{i}_c$, which is valid because the protocol is additive. Since a is producible from $\mathbf{i}_n$ and m is producible from $\mathbf{i}_c$, a and m are simultaneously producible from $\mathbf{i}_{n'}$. By interacting the agent in state a with the agent with message m, the state $b \notin S(n)$ is produced. Thus $|S(n')| \geq |S(n)| + 1$. □

Population protocols using $O(1)$ states compute only semilinear predicates [20], resulting in the following corollary. Since we require additivity of valid initial configurations, the corollary applies only to leaderless protocols.

Corollary 5.3.9. *If a leaderless, $o(n)$-state, $O(1)$-message protocol stably computes a predicate ϕ, then ϕ is semilinear.*

Corollary 5.3.9 is asymptotically tight by Observation 5.3.10.

Observation 5.3.10. *For every $\epsilon > 0$, there is a leaderless, $(\epsilon n + O(1))$-state, 6-message protocol stably computing a non-semilinear predicate.*

Proof. Let $c \in \mathbb{N}^+$ and consider the protocol where each agent's internal state is a natural number $k \in \mathbb{N}$, initially 1, representing a number of "balls." Each message $m \in \{0,1,c\} \times \{L,F\}$ represents a number of balls to give away and a leader bit. Each state is a leader bit and a counter k. Agents conduct leader election by fratricide ($L, L \to L, F$). The leaders will collect balls from only the followers, and only in units of c balls. Thus all followers with counter $k \geq c$ display the message $m = (c, F)$, and only interact with a leader. In this interaction, the leader increments k by c and the follower decrements k by c. This guarantees the leader's counter k only actually uses values $\{1 + ic : i \in \mathbb{N}\}$. Finally, followers with counter $1 \leq k < c$ display the message $m = (1, F)$. If two agents with $m = (1, F)$ interact, the initiator gives one ball to the responder (i.e. one increments k, one decrements k).

It is straightforward to show that eventually this protocol will stabilize to a single leader with count "about n": $k = 1 + c\lfloor \frac{n-1}{c} \rfloor$. The sum of counts are clearly preserved. Followers with $k \geq c$ balls must eventually give all units of c balls to the leader, who never decreases its count. While there are still $j \geq c$ total balls among the followers, eventually some follower will collect c balls to give to the leader.

Notice that this protocol can only achieve counter values $k \in \{0, 1, \ldots, c, 1 + c, 1 + 2c, 1 + 3c, \ldots, 1 + c\lfloor \frac{n-1}{c} \rfloor\}$, thus this counter uses $\frac{n}{c} + O(1)$ states. Counting the 6 messages gives $\frac{n}{6c} + O(1)$ states.

Finally, the leader can compute some non-semilinear predicate of its count $k = 1 + c\lfloor \frac{n-1}{c} \rfloor$ (e.g., whether $\lfloor \frac{n-1}{c} \rfloor = 2^j$ is a power of two[11]), and use its message bit to tell the output to the rest of the population. (So a follower seeing message $m = (i, L)$ sets its output bit to i.) □

5.4. Timing Lemmas

The following technical lemmas are about the relationship between agents' local counts and the global number of interactions, and the time it takes to spread information by epidemic. They are used in the runtime analysis of our time efficient protocols.

5.4.1. Clock Drift Lemma.

[11] Our model does not directly count the memory required to compute the transition δ. However, for this argument the Turing Machine would only need $O(1)$ bits of overhead, storing a counter i to represent $1 + ic$, which is a power of two if and only if its binary expansion matches the regex 10^*. Thus this asymptotic tightness on states holds even under a stricter state-complexity definition that counts space requirements of local computation.

Lemma 5.4.1. *Consider some interval of an execution of a population protocol with uniform random scheduling. Let A_{is} be the indicator variable for the event that agent i is one of the two agents that interact in step s of this interval. Let $C_{it} = \sum_{s=1}^{t} A_{is}$ be the cumulative number of interactions involving agent i during the first t steps of the interval. Fix two agents i and j, and let τ be the first time at which $C_{i\tau} + C_{j\tau} = m$. Then $\mathbb{P}\left[\max_{t \leq \tau}(C_{it} - C_{jt}) > b\right] \leq e^{-b^2/2m}$.*

Proof. Because only steps involving at least one of i or j change C_{is} and C_{js}, we can restrict our attention to the sequence of steps $s_1, s_2, \dots$ at which at least one of i or j interacts. Let $X_k = A_{it_k} - A_{jt_k}$; then $\mathbb{E}\left[X_k \mid X_1, \dots, X_{k-1}\right] = 0$ and the X_k form a martingale difference sequence with $|X_k| \leq 1$. We also have that $C_{it_k} - C_{jt_k} = \sum_{\ell=1}^{k} X_\ell$, so $\max_{t \leq \tau}(C_{it} - C_{jt}) > b$ if and only if there is some k with $t_k \leq \tau$ such that $\sum_{\ell=1}^{k} X_\ell > b$. Define the truncated martingale difference sequence $Y_k = X_k$ if $t_k \leq \tau$ and $\sum_{\ell=1}^{k-1} X_\ell \leq b$, and $Y_k = 0$ otherwise. Let $S_k = \sum_{\ell=1}^{k} Y_\ell$.

We have defined S_k so that it tracks $C_{it_k} - C_{jt_k}$ until that quantity reaches $b+1$ or t_k reaches τ, after which S_k does not change. The condition $t_k = \tau$ occurs for some $k \leq m$, so if $C_{it_k} - C_{jt_k}$ reaches $b+1$ before $t_k > \tau$, it must do so for some $k \leq m$, after which S will not change, giving $S_m = S_k$. So $\mathbb{P}\left[\max_{t \leq \tau}(C_{it} - C_{jt}) > b\right] = \mathbb{P}\left[S_m > b\right] \leq e^{-b^2/2m}$, by the Azuma-Hoeffding inequality. □

Corollary 5.4.2. *For any agents i and j, and any m and b, $\mathbb{P}\left[\exists t : C_{it} = m \wedge C_{jt} > m + b\right] < e^{-b^2/(2m+b+1)}$.*

Proof. If $C_{jt} - C_{it} > b$ when $C_{it} = m$, then there is some first time s at which $C_{js} - C_{is} > b$. Because $s \leq t$, $C_{is} \leq C_{it} = m$, and because this is the first time at which $C_{js} - C_{is} > b$, we have $C_{js} = C_{is} + b + 1 \leq m + b + 1$. So s is a time at which $C_{is} + C_{js} \leq 2m + b + 1$ with $C_{js} - C_{is} > b$. Now apply Lemma 5.4.1. □

5.4.2. Drift Fraction Lemma.

Recall that μ units of time is defined as $\frac{n}{2} \cdot \mu$ interactions.

Lemma 5.4.3. *Consider some set S of agents and interval of length $T = \frac{n}{2} \cdot \mu$ interactions. Let $L \subseteq S$ be the subset of S who have less than $\mu - l$ interactions during the interval. Then for $\epsilon_L = 2\sqrt{2\ln(n)/|S|} + \exp\left(-l^2/2\mu\right)$, $\mathbb{P}\left[|L| \leq \epsilon_L |S|\right] \geq 1 - 1/n^2$ and $\mathbb{P}\left[|L| = 0\right] \geq 1 - |S| \exp\left(-l^2/2\mu\right)$.*

Likewise, let $H \subseteq S$ be the subset of S who have more than $\mu + h$ interactions during the interval, where $h \leq \mu$. Then for $\epsilon_H = 2\sqrt{2\ln(n)/|S|} + \exp\left(-h^2/3\mu\right)$, $\mathbb{P}\left[|H| \leq \epsilon_H |S|\right] \geq 1 - 1/n^2$ and $\mathbb{P}\left[|H| = 0\right] \geq 1 - |S| \exp\left(-h^2/3\mu\right)$.

Proof. For each agent v and step t of the interval, let $A_{v,t}$ be the indicator variable for the event that agent v is one of the two agents that interact in step t (with $\mathbb{P}[A_{v,t} = 1] = \frac{2}{n}$). For each agent v, let L_v, H_v to be the indicator variables for the events that agent v participates in fewer than $\mu - l$ interactions and more than $\mu + h$ interactions, respectively. Then $L = \{v \in S : L_v = 1\}$ and $H = \{v \in S : H_v = 1\}$, so $S' = L \cap H$.

Let $C_v = \sum_{t=1}^{T} A_{v,t}$ be the number of interactions in which agent v participates, with $\mathbb{E}[C_v] = T \cdot \frac{2}{n} = \mu$. Since each step is independent, we can apply standard Chernoff bounds on the probability

$$\mathbb{P}[L_v = 1] = \mathbb{P}[C_v < \mu - l] = \mathbb{P}[C_v < \mu(1 - l/\mu)] \le \exp\left(-(l/\mu)^2\mu/2\right) = \exp\left(-\frac{l^2}{2\mu}\right).$$

Likewise, we can get an upper bound on the probability

$$\mathbb{P}[H_v = 1] = \mathbb{P}[C_v > \mu + h] = \mathbb{P}[C_v > \mu(1 + h/\mu)] \le \exp\left(-(h/\mu)^2\mu/3\right) = \exp\left(-\frac{h^2}{3\mu}\right).$$

Then $\mathbb{P}[L = 0] \ge 1 - |S| \exp\left(-\frac{l^2}{2\mu}\right)$ and $\mathbb{P}[H = 0] \ge 1 - |S| \exp\left(-\frac{h^2}{3\mu}\right)$ follow simply from the union bound over all $v \in S$. It is less straightforward to place high probability bounds on $|L|$ and $|H|$, because the indicator variables L_v and H_v are not independent. Intuitively, however, they have negative dependence, since if $L_v = 1$, that agent had a small number of interactions, making other agents more likely to have more. We formalize this intuition by showing that the joint distributions $\{L_v : v \in S\}$ and $\{H_v : v \in S\}$ are each **negatively associated**, which allows Chernoff bounds to be applied [92, Section 3.1].

First we show that for fixed t, the distribution $\{A_{v,t} : v \in [n]\}$ is negatively associated, since precisely two variables will have value 1 and the rest 0. Theorem 2.11 of [116] shows that all permutation distributions (random variables $X_1, \ldots, X_n$ whose values are a random permutation of $x_1, \ldots, x_n$) are negatively associated. The distribution $\{A_{v,t} : v \in [n]\}$ is a special case of a permutation distribution where the variables are a random permutation of $0, \ldots, 0, 1, 1$, and thus is negatively associated. Therefore the entire distribution $\{A_{v,t} : v \in [n], t \in [T]\}$ is negatively associated as the union of independent negatively associated variables (over the independent steps of the uniform scheduler).

Finally, the distribution $\{L_v \mid v \in S\}$ is a collection of monotone functions each defined on a disjoint subset of the distribution $\{A_{v,s} : v \in [n], s \in [T]\}$. This is precisely the property of disjoint monotone aggregation ([92]), which shows that $\{L_v \mid v \in S\}$ is negatively associated. The same argument also holds for $\{H_v : v \in S\}$.

Now we will actually apply Chernoff bounds to the complements $|S \setminus L| = \sum_{v\in S}(1 - L_v)$ and $|S\setminus H| = \sum_{v\in S}(1-H_v)$, which are justified by negative association. Then by linearity of expectation, $\sigma = \mathbb{E}\left[|S \setminus L|\right] \geq (1 - \exp\bigl(-l^2/2\mu\bigr))|S|$.

Now letting $\delta = 2\sqrt{2\ln n/|S|}$ and $\epsilon_L = \delta + \exp\bigl(-l^2/2\mu\bigr)$, we have

$$\begin{aligned}\mathbb{P}\left[|L| > \epsilon_L|S|\right] &= \mathbb{P}\left[|S \setminus L| < (1-\epsilon_L)|S|)\right] \\ &\leq \mathbb{P}\left[|S \setminus L| < (1-\delta)(1-\exp\bigl(-l^2/2\mu\bigr))|S|\right] \\ &\leq \mathbb{P}\left[|S \setminus L| < (1-\delta)\sigma\right],\end{aligned}$$

and we can apply the Chernoff bound to get

$$\mathbb{P}\left[|S \setminus L| < (1-\delta)\sigma)\right] \leq \exp\left(-\delta^2\sigma/2\right) \leq \exp\left(-\delta^2|S|/4\right) \leq 1/n^2,$$

where we also assumed $\sigma \geq |S|/2$ since $\mathbb{P}\left[C_v \geq \mu - l\right] > 1/2$. Thus $\mathbb{P}\left[|L| \leq \epsilon_L|S|\right] \geq 1 - 1/n^2$.

The same argument, for $\epsilon_H = \delta + \exp\bigl(-h^2/3\mu\bigr)$ also shows that $\mathbb{P}\left[|H| \leq \epsilon_H|S|\right] \geq 1 - 1/n^2$. □

5.4.3. Epidemics. Recall an **epidemic** process in a population protocol starts with a single agent **infected** (i) and all others **susceptible** (s), and every encounter between an infected and uninfected agent causes the latter to become infected (i.e., the transition $i, s \to i, i$).

The following lemma generalizes to a susceptible subset of the population, and considers a symmetric middle interval of an epidemic process, to be used in the analysis of Protocol 5.2.

Lemma 5.4.4. *Consider the two way epidemic process starting from a configuration of n agents with $a = \alpha n$ infected agents and $b = \gamma n \geq a$ susceptible agents (and $n - a - b$ agents not participating). Let T be the number of interactions to reach $b+1$ infected agents (with $a-1$ susceptible agents left). Then $T \leq \frac{5}{\gamma}n\ln\left(\frac{\gamma}{\alpha}\right)$ with probability $1 - (\gamma n)^{-2}$.*

Proof. The number of infected agents i will increase monotonically from a to $b+1$. At each stage, when there are i infected agents and $b+a-i$ susceptible agents, there are $i(b+a-i)$ pairs of agents whose interaction increases the number of infected agents, so the next interaction is one of these with probability $p_i = \frac{i(b+a-i)}{\binom{n}{2}}$. Thus $T = \sum_{i=a}^{b} G_i$ where G_i is a geometric random variable with parameter p_i. Then

$$\mu = E(T) = \sum_{i=a}^{b} \frac{1}{p_i} = \binom{n}{2} \sum_{i=a}^{b} \frac{1}{i(b+a-i)} = \frac{n(n-1)}{2(b+a)} \sum_{i=a}^{b} \frac{1}{i} + \frac{1}{b+a-i} = \frac{n(n-1)}{(b+a)}(H_b - H_{a-1})$$

where $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the nth Harmonic number. Then

$$\mu \approx \frac{n}{\alpha+\gamma} \ln\left(\frac{b}{a}\right) = \frac{n}{\alpha+\gamma} \ln\left(\frac{\gamma}{\alpha}\right)$$

and because $\gamma \geq \alpha > 0$ we can bound

$$\frac{n}{2\gamma} \ln\left(\frac{\gamma}{\alpha}\right) \leq \mu \leq \frac{n}{\gamma} \ln\left(\frac{\gamma}{\alpha}\right)$$

Then by Theorem 2.1 of [115], we have the large deviation bound

$$\mathbb{P}\left[T \geq \lambda\mu\right] \leq \exp\left(-p_*\mu(\lambda - 1 - \ln\lambda)\right)$$

where $p_* = \min_i p_i = \frac{ab}{\binom{n}{2}} \approx 2\alpha\gamma$.

Letting $\lambda = 5$, $\lambda - 1 - \ln\lambda > 2$, this gives

$$\mathbb{P}\left[T \geq \frac{5}{\gamma} n \ln\left(\frac{\gamma}{\alpha}\right)\right] \leq \mathbb{P}\left[T \geq 5\mu\right] \leq \exp\left(-2\alpha n \ln\left(\frac{\gamma}{\alpha}\right)\right).$$

Now observe that $\alpha \ln(1/\alpha)$ is increasing for all $\alpha \in (0,1)$. Since $\alpha \geq 1/n$, we can take this minimum value to get

$$\mathbb{P}\left[T \geq \frac{5}{\gamma} n \ln\left(\frac{\gamma}{\alpha}\right)\right] \leq \exp\left(-2\ln(\gamma n)\right) = (\gamma n)^{-2}.$$

□

5.5. Computability with polylogarithmic time complexity

In this section we study $O(1)$-message population protocols when the goal is "fast" computation (polylog(n) time).

5.5.1. High-probability junta election using 1-bit messages.

In this section, we describe a uniform protocol using 1-bit messages that, with high probability, elects a "junta" of $O(\sqrt{n})$ agents in polylogarithmic time. The protocol also lets each agent compute an integer $k \in \mathbb{N}^+$ that is the same for all agents and is one of $\lfloor \log\log n \rfloor$, $\lceil \log\log n \rceil$, or $\lceil \log\log n \rceil + 1$. Thus 2^k is an estimate of $\log n$ within a multiplicative factor 2.

Furthermore, JUNTA-ELECTION is composable, in that we can use the protocol as a black box to initialize other protocols that require either a junta for a phase clock, or an approximation of $\log n$ (e.g. for a leaderless phase clock). Thus, for any nonuniform protocol that requires k-bit messages, we can compose it with our JUNTA-ELECTION protocol and achieve a uniform protocol that uses $(k+1)$-bit messages with an additive time overhead of $O(\log^2 n)$. For example, we can compose the JUNTA-ELECTION protocol with the the leader election protocol of [**99**] using $\frac{1}{2}$-coin flips to convert the $O(\sqrt{n})$-size junta to size 1, i.e., elect a unique leader, in expected $O(\log^2 n)$ time and $O(1)$ messages, or with majority protocols that use $O(1)$ messages for doubling/cancelling phases, synchronized by the junta-driven phase clock [**36**].

Our protocol has a positive probability of failure. It is an open question if there exists an $O(1)$-message protocol that can stably (i.e., with probability 1) approximate $\log n$ or elect a junta of size n^ϵ for some $0 < \epsilon < 1$ in sublinear stabilization time.

Protocol 5.2 JUNTA-ELECTION(Agent v seeing message m)

```
initial state of agent:
geometric ← $\frac{1}{2}$-geometric random variable (used for estimating level)
$v$.level ← $\lceil \log(\texttt{geometric}) \rceil$
$v$.count ← 0
$v$.inJunta ← True

if $v$.count $= d_i$ and $v$.level $\le i$ then
    if $m =$ Go then
        $v$.count ← $v$.count + 1
else
    $v$.count ← $v$.count + 1
if $v$.count $\in G_i$ and $v$.level $\le i$ then
    $v$.message ← Go
else if $v$.count $\in R_i$ and $v$.level $\le i$ then
    $v$.message ← Stop
if $v$.count $\in [G_i \cup R_i]$ and $v$.level $> i$ then
    $v$.message ← Go
if $v$.count $= d_i$ then
    $v$.lognEstimation ← $2^i$
    $v$.inJunta ← $v$.level $\ge i$
```

5.5.1.1. *High-level description of protocol.* The protocol is described formally in Protocol 5.2. Intuitively, it works as follows. Most other leader/junta election protocols generate an **id**, where the agents generating the maximum id are the junta. In our protocol, we also generate an id (called **level**), but $O(1)$ messages prevent direct communication of levels, so we employ a timing-based strategy for agents to learn the maximum level. The message consists of a single bit, taking values Go and Stop.

Each agent initially generates a local geometric random variable G (number of fair coin flips until the first heads, i.e., an immediate heads results in $G = 1$) and computes its **level** as $\lceil \log G \rceil$. (We can also use synthetic coin techniques [5] to simulate fair coin flips and increment their level from i to $i+1$ as they flip 2^i consecutive tails.)

We define consecutive disjoint intervals $G_0, R_0, G_1, R_1, \ldots \subset \mathbb{N}$ (**green** and **red**) partitioning the natural number line. We call R_i's last element $d_i = \max R_i$ a **door**. (See Figure 5.1, formal definition below.) Each agent keeps a local counter, initially 0, that is incremented on some interactions. An agent is **in round** i if its counter is in $G_i \cup R_i$. The goal is to get every agent to count up until the round equal to the maximum level k generated by any agent and stop its counter at d_k. An agent with level l in round i is **eager** if $i < l$ and **cautious** otherwise. Intuitively, eager agents race through doors until their own level, telling all other agents to keep going, but become cautious at and beyond their own level, advancing past a door into the next round only if another agent tells them to do so (via a message $m =$ Go). More formally, an eager agent always sends a message of Go and increments its counter on every interaction. A cautious agent sends message Go if and only if its counter is in G_i for some i, increments its counter on every interaction in $G_i \cup R_i \setminus \{d_i\}$ unconditionally, and increments its counter beyond d_i if and only if the other agent's message is Go. Agents drop out of the junta when they leave their own level, so (assuming no agent leaves the maximum level) those who generated the maximum level are the eventual junta.

To formally defined the intervals, let $c \in \mathbb{N}^+$. Each G_i, with $|G_i| = c4^i$, is called a **green** interval, R_i, with $|R_i| = \frac{3c}{2}4^i$, a **red** interval. Note that $d_i = \sum_{j=0}^{i-1}(|G_j| + |R_j|) = c\big(1 + \frac{3}{2}\big)\frac{4^i-1}{4-1} < \frac{5c}{6}4^i = \frac{5}{6}|G_i|$, so $|G_i|$ is larger by a constant multiplicative factor than the union of all the previous intervals. The max level k is $\Theta(\log\log n)$ with high probability, so its corresponding interval $|G_k| = \Theta(4^{\log\log n}) = \Theta(\log^2 n)$. Thus, with high probability, the agents will use $O(\log^2 n)$ states for their counters and stop at the door d_k after $O(\log^2 n)$ time.

To compose Junta-Election with a downstream protocol P, agents can simply restart P whenever they move beyond a d_i, and then wait to start simulating P until they reach the next d_{i+1}

(Restarting is a common technique in distributed computing for composition and is not original to this paper, e.g., [99].) In the early stages of Junta-Election, the downstream protocol gets restarted many times, but eventually, all agents will move past d_{k-1}, after which they will restart the downstream protocol for the last time. The agents will all simultaneously be in the last interval $G_k \cup R_k$ before stopping at d_k. Thus all simulated downstream interactions of P will be between agents that agree on k.

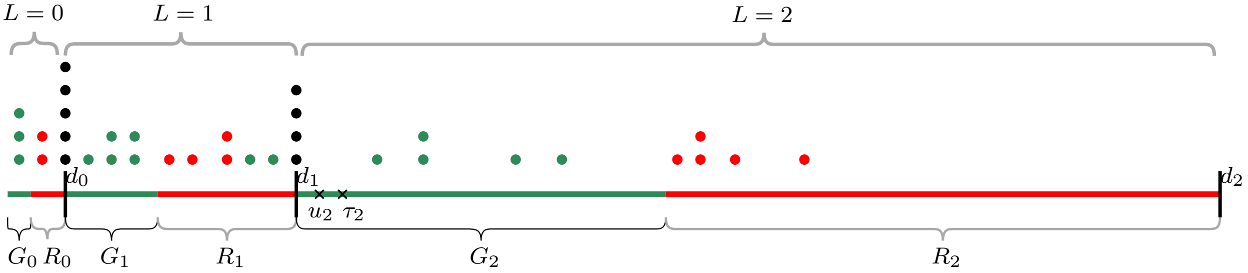


Figure 5.1. Agents, represented as dots, increment their counters through the $G_0, R_0, G_1, R_1, G_2, R_2$ intervals. Agents in green intervals or any interval before their own level have message Go. Agents in red intervals at their own level or later have message Stop. At the end of a red interval (the door d_i, shown with black horizontal line) at their own level or later, the agents (black dots) wait to increment their counter until they see a message Go. The special times marked u_i, τ_i are used in proving Lemma 5.5.4.

Theorem 5.5.1. *With probability* $1 - O(1/n)$, *Protocol 5.2 uses* $O(\log^2 n)$ *states and elects a junta of size* $O(\sqrt{n})$ *in* $O(\log^2 n)$ *time, after which* $v.\mathtt{count} = d_k$ *for all agents* v, *where* $k \in \{\lfloor \log\log n \rfloor, \lceil \log\log n \rceil, \lceil \log\log n \rceil + 1\}$.

Theorem 5.5.1 is proven formally via Lemmas 5.5.2, 5.5.3, 5.5.4, 5.5.5, 5.5.6 in Subsection 5.5.2.

Proof sketch. We must show the agents remain synchronized. By the time the interval lengths are $\Omega(\log n)$, we could argue that the number of interactions of each agent are tightly concentrated enough for agents to by synchronized in the same interval. However, the main challenge is how to reason about agents that might be stuck behind at a door.

Our argument shows that a constant fraction $n/4$ of agents stay synchronized in each green interval, up until near the max level (Lemma 5.5.4). Then, we argue that during the later green intervals, straggler agents are able to catch up, because they have a constant probability of passing through each door and the length of the green interval is more than the sum of all previous intervals. We then show the entire population in synchronized within the last few intervals (Lemma 5.5.5).

Thus all agents will have a Stop message when the population reaches the final door d_k, and the agents will stop their counters at d_k. □

Our proof techniques require setting $|G_i| = 700{\cdot}4^i$. However, simulation results (Figure 5.2) show successful convergence when $|G_i| = 16 \cdot 2^i$. Scaling the intervals this way would let $|G_M| = \Theta(\log n)$, so the protocol would take $O(\log n)$ time and $O(\log n)$ internal states.

5.5.2. Proof of Theorem 5.5.1. Theorem 5.5.1 follows from Lemmas 5.5.2, 5.5.3, 5.5.4, 5.5.5, 5.5.6, proven in this subsection.

5.5.2.1. *Distribution of levels.*

Lemma 5.5.2. *Let $n \in \mathbb{N}^+$ and consider n i.i.d. geometric random variables $V_1, \dots, V_n$. Let $i \in \mathbb{N}^+$, $E_i = |\{j \mid \lceil \log V_j \rceil = i\}|$. Let $0 < \delta < 1$. Let $\mu = n\Big(2^{-2^{i-1}} - 2^{-2^i}\Big)$. Then $\mathbb{P}\left[(1-\delta)\mu < E_i < (1+\delta)\mu\right] > 1 - 2 \cdot \exp\left(-n\delta^2 2^{-1-2^{i-1}}/3\right)$.*

Proof. Let G be a geometric random variable, so for each $a \in \mathbb{N}$, $\mathbb{P}[G > a] = 2^{-a}$. Then for each $i \in \mathbb{N}$, $\mathbb{P}[\lceil \log G \rceil > i] = \mathbb{P}\left[G > 2^i\right] = 2^{-2^i}$. Then $\mathbb{P}[\lceil \log G \rceil = 0] = 1/2$ and for $i > 0$, $\mathbb{P}[\lceil \log G \rceil = i] = \mathbb{P}[\lceil \log G \rceil > i-1] - \mathbb{P}[\lceil \log G \rceil > i] = 2^{-2^{i-1}} - 2^{-2^i}$. Note that for all $a \in \mathbb{N}^+$, $2^{-a} - 2^{-2a} \geq 2^{-1-a}$, so $2^{-2^{i-1}} - 2^{-2^i} \geq 2^{-1-2^{i-1}}$.

Now consider n i.i.d. variables $V_1, \dots, V_n$. For each $j \in \{1, \dots, n\}$, let I_j be the indicator for the event $V_j \in L_i$. Let $p = 2^{-2^{i-1}} - 2^{-2^i} \geq 2^{-1-2^{i-1}}$, noting $\mathbb{P}[I_j = 1] = p$ and $\sum_{j=1}^n I_j = E_i$, with $\mu = \mathbb{E}[E_i] = np$. Since $p > 2^{-1-2^{i-1}}$, it follows that $\mu \geq n \cdot 2^{-1-2^{i-1}}$. By independence of the V_j's and the Chernoff bound,

$$\mathbb{P}\left[E_i < (1-\delta)\mu \text{ or } E_i > (1+\delta)\mu\right] < 2 \cdot \exp\big(-\delta^2\mu/3\big) < 2 \cdot \exp\Big(-\delta^2 n 2^{-1-2^{i-1}}/3\Big).$$ □

Letting $\delta = 1/2$, and noting $n/2^{2^{i-1}+1} < \mu < n/2^{2^{i-1}-1}$, gives the following corollary.

Corollary 5.5.3. *Let $n \in \mathbb{N}^+$ and consider n i.i.d. geometric random variables $V_1, \dots, V_n$. For each $i \in \mathbb{N}^+$, let $E_i = |\{j \mid \lceil \log V_j \rceil = i\}|$. Then $\mathbb{P}\left[n/2^{2^{i-1}+2} < E_i < n/2^{2^{i-1}}\right] > 1 - 2{\cdot}\exp\left(-n2^{-5-2^{i-1}}\right)$.*

Note that Corollary 5.5.3 is useful as long as $i \leq \log\log n$. There is a $\Theta(1)$ failure probability when $i = 1 + \log\log n$, and a very large failure probability when $i \geq 2 + \log\log n$. But for $i = \log\log n$ (and smaller), the failure probability is at most $2\exp\left(-n^{1/2}/32\right)$. Of course, i is an integer and $\log\log n$ in general is not; nevertheless, with appropriate rounding we conclude that $k = \max_{j\in\{1,\ldots,n\}} \lceil \log V_j \rceil$ is very likely to be $\lfloor \log\log n \rfloor$, $\lceil \log\log n \rceil$, or $\lceil \log\log n \rceil + 1$.

In the following we use the fact that E_i is the number of agents choosing exactly level i.

For any field `field` of an agent v and any $t \in \mathbb{N}$, let $\texttt{field}v(t)$ denote the value of `field` in agent v at time t ($\frac{n}{2}t$ interactions). Write $v.\texttt{field}$ when the time is clear from context (or $v.\texttt{field}$ is constant over time, e.g. $v.\texttt{level}$).

Define u_i, τ_i to be the points $\frac{1}{16}$ and $\frac{1}{8}$ of the way through interval G_i (see Figure 5.1). Thus $u_i = d_{i-1} + \frac{c}{16}4^i$ and $\tau_i = d_{i-1} + \frac{c}{8}4^i$. (Recall we have the bound $d_{i-1} < \frac{5}{6}4^i$). At time τ_i, the average number of interactions is τ_i, and we hope for most agents' counters to also be near τ_i. S_i denotes the cautious agents that, at time τ_i, are synchronized with counters in the interval G_i. The following lemma shows that a constant fraction of the population are in S_i:

Lemma 5.5.4. *Let $i \in \{0, 1, \ldots, \lfloor \log\log n \rfloor - 1\}$. Let S_i be the set of agents v such that $v.\texttt{level} \le i$ (cautious by round i) and $v.\texttt{count}(\tau_i) \in G_i$. Then $\mathbb{P}\left[|S_i| \ge n/4\right] > 1 - O\left(\frac{\log\log n}{n^2}\right)$.*

Proof. We prove this by induction on i.

Base case. S_0 is the set of agents v with $v.\texttt{level} = 0$ and $v.\texttt{count}(\tau_0) \in G_0$. Then $\mathbb{E}\left[E_0\right] = n/2$, and taking $\delta = 1/4$ in Lemma 5.5.2, we get

$$\mathbb{P}\left[E_0 < 3n/8\right] = \mathbb{P}\left[E_0 > (1-\delta)\mu\right] \le 2 \cdot \exp\left(-n2^{-5.5}/3\right).$$

We now show that of these level 0 agents, we only lose an additional fraction of 1/3 due to drift in the first $\tau \cdot n/2$ interactions (τ units of time). We apply Lemma 5.4.3, with S as the agents at level 0, $\mu = \tau_0 = c/8$, and $\mu + h = \max G_i = c$. Then $S' = S_0$, and the error fraction

$$\epsilon_H = 2\sqrt{2\ln n/|S|} + \exp\left(-h^2/3\mu\right) \le 2\sqrt{2\ln n/(3n/8)} + \exp(-49c/24) \le 1/3,$$

for sufficiently large values of n. Then by Lemma 5.4.3, $|S_0| \ge (1-1/3)(3n/8) = n/4$ with probability $1 - 1/n^2$.

Inductive case. Assume $|S_i| \ge n/4$, recalling that for all $v \in S_i$, $v.\texttt{level} \le i$ and $v.\texttt{count}(\tau_i) \in G_i$. We will first wait until time u_{i+1}, and consider the agents from S_i and also those at level $i+1$ that have at least made it to the door $d_i < u_{i+1}$.

Let A_{i+1} be the set of agents v with $v.\texttt{level} = i+1$ and $v.\texttt{count}(u_{i+1}) \geq d_i$. Let B_{i+1} be the set of agents $v \in S_i$ with $v.\texttt{count}(u_{i+1}) \geq d_i$.

Intuitively, the agents in B_{i+1} have had enough interactions to at least be at the door d_i, but could be stuck waiting to see the signal. The agents in A_{i+1} are broadcasting a signal and moving the agents in B_{i+1} through the door. This process will be stochastically dominated by a section of an epidemic process. We must ensure $|A_{i+1}| + |B_{i+1}| > n/4$ so that we can wait for $|S_{i+1}| \geq n/4$ agents to finish this epidemic.

First, we must bound the size $|A_{i+1}|$. Let E_{i+1} be the set of agents v with $v.\texttt{level} = i+1$. Then by Corollary 5.5.3, $\mathbb{P}\left[|E_{i+1}| > n/2^{2+2^i}\right] > 1 - \exp\left(-n2^{-5-2^i}\right)$. Since $i+1 \leq \lfloor \log\log n \rfloor - 1$, we have

$$|E_{i+1}| > n/2^{2+2^{\log\log n - 2}} = n/(4 \cdot 2^{\log n/4}) = \frac{1}{4}n^{3/4}$$

Now A_{i+1} is the subset of E_{i+1} that have at least d_i interactions between time 0 and time u_{i+1}. We apply Lemma 5.4.3 with $S = E_{i+1}$, $l = u_{i+1} - d_i = \frac{c}{16}4^{i+1}$, and $\mu = u_{i+1} = d_i + \frac{c}{4}4^{i+1} < \left(\frac{5}{6} + \frac{1}{4}\right)c4^{i+1} < 18l$. Then we can use a simple upper bound for the fraction

$$\epsilon_L = 2\sqrt{2\ln n/|S|} + \exp\left(-l^2/2\mu\right) \leq 4\sqrt{8\ln n/n^{3/4}} + \exp\left(-\frac{c}{16 \cdot 18}4^{i+1}\right) \leq 1/2$$

as long as $c \geq 72$, for sufficiently large values of n. Then Lemma 5.4.3 will give that $|A_{i+1}| > (1 - \frac{1}{2})|E_{i+1}| > n/2^{3+2^i}$ with probability $1 - 1/n^2$.

Next we must bound the size $|B_{i+1}|$ again using Lemma 5.4.3. $S = S_i$, since we are starting from the agents in S_i at time τ_i (who have $v.\texttt{count}(\tau_i) < d_{i-1}$). We will consider the drift during the interval between time τ_i and u_{i+1}, so

$$\mu = u_{i+1} - \tau_i = c\left(\frac{7}{8}4^i + \frac{3}{2}4^i + \frac{1}{16}4^{i+1}\right) = \frac{21c}{8} \cdot 4^i.$$

The agents in B_{i+1} must have $v.\texttt{count}(u_{i+1}) \geq d_i$, meaning they have at least $\mu - l = d_i - d_{i-1} = c(1 + \frac{3}{2})4^i$ interactions.

Then $l = \frac{c}{8}4^i$, and Lemma 5.4.3 gives that $|B_{i+1}| \geq (1 - \epsilon_L)|S_i| \geq \frac{n}{4}(1 - \epsilon_B)$ with probability $1 - 1/n^2$, where

$$\epsilon_B = \epsilon_L = 2\sqrt{2\ln n/|S|} + \exp\left(-l^2/2\mu\right) \leq 4\sqrt{2\ln n/n} + \exp\left(-\frac{c}{336}4^i\right).$$

Now at time u_{i+1} we have $|A_{i+1}| + |B_{i+1}|$ agents that are at least at door d_i. Agents from B_{i+1} might be stuck at the door, but they will advance past as soon as they encounter an agent from A_{i+1} or another agent from B_{i+1} that has already past the door. Thus this looks like an epidemic process where $|A_{i+1}| + |B_{i+1}|$ agents are participating, and we start with at least $|A_{i+1}|$ infected agents. We will wait $\tau_{i+1} - u_{i+1}$ time and hope to reach at least $n/4$ infected agents.

However, there is the added complication that agents might drift past G_{i+1}, so they can't get counted in S_{i+1} and will no longer be acting as an infected agent in the epidemic. We will again use Lemma 5.4.3 to bound the count $|D|$ of any agents that have more than $\max G_{i+1}$ interactions by time τ_{i+1}. We use $\mu = \tau_{i+1} = d_i + \frac{c}{8}4^{i+1} < c(\frac{5}{6} + \frac{1}{8})4^{i+1}$, $h = \max G_{i+1} - \tau_{i+1} = \frac{7c}{8}4^{i+1}$, so $h^2/3\mu > \frac{49}{184}c4^{i+1} > c4^i$. We will consider the worst case for the size of $|D|$, with $|S| = n$ agents possible to drift. Lemma 5.4.3 gives that $|D| \leq \epsilon_D n$, where

$$\epsilon_D = 2\sqrt{2\ln n/|S|} + \exp\bigl(-h^2/3\mu\bigr) \leq 2\sqrt{2\ln n/n} + \exp\left(-c4^i\right)$$

Now we will make a worst case assumption that all drifted agents come from the initially infected agents A_{i+1}, and argue about an epidemic starting from $|A_{i+1}| - |D|$ infected agents with $|A_{i+1}| + |B_{i+1}| - |D|$ agents participating.

We will use the bound

$$|A_{i+1}| \geq \frac{1}{2}|E_{i+1}| = \frac{1}{4}|E_{i+1}| + \frac{1}{8}|E_{i+1}| + \frac{1}{8}|E_{i+1}| \geq \frac{1}{16}n^{3/4} + n/2^{5+2^i} + n/2^{5+2^i},$$

broken up with two terms that will dominate each of the terms in ϵ_B and ϵ_D.

Then we can bound

$$\begin{aligned}
|A_{i+1}| + |B_{i+1}| - |D| &\geq \frac{1}{16}n^{3/4} + n/2^{5+2^i} + n/2^{5+2^i} + \frac{n}{4}\Bigl(1 - 4\sqrt{2\ln n/n} - \exp\left(-\frac{c}{336}4^i\right)\Bigr) \\
&\quad - n\Bigl(2\sqrt{2\ln n/n} + \exp\left(-c4^i\right)\Bigr) \\
&\geq \frac{n}{4} + n/2^{5+2^i} + \left(\frac{1}{16}n^{3/4} - 16\sqrt{2n\ln n} - 2\sqrt{2n\ln n}\right) \\
&\quad + n\Bigl(2^{-5-2^i} - \frac{1}{4}\exp\left(-\frac{c}{336}4^i\right) - \exp\bigl(-c4^i\bigr)\Bigr) \\
&\geq \frac{n}{4} + n/2^{5+2^i} + 0 + 0
\end{aligned}$$

for sufficiently large values of n (since $\sqrt{n \ln n} = o(n^{3/4})$), and for $c \geq 700$ (since the rightmost difference is minimized at $i = 0$ at positive for $c \geq 700$).

These calculations also show that $|A_{i+1}| - |D| \geq n/2^{5+2^i}$. Thus, the true process will be stochastically dominated by a two-way epidemic, starting from $a = n/2^{5+2^i}$ infected agents and $n/4$ susceptible agents. Recall we are waiting $\tau_{i+1} - u_{i+1}$ time, which is $\frac{n}{2} \cdot \frac{c}{16} 4^{i+1} = n \cdot \frac{c}{8} 4^i$ interactions.

Now we can apply Lemma 5.4.4 (with $\alpha = 2^{-5-2^i}$ and $\gamma = 1/4$) to conclude this process will reach $s + 1 > n/4$ infected agents after T interactions, where

$$T \leq \frac{5}{\gamma} n \ln\left(\frac{\gamma}{\alpha}\right) = 20n \ln\left(2^{3+2^i}\right) = n \cdot 20 \ln 2 \cdot (3 + 2^i) \leq n \cdot \frac{c}{8} 4^i$$

with probability $1 - (\gamma n)^{-2} = 1 - 16/n^2$ (as long as $c \geq 333$).

Thus by time τ_{i+1}, at least $n/4$ agents have passed the door d_i without leave G_{i+1}. Therefore we have showed $|S_{i+1}| \geq n/4$ with probability $1 - O(1/n^2)$, completing the inductive case.

Note that we considered $i < \log \log n$ levels, where each of these inductive steps added an error probability $O(1/n^2)$. Therefore we have $\mathbb{P}\left[|S_i| \geq n/4\right] > 1 - O\left(\frac{\log \log n}{n^2}\right)$. □

Lemma 5.5.5. *Let $i \in \{\lfloor \log \log n \rfloor - 1, \ldots, k\}$, where $k = \max_v v.\texttt{level}$. Then with probability $1 - O(1/n)$, there is some time t such that $v.\texttt{count}(t) \in G_i$ for all agents v, and also some time t such that $v.\texttt{count}(t) \in R_i$ for all agents v.*

Proof. Notice that $i \geq (\log \log n) - 2$, so $4^i > \frac{1}{16} \log^2 n$, and for any constant $a > 0$, we have $\exp\left(-a4^i\right) < \exp\left(-\frac{a}{16} \log^2 n\right) < 1/n^2$ for sufficiently large n. Thus we will find that all error terms of the form $\exp\left(-l^2/3\mu\right)$ and $\exp\left(-h^2/3\mu\right)$ from Lemma 5.4.3 will now be small enough that we can use the union bound result and conclude $|L| = 0$ and $|H| = 0$.

First we consider $i = \lfloor \log \log n \rfloor - 1$. By Lemma 5.5.4, $|S_i| \geq n/4$, meaning at time τ_i, there are at least $n/4$ agents at level $\leq i$ in G_i.

Now let $\Omega = d_{i-1} + \frac{47c}{48} 4^i$ be the point $\frac{47}{48}$ of the way through the interval G_i. Let D be the set of agents that have more than $\max G_i$ interactions by time Ω. We apply Lemma 5.4.3 with $|S| = n$, $\mu = \Omega < (1 + \frac{5}{6})c4^i < 2c4^i$ and $\mu + h = \max G_i$, so $h = \frac{c}{48} 4^i$. Now we have $h^2/3\mu = \Theta(4^i)$, so as observed above $\exp\left(-h^2/3\mu\right) < 1/n^2$ and we can conclude that $|D| = 0$ with probability $1 - O(1/n)$.

Now we consider the time between τ_i and Ω. Since no agent has had more than $\max G_i$ interactions, there are at least $n/4$ agents in G_i during this entire interval. We can now show that every agent will enter G_i by the time Ω. In the worst case, an agent v could still have $v.\texttt{count}(\tau_i) = 0$. If v has an interaction while at a door during this interval, the chance of increasing its counter is at least $\frac{n}{4}/n = 1/4$. The probability of taking more than $8 \ln n$ interactions to pass a door is at most $(1 - 1/4)^{8 \ln n} < \exp\{-\frac{8}{4} \ln n\} = 1/n^2$. The probability of taking more than $8 \ln n \log \log n$ interactions to pass all doors is then at most $\log \log n/n^2$. This number of interactions is negligible compared to $d_i = \Theta(4^i) = \Theta(\log^2 n)$.

Now we are waiting

$$\Omega - \tau_i = \left(\frac{7}{8} - \frac{1}{48}\right) |G_i| > \frac{5}{6}|G_i| > d_i$$

time and need to have $d_i + o(\log^2 n)$ interactions. Thus we can again apply Lemma 5.4.3, where again $\mu = \Theta(4^i)$ and $l = \Theta(4^i)$. This will show that with probability $1 - O(1/n^2)$, every agent has enough interactions between τ_i and Ω to reach the interval G_i.

We have now shown that $v.\texttt{count}(\Omega) \in G_i$ for all v. Then the number of interactions for an agent to enter R_i is at most $|G_i| = c4^i$ and the number of interactions to reach d_i is at least $|R_i| = \frac{3c}{2}4^i$, so we can find some time $t \in R_i$ when Lemma 5.4.3 will give that with probability $1 - O(1/n^2)$, every agent will have at least enough interactions to enter R_i but not enough interactions to reach d_i.

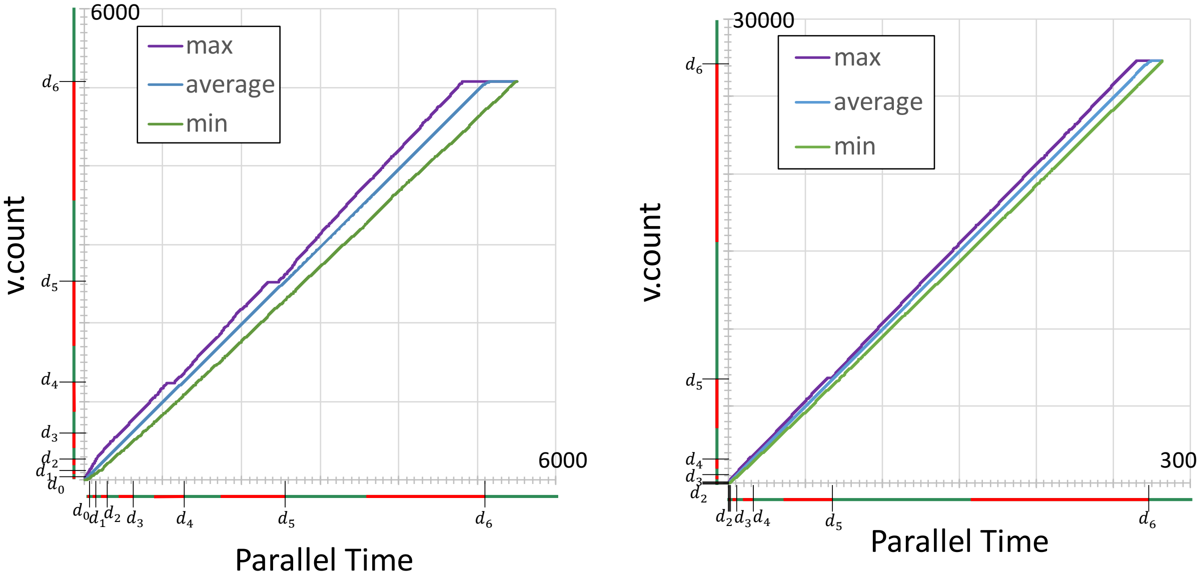
While $i < k$, there will be at least one agent at level $i+1$. We can then make the same epidemic argument as in the proof of Lemma 5.5.4. Now, we can assume in the worst case we have an epidemic that starts with one agent that must reach the whole population. By either Lemma 4.2.7 or Lemma 5.4.4, this takes $O(\log n)$ time ($O(n \log n)$ interactions, with probability $1 - 1/n^2$. This is now negligible compared to the $\Theta(\log^2 n)$ time during each green interval.

Thus we can inductively claim that for all intervals $G_{i+1}, R_{i+1}, \ldots, G_M, R_M$ there is some time t when all agents are synchronized with counts in that interval. □

Finally, we establish the state complexity bound for Protocol 5.2 stated in Theorem 5.5.1.

Lemma 5.5.6. *With probability* $1 - O(1/n)$*, Protocol 5.2 uses* $O(\log^2 n)$ *internal states.*

Proof. Note the space is dominated by the field `count` $= O(\log^2 n)$ with probability $1-O(1/n)$. All the intervals are properties of the transition function δ not the state. Finally, as written holding the level would take $O(\log \log n)$ state overhead, but we could actually compute the level on the fly and only need to track if we are still eager or cautious in each interval. □



(a) $|G_i| = 16 \cdot 2^i$, $|R_i| = 24 \cdot 2^i$

(b) $|G_i| = 2 \cdot 4^i$, $|R_i| = 3 \cdot 4^i$

FIGURE 5.2. Data from two simulations on $n = 2 \cdot 10^7$ agents, with different values of $|G_i|$ and $|R_i|$. Both yielded maximum level $k = 6$. The horizontal axis shows parallel time ($\frac{n}{2}$-interactions). The vertical axis represents the value v.count for agents (summarized for the whole population by min, max, and average). The purple line, the orange line, and the green line are respectively showing the maximum, average, and minimum v.count for agents. As shown in the figures the maximum count and minimum count drift in the middle of the protocol but eventually they all converge to d_k, where k represents the maximum level.

5.5.3. Leader-driven, $O(\log^2 n)$-convergence-time exact size counting. In this section we show a $O(\log^2 n)$ time, high-probability protocol for a problem that is natural for agents with non-constant memory: exact population size counting. The probability of error can be reduced to 0 with standard techniques; see Corollary 5.5.9. This problem has been studied in the context of open protocols, in both the exact [**43**, **85**] and approximate [**43**, **83**] settings, where it is known that open protocols can approximate n within multiplicative factor 2, by computing either $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$, using $O(\log n \log \log n)$ states, and $O(\log^2 n)$ time [**43**], and open protocols can compute the **exact** value of n, using $O(n \log n \log \log n)$ states, and $O(\log n)$ time [**43**]. Both protocols can be changed to probability-1, with a multiplicative factor increase of $O(\log n)$ states in case, i.e., $O(\log^2 n \log \log n)$ states for calculating $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$, and $O(n \log^2 n \log \log n)$ states for exactly computing n. However, note that our results below are leader-driven, so direct comparison with the leaderless results of [**43**] is not appropriate.

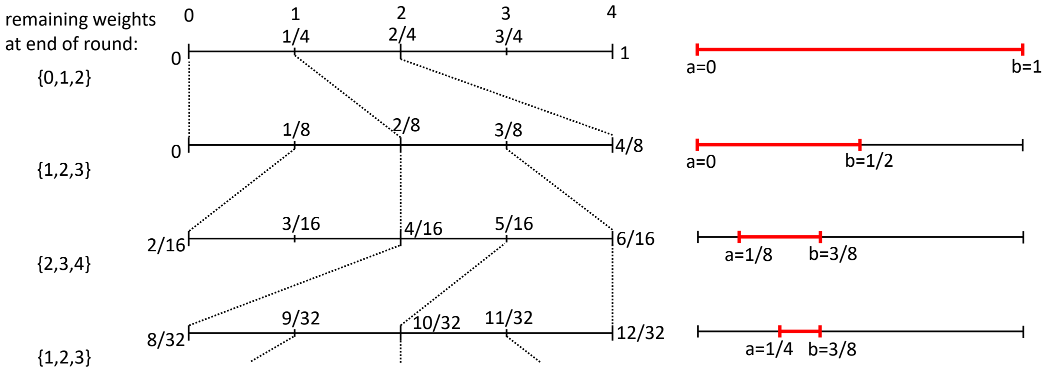


FIGURE 5.3. Update rule for fast exact counting protocol. All agents start with a `mass` of 0 and weight $\mathtt{w} = 0$, except the leader, who starts with $\mathtt{mass} = 1$ and $\mathtt{w} = 4$. They conduct averaging on weight w for one round, at which point (with high probability) three consecutive weights remain. The figure shows how the remaining masses map to the next subinterval, with the weight w updating to $2(\mathtt{w} - \mathtt{w}_{\min})$ where $\mathtt{w}_{\min}$ is the minimum value of w at the end of the Averaging phase. The right side shows the subintervals to scale. Each agent updates its internal state to represent the interval $[a, b]$. Once $[a, b]$ contains only a single number of the form $\frac{1}{n}$, the protocol terminates, and each agent knows the value n. The first $\log n$ rounds would always have 0 as the minimum remaining weights, but we allow other values to show concretely how the updating rule works.

Protocol 5.3 EXACT-COUNTING(Agent v seeing message m) is leader-driven with message fields $\mathtt{leader} \in \{L, F\}$, $\mathtt{w}, \mathtt{w}_{\min} \in \{0, 1, 2, 3, 4\}$, and internal field $\mathsf{a} \in \mathbb{Q}$.
Subroutine LEADER-DRIVEN-PHASE-CLOCK gives internal field round $\mathtt{r} \in \mathbb{N}$, message field $\mathtt{phase} \in \{\mathsf{Averaging}, \mathsf{Updating}\}$, and uses $O(1)$ message overhead to communicate the current phase.
initial state of agent v: $\mathtt{w} \leftarrow 0$ if $\mathtt{leader} = F$, $\mathtt{w} \leftarrow 4$ if $\mathtt{leader} = L$, $\mathsf{a} \leftarrow 0.00$, $\mathtt{r} \leftarrow 0$, $\mathtt{phase} \leftarrow \mathsf{Averaging}$

```
    ▷ w is a weight with a shrinking $2^{-2-\mathtt{r}}$ units of mass    ▷ each agent has $\mathtt{mass} = \mathsf{a} + \mathtt{w}/2^{2+\mathtt{r}}$
execute LEADER-DRIVEN-PHASE-CLOCK            ▷ round r has phases Averaging then Updating
if $v$.phase = $m$.phase = Averaging then
    if $v$ is initiator then
        $v.w \leftarrow \left\lceil \frac{v.\mathtt{w}+m.\mathtt{w}}{2} \right\rceil$            ▷ average both weights
    else
        $v.\mathtt{w} \leftarrow \left\lfloor \frac{v.\mathtt{w}+m.\mathtt{w}}{2} \right\rfloor$
if Averaging phase just ended then
    $v.\mathtt{w}_{\min} \leftarrow \min(v.\mathtt{w}, 3)$
if $v$.phase = $m$.phase = Updating then
    $v.\mathtt{w}_{\min} \leftarrow \min(v.\mathtt{w}_{\min}, m.\mathtt{w}_{\min})$        ▷ learn the minimum weight in the population
if Updating phase just ended then
    $v.\mathsf{a} \leftarrow v.\mathsf{a} + \frac{\mathtt{w}_{\min}}{4 \cdot 2^{\mathtt{r}}}$            ▷ update the mass interval lower bound
    $v.\mathtt{w} \leftarrow 2(v.\mathtt{w} - v.\mathtt{w}_{\min})$        ▷ update the weight to preserve mass for new round
    if $[v.\mathsf{a}, v.\mathsf{a} + 2^{-\mathtt{r}}]$ contains a unique $\frac{1}{n}$ then
        terminate with population size $n$
```

THEOREM 5.5.7. *There is an $O(1)$-message leader-driven population protocol (*EXACT-COUNTING*) that, with probability $1 - O(1/n)$, exactly counts the population size n (storing it in each agent's internal state), in $O(\log^2 n)$ time and using $O(n \log^2 n)$ states.*

PROOF. Intuitively, the protocol tries uses the discrete averaging technique that has been useful in other population protocols [**13**, **43**, **85**, **132**, **133**], in which each agent holds an integer and computes the transition $i, j \to \left\lfloor \frac{i+j}{2} \right\rfloor, \left\lceil \frac{i+j}{2} \right\rceil$. In the $O(1)$-message setting, of course, this will not work exactly as described.

Intuitively, the leader will distribute 1 unit of what we can imagine is a continuous mass into the population. Rational-valued averaging of this mass would result in each agent converging to $1/n$, from which n can be computed. $O(1)$ messages cannot represent arbitrary rationals. Instead, we allow agents to communicate a few bits of their number at a time, while ensuring that before moving on, they agree on an interval containing the true average, which shrinks by half each round, synchronized by a leader-driven phase clock [**21**]).

Figure 5.3 shows the updating rule.

Each agent's state will represent an interval $[a, b] \subseteq [0, 1]$, where $b - a = 2^{-r}$ during round $r \in \mathbb{N}$ (initialized to $r = 0$). a will be a dyadic rational, initialized to $a = 0.00$, containing $r + 2$ bits after the binary point. There is a message field $W = \{0, 1, 2, 3, 4\}$ describing varying amounts of extra weight. The value $w \in W$ counts for $\frac{w}{4 \cdot 2^r}$ units of mass in round r. An agent is interpreted as having $\texttt{mass} = a + \frac{w}{4 \cdot 2^r} \in [a, b]$ (note representing $\frac{w}{4 \cdot 2^r}$ is what requires $r + 2$ bits after the binary point). The leader is initialized with $w = 4$ (and $\texttt{mass} = 0.00 + \frac{4}{4 \cdot 1} = 1.00$), and the followers are initialized with $w = 0$ (and $\texttt{mass} = 0.00$).

We will prove that with high probability every agent will always have the same value of a. This will imply, via the averaging rule for weights, that $\texttt{mass}$ is conserved and the sum of $\texttt{mass}$ in the population is 1. Thus for all agents at all times, it holds that the true average $\frac{1}{n}$ stays within the interval $[a, b]$.

We first discuss the guarantees of the leader-driven phase clock from [**21**]. The agents will go through consecutive rounds, where each round contains an Averaging phase followed by an Updating phase. Choosing appropriate constant parameters, we can ensure with probability $1 - O(1/n)$ that for time $\Omega(n)$, all agents are synchronized within each phase for $\Theta(\log n)$ time, and no two agents are ever more than one phase apart (thus we can be sure that two agents in the same type of phase are also in the same round).

Averaging phase When two agents meet and both are in the Averaging phase, they each update just their weights via the standard averaging rule $i, j \to \left\lfloor \frac{i+j}{2} \right\rfloor, \left\lceil \frac{i+j}{2} \right\rceil$ for any $i, j \in W$. Note that assuming the invariant that every agent agrees on a, this rule preserves the sum of $\texttt{mass}$ in the population.

Averaging takes time $\Theta(n)$ to converge in the worst case, where convergence happens when all agents agree on one of two consecutive integers a and $a + 1$. (Thus further interactions are null.) However, Berenbrink, Friedetzky, Kaaser, and Kling [38] show that with probability $1 - O(1/n^2)$ it takes only $O(\log n)$ time to reach a configuration where all agents share **three** consecutive integers, two of which are a and $a + 1$. The third could be either $a - 1$ or $a + 2$, depending on the true population-wide average; full convergence happens when all remaining $a - 1$'s encounter $a + 1$'s in the former case, and when all remaining $a + 2$'s all encounter a's in the latter case. Thus, for appropriate constant parameters of the phase clock, with probability $1 - O(1/n)$, every Averaging phase lasts long enough such that at the end of each Averaging phase, we have $\max_w \leq \min_w + 2$.

Updating phase During the Updating phase, each agent spreads by epidemic the minimum weight $\texttt{w}_{\min}$ they have seen since the start of the Updating phase. By Lemma 4.2.7, we can guarantee with probability $1 - O(1/n)$ that every Updating phase lasts long enough for every agent to learn the minimum remaining message. (To ensure the $\texttt{w}_{\min}, \texttt{w}_{\min} + 1, \texttt{w}_{\min} + 2 \in W$ in case $\texttt{w}_{\min} > 3$ we take $\min(\texttt{w}_{\min}, 3)$).

At the end of the Updating phase, the agents update their lower bound to $a_{r+1} = a + \frac{\texttt{w}_{\min}}{4 \cdot 2^r}$. Because all agents agree on $\texttt{w}_{\min}$, they still agree on the value a as desired. Assuming the weight in round r was w_r, the weight updates to $w_{r+1} = 2(w_r - \texttt{w}_{\min})$. Because we have $\max_w \leq \texttt{w}_{\min} + 2$ after the successful Averaging phase, the set of weights at the start of every round will be in $\{0, 2, 4\}$ (see Figure 5.3). Notice that $\texttt{mass} = a_{r+1} + \frac{m_{r+1}}{4 \cdot 2^{r+1}} = a + \frac{m}{4 \cdot 2^r}$ is preserved during the update as desired.

Thus we have finished proving the invariant that all agents store the same interval $[a, b]$, and the sum of $\texttt{mass}$ is conserved, so $\frac{1}{n} \in [a, b]$. Finally, we can consider the time and space complexity, assuming that with probability $1 - O(1/n)$, this invariant holds and each round is $\Theta(\log n)$ time.

We first analyze the first round r when the minimum weight $\mathtt{w}_{\text{min}} > 0$, so through this round we have $a = 0$ for all agents. Since the minimum weight $\mathtt{w}_{\text{min}} \geq 1$, the minimum $\mathtt{mass}$ is at least $0 + \frac{1}{4 \cdot 2^r} \leq \frac{1}{n}$, so $r \geq \log n - 2$. The maximum $\mathtt{mass}$ is at most $0 + \frac{4}{4 \cdot 2^r} \geq \frac{1}{n}$, so $r \leq \log n$. Thus we will have $\mathtt{w}_{\text{min}} > 0$ for the first time, increasing the lower bound a after the $\mathsf{Updating}$ phase, at the end of round r, where $\log n \leq r \leq \log n + 2$. Corollary 5.5.8 argues how to use this fact to obtain a $O(\log n)$-state protocol for estimating $\log n$.

We next analyze the first round r when the interval $[a, b]$ contains a unique reciprocal $\frac{1}{n}$. It is necessary and sufficient to have $\frac{1}{n+1} < a \leq b = a + 2^{-r} < \frac{1}{n-1}$. Thus it is necessary for $2^{-(r+1)} < \frac{1}{n-1} - \frac{1}{n+1} = \frac{2}{n^2-1}$, so $r > \log\big(n^2 - 1\big) - 1$. In the other direction, if round $r - 1$ did not uniquely determine n, then $2^{-(r-1)} \geq \frac{1}{n} - \frac{1}{n+1} = \frac{1}{n(n+1)}$, so $r \leq \log(n(n+1)) + 1$. Thus the protocol will terminate at the start of round $r = 2 \log n + O(1)$, and will take $O(\log^2 n)$ time.

Next, we analyze the space complexity. Naively storing a (with $r+2$ bits after the binary point) would use $2^{2 \log n + O(1)} = O(n^2)$ states. However, by the arguments given above, we have $a = 0$ until round $r_1 \approx \log n$, so simply store the counter r_1 to denote how many leading zeros a has. The protocol will terminate at round $r_2 \approx 2 \log n$, so we can store a with $\log n + \log \log n + O(1)$ bits. Including the counter r and all constant space overhead gives a space bound of $\log n + 2 \log \log n + O(1)$ bits, so the total number of states is $O(n \log^2 n)$. □

Terminating Exact-Counting early gives a more space efficient protocol for estimating $\log n$:

Corollary 5.5.8. *A leader-driven, $O(1)$-message protocol, with probability $1 - O(1/n)$, computes $r \in \{\lfloor \log n \rfloor, \lceil \log n \rceil\}$, in $O(\log^2 n)$ time using $O(\log n)$ states.*

Proof sketch. We run Exact-Counting until the interval $[a, b]$ contains exactly one power of two 2^{-k}, and then output k, unless it contains no powers of two, in which case we output arbitrarily either of the powers of 2 contained in the interval of the **previous** round. If $n = 2^k$, then $k = \log n$ exactly. Otherwise, since the interval contains no other power of 2, but it contains $1/n$, then $k \in \{\lfloor \log n \rfloor, \lceil \log n \rceil\}$. □

Proof. We run Exact-Counting until the interval $[a, b]$ contains exactly one power of two 2^{-k}, and then output k (with one exception described below). If $n = 2^k$, then $k = \log n$ exactly. Otherwise, since the interval contains no other power of 2, but it contains $1/n$, then $k \in \{\lfloor \log n \rfloor, \lceil \log n \rceil\}$.

The interval $[a,b]$ endpoints are eventually arbitrary dyadic rationals. However, as observed in the proof of Theorem 5.5.7, in the first $\log n - 1$ rounds, the interval is of the form $[0, 2^{-j}]$, storable using only $O(\log n)$ states, because the minimum weight $\mathtt{w}_{\min} = 0$. In the next round, depending on $\mathtt{w}_{\min}$, the interval becomes one of $I_1 = [2^{-(j+2)}, 2^{-(j+2)} + 2^{-(j+1)}]$ (if $\mathtt{w}_{\min} = 1$), or $I_2 = [2^{-(j+1)}, 2^{-j}]$ (if $\mathtt{w}_{\min} = 2$). If I_1, then $2^{-(j+2)}$ is the only power of two in the interval.

If I_2, then one more round must pass. Note I_2 has only two powers of 2, the endpoints, and the interval will shrink to I_1' in the next round, containing one of the endpoints (if $w = 0$ or 2), or neither (if $w = 1$). If I_1' contains neither, then we know n is not a power of 2, so we output $2^{-(j+1)}$ or 2^{-j} arbitrarily, since $j = \lfloor \log n \rfloor$ and $j + 1 = \lceil \log n \rceil$.

Only $O(1)$ extra states are needed to advance one more round, so $O(\log n)$ total states suffice. □

By the standard technique of running in parallel with a slower deterministic counting protocol, we can convert Exact-Counting to have probability 0 of error while retaining fast convergence time.

Corollary 5.5.9. *There are $O(1)$-message, leader-driven population protocols that, with probability 1, respectively count the exact population size n and estimate it by computing $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$, both with expected $O(\log^2 n)$ convergence time and $O(n \log^2 n)$ stabilization time. With probability $1 - O(1/n)$, they use $O(n^4 \log^4 n)$ and $O(\log^2 n)$ states, respectively.*

Proof. First consider the case of exact size counting. We can compose Exact-Counting with a slow stable counting algorithm. As a backup, we could use the deterministic broadcast mechanism sketched in Corollary 5.3.6, which stably counts the population with $O(n^3 \log^2 n)$ states in expected $O(n \log^2 n)$ time. Together with the $O(n \log^2 n)$ states of Exact-Counting, this is $O(n^4 \log^4 n)$ states.

Because our protocol is leader-driven, we can use standard tricks to have the leader set a timer (see [**21**]) for when to tell all agents via epidemic to change their output to the deterministic backup. With probability $1 - O(1/n)$, the fast Exact-Counting will correctly compute n, and the timer will not go off until the backup has also stabilized. The probability $O(1/n)$ for errors add an expected $O((n \log^2 n)/n) = O(\log^2 n)$ convergence time to wait for the slow backup. Note the stabilization time is $\Omega(n \log^2 n)$ because until the slow backup has stabilized, there is a chance of switching to the backup before it is correct.

The case of size estimation is similar, although unlike the case of exact counting, in this paper we do not have a $O(1)$-message protocol that directly computes $\log n$ with probability 1. However, there is a simple open $O(\log n)$-state protocol that computes $\lfloor \log n \rfloor$: All agents start in state ℓ_1, and for each i and $j < i$ we have the transitions $\ell_i, \ell_i \to \ell_{i+1}, f_{i+1}$ and $f_i, f_j \to f_i, f_i$. This takes expected time $O(n)$ to elect a leader $\ell_{\lfloor \log n \rfloor}$ by the first type of transition and expected time $O(\log n)$ to propagate the value $\lfloor \log n \rfloor$ to all agents by epidemic.

Theorem 5.3.3 shows how any open, $s(n)$-state protocol can be simulated by a leader-driven, $O(1)$-message protocol with $O(n \log s(n))$ expected slowdown. This implies a leader-driven, probability-1 protocol for calculating $\lfloor \log n \rfloor$ with expected time $O(n^2 \log \log n)$. By combining this with the fast, error prone protocol described in Corollary 5.5.8, and setting the phase clock parameters of Exact-Counting to ensure probability of error at most $1/n^2$, the contribution to the expected time of the slow, probability-1 protocol is negligible, and the whole protocol runs in expected time $O(\log^2 n)$ time as in Corollary 5.5.8. It contributes $O(\log n)$ state complexity, so the total number of states is $O(\log^2 n)$. □

The next corollary shows that Exact-Counting can be made leaderless by composing with the leader election protocol derived from junta election (Protocol 5.2).

Corollary 5.5.10. *There is a leaderless, $O(1)$-message population protocol that exactly counts the population size n in $O(\log^2 n)$ time and $O(n \operatorname{polylog} n)$ states, succeeding with probability $1-O(1/n)$. There is also a leaderless, $O(1)$-message population protocol that computes $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$ in $O(\log^2 n)$ time and $O(\operatorname{polylog} n)$ states, succeeding with probability $1 - O(1/n)$.*

Proof. Both protocols work similarly. We can use Junta-Election (Protocol 5.2) to get a leader election protocol as in [**99**]. This will use $O(\log^2 n)$ state overhead and take $O(\log^2 n)$ parallel time. With probability $1 - O(1/n)$, all agents in Protocol 5.2 will restart when they enter the last level together with the same constant-factor estimate of $\log n$. They can use this estimate to set a timer to wait for the leader election to converge after $O(\log^2 n)$ time. Then we can start the downstream Exact-Counting to count the population, either exactly as in Theorem 5.5.7 or approximately as in Corollary 5.5.8. □

5.5.4. Leader-driven, $O(\log^2 n)$-time predicate computation. We can use techniques from Theorem 5.5.7 to show how to compute, using a leader and with high probability, any predicate on a constant alphabet Σ, up to the space bounds allowed by the agents. We assume that there is one leader agent, and that every other agent has a state from a fixed alphabet Σ. Exactly the semilinear predicates are computable with probability 1 by $O(1)$-state open protocols [**20**] (with $> \log n$ states, more predicates are possible [**62**]).

Corollary 5.5.11. *Let $d \in \mathbb{N}^+$ and let Σ be a d-symbol input alphabet. Then there is an $O(1)$ message leader-driven population protocol that, with probability $1 - O(1/n)$, exactly counts the input vector $\mathbf{i} \in \mathbb{N}^d$ (storing it in each agent's internal state), in $O(d \log^2 n)$ time and using $O(n^d \log^2 n)$ states.*

Proof sketch. Agents first run the Exact-Counting of Theorem 5.5.7 to store locally the value n. Agents then use a similar strategy to Exact-Counting to count how many agents have input x for each symbol $x \in \Sigma$. Having now stored the entire initial population's input in their internal state, they can simply compute any computable predicate ϕ locally. □

Proof. First, agents run the Exact-Counting of Theorem 5.5.7 to store locally the value n. This protocol is terminating (i.e., agents signal when they are done and with high probability, no agent signals before all agents have converged), so it can be straightforwardly composed with the subsequently described protocol. Note that the state bound was $O(n \log^2 n)$, but we can store n in a separate field that will only contribute $O(n)$ additional state overhead.

Next, we iterate over each element $x \in \Sigma$, counting the number of elements with symbol x in the population, using the same transitions as Exact-Counting, except now each agent storing x starts with weight $w = 4$ and other agents start with $w = 0$. The same argument as the proof of Theorem 5.5.7, only now the total mass is $|x|$, and the agents will wait until the interval $[a, b]$ contains only one number $\frac{k}{n}$ for $k \in \mathbb{N}$. (Note this will only require an interval of length $O(\frac{1}{n})$ and thus take $\log n + O(1)$ rounds). After each of these sub-protocols terminates, the agents store the number $k = |x|$ in their internal state. Finally, note that the last element does not need to be counted, as it can recovered as the difference between n and the counts of the other inputs.

It follows that this protocol will take $O(d \log^2 n)$ time and use $O(n^d \log^2 n)$ states. Alternately, we could run all these steps in parallel, which would reduce the time by a constant factor d, but increase the amount of messages used (to have d independent copies of the weight and min weight fields w, i for Exact-Counting). The error probability follows from that of Theorem 5.5.7, taking a union bound over each of the $d+1$ instances of Exact-Counting. □

If an agent can locally store the entire initial configuration, it can compute any predicate computable by the transition function δ. Formally, we required that δ be computable by a Turing Machine with $O(\log s(n))$ bits of memory, to make our model comparable with [**62**]. Thus we can compute all predicates computable by $O(\log n)$ bit space-bounded Turing Machines via Corollary 5.5.11.

5.6. Computability with one-bit messages

We will show that with one-bit messages, it is possible to simulate a synchronous system that provides a one-bit broadcast channel. This in turn will be used to simulate more complex systems. The price is that we sacrifice stabilization for convergence, and rely on unbounded counters to ensure convergence in the limit with probability 1.

Let us begin by defining the simulated system. A **synchronous broadcast system** consists of n synchronous agents that carry out a sequence of **rounds**. In a broadcast round, each agent generates a one-bit outgoing message. These outgoing messages are combined using the OR function to produce the outcome for this round.

Broadcast operations can be used to detect conditions such as the presence of a leader, or ordinary message transmission if a unique agent is allowed to broadcast in a particular round. However, because broadcast operations are symmetric, they cannot be used for symmetry breaking. For the purpose of electing a leader, we assume that agents have the ability to flip coins; once we have a leader, further agents may be recruited for particular roles using an auxiliary protocol that allows the leader to select a single agent from the population in some round. The broadcast and selection protocols are mutually exclusive: either all agents participate in a broadcast in some round or all agents participate in selection. This is possible by showing that all agents eventually agree on the round number forever with probability 1.

Simulating this model in a population protocol requires (a) enforcing synchrony across agents, so that each agent updates its state consistently with the round structure; (b) implementing the broadcast channel that computes the OR of the agents' outputs; and (c) implementing the selection protocol. We show how to do this in the following section.

5.6.1. Implementing the core primitives. Broadcasts are implemented by epidemics that propagate 1 messages, separated by barrier phases in which all agents display 0. Selection is implemented by having the leader display a 1 to the first agent it meets. Both protocols depend on the number of steps at each agent being approximately synchronized with high probability; after $t(n/2)$ steps, all agents' step counts should be within the range $t \pm O(\sqrt{t \log n})$ with high probability (see Lemma 5.4.1). The time to carry out a broadcast is also $O(\log n)$ with high probability (see Lemma 5.4.4). By increasing the length of each round over time, the total probability across all rounds of an error occurring in either the broadcast or selection protocol due to out-of-sync agents or slow broadcasts converges to a finite value. Applying the Borel-Cantelli lemma then shows that there is a round after which no further failures occur with probability 1.

5.6.1.1. *Details.* Observe that the probability that a particular agent i participates in an interaction is exactly $2/n$, and that the events that i participates in distinct interactions are independent. If we let X_i^t be the indicator variable that agent i participates in the t-th interaction, then $S_i^t = \sum_{j=1}^{t} X_i^t$ is a sum of independent Bernoulli random variables, and obeys the Chernoff bound $\mathbb{P}\left[|S_i^t - \mu| > \mu\delta\right] < 2e^{-\mu\delta^2/3}$, where $\mu = \mathbb{E}\left[S_i^t\right] = 2t/n$ and $0 \leq \delta \leq 1$.

The execution of each agent is organized as a sequence of rounds, where each round r for $r = 1, 2, \ldots$ consists of exactly $5r^2$ steps. The first $2r^2$ steps will be a **barrier phase** during which the agent displays message 0 and updates its state during an interaction only by incrementing its step counter. The remaining $3r^2$ steps will be an **interaction phase** in which the agents may execute one of two protocols. In a **broadcast phase**, each agent will propagate an epidemic represented by message 1, recording if it observed such an epidemic and possibly initiating the epidemic itself if instructed to do so by the protocol. In a **selection phase**, a leader agent displays 1 for its first encounter, and the agent interacting with the leader receives a special mark. The choice of broadcast/selection phase is determined by the controlling protocol and is the same for all agents. As in a barrier phase, an agent in an interaction phase continues to update its step counter with each interaction.

The controlling protocol updates the state of the agent at the end of each round. Each agent v has a state $v.\texttt{state}$ that is one of $\mathsf{broadcasting}$ (agent is initiating a broadcast of value 1) $\mathsf{receiving}$ (agent is waiting to detect a 1), $\mathsf{received}$ (agent has detected a 1), $\mathsf{selecting}$ (agent is attempting to select another agent), $\mathsf{candidate}$ (agent is a candidate for selection), $\mathsf{selected}$ (agent has been selected), or idle (agent has selected another agent and is now waiting for the end of the round). We assume that the controlling protocol assigns consistent values to the agents in each phase: if one or more agents start in state broadcasting, the rest should start in state receiving; while if some agent starts in state selecting, the rest should start in state candidate. Pseudocode for the communication protocol is given in Algorithm 5.4 in 5.6.2, followed by a proof of its correctness. The correctness of this protocol relies on carefully chosen phase interval lengths which allow us to simulate synchronous rounds wherein the above operations are executed in sequence.

Protocol 5.4 Convergent-Broadcast(Agent v seeing message m)

```
$v.\texttt{tick} \leftarrow v.\texttt{tick} + 1$
if $v.\texttt{tick} < 2r^2$ then                                        ▷ Barrier phase: do nothing
else if $v.\texttt{tick} = 2r^2$ and $v.\texttt{state} =$ broadcasting then   ▷ End of barrier phase: start epidemic
    $v.m \leftarrow 1$
else if $v.\texttt{tick} = 5r^2$ then                                   ▷ End of interaction phase
    Update $v.\texttt{state}$ according to controlling protocol
    $r \leftarrow r + 1$
    $v.\texttt{tick} \leftarrow 0$
else if $v.\texttt{tick} > 2r^2$ and $v.\texttt{state} =$ receiving and $m = 1$ then   ▷ Receive and propagate epidemic
    $v.\texttt{state} \leftarrow$ received
    $v.m \leftarrow 1$
else if $v.\texttt{tick} = 3r^2$ and $v.\texttt{state} =$ selecting then   ▷ Attempt to select
    $v.m \leftarrow 1$
else if $v.\texttt{tick} > 3r^2$ and $v.\texttt{state} =$ selecting and $m = 0$ then   ▷ Selected a candidate
    $v.m \leftarrow 0$
    $v.\texttt{state} \leftarrow$ idle
else if $v.\texttt{tick} > 2r^2$ and $v.\texttt{state} =$ candidate and $m = 1$ then ▷ We are the selected candidate
    $v.\texttt{state} \leftarrow$ selected
```

5.6.2. Convergent Broadcast Algorithm. Define $s_r = \sum_{j=1}^{r-1} 5r^2$; this is the total length of all rounds up to but not including r. Observe that $s_r = \Theta(r^3)$. Consider the midpoint $a_r = s_r + r^2$ of the barrier phase of round r. Let A_{ir} be the event $|S_i^t - a_r| > r^2 - 1$, where $t = (n/2)a_r$ so that $\mathbb{E}\left[S_i^t\right] = a_r$. Then the Chernoff bound gives $\mathbb{P}\left[A_{ir} < 2e^{-a_r((r^2-1)/a_r)^2/3}\right] = e^{-\Theta(r)}$. Similarly define $b_r = s_r + 3r^2$ and $c_r = s_r + 4r^2$ as the steps $1/3$ and $2/3$ of the way through the interaction phase of round r, and define B_{ir} as the event $|S_i^t - b_r| > r^2 - 1$ when $t = (n/2)b_r$ and C_{ir} as the event $|S_i^t - c_r| > r^2 - 1$ when $t = (n/2)c_r$. Then we also have $\mathbb{P}\left[B_{ir}\right] = e^{-\Theta(r)}$ and $\mathbb{P}\left[C_{ir}\right] = e^{-\Theta(r)}$.

Finally, define D_{ir} as the event that the schedule of interactions is such that an epidemic that has infected agent i after $(n/2)b_r$ steps has not infected all agents after $(n/2)c_r$ steps. Note that this definition does not depend on whether an actual epidemic is in progress after $(n/2)b_r$ steps; instead, we consider a hypothetical epidemic starting at i running on the same schedule. From Lemma 4.2.7, we have that for any two-way epidemic on n processes, the expected value $\mathbb{E}[T_n]$ of the number of interactions to infect all agents is $O(n \log n)$ and $\mathbb{P}[T_n > (1+\delta)\mathbb{E}[T_n]] \leq 2.5 \ln(n) \cdot n^{-2\delta}$. For D_{ir} to occur, we need $T_n > r^2$, giving $\delta = r^2/O(n \log n) - 1$ and thus $\mathbb{P}[D_{ir}] = e^{-\Omega(r^2/n \log n)} \ln n = e^{-\Omega(r^2/n \log n)}$.

Call a round r **safe** if none of the events A_{ir}, B_{ir}, C_{ir}, or D_{ir} occur for any $i \in \{1, \dots, n\}$. These events are not even remotely independent, but the the union bound still applies, giving a probability that round r is not safe of at most $3ne^{-\Omega(r)} + ne^{-\Omega(r^2/n \log n)} = e^{\Omega \log n - r} + e^{\Omega \log n - r^2/n \log n}$. The sum of of these bounds over all rounds converges to a finite value for any fixed n, so by the Borel-Cantelli lemma, with probability 1 all but finitely many rounds are safe.

The following lemmas demonstrate that the protocol does what it is supposed to, once we reach the suffix of the execution containing only safe rounds. We start by excluding false positive broadcasts.

Lemma 5.6.1. *If rounds r and $r+1$ are both safe, then no process observes a 1 in round r unless some process initiates a broadcast or selection in round r.*

Proof. If rounds r and $r+1$ are both safe, then the events A_{ir} and $A_{i,r+1}$ do not occur for any i. In particular, this means that at time $t = (n/2)a_r$, all agents have an internal clock S_i^t that is within the interval $a_r \pm r^2 - 1$, which lies within the barrier phase for round r. So at this time all agents display message 0, have completed the interaction phase for round $r-1$, and have not yet started the interaction phase for round r. A similar constraint holds at time $t' = (n/2)a_{r+1}$. It follows that any 1 observed by an agent during its round-r interaction phase must result from some process setting its message to 1 either because it initiated an epidemic or selection during its own round-r interaction phase, or because it is propagating an epidemic initiated by such a process. □

Similarly, a safe round has no false negative broadcasts:

Lemma 5.6.2. *If round r is safe and all agents start round r in either a* broadcasting *or* receiving *state, then any epidemic initiated in round r is observed by all agents.*

Proof. Because B_{ir} does not occur for any i, after $(n/2)b_r$ steps, all agents are in their round-r interaction phase, and because C_{ir} does not occur for any i, all agents remain in their round-r

interaction phase until at least $(n/2)c_r$ steps. If some agent i initiates an epidemic in round r, then $i.\mathtt{state} = \mathsf{broadcasting}$ after $(n/2)b_r$ steps, and under the assumptions, of the lemma every other agent is either in the $\mathsf{broadcasting}$, $\mathsf{receiving}$, or $\mathsf{received}$ state. A simple induction shows that the set of infected agents in the real process throughout the $[(n/2)b_r, (n/2)c_r]$ interval is bounded below by the set of infected agents in the hypothetical epidemic considered in the definition of D_{ir}. This means that if D_{ir} does not occur, both such sets contain all processes after $(n/2)c_r$ interactions. □

And a safe round allows selection. Selection is not necessarily uniform conditioned on safety, but each agent has an $\Omega(1/n)$ chance of being selected when r is sufficiently large:

Lemma 5.6.3. *If round r is safe, exactly one agent i starts round r in a* $\mathsf{selecting}$ *state, and all other agents start round r a* $\mathsf{candidate}$ *state, then exactly one agent finishes round r in a* $\mathsf{selected}$ *state. For each agent j, the probability p that its is chosen conditioned on the safety of round r and the events of previous rounds is at least $\frac{1}{2n}$ for sufficiently large r.*

Proof. Use the non-occurrence of any B_{ir} or C_{ir} to argue that agent i reaches tick $3r^2$ while all agents are in the interaction phase. Then the next interaction between i and any j causes j to observe a 1 and switch to a $\mathsf{selected}$ state.

We would like to argue that the next interaction between i and another agent j chooses each j with independent probability $1/n$. Unfortunately, we are conditioning on safety of round r. Let A be the event that i selects j and B the event that round r is unsafe. Then $\mathbb{P}[A \mid \neg B] = \frac{\mathbb{P}[A \wedge \neg B]}{\mathbb{P}[\neg B]} \geq \mathbb{P}[A \wedge \neg B] = \mathbb{P}[A] - \mathbb{P}[A \wedge B] \geq \mathbb{P}[A] - \mathbb{P}[B] = 1/(n-1) - \left(e^{-\Omega(r)} + e^{-\Omega(r^2/n \log n)}\right) \geq \frac{1}{2n}$ for sufficiently large r. □

5.6.3. Convergent computation of arbitrary symmetric functions. Early rounds produce incorrect results, so we need an error-recovery mechanism. We describe a basic protocol for electing a leader and having it gather inputs from the other agents. This in principle allows the leader to compute the output of an arbitrary symmetric function and broadcast it to the other agents. The protocol guarantees termination with probability 1 even in executions where some of the rounds exhibit errors in the underlying broadcast mechanism. By restarting the protocol when it terminates, we can guarantee that the protocol eventually runs without errors, thus converging to the correct output.

Each agent v maintains a Boolean field v.`leader` that marks it as a leader (or candidate leader) and a field v.`processed` that marks whether it has reported its input v.`input` to the leader. All agents rotate through a repeating sequence of 7 rounds, where the round number for the purposes of the protocol is $r \bmod 7$. These are organized as follows:

Round 0: Any leader broadcasts 1. A non-leader that receives 0 sets its `leader` bit. This round allows recovery from states with no leaders.

Round 1: Any leader broadcasts 1 with probability 1/2. A leader that does not broadcast but receives a 1 clears its `leader` bit.

Round 2: Any agent that cleared its `leader` bit in the previous round broadcasts 1. This causes any remaining leaders that receive a 1 to restart the information-gathering protocol and causes any non-leaders that receive a 1 to clear their `processed` bits. Broadcasting a 1 in this round is also used by the leader to restart the protocol after completion.

Round 3: Any agent v with v.`processed` $= 1$ broadcasts 1. This is used by the leader and other agents to detect unprocessed inputs.

Round 4: If a leader received a 1 in the previous round, and there is no transmission in progress from a non-leader agent, the leader executes a selection operation. The selected agent sets its `processed` bit and transmits its input if its `processed` bit is not already set. If the `processed` bit is set, the agent transmits nothing in the following two rounds.

Rounds 5 and 6: These are used to transmit either (a) one bit of a selected agent's input, or (b) one bit of the protocol output. In either case the bit is encoded as two bits using the convention $01 = 0$, $10 = 1$, $00 =$ `stop`. Note that the absence of a broadcast in both rounds is interpreted as `stop`, which both allows a selected agent to signal it has already been processed and guarantees eventual termination after an agent finishes transmitting its input even if some of the broadcasts are garbled.

It is possible for two agents to be transmitting simultaneously (this can occur if there are multiple surviving leaders). This requires that agents be prepared to handle receiving 11. The simplest way to handle 11 may be to have agents just interpret it as a fixed value: $11 = 1$. Alternatively, we could implement an optimization where any agent that observes 11 triggers a restart of the protocol by broadcasting a 1 in the next Round 2.

The protocol terminates when the leader has collected all inputs (detected by the absence of a signal in Round 3) and transmits the computed output to all agents (using Rounds 5 and 6 over however many iterations are needed). We assume that the computed output has finite length for any combination of inputs. After transmitting the output, the leader broadcasts a 1 in Round 2 to restart the information-gathering component of the protocol.

Lemma 5.6.4. *In any execution with finite errors in the underlying broadcast protocol, with probability* 1*, the above protocol converges to a single leader and then restarts infinitely often.*

Lemma 5.6.4 is proven correct by demonstrating that, as defined, this protocol elects exactly one leader by Round 2 which correctly processes all non-leader agents with probability 1 by the end of Round 6. The full proof can be found in 5.6.4

5.6.4. Proof of Lemma 5.6.4.

Proof. Consider a sequence of iterations in which no errors occur.

If there are no leaders initially, the first execution of Round 0 sets the `leader` bit in all agents. The only way that an agent can lose its `leader` bit is if it sees another leader broadcast a 1 in Round 1. But this always leaves at least one leader. If there is more than one leader, half the remaining leaders on average will drop out in each execution of Round 1. This guarantees that there will eventually be exactly one leader with probability 1.

If there is a leader, the leader believes that there is no transmission in progress, and at least one agent v with $v.\texttt{processed} = \mathsf{False}$, then v is selected with probability at least $\frac{1}{2n}$ for sufficiently large r in Round 3 (Lemma 5.6.3). This causes some agent to be selected in Round 3 eventually with probability 1, reducing the number of unprocessed agents by one.

If there is a transmission in progress, each transmitting agent sends finitely many bits before stopping. Once all transmitting agents have stopped, any agent waiting for a transmission to finish will observe 00 in Rounds 5 and 6.

It follows that starting from an arbitrary initial configuration, with probability 1 the protocol reaches a configuration with exactly one leader, the leader finishes waiting for any outstanding transmissions, and the leader then selects an unprocessed agent and collects its input until no unprocessed agents are left. After the leader transmits its computed (though possibly incorrect) output, the protocol restarts. □

Once the protocol restarts with a single leader, any subsequent error-free execution produces correct output. This follows from the fact that the leader collects the input from every agent exactly once. Since each agent records as its output the last output broadcast by the leader, this causes all agents to converge to holding the correct output with probability 1.

Because the leader has unbounded states, it can simulate an arbitrary Turing machine. This allows the output to converge to the value of any computable symmetric function. The restriction to symmetric functions follows from uniformity of the agents in the initial configuration, but can be overcome, assuming inputs include indexes. We have thus shown:

THEOREM 5.6.5. *For any computable symmetric function f, there is a population protocol using* 1*-bit messages and unbounded internal states that starts in an initial configuration where each agent i is distinguished only by its input x_i, that converges to having each agent holding output $f(x_1, \dots, x_n)$.*

Our construction exploits the unbounded state at each agent to allow the leader to simulate the entire computation. While the probability-1 convergence property requires unbounded state in the limit (otherwise there is a nonzero probability that any round fails), it may be desirable to put off expanding the state as long as possible. In 5.6.5, we argue that with some small tweaks, the construction can be adapted to distribute the contents of a Turing machine tape of s bits across all agents of the population as in [**62**], reducing the storage overhead at each agent for the Turing machine computation to $O(s/n + \log s)$ bits.

5.6.5. Simulating a Turing machine. In this section, we show how to adapt the construction of Section 5.6.3 to simulate a Turing machine directly. For the most part, we retain the round structure of the previous construction, but make some adjustments to how the leader interacts with the other agents.

As in the construction in Section 5.6.3, we elect a leader using Rounds 0 and 1, which resets the other agents by broadcasting in Round 2. Non-leader agents reset to an **unallocated** state in which they hold only their input while waiting to be recruited to hold tape cells; such agents will set `processed` to False until recruited to hold a tape cell. The leader agent holds the state of the finite-state controller and the index for the current head position and manages communication with the other agents through Round 5 and 6 broadcasts. Using the same self-delimiting encoding

Protocol 5.5 TM-Simulation-Leader

```
$h \leftarrow 0$                                         ▷ initial head position
$q \leftarrow q_0$                                       ▷ initial state
$n \leftarrow 1$                                         ▷ count of agents
counted ← False                                          ▷ indicates if $n$ is correct
Restart computation by sending 1 in Round 2
while counted = False do                                 ▷ Initialize tape
    if at least one agent reported processed = False in Round 3 then
        Select an agent in Round 4
        Transmit Recruit($n+1$)
        if some agent responds then
            $n \leftarrow n+1$
    else
        Transmit PopulationSize($n$)                     ▷ No agents remaining!
        counted ← True
        $s \leftarrow f(n)$                              ▷ Initialize runtime bound
for each Turing machine step do
    Read cell at current head position $h$ by transmitting Read($h$)
    if some agent responds with $c$ then
        $(q', c', d) \leftarrow \delta(q, c)$
        Write $c'$ to cell $h$ by transmitting Write($h, c'$)
        $q \leftarrow q'$
        $h \leftarrow h + d$
        $s \leftarrow s - 1$
        if $q$ is a halting state then
            Transmit result of computation
            Jump to start of algorithm
        else if $s = 0$ then
            Jump to start of algorithm                   ▷ Runtime bound exceeded
    else
        Jump to start of algorithm                       ▷ No agent holds $h$ ⇒ initialization failed
```

as before allows transmission of messages of arbitrary length, so long as all agents agree on which agent's turn it is to speak.

A very high level overview of the simulation is given in Algorithms 5.5 and 5.6. Algorithm 5.5 is written from the perspective of the leader, and assumes that we have already elected a unique leader and reset all the other agents. The function $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{-1, 0, +1\}$ is the transition function for the simulated Turing machine. Algorithm 5.6 is written from the perspective of a non-leader and describes how it responds to transmissions from the leader.

The simulation starts by organizing the agents into a Turing machine tape. This involves selecting agents one at a time and assigning them indices. Because an agent might be selected more than once, the expected number of rounds to find all agents scales as $O(n \log^2 n)$, where $O(n \log n)$

Protocol 5.6 TM-Simulation-Follower

```
if receiving Restart then
    processed ← False                                   ▷ I am unallocated
    $j \leftarrow \bot$                                 ▷ $j$ indexes which cells I hold
    $n \leftarrow \infty$    ▷ $n$ is initially unknown. We use the convention that $x \bmod \infty = x$ and $x/\infty = 0$
    Clear list $T[]$         ▷ $T[i]$ holds cell $(n-1)i + c$. Unset locations default to blank
    Set $T[0]$ to my input
if receiving Read($i$) then
    if $i \bmod (n-1) = j$ then
        Transmit $T[\lfloor i/n \rfloor]$
if receiving Write($i, c$) then
    if $i \bmod (n-1) = j$ then
        $T[\lfloor i/n \rfloor] \leftarrow c$
if receiving Recruit($h$) then
    if I have been selected then
        if $j = \bot$ then
            $j = h$
            Transmit acknowledgment
    Clear selected status
if receiving PopulationSize($m$) then
    $n \leftarrow m$
if receiving result of computation then
    Record result as output
```

comes from the expected time to finish a coupon collector process and the extra $O(\log n)$ comes from the time to transmit indexes one bit at a time. We assume that counting n is also enough for the leader to compute a bound $f(n)$ on the number of steps used by the Turing machine; this is needed to enforce restarts if the simulated machine does not terminate on its own.

For simplicity we assume that inputs can be placed in arbitrary order on the first $n-1$ cells of the tape (this will require special handling of any input on the leader, which we omit for simplicity of presentation). A complication is that inputs to the protocol might exceed the size of the constant tape alphabet. This does not affect the simulation directly, since no restriction on tape alphabet is assumed, but it may require adding a preamble to the Turing machine computation that unpacks large-alphabet inputs into the constant-size TM alphabet. We leave the details of this tedious and unenlightening preamble to the imagine of the reader.

The analysis of Algorithms 5.5 and 5.6 essentially follows the proof of Theorem 5.6.5. Once the simulation reaches the safe phase of the construction, it reaches a configuration with one leader after some finite time with probability 1. At this point the leader may already have an inaccurate estimate

$\hat{n}$ of n, but whether the estimate is accurate or not, each iteration of the main loop will require at most $O(\log f(\hat{n}))$ rounds to finish, leading to a restart after at most $O(n \log^2 n + f(\hat{n}) \log f(\hat{n}))$ rounds on average. Each subsequent iteration will run the Turing machine to completion and produce the correct output. The space complexity at each agent, measured in bits, is bounded by $O(s/n + \log s)$ during non-faulty simulations, where s is the largest tape cell index used. For faulty executions, we accept a small probability that a larger estimate of $\hat{n}$ at some leader agent may leader to larger space overhead. In either case the state complexity is dominated in the limit by the unbounded round and tick counters.

THEOREM 5.6.6. *Algorithms 5.5 and 5.6 use the synchronous broadcast primitive to simulate a Turing machine with known time complexity $f(n)$, converging to the correct output with probability* 1*. In any execution, the additional space required at each agent to simulate a Turing machine that uses s tape cells is bounded after an initial prefix by $O(s/n + \log s)$ with probability* 1*.*

5.7. Open problems

Probability-1 computation. Many of our protocols have a positive probability of error. Common techniques for achieving zero error probability in $\omega(1)$-state protocols require $\omega(1)$ messages. Based on this, we conjecture that probability-1 leader election using $O(1)$ messages requires $\Omega(n)$ time to stabilize. This is known to hold for $O(1)$ states [88], though sublinear-time **convergence** is possible with $O(1)$ states [120].

Time lower bounds. A tool for time lower bounds (e.g., probability-1 leader election [5, 88]) is a "density lemma" [5, 82] showing that when the state complexity is $\leq \frac{1}{2} \log \log n$, all states appear in "large" count. This is false for $s(n) > \log \log n$, which is the key to the fastest space-optimal leader election protocols [40, 99, 100]. A density lemma applies to the **messages** of $O(1)$-message protocols, no matter the state complexity (derivable from [83, Lemma 4.2]). Does this imply that $O(1)$-message leader election requires linear stabilization time?

Power of 1-bit messages with $O(1)$-states. $O(1)$-state **open** protocols can stably compute exactly the semilinear predicates [20]. Can all semilinear predicates be stably computed with 1-bit messages? A related question is whether there is a direct simulation of $O(1)$-message protocols by 1-bit message protocols (similar to Theorem 5.3.3).

Efficient predicate computation. Corollary 5.5.11 can be used to efficiently compute any computable predicate $\phi : \mathbb{N}^d \to \{0, 1\}$ but requires storing the entire initial configuration locally in each agent ($\Theta(n^d)$ states). Corollary 5.3.6 can be used to compute any computable predicate storing unique IDs in each agent ($O(n)$ states), but it is slow since communication is routed through a leader. What predicates can be computed time- **and** space-efficiently?

CHAPTER 6

Composable Computation in Discrete Chemical Reaction Networks

This chapter is joint work with David Doty and David Haley. It was originally published as [146].

6.1. Introduction

6.1.1. Function computation. Computation of functions $f : \mathbb{N}^d \to \mathbb{N}$ was discussed briefly in the first population protocols paper [19, Section 3.4], which focused more on Boolean predicate computation, and it was defined formally first for CRNs [63, 86] and later for population protocols [32]. The class of functions stably computable in either model is the same: the semilinear functions [23, 63]. We use the CRN model because it is more natural for describing functions, but our results also apply to the population protocol model.

To represent an input $\mathbf{x} \in \mathbb{N}^d$, we start in a configuration with counts $\mathbf{x}(i)$ of species X_i for each $i \in \{1, \ldots, d\}$, and count 1 of a "leader" species L.[1] A function $f : \mathbb{N}^d \to \mathbb{N}$ is **stably computable** by a CRN if a correct and **stable** configuration $\mathbf{O}$ (i.e., on input $\mathbf{x}$ the count of Y is $f(\mathbf{x})$ in all configurations reachable from $\mathbf{O}$) remains reachable no matter the order in which reactions occur.[2]

[1] The leader is discussed in Section 6.1.3. A CRN may ignore its leader, as in Fig. 6.2 (left side), or include it, as in Fig. 6.2 (right side).

[2] We use this definition of stably computable throughout this chapter, but we mention here that it is equivalent to two other natural definitions. The first definition is that any fair sequence of reactions **will** take the CRN to such a correct stable configuration, where **fair** means that any configuration that is infinitely often reachable is eventually reached. The second definition is that a correct stable configuration is actually reached with probability 1.

$$\boxed{\begin{array}{c} f(x) = 2x \\ \hline X \to 2Y \end{array}} \quad \boxed{\begin{array}{c} f(x_1, x_2) = \min(x_1, x_2) \\ \hline X_1 + X_2 \to Y \end{array}}$$

$$\boxed{\begin{array}{rcl} \multicolumn{3}{c}{f(x_1, x_2) = \max(x_1, x_2)} \\ \hline X_1 & \to & Z_1 + Y \\ X_2 & \to & Z_2 + Y \\ Z_1 + Z_2 & \to & K \\ K + Y & \to & \emptyset \end{array}}$$

FIGURE 6.1. Functions stably computed by CRNs. Note max is computed as $x_1 + x_2 - \min(x_1, x_2)$.

See Fig. 6.1 for examples. It is known that a function $f : \mathbb{N}^d \to \mathbb{N}$ is stably computable by a CRN if and only if it is **semilinear**: intuitively, it is a piecewise affine function. (See Definition 6.2.6.)

6.1.2. Composability. Note a key difference between the CRNs for min and max in Fig. 6.1: the former produces, but does not consume, the output species Y, whereas the latter also contains reactions that consume Y. In one possible sequence of reactions for the max CRN, the inputs can be exhausted through the first two reactions before ever executing the last two reactions. In doing so, the count of Y overshoots its correct value of $\max(x_1, x_2)$ before the excess is consumed by the reaction $K + Y \to \emptyset$.

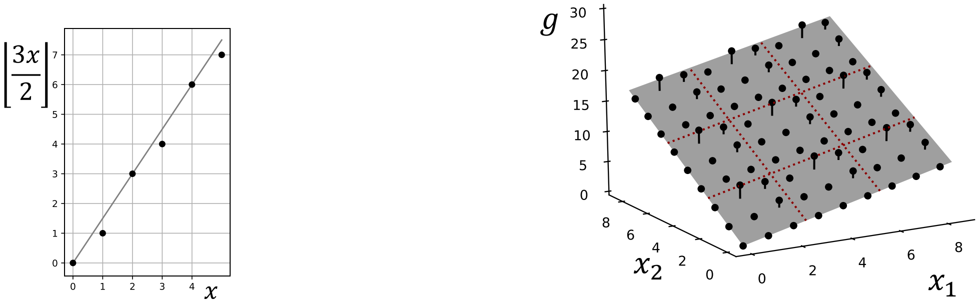
For this reason, the min CRN is more easily composed with a downstream CRN. For example, the function $2 \cdot \min(x_1, x_2)$ is stably computed by the reactions $X_1 + X_2 \to W$ (computing $w = x_1 + x_2$) and $W \to 2Y$ (computing $y = 2w$), renaming the output of the min CRN to match the input of the multiply-by-2 CRN. However, this approach does not work to compute $2 \cdot \max(x_1, x_2)$; changing Y to W in the four-reaction max CRN and adding the reaction $W \to 2Y$ can erroneously result in up to $2(x_1 + x_2)$ copies of Y being produced. Intuitively, the multiply-by-2 reaction $W \to 2Y$ competes with the upstream reaction $K + W \to \emptyset$ from the max CRN.

This motivates us to study the class of functions $f : \mathbb{N}^d \to \mathbb{N}$ stably computable by **output-oblivious** CRNs: those in which the output species Y appears only as a product, never as a reactant. We call such a function **obliviously-computable**. Any obliviously-computable function must be nondecreasing, otherwise reactions could incorrectly overproduce output (see Observation 6.2.1).

An obliviously-computable function, being a special kind of stably computable function, must be semilinear [**20**, **63**], so it is reasonable to conjecture that a function is obliviously-computable if and only if it is semilinear and nondecreasing. In fact, this is true for 1D functions $f : \mathbb{N} \to \mathbb{N}$ (see Section 6.3). However, in higher dimensions, the function $\max : \mathbb{N}^2 \to \mathbb{N}$ is semilinear and nondecreasing, yet not obliviously-computable; its consumption of output turns out to be unavoidable. Assuming there is no leader, this is simple to prove: Since $\max(1, 0) = 1$, starting with one X_1, a Y can be produced. Similarly, a Y can be produced starting with one X_2. Then with one X_1 and one X_2, these reactions can happen in parallel and produce two Y's, too many

$$\boxed{\begin{array}{c} f(x) = \min(1,x) \\ \hline X \to Y \\ 2Y \to Y \end{array}} \quad \boxed{\begin{array}{c} f(x) = \min(1,x) \\ \hline L + X \to Y \end{array}}$$

FIGURE 6.2. $\min(1,x)$ is stably computed by a **leaderless** non-output-oblivious CRN (left), and an output-oblivious CRN with a single leader L (right).



(a) A 1D (single-input) quilt-affine function $\lfloor \frac{3x}{2} \rfloor = \frac{3}{2}x + B(\overline{x} \mod 2)$, where $B(\overline{0}) = 0$ and $B(\overline{1}) = -\frac{1}{2}$.

(b) A 2D quilt-affine function $g(\mathbf{x}) = (1,2) \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod 3)$, where $B(\overline{\mathbf{x}}) = 0$ except for $B(\overline{(1,2)}) = 2$ and $B(\overline{(2,1)}) = B(\overline{(2,2)}) = 1$.

FIGURE 6.3. Examples of 1D and 2D quilt-affine functions.

since $\max(1,1) = 1$. It is more involved to prove that even with a leader, it remains impossible to obliviously compute max; see Section 6.4.[3]

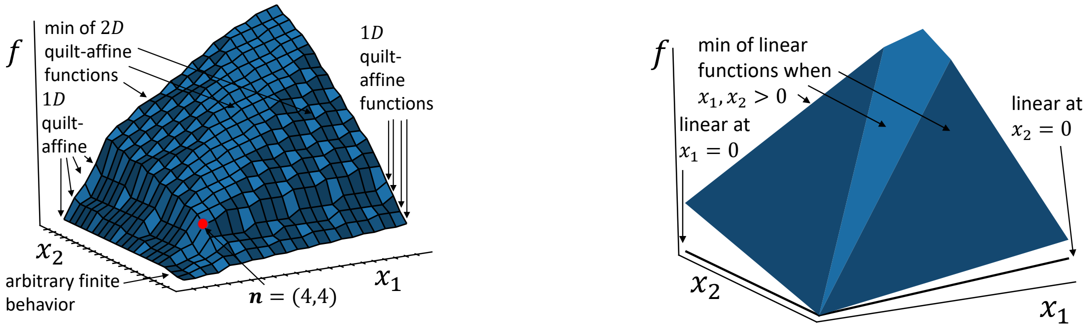
6.1.3. The role of the leader.

The class of stably computable functions is identical whether an initial leader is allowed or not [86], as is the class of stably computable predicates [23]. Our model includes an initial leader, which is essential for our general constructions (see Sections 6.3 and 6.6).

In fact, the class of obliviously-computable functions we study is provably larger when an initial leader is allowed. For example, consider the function $f(x) = \min(1,x)$ (see Fig. 6.2). f is stably computable with or without a leader, but only the construction with a leader is output-oblivious. Without using a leader, f is not obliviously-computable (see Observation 6.9.1).

Including the leader gives additional power to the model. This gives more power to our CRN constructions, but makes our impossibility results stronger. This work left fully classifying the obliviously-computable functions in a leaderless model as an open question, which has since been resolved in [111].

[3] This result was obtained independently by Chugg, Condon, and Hashemi [68].



(a) A 2D function satisfying Theorem 6.5.2.

(b) The scaling limit gives a 2D real-valued obliviously-computable function from [61].

FIGURE 6.4. Example discrete and real-valued obliviously-computable functions.

6.1.4. Contribution. Our main result, Theorem 6.5.2, provides a complete characterization of the class of obliviously-computable functions. It builds off of a key definition: a **quilt-affine** function is a nondecreasing function that is the sum of a rational linear function and periodic function (formalized as Definition 6.5.1). For example, functions such as $\lfloor \frac{3x}{2} \rfloor$ are quilt-affine (see Fig. 6.3a). Such floored division functions are natural to the discrete CRN model ($\lfloor \frac{3x}{2} \rfloor$ is stably computed by reactions $X \to 3Z$, $2Z \to Y$). Fig. 6.3b shows a higher-dimensional quilt-affine function, with a "bumpy quilt" structure that motivates the name. Quilt-affine functions are also characterized by nonnegative periodic finite differences, a structure key to showing they are obliviously-computable.

Theorem 6.5.2 states that a function $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable if and only if

i) [nondecreasing] f is nondecreasing,

ii) [eventually-min] for sufficiently large inputs, f is the minimum of a finite number of quilt-affine functions, and

iii) [recursive] every restriction $f_r : \mathbb{N}^{d-1} \to \mathbb{N}$ obtained by fixing an input to a constant value[4] is obliviously-computable (i.e., eventually the minimum of quilt-affine functions).

Condition ii) characterizes f when all inputs are sufficiently large (greater than some $\mathbf{n} \in \mathbb{N}^d$), whereas condition iii) characterizes f when some inputs are fixed to smaller values. See Fig. 6.4a for a representative example of an obliviously-computable $f : \mathbb{N}^2 \to \mathbb{N}$. This pictured function has arbitrary nondecreasing values in the "finite region" where $\mathbf{x} < (4, 4)$, has eventual 1D quilt-affine

[4] Note that Theorem 6.5.2 defines fixed-input restrictions slightly differently; see Section 6.5 for an explanation.

behavior along the lines $x_1 = 0, 1, 2, 3$ and $x_2 = 0, 1, 2, 3$, and is the minimum of 3 different quilt-affine functions in the "eventual region" where $\mathbf{x} \geq (4, 4)$. This behavior generalizes naturally to higher dimensions.

The most technically sophisticated part of our result is the proof that the eventually-min condition ii) of Theorem 6.5.2 is necessary; the main ideas of this proof are outlined in Section 6.7.1.

6.1.5. Related work. Chalk, Kornerup, Reeves, and Soloveichik [**61**] showed an analogous result in the **continuous** model of CRNs in which species amounts are given by nonnegative real concentrations. A consequence of their characterization is that any obliviously-computable real-valued function is a minimum of linear functions when all inputs are positive. In Theorem 6.8.2, we demonstrate that the limit of "scalings" of a function $f : \mathbb{N}^d \to \mathbb{N}$ satisfying our main Theorem 6.5.2 is in fact a function $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ satisfying the main theorem of [**61**] (see Fig. 6.4b). The discrete details lost in the scaling limit constitute precisely the unique challenges of proving Theorem 6.5.2 that are not handled by [**61**]. In particular, our function class can contain arbitrary finite behavior and repeated finite irregularities.

Returning to the **discrete** (a.k.a., **stochastic**) CRN model we study, Chugg, Condon, and Hashemi [**68**] independently investigated the special case of **two-input** functions $f : \mathbb{N}^2 \to \mathbb{N}$ computable by output-oblivious CRNs, obtaining a characterization equivalent to ours when restricted to 2D. In a followup result [**111**], they extended the results of this work to give a characterization of the leaderless case.

6.1.6. Other ways of composing computation. In Section 6.2.3 we show that for a CRN $\mathcal{C}$ be composable with downstream CRN $\mathcal{D}$ by "concatenation" (renaming $\mathcal{C}$'s output species to match $\mathcal{D}$'s input species and ensuring all other species names are disjoint between $\mathcal{C}$ and $\mathcal{D}$), it is (in a sense) necessary and sufficient for $\mathcal{C}$ to be output-oblivious. There are other ways to compose computations, however.

A common technique (e.g. [**99**]) is for $\mathcal{C}$ to detect when its output has changed and send a **restart** signal to $\mathcal{D}$. However, it is not obvious how to do this with function computation as defined in this paper, where $\mathcal{D}$ changes $\mathcal{C}$'s output by consuming it.

Another technique (e.g. [**21**]) is to set a **termination signal**, which is a sub-CRN that, with high probability, creates a copy of a signal species T, but not before $\mathcal{C}$ has converged. T then "activates" the reactions of $\mathcal{D}$, so that $\mathcal{D}$ will not consume the output of $\mathcal{C}$ until it is safe to do so. However, this has some positive failure probability. In fact, if we require T to be guaranteed with probability 1 to be produced only after the CRN has converged, only constant functions can be stably computed [70]. Worse yet, in the leaderless case, it is provably impossible to achieve this guarantee even with positive probability [83].

6.2. Preliminaries

6.2.1. Notation. For a set $\mathcal{S}$ (of species), we write $\mathbb{N}^{\mathcal{S}}$ to denote the set of vectors indexed by the elements of $\mathcal{S}$ (equivalently, functions $f : \mathcal{S} \to \mathbb{N}$). Vectors appear in boldface, and we reserve uppercase $\mathbf{A} \in \mathbb{N}^{\mathcal{S}}$ for such vectors indexed by species, and lowercase $\mathbf{a} \in \mathbb{N}^d, \mathbb{Z}^d, \mathbb{Q}^d, \mathbb{R}^d$ for vectors indexed by integers. $\mathbf{A}(S)$ or $\mathbf{a}(i)$ denotes the element indexed by $S \in \mathcal{S}$ or $i \in \{1, \ldots, d\}$. We write $\mathbf{a} \leq \mathbf{b}$ to denote pointwise vector inequality $\mathbf{a}(i) \leq \mathbf{b}(i)$ for all i.

For $p \in \mathbb{N}_+$, $\mathbb{Z}/p\mathbb{Z}$ denotes the additive group of integers modulo p, whose elements are congruence classes. Generalizing to higher dimensions, $\mathbb{Z}^d/p\mathbb{Z}^d$ denotes the additive group of $\mathbb{Z}^d$ modulo p, whose elements are congruence classes. For $\mathbf{x} \in \mathbb{N}^d$ where $d \geq 1$, we write $\overline{\mathbf{x}} \mod p$ to denote the congruence class $\{\mathbf{x} + p\mathbf{z} : \mathbf{z} \in \mathbb{Z}^d\} \in \mathbb{Z}^d/p\mathbb{Z}^d$, also denoted $\overline{\mathbf{x}}$ when p is clear from context.

$\mathbb{R}^d_{\geq 0}$ denotes the nonnegative orthant in $\mathbb{R}^d$. We consider **regions** $R = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : T\mathbf{x} - \mathbf{h} \geq 0\}$ which are convex polyhedra given by a set of inequalities. $R \cap \mathbb{N}^d$ denotes all integer points in R , and for $\mathbf{x} \in \mathbb{N}^d$, $R \cap (\overline{\mathbf{x}} \mod p)$ denotes the integer points in R in the same congruence class as $\mathbf{x}$.

$\|\mathbf{x}\|$ denotes the standard Euclidean norm. $B_r(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{y}\| < r\}$ denotes the open ball of radius r centered at $\mathbf{x}$.

6.2.2. Chemical reaction networks. We use the established definitions of stable function computation by (discrete) chemical reaction networks [20, 68]:

A **chemical reaction network** (CRN) $\mathcal{C} = (\mathcal{S}, \mathcal{R})$ is defined by a finite set $\mathcal{S}$ of species and a finite set $\mathcal{R}$ of reactions, where a reaction $(\mathbf{R}, \mathbf{P}) \in \mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}$ describes the counts of consumed **reactant** species and produced **product** species.[5] For example, given $\mathcal{S} = \{A, B, C\}$, the reaction $((1,0,2),(0,2,1))$ would represent $A + 2C \to 2B + C$.

[5] We do not limit ourselves to **bimolecular** (two input) reactions, but the higher-order reactions we use can easily be converted to have this form. For example, $3X \to Y$ is equivalent in its reachability to two reactions $2X \rightleftharpoons X_2$ and $X + X_2 \to Y$.

A configuration $\mathbf{C} \in \mathbb{N}^{\mathcal{S}}$ specifies the integer counts of all species. Reaction $(\mathbf{R}, \mathbf{P})$ is **applicable** to $\mathbf{C}$ if $\mathbf{R} \leq \mathbf{C}$, and yields $\mathbf{C}' = \mathbf{C} - \mathbf{R} + \mathbf{P}$, so we write $\mathbf{C} \to \mathbf{C}'$. A configuration $\mathbf{D}$ is **reachable** from $\mathbf{C}$ if there exists a finite sequence of configurations such that $\mathbf{C} \to \mathbf{C}_1 \to \ldots \to \mathbf{C}_n \to \mathbf{D}$; we write $\mathbf{C} \to^* \mathbf{D}$ to denote that $\mathbf{D}$ is reachable from $\mathbf{C}$. Note this reachability relation is additive: if $\mathbf{A} \to^* \mathbf{B}$, then $\mathbf{A} + \mathbf{C} \to^* \mathbf{B} + \mathbf{C}$. This property is key in future proofs to show the reachability of configurations which overproduce output.

To compute a function[6] $f : \mathbb{N}^d \to \mathbb{N}$, the CRN $\mathcal{C}$ will include an ordered subset $\{X_1, \ldots, X_d\} \subset \mathcal{S}$ of **input species**, an **output species** Y, and a **leader species** $L \in \mathcal{S}$. (Note that we consider removing the leader in Section 6.9).

The computation of $f(\mathbf{x})$ will start from an **initial configuration** $\mathbf{I}_{\mathbf{x}}$ encoding the input with $\mathbf{I}_{\mathbf{x}}(X_i) = \mathbf{x}(i)$ for all $i = 1, \ldots, d$, along with a single leader $\mathbf{I}_{\mathbf{x}}(L) = 1$, and count 0 of all other species. A **stable** configuration $\mathbf{C}$ has unchanged output $\mathbf{C}(Y) = \mathbf{D}(Y)$ for any configuration $\mathbf{D}$ reachable from $\mathbf{C}$. The CRN $\mathcal{C}$ **stably computes** $f : \mathbb{N}^d \to \mathbb{N}$ if for each initial configuration $\mathbf{I}_{\mathbf{x}}$ encoding any $\mathbf{x} \in \mathbb{N}^d$, and configuration $\mathbf{C}$ reachable from $\mathbf{I}_{\mathbf{x}}$, there is a stable configuration $\mathbf{O}$ reachable from $\mathbf{C}$ with correct output $\mathbf{O}(Y) = f(\mathbf{x})$.

6.2.3. Composition via output-oblivious CRNs.

This section formally defines our notions of "composable computation with CRNs via concatenation of reactions" and "output-oblivious" CRNs that don't consume their output, showing these notions to be essentially equivalent.

A CRN is **output-oblivious** if the output species Y is never a reactant[7]: for any reaction $(\mathbf{R}, \mathbf{P})$, $\mathbf{R}(Y) = 0$. A function $f : \mathbb{N}^d \to \mathbb{N}$ is **obliviously-computable** if f is stably computed by an output-oblivious CRN.

We begin with an easy observation:

Observation 6.2.1. *An obliviously-computable function $f : \mathbb{N}^d \to \mathbb{N}$ must be nondecreasing.*

Proof. Assume a CRN $\mathcal{C}$ (with output species Y) stably computes f, but $f(\mathbf{a}) > f(\mathbf{b})$ for $\mathbf{a} \leq \mathbf{b}$. To stably compute $f(\mathbf{a})$, input configuration $\mathbf{I}_{\mathbf{a}} \to^* \mathbf{O}$ for some configuration with $\mathbf{O}(Y) = f(\mathbf{a})$. However, since $\mathbf{a} \leq \mathbf{b}$, that same sequence of reactions can be applied from the input configuration

[6] We consider codomain $\mathbb{N}$ without loss of generality, since $f : \mathbb{N}^d \to \mathbb{N}^l$ is stably computable if and only if each output component is stably computable by parallel CRNs.

[7] A more general definition in [68] of **output-monotonic** CRNs just requires no reaction to reduce the count of output species. This can be directly seen to classify the same set of functions, see Observation 6.2.4.

$\mathbf{I_b} \geq \mathbf{I_a}$. This overproduces Y since $f(\mathbf{a}) > f(\mathbf{b})$. Thus to stably compute $f(\mathbf{b})$, some reaction must consume Y as a reactant, so $\mathcal{C}$ cannot be output-oblivious. □

A CRN being output-oblivious was shown in [**61**] (for **continuous** CRNs) to be equivalent to being "composable via concatenation", meaning renaming the output species of one CRN to match the input of another. This equivalence still holds in our discrete CRN model. This is formalized as Observation 6.2.2 and Lemma 6.2.3.

For CRNs $\mathcal{C}_f$ stably computing $f : \mathbb{N}^d \to \mathbb{N}$ and $\mathcal{C}_g$ stably computing $g : \mathbb{N} \to \mathbb{N}$, define the **concatenated** CRN $\mathcal{C}_{g \circ f}$ by combining species and reactions, with $\mathcal{C}_f$'s output species as $\mathcal{C}_g$'s input species and no other common species, plus a reaction $L \to L^f + L^g$ creating copies of the leader available to each of $\mathcal{C}_f$ and $\mathcal{C}_g$.

We first observe that this composition works correctly if the **upstream** CRN $\mathcal{C}_f$ is output-oblivious. Intuitively, the reactions from $\mathcal{C}_g$ can only affect the reactions from $\mathcal{C}_f$ via the common species W, but this output species of $\mathcal{C}_f$ is never used as a reactant to stably compute $f(\mathbf{x})$.

Observation 6.2.2. *If $\mathcal{C}_f$ stably computes $f : \mathbb{N}^d \to \mathbb{N}$, $\mathcal{C}_g$ stably computes $g : \mathbb{N} \to \mathbb{N}$, and $\mathcal{C}_f$ is output-oblivious, then the concatenated CRN $\mathcal{C}_{g \circ f}$ stably computes the composition $g \circ f : \mathbb{N}^d \to \mathbb{N}$.*

Note that the downstream CRN $\mathcal{C}_g$ need not be output-oblivious, but if two output-oblivious CRNs are composed, then the composition $\mathcal{C}_{g \circ f}$ remains output-oblivious. More generally, g can take any number of inputs from output-oblivious CRNs, which act as modules for arbitrary feedforward composition.

The converse shows that a composable CRN is essentially output-oblivious. If $\mathcal{C}_f$ can be correctly composed with any downstream $\mathcal{C}_g$, then $\mathcal{C}_f$ must function correctly even if downstream reactions from $\mathcal{C}_g$ starve it of the common species W. Thus $\mathcal{C}_f$ will still stably compute f if we remove all reactions with output W as a reactant, making it output-oblivious.

Lemma 6.2.3. *Let $\mathcal{C}_f$ stably compute $f : \mathbb{N}^d \to \mathbb{N}$ such that for any $\mathcal{C}_g$ stably computing $g : \mathbb{N} \to \mathbb{N}$, the concatenated CRN $\mathcal{C}_{g \circ f}$ stably computes the composition $g \circ f : \mathbb{N}^d \to \mathbb{N}$. Let $\mathcal{C}'_f$ be the output-oblivious CRN removing any reactions using the output species as a reactant from $\mathcal{C}_f$. Then $\mathcal{C}'_f$ stably computes f.*

Proof. Let C_f (with output species W) stably compute $f : \mathbb{N}^d \to \mathbb{N}$. Let $g(x) = x$ be the identity function, stably computed by $W \to Y$. Assume the concatenated CRN $\mathcal{C}_{g \circ f}$ stably computes $g \circ f = f$. Let $\mathcal{C}'_f$ be the output-oblivious CRN with all reactions using W as a reactant removed from $\mathcal{C}_f$. We now show that $\mathcal{C}'_f$ also stably computes f.

For any $\mathbf{x} \in \mathbb{N}^d$, let $\mathbf{I_x}$ be the initial configuration encoding $\mathbf{x}$ in $\mathcal{C}'_f$, and $\mathbf{C}$ be any configuration reachable from $\mathbf{I_x}$. Now we will naturally view $\mathbf{I_x}$ and $\mathbf{C}$ also as configurations in the concatenated CRN $\mathcal{C}_{g\circ f}$. We first consider configuration $\mathbf{D}$ reachable from $\mathbf{C}$ by applying the reaction $W \to Y$ $\mathbf{C}(W)$ times, so $\mathbf{D}(W) = 0$. Now because $\mathcal{C}_{g\circ f}$ stably computes f, there exists a sequence of reactions α from $\mathbf{D}$ to a stable configuration $\mathbf{O}$ with $\mathbf{O}(Y) = f(\mathbf{x})$. α could contain reactions that use W as a reactant, but because $\mathbf{D}(W) = 0$, any such reactions must occur after additional W has been produced.

Before any such reactions using W as a reactant, we can insert more reactions $W \to Y$, reaching an intermediate configuration $\mathbf{D}_2$ again with $\mathbf{D}_2(W) = 0$. There must exist some new sequence of reactions α_2 from $\mathbf{D}_2$ to a stable configuration $\mathbf{O}_2$ with $\mathbf{O}_2(Y) = f(\mathbf{x})$. Repeating this process, we can construct a sequence of reactions β from $\mathbf{C}$ to a correct stable configuration $\mathbf{O}_n$, where β does not contain any reactions with W as a reactant. Notice that we only have to "splice in" new reactions when W is produced, and this can only happen at most $f(\mathbf{x})$ times, so this process will terminate.

Now $\mathbf{O}_n(Y) = f(\mathbf{x})$, so β contains precisely $f(\mathbf{x})$ copies of the reaction $W \to Y$. Ignoring these reactions then gives a sequence of reactions β' in $\mathcal{C}'_f$ from $\mathbf{C}$ to a correct configuration $\mathbf{O}'$ with $\mathbf{O}'(W) = f(\mathbf{x})$. Notice that $\mathbf{O}_n$ being stable in $\mathcal{C}_{g\circ f}$ implies $\mathbf{O}'$ is stable in $\mathcal{C}'_f$ since no additional W can be produced. This shows $\mathcal{C}'_f$ stably computes f as desired. □

We finally observe that the more general definition of **output-monotonic** CRNs (which cannot decrease the count of the output species) stably compute precisely the same set of functions as output-oblivious CRNs:

Observation 6.2.4. $f : \mathbb{N}^d \to \mathbb{N}$ *is stably computable by an output-oblivious CRN* $\iff$ f *is stably computable by an output-monotonic CRN.*

Proof. $\implies$: Any output-oblivious CRN must be output-monotonic.

$\impliedby$: If f was stably computed by an output-monotonic CRN $\mathcal{C}$ which is not already output-oblivious, then there must be reactions of the form $aY + \ldots \to aY + \ldots$ with $a \geq 1$ copies of output species Y acting as a catalyst. $\mathcal{C}$ can be made output-oblivious by replacing all such occurrences of Y as a catalyst by a new catalyst species Z that is always produced alongside Y. Since $\mathcal{C}$ was output-monotonic, if a Y is ever produced, it cannot be consumed. Thus any reactions with aY as a catalyst are "turned on" the moment a copies of Y are produced and never turn off again. So it does not change the reachable configurations to irreversibly produce a Z alongside Y and use Z as the catalyst in place of Y. This output-oblivious CRN thus also stably computes f. □

6.2.4. Semilinear functions.

The functions stably computable by a CRN were shown in [63], building from work in [20], to be precisely the **semilinear** functions, which are defined based on semilinear sets. The following definition is useful for the proofs in this work, see Section 1.3.1 for other definitions and discussion.

Definition 6.2.5. *A subset $S \subseteq \mathbb{N}^d$ is* ***semilinear*** *if S is a finite Boolean combination (union, intersection, complement) of* ***threshold*** *sets of the form $\{\mathbf{x} \in \mathbb{N}^d : \mathbf{a} \cdot \mathbf{x} \geq b\}$ for $\mathbf{a} \in \mathbb{Z}^d, b \in \mathbb{Z}$ and* ***mod*** *sets of the form $\{\mathbf{x} \in \mathbb{N}^d : \mathbf{a} \cdot \mathbf{x} \equiv b \mod c\}$ for $\mathbf{a} \in \mathbb{Z}^d, b \in \mathbb{Z}, c \in \mathbb{N}_+$.*

A **semilinear function** can be concisely defined as having a semilinear graph, but a more useful and equivalent definition follows from Lemma 4.3 in [63][8]:

Definition 6.2.6 ([63]). *A function $f : \mathbb{N}^d \to \mathbb{N}$ is* ***semilinear*** *if f is the finite union of affine partial functions, whose domains are disjoint semilinear subsets of $\mathbb{N}^d$.*

Thus we will view semilinear functions as piecewise affine, where a finite number of theshold and mod sets determine which affine function. All functions discussed so far have been semilinear. For example, the function

$$\min(x_1, x_2) = \begin{cases} x_1, & \text{if } x_1 \leq x_2 \\ x_2, & \text{if } x_1 > x_2 \end{cases}$$

is semilinear with affine partial functions on disjoint domains which are defined by a single threshold and thus semilinear.

Similarly, the function

$$\left\lfloor \frac{3x}{2} \right\rfloor = \begin{cases} \frac{3}{2}x, & \text{if } x \text{ is even} \\ \frac{3}{2}x - \frac{1}{2}, & \text{if } x \text{ is odd} \end{cases}$$

[8] Lemma 4.3 in [63] has domains that are non-disjoint linear sets. We assume the domains are disjoint for convenience, making the domains semilinear sets.

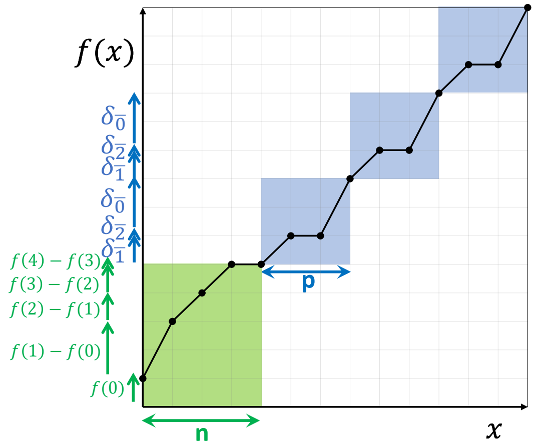


FIGURE 6.5. Every semilinear nondecreasing $f:\mathbb{N}\to\mathbb{N}$ is eventually quilt-affine, with periodic finite differences $\delta_{\overline{x}}$.

is semilinear with affine partial functions on disjoint domains which are defined by parity (a single mod predicate) and thus semilinear. All quilt-affine $f:\mathbb{N}^d\to\mathbb{N}$ (see Fig. 6.3b, Definition 6.5.1) are semilinear by the same argument.

Lemma 6.2.7 ([**63**]). *A function $f:\mathbb{N}^d\to\mathbb{N}$ is stably computable $\iff$ f is semilinear.*

6.3. Warm-up: One-dimensional case

For functions with one-dimensional input, the necessary conditions of being nondecreasing and semilinear are also sufficient.

THEOREM 6.3.1. *$f:\mathbb{N}\to\mathbb{N}$ is obliviously-computable $\iff$ f is semilinear and nondecreasing.*

Intuitively, the proof works as follows. We show semilinear, nondecreasing $f:\mathbb{N}\to\mathbb{N}$ must have the eventually quilt-affine structure in Fig. 6.5. From this structure, we define a CRN that uses auxiliary leader states to track the value of x (or $\overline{x} \mod p$ once $x\geq n$), while outputting the correct finite differences from adding each input.

PROOF. $\Longrightarrow$: Lemma 6.2.7 and Observation 6.2.1.

$\Longleftarrow$: Assume $f:\mathbb{N}\to\mathbb{N}$ is semilinear and nondecreasing. We will show that it is eventually **quilt-affine** (generalized to higher-dimensional functions as Definition 6.5.1), thus has periodic finite differences: for some $n\in\mathbb{N}$, period $p\in\mathbb{N}_+$, and finite differences $\delta_{\overline{0}},\ldots,\delta_{\overline{p-1}}\in\mathbb{N}$. This will imply that for all $x\geq n$, $f(x+1)-f(x)=\delta_{(\overline{x} \mod p)}$ (see Fig. 6.5).

Because f is semilinear, by Definitions 6.2.5 and 6.2.6, it can be represented as a disjoint union of affine partial functions, whose domains are semilinear sets, and thus represented as finite Boolean combinations of threshold $\{x \in \mathbb{N} : x \geq a\}$ and mod $\{x \in \mathbb{N} : x \equiv b \mod c\}$ sets. Now take $n \in \mathbb{N}$ greater than all such a and $p = \mathrm{lcm}(c)$ for all such c. Then for all $x \geq n$, f periodically cycles between affine partial functions. Because f is nondecreasing, these periodically-repeated affine partial functions must all have the same slope. This implies f is eventually quilt-affine, with periodic finite differences for all $x \geq n$ as claimed.

The CRN $\mathcal{C}$ to stably compute f uses input species X, output species Y, leader L, and species $L_0, \ldots, L_{n-1}, P_{\overline{0}}, \ldots, P_{\overline{p-1}}$ corresponding to auxiliary "states" of the leader, i.e., exactly one of $L, L_0, \ldots, L_{n-1}, P_{\overline{0}}, \ldots, P_{\overline{p-1}}$ is present at any time. Intuitively the leader tracks how many input X it has seen, where the count past n wraps around mod p, and outputs the correct finite differences. The reactions of $\mathcal{C}$ are as follows

$$\begin{aligned} L &\to f(0)Y + L_0 \\ L_i + X &\to [f(i+1) - f(i)]Y + L_{i+1} \qquad \text{for all } i = 0, \ldots, n-2 \\ L_{n-1} + X &\to [f(n) - f(n-1)]Y + P_{\overline{n}} \\ P_{\overline{a}} + X &\to \delta_{\overline{a}} Y + P_{\overline{a+1}} \qquad \text{for all } \overline{a} = \overline{0}, \ldots, \overline{p-1}. \end{aligned}$$

□

In the 1D case, we can also characterize the functions obliviously-computable **without** a leader: they are semilinear and **superadditive**: meaning $f(x)+f(y) \leq f(x+y)$ for all x, y. (Theorem 6.9.2)

6.4. Impossibility result

The characterization of obliviously-computable functions as precisely semilinear and nondecreasing from Theorem 6.3.1 is insufficient in higher dimensions. As an example, consider the function $\max : \mathbb{N}^2 \to \mathbb{N}$, which is both semilinear and nondecreasing. We prove max is not obliviously-computable via a more general lemma:

Lemma 6.4.1. *Let $f : \mathbb{N}^d \to \mathbb{N}$. If there exists an increasing sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in \mathbb{N}^d$ such that for all $i < j$ there exists some $\mathbf{\Delta}_{ij} \in \mathbb{N}^d$ with*

$$f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j),$$

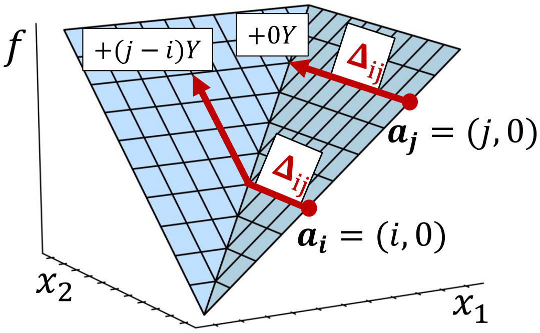


FIGURE 6.6. Lemma 6.4.1 applied to $f = \max(x_1, x_2)$.

then f is not obliviously-computable.

Before proving Lemma 6.4.1, we use it to show max is not obliviously-computable.

For $f = \max(x_1, x_2)$, we let $\mathbf{a}_i = (i, 0)$ and $\mathbf{\Delta}_{ij} = (0, j)$, so for $i < j$,

$$\max(i, j) - \max(i, 0) = j - i > \max(j, j) - \max(j, 0) = 0$$

as desired (see Fig. 6.6). Adding $\mathbf{\Delta}_{ij}$ input after computing $f(\mathbf{a}_i)$ should produce $j - i$ additional output Y. However, adding $\mathbf{\Delta}_{ij}$ input after computing $f(\mathbf{a}_j)$ should not. Lemma 6.4.1 uses this to show there exists a reaction sequence that overproduces Y, thus max is not obliviously-computable. We now prove Lemma 6.4.1.

PROOF. Assume toward contradiction an output-oblivious CRN $\mathcal{C}$ stably computes f. To stably compute each $f(\mathbf{a}_i)$, the initial configuration $\mathbf{I}_{\mathbf{a}_i} \to^* \mathbf{O}_i$ for some configuration with $\mathbf{O}_i(Y) = f(\mathbf{a}_i)$, giving a sequence of configurations $(\mathbf{O}_i)_{i=1}^{\infty}$. By Dickson's Lemma [78], any sequence of nonnegative integer vectors has a nondecreasing subsequence, so there must be $\mathbf{O}_i \leq \mathbf{O}_j$ for some $i < j$. By assumption there exists $\mathbf{\Delta}_{ij} \in \mathbb{N}^d$ such that

$$f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j).$$

Now consider the initial configuration $\mathbf{I}_{\mathbf{a}_i + \mathbf{\Delta}_{ij}} \geq \mathbf{I}_{\mathbf{a}_i}$, so define the difference $\mathbf{D} = \mathbf{I}_{\mathbf{a}_i + \mathbf{\Delta}_{ij}} - \mathbf{I}_{\mathbf{a}_i} \in \mathbb{N}^{\mathcal{S}}$. Then the same sequence of reactions as $\mathbf{I}_{\mathbf{a}_i} \to^* \mathbf{O}_i$ is applicable to $\mathbf{I}_{\mathbf{a}_i + \mathbf{\Delta}_{ij}}$ reaching configuration $\mathbf{C}_i = \mathbf{O}_i + \mathbf{D}$, with $\mathbf{C}_i(Y) = \mathbf{O}_i(Y) = f(\mathbf{a}_i)$. Then to stably compute $f(\mathbf{a}_i + \mathbf{\Delta}_{ij})$ there must exist a further sequence of reactions α from $\mathbf{C}_i$ that produce an additional $f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i)$ copies of output Y.

By the same argument, from initial configuration $\mathbf{I}_{\mathbf{a}_j+\mathbf{\Delta}_{ij}}$ the configuration $\mathbf{C}_j = \mathbf{O}_j + \mathbf{D}$ is reachable, with $\mathbf{C}_j(Y) = \mathbf{O}_j(Y) = f(\mathbf{a}_j)$. Then since $\mathbf{O}_i \leq \mathbf{O}_j$, we have $\mathbf{C}_i \leq \mathbf{C}_j$, so the same sequence of reactions α is applicable to $\mathbf{C}_j$, reaching some configuration $\mathbf{C}'_j$ with an additional $f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i)$ copies of output Y. Now

$$\mathbf{C}'_j(Y) = f(\mathbf{a}_j) + f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}).$$

Since $\mathbf{I}_{\mathbf{a}_j+\mathbf{\Delta}_{ij}} \to^* \mathbf{C}'_j$, which overproduces Y, the output-oblivious CRN $\mathcal{C}$ cannot stably compute $f(\mathbf{a}_j + \mathbf{\Delta}_{ij})$. □

Lemma 6.4.1 is our main technical tool used to show that a particular semilinear, nondecreasing function is not obliviously-computable, the key challenge in the impossibility direction of Theorem 6.5.2. In fact, it is the only way a semilinear and nondecreasing function can fail to be obliviously-computable (see Theorem 6.5.4).

6.5. Main result: Full-dimensional case

To formally state our main result, Theorem 6.5.2, we must first define a **quilt-affine** function as the sum of a linear and periodic function (see Fig. 6.3b):

Definition 6.5.1. *A nondecreasing function $g : \mathbb{N}^d \to \mathbb{Z}$ is quilt-affine (with period p) if there exists $\nabla_g \in \mathbb{Q}^d_{\geq 0}$ and $B : \mathbb{Z}^d/p\mathbb{Z}^d \to \mathbb{Q}$ such that*

$$g(\mathbf{x}) = \nabla_g \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod p).$$

We call ∇_g the **gradient** of g, and the periodic function B the **periodic offset**. Without loss of generality we have the same period p along all inputs, since p could be the least common multiple of the periods along each input component $i = 1, \ldots, d$. Note that $\nabla_g \cdot \mathbf{x}$ and B can each be rational, but the sum $g(x) \in \mathbb{Z}$ will be integer-valued. We allow g to have negative output for technical reasons[9], but in the case that g is quilt-affine with nonnegative output (i.e. $g : \mathbb{N}^d \to \mathbb{N}$), there is a simple output-oblivious CRN construction to stably compute g. The intuitive idea is to use a single leader that reacts with every input species sequentially, tracks the periodic value $\overline{\mathbf{x}} \mod p$, and outputs the correct changes in g.

[9] The quilt-affine functions that describe f for large inputs may be negative on inputs close to the origin.

Our main result has a recursive condition where we fix an input of a function $f : \mathbb{N}^d \to \mathbb{N}$. For each $i = 1, \ldots, d$ and $j \in \mathbb{N}$, define the **fixed-input restriction**[10] $f_{[\mathbf{x}(i)\to j]} : \mathbb{N}^d \to \mathbb{N}$ of f for all $\mathbf{x} \in \mathbb{N}^d$ by $f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) = f(\mathbf{x}(1), \ldots, \mathbf{x}(i-1), j, \mathbf{x}(i+1), \ldots, \mathbf{x}(d))$.

We can now formally state our main result:

Theorem 6.5.2. *$f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable $\iff$*

(i) [nondecreasing] f is nondecreasing,

(ii) [eventually-min] there exist quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ such that for all $\mathbf{x} \geq \mathbf{n}$, $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$, and

(iii) [recursive] all fixed-input restrictions $f_{[\mathbf{x}(i)\to j]}$ are obliviously-computable.

We first prove that these conditions imply f is obliviously-computable via a general CRN construction in Section 6.6.

The nondecreasing condition (i) is necessary by Observation 6.2.1. It is immediate to see the recursive condition (iii) is also necessary:

Observation 6.5.3. *If $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, then any fixed-input restriction $f_{[\mathbf{x}(i)\to j]} : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable.*

Proof. Let the output-oblivious CRN $\mathcal{C}$ stably compute f. We define the output-oblivious CRN $\mathcal{C}'$ to "hardcode" the input $\mathbf{x}(i) = j$ by modifying the reactions of $\mathcal{C}$. Replace all instances of the leader L and input species X_i by L' and X_i' respectively, then add the initial reaction $L \to jX_i' + L'$. It is straightforward to verify that $\mathcal{C}'$ stably computes $f_{[\mathbf{x}(i)\to j]}$. □

Then the remaining work (and biggest effort of this paper) is to show the necessity of the eventually-min condition (ii): that every obliviously-computable function can be represented as eventually a minimum of a finite number of quilt-affine functions, which is shown as Theorem 6.7.1. The proof relies on f being semilinear, nondecreasing, and not having any "contradiction sequences" of the type described by Lemma 6.4.1. Thus the proof of Theorem 6.7.1 also yields the following alternative characterization to Theorem 6.5.2:

Theorem 6.5.4. *$f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable $\iff$ f is semilinear, nondecreasing, and has no sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots)$ meeting the conditions of Lemma 6.4.1.*

[10] We define $f_{[\mathbf{x}(i)\to j]}$ to have domain $\mathbb{N}^d$ because it is notationally convenient to have the same domain as f, but $f_{[\mathbf{x}(i)\to j]}$ only has relevant input in $d-1$ of its input components, making condition (iii) recursive.

This gives a "negative characterization" identifying behavior obliviously-computable functions must **avoid**, whereas Theorem 6.5.2, is a "positive characterization" describing the allowable behavior of such functions. We include Theorem 6.5.4, though it is less descriptive of the function, because it may be useful in other contexts.

6.6. Construction

First we show that any quilt-affine function with nonnegative range is stably computed by an output-oblivious CRN:

Lemma 6.6.1. *Every quilt-affine function $g : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable.*

Proof. Let $g : \mathbb{N}^d \to \mathbb{N}$ be quilt-affine with period p (recalling Definition 6.5.1). Notice that g has periodic finite differences. For each congruence class $\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$ and input component $i = 1, \ldots, d$, where $\mathbf{e}_i$ is the ith standard basis vector, define

$$\delta^i_{\overline{\mathbf{a}}} = \nabla_g \cdot \mathbf{e}_i + B(\overline{\mathbf{a} + \mathbf{e}_i} \mod p) - B(\overline{\mathbf{a}} \mod p) \in \mathbb{N}.$$

Observe that for all $\mathbf{x} \in \overline{\mathbf{a}}$, $g(\mathbf{x} + \mathbf{e}_i) - g(\mathbf{x}) = \delta^i_{\overline{\mathbf{a}}}$. We now use these periodic finite differences to construct an output-oblivious CRN $\mathcal{C}$ to stably compute g.

The CRN $\mathcal{C}$ has input species $X_1, \ldots, X_d$, output species Y, leader species L and p^d additional species $L_{\overline{\mathbf{a}}}$ for each $\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$ corresponding to auxiliary "states" of the leader. The initial reaction $L \to g(\mathbf{0})Y + L_{\overline{\mathbf{0}}}$ is accompanied by dp^d reactions of the form

$$L_{\overline{\mathbf{a}}} + X_i \to \delta^i_{\overline{\mathbf{a}}} Y + L_{\overline{\mathbf{a}+\mathbf{e}_i}}$$

for each $\overline{\mathbf{a}} \in \mathbb{Z}^d/P\mathbb{Z}^d$ and $i = 1, \ldots, d$. This CRN first creates $g(\mathbf{0})$ output, then sequentially outputs all finite differences, and is easily verified to stably compute g. □

We now prove (in Lemma 6.6.2) one direction of Theorem 6.5.2: that conditions (i), (ii), and (iii) imply an output-oblivious CRN can stably compute f. Intuitively, by the eventually-min condition (ii) we can compute $f(\mathbf{x})$ for $\mathbf{x} \geq \mathbf{n}$ by composing min and quilt-affine functions. If $\mathbf{x} \not\geq \mathbf{n}$, then for some input i, $\mathbf{x}(i) = j$, where $j < \mathbf{n}(i)$. By the recursive condition (iii) we can compute $f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) = f(\mathbf{x})$[11]. The key remaining insight is a technique (similar to a proof in [61]) to compose these pieces using minimum and indicator functions.

The proof of Lemma 6.6.2 then expresses f as such a minimum of finitely many pieces. We justify that f is obliviously-computable by showing that each piece is obliviously-computable, since by Observation 6.2.2 obliviously-computable functions are closed under composition.

Lemma 6.6.2. *If $f : \mathbb{N}^d \to \mathbb{N}$ satisfies the conditions of Theorem 6.5.2, f is obliviously-computable.*

Proof. Assume $f : \mathbb{N}^d \to \mathbb{N}$ satisfies the conditions of Theorem 6.5.2. Then by eventually-min condition (ii), there exist quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ such that $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$ for all $\mathbf{x} \geq \mathbf{n}$. Without loss of generality, assume $\mathbf{n} = (n, \ldots, n)$. This assumption may be made because if $\mathbf{n}$ is not of this form, we can take $n = \max\{\mathbf{n}(1), \mathbf{n}(2), \ldots, \mathbf{n}(d)\}$ and still satisfy condition (ii). (Note that condition (ii) is the only condition to reference $\mathbf{n}$.)

Let $\mathbf{x} \vee \mathbf{n} = (\max(\mathbf{x}(1), n), \ldots, \max(\mathbf{x}(d), n))$ denote the componentwise max of $\mathbf{x}$ and $\mathbf{n}$. Let $\mathbb{1}_{\{\mathbf{x}(i)>j\}} : \mathbb{N}^d \to \{0, 1\}$ denote the indicator function that is 1 $\iff$ its input $\mathbf{x}$ obeys $\mathbf{x}(i) > j$. Recall $f_{[\mathbf{x}(i)\to j]}$ is the fixed-input restriction setting input $\mathbf{x}(i) = j$. We claim that f can be expressed as

$$f(\mathbf{x}) = \min\Big[f(\mathbf{x} \vee \mathbf{n}), \underbrace{f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot f(\mathbf{x} \vee \mathbf{n})}_{\substack{i=1,\ldots,d \\ j=0,\ldots,n-1}}\Big]. \tag{6.1}$$

We first show $f \geq \min[\ldots]$ since for all $\mathbf{x} \in \mathbb{N}^d$, $f(\mathbf{x})$ is achieved by some term. If $\mathbf{x} \geq \mathbf{n}$, then $f(\mathbf{x}) = f(\mathbf{x} \vee \mathbf{n})$. If $\mathbf{x} \not\geq \mathbf{n}$, there must be $\mathbf{x}(i) = j$ for some $i = 1, \ldots, d$ and $j = 0, \ldots, n-1$, so $f(\mathbf{x}) = f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) = f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot f(\mathbf{x} \vee \mathbf{n})$ since the indicator is 0.

We next show $f \leq \min[\ldots]$ since $f(\mathbf{x}) \leq$ each term for all $\mathbf{x} \in \mathbb{N}^d$. $f(\mathbf{x}) \leq f(\mathbf{x} \vee \mathbf{n})$ since $\mathbf{x} \leq (\mathbf{x} \vee \mathbf{n})$ and f is nondecreasing. When $\mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) = 1$, we then have $f(\mathbf{x}) \leq f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot f(\mathbf{x} \vee \mathbf{n})$. If $\mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) = 0$, then $\mathbf{x}(i) \leq j$ so $f(\mathbf{x}) \leq f_{[\mathbf{x}(i)\to j]}(\mathbf{x})$ since f is nondecreasing. Thus equation 6.1 holds as claimed.

It remains to show that f is obliviously-computable. From Observation 6.2.2, output-oblivious CRNs are closed under composition, and equation 6.1 gives a method to express f as a composition of functions. Thus it suffices to show that each piece is obliviously-computable. Specifically, we

[11] As a result, this construction is recursive, with an additional input being fixed at each level of the recursion, so the base case is simply a constant function.

show the functions $\min : \mathbb{N}^k \to \mathbb{N}$ (for any k), $f(\mathbf{x} \vee \mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$, $f_{[\mathbf{x}(i) \to j]}(\mathbf{x}) : \mathbb{N}^d \to \mathbb{N}$, and $c(a, b, \mathbf{x}) = a + \mathbb{1}_{\{\mathbf{x}(i) > j\}}(\mathbf{x}) \cdot b : \mathbb{N}^{d+2} \to \mathbb{N}$ are each obliviously-computable. Implicit in the composed CRN to stably compute f as the composition from equation 6.1 is the "fan out" operation where reactions of the form $X_i \to X_i^1, \ldots, X_i^m$ create multiple copies of species X_i to be used as independent inputs to multiple "modules" in this composition.

$\min : \mathbb{N}^k \to \mathbb{N}$ **is obliviously-computable:**

Consider the CRN with single reaction $X_1, \ldots, X_k \to Y$, the natural generalization of two-input min from Fig. 6.1.

$f(\mathbf{x} \vee \mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$ **is obliviously-computable:**

By condition (ii), $f(\mathbf{x} \vee \mathbf{n}) = \min_k(g_k(\mathbf{x} \vee \mathbf{n}))$ since $\mathbf{x} \vee \mathbf{n} \geq \mathbf{n}$, so it suffices to show for each quilt-affine $g_k : \mathbb{N}^d \to \mathbb{Z}$ that $g_k(\mathbf{x} \vee \mathbf{n})$ is obliviously-computable.

By condition (ii), $g_k(\mathbf{x} + \mathbf{n}) \geq f(\mathbf{x} + \mathbf{n}) \geq 0$ since $\mathbf{x} + \mathbf{n} \geq \mathbf{n}$. Then $g_k(\mathbf{x} + \mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$ is still quilt-affine since that property is preserved by translation, but now has guaranteed nonnegative output. Thus by Lemma 6.6.1, $g_k(\mathbf{x} + \mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable.

Letting $(\mathbf{x} - \mathbf{n})_+ = (\max(\mathbf{x}(1) - n, 0), \ldots, \max(\mathbf{x}(d) - n, 0))$, we then show the function $(\mathbf{x} - \mathbf{n})_+ : \mathbb{N}^d \to \mathbb{N}^d$ is obliviously-computable via the CRN with reactions $(n+1)X_i \to nX_i + Y_i$ for each component $i = 1, \ldots, d$.

Finally, because $\mathbf{x} \vee \mathbf{n} = (\mathbf{x} - \mathbf{n})_+ + \mathbf{n}$, we have shown $g_k(\mathbf{x} \vee \mathbf{n}) = g_k((\mathbf{x} - \mathbf{n})_+ + \mathbf{n})$ is obliviously-computable as the composition of obliviously-computable $g_k(\mathbf{x} + \mathbf{n})$ and $(\mathbf{x} - \mathbf{n})_+$.

$f_{[\mathbf{x}(i) \to j]}(\mathbf{x}) : \mathbb{N}^d \to \mathbb{N}$ **is obliviously-computable:**

This is precisely the assumed recursive condition (iii).

$c(a, b, \mathbf{x}) = a + \mathbb{1}_{\{\mathbf{x}(i) > j\}}(\mathbf{x}) \cdot b : \mathbb{N}^{d+2} \to \mathbb{N}$ **is obliviously-computable:**

Consider the output-oblivious CRN (with input species $A, B, X_1, \ldots, X_d$ and output species Y) with two reactions $A \to Y$ and $(j+1)X_i + B \to (j+1)X_i + Y$. The A is all converted to Y, and $(j+1)$ copies of input species X_i catalyze the conversion of B to Y, which will only happen when $\mathbb{1}_{\{\mathbf{x}(i) > j\}}(\mathbf{x}) = 1$. Thus this stably computes $c(a, b, \mathbf{x})$ as desired. □

6.7. Output-oblivious implies eventually min of quilt-affine functions

To complete the proof of Theorem 6.5.2, it remains to show the necessity of the eventually-min condition (ii):

THEOREM 6.7.1. *If $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, then there exist quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ such that for all $\mathbf{x} \geq \mathbf{n}$, $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$.*

For the remainder of Section 6.7, we fix an obliviously-computable $f : \mathbb{N}^d \to \mathbb{N}$, and Section 6.7 is devoted to finding $g_1, \ldots, g_m$ and $\mathbf{n}$ satisfying Theorem 6.7.1.

6.7.1. Proof outline.

Here we outline the proof of Theorem 6.7.1, which is spread over Sections 6.7.2-6.7.4.

Section 6.7.2.

Since $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, f is semilinear (recall Definition 6.2.6), and we first consider all threshold sets used to define the semilinear domains of the affine partial functions that define f. Each threshold set defines a hyperplane, and we use these hyperplanes to define **regions** (see Fig. 6.8a and Fig. 6.8c). We consider regions as subsets of $\mathbb{R}^d_{\geq 0}$, so they are convex polyhedra with useful geometric properties[12].

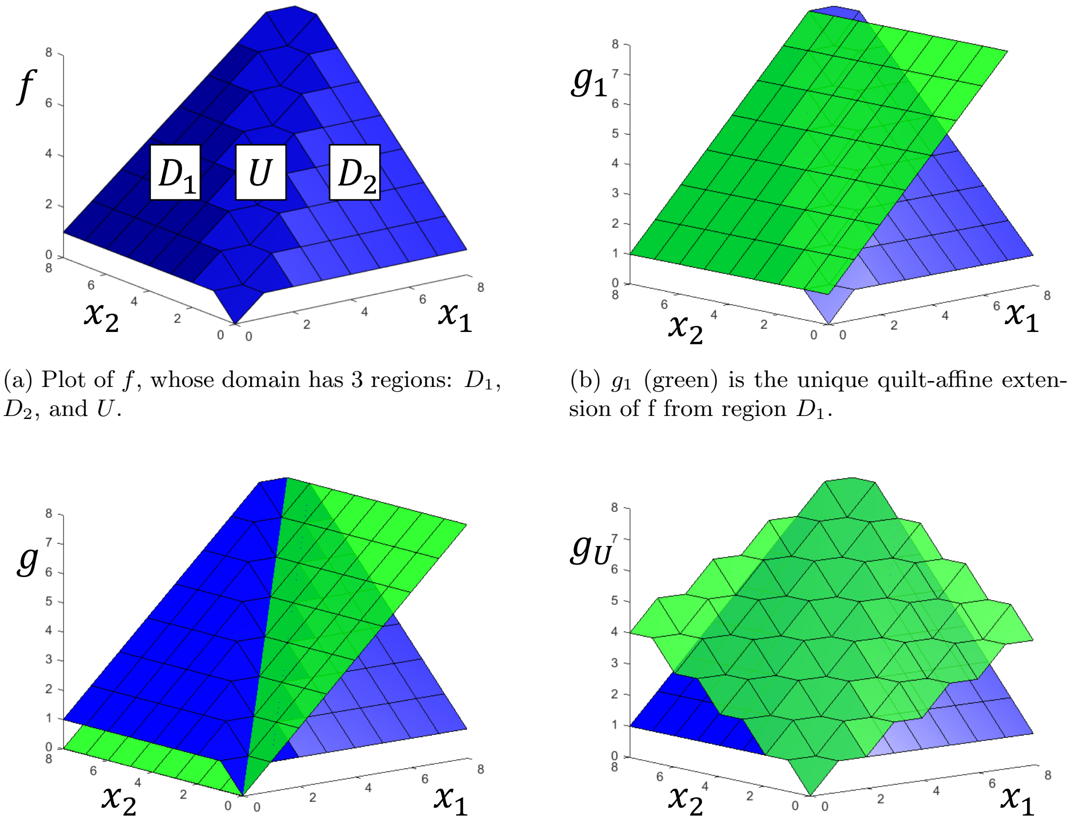
The regions partition[13] the points in the domain $\mathbb{N}^d$. To prove Theorem 6.7.1, for each region R_k we will identify a quilt-affine function g_k (the **extension** of f from region R_k) such that $g(\mathbf{x}) = f(\mathbf{x})$ for all integer $\mathbf{x} \in R_k$. To ensure $f = \min_k(g_k)$, we further require that these quilt-affine extensions **eventually dominate** f (each $g_k(\mathbf{x}) \geq f(\mathbf{x})$ for sufficiently large $\mathbf{x}$). Also, because we only care about sufficiently large $\mathbf{x}$, we need only consider **eventual** regions which are unbounded in all inputs (for example regions 3,4, and 5 in Fig. 6.8a).

As an example, consider the semilinear, nondecreasing function

$$f(x_1, x_2) = \begin{cases} x_1 + 1, & \text{if } x_1 < x_2 \text{ (region } D_1) \\ x_2 + 1, & \text{if } x_1 > x_2 \text{ (region } D_2) \\ x_1 & \text{if } x_1 = x_2 \text{ (region } U) \end{cases}$$

[12] What we consider is a restricted case of a **hyperplane arrangement** [154], with well-studied combinatorial properties.

[13] Without loss of generality, we assume that the hyperplanes do not intersect $\mathbb{N}^d$, so that the partition is well-defined (see Fig. 6.8a).



(a) Plot of f, whose domain has 3 regions: D_1, D_2, and U.

(b) g_1 (green) is the unique quilt-affine extension of f from region D_1.

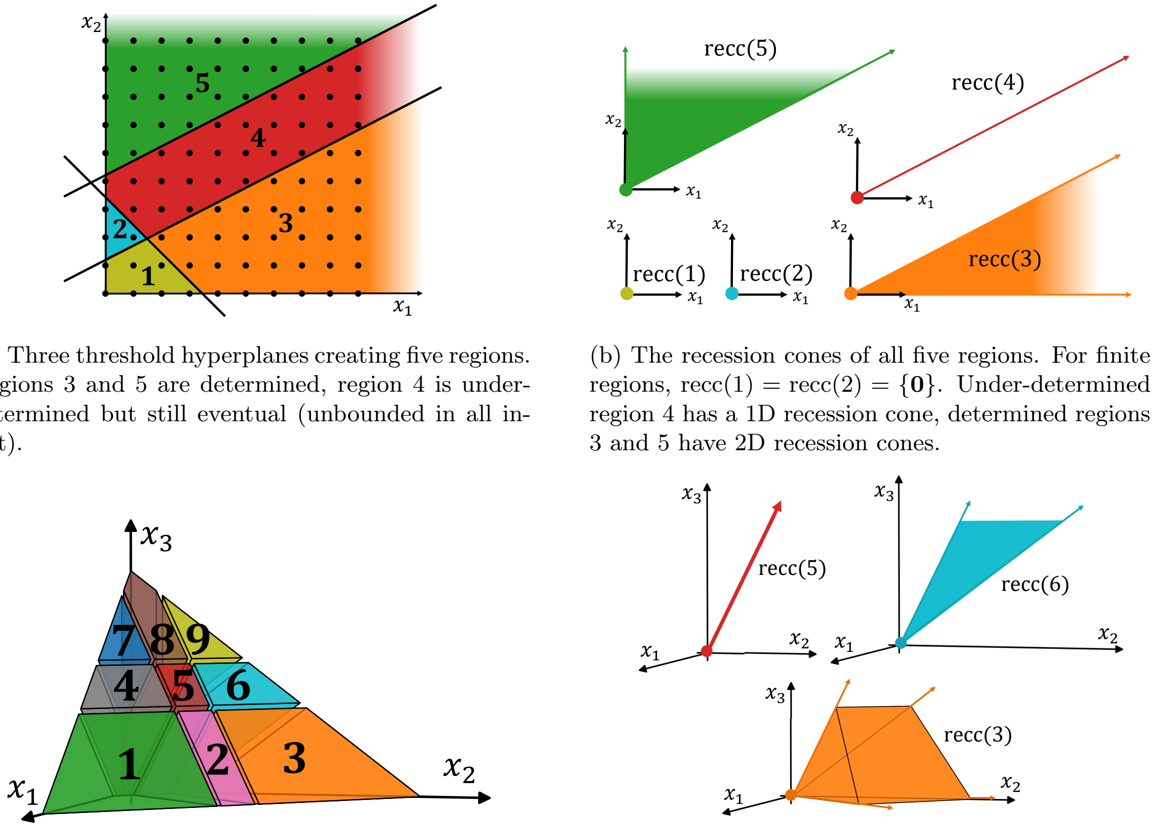
(c) g (green) is a quilt-affine extension of f from U, but $g < f$ on D_1.

(d) g_U (green) is a quilt-affine extension of f from U, and $g_U \geq f$.

FIGURE 6.7. Obliviously-computable f can be expressed as a min of quilt-affine functions.

As in Definition 6.2.6, f is piecewise-affine, with semilinear domains that happen to be only defined by threshold sets. These thresholds then partition the domain into three regions: D_1, D_2, and U (see Fig. 6.7a). For region D_1, there is a unique quilt-affine extension $g_1(x_1, x_2) = x_1 + 1$ (note an affine function is the special case of a quilt-affine function with period 1). Also, g_1 eventually dominates f as desired, since $g_1(\mathbf{x}) \geq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{N}^2$ (see Fig. 6.7b). By symmetry, we have the same for region D_2 and its extension $g_2(x_1, x_2) = x_2 + 1$.

These desirable properties follow from D_1 and D_2 being "wide" regions that we define to be **determined** (formalized later). On the other hand, U is a "narrow" region that is **under-determined** (formalized later). As a result, there is not a unique quilt-affine extension from U. For example, $g(x_1, x_2) = x_1$ is a quilt-affine extension, however, we do not have $g(\mathbf{x}) \geq f(\mathbf{x})$ for all sufficiently large $\mathbf{x}$ (see Fig. 6.7c).



(a) Three threshold hyperplanes creating five regions. Regions 3 and 5 are determined, region 4 is under-determined but still eventual (unbounded in all input).

(b) The recession cones of all five regions. For finite regions, recc(1) = recc(2) = $\{\mathbf{0}\}$. Under-determined region 4 has a 1D recession cone, determined regions 3 and 5 have 2D recession cones.

(c) Two pairs of parallel threshold hyperplanes creating nine eventual regions. Regions 1,3,7,9 are determined. Region 5 is under-determined with 1D recession cone. Regions 2,4,6,8 are under-determined with 2D recession cones.

(d) recc(5) $\subseteq$ recc(6) $\subseteq$ recc(3) so region 3 is a determined neighbor of under-determined region 5 and under-determined region 6. Also, region 6 is a neighbor of region 5.

FIGURE 6.8. Examples with domains $\mathbb{N}^2$ (top) and $\mathbb{N}^3$ (bottom), with threshold hyperplanes giving regions (left), which are classified by their recession cones (right).

In order to identify a quilt-affine extension from U that does eventually dominate f, we will refer to the unique extensions g_1 and g_2 from regions D_1 and D_2, which are **neighbors** of U (formalized later). We can construct a quilt-affine function with a gradient $(\frac{1}{2}, \frac{1}{2})$ that is the average of the gradients $(1, 0)$ of g_1 and $(0, 1)$ of g_2. In particular, we can let $g_U(x_1, x_2) = \lceil \frac{x_1+x_2}{2} \rceil$ (note this is a quilt-affine function with period 2, see Fig. 6.7d). We then have $f(\mathbf{x}) = \min(g_1(\mathbf{x}), g_2(\mathbf{x}), g_U(\mathbf{x}))$ for all $\mathbf{x} \geq \mathbf{n} = \mathbf{0}$ as guaranteed by Theorem 6.7.1.

We now describe how we formalize the notion of a **determined region**, **under-determined region**, and **neighbor**, for the general case of domain $\mathbb{N}^d$, where the regions are convex polyhedra in $\mathbb{R}^d$.

Section 6.7.3. To formally define determined regions, we identify the **recession cone** $\text{recc}(R) \subseteq \mathbb{R}^d$ of each region R: the set of vectors along infinite rays in R [144] (and $\mathbf{0}$; see Fig. 6.8b and Fig. 6.8d). A determined region D is defined as having a d-dimensional recession cone (see regions 3 and 5 in Fig. 6.8a and regions 1,3,7,9 in Fig. 6.8c). For determined regions, we can prove (see Lemmas 6.7.7 and 6.7.9) there is a unique quilt-affine extension, which eventually dominates f.

Section 6.7.4. Under-determined regions are then defined as having a recession cone with dimension $< d$ (see regions 1,2,4 in Fig. 6.8a and regions 2,4,5,6,8 in Fig. 6.8c). The above arguments do not work for under-determined regions. Instead, identify the **neighbors** of an under-determined region U as regions R with $\text{recc}(U) \subsetneq \text{recc}(R)$ (see Fig. 6.8b and Fig. 6.8d). We consider the neighbors of U that are determined regions. The possible behavior of f on U is constrained by the unique extensions from these regions, and we can define an extension from U based on an averaging process. (See Lemma 6.7.16.)

6.7.2. Domain Decomposition.

To identify the quilt-affine components g_k, the strategy will be to partition the domain $\mathbb{N}^d$ into regions where the restriction of f to that region yields a quilt-affine function. Definition 6.2.6 as stated is not enough to expose this structure of f, but we argue below that it implies Lemma 6.7.3, which characterizes the semilinear functions in a more detailed way conducive to our proof strategy.

By Lemma 6.2.7, f is semilinear, so by Definition 6.2.6, f is the union of affine partial functions, whose disjoint domains are semilinear subsets of $\mathbb{N}^d$. This representation is not unique, so for the rest of the section we fix some arbitrary such representation of f. Recall by Definition 6.2.5, these semilinear domains are finite Boolean combinations of threshold and mod sets, so consider the collection $\mathcal{T}$ of all threshold sets and the collection $\mathcal{M}$ of all mod sets that defined any of these semilinear domains.

Let $\mathcal{T}$ consist of l threshold sets $\{x \in \mathbb{N}^d : \mathbf{t}_i \cdot \mathbf{x} \geq h_i\}$ for each $i = 1, \ldots, l$, where $\mathbf{t}_i \in \mathbb{Z}^d$ and $h_i \in \mathbb{Z}$. These thresholds are equivalently written $2\mathbf{t}_i \cdot \mathbf{x} > 2h_i - 1$ (since $\mathbf{t}_i \cdot \mathbf{x} \geq h_i \iff \mathbf{t}_i \cdot \mathbf{x} > h_i - \frac{1}{2}$), so we can assume without loss of generality that the boundary hyperplanes $H_i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{t}_i \cdot \mathbf{x} = h_i\}$ contain no integer points (see Fig. 6.8a). These hyperplanes then partition the domain $\mathbb{N}^d$.

For all $\mathbf{y} \in \mathbb{N}^d$, we describe how to formally define the region which contains $\mathbf{y}$. Let $s_i = \text{sign}(\mathbf{t}_i \cdot \mathbf{y} - h_i) = \pm 1$ for each $i = 1, \ldots, l$. Defining the threshold matrix $T \in \mathbb{Z}^{l \times d}$, offset vector

$\mathbf{h} \in \mathbb{Z}^l$, and diagonal sign matrix $S \in \mathbb{Z}^{l \times l}$ as

$$T = \begin{pmatrix} \mathbf{t}_1^\intercal \\ \vdots \\ \mathbf{t}_l^\intercal \end{pmatrix} \quad \mathbf{h} = \begin{pmatrix} h_1 \\ \vdots \\ h_l \end{pmatrix} \quad S = \begin{pmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_l, \end{pmatrix}$$

where each $s_i \in \{-1, 1\}$, then $S(T\mathbf{y} - \mathbf{h}) \geq \mathbf{0}$. This concise form will let us define the region of points that are in precisely the same threshold sets as $\mathbf{y}$ (those that agree on the signs of the components of $T\mathbf{y} - \mathbf{h}$):

Definition 6.7.2. *Let S be a sign matrix: a diagonal matrix with diagonal entries $= \pm 1$. Then the* ***region (induced by S)*** *is defined as*

$$R = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : S(T\mathbf{x} - \mathbf{h}) \geq \mathbf{0}\}.$$

When referring to a region R, we use S_R to denote the sign matrix that induces R.

We consider nonnegative real vectors $\mathbf{x} \in \mathbb{R}^d$ rather than just $\mathbf{x} \in \mathbb{N}^d$, but we are only truly concerned with the integer points $R \cap \mathbb{N}^d$, and only consider regions where $R \cap \mathbb{N}^d \neq \emptyset$. Also, since each $\mathbf{y} \in \mathbb{N}^d$ induces a unique sign matrix as shown above, it follows that every $\mathbf{y} \in \mathbb{N}^d$ is contained in some unique region. The reason to consider $R \subset \mathbb{R}^d$ is that each region R is a convex polyhedron, with convenient properties from convex geometry (see Fig. 6.8a and Fig. 6.8c).

Now consider the collection $\mathcal{M}$, consisting of m mod sets $\{\mathbf{x} \in \mathbb{N}^d : \mathbf{a_i} \cdot \mathbf{x} \equiv b_i \mod c_i\}$ for each $i = 1, \ldots, m$, where $\mathbf{a_i} \in \mathbb{Z}^d, b_i \in \mathbb{Z}, c_i \in \mathbb{N}_+$. Then let the **global period** p be the least common multiple $\mathrm{lcm}_i(c_i)$, so all elements of each congruence class $\overline{\mathbf{a}} \in \mathbb{Z}^d / p\mathbb{Z}^d$ are contained in precisely the same mod sets. Thus for a region R, the set $R \cap \overline{\mathbf{a}}$ is contained in precisely the same threshold and mod sets, so the restriction $f|_{R \cap \overline{\mathbf{a}}}$ is an affine partial function (recall Definition 6.2.6).

We now summarize this decomposition in the following lemma (proof argued above) as a characterization of a semilinear function. Note that this applies to all semilinear functions, even those that are decreasing.

Lemma 6.7.3. *Let $f : \mathbb{N}^d \to \mathbb{N}$ be a semilinear function. Then there exist a finite set of* ***regions*** *$R_1, \ldots, R_n \subseteq \mathbb{R}^d$ and a* ***global period*** *$p \in \mathbb{N}_+$ such that for each region R_i and congruence class*

$\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$, *there exist* $\nabla_{R_i,\overline{\mathbf{a}}} \in \mathbb{Q}^d$ *and* $b_{R_i,\overline{\mathbf{a}}} \in \mathbb{Q}$ *such that the restriction of* f *defined for all* $\mathbf{x} \in R_i \cap \overline{\mathbf{a}}$ *by*

$$f|_{R_i \cap \overline{\mathbf{a}}}(\mathbf{x}) = \nabla_{R_i,\overline{\mathbf{a}}} \cdot \mathbf{x} + b_{R_i,\overline{\mathbf{a}}}$$

is a (rational) affine partial function.

Notice the similarity in form between these affine partial functions in Lemma 6.7.3 and Definition 6.5.1 of quilt-affine functions. In fact, we will show the behavior of f on some regions has a unique quilt-affine structure. This will require the region to be "infinitely wide in all directions", a notation we now make precise to define these determined regions.

6.7.3. Determined Regions. To formally define determined regions, we must first make the connection between regions and their *recession cones* (more information on recession cones can be found in [**144**]).

Definition 6.7.4. *For a region* R*, the* ***recession cone*** *of* R *is*

$$\operatorname{recc}(R) = \{\mathbf{y} \in \mathbb{R}^d : \mathbf{x} + \lambda\mathbf{y} \in R \textit{ for all } \mathbf{x} \in R, \lambda \in \mathbb{R}_{\geq 0}\}.$$

The recession cone corresponds to the vectors $\mathbf{y}$ along infinite rays in the region R, and is a convex polyhedral cone (recall that a subset of $\mathbb{R}^d$ is a **cone** if it is closed under positive scalar multiplication) (see Fig. 6.8b and Fig. 6.8d). Note that vectors $\mathbf{y} \in \operatorname{recc}(R)$ should not be thought of as **positions** within R, but instead as **directions of motion** that will keep one within R no matter how far one moves in that direction.

Recall by Definition 6.7.2 a region $R = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : S(T\mathbf{x} - \mathbf{h}) \geq \mathbf{0}\}$. It is possible to equivalently define

$$\operatorname{recc}(R) = \{\mathbf{y} \in \mathbb{R}^d_{\geq 0} : S_R T\mathbf{y} \geq \mathbf{0}\}.$$

In other words, the recession cone is defined by the homogenized version of the same inequalities that defined R. One can easily verify this is equivalent to Definition 6.7.4.

We then say a region R is **determined** if $\dim \operatorname{recc}(R) = d$. Otherwise, the region R is **underdetermined**. We now make precise the idea that a determined region with full-dimensional recession cone is "infinitely wide in all directions":

Lemma 6.7.5. *Let D be a determined region. Then the recession cone* $\mathrm{recc}(D)$ *contains open balls of arbitrarily large radius.*

Proof. Since $\mathrm{recc}(D)$ is a d-dimensional convex polyhedron, it has nonempty interior, so there exists some $\mathbf{x} \in \mathrm{int}(\mathrm{recc}(D))$ and an open ball $B_\epsilon(\mathbf{x}) \subset \mathrm{recc}(D)$ of radius ϵ around $\mathbf{x}$ contained in the cone. Since recession cones are closed under positive scalar multiplication, for any positive scalar c, the ball $B_{c\epsilon}(c\mathbf{x}) \subset \mathrm{recc}(D)$. □

We next make precise the idea that the function has a unique quilt-affine structure on a determined region.

Definition 6.7.6. *An **extension** $g : \mathbb{N}^d \to \mathbb{Z}$ (of f) from a region R is a quilt-affine function that agrees with f on R: $f(\mathbf{x}) = g(\mathbf{x})$ for all integer $\mathbf{x} \in R \cap \mathbb{N}^d$.*

We will now show there is a unique extension from each determined region (such as in Fig. 6.7(b)). The construction of the regions yields a periodic piecewise-affine structure f restricted to a region (Lemma 6.7.3). In order for f to be nondecreasing, these affine gradients must all agree, which will let us uniquely describe a quilt-affine extension g using Definition 6.5.1.

Lemma 6.7.7. *There is a unique extension $g : \mathbb{N}^d \to \mathbb{Z}$ from any determined region D.*

Proof. By Definition 6.5.1, it suffices to show that there is a $\nabla_g \in \mathbb{Q}^d$ and $B : \mathbb{Z}^d/p\mathbb{Z}^d \to \mathbb{Q}$ such that, defining $g : \mathbb{N}^d \to \mathbb{Z}$ for all $\mathbf{x} \in \mathbb{N}^d$ as

$$g(\mathbf{x}) = \nabla_g \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod p),$$

then for all $\mathbf{x} \in D \cap \mathbb{N}^d$, $g(\mathbf{x}) = f(\mathbf{x})$.

By Lemma 6.7.3, for each $\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$, there are $\nabla_{\overline{\mathbf{a}}} \in \mathbb{Q}^d$ and $b_{\overline{\mathbf{a}}} \in \mathbb{Q}$ such that

$$f|_{D \cap \overline{\mathbf{a}}}(\mathbf{x}) = \nabla_{\overline{\mathbf{a}}} \cdot \mathbf{x} + b_{\overline{\mathbf{a}}}.$$

Since D contains arbitrarily large open balls by Lemma 6.7.5, it contains points in all congruence classes, so all p^d constants $b_{\overline{\mathbf{a}}}$ are well-defined. Then we define the periodic offset function $B(\overline{\mathbf{x}} \mod p) = b_{\overline{\mathbf{x}} \mod p}$. This is almost what we require to prove the lemma, except that $\nabla_{\overline{\mathbf{a}}}$ depends on $\overline{\mathbf{a}}$. It remains to show that all vectors $\nabla_{\overline{\mathbf{a}}}$ are equal, so we can define the gradient $\nabla_g = \nabla_{\overline{\mathbf{a}}}$ for any $\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$.

Assuming for the sake of contradiction that $\nabla_{\overline{\mathbf{a}}} \neq \nabla_{\overline{\mathbf{b}}}$ for some equivalence classes $\overline{\mathbf{a}}, \overline{\mathbf{b}}$, we will show that f cannot be nondecreasing. Since the recession cone $\mathrm{recc}(D)$ is full-dimensional and $\nabla_{\overline{\mathbf{a}}} - \nabla_{\overline{\mathbf{b}}} \neq \mathbf{0}$, there must exist some $\mathbf{v} \in \mathrm{recc}(D)$ such that $\nabla_{\overline{\mathbf{a}}} \cdot \mathbf{v} \neq \nabla_{\overline{\mathbf{b}}} \cdot \mathbf{v}$. Furthermore, by density of the rationals, we can assume $\mathbf{v} \in \mathbb{Q}^d$, and by scaling by the denominator we can assume $\mathbf{v} \in \mathbb{N}^d$. Now without loss of generality assume $\nabla_{\overline{\mathbf{a}}} \cdot \mathbf{v} > \nabla_{\overline{\mathbf{b}}} \cdot \mathbf{v}$.

Now pick some $\mathbf{y} \in D \cap \overline{\mathbf{a}}$, and some $\mathbf{z} \in D \cap \overline{\mathbf{b}}$ with $\mathbf{y} < \mathbf{z}$, which must exist because again D contains arbitrarily large open balls by Lemma 6.7.5. But since $\nabla_{\overline{\mathbf{a}}} \cdot \mathbf{v} > \nabla_{\overline{\mathbf{b}}} \cdot \mathbf{v}$, moving along $\mathbf{v}$, the function values in $\overline{\mathbf{a}}$ grow faster than in $\overline{\mathbf{b}}$ when moving along $\mathbf{v}$ from $\mathbf{y}$ and $\mathbf{z}$, respectively. Note that $\mathbf{y} + cp\mathbf{v}, \mathbf{z} + cp\mathbf{v} \in D$ by definition of $\mathbf{v} \in \mathrm{recc}(D)$, and $cp\mathbf{v} \in p\mathbb{Z}^d$, so $\mathbf{y} + cp\mathbf{v} \in D \cap \overline{\mathbf{a}}$ and $\mathbf{z} + cp\mathbf{v} \in D \cap \overline{\mathbf{b}}$. Thus for some multiple cp of the period p, we must have

$$f(\mathbf{y} + cp\mathbf{v}) > f(\mathbf{z} + cp\mathbf{v})$$

but then f is not nondecreasing, since $\mathbf{y} + cp\mathbf{v} < \mathbf{z} + cp\mathbf{v}$.

Thus there is a uniquely determined gradient ∇_g. While there was not necessarily a unique choice for the period p, any valid choice will define the same function g. □

Note that although the function $g : \mathbb{N}^d \to \mathbb{Z}$ of Lemma 6.7.7 is uniquely defined based on D, it can have different **representations** in terms of the periodic term B. For example, doubling the period p [and appropriately defining B on the new values] would preserve the definition of g. (The lack of a unique extension from an **under-determined** region, and the fact that not all of them can be chosen to define f, is what makes such regions more challenging to deal with in Section 6.7.4.)

We next make precise why these extensions g_k can be used in the eventual-min that will define f to prove Theorem 6.7.1.

Definition 6.7.8. *An extension $g : \mathbb{N}^d \to \mathbb{Z}$ **eventually dominates** f if there exists $\mathbf{n} \in \mathbb{N}^d$ such that $f(\mathbf{x}) \leq g(\mathbf{x})$ for all $\mathbf{x} \geq \mathbf{n}$.*

We now show the uniquely defined extension g from a determined region D eventually dominates f. The idea is that if the extension g did not eventually dominate f, then we can apply Lemma 6.4.1 to show f is not obliviously-computable.

For example, the function

$$\max(x_1, x_2) = \begin{cases} x_2, & \text{if } x_1 \leq x_2 \\ x_1, & \text{if } x_1 > x_2 \end{cases}$$

is naturally identified by two determined regions, with unique extensions $g_1(\mathbf{x}) = (0,1) \cdot \mathbf{x}$ and $g_2(\mathbf{x}) = (1,0) \cdot \mathbf{x}$. These extensions do not eventually dominate f, and we already saw in Section 6.4 how Lemma 6.4.1 applies to max. Thus the following lemma generalizes this example, finding a "contradiction sequence" $(\mathbf{a}_1, \mathbf{a}_2, \ldots)$ to apply Lemma 6.4.1 whenever a determined extension g does not eventually dominate f:

Lemma 6.7.9. *The unique extension g from a determined region D eventually dominates f.*

Proof. Let g be the extension from some determined region D. Assume toward contradiction that g does not eventually dominate f, so for any point $\mathbf{n} \in \mathbb{N}^d$ there exists some "bad point" $\mathbf{b} \geq \mathbf{n}$ with $f(\mathbf{b}) > g(\mathbf{b})$. (Fig. 6.6 shows an example of the variables below as used in Lemma 6.4.1, although the sequence $(\mathbf{a}_i)$ is chosen more straightforwardly when proving max is not obliviously computable.) We will use Lemma 6.4.1 to show this implies f is not obliviously-computable. To satisfy the Lemma conditions, we must construct an increasing sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in \mathbb{N}^d$ such that for all $i < j$ there exists some $\mathbf{\Delta}_{ij} \in \mathbb{N}^d$ with

$$f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j).$$

We will choose $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in D \cap \overline{\mathbf{a}}$ to be points in D that are all in the same congruence class $\overline{\mathbf{a}} \mod p$, and a sequence of vectors $(\mathbf{v}_1, \mathbf{v}_2, \ldots) \in \mathbb{N}^d$ such that for all i, $\mathbf{a}_i + \mathbf{v}_i$ is a "bad point": $f(\mathbf{a}_i + \mathbf{v}_i) > g(\mathbf{a}_i + \mathbf{v}_i)$, while for all $j > i$, $\mathbf{a}_j + \mathbf{v}_i \in D$, so $f(\mathbf{a}_j + \mathbf{v}_i) = g(\mathbf{a}_j + \mathbf{v}_i)$. Then for $i < j$, let $\mathbf{\Delta}_{ij} = \mathbf{v}_i$, so

$$\begin{aligned} f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) &> g(\mathbf{a}_i + \mathbf{v}_i) - g(\mathbf{a}_i) \quad \text{since } \mathbf{a}_i + \mathbf{v}_i \text{ is a ``bad point''} \\ &= g(\mathbf{a}_j + \mathbf{v}_i) - g(\mathbf{a}_j) \quad \text{since } g \text{ is quilt-affine and } \mathbf{a}_i \equiv \mathbf{a}_j \mod p \\ &= f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j) \end{aligned}$$

as desired, where $f = g$ for points in $\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_j + \mathbf{v}_i \in D$.

It remains to show how to construct the sequences $(\mathbf{a}_1, \mathbf{a}_2, \ldots)$ and $(\mathbf{v}_1, \mathbf{v}_2, \ldots)$ recursively, to ensure for all $i < j$ that $\mathbf{a}_i \in \overline{\mathbf{a}}$, $\mathbf{a}_i + \mathbf{v}_i$ is a “bad point”, and $\mathbf{a}_j + \mathbf{v}_i \in D$. Let $\mathbf{a}_1 \in D$ arbitrarily, and the fixed congruence class $\overline{\mathbf{a}} = \overline{\mathbf{a}_1}$. For each $\mathbf{a}_i$, there must be a “bad point” above $\mathbf{a}_i$, so we recursively define each $\mathbf{v}_i$ based on $\mathbf{a}_i$ such that $\mathbf{a}_i + \mathbf{v}_i$ is this “bad point”: $f(\mathbf{a}_i + \mathbf{v}_i) > g(\mathbf{a}_i + \mathbf{v}_i)$.

Now we recursively define each $\mathbf{a}_j$ based on $\mathbf{a}_{j-1}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}$ to ensure the sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots)$ is increasing, congruent, and $\mathbf{a}_j + \mathbf{v}_i \in D$ for all $i < j$. Since the recession cone $\mathrm{recc}(D)$ contains open balls of arbitrary radius by Lemma 6.7.5, we can find some vector $\mathbf{y} \in \mathrm{recc}(D)$ such that such that the open ball $B_r(\mathbf{y}) \subset \mathrm{recc}(D)$ for a large radius $r \geq \max_{i<j} \|\mathbf{v}_i\|$. Also, we can ensure $\mathbf{y} \equiv \mathbf{0} \mod p$. Then we can let $\mathbf{a}_j = \mathbf{a}_{j-1} + \mathbf{y} \in \overline{\mathbf{a}}$. For all $i < j$, we have $\mathbf{a}_j + \mathbf{v}_i \in B_r(\mathbf{a}_j) \subset D$ as desired.

By the above analysis, this increasing sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots)$ with $\mathbf{\Delta}_{ij} = \mathbf{v}_i$ satisfies the conditions of Lemma 6.4.1, giving the contradiction that f is not obliviously-computable. □

The results of Lemmas 6.7.7 and 6.7.9 bring us close to proving Theorem 6.7.1. From each determined region $D_1, \ldots, D_q$ we have a quilt-affine extension $g_1, \ldots, g_q : \mathbb{N}^d \to \mathbb{Z}$ that eventually dominates f. Thus for some large enough $\mathbf{n} \in \mathbb{N}^d$ we have $f(\mathbf{x}) \leq \min_k g_k(\mathbf{x})$ for all $\mathbf{x} \geq \mathbf{n}$. Furthermore, $f(\mathbf{x}) = g_k(\mathbf{x})$ if $\mathbf{x} \in D_k$ for some determined region D_k. However, it is possible $f(\mathbf{x}) < \min_k g_k(\mathbf{x})$ for any $\mathbf{x}$ that belong to an under-determined region. Since the bound $\mathbf{n}$ can be arbitrarily large, we need only consider **eventual** under-determined regions that are unbounded in all inputs:

Definition 6.7.10. *A region R is **eventual** if for any $\mathbf{n} \in \mathbb{N}^d$, there exists some $\mathbf{x} \in \mathbb{N}^d \cap R$ such that $\mathbf{x} \geq \mathbf{n}$.*

To finish the proof of Theorem 6.7.1, it remains to show how to construct a quilt-affine extension g_U from each eventual under-determined region U, where g_U eventually dominates f. We describe how to do this in Sec. 6.7.4.

6.7.4. Under-determined regions.

The key property that makes under-determined regions more difficult to handle is that, unlike Lemma 6.7.7, there is not a unique extension from an under-determined region. The key challenge of this section then is to reason about possible extensions from U to show that one can be chosen that, as in Lemma 6.7.9, eventually dominates f.

Let $U \subset \mathbb{R}_{\geq 0}$ be an under-determined eventual region. The eventual condition implies that there is some $\mathbf{v} \in \mathrm{recc}(U)$ strictly positive on all coordinates. In order to reason about U we will need to identify other regions which are **neighbors**.

6.7.4.1. *Neighbors of under-determined regions.* We now formally define neighbor by recession cone containment:

Definition 6.7.11. *Region R is a **neighbor** of an under-determined region U if* $\mathrm{recc}(U) \subseteq \mathrm{recc}(R)$.

Intuitively, region R contains all the same infinite rays as region U, thus the two regions remain “close at infinity." This relationship will constrain the function behavior on the two regions to be similar.

We will construct the extension from U by referencing the determined regions that are neighbors of U. Any under-determined eventual region will in fact have at least two determined neighbors (proved as Corollary 6.7.19 to the later Lemma 6.7.18). Geometrically, we can think of the under-determined recession cones as faces of each of the recession cones of the determined neighbors (see Fig. 6.8d).

Unlike in the proof of Lemma 6.7.7, the affine partial functions defining f within U do not need to have equal gradients. However, these gradients will be equal along directions in $\mathrm{recc}(U)$, in other words projected onto the subspace $W = \mathrm{span}(\mathrm{recc}(U))$. We now show a stronger statement, that this common gradient (projected onto W) agrees with the gradient of the extension from any determined neighbor D. Intuitively, if these gradients disagreed within W, then moving along directions in $\mathrm{recc}(U)$, the differences between f on U and on D become arbitrarily large, contradicting that f must be nondecreasing.

Lemma 6.7.12. *Let U be an under-determined eventual region. Let D be a determined neighbor of U, with unique extension $g(\mathbf{x}) = \nabla_g \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod p)$ given by Lemma 6.7.7. For any $\mathbf{u} \in U \cap \mathbb{N}^d$, consider the affine partial function $f|_{U \cap (\overline{\mathbf{u}} \mod p)}(\mathbf{x}) = \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{x} + b_{\overline{\mathbf{u}}}$ given from Lemma 6.7.3. Then for all $\mathbf{w} \in W = \mathrm{span}(\mathrm{recc}(U))$, $\mathbf{w} \cdot \nabla_{\overline{\mathbf{u}}} = \mathbf{w} \cdot \nabla_g$.*

Proof. This proof uses similar techniques to the proof of Lemma 6.7.7: with two nonequal gradients, moving far enough along a recession cone direction to contradict the fact that f is nondecreasing.

Assume toward contradiction that for some $\mathbf{u} \in U \cap \mathbb{N}^d$ and $\mathbf{w} \in W$, $\mathbf{w} \cdot \nabla_{\overline{\mathbf{u}}} \neq \mathbf{w} \cdot \nabla_g$. Then since $W = \text{span}(\text{recc}(U))$, we have $\nabla_{\overline{\mathbf{u}}} \cdot \mathbf{y} \neq \nabla_g \cdot \mathbf{y}$ for some $\mathbf{y} \in \text{recc}(U)$. Again, we can assume $\mathbf{y} \in \mathbb{N}^d$ by density of the rationals then scaling to clear denominators. Without loss of generality further assume $\nabla_{\overline{\mathbf{u}}} \cdot \mathbf{y} > \nabla_g \cdot \mathbf{y}$.

Now pick some $\mathbf{d} \in D \cap \mathbb{N}^d$ such that $\mathbf{d} \geq \mathbf{u}$, so $f(\mathbf{d}) \geq f(\mathbf{u})$ because f is nondecreasing. Then since $\mathbf{y} \in \text{recc}(U) \subset \text{recc}(D)$, for any $c \in \mathbb{N}$, $\mathbf{u} + cp\mathbf{y} \in U \cap \overline{\mathbf{u}}$ and $\mathbf{d} + cp\mathbf{y} \in D \cap \overline{\mathbf{d}}$. Thus

$$f(\mathbf{u} + cp\mathbf{y}) = f|_{U\cap\overline{\mathbf{u}}}(\mathbf{u} + cp\mathbf{y}) = \nabla_{\overline{\mathbf{u}}} \cdot (\mathbf{u} + cp\mathbf{y}) + b_{\overline{\mathbf{u}}}$$

and

$$f(\mathbf{d} + cp\mathbf{y}) = f|_{D\cap\overline{\mathbf{d}}}(\mathbf{d} + cp\mathbf{y}) = \nabla_g \cdot (\mathbf{d} + cp\mathbf{y}) + B(\overline{\mathbf{d}}).$$

Since $\nabla_{\overline{\mathbf{u}}} \cdot \mathbf{y} > \nabla_g \cdot \mathbf{y}$, for some large enough $c \in \mathbb{N}$, we have $f(\mathbf{u} + cp\mathbf{y}) > f(\mathbf{d} + cp\mathbf{y})$. But this contradicts the fact that f is nondecreasing, since $\mathbf{u} + cp\mathbf{y} \leq \mathbf{d} + cp\mathbf{y}$. □

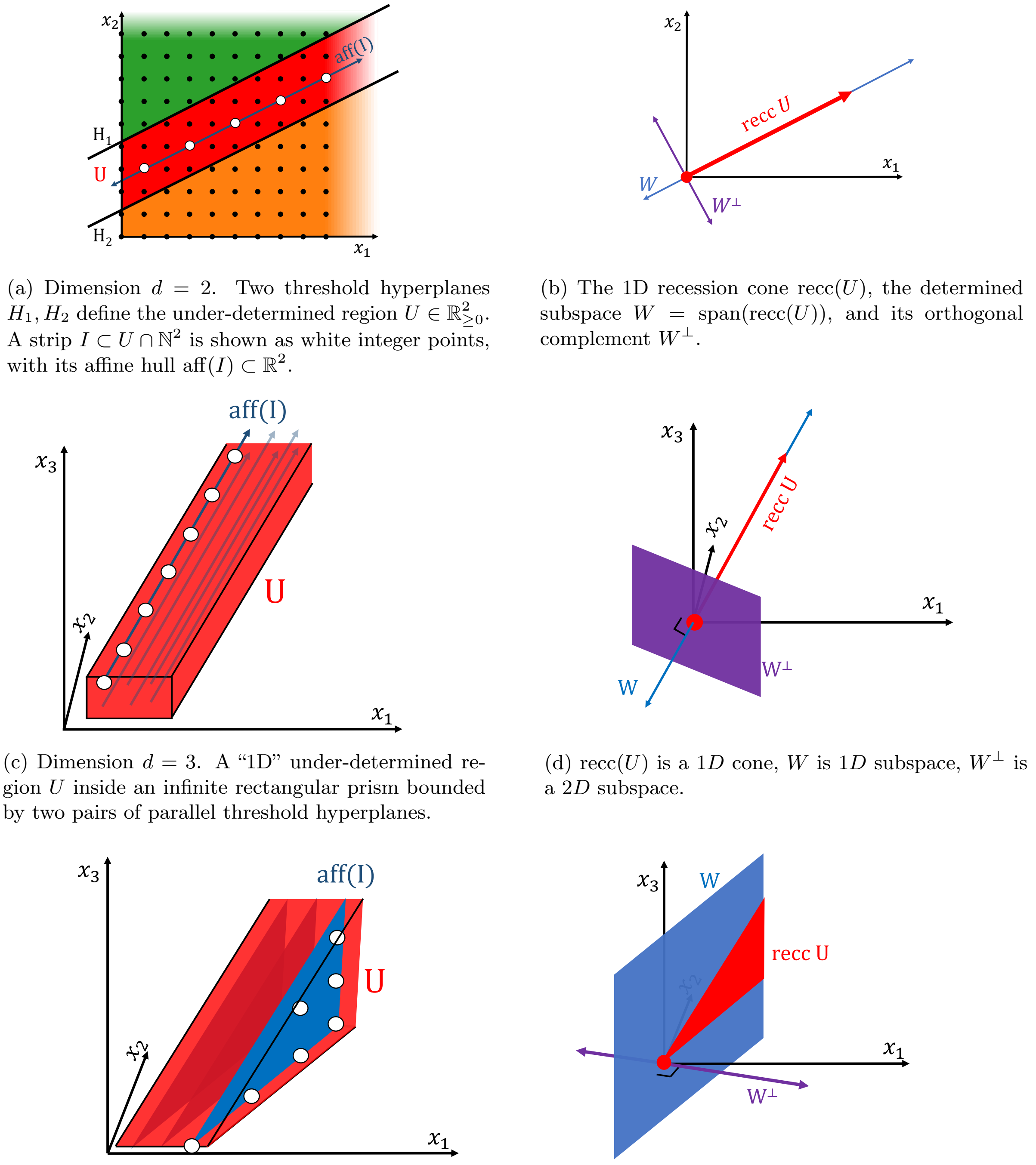
6.7.4.2. *Strips within under-determined regions.* Lemma 6.7.12 motivated the definition of $W = \text{span}(\text{recc}(U))$, which we call the **determined subspace** of U, with $1 \leq \dim W < d$. See Fig. 6.9b,6.9d,6.9f for examples.

Lemma 6.7.12 constrains the behavior of f moving from points in U within the subspace W. The region U, however, could have a finite "width" in other directions. This motivates us to separate U into "strips", partitioning its integer points to classes lying on translated versions of W:

Definition 6.7.13. *Let U be an under-determined region with $W = \text{span}(\text{recc}(U))$. The equivalence relation $\equiv_W$, where $\mathbf{x} \equiv_W \mathbf{y}$ if $\mathbf{x} - \mathbf{y} \in W$, partitions $U \cap \mathbb{N}^d$ into sets called **strips**. Thus a strip $I = \{\mathbf{x} \in U \cap \mathbb{N}^d : \mathbf{x} \equiv_W \mathbf{u}\}$ for some $\mathbf{u} \in U \cap \mathbb{N}^d$.*

For a strip $I \in \mathbb{N}^d$, we will consider the smallest affine set containing I: the **affine hull** $\text{aff}(I) \subset \mathbb{R}^d$ [74]. Note that for every $\mathbf{u} \in I$, $\text{aff}(I) = \mathbf{u} + W = \{\mathbf{u} + \mathbf{w} : \mathbf{w} \in W\}$, and $\text{aff}(I)$ is a rational affine subspace. See Fig. 6.9a,6.9c,6.9e for examples.

We next show a useful lemma about the distance from rational affine subspaces to surrounding integer points. This will be used to show there are only a finite number of strips, and will also be key to a trick in the later proof of Lemma 6.7.16.



(a) Dimension $d = 2$. Two threshold hyperplanes H_1, H_2 define the under-determined region $U \in \mathbb{R}^2_{\geq 0}$. A strip $I \subset U \cap \mathbb{N}^2$ is shown as white integer points, with its affine hull $\text{aff}(I) \subset \mathbb{R}^2$.

(b) The 1D recession cone $\text{recc}(U)$, the determined subspace $W = \text{span}(\text{recc}(U))$, and its orthogonal complement $W^\perp$.

(c) Dimension $d = 3$. A "1D" under-determined region U inside an infinite rectangular prism bounded by two pairs of parallel threshold hyperplanes.

(d) $\text{recc}(U)$ is a $1D$ cone, W is $1D$ subspace, $W^\perp$ is a $2D$ subspace.

(e) Dimension $d = 3$. A "2D" under-determined region U bounded by a pair of parallel threshold hyperplanes on the left and right side.

(f) $\text{recc}(U)$ is $2D$ cone, W is a $2D$ subspace, $w^\perp$ is a $1D$ subspace.

FIGURE 6.9. Three examples with different dimensions of an underdetermined region U, a strip I, its affine hull $\text{aff}(I)$, the recession cone $\text{recc}(U)$, and determined subspace W.

Lemma 6.7.14. *Let $A \subset \mathbb{R}^d$ be a rational affine subspace, containing some point $\mathbf{x} \in \mathbb{N}^d$. Then there exists a constant $c > 0$ such that for any period $p^* \in \mathbb{N}_+$, for all $\mathbf{y} \in (\overline{\mathbf{x}} \mod p^*)$ with $\mathbf{y} \notin A$, the distance* $\operatorname{dist}(\mathbf{y}, A) \geq cp^*$.

Proof. Let A be a rational affine subspace containing $\mathbf{x} \in \mathbb{N}^d$. Let $\mathbf{y} \in \overline{\mathbf{x}} \mod p^*$ with $\mathbf{y} \notin A$, so we can write $\mathbf{y} = \mathbf{x} + p^*\mathbf{v}$ for some $\mathbf{v} \in \mathbb{Z}^d$.

First we consider the case that A is a hyperplane, so we can write $A = \{\mathbf{z} \in \mathbb{R}^d : \mathbf{a} \cdot \mathbf{z} = b\}$ for some $\mathbf{a} \in \mathbb{Z}^d$ and $b \in \mathbb{Z}$. Then using a standard formula [67] for the distance from a point $\mathbf{y} \in \mathbb{R}^d$ to A:

$$\operatorname{dist}(\mathbf{y}, A) = \frac{|\mathbf{a} \cdot \mathbf{y} - b|}{\|\mathbf{a}\|} = \frac{|\mathbf{a} \cdot (\mathbf{x} + p^*\mathbf{v}) - b|}{\|\mathbf{a}\|} = \frac{p^*|\mathbf{a} \cdot \mathbf{v}|}{\|\mathbf{a}\|} \geq \frac{p^*}{\|\mathbf{a}\|}$$

where the inequality follows from $|\mathbf{a} \cdot \mathbf{v}| \geq 1$: $\mathbf{a} \cdot \mathbf{v} \in \mathbb{Z}$ and $\mathbf{a} \cdot \mathbf{v} \neq 0$ since $\mathbf{y} \notin A$. The desired result then holds taking $c = 1/\|\mathbf{a}\|$, which depends only on A (and not on p^*).

Finally if the rational affine space A is not a hyperplane, it is the intersection of a finite number of hyperplanes, so A is contained in some rational hyperplane H and then by the above result $\operatorname{dist}(\mathbf{y}, A) \geq \operatorname{dist}(\mathbf{y}, H) \geq cp^*$. □

Now we can show there are only a finite number of strips in each under-determined region.

Lemma 6.7.15. *The equivalence relation $\equiv_W$ partitions $U \cap \mathbb{N}^d$ into a finite number of strips.*

Proof. Consider the set of unique strips $\{I_1, I_2, \ldots\}$ each with some representative $\mathbf{u}_j \in I_j$. For each strip I_j, consider the affine hull $\operatorname{aff}(I_j) = \mathbf{u}_j + W$. These are rational affine spaces which are all parallel. For any $I_j \neq I_k$, since both $\operatorname{aff}(I_j)$ and $\operatorname{aff}(I_k)$ contain integer points, using Lemma 6.7.14 with $p = 1$ implies that $\operatorname{dist}(\operatorname{aff}(I_j), \operatorname{aff}(I_k)) \geq c$ for some constant $c > 0$. This lower bound c is the same for all j, k because the $\operatorname{aff}(I_j)$ are all parallel. We will now use the fact that the affine hulls of the strips are bounded away from each other to show there can only be finitely many strips.

Since U is a convex polyhedron, by [74] there exists a bounded polytope H such that $U = H + \operatorname{recc}(U) = \{\mathbf{h} + \mathbf{y} : \mathbf{h} \in H, \quad \mathbf{y} \in \operatorname{recc}(U)\}$. Thus each representative $\mathbf{u}_j = \mathbf{h}_j + \mathbf{y}_j$ for some $\mathbf{h}_j \in H$ and $\mathbf{y}_j \in \operatorname{recc}(U)$, so $\mathbf{h}_j = \mathbf{u}_j - \mathbf{y}_j \in \operatorname{aff}(I_j)$ (since $\mathbf{y}_j \in W$). Then $\operatorname{dist}(\mathbf{h}_j, \mathbf{h}_k) \geq c$ for all $j \neq k$. Since all $\mathbf{h}_j$ are contained in the bounded polytope H, there must be finitely many and thus finitely many strips. □

6.7.4.3. *Defining an extension from a strip of an under-determined region.* Since there are a finite number of under-determined regions, by Lemma 6.7.15 there are a finite total number of strips, so to finish proving Theorem 6.7.1 it suffices to consider each strip separately. For each strip I, we must define an extension g_I from I that eventually dominates f. The following lemma shows how to define this extension based on the extensions from the determined neighbors, if we crucially satisfy a technical condition that their gradients not all the same.

Lemma 6.7.16. *Let I be a strip of an under-determined eventual region U. Let $D_1, \ldots, D_m$ be the determined neighbors of U, with extensions $g_1, \ldots, g_m$. Assume for all $\mathbf{z} \in W^\perp$, the gradients of the extensions along $\mathbf{z}$ are not all equal: $\nabla_{g_i} \cdot \mathbf{z} \neq \nabla_{g_j} \cdot \mathbf{z}$ for some i, j. Then there exists an extension g_I from the strip I that eventually dominates f.*

Proof sketch. By Lemma 6.7.12, the gradient ∇_I of g_I must agree with the gradient from any determined neighbors in all directions within W. We will define ∇_I to be the average of the gradients of the extensions from all determined neighbors (see Fig. 6.7(d), and note that any weighted average of the gradients would work here). We crucially assume these gradients from the extensions are not all the same. (Lemma 6.7.20 shows how to handle the case that the gradients **are** all the same). This will imply that their average grows faster than the minimum (and thus grows faster than f) moving away from $\mathrm{aff}(I)$. It is then immediate that g_I will eventually dominate f sufficiently far from $\mathrm{aff}(I)$, but requires a subtle trick to define g_I to dominate f near $\mathrm{aff}(I)$.

We choose g_I to have a larger period p^* that will guarantee (via Lemma 6.7.14) that points congruent mod p^* to points in I are sufficiently far from $\mathrm{aff}(I)$ (where $g_I \geq f$). The offsets in these congruence classes mod p^* must be uniquely defined so $g_I = f$ on I. The remaining offsets are then maximized subject to the constraint that g_I is nondecreasing.

Finally, we must show g_I eventually dominates f on integer points in $\mathrm{aff}(I)$. This is only nontrivial for dimension $d \geq 3$. For example, for strip I in Fig. 6.9e, we must argue that g_I will eventually dominate f within the whole spanning plane $\mathrm{aff}(I)$ (which includes regions directly above U). We repeat the argument from Lemma 6.7.9, but now restrict our attention to $\mathrm{aff}(I)$. This works because U is “infinitely wide" within the directions of W. □

Proof. Let $D_1, \ldots, D_m$ be determined neighbors of U, each with a quilt-affine extension $g_i(\mathbf{x}) = \nabla_{g_i} \cdot \mathbf{x} + B_{g_i}(\overline{\mathbf{x}} \mod p)$. Let $\nabla_{\text{avg}} = \frac{1}{m} \sum_{i=1}^m \nabla_{g_i} \in \mathbb{Q}^d$. We will define $p^* \in \mathbb{N}_+$ and

$B^* : \mathbb{Z}^d/p^*\mathbb{Z}^d \to \mathbb{Q}$ and let the extension be

$$g_I(\mathbf{x}) = \nabla_{\text{avg}} \cdot \mathbf{x} + B^*(\overline{\mathbf{x}} \mod p^*),$$

which is quilt-affine with a potentially larger period $p^* = kp$ for some $k \in \mathbb{N}_+$, so each congruence class $(\overline{\mathbf{x}} \mod p^*) \subseteq (\overline{\mathbf{x}} \mod p)$. To ensure that g_I still has integer outputs, we pick p^* such that $p^*\nabla_{\text{avg}} \in \mathbb{N}^d$. We will show later that p^* can be chosen large enough to make g_I eventually dominate f. Let $\mathbf{r} \in I$ be a fixed reference vector we will use to define B^*.

First we show that for each $\mathbf{u} \in I$, $B^*(\overline{\mathbf{u}} \mod p^*)$ is uniquely defined so that $g_I(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in I \cap (\overline{\mathbf{u}} \mod p^*)$ (in other words there are fixed values of B^* for inputs in I that will make g_I an extension of f from I). By Lemma 6.7.3 we have affine partial function $f|_{U\cap(\overline{\mathbf{u}} \mod p)}(\mathbf{x}) = \nabla_{\overline{\mathbf{u}}}\cdot\mathbf{x}+b_{\overline{\mathbf{u}}}$ and by Lemma 6.7.12 we have $\text{proj}_W(\nabla_{\overline{\mathbf{u}}}) = \text{proj}_W(\nabla_{g_i})$ for all gradients of determined neighbor extensions g_i. Thus we also have $\text{proj}_W(\nabla_{\overline{\mathbf{u}}}) = \text{proj}_W(\nabla_{\text{avg}})$, so $\nabla_{\overline{\mathbf{u}}} \cdot \mathbf{w} = \nabla_{\text{avg}} \cdot \mathbf{w}$ for all $\mathbf{w} \in W$. We then define $B^*(\overline{\mathbf{u}} \mod p^*) = \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{r} - \nabla_{\text{avg}} \cdot \mathbf{r} + b_{\overline{\mathbf{u}}}$, which depends only on the congruence class $\overline{\mathbf{u}} \mod p$ (but doesn't depend on p^*). We can now verify that for any $\mathbf{x} \in I \cap (\overline{\mathbf{u}} \mod p^*)$, where $\mathbf{x} - \mathbf{r} \in W$ by definition of the strip I, we have

$$\begin{aligned}
g_I(\mathbf{x}) &= \nabla_{\text{avg}} \cdot \mathbf{x} + B^*(\overline{\mathbf{x}} \mod p^*) \\
&= \nabla_{\text{avg}} \cdot (\mathbf{x} - \mathbf{r}) + \nabla_{\text{avg}} \cdot \mathbf{r} + \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{r} - \nabla_{\text{avg}} \cdot \mathbf{r} + b_{\overline{\mathbf{u}}} \\
&= \nabla_{\overline{\mathbf{u}}} \cdot (\mathbf{x} - \mathbf{r}) + \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{r} + b_{\overline{\mathbf{u}}} \quad \text{since } \mathbf{x} - \mathbf{r} \in W \\
&= \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{x} + b_{\overline{\mathbf{u}}} = f|_{U\cap(\overline{\mathbf{u}} \mod p)}(\mathbf{x}) = f(\mathbf{x}).
\end{aligned}$$

Thus we have shown that $B^*(\overline{\mathbf{u}} \mod p^*)$ is uniquely defined so that $g_I(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in I \cap (\overline{\mathbf{u}} \mod p^*)$.

g_I is currently a partial function, only defined on the set $I^* = (I + p^*\mathbb{Z}^d) \cap \mathbb{N}^d$ of points congruent mod p^* to some $\mathbf{u} \in I$. For all other congruence classes $\overline{\mathbf{a}} \mod p^* \in \mathbb{Z}^d/p^*\mathbb{Z}^d$ such that $\overline{\mathbf{a}} \cap I = \emptyset$, we will define $B^*(\overline{\mathbf{a}})$ to be as large as possible while still having g_I be nondecreasing. For g_I to be nondecreasing, $g_I(\mathbf{x}) \leq \min_{\mathbf{y}\geq\mathbf{x}} g_I(\mathbf{y})$ for all $\mathbf{x} \in \mathbb{N}^d$. We maximize $B^*(\overline{\mathbf{a}})$ such that for all $\mathbf{x} \in \overline{\mathbf{a}}$,

$$g_I(\mathbf{x}) = \min_{\mathbf{y}\in I^*, \mathbf{y}\geq\mathbf{x}} g_I(\mathbf{y}).$$

Observe that since the finite differences above each $\mathbf{x}$ are periodic (as observed formally to prove Lemma 6.6.1), this required offset $B^*(\mathbf{a})$ depends only on the congruence class $\overline{\mathbf{a}}$ of $\mathbf{x}$.

Now in order to show that g_I eventually dominates f, we claim it suffices to show that g_I eventually dominates f on I^*: for some $\mathbf{n} \in \mathbb{N}^d$, $g_I(\mathbf{x}) \geq f(\mathbf{x})$ for all $\mathbf{x} \in I^*$ with $\mathbf{x} \geq \mathbf{n}$. If this holds, then for any $\mathbf{x} \notin I^*$ with $\mathbf{x} \geq \mathbf{n}$, we have $g_I(\mathbf{x}) = g_I(\mathbf{y})$ for some $\mathbf{y} \in I^*, \mathbf{y} \geq \mathbf{x}$, so

$$g_I(\mathbf{x}) = g_I(\mathbf{y}) \geq f(\mathbf{y}) \geq f(\mathbf{x})$$

showing that g_I eventually dominates f as long as g_I eventually dominates f on I^*.

We will next show g_I eventually dominates f for $\mathbf{x}$ sufficiently far from $\operatorname{aff}(I)$ by comparing g_I to the extension $g_j(\mathbf{x}) = \nabla_{g_j} \cdot \mathbf{x} + B_{g_j}(\overline{\mathbf{x}} \mod p)$ from some determined neighbor D_j. Let $\mathbf{x} \in I^*$ and let g_j be the extension of any determined neighbor D_j. Writing $\mathbf{x} = \mathbf{r} + \mathbf{w} + \mathbf{z}$ for the fixed reference $\mathbf{r} \in I$, $\mathbf{w} = \operatorname{proj}_W(\mathbf{x} - \mathbf{r}) \in W$ and $\mathbf{z} = \operatorname{proj}_{W^\perp}(\mathbf{x} - \mathbf{r}) \in W^\perp$, we have

$$\begin{aligned} g_I(\mathbf{x}) - g_j(\mathbf{x}) &= \nabla_{\text{avg}} \cdot (\mathbf{r} + \mathbf{w} + \mathbf{z}) + B^*(\overline{\mathbf{x}} \mod p^*) - \nabla_{g_j} \cdot (\mathbf{r} + \mathbf{w} + \mathbf{z}) - B_{g_j}(\overline{\mathbf{x}} \mod p) \\ &= \nabla_{\text{avg}} \cdot \mathbf{z} - \nabla_{g_j} \cdot \mathbf{z} + [\nabla_{\text{avg}} \cdot \mathbf{r} - \nabla_{g_j} \cdot \mathbf{r} + B^*(\overline{\mathbf{x}} \mod p^*) - B_{g_j}(\overline{\mathbf{x}} \mod p)]. \end{aligned}$$

Notice that the term $[\ldots]$ depends only on j and $\overline{\mathbf{x}} \mod p$ (since B^* was uniquely defined on I^* based only on $\overline{\mathbf{x}} \mod p$). Thus minimizing over all finitely many j and $\overline{\mathbf{x}} \mod p$ gives some (possibly negative) lower bound $-q \in \mathbb{Q}$ (which crucially does not depend on the choice p^*) such that

$$g_I(\mathbf{x}) - g_j(\mathbf{x}) \geq \nabla_{\text{avg}} \cdot \mathbf{z} - \nabla_{g_j} \cdot \mathbf{z} - q$$

for all $\mathbf{x} \in I^*$ and extensions g_j.

Now we use the crucial assumption that for any $\mathbf{z} \in W^\perp$, $\nabla_{g_i} \cdot \mathbf{z} \neq \nabla_{g_j} \cdot \mathbf{z}$ for some i, j. Considering first unit vectors $\mathbf{v} \in W^\perp$ with $\|\mathbf{v}\| = 1$, then there is some j minimizing $\nabla_{g_j} \cdot \mathbf{v}$ with $\nabla_{\text{avg}} \cdot \mathbf{v} - \nabla_{g_j} \cdot \mathbf{v} > 0$. We claim that there exists $\epsilon > 0$ such that $\nabla_{\text{avg}} \cdot \mathbf{v} - \nabla_{g_j} \cdot \mathbf{v} \geq \epsilon$ for all such $\mathbf{v}$ and their corresponding j. If not, then there is a sequence $(\mathbf{v}_i)$ of unit vectors with $\nabla_{\text{avg}} \cdot \mathbf{v}_i - \nabla_{g_j} \cdot \mathbf{v}_i \to 0$. Since the unit ball is compact, there must be a subsequence of $(\mathbf{v}_i)$ converging to some $\mathbf{v}$, which implies $\nabla_{\text{avg}} \cdot \mathbf{v} - \nabla_{g_j} \cdot \mathbf{v} = 0$. This completes the claim that such

$\epsilon > 0$ exists. Then as long as $\|\mathbf{z}\| \geq q/\epsilon$, we have for some j (which minimizes $\nabla_{g_j} \cdot \mathbf{z}$)

$$g_I(\mathbf{x}) - g_j(\mathbf{x}) \geq \|\mathbf{z}\| \left(\nabla_{\text{avg}} \cdot \frac{\mathbf{z}}{\|\mathbf{z}\|} - \nabla_{g_j} \cdot \frac{\mathbf{z}}{\|\mathbf{z}\|} \right) - q \geq \|\mathbf{z}\|\epsilon - q \geq 0.$$

Since $\mathbf{x} = \mathbf{r} + \mathbf{w} + \mathbf{z}$ for some $\mathbf{r} + \mathbf{w} \in \text{aff}(I)$ and $\mathbf{z} \in W^{\perp}$, we have $\|\mathbf{z}\| = \text{dist}(\mathbf{x}, \text{aff}(I))$. Thus we have shown for $\mathbf{x}$ sufficiently far from $\text{aff}(I)$, $g_I(\mathbf{x}) \geq g_j(\mathbf{x})$ for some quilt-affine g_j which itself eventually dominates f (by Lemma 6.7.9). Crucially this bound $\|\mathbf{z}\| \geq q/\epsilon$ did not depend on p^*, so we will now use Lemma 6.7.14 to choose p^* large enough that $\text{dist}(\mathbf{x}, \text{aff}(I)) \geq q/\epsilon$ for all $\mathbf{x} \in I^*$ with $\mathbf{x} \notin \text{aff}(I)$. Since such $\mathbf{x} \in (\overline{\mathbf{u}} \mod p^*)$ for some $\mathbf{u} \in I \subset \text{aff}(I)$ and $\text{aff}(I)$ is a rational affine subspace, by Lemma 6.7.14 there is some bound $c > 0$ (depending only on $\text{aff}(I)$) such that $\text{dist}(\mathbf{x}, \text{aff}(I)) \geq cp^*$. Thus we choose a large enough multiple $p^* = kp$ such that $cp^* \geq q/\epsilon$.

We have shown that $g_I(\mathbf{x})$ eventually dominates $f(\mathbf{x})$ for all $\mathbf{x} \in I^*$ with $\mathbf{x} \notin \text{aff}(I)$. It finally remains to show that g_I eventually dominates f on $\text{aff}(I)$. This is true for the same reasons as Lemma 6.7.9 (because g_I is a quilt-affine extension of f from I) following the same proof strategy. In the proof of Lemma 6.7.9 we assumed toward contradiction a sequence of "bad points", and used the fact that a determined region D contained arbitrarily large open balls to construct a contradiction sequence for Lemma 6.4.1. Now we are only showing g_I eventually dominates f on $\text{aff}(I)$, so all "bad points" would be in $\text{aff}(I)$. We can construct a contradiction sequence $(\mathbf{a}_i) \in I$ by the same argument. Since $W = \text{span}(\text{recc}(U))$, by the same argument as Lemma 6.7.5, $\text{recc}(U)$ contains arbitrarily large open balls within the subspace W. This is sufficient to ensure $(\mathbf{a}_i) \in I$, since the sequence of vectors $(\mathbf{v}_i)$ with $\mathbf{a}_i + \mathbf{v}_i$ being "bad points" are in W because all "bad points" are in $\text{aff}(I)$.

Thus g_I eventually dominates f everywhere, as desired. □

Lemma 6.7.16 crucially assumed that along any $\mathbf{z} \in W^{\perp}$, the gradients of all extensions from determined neighbors are not equal. If these gradients are all equal, there might not exist a quilt-affine extension g_I that eventually dominates f as we desired. For example, consider the function

$$f(x_1, x_2) = \begin{cases} x_1 + x_2 + 1, & \text{if } x_1 \neq x_2 \\ x_1 + x_2, & \text{if } x_1 = x_2 \end{cases} \tag{6.2}$$

which is a single affine function, depressed by 1 along the diagonal $x_1 = x_2$. f is semilinear and nondecreasing. The two determined regions where $x_1 > x_2$ and $x_1 < x_2$ have the same quilt-affine extension $((1,1)\cdot\mathbf{x}+1)$, which eventually dominates f. On the underdetermined region, which here consists of just the single strip where $x_1 = x_2$, f is strictly smaller. There does not actually exist a quilt-affine extension from this strip that eventually dominates f. (One can show directly f is not obliviously-computable by Lemma 6.4.1, with $\mathbf{a}_i = (i, 0)$ and $\mathbf{\Delta}_{ij} = (0, j)$).

6.7.4.4. *Remaining case: equal gradients from determined neighbors.* The remaining case thus serves to disallow general versions the counterexample 6.2. Lemma 6.7.16 assumed for all $\mathbf{z} \in W^{\perp}$, $\nabla_{g_i} \cdot \mathbf{z} \neq \nabla_{g_j} \cdot \mathbf{z}$ for some i, j. We will now consider the negation: that for some $\mathbf{z} \in W^{\perp}$ we have $\nabla_{g_i} \cdot \mathbf{z} = \nabla_{g_j} \cdot \mathbf{z}$ for all i, j. To proceed in this case, we will need to be able to identify the **neighbor of U in the direction of $\mathbf{z}$**.

For example, consider the under-determined eventual region 5 in Fig. 6.8c. The determined subspace W is 1D, so the orthogonal complement $W^{\perp}$ is 2D. For each $\mathbf{z} \in W^{\perp}$, the neighbor in the direction of $\mathbf{z}$ will correspond to one of the 8 other pictured regions.

We now identify which threshold hyperplanes can distinguish a region from its neighbors. Recall the threshold hyperplanes $H_i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{t}_i \cdot \mathbf{x} = h_i\}$ for $i = 1, \ldots, l$. We now show that some of these hyperplanes must be parallel to all vectors in $\mathrm{recc}(U)$. If not, we will show there is a vector in the interior of $\mathrm{recc}(U)$, so U must be a determined region.

Lemma 6.7.17. *Let U be an under-determined eventual region. Then there exists some threshold hyperplane $H_i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{t}_i \cdot \mathbf{x} = h_i\}$ such that $\mathbf{t}_i \cdot \mathbf{y} = 0$ for all $\mathbf{y} \in \mathrm{recc}(U)$.*

Proof. Assume toward contradiction that for all $\mathbf{t}_i$, there exists some $\mathbf{y}_i \in \mathrm{recc}(U)$ such that $\mathbf{t}_i \cdot \mathbf{y}_i \neq 0$, so $s_i(\mathbf{t}_i \cdot \mathbf{y}_i) > 0$, where s_i is the ith sign that defined the region U, and $s_j(\mathbf{t}_j \cdot \mathbf{y}_i) \geq 0$ for all $j = 1, \ldots, l$ since $\mathbf{y}_i \in \mathrm{recc}(U)$. Then $\mathbf{y} = \sum_{i=1}^{l} \mathbf{y}_i \in \mathrm{recc}(U)$ since $\mathrm{recc}(U)$ is closed under addition, so $s_i(\mathbf{t}_i \cdot \mathbf{y}) > 0$ for all $i = 1, \ldots, l$.

Recall that $\mathrm{recc}(U) = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : (\forall i) s_i(\mathbf{t}_i \cdot \mathbf{x}) \geq 0\}$ is a closed convex polyhedron that is the intersection of closed half-spaces. Then $\mathrm{int}(\mathrm{recc}(U)) = \{\mathbf{x} \in \mathbb{R}^d_{>0} : (\forall i) s_i(\mathbf{t}_i \cdot \mathbf{x}) > 0\}$ is the intersection of the respective open half-spaces, and we have $\mathbf{y} \in \mathrm{int}(\mathrm{recc}(U))$. Thus because $\mathrm{int}(\mathrm{recc}(U))$ is nonempty, $\mathrm{recc}(U)$ must be full-dimensional, contradicting that U is an under-determined region. □

We call such hyperplanes **neighbor-separating hyperplanes** for reasons that will be made clear shortly. If $\mathbf{t}_i \cdot \mathbf{y} = 0$ for all $\mathbf{y} \in \mathrm{recc}(U)$, we also have $\mathbf{t}_i \cdot \mathbf{y} = 0$ for all $\mathbf{y} \in W$, so by definition $\mathbf{t}_i \in W^\perp$. Then let $L_U = \{i \in \{1, \ldots, l\} \mid \mathbf{t}_i \in W^\perp\}$ be the subset of labels of all such neighbor-separating hyperplanes for U.

For example, in Fig. 6.8c, for under-determined region 5, all four hyperplanes are neighbor-separating hyperplanes. For under-determined region 6, only the pair of horizontally oriented hyperplanes are neighbor-separating hyperplanes.

Recalling Definition 6.7.2, let $S_U = \operatorname{diag}(s_1, \ldots, s_l)$ be the sign matrix that defined U. For $\mathbf{z} \in W^\perp$, define $R_\mathbf{z}$, the **neighbor of U in the direction of $\mathbf{z}$**, by a related sign matrix $S_\mathbf{z} = \operatorname{diag}(s'_1, \ldots, s'_l)$, where if $i \in L_U$ and $\operatorname{sign}(\mathbf{t}_i \cdot \mathbf{z}) = -s_i$, then let $s'_i = -s_i$, but otherwise $s'_i = s_i$ for all other $i = 1, \ldots, l$. Intuitively, for all neighbor-separating hyperplanes, $R_\mathbf{z}$ is on the same side as the direction $\mathbf{z}$, but is otherwise identical to U.

The following lemma justifies the use of the word "neighbor" in the previous definition, and further shows that such a neighbor is "more determined" (recession cone has higher dimension) than U. This allows us to reason recursively about neighbors of an under-determined region U helping us to define the extension on U, terminating in base cases of determined regions.

Lemma 6.7.18. *Let U be an under-determined eventual region with $W = \operatorname{span}(\operatorname{recc}(U))$, let $\mathbf{z} \in W^\perp$, and let the region $R_\mathbf{z}$ be the neighbor of U in the direction of $\mathbf{z}$. Then $R_\mathbf{z} \cap \mathbb{N}^d$ is nonempty, $R_\mathbf{z}$ is a neighbor of U and furthermore* $\dim \operatorname{recc}(U) < \dim \operatorname{recc}(R_\mathbf{z})$.

Proof. We first show that $R_\mathbf{z}$ is a neighbor of U, i.e., that

$$\operatorname{recc}(U) = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : S_U T\mathbf{x} \geq 0\} \subset \operatorname{recc}(R_\mathbf{z}) = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : S_\mathbf{z} T\mathbf{x} \geq 0\}.$$

Let $\mathbf{x} \in \operatorname{recc}(U)$. Then for all $i \in L_U$, $s_i \cdot \mathbf{x} = 0$, so $s'_i \cdot \mathbf{x} = 0$, and otherwise $s'_i = s_i$, so $S_\mathbf{z} T\mathbf{x} \geq 0$. This implies $\mathbf{x} \in \operatorname{recc}(R_\mathbf{z})$, i.e., $\operatorname{recc}(U) \subset \operatorname{recc}(R_\mathbf{z})$, proving the claim that $R_\mathbf{z}$ is a neighbor of U.

Next we argue that $\dim \operatorname{recc}(U) < \dim \operatorname{recc}(R_\mathbf{z})$. Now similar to the proof of Lemma 6.7.17, for each $i \notin L_U$ there exists some $\mathbf{y}_i \in \operatorname{recc}(U)$ such that $\mathbf{t}_i \cdot \mathbf{y}_i \neq 0$, so $s_i(\mathbf{t}_i \cdot \mathbf{y}_i) > 0$. Then taking $\mathbf{y} = \sum_{i \notin L_U} \mathbf{y}_i$ we have $\mathbf{y} \in \operatorname{recc}(U)$, and also $s_i(\mathbf{t}_i \cdot \mathbf{y}) > 0$ for all $i \notin L_U$.[14]

We will show for some $\epsilon > 0$, $\mathbf{y} + \epsilon\mathbf{z} \in \operatorname{recc}(R_\mathbf{z})$, which will imply $\mathbf{z} \in \operatorname{span}(\operatorname{recc}(R_\mathbf{z}))$ so $\dim \operatorname{recc}(R_\mathbf{z}) \geq \dim \operatorname{recc}(U) + 1$ since $\mathbf{z} \in W^\perp$. Intuitively, to show this, we perturb the vector $\mathbf{y}$ by a slight amount in the direction of $\mathbf{z}$ to be on the correct side of all neighbor-separating

[14] In fact $\mathbf{y}$ can be shown to be in the **relative interior** of $\operatorname{recc}(U)$, where the relative interior is the interior within the affine hull. [74]

hyperplanes, while remaining on the same side of all other hyperplanes. Formally, for all $i \in L_U$, we have $s'_i(\mathbf{t}_i \cdot \mathbf{z}) \geq 0$ by construction of s'_i. This might not hold for $i \notin L_U$, but in that case $s'_i(\mathbf{t}_i \cdot \mathbf{y}) > 0$. Thus we can pick some small enough $\epsilon > 0$ such that $s'_i(\mathbf{t}_i \cdot (\mathbf{y} + \epsilon\mathbf{z})) \geq 0$ for all $i \notin L_U$. For $i \in L_U$, we have $\mathbf{t}_i \cdot \mathbf{y} = 0$ (by definition of L_U since $\mathbf{y} \in \operatorname{recc}(U)$), so we also have $s'_i(\mathbf{t}_i \cdot (\mathbf{y} + \epsilon\mathbf{z})) \geq 0$ and thus $\mathbf{y} + \epsilon\mathbf{z} \in \operatorname{recc}(R_\mathbf{z})$. This concludes the claim that $\dim \operatorname{recc}(U) < \dim \operatorname{recc}(R_\mathbf{z})$.

Finally, we argue that $R_\mathbf{z} \cap \mathbb{N}^d$ is nonempty, so the region $R_\mathbf{z}$ is meaningfully defined. We can further assume that the vector $\mathbf{y}^+ = \mathbf{y} + \epsilon\mathbf{z} \in \mathbb{N}^d$ (again by density assuming that the pieces are rational and then scaling up to clear denominators). Now consider a point $\mathbf{u} \in U \cap \mathbb{N}^d$, so $S_U(T\mathbf{u} - \mathbf{h}) \geq 0$, and consider moving along $\mathbf{y}^+$. For all i such that $s_i \neq s'_i$, we have $s'_i(\mathbf{t}_i \cdot \mathbf{y}^+) > 0$. Thus for all sufficiently large constants c, $S_\mathbf{z}(T(\mathbf{u} + c\mathbf{y}^+) - \mathbf{h}) \geq 0$ so $\mathbf{u} + c\mathbf{y}^+ \in R_\mathbf{z}$. Intuitively, the path from U along the vector $\mathbf{y}^+$ will eventually remain in the region $R_\mathbf{z}$. This shows that the neighbor of U in the direction of $\mathbf{z}$ is well-defined. □

For under-determined eventual region U, by repeatedly applying Lemma 6.7.18 using directions $\pm\mathbf{z}$ for any $\mathbf{z} \in W^\perp$, we can show that determined neighbors must actually exist:

Corollary 6.7.19. *An under-determined eventual region U has at least 2 determined neighbors.*

Proof. Since U is an under-determined region, there exists some nonzero $\mathbf{z} \in W^\perp$. Applying Lemma 6.7.18 to $\mathbf{z}$ and $-\mathbf{z}$ will give regions $R_\mathbf{z}$ and $R_{-\mathbf{z}}$, which are the neighbors in the directions $\mathbf{z}$ and $-\mathbf{z}$, respectively. Note that the definition of $R_\mathbf{z}$ implies that $R_{-\mathbf{z}}$ will have a different sign matrix and thus be a separate region. Furthermore, the dimension of their recession cones is strictly larger than that of U. If $R_\mathbf{z}$ and $R_{-\mathbf{z}}$ are not determined, we can repeat this argument taking $R_\mathbf{z}$ as the under-determined region. Since the recession cone dimension is bounded by d, we will eventually find a pair of determined neighbors. □

We are now ready to consider the remaining case left after Lemma 6.7.16, when all determined neighbor gradients agree along some $\mathbf{z} \in W^\perp$.

Lemma 6.7.20. *Let I be a strip of an under-determined eventual region U. Let $D_1, \ldots, D_m$ be the determined neighbors of U, with extensions $g_1, \ldots, g_m$. Assume there exists $\mathbf{z} \in W^\perp$ such that the gradients of the extensions along $\mathbf{z}$ are all equal: $\nabla_{g_i} \cdot \mathbf{z} = \nabla_{g_j} \cdot \mathbf{z}$ for all i, j. Let $R_\mathbf{z}$ be the neighbor of U in the direction $\mathbf{z}$, with extension $g_\mathbf{z}$. Then $g_\mathbf{z}$ is also an extension from I: $g_\mathbf{z}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in I$, so taking $g_\mathbf{z}$ gives an extension from I that eventually dominates f.*

Proof. Let I be a strip of under-determined eventual region U, with extensions $g_1, \ldots, g_m$ from determined neighbors $D_1, \ldots, D_m$ respectively. Let $\mathbf{z} \in W^\perp$ such that $\nabla_{g_i} \cdot \mathbf{z} = \nabla_{g_j} \cdot \mathbf{z}$ for all i, j. We will consider the regions $R_{\mathbf{z}}$ and $R_{-\mathbf{z}}$ that are the neighbors of U in the directions $\mathbf{z}$ and $-\mathbf{z}$. Recall by Lemma 6.7.18 that $\dim \operatorname{recc}(U) < \dim \operatorname{recc}(R_{\pm \mathbf{z}})$. The proof will proceed by induction on the codimension: $d - \dim \operatorname{recc}(U)$. For U with codimension 1, $R_{\pm \mathbf{z}}$ must be determined regions with unique extensions. In general, $R_{\pm \mathbf{z}}$ could be under-determined, but will have lower codimension, so by the inductive hypothesis we assume $R_{\pm \mathbf{z}}$ have extensions that eventually dominate f (considering this Lemma alongside Lemma 6.7.16). Thus there exist quilt-affine extensions $g_{\mathbf{z}}$ and $g_{-\mathbf{z}}$ (from $R_{\mathbf{z}}$ and $R_{-\mathbf{z}}$) that eventually dominate f. Note these may have a larger period p^* as used in the proof of Lemma 6.7.16. We assume $g_{\mathbf{z}}$ and $g_{-\mathbf{z}}$ have common period p^* by taking the least common multiple if necessary. Thus we can write $g_{\mathbf{z}}(\mathbf{x}) = \nabla_{g_{\mathbf{z}}} \cdot \mathbf{x} + B_{g_{\mathbf{z}}}(\overline{\mathbf{x}} \mod p^*)$ and $g_{-\mathbf{z}}(\mathbf{x}) = \nabla_{g_{-\mathbf{z}}} \cdot \mathbf{x} + B_{g_{-\mathbf{z}}}(\overline{\mathbf{x}} \mod p^*)$.

Now by assumption $\nabla_{g_i} \cdot \mathbf{z} = \nabla_{g_j} \cdot \mathbf{z}$ for any of U's determined neighbors D_i, D_j. Also by Lemma 6.7.12, $\operatorname{proj}_W(\nabla_{g_i}) = \operatorname{proj}_W(\nabla_{g_j})$. Thus all determined gradients agree along $\operatorname{span}(W, \mathbf{z})$. The regions $R_{\pm \mathbf{z}}$ are either determined, or their determined neighbors are among $D_1, \ldots, D_m$ (by transitivity of the neighbor relation). Regardless, we can say $\operatorname{proj}_{\operatorname{span}(W,\mathbf{z})}(\nabla_{g_{\mathbf{z}}}) = \operatorname{proj}_{\operatorname{span}(W,\mathbf{z})}(\nabla_{g_{-\mathbf{z}}})$. For this proof, we will consider the affine space $A = \operatorname{aff}(I) + \operatorname{span}(\mathbf{z}) = \{\mathbf{i} + \mathbf{w} + c\mathbf{z} : \mathbf{i} \in I, \mathbf{w} \in W, c \in \mathbb{R}\}$ containing all points reachable from I by vectors in $\operatorname{span}(W, \mathbf{z})$. We now claim that $g_{\mathbf{z}}(\mathbf{x}) = g_{-\mathbf{z}}(\mathbf{x})$ for all $\mathbf{x} \in A \cap \mathbb{N}^d$. The gradients along directions in A ($\operatorname{span}(W, \mathbf{z})$) were already shown to be equal. Thus if $g_{\mathbf{z}}(\mathbf{x}) \neq g_{-\mathbf{z}}(\mathbf{x})$ (without loss of generality $g_{\mathbf{z}}(\mathbf{x}) < g_{-\mathbf{z}}(\mathbf{x})$), then $g_{\mathbf{z}}(\mathbf{y}) < g_{-\mathbf{z}}(\mathbf{y})$ for all congruent $\mathbf{y} \in \overline{\mathbf{x}} \mod p^*$. However, $g_{-\mathbf{z}}$ is an extension of f from $R_{-\mathbf{z}}$, so we have $g_{\mathbf{z}}(\mathbf{y}) < f(\mathbf{y})$ for all $\mathbf{y} \in R_{-\mathbf{z}} \cap (\overline{\mathbf{x}} \mod p^*)$. This contradicts the fact that $g_{\mathbf{z}}$ eventually dominates f, and completes the claim that $g_{\mathbf{z}}(\mathbf{x}) = g_{-\mathbf{z}}(\mathbf{x})$ on $A \cap \mathbb{N}^d$.

Thus we must have $g_{\mathbf{z}}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in A \cap R_{\mathbf{z}} \cap \mathbb{N}^d$ (since $g_{\mathbf{z}}$ is an extension from $R_{\mathbf{z}}$) and $\mathbf{x} \in A \cap R_{-\mathbf{z}} \cap \mathbb{N}^d$ (since $g_{\mathbf{z}} = g_{-\mathbf{z}}$, the extension from $R_{-\mathbf{z}}$). We now show also that $g_{\mathbf{z}}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in I$. Assume toward contradiction that $g_{\mathbf{z}}(\mathbf{u}) \neq f(\mathbf{u})$ for some $\mathbf{u} \in I$. By Lemma 6.7.3 we have affine partial function $f|_{U \cap (\overline{\mathbf{u}} \mod p)}(\mathbf{x}) = \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{x} + b_{\overline{\mathbf{u}}}$ and by Lemma 6.7.12 we have $\operatorname{proj}_W(\nabla_{\overline{\mathbf{u}}}) = \operatorname{proj}_W(\nabla_{g_{\mathbf{z}}})$. Then for any $\mathbf{x} \in I \cap (\overline{\mathbf{u}} \mod p^*)$, $\mathbf{x} = \mathbf{u} + \mathbf{w}$ for some $\mathbf{w} \in W$ by definition of I, so

$$\begin{aligned} g_{\mathbf{z}}(\mathbf{x}) - f(\mathbf{x}) &= \nabla_{g_{\mathbf{z}}} \cdot (\mathbf{u}+\mathbf{w}) + B_{g_{\mathbf{z}}}(\overline{\mathbf{u}} \mod p^*) - \nabla_{\overline{\mathbf{u}}} \cdot (\mathbf{u}+\mathbf{w}) - b_{\overline{\mathbf{u}}} \\ &= \underbrace{(\nabla_{g_{\mathbf{z}}} \cdot \mathbf{w} - \nabla_{\overline{\mathbf{u}}} \cdot \mathbf{w})}_{=0 \text{ since } \mathbf{w} \in W} + (\nabla_{g_{\mathbf{z}}} \cdot \mathbf{u} + B_{g_{\mathbf{z}}}(\overline{\mathbf{u}} \mod p^*)) - (\nabla_{\overline{\mathbf{u}}} \cdot \mathbf{u} + b_{\overline{\mathbf{u}}}) \\ &= g_{\mathbf{z}}(\mathbf{u}) - f(\mathbf{u}) \qquad (6.3) \end{aligned}$$

In other words, if $g_{\mathbf{z}}(\mathbf{u}) \neq f(\mathbf{u})$, they are also unequal for all $\mathbf{x}$ within the strip I on the entire congruence class $\overline{\mathbf{u}}$.

If we had $g_{\mathbf{z}}(\mathbf{u}) < f(\mathbf{u})$, then $g_{\mathbf{z}}(\mathbf{x}) < f(\mathbf{x})$ for all $\mathbf{x} \in I \cap (\overline{\mathbf{u}} \mod p^*)$ by 6.3, which contradicts that $g_{\mathbf{z}}$ eventually dominates f.

The other case is that $g_{\mathbf{z}}(\mathbf{u}) > f(\mathbf{u})$, so again by 6.3, we have $g_{\mathbf{z}}(\mathbf{x}) > f(\mathbf{x})$ for all $\mathbf{x} \in I \cap (\overline{\mathbf{u}} \mod p^*)$. (This is the behavior of our example 6.2, which will be shown to not be obliviously-computable by the following general argument). Here, similar to the proof of Lemma 6.7.9, we will apply Lemma 6.4.1 by creating a contradiction sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in \mathbb{N}^d$ such that for all $i < j$ there exists some $\mathbf{\Delta}_{ij} \in \mathbb{N}^d$ with

$$f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j).$$

To do this, we will find a sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in A \cap R_{\mathbf{z}} \cap (\overline{\mathbf{u}} \mod p^*)$, so $g_{\mathbf{z}}(\mathbf{a}_i) = f(\mathbf{a}_i)$ for all i. We will then find another sequence $(\mathbf{v}_1, \mathbf{v}_2, \ldots) \in \mathbb{N}^d$ such that $\mathbf{a}_i + \mathbf{v}_i \in I \cap (\overline{\mathbf{u}} \mod p^*)$ for all i, implying $g_{\mathbf{z}}(\mathbf{a}_i + \mathbf{v}_i) > f(\mathbf{a}_i + \mathbf{v}_i)$. Also, we need that for all $i < j$, $\mathbf{a}_i + \mathbf{v}_j \in A \cap R_{-\mathbf{z}} \cap (\overline{\mathbf{u}} \mod p^*)$, so $g_{\mathbf{z}}(\mathbf{a}_i + \mathbf{v}_j) = f(\mathbf{a}_i + \mathbf{v}_j)$. If these are both true, then choosing $\mathbf{\Delta}_{ij} = \mathbf{v}_j$ for $i < j$ gives

$$\begin{aligned} & f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) \\ &\quad = g_{\mathbf{z}}(\mathbf{a}_i + \mathbf{v}_j) - g_{\mathbf{z}}(\mathbf{a}_i) \\ &\quad = g_{\mathbf{z}}(\mathbf{a}_j + \mathbf{v}_j) - g_{\mathbf{z}}(\mathbf{a}_j) \qquad \text{since } g_{\mathbf{z}} \text{ is quilt-affine and } \mathbf{a}_i \equiv \mathbf{a}_j \mod p^* \\ &\quad > f(\mathbf{a}_j + \mathbf{v}_j) - f(\mathbf{a}_j) = f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j). \end{aligned}$$

Lemma 6.4.1 then tells us that f is not obliviously-computable, a contradiction. It remains to show that such sequences $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in R_{\mathbf{z}}$ and $(\mathbf{v}_1, \mathbf{v}_2, \ldots) \in \mathbb{N}^d$ can be found satisfying $\mathbf{a}_i + \mathbf{v}_i \in I \cap (\overline{\mathbf{u}} \mod p^*)$ for all i, and for all $i < j$, $\mathbf{a}_i + \mathbf{v}_j \in A \cap R_{-\mathbf{z}} \cap (\overline{\mathbf{u}} \mod p^*)$.

Now from the proof of Lemma 6.7.18, we take the same $\mathbf{y} \in \operatorname{recc}(U)$ and perturbed $\mathbf{y}^+ = \mathbf{y} + \epsilon\mathbf{z} \in \operatorname{recc}(R_\mathbf{z})$ that were defined in that proof. Recall we showed for $\mathbf{x} \in U$, for all large enough c, $\mathbf{x} + c\mathbf{y}^+ \in R_\mathbf{z}$. Likewise, we also have $\mathbf{y}^- = \mathbf{y} - \epsilon\mathbf{z} \in \operatorname{recc}(R_{-\mathbf{z}})$ (taking ϵ small enough to work for both $R_\mathbf{z}$ and $R_{-\mathbf{z}}$) and we can assume (by density of rationals and scaling up denominators) that $\mathbf{y}^+, \mathbf{y}^- \in \mathbb{N}^d$.

Pick $c \in \mathbb{N}$ large enough that $\mathbf{u} + cp^*\mathbf{y}^+ \in R_\mathbf{z}$ and $\mathbf{u} + cp^*\mathbf{y}^- \in R_{-\mathbf{z}}$. Then for all $i \in \mathbb{N}_+$, let $\mathbf{a}_i = \mathbf{u} + icp^*\mathbf{y}^+$ and $\mathbf{v}_i = icp^*\mathbf{y}^-$. Since $\mathbf{y}^+, \mathbf{y}^- \in \operatorname{span}(W, \mathbf{z})$, we have $\mathbf{a}_i \in A$ for all i, and the multiple of p^* ensures all points are in $(\overline{\mathbf{u}} \mod p^*)$ as desired. Finally, we can check that

$$\mathbf{a}_i + \mathbf{v}_i = \mathbf{u} + icp^*\mathbf{y}^+ + icp^*\mathbf{y}^- = \mathbf{u} + 2icp^*\mathbf{y} \in I$$

since $\mathbf{y}^+ + \mathbf{y}^- = 2\mathbf{y} \in \operatorname{recc}(U)$. Also, for $i < j$ we have

$$\mathbf{a}_i + \mathbf{v}_j = \mathbf{u} + icp^*\mathbf{y}^+ + jcp^*\mathbf{y}^- = \mathbf{u} + (j-i)cp^*\mathbf{y}^- + icp^*\mathbf{y}$$

Note that $j - i \geq 1$, $\mathbf{u} + (j-i)cp^*\mathbf{y}^- \in R_{-\mathbf{z}}$, and $icp^*\mathbf{y} \in \operatorname{recc}(R_{-\mathbf{z}})$. Thus

$$\mathbf{a}_i + \mathbf{v}_j = \mathbf{u} + (j-i)cp^*\mathbf{y}^- + icp^*\mathbf{y} \in R_{-\mathbf{z}}$$

as required.

Thus Lemma 6.4.1 gives a contradiction that f cannot be obliviously-computable. We reached this contradiction by assuming that $g_\mathbf{z}(\mathbf{u}) \neq f(\mathbf{u})$ for some $\mathbf{u} \in I$. Thus we conclude that $g_\mathbf{z}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in I$, so $g_\mathbf{z}$ is an extension from I that eventually dominates f. □

For any strip I of an under-determined eventual region U, one of the cases from Lemmas 6.7.16 or 6.7.20 applies to show there exists an extension from I that eventually dominates f. There are only finitely many such strips (Lemma 6.7.15), so alongside the unique extensions from the determined regions (Lemmas 6.7.7 and 6.7.9), we have identified a finite collection $g_1, \ldots, g_m$ of quilt-affine functions to complete the proof of Theorem 6.7.1.

6.8. Comparison to continuous case

In [61], the authors classified the power of output-oblivious **continuous** CRNs to stably compute real-valued functions $f : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$. We can generalize to also consider such functions by introducing the following natural scaling:

Definition 6.8.1. *For a function $f : \mathbb{N}^d \to \mathbb{N}$, the ∞-**scaling** $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ is given by*

$$\hat{f}(\mathbf{z}) = \lim_{c \to \infty} \frac{f(\lfloor c\mathbf{z} \rfloor)}{c}.$$

Note this limit may not exist for arbitrary $f : \mathbb{N}^d \to \mathbb{N}$, but it will exist for all obliviously-computable f.

The next theorem shows that in this scaling limit, our output-oblivious function class exactly corresponds to the real-valued function class from [**61**] (see Fig. 6.4b).

Theorem 6.8.2. *If $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, then the ∞-scaling $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ is obliviously-computable by a continuous CRN. Furthermore, every function obliviously-computable by a continuous CRN is the ∞-scaling of some function obliviously-computable by a discrete CRN.*

Proof. To prove the first statement, let $f : \mathbb{N}^d \to \mathbb{N}$ be obliviously-computable. We will show the ∞-scaling $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ satisfies the main classification of [**61**]: that $\hat{f}$ is superadditive, positive-continuous, and piecewise rational-linear.

First we prove that for any quilt-affine $g : \mathbb{N}^d \to \mathbb{Z}$, the ∞-scaling $\hat{g}$ is nonnegative and rational-linear. From Definition 6.5.1, we can express $g(\mathbf{x}) = \nabla_g \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod p)$ for $\nabla_g \in \mathbb{Q}^d_{\geq 0}$ and $B : \mathbb{Z}^d / p\mathbb{Z}^d \to \mathbb{Q}$. Then for any $\mathbf{z} \in \mathbb{R}^d_{\geq 0}$,

$$\hat{g}(\mathbf{z}) = \lim_{c \to \infty} \frac{\nabla_g \cdot \lfloor c\mathbf{z} \rfloor + B(\overline{\lfloor c\mathbf{z} \rfloor} \mod p)}{c} = \nabla_g \cdot \mathbf{z},$$

since B is bounded. Because $\nabla_g \in \mathbb{Q}^d_{\geq 0}$, $\hat{g}$ is nonnegative and rational-linear.

Now by the eventually-min condition (ii) of Theorem 6.5.2, there exists quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ such that $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$ for all $\mathbf{x} \geq \mathbf{n}$. Then for any $\mathbf{z} \in \mathbb{R}^d_{>0}$, $\lfloor c\mathbf{z} \rfloor \geq \mathbf{n}$ for large enough c, so

$$\hat{f}(\mathbf{z}) = \lim_{c \to \infty} \frac{f(\lfloor c\mathbf{z} \rfloor)}{c} = \lim_{c \to \infty} \frac{\min_k(g_k(\lfloor c\mathbf{z} \rfloor))}{c} = \min_k \left(\lim_{c \to \infty} \frac{g_k(\lfloor c\mathbf{z} \rfloor)}{c} \right) = \min_k(\hat{g}_k(\mathbf{z})), \tag{6.4}$$

where we pass the limit through the min function because min is continuous. Since $\hat{g}_k(\mathbf{z}) = \nabla_{g_k} \cdot \mathbf{z}$ are all rational-linear, on the domain $\mathbb{R}_{>0}$, $\hat{f}(\mathbf{z}) = \min_k(\hat{g}_k(\mathbf{z}))$ is continuous and piecewise rational-linear.

We will now generalize this argument to show on the full domain $\mathbb{R}^d_{\geq 0}$, $\hat{f}$ is piecewise rational-linear and **positive-continuous**: for each subset $S \subseteq \{1,\ldots,d\}$, $\hat{f}$ is continuous on domain $D_S = \{\mathbf{z} \in \mathbb{R}^d_{\geq 0} : \mathbf{z}(i) = 0 \iff i \in S\}$. Fix any such S. By repeatedly applying the recursive condition (iii) of Theorem 6.5.2, the fixed-input restriction $f_{[(\forall i \in S)\ \mathbf{x}(i) \to 0]}$ fixing input coordinates in S to 0, is obliviously-computable. Then by the eventually-min condition (ii), there exists quilt-affine $g^S_1, \ldots, g^S_m$ and $\mathbf{n} \in \mathbb{N}^d$ such that $f_{[(\forall i \in S)\ \mathbf{x}(i) \to 0]}(\mathbf{x}) = \min_k(g^S_k(\mathbf{x}))$ for all $\mathbf{x} \geq \mathbf{n}$. Because $f_{[(\forall i \in S)\ \mathbf{x}(i) \to 0]}(\mathbf{x})$ does not actually depend on the input coordinates $i \in S$, we only actually require $\mathbf{x}(i) \geq \mathbf{n}(i)$ for all $i \notin S$. Now let $\mathbf{z} \in D_S$, so $\mathbf{z}(i) = 0 \iff i \in S$. Then for large enough c, $\lfloor c\mathbf{z} \rfloor(i) \geq \mathbf{n}(i)$ for all $i \notin S$, so

$$f(\lfloor c\mathbf{z} \rfloor) = f_{[(\forall i \in S)\ \mathbf{x}(i) \to 0]}(\lfloor c\mathbf{z} \rfloor) = \min_k \left(g^S_k(\lfloor c\mathbf{z} \rfloor)\right).$$

Now repeating equation 6.4, we have $\hat{f}(\mathbf{z}) = \min_k(\hat{g}^S_k(\mathbf{z}))$, so $\hat{f}$ is continuous and piecewise rational-linear on D_S. This holds for all $S \subseteq \{1,\ldots,d\}$, so $\hat{f}$ is positive-continous and piecewise rational linear.

It remains to show that $\hat{f}$ must be superadditive: $\hat{f}(\mathbf{a}) + \hat{f}(\mathbf{b}) \leq \hat{f}(\mathbf{a}+\mathbf{b})$ for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d_{\geq 0}$. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d_{\geq 0}$ with $\mathbf{a}+\mathbf{b} \in D_S$ for a domain D_S defined as above. Then

$$\hat{f}(\mathbf{a}+\mathbf{b}) = \min_k(\hat{g}^S_k(\mathbf{a}+\mathbf{b})) = \hat{g}^S_i(\mathbf{a}+\mathbf{b}) = \hat{g}^S_i(\mathbf{a}) + \hat{g}^S_i(\mathbf{b})$$

for some minimizing rational-linear $\hat{g}^S_i$. It remains to show $\hat{g}^S_i(\mathbf{a}) \geq \hat{f}(\mathbf{a})$ (and by symmetry $\hat{g}^S_i(\mathbf{b}) \geq \hat{f}(\mathbf{b})$). This is immediate if $\mathbf{a} \in D_S$ since $\hat{f}(\mathbf{a}) = \min_k(\hat{g}^S_k)$. Otherwise if $\mathbf{a} \notin D_S$, assume toward contradiction that $\hat{f}(\mathbf{a}) > \hat{g}^S_k(\mathbf{a})$. Then for some small enough $\epsilon > 0$, we also have $\hat{f}(\mathbf{a}) > \hat{g}^S_k(\mathbf{a}+\epsilon\mathbf{b})$. Observing that $\mathbf{a}+\epsilon\mathbf{b} \in D_S$, then $\hat{g}^S_k(\mathbf{a}+\epsilon\mathbf{b}) \geq \hat{f}(\mathbf{a}+\epsilon\mathbf{b})$. But then $\hat{f}(\mathbf{a}) > \hat{f}(\mathbf{a}+\epsilon\mathbf{b})$, a contradiction since $\hat{f}$ must be nondecreasing as the ∞-scaling of the nondecreasing function f.

Thus $\hat{f}$ is semilinear, positive-continuous, and piecewise rational-linear as desired.

Next, to prove the second statement, let $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ be any semilinear, positive-continuous, and piecewise rational-linear function. We will show that there exists some obliviously-computable $f : \mathbb{N}^d \to \mathbb{N}$ such that its ∞-scaling is $\hat{f}$.

On each domain $D_S = \{\mathbf{z} \in \mathbb{R}^d_{\geq 0} : \mathbf{z}(i) = 0 \iff i \in S\}$ for $S \subseteq \{1,\ldots,d\}$, $\hat{f}|_{D_S}$ is superadditive, continuous, and piecewise rational-linear. By Lemma 8 in [61], $\hat{f}|_{D_S}$ can be written as the minimum of a finite number of rational linear functions $\hat{g}^S_k(\mathbf{z}) = \nabla_{g_k} \cdot \mathbf{z}$. For each $\hat{g}^S_k$, we will identify a quilt-affine g_k^S with gradient ∇_{g_k}. In particular, we can define $g_k^S : \mathbb{N}^d \to \mathbb{N}$ for all $\mathbf{x} \in \mathbb{N}^d$ by $g_k^S(\mathbf{x}) = \lfloor \nabla_{g_k} \cdot \mathbf{x} \rfloor$, which will be quilt-affine.

Now for all S and integer $\mathbf{x} \in D_S \cap \mathbb{N}^d$, define $f(\mathbf{x}) = \min_k(g_k^S(\mathbf{x}))$. From the above proof it follows that $\hat{f}$ is the ∞-scaling of f. It is also straightforward to verify that f is obliviously-computable by satisfying Theorem 6.5.2. f is nondecreasing, satisfying condition ((i)), because $\hat{f}$ was semilinear and thus nondecreasing. f satisfies eventually-min condition ((ii)) since for all $\mathbf{x} \geq (1, \ldots, 1)$, $\mathbf{x} \in D_\emptyset = \mathbb{R}^d_{>0}$, so $f(\mathbf{x}) = \min_k(g_k^\emptyset(\mathbf{x}))$. For all other $S \neq \emptyset$, the fixed-input restriction $f_{[(\forall i \in S)\ \mathbf{x}(i) \to 0]}(\mathbf{x}) = \min_k(g_k^S(\mathbf{x}))$. It follows that f satisfies recursive condition ((iii)), because any fixed-input restriction will be eventually-min of quilt-affine functions.

Thus any function $\hat{f}$ obliviously-computable by a continuous CRN is the ∞-scaling limit of some f obliviously-computable by a discrete CRN. □

6.9. Leaderless one-dimensional case

In this section we show a characterization of 1D functions $f : \mathbb{N} \to \mathbb{N}$ that are obliviously-computable **without** a leader.

Note that the following observation applies to any number of dimensions. We say $f : \mathbb{N}^d \to \mathbb{N}$ is **superadditive** if $f(\mathbf{x}) + f(\mathbf{y}) \leq f(\mathbf{x} + \mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{N}^d$.

Observation 6.9.1. *Every f obliviously-computable by a leaderless CRN is superadditive.*

Proof. Let $\mathcal{C}$ be a leaderless CRN stably computing f. We prove the observation by contrapositive. Suppose f is not superadditive. Then there are $\mathbf{x}, \mathbf{z} \in \mathbb{N}^d$ such that $f(\mathbf{x}) + f(\mathbf{z}) > f(\mathbf{x} + \mathbf{z})$. Recall $\mathbf{I}_\mathbf{w}$ is the initial configuration of $\mathcal{C}$ representing input $\mathbf{w}$. Let $\alpha_\mathbf{x}$ be a sequence of reactions applied to $\mathbf{I}_\mathbf{x}$ to produce $f(\mathbf{x})$ copies of Y, and let $\alpha_\mathbf{z}$ be a sequence of reactions applied to $\mathbf{I}_\mathbf{z}$ to produce $f(\mathbf{z})$ copies of Y.

Since $\mathcal{C}$ is leaderless, $\mathbf{I}_{\mathbf{x}+\mathbf{z}} = \mathbf{I}_\mathbf{x} + \mathbf{I}_\mathbf{z}$. Thus we can apply $\alpha_\mathbf{x}$ to $\mathbf{I}_{\mathbf{x}+\mathbf{z}}$, followed by $\alpha_\mathbf{z}$, producing $f(\mathbf{x}) + f(\mathbf{z})$ copies of Y. Since this is greater than $f(\mathbf{x} + \mathbf{z})$, to stably compute f, $\mathcal{C}$ must have a reaction consuming Y, so it is not output-oblivious. Since $\mathcal{C}$ was arbitrary, f cannot be obliviously-computable. □

This added condition of superadditivity gives us the 1D leaderless characterization.

Theorem 6.9.2. *For any $f : \mathbb{N} \to \mathbb{N}$, f is obliviously-computable by a leaderless CRN $\iff$ f is semilinear and superadditive.*

Proof. $\implies$: By Lemma 6.2.7 and Observation 6.9.1.

$\impliedby$: If f is superadditive, then f is also nondecreasing (since $f(x+1) \geq f(x) + f(1) \geq f(x)$). Then as in the Proof of Theorem 6.3.1, f is eventually quilt-affine, so there exist $n \in \mathbb{N}$, period $p \in \mathbb{N}_+$, and finite differences $\delta_{\overline{0}}, \ldots, \delta_{\overline{p-1}} \in \mathbb{N}$, such that for all $x \geq n$, $f(x+1) - f(x) = \delta_{(\overline{x} \mod p)}$. Also, without loss of generality assume p divides n, so $\overline{n} \mod p = \overline{0}$.

The new CRN construction is motivated by trying to simply remove the leader species L from the construction used in Theorem 6.3.1. Recall that set of reactions was

$$\begin{aligned}
L &\to f(0)Y + L_0 \\
L_i + X &\to [f(i+1) - f(i)]Y + L_{i+1} \qquad \text{for all } i = 0, \ldots, n-2 \\
L_{n-1} + X &\to [f(n) - f(n-1)]Y + P_{\overline{n}} \\
P_{\overline{a}} + X &\to \delta_{\overline{a}} Y + P_{\overline{a+1}} \qquad \text{for all } \overline{a} = \overline{0}, \ldots, \overline{p-1}.
\end{aligned}$$

Since f is superadditive, we must have $f(0) = 0$. We then remove the species L and L_0, and the two reaction that contain them, and add the first reaction

$$X \to f(1)Y + L_1$$

If this reaction only occurred once, this would still correctly compute f. Otherwise, however, there will be multiple "auxiliary leader species" from $\{L_1, \ldots, L_{n-1}, P_{\overline{0}}, \ldots, P_{\overline{p-1}}\}$ in the system. To correctly compute f, we must introduce pairwise reactions between these species that reduce the count of auxiliary leaders and add a corrective difference.

For all $i, j \in \{1, \ldots, n-1\}$, the reaction between L_i and L_j is

$$L_i + L_j \to D_{i,j} Y + \begin{cases} L_{i+j} \text{ if } i + j < n \\ P_{\overline{i+j}} \text{ if } i + j \geq n \end{cases}$$

where $D_{i,j} := f(i+j) - f(i) - f(j) \geq 0$ by superadditivity, and is the difference between how much output Y was released in the reactions that produced L_i and L_j and how much should have been produced from the input that led to L_i and L_j.

We have similar reactions between L_i and P_j for all $i \in \{1, \ldots, n-1\}$ and $\overline{a} \in \{\overline{0}, \ldots, \overline{p-1}\}$

$$L_i + P_{\overline{a}} \to D_{i,\overline{a}} Y + P_{\overline{i+a}}$$

where $D_{i,\overline{a}} := f(i+n+a) - f(i) - f(n+a) \geq 0$ by superadditivity. The reaction sequences that produced L_i consumed i copies of input X, and those that produced $P_{\overline{a}}$ consumed $n+a+kp$ for some $k \in \mathbb{N}$, so we have undercounted by $f(i+n+a+kp) - f(i) - f(n+a+kp) = D_{i,\overline{a}}$ since the periodic differences cancel.

Finally, the reactions between $P_{\overline{a}}$ and $P_{\overline{b}}$ for all $\overline{a}, \overline{b} \in \{\overline{0}, \ldots, \overline{p-1}\}$ are

$$P_{\overline{a}} + P_{\overline{b}} \to D_{\overline{a},\overline{b}} Y + P_{\overline{a+b}}$$

where $D_{\overline{a},\overline{b}} := f(n+a+n+b) - f(n+a) - f(n+b) \geq 0$ by superadditivity, and this gives the corrective difference in output by a similar argument.

Note that the rest of the reactions used in Theorem 6.3.1 are not strictly necessary, since if all input X undergoes the first reaction $X \to f(1)Y + L_1$, the corrective difference reactions will then reduce the count of auxiliary leader species down to 1, while outputting the correct differences to produce precisely $f(x)$ output. □

6.10. Conclusion

This work left open the question of the computational power of output-oblivious CRNs without an initial leader. A leaderlessly-obliviously-computable function must be superadditive, which is a strictly stronger condition than being nondecreasing. The continuous result [61] had the same restriction of superadditivity, so our "scaling limit" reduction to their function class (Theorem 6.8.2) shows our main function class is already "almost superadditive." We also showed in the 1D case, $f : \mathbb{N} \to \mathbb{N}$ is leaderlessly-obliviously-computable if and only if f is semilinear and superadditive (Theorem 6.9.2).

The more recent result of [111] resolved this question in the higher dimensional case. There it was shown that adding superadditivity as a condition to our full result (Theorem 6.5.2) gives an exact classification of leaderlessly-obliviously-computable functions $f : \mathbb{N}^d \to \mathbb{N}$. Moreover, they give an alternative description of the classification of obliviously-computable functions with a leader. This is based on a novel definition of **well-ordered quilt-affine functions**. Using this definition, they give a simplified version of our main characterization (Theorem 6.5.2), that is now just a minimum of a finite number of nondecreasng well-ordered quilt-affine functions. Eliminating the recursive characterization needed in our Theorem 6.5.2 then makes the CRN construction (our Lemma 6.6.2) much simpler, which is crucial for making the argument then work in a leaderless setting.

Bibliography

[1] https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/majority.ipynb.

[2] https://github.com/eftekhari-mhs/population-protocols/tree/master/Exact_Majority.

[3] https://github.com/eftekhari-mhs/population-protocols/tree/master/Exact_Majority/animated_simulations.

[4] Aldous, D. J. When knowing early matters: Gossip, percolation and nash equilibria. In *Prokhorov and contemporary probability theory*. Springer, 2013, pp. 3–27.

[5] Alistarh, D., Aspnes, J., Eisenstat, D., Gelashvili, R., and Rivest, R. L. Time-space trade-offs in population protocols. In *SODA 2017: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), SIAM, pp. 2560–2579.

[6] Alistarh, D., Aspnes, J., and Gelashvili, R. Space-optimal majority in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (2018), SIAM, pp. 2221–2239.

[7] Alistarh, D., Attiya, H., Gilbert, S., Giurgiu, A., and Guerraoui, R. Fast randomized test-and-set and renaming. In *DISC 2010: International Symposium on Distributed Computing* (2010), Springer, pp. 94–108.

[8] Alistarh, D., Denysyuk, O., Rodrigues, L., and Shavit, N. Balls-into-leaves: Sub-logarithmic renaming in synchronous message-passing systems. In *PODC 2014: Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing* (2014), pp. 232–241.

[9] Alistarh, D., Dudek, B., Kosowski, A., Soloveichik, D., and Uznański, P. Robust detection in leak-prone population protocols. In *DNA 2017: International Conference on DNA Computing and Molecular Programming* (2017), Springer, pp. 155–171.

[10] Alistarh, D., and Gelashvili, R. Polylogarithmic-time leader election in population protocols. In *ICALP 2015: 42nd International Colloquium on Automata, Languages, and Programming* (2015), vol. 9135 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 479 – 491.

[11] Alistarh, D., and Gelashvili, R. Recent algorithmic advances in population protocols. *ACM SIGACT News 49*, 3 (2018), 63–73.

[12] Alistarh, D., Gelashvili, R., and Rybicki, J. Fast graphical population protocols. *arXiv preprint arXiv:2102.08808* (2021).

[13] Alistarh, D., Gelashvili, R., and Vojnović, M. Fast and exact majority in population protocols. In *PODC 2015: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing* (2015), ACM, pp. 47–56.

[14] Alistarh, D., Töpfer, M., and Uznański, P. Robust comparison in population protocols. In *PODC 2021: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (2021).

[15] Amir, T., Aspnes, J., Doty, D., Eftekhari, M., and Severson, E. Message complexity of population protocols. In *DISC 2020: 34th International Symposium on Distributed Computing* (Dagstuhl, Germany, 2020), H. Attiya, Ed., vol. 179 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 6:1–6:18.

[16] Amir, T., Aspnes, J., and Lazarsfeld, J. Approximate majority with catalytic inputs. *arXiv preprint arXiv:2009.08847* (2020).

[17] Angluin, D., Aspnes, J., Chan, M., Fischer, M. J., Jiang, H., and Peralta, R. Stably computable properties of network graphs. In *1st IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)* (June 2005), vol. 3560 of *LNCS*, Springer-Verlag, pp. 63–74.

[18] Angluin, D., Aspnes, J., and Chen, D. A population protocol for binary signaling consensus. Tech. Rep. YALEU/DCS/TR-1527, Yale University Department of Computer Science, 2016.

[19] Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M., and Peralta, R. Computation in networks of passively mobile finite-state sensors. *Distributed Computing 18* (2006), 235–253. Preliminary version appeared in PODC 2004.

[20] Angluin, D., Aspnes, J., and Eisenstat, D. Stably computable predicates are semilinear. In *PODC 2006: Proceedings of the twenty-fifth Annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 2006), ACM Press, pp. 292–299.

[21] Angluin, D., Aspnes, J., and Eisenstat, D. Fast computation by population protocols with a leader. *Distributed Computing 21*, 3 (Sept. 2008), 183–199. Preliminary version appeared in DISC 2006.

[22] Angluin, D., Aspnes, J., and Eisenstat, D. A simple population protocol for fast robust approximate majority. *Distributed Computing 21*, 2 (July 2008), 87–102.

[23] Angluin, D., Aspnes, J., Eisenstat, D., and Ruppert, E. The computational power of population protocols. *Distributed Computing 20*, 4 (November 2007), 279–304.

[24] Angluin, D., Aspnes, J., Fischer, M. J., and Jiang, H. Self-stabilizing population protocols. In *Principles of Distributed Systems*. Springer, 2006, pp. 103–117.

[25] Aspnes, J., Beauquier, J., Burman, J., and Sohier, D. Time and Space Optimal Counting in Population Protocols. In *20th International Conference on Principles of Distributed Systems (OPODIS 2016)* (2017), vol. 70, pp. 13:1–13:17.

[26] Aspnes, J., and Ruppert, E. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science 93* (2007), 98–117.

[27] Bailey, N. T., et al. *The mathematical theory of infectious diseases and its applications*. No. 2nd ediition. Charles Griffin & Company Ltd 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[28] Beauquier, J., Blanchard, P., and Burman, J. Self-stabilizing leader election in population protocols over arbitrary communication graphs. In *OPODIS* (2013), pp. 38–52.

[29] Beauquier, J., Burman, J., Claviere, S., and Sohier, D. Space-optimal counting in population protocols. In *DISC 2015: International Symposium on Distributed Computing* (2015), Springer, pp. 631–646.

[30] Beauquier, J., Burman, J., Clavière, S., and Sohier, D. Space-optimal counting in population protocols. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings* (2015), Y. Moses, Ed., vol. 9363 of *Lecture Notes in Computer Science*, Springer, pp. 631–646.

[31] Beauquier, J., Clement, J., Messika, S., Rosaz, L., and Rozoy, B. Self-stabilizing counting in mobile sensor networks with a base station. In *Distributed Computing* (2007), Springer Berlin Heidelberg, pp. 63–76.

[32] Belleville, A., Doty, D., and Soloveichik, D. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP 2017: 44th International Colloquium on Automata, Languages, and Programming* (2017), vol. 80 of *LIPIcs*, pp. 141:1–141:14.

[33] Ben-Nun, S., Kopelowitz, T., Kraus, M., and Porat, E. An $O(\log^{3/2} n)$ parallel time population protocol for majority with $O(\log n)$ states. In *PODC 2020: Proceedings of the 39th Symposium on Principles of Distributed Computing* (2020), pp. 191–199.

[34] Berenbrink, P., Czumaj, A., Steger, A., and Vöcking, B. Balanced allocations: The heavily loaded case. *SIAM Journal on Computing 35*, 6 (2006), 1350–1385.

[35] Berenbrink, P., Elsässer, R., Friedetzky, T., Kaaser, D., Kling, P., and Radzik, T. A population protocol for exact majority with $O(\log^{5/3} n)$ stabilization time and $\Theta(\log n)$ states. In *DISC 2018: Proceedings of the 32nd International Symposium on Distributed Computing* (2018), vol. 10, pp. 1—18.

[36] Berenbrink, P., Elsässer, R., Friedetzky, T., Kaaser, D., Kling, P., and Radzik, T. Time-space trade-offs in population protocols for the majority problem. *Distributed Computing* (2020).

[37] Berenbrink, P., Friedetzky, T., Giakkoupis, G., and Kling, P. Efficient plurality consensus, or: The benefits of cleaning up from time to time. In *43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)* (2016).

[38] Berenbrink, P., Friedetzky, T., Kaaser, D., and Kling, P. Tight & simple load balancing. In *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Rio de Janeiro, Brazil, May 20-24, 2019* (2019), IEEE, pp. 718–726.

[39] Berenbrink, P., Friedetzky, T., Kaaser, D., and Kling, P. Tight and simple load balancing. In *IPDPS 2019: IEEE International Parallel and Distributed Processing Symposium* (2019), pp. 718–726.

[40] Berenbrink, P., Giakkoupis, G., and Kling, P. Optimal time and space leader election in population protocols. In *STOC 2020: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (New York, NY, USA, 2020), STOC 2020, Association for Computing Machinery, p. 119–129.

[41] Berenbrink, P., Hammer, D., Kaaser, D., Meyer, U., Penschuck, M., and Tran, H. Simulating population protocols in sub-constant time per interaction. In *ESA 2020: 28th Annual European Symposium on Algorithms* (2020), vol. 173, pp. 16:1–16:22.

[42] Berenbrink, P., Kaaser, D., Kling, P., and Otterbach, L. Simple and Efficient Leader Election. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)* (2018), vol. 61, pp. 9:1–9:11.

[43] Berenbrink, P., Kaaser, D., and Radzik, T. On counting the population size. In *PODC 2019: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing* (New York, NY, USA, 2019), Association for Computing Machinery, p. 43–52.

[44] Bilke, A., Cooper, C., Elsässer, R., and Radzik, T. Brief announcement: Population protocols for leader election and exact majority with $O(\log^2 n)$ states and $O(\log^2 n)$ convergence time. In *PODC 2017: Proceedings of the ACM Symposium on Principles of Distributed Computing* (2017), ACM, pp. 451–453.

[45] Blondin, M., Esparza, J., Genest, B., Helfrich, M., and Jaax, S. Succinct population protocols for presburger arithmetic. *arXiv preprint arXiv:1910.04600* (2019).

[46] Blondin, M., Esparza, J., and Jaax, S. Large flocks of small birds: on the minimal size of population protocols. In *35th Symposium on Theoretical Aspects of Computer Science* (2018).

[47] Blondin, M., Esparza, J., Jaax, S., and Kučera, A. Black ninjas in the dark: Formal analysis of population protocols. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (New York, NY, USA, 2018), LICS '18, Association for Computing Machinery, p. 1–10.

[48] Bournez, O., Chalopin, J., Cohen, J., Koegler, X., and Rabie, M. Computing with pavlovian populations. In *OPODIS 2011: International Conference On Principles Of Distributed Systems* (2011), Springer, pp. 409–420.

[49] Bournez, O., Chalopin, J., Cohen, J., Koegler, X., and Rabie, M. Population protocols that correspond to symmetric games. *International Journal of Unconventional Computing 9* (2013).

[50] Bower, J. M., and Bolouri, H. *Computational modeling of genetic and biochemical networks.* MIT press, 2001.

[51] Boyd, D. W., and Steele, J. M. Random exchanges of information. *Journal of Applied Probability 16*, 3 (1979), 657–661.

[52] Burman, J., Beauquier, J., and Sohier, D. Space-optimal naming in population protocols. In *DISC'19* (2019), J. Suomela, Ed., vol. 146, pp. 9:1–9:16.

[53] Burman, J., Chen, H.-L., Chen, H.-P., Doty, D., Nowak, T., Severson, E., and Xu, C. Time-optimal self-stabilizing leader election in population protocols. In *PODC 2021: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (2021).

[54] Cai, S., Izumi, T., and Wada, K. How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model. *Theory of Computing Systems 50*, 3 (2012), 433–445.

[55] Cao, Y., Gillespie, D. T., and Petzold, L. R. Efficient step size selection for the tau-leaping simulation method. *The Journal of chemical physics 124*, 4 (2006), 044109.

[56] Cardelli, L. Strand algebras for DNA computing. *Natural Computing 10*, 1 (2011), 407–428.

[57] Cardelli, L., and Csikász-Nagy, A. The cell cycle switch computes approximate majority. *Scientific Reports 2* (2012).

[58] Cardelli, L., Kwiatkowska, M., and Laurenti, L. Stochastic analysis of chemical reaction networks using linear noise approximation. *Biosystems 149* (2016), 26–33. Selected papers from the Computational Methods in Systems Biology 2015 conference.

[59] Cardoza, E., Lipton, R. J., and Meyer, A. R. Exponential space complete problems for Petri nets and commutative semigroups (preliminary report). In *STOC 1976: Proceedings of the 8th annual ACM Symposium on Theory of Computing* (1976), ACM, pp. 50–54.

[60] Casteigts, A., Raskin, M., Renken, M., and Zamaraev, V. Sharp thresholds in random simple temporal graphs, 2020.

[61] Chalk, C., Kornerup, N., Reeves, W., and Soloveichik, D. Composable rate-independent computation in continuous chemical reaction networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 18*, 1 (2021), 250–260. Special issue of invited papers from CMSB 2018.

[62] Chatzigiannakis, I., Michail, O., Nikolaou, S., Pavlogiannis, A., and Spirakis, P. G. Passively mobile communicating machines that use restricted space. *Theoretical Computer Science 412*, 46 (October 2011), 6469–6483.

[63] Chen, H.-L., Doty, D., and Soloveichik, D. Deterministic function computation with chemical reaction networks. *Natural Computing 13*, 4 (2013), 517–534. Special issue of invited papers from DNA 2012.

[64] Chen, H.-P., and Chen, H.-L. Self-stabilizing leader election. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing* (New York, NY, USA, 2019), PODC '19, Association for Computing Machinery, p. 53–59.

[65] Chen, H.-P., and Chen, H.-L. Self-stabilizing leader election in regular graphs. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (New York, NY, USA, 2020), PODC '20, Association for Computing Machinery, p. 210–217.

[66] Chen, Y.-J., Dalchau, N., Srinivas, N., Phillips, A., Cardelli, L., Soloveichik, D., and Seelig, G. Programmable chemical controllers made from DNA. *Nature Nanotechnology 8*, 10 (2013), 755–762.

[67] Cheney, W., and Kincaid, D. *Linear Algebra: Theory and Applications*. Jones and Bartlett Publishers, 2010.

[68] Chugg, B., Condon, A., and Hashemi, H. Output-oblivious stochastic chemical reaction networks. In *OPODIS 2018: Proceedings of the 22nd International Conference on Principles of Distributed Systems* (2018).

[69] Condon, A., Hajiaghayi, M., Kirkpatrick, D., and Maňuch, J. Approximate majority analyses using tri-molecular chemical reaction networks. *Natural Computing 19*, 1 (2020), 249–270.

[70] Cummings, R., Doty, D., and Soloveichik, D. Probability 1 computation with chemical reaction networks. *Natural Computing 15*, 2 (2016), 245–261. Special issue of invited papers from DNA 2014.

[71] Czerner, P., and Esparza, J. Lower bounds on the state complexity of population protocols. In *PODC 2021: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (2021).

[72] Czerwiński, W., and Orlikowski, L. Reachability in vector addition systems is ackermann-complete. *arXiv preprint arXiv:2104.13866* (2021).

[73] Daley, D. J., and Kendall, D. G. Stochastic rumours. *IMA Journal of Applied Mathematics 1*, 1 (1965), 42–55.

[74] De Loera, J. A., Hemmecke, R., and Köppe, M. *Algebraic and geometric ideas in the theory of discrete optimization*, vol. 14. SIAM, 2013.

[75] Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., and Ruppert, E. When birds die: Making population protocols fault-tolerant. In *DCOSS* (2006), pp. 51–66.

[76] Diakite, L. H., and Yu, L. Energy and bandwidth efficient wireless sensor communications for improving the energy efficiency of the air interface for wireless sensor networks. In *2013 IEEE Third International Conference on Information Science and Technology (ICIST)* (March 2013), pp. 1426–1429.

[77] Diamadi, Z., and Fischer, M. J. A simple game for the study of trust in distributed systems. *Wuhan University Journal of Natural Sciences 6*, 1 (Mar 2001), 72–82.

[78] Dickson, L. E. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics 35*, 4 (1913), 413–422.

[79] Dijkstra, E. W. Self-stabilizing systems in spite of distributed control. *Commun. of the ACM 17*, 11 (Nov. 1974), 643–644.

[80] Dolev, S. *Self-stabilization*. MIT press, 2000.

[81] Dolev, S., Israeli, A., and Moran, S. Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Comput. 7*, 1 (1993), 3–16.

[82] Doty, D. Timing in chemical reaction networks. In *SODA 2014: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms* (2014), SIAM, pp. 772–784.

[83] Doty, D., and Eftekhari, M. Efficient size estimation and impossibility of termination in uniform dense population protocols. In *PODC 2019: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing* (2019), pp. 34–42.

[84] Doty, D., Eftekhari, M., Gąsieniec, L., Severson, E., Stachowiak, G., and Uznański, P. A stable majority population protocol using logarithmic time and states. *arXiv e-prints*, arXiv:2012.15800 (Dec. 2020). https://arxiv.org/abs/2012.15800.

[85] Doty, D., Eftekhari, M., Michail, O., G. Spirakis, P., and Theofilatos, M. Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time. In *DISC 2018: 32nd International Symposium on Distributed Computing* (2018).

[86] Doty, D., and Hajiaghayi, M. Leaderless deterministic chemical reaction networks. *Natural Computing 14*, 2 (2015), 213–223. Preliminary version appeared in DNA 2013.

[87] Doty, D., and Severson, E. ppsim: A software package for efficiently simulating and visualizing population protocols. In *CMSB 2021: Proceedings of the 19th International Conference on Computational Methods in Systems Biology* (2021).

[88] Doty, D., and Soloveichik, D. Stable leader election in population protocols requires linear time. *Distributed Computing 31*, 4 (2018), 257–271. Special issue of invited papers from DISC 2015.

[89] Draief, M., and Vojnović, M. Convergence speed of binary interval consensus. *SIAM Journal on control and Optimization 50*, 3 (2012), 1087–1109.

[90] Drmota, M. The height of increasing trees. *Annals of Combinatorics 12*, 4 (2009), 373–402.

[91] Drmota, M. *Random Trees: An Interplay between Combinatorics and Probability.* Springer, Heidelberg, 2009.

[92] Dubhashi, D., and Panconesi, A. *Concentration of Measure for the Analysis of Randomized Algorithms*, 1st ed. Cambridge University Press, USA, 2009.

[93] Dudek, B., and Kosowski, A. Universal protocols for information dissemination using emergent signals. In *STOC 2018: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018), pp. 87–99.

[94] Elsässer, R., and Radzik, T. Recent results in population protocols for exact majority and leader election. *Bulletin of EATCS 3*, 126 (2018).

[95] Érdi, P., and Tóth, J. *Mathematical models of chemical reactions: theory and applications of deterministic and stochastic models.* Manchester University Press, 1989.

[96] Fanti, G., Holden, N., Peres, Y., and Ranade, G. Communication cost of consensus for nodes with limited memory. *Proceedings of the National Academy of Sciences 117*, 11 (2020), 5624–5630.

[97] Fischer, M. J., and Jiang, H. Self-stabilizing leader election in networks of finite-state anonymous agents. In *OPODIS* (2006), pp. 395–409.

[98] Gąsieniec, L., Hamilton, D., Martin, R., Spirakis, P. G., and Stachowiak, G. Deterministic population protocols for exact majority and plurality. In *OPODIS 2016: 20th International Conference on Principles of Distributed Systems* (2017), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[99] Gasieniec, L., and Stachowiak, G. Fast space optimal leader election in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (2018), pp. 2653–2667.

[100] Gąsieniec, L., Stachowiak, G., and Uznański, P. Almost logarithmic-time space optimal leader election in population protocols. In *SPAA 2019: 31st ACM Symposium on Parallelism in Algorithms and Architectures* (2019), pp. 93–102.

[101] Ghaffari, M., and Parter, M. A polylogarithmic gossip algorithm for plurality consensus. In *PODC 2016: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing* (2016), pp. 117–126.

[102] Gibson, M. A., and Bruck, J. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A 104*, 9 (2000), 1876–1889.

[103] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry 81 (25)* (1977), 2340 – 2361.

[104] Gillespie, D. T. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications 188*, 1-3 (1992), 404–425.

[105] Gillespie, D. T. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics 115*, 4 (2001), 1716–1733.

[106] Gillespie, D. T. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem. 58* (2007), 35–55.

[107] GillesPy2. https://github.com/StochSS/GillesPy2.

[108] Guerraoui, R., and Ruppert, E. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *ICALP (2)* (2009), pp. 484–495.

[109] Haase, C. A survival guide to presburger arithmetic. *ACM SIGLOG News 5*, 3 (2018), 67–82.

[110] Haigh, J. Random exchanges of information. *Journal of Applied Probability 18*, 3 (1981), 743–746.

[111] Hashemi, H., Chugg, B., and Condon, A. Composable computation in leaderless, discrete chemical reaction networks. In *26th International Conference on DNA Computing and Molecular Programming (DNA 26)* (2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[112] Holst, L. Extreme value distributions for random coupon collector and birthday problems. *Extremes 4*, 2 (2001), 129–145.

[113] Izumi, T. On space and time complexity of loosely-stabilizing leader election. In *SIROCCO* (2015), vol. 9439 of *Lecture Notes in Computer Science*, Springer, pp. 299–312.

[114] Izumi, T., Kinpara, K., Izumi, T., and Wada, K. Space-efficient self-stabilizing counting population protocols on mobile sensor networks. *Theoretical Computer Science 552* (2014), 99–108.

[115] Janson, S. Tail bounds for sums of geometric and exponential variables. *Statistics and Probability Letters 135* (April 2018), 1–6.

[116] Joag-Dev, K., and Proschan, F. Negative association of random variables with applications. *Ann. Statist. 11*, 1 (03 1983), 286–295.

[117] Johnson, D., Stack, T., Fish, R., Flickinger, D. M., Stoller, L., Ricci, R., and Lepreau, J. Mobile emulab: A robotic wireless and sensor network testbed. In *INFOCOM* (2006), IEEE.

[118] Johnson, R., Dong, Q., and Winfree, E. Verifying chemical reaction network implementations: a bisimulation approach. *Theoretical Computer Science 765* (2019), 3–46.

[119] Karp, R. M., and Miller, R. E. Parallel program schemata. *Journal of Computer and System Sciences 3*, 2 (1969), 147–195.

[120] Kosowski, A., and Uznanski, P. Brief announcement: Population protocols are fast. In *PODC 2018: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing* (2018), ACM, pp. 475–477.

[121] Kurtz, T. G. The relationship between stochastic and deterministic models for chemical reactions. *The Journal of Chemical Physics 57*, 7 (1972), 2976–2978.

[122] Lathrop, J. I., Lutz, J. H., Lutz, R. R., Potter, H. D., and Riley, M. R. Population-induced phase transitions and the verification of chemical reaction networks. In *DNA 26: 26th International Conference on DNA Computing and Molecular Programming* (Dagstuhl, Germany, 2020), C. Geary and M. J. Patitz, Eds., vol. 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 5:1–5:17.

[123] Leroux, J. The reachability problem for petri nets is not primitive recursive. *arXiv preprint arXiv:2104.12695* (2021).

[124] Liggett, T. M. *Interacting particle systems*, vol. 276. Springer Science & Business Media, 2012.

[125] Lu, R., Lin, X., Zhu, H., Liang, X., and Shen, X. Becan: A bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 23*, 1 (Jan 2012), 32–43.

[126] Mayr, E. W., and Meyer, A. R. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics 46*, 3 (1982), 305–329.

[127] Mertzios, G. B., Nikoletseas, S. E., Raptopoulos, C. L., and Spirakis, P. G. Determining majority in networks with local interactions and very small local memory. In *International Colloquium on Automata, Languages, and Programming* (2014), Springer, pp. 871–882.

[128] Michail, O., Chatzigiannakis, I., and Spirakis, P. G. *New Models for Population Protocols.* N. A. Lynch (Ed), Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2011.

[129] Michail, O., Chatzigiannakis, I., and Spirakis, P. G. Naming and counting in anonymous unknown dynamic networks. In *SSS* (2013), pp. 281–295.

[130] Mitzenmacher, M. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems 12*, 10 (2001), 1094–1104.

[131] Mizoguchi, R., Ono, H., Kijima, S., and Yamashita, M. On space complexity of self-stabilizing leader election in mediated population protocol. *Distributed Computing 25*, 6 (2012), 451–460.

[132] Mocquard, Y., Anceaume, E., Aspnes, J., Busnel, Y., and Sericola, B. Counting with population protocols. In *14th IEEE International Symposium on Network Computing and Applications* (2015), pp. 35–42.

[133] Mocquard, Y., Anceaume, E., and Sericola, B. Optimal proportion computation with population protocols. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)* (2016), IEEE, pp. 216–223.

[134] Mocquard, Y., Sericola, B., Robert, S., and Anceaume, E. Analysis of the propagation time of a rumour in large-scale distributed systems. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)* (2016), IEEE, pp. 264–271.

[135] Mocquard, Y., Sericola, B., Robin, F., and Anceaume, E. Stochastic analysis of average based distributed algorithms.

[136] Moon, J. W. Random exchanges of information. *Nieuw Archief voor Wiskunde 20* (1972), 246—-249.

[137] Peres, Y., Talwar, K., and Wieder, U. Graphical balanced allocations and the (1+ β)-choice process. *Random Structures & Algorithms 47*, 4 (2015), 760–775.

[138] Perron, E., Vasudevan, D., and Vojnovic, M. Using three states for binary consensus on complete graphs. In *IEEE INFOCOM 2009* (April 2009), pp. 2527–2535.

[139] Petri, C. A. Communication with automata. Tech. rep., DTIC Document, 1966.

[140] Polastre, J., Hill, J. L., and Culler, D. E. Versatile low power media access for wireless sensor networks. In *SenSys* (2004), ACM, pp. 95–107.

[141] ppsim Python package.

source code: https://github.com/UC-Davis-molecular-computing/ppsim

API documentation: https://ppsim.readthedocs.io/

Python package for installation via pip: https://pypi.org/project/ppsim/, 2021.

[142] Qian, L., Soloveichik, D., and Winfree, E. Efficient Turing-universal computation with DNA polymers. In *DNA 2010: Proceedings of The Sixteenth International Meeting on DNA Computing and Molecular Programming* (2010), vol. 6518 of *Lecture Notes in Computer Science*, Springer.

[143] Rathinam, M., and El Samad, H. Reversible-equivalent-monomolecular tau: A leaping method for "small number and stiff" stochastic chemical systems. *Journal of Computational Physics 224*, 2 (2007), 897–923.

[144] Rockafeller, T. *Convex Analysis*. Princeton University Press, 1970, ch. 8.

[145] Sanft, K. R., Wu, S., Roh, M., Fu, J., Rone, K. L., and Petzold, L. R. Stochkit2: software for discrete stochastic simulation of biochemical systems with events. *Bioinformatics 27*, 17 (2011), 501–522.

[146] Severson, E., Haley, D., and Doty, D. Composable computation in discrete chemical reaction networks. *Distributed Computing* (2020). Special issue of invited papers from PODC 2019.

[147] Shah, D. *Gossip algorithms*. Now Publishers Inc, 2009.

[148] Shin, S. W., Thachuk, C., and Winfree, E. Verifying chemical reaction network implementations: A pathway decomposition approach. *Theoretical Computer Science 765* (2019), 67–96.

[149] Slepoy, A., Thompson, A. P., and Plimpton, S. J. A constant-time kinetic monte carlo algorithm for simulation of large biochemical reaction networks. *The journal of chemical physics 128*, 20 (2008), 05B618.

[150] Soloveichik, D. Robust stochastic chemical reaction networks and bounded tau-leaping. *Journal of Computational Biology 16*, 3 (2009), 501–522.

[151] Soloveichik, D., Cook, M., Winfree, E., and Bruck, J. Computation with finite stochastic chemical reaction networks. *Natural Computing 7* (2008), 615–633.

[152] Soloveichik, D., Seelig, G., and Winfree, E. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences 107* (2010), 5393–5398.

[153] Srinivas, N., Parkin, J., Seelig, G., Winfree, E., and Soloveichik, D. Enzyme-free nucleic acid dynamical systems. *Science 358*, 6369 (2017), eaal2052.

[154] Stanley, R. P., et al. An introduction to hyperplane arrangements. *Geometric combinatorics 13*, 389-496 (2004), 24.

[155] Sudo, Y., Eguchi, R., Izumi, T., and Masuzawa, T. Time-optimal loosely-stabilizing leader election in population protocols. *arXiv preprint arXiv:2005.09944* (2020).

[156] Sudo, Y., and Masuzawa, T. Leader election requires logarithmic time in population protocols. *Parallel Processing Letters 30*, 01 (2020), 2050005.

[157] Sudo, Y., Nakamura, J., Yamauchi, Y., Ooshita, F., Kakugawa, H., and Masuzawa, T. Loosely-stabilizing leader election in a population protocol model. *Theoretical Computer Science 444* (2012), 100 – 112. Structural Information and Communication Complexity – SIROCCO 2009.

[158] Sudo, Y., Ooshita, F., Izumi, T., Kakugawa, H., and Masuzawa, T. Brief announcement: Logarithmic expected-time leader election in population protocol model. In *PODC 2019: 38th ACM Symposium on Principles of Distributed Computing* (2019).

[159] Sudo, Y., Ooshita, F., Izumi, T., Kakugawa, H., and Masuzawa, T. Time-optimal leader election in population protocols. *IEEE Transactions on Parallel and Distributed Systems 31*, 11 (2020), 2620–2632.

[160] Sudo, Y., Ooshita, F., Kakugawa, H., Masuzawa, T., Datta, A. K., and Larmore, L. L. Loosely-stabilizing leader election with polylogarithmic convergence time. *Theor. Comput. Sci. 806* (2020), 617–631.

[161] Sudo, Y., Shibata, M., Nakamura, J., Kim, Y., and Masuzawa, T. The power of global knowledge on self-stabilizing population protocols. In *SIROCCO* (2020), vol. 12156, Springer, pp. 237–254.

[162] Thachuk, C., Winfree, E., and Soloveichik, D. Leakless DNA strand displacement systems. In *DNA 2015: Proceedings of the 21st International Conference on DNA Computing and Molecular Programming* (2015), Springer, pp. 133–153.

[163] Udayakumar, P., Vyas, R., and Vyas, O. P. Energy efficient election protocol for wireless sensor networks. In *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT)* (March 2013), pp. 1028–1033.

[164] Vichniac, G. Y. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena 10*, 1-2 (1984), 96–116.

[165] Volterra, V. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. *Memoria della Reale Accademia Nazionale dei Lincei 2* (1926), 31–113.

[166] Warnke, L. On wormald's differential equation method. *arXiv preprint arXiv:1905.08928* (2019).

[167] Woods, D., Doty, D., Myhrvold, C., Hui, J., Zhou, F., Yin, P., and Winfree, E. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature 567* (2019), 366–372.

[168] Wormald, N. C., et al. The differential equation method for random graph processes and greedy algorithms. *Lectures on approximation and randomized algorithms 73* (1999), 155.

[169] Xu, X., Yamauchi, Y., Kijima, S., and Yamashita, M. Space complexity of self-stabilizing leader election in population protocol based on k-interaction. In *SSS* (2013), pp. 86–97.

[170] Yokota, D., Sudo, Y., and Masuzawa, T. Time-optimal self-stabilizing leader election on rings in population protocols. In *SSS* (2020), vol. 12514, Springer, pp. 301–316.

[171] Yurke, B., Turberfield, A. J., Mills, Jr., A. P., Simmel, F. C., and Nuemann, J. L. A DNA-fuelled molecular machine made of DNA. *Nature 406* (2000), 605–608.