Stable Leader Election in Population Protocols Requires Linear Time

David Doty, David Soloveichik

Stable Leader Election in Population Protocols Requires Linear Time

David Doty[1,⋆] and David Soloveichik[2,⋆⋆]

[1] University of California, Davis, Davis, CA, USA, doty@ucdavis.edu
[2] University of Texas at Austin, Austin, TX, USA, david.soloveichik@utexas.edu

Abstract. A population protocol *stably elects a leader* if, for all n, starting from an initial configuration with n agents each in an identical state, with probability 1 it reaches a configuration $\mathbf{y}$ that is *correct* (exactly one agent is in a special leader state ℓ) and *stable* (every configuration reachable from $\mathbf{y}$ also has a single agent in state ℓ). We show that any population protocol that stably elects a leader requires $\Omega(n)$ expected "parallel time" — $\Omega(n^2)$ expected total pairwise interactions — to reach such a stable configuration. Our result also informs the understanding of the time complexity of chemical self-organization by showing an essential difficulty in generating exact quantities of molecular species quickly.

1 Introduction

Background. Population protocols (PPs) were introduced by Angluin, Aspnes, Diamadi, Fischer, and Peralta [2] as a model of distributed computing in which the agents have very little computational power and no control over their schedule of interaction with other agents. They also can be thought of as a special case of Petri nets/vector addition systems [15,16], which were introduced in the 1960s as a model of concurrent processing. In addition to being an appropriate model for electronic computing scenarios such as mobile sensor networks, they are a useful abstraction of "fast-mixing" physical systems such as animal populations [18], chemical reaction networks, and gene regulatory networks [7].

A PP is defined by a finite set Λ of *states* that each agent may have, together with a *transition function* $\delta : \Lambda \times \Lambda \to \Lambda \times \Lambda$.[3] Given states $r_1, r_2, p_1, p_2 \in \Lambda$, if $\delta(r_1, r_2) = (p_1, p_2)$ (denoted $r_1, r_2 \to p_1, p_2$) and a pair of agents in respective states r_1 and r_2 interact, then their states become p_1 and p_2.[4] A *configuration*

⋆ Author was supported by NSF grants CCF-1219274 and CCF-1442454 and the Molecular Programming Project under NSF grant 1317694.
⋆⋆ Author was supported by an NIGMS Systems Biology Center grant P50 GM081879 and NSF grant CCF-1442454.

[3] Some work on PPs allows "non-deterministic" transitions, in which the transition function maps to subsets of $\Lambda \times \Lambda$. Our results are independent of whether the PP is deterministic or nondeterministic in this manner.

[4] In the most generic model, there is no restriction on which agents are permitted to interact. If one prefers to think of the agents as existing on nodes of a graph, then it is the complete graph K_n for a population of n agents.

of a PP is a vector $\mathbf{c} \in \mathbb{N}^{\Lambda}$ describing, for each state $s \in \Lambda$, the *count* $\mathbf{c}(s)$ of how many agents are in state s. Executing a transition $r_1, r_2 \to p_1, p_2$ alters the configuration by decrementing the counts of states r_1 and r_2 by 1 each and incrementing p_1 and p_2 by 1 each.[5]

Associated with a PP is a set of *valid initial configurations* that we expect the PP to be able to handle.[6] Agents interact in a pairwise manner and change state based on the transition function. The next pair of agents to interact is chosen uniformly at random among the n agents. (An interaction may be a "null transition" $r_1, r_2 \to r_1, r_2$.) We count the expected number of *interactions* until some event occurs, and then define the "parallel time" until this event as the expected number of interactions divided by the number of agents n. This measure of time is based on the natural parallel model where each agent participates in a constant number of interactions in one unit of time, hence $\Theta(n)$ total interactions are expected per unit time [4]. In this paper all references to "time" refer to parallel time.

In order to define error-free computation in PPs, we rely on to the notion of *stable* computation [5]. The PP must get to a configuration that is correct[7] and "stable" in the sense that no subsequent sequence of transitions can take the PP to an incorrect configuration. Error-free computation must be correct in an "adversarial" schedule of transitions: we require that from every configuration reachable by *any* sequence of transitions from the initial configuration, it is possible to reach to a correct stable configuration. Since the configuration space is finite, requiring stability is equivalent to requiring, under the randomized model, that a correct stable configuration is reached with probability 1.[8]

A PP works "with a leader" if there is a special "leader" state ℓ, and every valid initial configuration $\mathbf{i}$ satisfies $\mathbf{i}(\ell) = 1$. This is in contrast to a uniform initial configuration ($\mathbf{i}(x) = n$ for some state x and $\mathbf{i}(y) = 0$ for all states $y \neq x$) or an initial configuration only encoding the input ($\mathbf{i}(x_i) = n_i$ for $i \in \{1, \ldots, k\}$ to represent any input $(n_1, n_2, \ldots, n_k) \in \mathbb{N}^k$). It is known that the predicates $\phi : \mathbb{N}^k \to \{0, 1\}$ stably computable by PPs are exactly the semilinear predicates, whether an initial leader is allowed or not [5]. Although the initial leader does not alter the class of computable predicates, it may allow faster computation. Specifically, the fastest known PPs to stably compute semilinear predicates

[5] Possibly some of r_1, r_2, p_1, p_2 are equal to each other, so the count of a state could change by 0, 1, or 2.

[6] The set of valid initial configurations for a "self-stabilizing" PP is $\mathbb{N}^{\Lambda}$, where leader election is provably impossible [6]. We don't require the PP to work if started in any possible configuration, but rather allow potentially "helpful" initial configurations as long as they don't already have small count states (see "α-dense" below).

[7] What "correct" means depends on the task. For computing a predicate, for example, Λ is partitioned into "yes" and "no" voters, and a "correct" configuration is one in which every state present has the correct vote.

[8] It is also equivalent to requiring that every *fair* sequence of transitions reaches a correct stable configuration, where "fair" means that every configuration infinitely often reachable is infinitely often reached [5].

without a leader take as long as $\Theta(n)$ to converge.[9] In contrast, with a leader, it is known that any semilinear predicate can be stably computed with expected convergence time $O(\log^5 n)$ [4]. Thus, in certain circumstances, the presence of a initial leader seems to give PPs more computational power (e.g., to converge quickly). Angluin, Aspnes, and Eisenstat [4] asked whether polylogarithmic time stable computation of semilinear predicates is possible without a leader; absent a positive answer, the presence of a leader appears to add power to the model.

Statement of main result. Motivated in part by the apparent speedup possible with an initial leader, we ask how quickly a leader may be elected from a configuration lacking one. We pose the problem as follows: design a PP $\mathcal{P}$ with two special states x (the initial state) and ℓ (the leader state, which may or may not be identical to x) such that, for every $n \in \mathbb{N}$, from the initial configuration $\mathbf{i}_n$ defined as $\mathbf{i}_n(x) = n$ and $\mathbf{i}_n(y) = 0$ for all other states y, has the following property. For every configuration $\mathbf{c}$ reachable from $\mathbf{i}_n$, there is a configuration $\mathbf{y}$ reachable from $\mathbf{c}$ that *has a stable leader*. By this we mean that in all configurations $\mathbf{y}'$ reachable from $\mathbf{y}$ (including $\mathbf{y}$ itself), $\mathbf{y}'(\ell) = 1$.[10]

There is a simple $O(n)$ expected time PP for stable leader election, with (assuming $x \equiv \ell$) the single transition $\ell, \ell \to \ell, f$. Our main theorem shows that *every* PP that stably elects a leader requires time $\Omega(n)$ to reach a state with a stable leader; thus the previous PP is asymptotically optimal.

Multiple leader states, multiple leaders, and other initial configurations. A more general notion of leader election is to identify a subset $\Psi \subset \Lambda$ of states that are all considered leader states, and to require the PP to eventually reach a configuration $\mathbf{y}$ in which $\sum_{\ell \in \Psi} \mathbf{y}(\ell) = 1$, and this sum is 1 in every configuration reachable from $\mathbf{y}$. This corresponds more appropriately to how leader states actually coordinate computation in PPs: a leader agent must remember some state information in between transitions (hence it changes state while remaining the unique leader). Our techniques actually show this stronger result as well (as explained in Section 3.2). Further, our result implies that a PP cannot elect any fixed quantity of leaders (e.g. exactly 256) or variable quantity of leaders under a fixed bound (e.g. at most 256) in sublinear expected time.

In the simplest formulation of the task of leader election, we always start with n agents in state x (as described above). Can we capture more generally leader election from a configuration "without a pre-existing leader"? Intuitively, we want to exclude initial configurations with states present in small but non-zero

[9] See "Open questions" for the distinction between time to *converge* and time to *stabilize*. In this paper, the time lower bound we prove is on stabilization.

[10] Note that this problem abstracts away the idea that the leader might be *useful* for something (such as computing predicates quickly). In particular, if a certain PP requires an initial leader, and the correctness of the PP depends on the count of the leader never exceeding 1, prior to the conclusion of the leader election, the presence of multiple leaders may result in unintended transitions. However, our main result is a impossibility theorem, showing that even if the objective is simplified to stable leader election, without requiring the leader to be useful for any subsequent task, this *still* requires $\Omega(n)$ time.

count. We can exclude such initial configurations, but allow otherwise deliberately prepared starting conditions, using the notion of α-dense configurations: any state present in the initial configuration has count $\geq \alpha n$. Our general negative result (Theorem 3.2) implies that even starting with best-case initial configurations, as long as, for some constant $\alpha > 0$, they are all α-dense, sublinear time leader election is impossible. An open question relates to weakening the notion of α-dense (see below).

Why simple proofs fail. It is tempting to believe that our negative result could follow by a simple argument based on reasoning about the last transition to change the count of the leader.[11] However, as the following example illustrates, reasoning about the last transition to change the count of the leader is insufficient if some transition can produce a new leader. Consider the following PP, with initial configuration $\mathbf{i}$ given by $\mathbf{i}(r) = n^{1/4}$, $\mathbf{i}(x) = n - n^{1/4}$, and transitions:

$$r, r \to \ell, k \quad (1) \qquad x, k \to k, k \quad (3)$$
$$r, k \to k, k \quad (2) \qquad \ell, \ell \to \ell, k \quad (4)$$

It can be shown (the analysis is presented in the full version of this paper) that this PP stably elects a leader in sublinear time $O(n^{1/2} \log n)$ from the above described non-α-dense initial configuration. Intuitively, it takes expected time $\Theta(n^{1/2})$ for transition (1) to occur for the first time, producing a single leader. Transition (4) ensures that if transition (1) occurs more than once, the PP will eventually stabilize to a single leader. However, with high probability, transitions (2) and (3) consume all r and x in $O(\log n)$ time *before* (1) executes a second time. The probability is high enough that the overall expected time to reach a state with a stable leader is sublinear. Although the above example does not violate our theorem (since it relies on a non-dense initial configuration), it shows that any proof of the main result cannot be based solely on reasoning about the final transition. The proof must also effectively establish that configurations, such as the initial configuration of the above PP, cannot be reached with high probability in sublinear time.

Chemical reaction networks. The main result and proof are stated in the language of PPs; however, the result holds for more general systems that have PPs as a special case. The discrete, stochastic chemical reaction network (CRN) model has been extensively used in the natural sciences to model chemical kinetics in a well-mixed solution [14], and the model is also used prescriptively for specifying the behavior of synthetic chemical systems [10, 17]. As an essential form of self-organization, biological cells seem able to precisely control the count of certain molecules (centriole number [11] is a well studied example). How chemical systems transform relatively uncontrolled initial conditions to precisely controlled amounts of desired species is still not well understood. Our negative

[11] Indeed, if we start with more than one leader, and no transition rule can produce a new leader, then we can easily prove the impossibility of sublinear time leader election as follows. To quickly reduce from two leaders to one, the other agent's state must be numerous in the population. Thus, the same transition could occur again, leaving us with no leaders.

result applied to CRNs[12] implies that generating with probability 1 an exact count of a certain species, whether 1 or 256, is *necessarily* slower ($\Omega(n)$ time) than, for example, destroying all molecules of the species (through the reaction $X \to \varnothing$), which takes $O(\log(n))$ time.

Open questions. An important open question concerns the contrast between *convergence* and *stabilization*. We say a PP electing a leader *converges* when it stops changing the count of the leader (if it is correct, this count should be 1), and we say it *stabilizes* when it first enters a configuration from which the count of the leader *cannot* change. In many PPs these two events coincide, but it is possible to converge strictly before stabilizing.[13] Our proof shows only that stabilization must take expected $\Omega(n)$ time. We leave as an open question whether there is a PP that stably elects a leader and converges in expected $o(n)$ time. Recall that there are PPs that work with a leader to stably compute semilinear predicates with convergence time $O(\log^5 n)$ [4]. Thus if stable leader election can converge in expected sublinear time, by coupling the two PPs it might be possible to achieve stable computation of arbitrary semilinear predicates with sublinear convergence time.

It is similarly open to determine the optimal stabilization time for computing semilinear predicates. The stably computing PPs converging in $O(\log^5 n)$ time [4] provably require expected time $\Omega(n)$ to stabilize, and it is unknown whether faster stabilization is possible even with an initial leader.

The open question of Angluin, Aspnes, and Eisenstat [4] asks whether their efficient high-probability simulation of a space-bounded Turing machine by a PP could remove the assumption of an initial leader. That simulation has some small probability $\epsilon > 0$ of failure, so if one could elect a leader with a small probability $\epsilon' > 0$ of error and subsequently use it to drive the simulation,by the union bound the total probability of error would be at most $\epsilon + \epsilon'$ (i.e., still close to 0). However, it remains an open question whether the necessary PP exists. Alistairh and Gelashvili [1] showed that relaxing the requirement of $O(1)$ states to $O(\log^3 n)$ states allows for a leader to be elected with high probability in expected time $O(\log^3 n)$.[14]

Our general negative result applies to α-dense initial configurations. However, is sublinear time stable leader election possible from other kinds of initial configurations that satisfy our intuition of not having preexisting leaders? It is known, for example, that for each $0 < \varepsilon < 1$, an initial configuration with $\Theta(n)$ agents in one state and $\Theta(n^\varepsilon)$ in another state can elect a leader in expected

[12] Our result holds for any CRN that obeys Theorem 4.3, the precise constraints of which are specified in [13] (those constraints automatically apply to all PPs).

[13] For example, consider the execution of the PP example above (1)–(4). Suppose (1) occurs just once, and then transition (2) occurs repeatedly and eliminates all r from the population. In this case, convergence happened when (1) occurred, but the PP stabilized only when all r was eliminated. Although in our example both convergence and stabilization occur in sublinear expected time, in general stabilization may occur with a substantial delay after convergence.

[14] Indeed, our proof technique fails if the number of states is not constant with respect to n.

time $O(\log^2 n)$ with high probability [4], although this protocol has a positive probability of failure. Above we give an example PP that stably elects a leader (convergence and stabilization) in $O(n^{1/2} \log n)$ time starting from an initial configuration with $\Theta(n)$ agents in one state and $\Theta(n^{1/4})$ in another state. In general we want to better characterize the initial configurations for which sublinear time leader election is possible.

2 Preliminaries

If Λ is a finite set (in this paper, of *states*), we write $\mathbb{N}^\Lambda$ to denote the set of functions $\mathbf{c} : \Lambda \to \mathbb{N}$. Equivalently, we view an element $\mathbf{c} \in \mathbb{N}^\Lambda$ as a vector of $|\Lambda|$ nonnegative integers, with each coordinate "labeled" by an element of Λ. Given $s \in \Lambda$ and $\mathbf{c} \in \mathbb{N}^\Lambda$, we refer to $\mathbf{c}(s)$ as the *count of s in $\mathbf{c}$*. Let $\|\mathbf{c}\| = \|\mathbf{c}\|_1 = \sum_{s \in \Lambda} \mathbf{c}(s)$ denote the total number of agents. We write $\mathbf{c} \leq \mathbf{c}'$ to denote that $\mathbf{c}(s) \leq \mathbf{c}'(s)$ for all $s \in \Lambda$. Since we view vectors $\mathbf{c} \in \mathbb{N}^\Lambda$ equivalently as multisets of elements from Λ, if $\mathbf{c} \leq \mathbf{c}'$ we say $\mathbf{c}$ is a *subset* of $\mathbf{c}'$. It is sometimes convenient to use multiset notation to denote vectors, e.g., $\{x, x, y\}$ and $\{2x, y\}$ both denote the vector $\mathbf{c}$ defined by $\mathbf{c}(x) = 2$, $\mathbf{c}(y) = 1$, and $\mathbf{c}(z) = 0$ for all $z \notin \{x, y\}$. Given $\mathbf{c}, \mathbf{c}' \in \mathbb{N}^\Lambda$, we define the vector component-wise operations of addition $\mathbf{c} + \mathbf{c}'$, subtraction $\mathbf{c} - \mathbf{c}'$, and scalar multiplication $m\mathbf{c}$ for $m \in \mathbb{N}$. For a set $\Delta \subset \Lambda$, we view a vector $\mathbf{c} \in \mathbb{N}^\Delta$ equivalently as a vector $\mathbf{c} \in \mathbb{N}^\Lambda$ by assuming $\mathbf{c}(s) = 0$ for all $s \in \Lambda \setminus \Delta$.

A *population protocol (PP)* is a pair $\mathcal{P} = (\Lambda, \delta)$,[15] where Λ is a finite set of *states*, and $\delta : \Lambda \times \Lambda \to \Lambda \times \Lambda$ is the (symmetric) *transition function*. A *configuration* of a PP is a vector $\mathbf{c} \in \mathbb{N}^\Lambda$, with the interpretation that $\mathbf{c}(s)$ agents are in state s. By convention, the value $n \in \mathbb{Z}^+$ represents the total number of agents $\|\mathbf{c}\|$. A *transition* is a 4-tuple $\alpha = (r_1, r_2, p_1, p_2) \in \Lambda^4$, written $\alpha : r_1, r_2 \to p_1, p_2$, such that $\delta(r_1, r_2) = (p_1, p_2)$. This paper typically defines a PP by a list of transitions, with δ implicit (there is a null transition $\delta(r_1, r_2) = (r_1, r_2)$ if a different transition is not specified). If an agent in state r_1 interacts with an agent in state r_2, then they change states to p_1 and p_2.

More formally, given a configuration $\mathbf{c}$ and transition $\alpha : r_1, r_2 \to p_1, p_2$, we say that α is *applicable* to $\mathbf{c}$ if $\mathbf{c} \geq \{r_1, r_2\}$, i.e., $\mathbf{c}$ contains 2 agents, one in state r_1 and one in state r_2. If α is applicable to $\mathbf{c}$, then write $\alpha(\mathbf{c})$ to denote the configuration $\mathbf{c} - \{r_1, r_2\} + \{p_1, p_2\}$ (i.e., the configuration that results from applying α to $\mathbf{c}$); otherwise $\alpha(\mathbf{c})$ is undefined. A finite or infinite sequence of transitions (α_i) is a *transition sequence*. Given an initial configuration $\mathbf{c}_0$ and a transition sequence (α_i), the induced *execution sequence* (or *path*) is a finite or infinite sequence of configurations $(\mathbf{c}_0, \mathbf{c}_1, \ldots)$ such that, for all $\mathbf{c}_i$ $(i \geq 1)$, $\mathbf{c}_i =$

[15] We give a slightly different formalism than that of [5] for population protocols. The main difference is that since we are not deciding a predicate, there is no notion of inputs being mapped to states or states being mapped to outputs. Another difference is that we assume (for the sake of brevity in some explanations, not because the difference is essential to the proof) the transition function is symmetric (so there is no notion of a "sender" and "receiver" agent as in [5]; the unordered pair of states completely determines the next pair of states).

$\alpha_{i-1}(\mathbf{c}_{i-1})$. If a finite execution sequence, with associated transition sequence q, starts with $\mathbf{c}$ and ends with $\mathbf{c}'$, we write $\mathbf{c} \Longrightarrow_q \mathbf{c}'$. We write $\mathbf{c} \Longrightarrow \mathbf{c}'$ if such a transition sequence exists (i.e., it is possible for the system to reach from $\mathbf{c}$ to $\mathbf{c}'$) and we say that $\mathbf{c}'$ is *reachable* from $\mathbf{c}$. If it is understood from context what is the initial configuration $\mathbf{i}$, then say $\mathbf{c}$ is simply *reachable* if $\mathbf{i} \Longrightarrow \mathbf{c}$. Note that this notation omits mention of $\mathcal{P}$; we always deal with a single PP at a time, so it is clear from context which PP is defining the transitions. If a transition $\alpha : r_1, r_2 \to p_1, p_2$ has the property that for $i \in \{1, 2\}$, $r_i \notin \{p_1, p_2\}$, or if ($r_1 = r_2$ and ($r_i \neq p_1$ or $r_i \neq p_2$)), then we say that α *consumes* r_i. In other words, applying α reduces the count of r_i. We similarly say that α *produces* p_i if it increases the count of p_i.

We will find ourselves frequently dealing with infinite sequences of configurations.[16] The following lemma, used frequently in reasoning about population protocols, shows that we can always take a nondecreasing subsequence.

Lemma 2.1 (Dickson's Lemma [12]) *Any infinite sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots \in \mathbb{N}^k$ has an infinite nondecreasing subsequence $\mathbf{x}_{i_0} \leq \mathbf{x}_{i_1} \leq \ldots$, where $i_0 < i_1 < \ldots$.*

In any configuration the next interaction is chosen by selecting a pair of agents uniformly at random and applying transition function δ. To measure time we count the expected total number of interactions (including null), and divide by the number of agents n. (In the population protocols literature, this is often called "parallel time"; i.e. n interactions among a population of n agents corresponds to one unit of time). Let $\mathbf{c} \in \mathbb{N}^\Lambda$ and $C \subseteq \mathbb{N}^\Lambda$. Denote the probability that the PP reaches from $\mathbf{c}$ to some configuration $\mathbf{c}' \in C$ by $\mathsf{Pr}[\mathbf{c} \Longrightarrow C]$. If $\mathsf{Pr}[\mathbf{c} \Longrightarrow C] = 1$,[17] define the *expected time to reach from* $\mathbf{c}$ *to* C, denoted $\mathsf{T}[\mathbf{c} \Longrightarrow C]$, to be the expected number of interactions to reach from $\mathbf{c}$ to some $\mathbf{c}' \in C$, divided by the number of agents n.

3 Main Results

3.1 Impossibility of Sublinear Time Stable Leader Election

We consider the following *stable leader election* problem. Suppose that each PP $\mathcal{P} = (\Lambda, \delta)$ we consider has a specially designated state $\ell \in \Lambda$, which we call the *leader state*. Informally, the goal of stable leader election is to be guaranteed to reach a configuration with count 1 of ℓ (a leader has been "elected"), from which no transition sequence can change the count of ℓ (the leader is "stable"). We also assume there is a special initial state x (it could be that $x \equiv \ell$ but it is not required), such that the only valid initial configurations $\mathbf{i}$ are of the form $\mathbf{i}(x) > 0$ and $\mathbf{i}(y) = 0$ for all states $y \in \Lambda \setminus \{x\}$. We write $\mathbf{i}_n$ to denote such an initial configuration with $\mathbf{i}_n(x) = n$.

[16] In general these will not be *execution* sequences. Typically none of the configurations are reachable from any others because they are configurations with increasing numbers of agents.

[17] Since PP's have a finite reachable configuration space, this is equivalent to requiring that for all $\mathbf{x}$ reachable from $\mathbf{c}$, there is a $\mathbf{c}' \in C$ reachable from $\mathbf{x}$.

Definition 3.1. *A configuration* $\mathbf{y}$ *is* stable *if, for all* $\mathbf{y}'$ *such that* $\mathbf{y} \Longrightarrow \mathbf{y}'$, $\mathbf{y}'(\ell) = \mathbf{y}(\ell)$ *(in other words, after reaching* $\mathbf{y}$*, the count of* ℓ *cannot change);* $\mathbf{y}$ *is said to have a* stable leader *if it is stable and* $\mathbf{y}(\ell) = 1$.

The following definition captures our notion of stable leader election. It requires the PP to be "guaranteed" eventually to reach a configuration with a stable leader.

Definition 3.2. *We say a PP elects a leader* stably *if, for all* $n \in \mathbb{Z}^+$*, for all* $\mathbf{c}$ *such that* $\mathbf{i}_n \Longrightarrow \mathbf{c}$*, there exists* $\mathbf{y}$ *with a stable leader such that* $\mathbf{c} \Longrightarrow \mathbf{y}$.

In other words, every reachable configuration can reach to a configuration with a stable leader. It is well-known [5] that the above definition is equivalent to requiring that the PP reaches a configuration with a stable leader with probability 1.

Definition 3.3. *Let* $t : \mathbb{Z}^+ \to \mathbb{R}^+$*, and let* Y *be the set of all configurations with a stable leader. We say a PP elects a leader stably* in time $t(n)$ *if, for all* $n \in \mathbb{Z}^+$, $\mathsf{T}[\mathbf{i}_n \Longrightarrow Y] \leq t(n)$.

Our main theorem says that stable leader election requires at least linear time to stabilize:

Theorem 3.1 *If a PP stably elects a leader in time* $t(n)$*, then* $t(n) = \Omega(n)$.

Thus a PP that elects a leader in sublinear time cannot do so stably, i.e., it must have a positive probability of failure.

The high-level strategy to prove Theorem 3.1 is as follows. With high probability the PP initially goes from configuration $\mathbf{i}_n$ to configuration $\mathbf{x}_n$, such that in the sequence $(\mathbf{x}_n)$ for increasing population size n, every state count grows without bound as $n \to \infty$ (indeed $\Omega(n)$); this follows from Theorem 4.3. We then show that any such configuration must have an "$O(1)$-bottleneck transition" before reaching a configuration with a stable leader (informally this means that every transition sequence from $\mathbf{x}_n$ to a configuration $\mathbf{y}$ with a stable leader must have a transition in which both input states have count $O(1)$, depending on the PP but not on n). Since it takes expected time $\Omega(n)$ to execute a transition when both states have constant count, from any such configuration it requires linear time to stably elect a leader. Since one of these configurations is reached from the initial configuration with high probability, those configurations' contribution to the overall expected time dominates, showing that the expected time to stably elect a leader is linear.

3.2 More General Impossibility Result in Terms of Inapplicable Transitions and Dense Configurations

Rather than proving Theorem 3.1 using the notion of leader stability directly, we prove a more general result concerning the notion of a set of inapplicable transitions. The two generalizations are as follows. (1) A configuration $\mathbf{y}$ is stable by Definition 3.1 if no transition altering the count of ℓ is applicable in any configuration reachable from $\mathbf{y}$; Definition 3.4 generalizes this to an arbitrary subset Q of transitions. (2) The valid initial configurations of Section 3.1 are

those with $\mathbf{i}_n(x) = n$ and $\mathbf{i}_n(y) = 0$ for all $y \in \Lambda \setminus \{x\}$; Theorem 3.2 generalizes this to any set I of configurations that are all "α-dense" (defined below) for a fixed $\alpha > 0$ independent of n, with a weak sort of "closure under addition" property: namely, that for infinitely many $\mathbf{i}, \mathbf{i}' \in I$, we have $\mathbf{i} + \mathbf{i}' \in I$.

Definition 3.4. *Let Q be a set of transitions. A configuration $\mathbf{y} \in \mathbb{N}^\Lambda$ is said to be* Q-stable *if no transition in Q is applicable in any configuration reachable from $\mathbf{y}$.*

If we let Q be the set of transitions that alter the count of the leader state ℓ, then a Q-stable configuration $\mathbf{y}$ with $\mathbf{y}(\ell) = 1$ exactly corresponds to the property of having a stable leader.

Let $I \subseteq \mathbb{N}^\Lambda$ and Q be a set of transitions. Let Y be the set of Q-stable configurations reachable from some configuration in I. We say that a PP $\mathcal{P} = (\Lambda, \delta)$ *Q-stabilizes from I* if, for any $\mathbf{i} \in I$, $\mathsf{Pr}[\mathbf{i} \Longrightarrow Y] = 1$.[18] If I and Q are understood from context, we say that $\mathcal{P}$ *stabilizes*. For a time bound $t(n)$, we say that $\mathcal{P}$ stabilizes *in expected time $t(n)$* if, for all $\mathbf{i} \in I$ such that $\|\mathbf{i}\| = n$, $\mathsf{T}[\mathbf{i} \Longrightarrow Y] \leq t(n)$.

To prove our time lower bound, we show that a "slow" transition necessarily occurs, which means that the counts of the two states in the transition are "small" when it occurs. We will pick a particular nondecreasing infinite sequence C of configurations and define "small" relative to it: the "small count" states are those whose counts are bounded in C (denoted $\mathsf{bdd}(C)$ below).

Definition 3.5. *For an (infinite) set/sequence of configurations C, let $\mathsf{bdd}(C)$ be the set of states $\{\ s \in \Lambda \mid (\exists b \in \mathbb{N})(\forall \mathbf{c} \in C)\ \mathbf{c}(s) < b\ \}$. Let $\mathsf{unbdd}(C) = \Lambda \setminus \mathsf{bdd}(C)$.*

Remark 3.1. Note that if $C = (\mathbf{c}_m)$ is a nondecreasing sequence, then for all $k \in \mathbb{N}$, there is $\mathbf{c}_m$ such that for all $s \in \mathsf{unbdd}(\mathbf{c}_m)$, $\mathbf{c}_m(s) \geq k$. (Note that if C is not nondecreasing, the conclusion can fail; e.g., $\mathbf{c}_m(s_1) = m, \mathbf{c}_m(s_2) = 0$ for m even and $\mathbf{c}_m(s_1) = 0, \mathbf{c}_m(s_2) = m$ for m odd.)

Let $0 < \alpha \leq 1$. We say that a configuration $\mathbf{c}$ is *α-dense* if for all $s \in \Lambda$, $\mathbf{c}(s) > 0$ implies that $\mathbf{c}(s) \geq \alpha\|\mathbf{c}\|$, i.e., all states present in $\mathbf{c}$ occupy at least an α fraction of the total count of agents.

Theorem 3.1 is implied by the next theorem, which the rest of the paper is devoted to proving.

Theorem 3.2 *Let $\mathcal{P} = (\Lambda, \delta)$, let Q be any subset of transitions of $\mathcal{P}$, let $\alpha > 0$, and let $I \subseteq \mathbb{N}^\Lambda$ be a set of α-dense initial configurations such that, for infinitely many $\mathbf{i}, \mathbf{i}' \in I$, $\mathbf{i} + \mathbf{i}' \in I$. Let Y be the set of Q-stable configurations reachable from I, and let $\Delta = \mathsf{bdd}(Y)$. Suppose $\mathcal{P}$ Q-stabilizes from I in expected time $o(n)$. Then there are infinitely many $\mathbf{y} \in Y$ such that $\forall s \in \Delta$, $\mathbf{y}(s) = 0$.*

In other words, if some states have "small" count in all reachable stable configurations, then there is a reachable stable configuration in which those states have count 0. A PP $\mathcal{P}$ that stably elects a leader is a PP in which Q is

[18] Recall that the condition $\mathsf{Pr}[\mathbf{i} \Longrightarrow Y] = 1$ is equivalent to $[(\forall \mathbf{c} \in \mathbb{N}^\Lambda)\ \mathbf{i} \Longrightarrow \mathbf{c}$ implies $(\exists \mathbf{y} \in Y)\ \mathbf{c} \Longrightarrow \mathbf{y}]$.

the set of transitions that alter the count of ℓ, $I = \{ \mathbf{i}_n \mid n \in \mathbb{N} \}$ (note all $\mathbf{i}_n$ are 1-dense), Y is the set of configurations reachable from I with a stable leader, and $\mathcal{P}$ Q-stabilizes from I. Hence by Theorem 3.2, if $\mathcal{P}$ stabilizes in expected time $o(n)$, there is a stable reachable $\mathbf{y}$ where $\mathbf{y}(\ell) = 0$, a contradiction. Thus Theorem 3.1 follows from Theorem 3.2.

We can also use Theorem 3.2 to prove that stable leader election requires linear time under the more relaxed requirement that there is a set $\Psi \subset \Lambda$ of "leader states," and the goal of the PP is to reach a configuration $\mathbf{y}$ in which $\sum_{\ell \in \Psi} \mathbf{y}(\ell) = 1$ and stays 1 in any configuration reachable from $\mathbf{y}$. Choosing Q as the set of transitions that alter that sum, Theorem 3.2 implies this form of stable leader election also requires $\Omega(n)$ expected time.

Throughout the rest of this paper, fix $\mathcal{P} = (\Lambda, \delta)$, α, I, and Q as in the statement of Theorem 3.2.

4 Technical Tools

4.1 Bottleneck Transitions Require Linear Time

This section proves a straightforward observation used in the proof of our main theorem. It states that, if to get from a configuration $\mathbf{x} \in \mathbb{N}^\Lambda$ to some configuration in a set $Y \subseteq \mathbb{N}^\Lambda$, it is necessary to execute a transition $r_1, r_2 \to p_1, p_2$ in which the counts of r_1 and r_2 are both at most some number b, then the expected time to reach from $\mathbf{x}$ to some configuration in Y is $\Omega(n/b^2)$.

Let $b \in \mathbb{N}$. We say that transition $\alpha : r_1, r_2 \to p_1, p_2$ is a *b-bottleneck* for configuration $\mathbf{c}$ if $\mathbf{c}(r_1) \leq b$ and $\mathbf{c}(r_2) \leq b$.

Observation 4.1 *Let $b \in \mathbb{N}$, $\mathbf{x} \in \mathbb{N}^\Lambda$, and $Y \subseteq \mathbb{N}^\Lambda$ such that $\mathsf{Pr}[\mathbf{x} \Longrightarrow Y] = 1$. If every transition sequence taking $\mathbf{x}$ to a configuration $\mathbf{y} \in Y$ has a b-bottleneck, then $\mathsf{T}[\mathbf{x} \Longrightarrow Y] \geq \frac{n-1}{2(b \cdot |\Lambda|)^2}$.*

A proof of Observation 4.1 is given in the full version of this paper. Intuitively, it follows because if two states r_1 and r_2 have count at most b, where b is a constant independent of n, then we expect to wait $\Omega(n)$ time before agents in states r_1 and r_2 interact.

Corollary 4.2 *Let $\gamma > 0$, $b \in \mathbb{N}$, $\mathbf{c} \in \mathbb{N}^\Lambda$, and $X, Y \subseteq \mathbb{N}^\Lambda$ such that $\mathsf{Pr}[\mathbf{c} \Longrightarrow X] \geq \gamma$, $\mathsf{Pr}[\mathbf{c} \Longrightarrow Y] = 1$, and every transition sequence from every $\mathbf{x} \in X$ to some $\mathbf{y} \in Y$ has a b-bottleneck. Then $\mathsf{T}[\mathbf{c} \Longrightarrow Y] \geq \gamma \frac{n-1}{2(b \cdot |\Lambda|)^2}$.*

4.2 Sublinear Time From Dense Configurations Implies Bottleneck Free Path From Configurations With Every State "Populous"

The following theorem, along with Corollary 4.2, fully captures the probability theory necessary to prove our main theorem.[19] Given it and Corollary 4.2, Theorem 3.2 is provable (through Lemma 4.1) using only combinatorial arguments about reachability between configurations.

[19] Theorem 4.3 was proven for a more general model called Chemical Reaction Networks (CRNs) that obey a certain technical condition [13]; as observed in that paper, the class of CRNs obeying that condition includes all PPs, so the theorem holds un-

For ease of notation, we assume throughout this paper that all states in Λ are *producible*, meaning they have positive count in some reachable configuration. Otherwise the following theorem applies only to states that are actually producible. Recall that for $\alpha > 0$, a configuration $\mathbf{c}$ is α-*dense* if for all $s \in \Lambda$, $\mathbf{c}(s) > 0$ implies that $\mathbf{c}(s) \geq \alpha\|\mathbf{c}\|$. Say that $\mathbf{c} \in \mathbb{N}^\Lambda$ is *full* if $(\forall s \in \Lambda)\ \mathbf{c}(s) > 0$, i.e., every state is present. The following theorem states that with high probability, a PP will reach from an α-dense configuration to a configuration in which all states are present (full) in "high" count (β-dense, for some $0 < \beta < \alpha$).

Theorem 4.3 (adapted from [13]) *Let $\mathcal{P} = (\Lambda, \delta)$ be a PP and $\alpha > 0$. Then there are constants $\epsilon, \beta > 0$ such that, letting , $X = \{\ \mathbf{x} \in \mathbb{N}^\Lambda \mid \mathbf{x}$ is full and β-dense $\}$, for all α-dense configurations $\mathbf{i}$, $\mathsf{Pr}[\mathbf{i} \Longrightarrow X] \geq 1 - 2^{-\epsilon\|\mathbf{i}\|}$.*

In [13], the theorem is stated for "sufficiently large" $\|\mathbf{i}\|$, but of course one can always choose ϵ to be small enough to make it true for all $\mathbf{i}$.

The following lemma reduces the problem of proving Theorem 3.2 to a combinatorial statement involving only reachability among configurations (and the lack of bottleneck transitions between them). In Section 5 we will prove Theorem 3.2 by showing that the existence of the configurations $\mathbf{x}_m$ and $\mathbf{y}_m$ and the transition sequence p_m in the following lemma implies that we can reach a Q-stable configuration $\mathbf{v} \in \mathbb{N}^\Gamma$, where $\Gamma = \mathsf{unbdd}(Y)$ and Y is the set of Q-stable configurations reachable from I.

Lemma 4.1. *Let $\alpha > 0$. Let $\mathcal{P} = (\Lambda, \delta)$ be a PP such that, for some set of transitions Q and infinite set of α-dense initial configurations I, $\mathcal{P}$ reaches a set of Q-stable configurations Y in expected time $o(n)$. Then for all $m \in \mathbb{N}$, there is a configuration $\mathbf{x}_m$ reachable from some $\mathbf{i} \in I$ and transition sequence p_m such that (1) $\mathbf{x}_m(s) \geq m$ for all $s \in \Lambda$, (2) $\mathbf{x}_m \Longrightarrow_{p_m} \mathbf{y}_m$, where $\mathbf{y}_m \in Y$, and (3) p_m has no m-bottleneck transition.*

The proof of Lemma 4.1 is in the full version of this paper. Intuitively, the lemma follows from the fact that states $\mathbf{x}_m$ are reached with high probability by Theorem 4.3, and if no paths such as p_m existed, then all paths from $\mathbf{x}_m$ to a stable configuration would have a bottleneck and require linear time. Since $\mathbf{x}_m$ is reached with high probability, this would imply the entire expected time is linear.

4.3 Transition Ordering Lemma

The following lemma was first proven (in the more general model of Chemical Reaction Networks) in [8]. Intuitively, the lemma states that a "fast" transition sequence (meaning one without a bottleneck transition) that decreases certain states from large counts to small counts must contain transitions of a certain

conditionally for PPs. The theorem proved in [13] is more general than Theorem 4.3, but we have stated a corollary of it here. A similar statement is implicit in the proof sketch of Lemma 5 of a technical report on a variant model called "urn automata" that has PPs as a special case [3].

restricted form. In particular the form is as follows: if Δ is the set of states whose counts decrease from large to small, then we can write the states in Δ in some order $d_1, d_2, \ldots, d_k$, such that for each $1 \leq i \leq k$, there is a transition α_i that consumes d_i, and every other state involved in α_i is either not in Δ, or comes later in the ordering. These transitions will later be used to do controlled "surgery" on fast transition sequences, because they give a way to alter the count of d_i, by inserting or removing the transitions α_i, knowing that this will not affect the counts of $d_1, \ldots, d_{i-1}$.

Lemma 4.2 (Adapted from [8]). *Let $b_1, b_2 \in \mathbb{N}$ such that $b_2 > |\Lambda| \cdot b_1$. Let $\mathbf{x}, \mathbf{y} \in \mathbb{N}^\Lambda$ such that $\mathbf{x} \Longrightarrow \mathbf{y}$ via transition sequence q that does not contain a b_2-bottleneck. Define $\Delta = \{\ d \in \Lambda \mid \mathbf{x}(d) \geq b_2 \text{ and } \mathbf{y}(d) \leq b_1\ \}$. Then there is an order on Δ, so that we may write $\Delta = \{d_1, d_2, \ldots, d_k\}$, such that, for all $i \in \{1, \ldots, k\}$, there is a transition α_i of the form $d_i, s_i \to o_i, o_i'$, such that $s_i, o_i, o_i' \notin \{d_1, \ldots, d_i\}$, and α_i occurs at least $(b_2 - |\Lambda| \cdot b_1)/|\Lambda|^2$ times in q.*

The intuition behind the proof is that the ordering is given (this is somewhat oversimplified) by the last time in q the state's count drops below b_2. Each state in Δ must go from "large" count (b_2) to "small" count (b_1), so when a state d_i is below count b_2, if a non-b_2-bottleneck transition $d_i, d_j \to \ldots$ occurs, then d_j must exceed b_2. This, in turn, means that state d_j cannot yet have dropped below count b_2 for the last time, so d_j is later in the ordering. The full argument is more subtle (and uses a different ordering) because it must establish that the transition's *outputs* in Δ also come later in the ordering.

5 Proof of Theorem 3.2

By Lemma 4.1, there are sequences $(\mathbf{x}_m)$ and $(\mathbf{y}_m)$ of configurations, and a sequence (p_m) of transition sequences, such that, for all m, (1) $\mathbf{x}_m(s) \geq m$ for all $s \in \Lambda$, and for some $\mathbf{i} \in I$, $\mathbf{i} \Longrightarrow \mathbf{x}_m$, (2) $\mathbf{y}_m$ is Q-stable, and (3) $\mathbf{x}_m \Longrightarrow_{p_m} \mathbf{y}_m$ and p_m does not contain an m-bottleneck.

By Dickson's Lemma there is an infinite subsequence of $(\mathbf{x}_m)$ for which both $(\mathbf{x}_m)$ and $(\mathbf{y}_m)$ are nondecreasing. Without loss of generality, we take $(\mathbf{x}_m)$ and $(\mathbf{y}_m)$ to be these subsequences. Let $\Delta = \mathsf{bdd}(\mathbf{y}_m)$ and $\Gamma = \mathsf{unbdd}(\mathbf{y}_m)$.

To prove Theorem 3.2 we need to show that there are configurations in Y (the set of Q-stable configurations reachable from I) that contain states only in Γ. Note that stability is closed downward: subsets of a Q-stable configuration are Q-stable. For any fixed $\mathbf{v}^\Gamma \in \mathbb{N}^\Gamma$, $\mathbf{v}^\Gamma \leq \mathbf{y}_m$ for sufficiently large m, by the definition of Γ (the states that grow unboundedly in $\mathbf{y}_m$ as $m \to \infty$). Thus *any* state $\mathbf{v}^\Gamma \in \mathbb{N}^\Gamma$ is automatically Q-stable. This is why Claims 5.1, 5.2, and 5.3 of this proof center around reaching configurations that have count 0 of every state in Δ.

Recall the path $\mathbf{x}_m \Longrightarrow_{p_m} \mathbf{y}_m$ from Lemma 4.1. Intuitively, Claim 5.1 below says that because this path is m-bottleneck free, Lemma 4.2 applies, and its transitions can appended to the path to consume all states in Δ from $\mathbf{y}_m$, resulting in a configuration $\mathbf{z}_m^\Gamma$ that contains only states in Γ. The "cost" of this manipulation is that, to ensure the appended transitions are applicable, we add extra

agents in specific states corresponding to $\mathbf{e} \in \mathbb{N}^{\Lambda}$. Claim 5.1 is not sufficient to prove Theorem 3.2 because of this additional $\mathbf{e}$; the subsequent Claims 5.2 and 5.3 will give us the machinery to handle it. The full proofs of Claims 5.1, 5.2, and 5.3 are given in the full version of this paper, and we give examples and intuition to explain them here.

Claim 5.1 *There is $\mathbf{e} \in \mathbb{N}^{\Lambda}$ such that for all large enough m, there is $\mathbf{z}_m^{\Gamma} \in \mathbb{N}^{\Gamma}$, such that $\mathbf{x}_m + \mathbf{e} \Longrightarrow \mathbf{z}_m^{\Gamma}$.*

Example. We illustrate Claim 5.1 through an example. Define a PP by the transitions

$$b, a \to f, c \quad (5) \qquad b, c \to f, a \quad (6) \qquad a, c \to f, f \quad (7)$$
$$f, c \to f, b \quad (8) \qquad f, b \to f, f \quad (9)$$

For convenience, for state $s \in \Lambda$, let s also denote the count of that state in the configuration considered. Let configuration $\mathbf{x}_m$ be where $f = 100$, $a = 100$, $b = 100$, $c = 100$. Suppose a transition sequence p_m without an m-bottleneck ($m = 100$) takes the PP from $\mathbf{x}_m$ to $\mathbf{y}_m$, in which $a = 3$, $b = 2$, $c = 1$, and $f = 394$. Then in the language of Lemma 4.2, $\Delta = \{a, b, c\}$; these states go from "large" count in $\mathbf{x}_m$ to "small" count in $\mathbf{y}_m$.

Our strategy is to add interactions to p_m in order to reach a configuration $\mathbf{z}_m^{\Gamma}$ with $a = b = c = 0$. There are two issues we must deal with. First, to get rid of a we may try to add 3 instances of (5) at the end of p_m. However, there is only enough b for 2 instances. To eliminate such dependency, in Claim 5.1, whenever we add a transition $b, a \to f, c$, we add an extra agent in state b to $\mathbf{e}$. (In general if we consume r_2 by adding transition $r_1, r_2 \to p_1, p_2$, we add an extra agent in state r_1 to $\mathbf{e}$.) Second, we need to prevent circularity in consuming and producing states. Imagine trying to add more executions of (5) to get a to 0 and more of (6) to get c to 0; this will fail because these transitions conserve the quantity $a + c$. To drive each of these states to 0, we must find some ordering on them so that each can be driven to 0 using a transition that does not affect the count of any state previously driven to 0.

Lemma 4.2 gives us a way to eliminate such dependency systematically. In the example above, we can find the ordering $d_1 \equiv a$, $d_2 \equiv c$, and $d_3 \equiv b$, with respective transitions (5) to drive a to 0 (3 executions), (8) to drive c to 0 (4 executions: 1 to consume the 1 copy of c in $\mathbf{y}_m$, and 3 more to consume the extra 3 copies that were produced by the 3 extra executions of (5)), and (9) to drive b to 0 (6 executions: 2 to consume 2 copies of b in $\mathbf{y}_m$, and 4 more to consume the extra 4 copies that were produced by the 4 extra executions of (8)).

Intuitively, Claim 5.2 below works toward generating the vector of states $\mathbf{e}$ that we needed for Claim 5.1. The vector $\mathbf{e}$ can be split into the Δ component and the Γ component; we will handle the Γ component later. The "cost" for Claim 5.2 is that the path must be taken "in the context" of additional agents in states captured by $\mathbf{p}$. Importantly, the net effect of the path preserves $\mathbf{p}$, which will give us a way to "interleave" Claims 5.1 and 5.2 as shown in Claim 5.3.

Claim 5.2 *For all $\mathbf{e}^\Delta \in \mathbb{N}^\Delta$, there is $\mathbf{p} \in \mathbb{N}^\Lambda$, such that for all large enough m, there is $\mathbf{w}_m^\Gamma \in \mathbb{N}^\Gamma$, such that $\mathbf{p} + \mathbf{x}_m \Longrightarrow \mathbf{p} + \mathbf{w}_m^\Gamma + \mathbf{e}^\Delta$, and $\mathsf{unbdd}(\mathbf{w}_m^\Gamma) = \Gamma$.*

Example. Recall the example above illustrating Claim 5.1. Claim 5.2 is more difficult than Claim 5.1 for two reasons. First, we need to be able to obtain any counts of states a, b, c (ie $\mathbf{e}^\Delta$) and not only $a = b = c = 0$. Second, we no longer have the freedom to add extra states as $\mathbf{e}$ and consume them. Note that $\mathbf{p}$ cannot fulfill the same role as $\mathbf{e}$ because $\mathbf{p}$ must be recovered at the end.

For instance suppose $\mathbf{e}^\Delta$ is $a = 7$, $b = 2$, $c = 1$. Recall that $\mathbf{y}_m$ has $a = 3$, $b = 2$, $c = 1$. How can we generate additional 4 copies of a? Note that all transitions preserve or decrease the sum $a + b + c$. Thus we cannot solely add interactions to p_m to get to our desired $\mathbf{e}^\Delta$. The key is that we can increase a by removing existing interactions from p_m that consumed it. Indeed, Lemma 4.2 helps us by giving a lower bound on the number of instances of transitions (5),(8),(9) that must have occurred in p_m. (Note that in Claim 5.1, we didn't need to use the fact that these transitions occurred in p_m. Now, we need to ensure that there are enough instances for us to remove.) In our case, we can remove 4 instances of interaction (5), which also decreases c by 4. To compensate for this, we can remove 4 instances of interaction (8), which also decreases b by 4. Finally, we remove 4 instances of interaction (9). The net result is that we reach the configuration $a = 7$, $b = 2$, $c = 1$, $f = 130$.

Note that unlike in Claim 5.1, we have more potential for circularity now because we cannot add the other input to a transition as $\mathbf{e}$. For example, we can't use transition (7) to affect c because it affects a (which we have previously driven to the desired count). Luckily, the ordering given by Lemma 4.2 avoids any circularity because the other input and both of the outputs come later in the ordering.

Importantly, as we remove interactions from p_m, we could potentially drive the count of some state temporarily negative. Performing these interactions in the context of more agents ($\mathbf{p}$) ensures that the path can be taken.

Claim 5.3 *For infinitely many $\mathbf{i} \in I$, there is $\mathbf{v}^\Gamma \in \mathbb{N}^\Gamma$ such that $\mathbf{i} \Longrightarrow \mathbf{v}^\Gamma$.*

Intuitively, Claim 5.3 follows by expressing $\mathbf{i} = \mathbf{i}_1 + \mathbf{i}_2$ where $\mathbf{i}_1 \Longrightarrow \mathbf{x}_{m_1}$ and $\mathbf{i}_2 \Longrightarrow \mathbf{x}_{m_2}$, so $\mathbf{i} \Longrightarrow \mathbf{x}_{m_1} + \mathbf{x}_{m_2}$. We then apply Claim 5.2 to $\mathbf{x}_{m_2}$ (with $\mathbf{x}_{m_1}$ playing the role of $\mathbf{p}$) to get to a configuration with the correct $\mathbf{e}$ for Claim 5.1, and then apply Claim 5.1 to remove all states in Δ.

Finally, Theorem 3.2 is proven because $\mathbf{v}^\Gamma$ is Q-stable and it contains zero count of states in Δ. To see that $\mathbf{v}^\Gamma$ is Q-stable recall that $\mathbf{v}^\Gamma \leq \mathbf{y}_{m'}$ for sufficiently large m' since $\Gamma = \mathsf{unbdd}(y_m)$ and $\mathbf{v}^\Gamma$ contains only states in Γ. Since stability is closed downward, and $\mathbf{y}_{m'}$ is Q-stable, we have that $\mathbf{v}^\Gamma$ is Q-stable as well.

Acknowledgements. The authors thank Anne Condon and Monir Hajiaghayi for several insightful discussions. We also thank the attendees of the 2014 Workshop on Programming Chemical Reaction Networks at the Banff International

Research Station, where the first incursions were made into the solution of the problem of PP stable leader election.

References

1. Alistarh, D., Gelashvili, R.: Polylogarithmic-time leader election in population protocols. In: ICALP 2015: Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming, Kyoto, Japan (2015)
2. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed Computing 18, 235–253 (2006), http://dx.doi.org/10.1007/s00446-005-0138-3, preliminary version appeared in PODC 2004
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Urn automata. Tech. Rep. YALEU/DCS/TR-1280, Yale University (November 2003)
4. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distributed Computing 21(3), 183–199 (Sep 2008), preliminary version appeared in DISC 2006
5. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing 20(4), 279–304 (2007)
6. Angluin, D., Aspnes, J., Fischer, M.J., Jiang, H.: Self-stabilizing population protocols. In: Principles of Distributed Systems, pp. 103–117. Springer (2006)
7. Bower, J.M., Bolouri, H.: Computational modeling of genetic and biochemical networks. MIT press (2004)
8. Chen, H., Cummings, R., Doty, D., Soloveichik, D.: Speed faults in computation by chemical reaction networks. In: DISC 2014: Proceedings of the 28th International Symposium on Distributed Computing, Austin, TX, USA. pp. 16–30 (2014), http://dx.doi.org/10.1007/978-3-662-45174-8_2
9. Chen, H.L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Natural Computing 13(4), 517–534 (2014), preliminary version appeared in DNA 2012.
10. Chen, Y.J., Dalchau, N., Srinivas, N., Phillips, A., Cardelli, L., Soloveichik, D., Seelig, G.: Programmable chemical controllers made from DNA. Nature Nanotechnology 8(10), 755–762 (2013)
11. Cunha-Ferreira, I., Bento, I., Bettencourt-Dias, M.: From zero to many: control of centriole number in development and disease. Traffic 10(5), 482–498 (2009)
12. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. American Journal of Mathematics 35(4), 413–422 (October 1913)
13. Doty, D.: Timing in chemical reaction networks. In: SODA 2014: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 772–784 (January 2014)
14. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81(25), 2340–2361 (1977)
15. Karp, R.M., Miller, R.E.: Parallel program schemata. Journal of Computer and System Sciences 3(2), 147–195 (1969)
16. Petri, C.A.: Communication with automata. Tech. rep., DTIC Document (1966)
17. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Sciences 107(12), 5393 (2010), preliminary version appeared in DNA 2008
18. Volterra, V.: Variazioni e fluttuazioni del numero dindividui in specie animali conviventi. Mem. Acad. Lincei Roma 2, 31–113 (1926)