Population-Induced Phase Transitions and the Verification of Chemical Reaction Networks

James I. Lathrop
Iowa State University, Ames, IA, USA
jil@iastate.edu

Jack H. Lutz
Iowa State University, Ames, IA, USA
lutz@iastate.edu

Robyn R. Lutz
Iowa State University, Ames, IA, USA
rlutz@iastate.edu

Hugh D. Potter
Iowa State University, Ames, IA, USA
hdpotter@iastate.edu

Matthew R. Riley
Iowa State University, Ames, IA, USA
mrriley@iastate.edu

Abstract

We show that very simple molecular systems, modeled as chemical reaction networks, can have behaviors that exhibit dramatic phase transitions at certain population thresholds. Moreover, the magnitudes of these thresholds can thwart attempts to use simulation, model checking, or approximation by differential equations to formally verify the behaviors of such systems at realistic populations. We show how formal theorem provers can successfully verify some such systems at populations where other verification methods fail.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases chemical reaction networks, molecular programming, phase transitions, population protocols, verification

Digital Object Identifier 10.4230/LIPIcs.DNA.2020.5

Funding This research was supported in part by National Science Foundation grants 1545028, 1900716, and 1909688.

Acknowledgements The second and third authors thank Erik Winfree for his hospitality while they did part of this work during a 2020 sabbatical visit at Caltech. We thank Neil Lutz for technical assistance. We thank the reviewers for detailed suggestions that have improved our exposition, both here and in an expansion of this paper in preparation.

1 Introduction

Chemical reaction networks, mathematical abstractions similar to Petri nets, are used as a programming language to specify the dynamic behaviors of engineered molecular systems. Existing software can compile chemical reaction networks into DNA strand displacement systems that simulate them with growing generality and precision [52, 14, 6, 53]. Programming is a challenging discipline in any case, but this is especially true of molecular programming, because chemical reaction networks – in addition to being Turing universal [51, 18, 21] and hence subject to all the uncomputable aspects of sequential, imperative programs–are, like the systems that they specify, distributed, asynchronous, and probabilistic. Since many envisioned

applications of molecular programming will be safety critical [54, 55, 19, 33, 32, 50, 44], programmers thus seek to create chemical reaction networks that can be *verified* to correctly carry out their design intent.

One principle that is sometimes used in chemical reaction network design is the *small population heuristic* [31, 11, 20]. The idea here is to verify various stages of a design by model checking or software simulation to ferret out bugs in the design prior to laboratory experimentation or deployment. Since the number of states of a molecular system is typically much larger than its population (the number of molecules present), and since molecular systems typically have very large populations, this model checking or simulation can usually only be carried out on populations that are far smaller than those of the intended molecular systems. It is nevertheless reasonable to hope that, if a system is going to consist of a very large number of "devices" of various sorts, then any unforeseen errors in these devices' interactions will manifest themselves even with very small populations of each device. It is this reasonable hope that is the underlying premise of the small population heuristic. (Note that the small population heuristic can be regarded as a molecular version of the small scope hypothesis [24].)

The question that we address here is whether real molecular systems can thwart the small population heuristic. That is, can a real molecular system behave very differently at large populations than at small populations? If so, *how sensitive* can its behavior be to its population, and *how simple a mechanism* can achieve such sensitivity?

In order to ensure that we are only investigating population effects, we focus our attention on chemical reaction networks that are *population protocols* in the sense that their populations remain constant throughout their operations. If we have such a chemical reaction network, and if we vary its initial population *and nothing else*, then we are assured that any resulting variations of behavior are due solely to the differing populations.

In this paper we show that very simple chemical reaction networks can be very sensitive to their own populations. In fact, they can exhibit *population-induced phase transitions*, behaving one way below a threshold population and behaving very differently above that threshold. After reviewing chemical reaction networks in Section 2, we present in Section 3 a chemical reaction network $\mathbf{N}_1$, and we prove that $\mathbf{N}_1$ exhibits a population-induced phase transition in the following sense. There are two parameters, m and n, in the construction. For this discussion, we may take $m = 34$ and $n = 67$, but the construction is general. There are $n + 1$ reactions among $n + 2$ species (molecule types) in $\mathbf{N}_1$. A species Z_0 is given an initial population p, and all other species counts are initially 0. Each reaction of $\mathbf{N}_1$ has two reactants and two products, so the total population of $\mathbf{N}_1$ is p at all times. There are in $\mathbf{N}_1$ two distinguished species, B and R. These "blue" and "red" species are abstract stand-ins for two different behaviors of $\mathbf{N}_1$. Our construction exploits the inherent nonlinearity of chemical kinetics to ensure that, if $p < 2^m$, then $\mathbf{N}_1$ terminates with essentially all its population blue, while if $p \geq 2^m$, then $\mathbf{N}_1$ terminates with essentially all its population red. Thus $\mathbf{N}_1$ exhibits a sharp phase transition at the population threshold $p = 2^m$.

Our construction is very simple. The chemical reaction network $\mathbf{N}_1$ changes its behavior at the threshold $p = 2^m$ by merely computing successive bits of p, starting at the least significant bit. This mechanism is so simple that it could be hidden, by accident or by malice, in a larger chemical reaction network. Moreover, for suitable values of m (e.g., $m = 34$, so that the threshold $p = 2^m$ is roughly 1.7×10^{10}),

(1) any attempt to model-check or simulate $\mathbf{N}_1$ will perforce use a population much less than the threshold and conclude that $\mathbf{N}_1$ will always turn blue; while

(2) any realistic wet-lab molecular implementation of $\mathbf{N}_1$ will have a population greater than the threshold and thus turn red.

If the behaviors represented by blue and red here are a desired, "good" behavior of $\mathbf{N}_1$ (or of a network containing $\mathbf{N}_1$) and an undesired, "bad" behavior of this network, respectively, then the possibility of such a phase transition is a serious challenge to verifying the correct behavior of the chemical reaction network. Simply put, this is a context in which the small population heuristic can lead us astray.

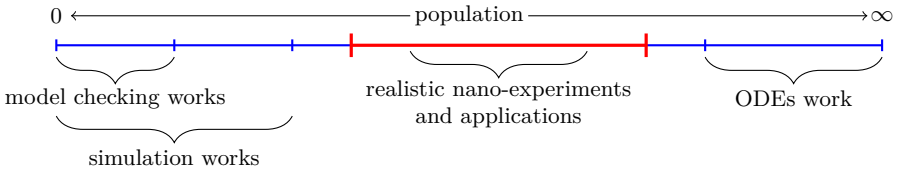

Figure 1 Scales at which different verification methods (simulation, model checking, and ODE's) work. The gap in the middle shows the scale at which none of these methods will catch the "produce blue" behavior of the system design. This gap is problematic because it is the scale of realistic programmed molecular systems. We show in Section 5.4 how such systems can be verified using automated theorem proving.

There is a dual *large population heuristic* that is used even more often than the small population heuristic. A theorem of Kurtz [27, 2, 3] draws a connection between the behavior of a *stochastic* chemical reaction network (the type of chemical reaction network used in our work here and in most of molecular programming) at large populations and the behavior of a *deterministic* chemical reaction network, which is governed by a system of ordinary differential equations. Kurtz's theorem involves several preconditions and caveats, and it does not always transparently equate stochastic and deterministic behavior. When it does apply, however, we can use a mathematical software package to numerically solve the deterministic system and thereby understand the behavior of the stochastic chemical reaction network at sufficiently large populations.

In Section 4 we add a single reaction to the chemical reaction network $\mathbf{N}_1$, creating a chemical reaction network $\mathbf{N}_2$ that we prove (in Theorem 4.6) to exhibit two *coupled population-induced phase transitions* in the following sense. If $p < 2^m$ or $p \geq 2^n$, then $\mathbf{N}_2$ terminates with essentially all its population blue, while if $2^m \leq p < 2^n$, then $\mathbf{N}_2$ terminates with essentially all its population red. Thus $\mathbf{N}_2$ exhibits sharp phase transitions at the two population thresholds, $p = 2^m$ and $p = 2^n$. These phase transitions are *coupled* in that exceeding the second threshold returns the behavior of $\mathbf{N}_2$ to its behavior below the first threshold. For suitable values of m and n (e.g. $m = 34$ and $n = 67$ as above, so that the thresholds $p = 2^m$ and $p = 2^n$ are roughly 1.7×10^{10} and 1.5×10^{20}), this implies (see Figure 1) that

(1) any attempt to model-check or simulate $\mathbf{N}_2$ will perforce use a population much less than the smaller threshold and conclude that $\mathbf{N}_2$ will always turn blue, and

(2) any realistic wet-lab molecular implementation of $\mathbf{N}_2$ will have a population between the two thresholds and thus turn red.

As we discuss later, when we analyze $\mathbf{N}_2$ with a numerical approach based on differential equations, we also do not observe a red outcome. The chemical reaction network $\mathbf{N}_2$ thus exemplifies a class of contexts in which the small population heuristic and the large population heuristic can both lead us astray.

We emphasize that the phase transitions in the chemical reaction networks $\mathbf{N}_1$ and $\mathbf{N}_2$ occur at thresholds in their *absolute populations*. In contrast, phase transitions in chemical reaction networks for approximate majority [4, 10, 17] occur at threshold *ratios between sub-populations*, and phase transitions in bacterial quorum sensing [36] occur at threshold *population densities*.

Section 5 discusses the consequences of our results for the verification of programmed molecular systems in some detail. Here we summarize these consequences briefly. Phase transitions are ubiquitous in natural and engineered systems [37, 45, 46, 47, 9, 43]. Our results are thus cautionary, but they should not be daunting. Fifteen years after Turing proved the undecidability of the halting problem, Rice [48, 49] proved his famous generalization stating that *every* nontrivial input/output property of programs is undecidable. Rice's theorem saves valuable time, but it has never prevented computer scientists from developing specific programs in disciplined ways that enable them to be verified. Similarly, Sections 3 and 4 give mathematical *proofs* that the chemical reaction networks $\mathbf{N}_1$ and $\mathbf{N}_2$ have the properties described above, and Section 5 describes how we have implemented such proofs in the Isabelle proof assistant [40, 41]. As molecular programming develops, simulators, model checkers, theorem provers, and other tools will evolve with it, as will disciplined scientific judgment about how and when to use such tools.

2 Chemical Reaction Networks

Chemical reaction networks (CRNs) are abstract models of molecular processes in well-mixed solutions. They are roughly equivalent to three models used in distributed computing, namely, Petri nets, population protocols, and vector addition systems [18]. This paper uses stochastic chemical reaction networks.

For our purposes, a (*stochastic*) *chemical reaction network* $\mathbf{N}$ consists of finitely many *reactions*, each of which has the form

$$A + B \to C + D, \tag{2.1}$$

where A, B, C, and D (not necessarily distinct) are *species*, i.e., abstract types of molecules. Intuitively, if this reaction occurs in a solution at some time, then one A and one B disappear from the solution and are replaced by one C and one D, these things happening instantaneously. A *state* of the chemical reaction network $\mathbf{N}$ with species $A_1, \ldots, A_n$ at a particular moment of time is the vector $(a_1, \ldots, a_s)$, where each a_i is the nonnegative integer count of the molecules of species A_i in solution at that moment. Note that we are using the so called "lower-case convention" for denoting species counts.

In the full stochastic chemical reaction network model, each reaction also has a positive real *rate constant*, and the random behavior of $\mathbf{N}$ obeys a continuous-time Markov chain derived from these rate constants. However, our results here are so robust that they hold for *any* assignment of rate constants, so we need not concern ourselves with rate constants or continuous-time Markov chains. In fact, for this paper, we can consider the reaction (2.1) to be the if-statement

$$\text{if } a > 0 \text{ and } b > 0 \text{ then } a, b, c, d := a - 1, b - 1, c + 1, d + 1 \tag{2.2}$$

(with the obvious modifications if A, B, C, and D are not distinct), where ":=" is parallel assignment. The reaction (2.1) is *enabled* in a state q of $\mathbf{N}$ if $a > 0$ and $b > 0$ in q; otherwise, this reaction is *disabled* in q. A state q of $\mathbf{N}$ is *terminal* if no reaction is enabled in q.

A *trajectory* of a chemical reaction network $\mathbf{N}$ is a sequence $\tau = (q_i \mid 0 \leq i < \ell)$ of states of $\mathbf{N}$, where $\ell \in \mathbb{Z}^+ \cup \{\infty\}$ is the *length* of τ and, for each $i \in \mathbb{N}$ with $i + 1 < \ell$, there is a reaction of $\mathbf{N}$ that is enabled in q_i and whose effect, as defined by (2.2), is to change the state of $\mathbf{N}$ from q_i to q_{i+1}. A trajectory $\tau = (q_i \mid 0 \leq i < \ell)$ is *terminal* if $\ell < \infty$ and $q_{\ell-1}$ is a terminal state of $\mathbf{N}$.

Assume for this paragraph that the context specifies an initial state q_0 of $\mathbf{N}$, as it does in this paper. A state q of $\mathbf{N}$ is *reachable* if there is a finite trajectory $\tau = (q_i \mid 0 \leq i < \ell)$ of $\mathbf{N}$ with $q_{\ell-1} = q$. A *full trajectory* of $\mathbf{N}$ is a trajectory $\tau = (q_i \mid 0 \leq i < \ell)$ that is either terminal or infinite.

The fact that each reaction (2.1) has two *reactants* (A and B) and two *products* (C and D) means that $\mathbf{N}$ is a *population protocol* [5]. This condition implies that the total population of all species never changes in the course of a trajectory. If such a chemical reaction network has s species and initial population p, its state space is thus the $(s-1)$-dimensional integer simplex

$$\Delta^{s-1}(p) = \left\{ (a_1, \ldots, a_s) \in \mathbb{N}^s \;\middle|\; \sum_{i=1}^{s} a_i = p \right\}. \tag{2.3}$$

Note that $|\Delta^{s-1}(p)| = \binom{p+s-1}{s-1}$. Of course, fewer than this many states may be reachable from a particular initial state of $\mathbf{N}$.

A full trajectory $\tau = (q_i \mid 0 \leq i < \ell)$ of a CRN $\mathbf{N}$ is (*strongly*) *fair* [30, 7] if it has the property that, for every state q and reaction ρ that is enabled in q,

$$(\exists^\infty i) q_i = q \implies (\exists^\infty j)[q_j = q \text{ and } \rho \text{ occurs at } j \text{ in } \tau], \tag{2.4}$$

where $(\exists^\infty i)$ means "there exist infinitely many i such that." Note that every terminal trajectory of $\mathbf{N}$ is vacuously fair, because it does not satisfy the hypothesis of (2.4).

The stochastic kinetics of chemical reaction networks implies that, regardless of the rate constants of the reactions, for every population protocol $\mathbf{N}$ and every initial population p of $\mathbf{N}$, there is a real number $\varepsilon > 0$ such that, for every state q of $\mathbf{N}$ and reaction ρ that is enabled in q, the probability that ρ occurs in q depends only on q and is at least ε. This in turn implies that, with probability 1, $\mathbf{N}$ follows a fair trajectory. Hence, if $\mathbf{N}$ has a given behavior on all fair trajectories, then $\mathbf{N}$ has that behavior with probability 1.

We use the following two facts in Section 4. The first is an obvious consequence of the definition of fairness.

▶ **Observation 2.1.** *If $\tau = (q_i \mid 0 \leq i < \ell)$ is a fair trajectory of a population protocol $\mathbf{N}$, then, for every reaction ρ of $\mathbf{N}$,*

$$(\exists^\infty i)[\rho \textit{ is enabled in } q_i] \implies (\exists^\infty j)[\rho \textit{ occurs at } j \textit{ in } \tau]. \tag{2.5}$$

A famous theorem of Harel [22, 26] implies that the general problem of deciding whether a chemical reaction network terminates on all fair trajectories is undecidable. Nevertheless, the following lemma gives a useful sufficient condition for termination of a population protocol on all fair trajectories. This lemma undoubtedly follows from a very old result on fairness, but we do not know a proper reference at the time of this writing. A proof appears in the Appendix.

▶ **Lemma 2.2** (fair termination lemma)**.** *If a population protocol with a specified initial state has a terminal trajectory from every reachable state, then all its fair trajectories are terminal.*

3 Single Phase Transition

This section presents the chemical reaction network $\mathbf{N}_1$ and proves that it exhibits a population-induced phase transition as described in the introduction.

Fix $m, n, p \in \mathbb{Z}^+$ with $n > m + 1$. Let $\mathbf{N}_1$ be a chemical reaction network consisting of the $n + 1$ *ζ-reactions*

$$\zeta_i \equiv Z_i + Z_i \to \begin{cases} Z_{i+1} + B & (0 \le i < m) \\ Z_{i+1} + R & (m \le i < n) \\ Z_i + R & (i = n) \end{cases}$$

and the *χ-reaction*

$$\chi \equiv B + R \to R + R.$$

All results here hold regardless of the rate constants of these $n + 2$ reactions.

We initialize $\mathbf{N}_1$ with $z_0 = p$ and all other counts 0.

Intuitively, B is *blue*, R is *red*, and the species Z_i are all *colorless*.

▶ **Lemma 3.1.** $\mathbf{N}_1$ *terminates on all possible trajectories.*

▶ Notation. For $1 \le k \le n + 1$, let

$$S_k = \sum_{i=0}^{k-1} 2^i z_i,$$

noting that this quantity depends on the state of $\mathbf{N}_1$.

▶ **Lemma 3.2.** *Let $0 \le j \le n$ and $1 \le k \le n + 1$.*

1. *If $j \ne k - 1$, then the reaction ζ_j preserves the value of S_k.*
2. *If $j = k - 1$, then the reaction ζ_j reduces the value of S_k.*

▶ **Corollary 3.3.** *For every $1 \le k \le n + 1$, the inequality $S_k \le p$ is an invariant of $\mathbf{N}_1$.*

▶ **Corollary 3.4.** *If $1 \le k \le n$ and $z_k > 0$ in some reachable state of $\mathbf{N}_1$, then $p \ge 2^k$.*

In the following, for $d \in \mathbb{Z}^+$, we use both the mod-d *congruence* (equivalence relation)

$$a \equiv b \bmod d,$$

which asserts of integers $a, b \in \mathbb{Z}$ that $b - a$ is divisible by d, and the **mod**-d *operation*

$$b \bmod d$$

whose value, for $b \in \mathbb{Z}$, is the unique $r \in \mathbb{Z}$ such that $0 \le r < d$ and $r \equiv b \bmod d$.

▶ **Corollary 3.5.** *The congruence*

$$S_n \equiv p \bmod 2^n \tag{3.1}$$

is an invariant of $\mathbf{N}_1$.

▶ **Corollary 3.6.** *For every $1 \le k \le n$, the condition*

$$\Theta_k \;\equiv\; [z_k = \cdots = z_n = 0 \implies S_k = p]$$

is an invariant of $\mathbf{N}_1$.

▶ **Corollary 3.7.** *Let $(q_0, \ldots, q_t)$ be a trajectory of $\mathbf{N}_1$, where q_t is a terminal state, and let $1 \le k \le n$. If $p \ge 2^k$, then there exists $1 \le s \le t$ such that $z_k > 0$ in q_s.*

▶ Notation. For each $r \in \{0, \ldots, 2^n - 1\}$, let $\lambda(r)$ be the number of 1s in the n-bit binary representation of r (leading 0s allowed), and let

$$\varepsilon = \begin{cases} \lambda(p) & \text{if } p < 2^n \\ 1 + \lambda(p \bmod 2^n) & \text{if } p \geq 2^n. \end{cases}$$

Note that ε is an integer depending on n and p, and that ε is negligible in the sense that $\varepsilon = o(p)$ as $p \to \infty$.

The boolean value of a condition φ is $[\![\varphi]\!] = \textbf{if } \varphi \textbf{ then } 1 \textbf{ else } 0$.

▶ **Theorem 3.8.** $\mathbf{N}_1$ *terminates on all trajectories in the state* $(z_0, \ldots, z_n, b, r)$ *specified as follows.*

(i) $z_{n-1} \cdots z_0$ *is the* n*-bit binary expansion of* $p \bmod 2^n$.
(ii) $z_n = [\![p \geq 2^n]\!]$.
(iii) $b = (p - \varepsilon) \cdot [\![p < 2^m]\!]$
(iv) $r = (p - \varepsilon) \cdot [\![p \geq 2^m]\!]$.

Proof. Lemma 3.1 tells us that $\mathbf{N}_1$ terminates on all trajectories. Let $q = (z_0, \ldots, z_n, b, r)$ be a terminal state of $\mathbf{N}_1$, and note the following.

(a) For all $0 \leq i \leq n$, ζ_i is not enabled in q, so $z_i \in \{0, 1\}$.
(b) χ is not enabled in q, so $b = 0$ or $r = 0$.
(c) By (a), $S_n \leq \sum_{i=0}^{n-1} 2^i = 2^n - 1$, so Corollary 3.5 tells us that $S_n = p \bmod 2^n$, i.e., that (i) holds.
(d) If $p < 2^n$, then Corollary 3.4 tells us that $z_n = 0$. If $p \geq 2^n$, then Corollary 3.7 tells us that $z_n \geq 1$ somewhere along every trajectory leading to q. Since z_n can never become 0 after becoming positive, this implies that $z_n = 1$ in q. Hence (ii) holds.
(e) By (c) and (d) we have $\sum_{i=0}^{n} z_i = \varepsilon$.
(f) Since $b + r + \sum_{i=0}^{n} z_i$ (the total population p) is an invariant of $\mathbf{N}_1$, (b) and (e) tell us that one of b and r is $p - \varepsilon$ and the other is 0.
(g) If $p < 2^m$, then Corollary 3.4 tells us that $z_m = \cdots = z_n = 0$ holds throughout every trajectory leading to q. This implies that none of the reactions $\zeta_m, \ldots, \zeta_n$ occurs along any trajectory leading to q, whence $r = 0$.
(h) If $p \geq 2^m$, then Corollary 3.7 tells us that $z_m > 0$ holds somewhere along every trajectory leading to q. This implies that the reaction ζ_{m-1} occurs, whence r becomes positive, somewhere along every trajectory leading to q. Since r can never become 0 after becoming positive, this implies that $r > 0$.
(i) By (f), (g), and (h), (iii) and (iv) hold. ◀

Since ε is negligible with respect to p, Theorem 3.8 says that $\mathbf{N}_1$ terminates in an overwhelmingly blue state if $p < 2^m$ and in an overwhelmingly red state if $p \geq 2^m$. This is a very sharp phase transition at the population threshold 2^m.

4 Coupled Phase Transitions

Let m, n, p, and $\mathbf{N_1}$ be as in Section 3, and let $\mathbf{N_2}$ be a CRN consisting of the $n+2$ reactions of $\mathbf{N_1}$ and the *ω-reaction*

$$\omega \equiv R + Z_n \to B + Z_n.$$

This section proves that $\mathbf{N_2}$ exhibits two coupled population-induced phase transitions as described in the introduction.

We use the same initialization for $\mathbf{N_2}$ as for $\mathbf{N_1}$. Again, all our results hold regardless of the rate constants of the $n+3$ reactions of $\mathbf{N_2}$.

Routine inspection verifies the following.

▶ **Observation 4.1.** *Lemma 3.2 and Corollaries 3.3-3.7 hold for* $\mathbf{N_2}$ *as well as for* $\mathbf{N_1}$.

If $p < 2^n$, then Corollary 3.4 tells us that z_n never becomes positive in $\mathbf{N_2}$, so the ω-reaction never occurs in $\mathbf{N_2}$. Thus, for $p < 2^n$, $\mathbf{N_2}$ behaves exactly like $\mathbf{N_1}$.

On the other hand, if $p \geq 2^n$, then the behavior of $\mathbf{N_2}$ is very different from that of $\mathbf{N_1}$. For example, in contrast with Lemma 3.1, we have the following.

▶ **Lemma 4.2.** *If* $p \geq 2^n$, *then not all trajectories of* $\mathbf{N_2}$ *terminate.*

It is easy to see that the infinite trajectory of $\mathbf{N}_2$ exhibited in the proof of Lemma 4.2 is not fair. In fact, we prove below that all fair paths of $\mathbf{N}_2$ terminate. First, however, we note that $\mathbf{N}_2$, like $\mathbf{N}_1$, has a unique terminal state.

Let ε be as defined before Theorem 3.8.

▶ **Lemma 4.3.** *If* $p \geq 2^n$ *and* $\boldsymbol{N}_2$ *terminates, then it does so in the state* $(z_0, \ldots, z_n, b, r)$ *specified as follows.*

(i) $z_{n-1} \cdots z_0$ *is the* $n-$*bit binary expansion of* p ***mod*** 2^n.
(ii) $z_n = 1$.
(iii) $b = p - \varepsilon$.
(iv) $r = 0$.

▶ **Lemma 4.4.** *On any fair trajectory of* $\boldsymbol{N}_2$, *after finitely many steps, all* ζ*-reactions are permanently disabled.*

▶ **Lemma 4.5.** *With any initialization, all fair trajectories of the chemical reaction network* $\boldsymbol{N}\chi\omega$, *consisting of just the reactions* χ *and* ω, *are terminal.*

Recall the notation defined just before Theorem 3.8. The following result is our main theorem.

▶ **Theorem 4.6.** *Let* $(z_0, \ldots, z_n, b, r)$ *be the state of* $\boldsymbol{N}_2$ *specified as follows.*

(i) $z_{n-1} \cdots z_0$ *is the* n*-bit binary expansion of* p ***mod*** 2^n.
(ii) $z_n = [\![p \geq 2^n]\!]$.
(iii) $b = (p - \varepsilon) \cdot [\![p < 2^m \text{ or } p \geq 2^n]\!]$.
(iv) $r = (p - \varepsilon) \cdot [\![2^m \leq p < 2^n]\!]$.

If $p < 2^n$, *then* $\boldsymbol{N}_2$ *terminates in this state on all trajectories. If* $p \geq 2^n$, *then* $\boldsymbol{N}_2$ *terminates in this state on all fair trajectories.*

Proof. If $p < 2^n$, then Corollary 3.3 tells us that z_n never becomes positive in $\mathbf{N}_2$, so ω is never enabled. Hence, in this case $\mathbf{N}_2$ behaves exactly like $\mathbf{N}_1$. Theorem 3.8 tells us that $\mathbf{N}_2$ terminates on all trajectories to the state satisfying (i) and (ii) above and, since $[\![p < 2^m]\!] = [\![p < 2^m \text{ or } p \geq 2^n]\!]$ and $[\![p \geq 2^m]\!] = [\![2^m \leq p < 2^n]\!]$, also satisfying (iii) and (iv) above.

If $p \geq 2^n$, then Lemmas 4.4 and 4.5 together tell us that $\mathbf{N}_2$ terminates on all fair trajectories. Since $[\![p \geq 2^n]\!] = 1$, $[\![p < 2^m \text{ or } p \geq 2^n]\!] = 1$, and $[\![2^m \leq p < 2^n]\!] = 0$, Lemma 4.3 tells us that termination must occur in the state satisfying (i)-(iv) above. ◀

Since ε is again negligible with respect to p, Theorem 4.6 says that $\mathbf{N}_2$ terminates in an overwhelmingly blue state if $p < 2^m$ or $p \geq 2^n$ but in an overwhelmingly red state if $2^m \leq p < 2^n$. Hence $\mathbf{N}_2$ exhibits very sharp phase transitions at the population thresholds 2^m and 2^n. As noted in the introduction and elaborated in Section 5 below, this has significant implications for the verification of chemical reaction networks.

5 Implications for Verification

The coupled phase transitions in the chemical reaction network $\mathbf{N_2}$ make it difficult to verify its behavior. In this section we describe the use and limitations of verifying the chemical reaction network using simulation, model checking and differential equations. None of these methods detected that the system turned red when the population is between $2^m = 2^{34} \approx 1.7 \times 10^{10}$ and $2^n = 2^{67} \approx 1.5 \times 10^{20}$. We then describe how the use of an interactive theorem prover enabled us to verify the chemical reaction network's behavior at both phase transitions, i.e., that it turned from blue to red at 2^m and from red to blue at 2^n. The fact that theorem proving could verify behavior that was otherwise not verified for the chemical reaction network suggests that interactive theorem proving may have a useful role to play in future verification of a class of chemical reaction networks.

Recall that the chemical reaction networks $\mathbf{N_1}$ and $\mathbf{N_2}$ have fixed populations throughout any given execution, and that their initial states have z_0 as the entire population.

5.1 Simulation

The MATLAB SimBiology package is widely used to explore the behavior of a number of devices (molecules) executing concurrently [35]. Using SimBiology, simulations of the $\mathbf{N_2}$ chemical reaction network were performed on an Intel processor computer with a processor clock of 5.0 GHz and 64GB of RAM. Several simulations were performed with increasing populations z_0. With a population of 10^7, the simulation performed as expected. However, with a population of 10^8, the simulation failed and terminated with no output or error message. Thus, the stochastic simulation was unable to detect that the behavior of the $\mathbf{N_2}$ chemical reaction network could experience a phase transition.

5.2 Model Checking

The chemical reaction network $\mathbf{N_2}$ simulated in SimBiology and described above also was verified using the PRISM 4.6 probabilistic model checker [28]. Kwiatkowska and Thachuk, among others, have described the use of PRISM for the probabilistic verification of chemical reaction networks for biological systems [29].

To verify the chemical reaction network behavior we first converted the $\mathbf{N_2}$ model to SBML using the export function in SimBiology, and then converted the SBML model to PRISM using the sbml2prism conversion tool supplied with the PRISM software. PRISM was used to verify six key properties of the $\mathbf{N_2}$ chemical reaction network at multiple populations. For example, one of the properties stated that "$P >= 1[\mathrm{F\ G}\ r = 0]$", i.e., that with probability 1, the eventual state of the R species has 0 molecules, and never changes from that. With a population of 100, PRISM generated the CTMC state model in 1.65 seconds using the same processor and memory as for the SimBiology simulations, and the verification of the six properties required less than 2 seconds of CPU time. For a population of 100 molecules, 97 are blue and 3 are colorless in the final state. PRISM also verified that in the final state $z_0 = z_1 = z_3 = z_4 = 0$ and $z_2 = z_5 = z_6 = 1$, so that $z_6z_5z_4z_3z_2z_1z_0$ is the binary expansion of one hundred.

However, we were unable to model check $\mathbf{N_2}$ with a population of 400 due to the rapid increase in states and limited memory. Thus, model checking confirmed the expected behavior of the $\mathbf{N_2}$ chemical reaction network for a population of 100 but could not detect the behavioral change to red when the population is greater than 2^{34}.

Advanced methods to prune a model so that meaningful model checking can occur include symmetry reduction [23], statistical model checking [11], and automated partial exploration of the model [42]. Recent work by Cauchi, et al. using formal synthesis allowed verification of systems with 10 continuous variables [12]. However, even these methods would not be likely to help with the exceedingly large number of states when the number of molecules is scaled to a realistic value for experiments.

5.3 Differential Equations

We have seen how model checking and simulation fail to detect the "red" behavior in our chemical reaction network $\mathbf{N}_2$ due to the processing time and memory required for a large population. The red behavior also is not detected when $\mathbf{N}_2$ is approximated by deterministic semantics. In this model, a chemical reaction network is represented by a system of polynomial autonomous differential equations. Our purpose here is to investigate the usefulness of the large population heuristic in this context; we do not make any claims that our results respect the preconditions and caveats of Kurtz's theorem [27], which provides a mathematical link between deterministic and high-population stochastic systems.

In general, the system of differential equations induced by a chemical reaction network is difficult or impossible to solve exactly, and numerical methods are often used to approximate solutions. Here, we utilized MATLAB and the SimBiology package [35] to numerically integrate the system of differential equations for $\mathbf{N}_2$. We found that $\mathbf{N}_2$ reached and remained in a predominantly blue state for the duration of the simulation, again missing the red behavior.

We identify three potential causes for this failure. One potential cause is numerical failure; it may be that MATLAB's numerical integration was not robust enough to capture the relevant deterministic behavior, or that we did not let the simulation run long enough to converge. (We note that, at least in the stochastic case, we expect $\mathbf{N}_2$ to take an extreme amount of time to converge.) Another potential cause is that, as suggested by Kurtz's theorem, the deterministic system might correctly approximate high-population stochastic behavior, which falls above the second phase-transition threshold (and well above the range of a realistic wet-lab implementation of $\mathbf{N}_2$.) Finally, it may be that the stochastic and deterministic behaviors of $\mathbf{N}_2$ are not actually closely related, and the deterministic result does not imply anything conclusive about the underlying stochastic system. Regardless of the cause, however, we see that differential equation methods are not sufficient to capture the red behavior of $\mathbf{N}_2$.

5.4 Theorem Proving

The simulation, model checking, and differential equations approaches to chemical reaction network verification outlined above all make some simplifying assumptions: reduced state space or generalization to the continuum. In the case of our chemical reaction network, these assumptions lead to an incorrect verification result.

Interactive theorem proving, however, offers an exact approach that is guaranteed to apply at every scale. In the interactive theorem proving paradigm, users create a machine-checkable mathematical proof of verification properties in collaboration with a software system. Model checking also constructs a mathematical proof of correctness, but it relies more on a complete or semi-complete search of the state space in question. By contrast, the goal of interactive theorem proving is to construct a more traditional mathematical proof that is also machine-checkable. The result then applies to any population scale; a mathematical proof parameterized by population N is valid at every possible value of N.

In a typical interactive theorem proving session, a user starts with a base of trusted facts generated from axioms and assumptions, and uses well-understood rules like modus ponens and double negation removal to construct new trusted facts and lemmas. As with a conventional mathematical proof, the user's goal is to add new trusted facts in a strategic way until reaching the goal of the proof.

We have verified our chemical reaction network with Isabelle/HOL [39, 40], a popular interactive theorem prover with several useful proof automation features. Instead of working at the level of rules like modus ponens, users can instruct Isabelle to execute more general proof methods that can apply sequences of basic rules without user direct input. For example, Isabelle can often prove the equivalence of predicate logic formulas with only one user-generated method invocation. Once invoked, such a method attempts to automatically construct a series of low-level logical rules whose application proves the equivalence. An Isabelle proof, then, consists of a directed acyclic graph of facts, connected by applications of these methods. The user's task is to choose a chain of intermediate goal facts in a way that allows Isabelle to connect them easily on the way to the overall goal.

Isabelle also provides the powerful Sledgehammer automation tool, which makes calls to external proof systems to automate aspects of proof creation. Sledgehammer takes a goal fact as input and attempts to generate a method invocation that proves it, operating at one level of abstraction above the proof methods invocations discussed above. Since it is often unclear which method to invoke (or which arguments to supply to it), this functionality can increase proof construction speed substantially.

We have used Isabelle to verify that our chemical reaction network has the desired behavior for all possible initializations. That is, if we initialize it with $N < 2^{34}$ or $N \geq 2^{67}$, the chemical reaction network terminates with majority blue, but if we initialize it with $2^{34} \leq N < 2^{67}$, it terminates with majority red. Theorem proving is able to verify behavior correctly in all regions, including the middle region that is inaccessible to model checking, simulation, and ODE methods. Figure 2 shows an image taken from the end of our Isabelle proof; it contains the three goal facts that we successfully verified, which summarize the behavior of the chemical reaction network.

Our Isabelle proof is loosely based on the proofs presented in Sections 3 and 4. Whereas those proofs define two chemical reaction networks $\mathbf{N_1}$ and $\mathbf{N_2}$, we use Isabelle's *locale* feature to associate assumptions about the population of N with various parts of our proof. In the locale where $N < 2^{35}$, for example, we are able to prove that our chemical reaction network terminates with majority blue. Figure 2 shows how we enter these locales at the end of the proof to bring together our final results.

We refer to the three final locales as the lower blue region, the middle red region, and the upper blue region. For each region, our proof must show both termination and correctness; i.e., we must show that our chemical reaction network reaches a final state where no reactions are possible, and that any possible final state has the specified red or blue population.

As in Lemma 3.1, we show termination in the lower two regions via a "countdown" expression that is guaranteed to decrease with every reaction. See Figure 3 for our Isabelle definitions of termination and a general lemma we proved that allows us to use the countdown technique. In the upper blue region, it is impossible to prove termination without assuming that executions are fair. Our Isabelle proof includes Equation 2.4 as an unproven assumption; we are not interested in unfair trajectories, but since they exist we cannot prove that all trajectories are fair. For convenience, we also include Observation 2.1 as an an assumption. These two fairness assumptions allow us to prove that our chemical reaction network terminates in the upper blue region as well.

```
theory results
  imports
    zeta_termination
    blue_zeta_correctness
    red_zeta_correctness
    omega_termination
    omega_correctness
begin

context blue_zeta begin

lemma blue_zeta_result: "∃t. terminal (p t) ∧ b (p t) + 68 ≥ N"
proof -
  show ?thesis using blue_zeta_terminal_correct zeta_term path_term_def by auto
qed

end

context red_zeta begin

lemma red_zeta_result: "∃t. terminal (p t) ∧ r (p t) + 68 ≥ N"
proof -
  show ?thesis using red_zeta_terminal_correct zeta_term path_term_def by auto
qed

end

context final_omega begin

lemma omega_result: "∃t. terminal (p t) ∧ b (p t) + 68 ≥ N"
proof -
  show ?thesis using omega_term_state omega_terminal_correct by auto
qed

end
end
```

Figure 2 The end of the Isabelle proof, which summarizes its results in three lemmas. The `context` statements bring our assumptions about the value of N into context. The `using` statements bring in trusted facts from the rest of our proof and supply them as arguments to Isabelle's `auto` proof method. The identifier `p` refers to an arbitrary trajectory that is part of each context. Isabelle displays all statements with a white or light gray background to indicate that it has checked them completely, and they are valid.

Our correctness proofs rely heavily on the sum $S_{68} = \sum_{i=0}^{67} 2^i z_i$, using the notation of Section 3, which is an invariant in the lower two regions. In the upper blue region, it is an invariant until at least one Z_{67} is produced. This invariant allows us to reason about the composition of terminal states. In the lower blue region, for example, we know that no red can ever be produced; the chemical reaction network can only produce its first red molecule alongside Z species that would make the invariant too large. Following the proof of Theorem 3.8, then, we prove that any terminal state must be majority blue.

6 Conclusion

Taken together, the near-ubiquity of phase transitions in nature [47, 9], the sheer size of molecular populations, and the simplicity of the chemical reaction networks that we have shown to exhibit population-induced phase transitions, indicate that molecular programming will present us with many exceptions to the otherwise useful notion that most bugs can be demonstrated with small counterexamples. As we have seen, this presents a significant challenge to the verification of chemical reaction networks. Here we suggest some directions of current and future research that might help meet this challenge.

```
theory termin
  imports piptcrn
begin

definition terminal :: state ⇒ bool where terminal s1 = (¬(∃ s2. K s1 s2))
definition nonterm :: state ⇒ bool where nonterm s = (¬(terminal s))

definition path-term :: (nat ⇒ state) ⇒ bool where
  (path-term p) = (∃ t. (terminal (p t)))

definition state-term :: (state ⇒ bool) where
  (state-term s) = (∀ (p :: (nat ⇒ state)).
     ((∃ t. ((p t) = s))
      ⟶ (path-term p)))

lemma dec-imp-term:
    fixes f :: state ⇒ nat
    fixes p :: nat ⇒ state
    fixes c :: nat
    assumes evterm: ((f s) ≤ c) ⟶ (terminal s)
    assumes dec: ∀ i. ((¬(terminal (p i))) ⟶ (
      (f (p (i + 1)) < (f (p i)))))
    shows path-term p
proof −
  {
    fix n::nat
    have ((∃ t. ((f (p t)) ≤ n)) ⟶ (path-term p))
    proof (induction n)
      case 0
      then show ?case
        using dec gr-implies-not0 path-term-def by blast
    next
      case (Suc n)
      then show ?case
        by (metis dec le-SucE less-Suc-eq-le path-term-def)
    qed
  }
  then show ?thesis by blast
qed

end
```

Figure 3 This Isabelle code defines a terminal state as a state with no outgoing reactions; K is a relation that encodes which state transitions our reaction set allows. We also show a sample lemma that helps prove termination: if we identify a countdown expression f and a constant C such that all states with $f < C$ are terminal, then our system is guaranteed to terminate.

A great deal of creative work has produced a steady scaling up of model checking to larger and larger state spaces [16, 15, 1, 8, 34, 13]. Perhaps the most hopeful approach for dealing with population-induced phase changes, or with more general population-sensitive behaviors, is the model checking of parametrized systems [1].

Our results clearly demonstrate the advantage of including theorem proving (by humans and by software) in the verification toolbox for chemical reaction networks and other molecular programming languages. This in turn suggests that software proof assistants such as Isabelle [40, 39] be augmented with features to deal more directly with chemical reaction networks and with population-sensitive phenomena. It would also be useful to know how much of such work could be carried out with more fully automated theorem provers such as Vampire [25].

Some future programmed molecular applications will be safety-critical, such as in health diagnostics and therapeutics. It is likely that evidence that such systems behave as intended will be required for certification by regulators prior to deployment. Toward providing such evidence, Nemouchi et al. have recently shown how a descriptive language for safety cases can be incorporated into Isabelle in order to formalize argument-based safety assurance cases [38].

We conclude with a more focused, theoretical question. Our chemical reaction network $\mathbf{N}_1$ exhibits its phase transition on all trajectories, while $\mathbf{N}_2$ exhibits its coupled phase transitions only on all fair trajectories. Is there a chemical reaction network that achieves $\mathbf{N}_2$'s coupled phase transitions on *all* trajectories?

References

1 Parosh Aziz Abdulla, A. Prasad Sistla, and Muralidhar Talupur. Model checking parameterized systems. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 685–725. Springer, 2018. doi:10.1007/978-3-319-10575-8_21.

2 David F. Anderson and Thomas G. Kurtz. Continuous time Markov chain models for chemical reaction networks. In Heinz Koeppl, Gianluca Setti, Mario di Bernardo, and Douglas Densmore, editors, *Design and Analysis of Biomolecular Circuits*, pages 3–42. Springer, 2011.

3 David F. Anderson and Thomas G. Kurtz. *Stochastic Analysis of Biochemical Systems*. Springer, 2015.

4 Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

5 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

6 Stefan Badelt, Seung Woo Shin, Robert F. Johnson, Qing Dong, Chris Thachuk, and Erik Winfree. A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. In *Proceedings of the 23rd International Conference on DNA Computing and Molecular Programming*, Springer, pages 232–248, 2017.

7 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

8 Luca Bortolussi, Luca Cardelli, Marta Kwiatkowska, and Luca Laurenti. Central limit model checking. *ACM Trans. Comput. Log.*, 20(4):19:1–19:35, 2019. doi:10.1145/3331452.

9 Sarah Cannon, Sarah Miracle, and Dana Randall. Phase transitions in random dyadic tilings and rectangular dissections. *SIAM J. Discret. Math.*, 32(3):1966–1992, 2018. doi:10.1137/17M1157118.

10 Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific Reports*, 2, 2012.

11 Luca Cardelli, Marta Kwiatkowska, and Max Whitby. Chemical reaction network designs for asynchronous logic circuits. *Natural Computing*, 17(1):109–130, 2018. doi:10.1007/s11047-017-9665-7.

12 Nathalie Cauchi, Luca Laurenti, Morteza Lahijanian, Alessandro Abate, Marta Kwiatkowska, and Luca Cardelli. Efficiency through uncertainty: scalable formal synthesis for stochastic hybrid systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019.*, pages 240–251, 2019. doi:10.1145/3302504.3311805.

13 Milan Ceska, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Shepherding hordes of Markov chains. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 172–190. Springer, 2019.

14 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8(10):755–762, 2013.

15 Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Asp. Comput.*, 30(1):45–75, 2018. doi:10.1007/s00165-017-0432-4.

16 Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Commun. ACM*, 52(11):74–84, 2009. doi:10.1145/1592761.1592781.

17 Anne Condon, Monir Hajiaghayi, David G. Kirkpatrick, and Ján Manuch. Simplifying analyses of chemical reaction networks for approximate majority. In *Proceedings of the 23rd International Conference on DNA Computing and Molecular Programming*, pages 188–209. Springer, 2017.

18 Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. Programmability of chemical reaction networks. In Anne Condon, David Harel, Joost N. Kok, Arto Salomaa, and Erik Winfree, editors, *Algorithmic Bioprocesses*, Natural Computing Series, pages 543–584. Springer, 2009.

19 Shawn M. Douglas, Ido Bachelet, and George M. Church. A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, 335(6070):831–834, 2012.

20 Samuel J. Ellis, Titus H. Klinge, James I. Lathrop, Jack H. Lutz, Robyn R. Lutz, Andrew S. Miner, and Hugh D. Potter. Runtime fault detection in programmed molecular systems. *ACM Trans. Softw. Eng. Methodol.*, 28(2):6:1–6:20, 2019. doi:10.1145/3295740.

21 François Fages, Guillaume Le Guludec, Olivier Bournez, and Amaury Pouly. Strong Turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In *Proceedings of the 15th International Conference on Computational Methods in Systems Biology*, pages 108–127. Springer, 2017.

22 David Harel. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *J. ACM*, 33(1):224–248, 1986. doi:10.1145/4904.4993.

23 J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In *Computational Methods in Systems Biology*, pages 32–47, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

24 Daniel Jackson. Alloy: a language and tool for exploring software designs. *Commun. ACM*, 62(9):66–76, 2019. doi:10.1145/3338843.

25 Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 1–35. Springer, 2013. doi:10.1007/978-3-642-39799-8_1.

26 Dexter Kozen. *Theory of Computation*. Texts in Computer Science. Springer, 2006. doi:10.1007/1-84628-477-5.

27 Thomas G. Kurtz. The relationship between stochastic and deterministic models for chemical reactions. *The Journal of Chemical Physics*, 57(7):2976–2978, 1972.

28 Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification*, pages 585–591. Springer, 2011.

29 Marta Kwiatkowska and Chris Thachuk. Probabilistic model checking for biology. *Software Systems Safety*, 36:165–189, 2014.

30 Marta Z. Kwiatkowska. Survey of fairness notions. *Information and Software Technology*, 31(7):371–386, 1989. doi:10.1016/0950-5849(89)90159-6.

31 Matthew R. Lakin, David Parker, Luca Cardelli, Marta Kwiatkowska, and Andrew Phillips. Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of the Royal Society Interface*, 9(72):1470–1485, 2012.

32 Suping Li, Qiao Jiang, Shaoli Liu, Yinlong Zhang, Yanhua Tian, Chen Song, Jing Wang, Yiguo Zou, Gregory J Anderson, Jing-Yan Han, Yung Chang, Yan Liu, Chen Zhang, Liang Chen, Guangbiao Zhou, Guangjun Nie, Hao Yan, Baoquan Ding, and Yuliang Zhao. A DNA nanorobot functions as a cancer therapeutic in response to a molecular trigger in vivo. *Nature Biotechnology*, 36:258, 2018.

33 Xiaowei Liu, Yan Liu, and Hao Yan. Functionalized DNA nanostructures for nanomedicine. *Israel Journal of Chemistry*, 53(8):555–566, 2013.

34 Alessio Lomuscio and Edoardo Pirovano. A counter abstraction technique for the verification of probabilistic swarm systems. In *Proceedings of the 18th International Conference on*

Autonomous Agents and MultiAgent Systems, AAMAS'19, pages 161–169, 2019. URL: http://dl.acm.org/citation.cfm?id=3331689.

35 MATLAB. *version 9.7.0 (R2019b, Update 4)*. The MathWorks Inc., Natick, Massachusetts, 2019.

36 Melissa B. Miller and Bonnie L. Bassler. Quorum sensing in bacteria. *Annual Review of Microbiology*, 55(1):165–199, 2001. PMID: 11544353. doi:10.1146/annurev.micro.55.1.165.

37 Cristopher Moore and Stephan Mertens. *The Nature of Computation*. Oxford University Press, 2011.

38 Yakoub Nemouchi, Simon Foster, Mario Gleirscher, and Tim Kelly. Isabelle/SACM: Computer-assisted assurance cases with integrated formal methods. In *Proceedings of the 15th International Conference on Integrated Formal MethodsIFM 2019*, pages 379–398. Springer, 2019. doi:10.1007/978-3-030-34968-4_21.

39 Tobias Nipkow and Gerwin Klein. *Concrete Semantics–With Isabelle/HOL*. Springer, 2014.

40 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 1 edition, 2002.

41 Lawrence C. Paulson, Tobias Nipkow, and Makarius Wenzel. From LCF to Isabelle/HOL. *Formal Asp. Comput.*, 31(6):675–698, 2019. doi:10.1007/s00165-019-00492-1.

42 Esteban Pavese, Víctor Braberman, and Sebastián Uchitel. Less is more: Estimating probabilistic rewards over partial system explorations. *ACM Transactions on Software Engineering and Methodology*, 25(2):16:1–16:47, 2016.

43 Gerald Pollack and Wei-Chun Chin, editors. *Phase Transitions in Cell Biology*. Springer, 2008.

44 Hamid Ramezani and Hendrik Dietz. Building machines with DNA molecules. *Nature Reviews Genetics*, 21(1):5–26, 2020.

45 Dana Randall. Phase transitions in sampling algorithms and the underlying random structures. In Haim Kaplan, editor, *Proceedings Scandinavian Symposium and Workshops on Algorithm Theory SWAT*, page 309. Springer, 2010. doi:10.1007/978-3-642-13731-0_29.

46 Dana Randall. Phase Transitions and Emergent Phenomena in Random Structures and Algorithms (Keynote Talk). In *31st International Symposium on Distributed Computing (DISC 2017)*, pages 3:1–3:2. Schloss Dagstuhl LZI, 2017. doi:10.4230/LIPIcs.DISC.2017.3.

47 Dana Randall. Statistical Physics and Algorithms (Invited Talk). In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, pages 1:1–1:6. Schloss Dagstuhl LZI, 2020.

48 H. G. Rice. *Classes of Recursively Enumerable Sets and Their Decision Problems*. PhD thesis, Syracuse University, 1951.

49 H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74:358–366, 1953. doi:10.1090/s0002-9947-1953-0053041-6.

50 Apoorva Sarode, Akshaya Annapragada, Junling Guo, and Samir Mitragotri. Layered self-assemblies for controlled drug delivery: A translational overview. *Biomaterials*, 242:119929, 2020. doi:10.1016/j.biomaterials.2020.119929.

51 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008.

52 David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. In *Proceedings of the 14th International Meeting on DNA Computing*, pages 57–69. Springer, 2009.

53 Anupama J. Thubagere, Chris Thachuk, Joseph Berleant, Robert F. Johnson, Diana A. Ardelean, Kevin M. Cherry, and Lulu Qian. Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components. *Nature Communications*, 8, 2017 .

54 John C. Wooley and Herbert S. Lin. *Catalyzing Inquiry at the Interface of Computing and Biology*. National Academies Press, 2005.

55 David Yu Zhang and Georg Seelig. Dynamic DNA nanotechnology using strand-displacement reactions. *Nature Chemistry*, 3(2):103–113, 2011.

A Proof of Fair Termination Lemma

▶ **Lemma A.1** (fair termination lemma). *If a population protocol with a specified initial state has a terminal trajectory from every reachable state, then all its fair trajectories are terminal.*

Proof. Let $\mathbf{N}$ be a population protocol with initial state q_0, and assume that $\mathbf{N}$ has a terminal trajectory from every reachable state. Let $\tau = (q_i \mid 0 \leq i < \infty)$ be an infinite trajectory of $\mathbf{N}$. It suffices to show that τ is not fair.

For each state q of $\mathbf{N}$, let

$$I_q = \{i \in \mathbb{N} \mid q_i = q\}. \tag{A.1}$$

Since $\mathbf{N}$ is a population protocol, it has finitely many reachable states, so there is a state q^* of $\mathbf{N}$ such that the set I_{q^*} is infinite. This state q^* is reachable, so our assumption tells us that there is a finite trajectory $\tau^* = (q^*_i \mid 0 \leq i < \ell)$ of $\mathbf{N}$ such that $q^*_0 = q^*$ and $q^*_{\ell-1}$ is terminal.

Now $I_{q^*_0} = I_{q^*}$ is infinite and $I_{q^*_{\ell-1}} = \emptyset$ (because $q^*_{\ell-1}$ is terminal, so it does not appear in the infinite trajectory τ), so there exists $0 \leq k < \ell - 1$ such that $I_{q^*_k}$ is infinite and $I_{q^*_{k+1}}$ is finite. Let $q^{**} = q^*_k$, and let ρ be the reaction that takes q^*_k to q^*_{k+1}. Then ρ is enabled in q^{**} and there exist infinitely many i such that $q_i = q^{**}$ (because $I_{q^{**}}$ is infinite), but there are only finitely many j for which $q_j = q^*$ and ρ occurs at j in τ (because $I_{q^*_{k+1}}$ is finite). Hence τ is not fair. ◀