Leaderless deterministic chemical reaction networks

David Doty* Monir Hajiaghayi†

Abstract

This paper answers an open question of Chen, Doty, and Soloveichik [4], who showed that a function $f : \mathbb{N}^k \to \mathbb{N}^l$ is deterministically computable by a stochastic chemical reaction network (CRN) if and only if the graph of f is a semilinear subset of $\mathbb{N}^{k+l}$. That construction crucially used "leaders": the ability to start in an initial configuration with constant but non-zero counts of species other than the k species $X_1, \ldots, X_k$ representing the input to the function f. The authors asked whether deterministic CRNs without a leader retain the same power.

We answer this question affirmatively, showing that every semilinear function is deterministically computable by a CRN whose initial configuration contains only the input species $X_1, \ldots, X_k$, and zero counts of every other species, so long as $f(\mathbf{0}) = \mathbf{0}$. We show that this CRN completes in expected time $O(n)$, where n is the total number of input molecules. This time bound is slower than the $O(\log^5 n)$ achieved in [4], but faster than the $O(n \log n)$ achieved by the *direct* construction of [4].

Keywords: chemical reaction network, leader election, semilinear function, time complexity

Abbreviations:
CRD: chemical reaction decider
CRN: chemical reaction network
CRC: chemical reaction computer

*California Institute of Technology, Pasadena, CA, USA, ddoty@caltech.edu. This author was supported by the Molecular Programming Project under NSF grant 0832824 and by NSF grants CCF-1219274 and CCF-1162589.
†University of British Columbia, Vancouver, BC, Canada, monirh@cs.ubc.ca.

1 Introduction

In the last two decades, theoretical and experimental studies in molecular programming have shed light on the problem of integrating logical computation with biological systems. One goal is to repurpose the *descriptive* language of chemistry and physics, which describes how the natural world works, as a *prescriptive* language of programming, which prescribes how an artificially engineered system *should* work. When the programming goal is the manipulation of individual molecules in a well-mixed solution, the language of chemical reaction networks (CRNs) is an attractive choice. A CRN is a finite set of reactions such as $X + Y \to X + Z$ among abstract molecular species, each describing a rule for transforming reactant molecules into product molecules.

CRNs may model the "amount" of a species as a real number, namely its concentration (average count per unit volume), or as a nonnegative integer (total count in solution, requiring the total volume of the solution to be specified as part of the system). The latter integer counts model is called "stochastic" because reactions that discretely change the state of the system are assumed to happen probabilistically, with reactions whose reactants have high molecular counts more likely to happen first than reactions whose molecular counts are smaller. The computational power of CRNs has been investigated with regard to simulating boolean circuits [15], neural networks [12], digital signal processing [13], and simulating bounded-space Turing machines with an arbitrary small, non-zero probability of error with only a polylogarithmic slowdown [3]. CRNs are even efficiently Turing-universal, again with a small, nonzero probability of error over all time [16]. Certain CRN termination and producibility problems are undecidable [8,19], and others are hard for EXPSPACE [14] or PSPACE [18]. It is also difficult to design a CRN to "delay" the production of a certain species [6,7,9]. Using a theoretical model of DNA strand displacement, it was shown that any CRN can be transformed into a set of DNA complexes that approximately emulate the CRN [17]. Therefore even hypothetical CRNs may one day be reliably implementable by real chemicals.

While these papers focus on the stochastic behavior of chemical kinetics, our focus is on CRNs with deterministic guarantees on their behavior. Some CRNs have the property that they deterministically progress to a correct state, no matter the order in which reactions occur. For example, the CRN with the reaction $X \to 2Y$ is guaranteed eventually to reach a state in which the count of Y is twice the initial count of X, i.e., computes the function $f(x) = 2x$, representing the input by species X and the output by species Y. Similarly, the reactions $X_1 \to 2Y$ and $X_2 + Y \to \varnothing$, under arbitrary choice of sequence of the two reactions, compute the function $f(x_1, x_2) = \max\{0, 2x_1 - x_2\}$.

Angluin, Aspnes and Eisenstat [2] investigated the computational behavior of CRNs that are deterministic (in the sense described in the previous paragraph) under a different name known as *population protocols* [1]. They showed that the input sets $S \subseteq \mathbb{N}^k$ decidable by deterministic CRNs (i.e. providing "yes" or "no" answers by the presence or absence of certain indicator species) are precisely the *semilinear* subsets of $\mathbb{N}^k$.[1] Chen, Doty, and Soloveichik [4] extended these results to function computation and showed that precisely the semilinear functions (functions f whose graph $\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{N}^{k+l} \mid f(\mathbf{x}) = \mathbf{y} \}$ is a semilinear set) are deterministically computable by CRNs. We say a function $f : \mathbb{N}^k \to \mathbb{N}^l$ is *stably* (a.k.a., *deterministically*) computable by a CRN $\mathcal{C}$ if there are "input" species $X_1, \ldots, X_k$ and "output" species $Y_1, \ldots, Y_l$ such that, if $\mathcal{C}$ starts with $x_1, \ldots, x_k$ copies of $X_1, \ldots, X_k$ respectively, then with probability one, it reaches a count-stable configuration

[1] Semilinear sets are defined formally in Section 2. Informally, they are finite unions of "periodic" sets, where the definition of "periodic" is extended in a natural way to multi-dimensional spaces such as $\mathbb{N}^k$.

in which the counts of $Y_1, \ldots, Y_l$ are expressed by the vector $f(x_1, ..., x_k)$, and these counts never again change [4].

The method proposed in [4] uses some auxiliary "leader" species present initially, in addition to the input species. To illustrate their utility, suppose that we want to compute function $f(x) = x+1$ with CRNs. Using the previous approach, we have an input species X (with initial count x), an output species Y (with initial count 0) and an auxiliary "leader" species L (with initial count 1). The following reactions compute $f(x)$:

$$X \to Y$$
$$L \to Y$$

However, it is experimentally difficult to prepare a solution with a single copy (or a small constant number) of a certain species. The authors of [4] asked whether it is possible to do away with the initial "leader" molecules, i.e., to require that the initial configuration contains initial count $x_1, x_2, \ldots, x_k$ of input species $X_1, X_2, \ldots, X_k$, and initial count 0 of every other species. It is easy to "elect" a single leader molecule from an arbitrary initial number of copies using a reaction such as $L + L \to L$, which eventually reduces the count of L to 1. However, the problem with this approach is that, since L is a reactant in other reactions, there is no way in general to prevent L from participating in these reactions until the reaction $L + L \to L$ has reduced it to a single copy.

Despite these difficulties, we answer the question affirmatively, showing that each semilinear function can be computed by a "leaderless" CRN, i.e., a CRN whose initial configuration contains only the input species, so long as $f(\mathbf{0}) = \mathbf{0}$.[2] To illustrate one idea used in our construction, consider the function $f(x) = x + 1$ described above. In order to compute the function without a leader (i.e., the initial configuration has x copies of X and 0 copies of every other species), the following reactions suffice:

$$X \to B + 2Y \tag{1.1}$$
$$B + B \to B + K \tag{1.2}$$
$$Y + K \to \varnothing \tag{1.3}$$

Reaction 1.1 produces x copies of B and $2x$ copies of Y. Reaction 1.2 consumes all copies of B except one, so reaction 1.2 executes precisely $x - 1$ times, producing $x - 1$ copies of K. Therefore reaction 1.3 consumes $x - 1$ copies of output species Y, eventually resulting in $2x - (x - 1) = x + 1$ copies of Y. Note that this approach uses a sort of leader election on the B molecules.

In Section 3, we generalize this example, describing a leaderless CRN construction to compute any semilinear function. We use a similar framework to the construction of [4], decomposing the semilinear function into a finite union of affine partial functions (linear functions with an offset; defined formally in Section 2). We show how to compute each affine function with leaderless CRNs, using a fundamentally different construction than the affine-function computing CRNs of [4]. This result, Lemma 3.1, is the primary technical contribution of this paper. Next, in order to decide which affine function should be applied to a given input, we employ the leaderless semilinear predicate computation of Angluin, Aspnes, and Eisenstat [3]; this latter part of the construction is actually identical to the construction of [4], but we include it because our time analysis is different.

[2] It is easy to see that no leaderless CRN could reach an output stable state with positive count of output species Y from an initial state with no molecules, since it would need to contain reaction(s) of the form $\varnothing \to A$ for some species A from which (unbounded counts of) Y could be produced.

Let $n = \|\mathbf{x}\| = \|\mathbf{x}\|_1 = \sum_{i=1}^{k} \mathbf{x}(i)$ be the number of molecules present initially, as well as the volume of the solution. The authors of [4] showed, for each semilinear function f, a direct construction of a CRN that computes f (using leaders) on input $\mathbf{x}$ in expected time $O(n \log n)$. They then combined this direct, error-free construction in parallel with a fast ($O(\log^5 n)$) but error-prone CRN that uses a leader to compute *any* computable function (including semilinear), using the error-free computation to change the answer of the error-prone computation only if the latter is incorrect. This combination speeds up the computation from expected time $O(n \log n)$ for the direct construction to expected time $O(\log^5 n)$ for the combined construction.

Since we assume no leaders may be supplied in the initial configuration, and since the problem of computing arbitrary computable functions without a leader remains a major open problem [3], this trick does not work for speeding up our construction. However, we show that with some care in the choice of reactions, the direct stable computation of a semilinear function can be done in expected time $O(n)$, improving upon the $O(n \log n)$ bound of the direct construction of [4].

2 Preliminaries

Given a vector $\mathbf{x} \in \mathbb{N}^k$, let $\|\mathbf{x}\| = \|\mathbf{x}\|_1 = \sum_{i=1}^{k} |\mathbf{x}(i)|$, where $\mathbf{x}(i)$ denotes the ith coordinate of $\mathbf{x}$. A set $A \subseteq \mathbb{N}^k$ is *linear* if there exist vectors $\mathbf{b}, \mathbf{u}_1, \ldots, \mathbf{u}_p \in \mathbb{N}^k$ such that

$$A = \{\ \mathbf{b} + n_1\mathbf{u}_1 + \ldots + n_p\mathbf{u}_p \mid\ n_1, \ldots, n_p \in \mathbb{N}\ \}.$$

A is *semilinear* if it is a finite union of linear sets. If $f : \mathbb{N}^k \to \mathbb{N}^l$ is a function, define the *graph* of f to be the set $\{\ (\mathbf{x}, \mathbf{y}) \in \mathbb{N}^k \times \mathbb{N}^l \mid\ f(\mathbf{x}) = \mathbf{y}\ \}$. A function is *semilinear* if its graph is semilinear. A predicate $\phi : \mathbb{N}^k \to \{0, 1\}$ is *semilinear* if the set $\{\ \mathbf{x} \in \mathbb{N}^k \mid\ \phi(\mathbf{x}) = 1\ \}$ is a semilinear set.

We say a partial function $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$ is *affine* if there exist kl rational numbers $a_{1,1}, \ldots, a_{k,l} \in \mathbb{Q}$ and $l + k$ nonnegative integers $b_1, \ldots, b_l, c_1, \ldots, c_k \in \mathbb{N}$ such that, if $\mathbf{y} = f(\mathbf{x})$, then for each $j \in \{1, \ldots, l\}$, $\mathbf{y}(j) = b_j + \sum_{i=1}^{k} a_{i,j}(\mathbf{x}(i) - c_i)$, and for each $i \in \{1, \ldots, k\}$, $\mathbf{x}(i) - c_i \geq 0$. In matrix notation, there exist a $k \times l$ rational matrix $\mathbf{A}$ and vectors $\mathbf{b} \in \mathbb{N}^l$ and $\mathbf{c} \in \mathbb{N}^k$ such that $f(\mathbf{x}) = \mathbf{A}(\mathbf{x} - \mathbf{c}) + \mathbf{b}$.

This definition of affine function may appear contrived; see [4] for an explanation of its various intricacies. For reading this paper, the main utility of the definition is that it satisfies Lemma 3.2.

Note that by appropriate integer arithmetic, a partial function $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$ is affine if and only if there exist kl integers $n_{1,1}, \ldots, n_{k,l} \in \mathbb{Z}$ and $2l+k$ nonnegative integers $b_1, \ldots, b_l, c_1, \ldots, c_k, d_1, \ldots, d_l \in \mathbb{N}$ such that, if $\mathbf{y} = f(\mathbf{x})$, then for each $j \in \{1, \ldots, l\}$, $\mathbf{y}(j) = b_j + \frac{1}{d_j} \sum_{i=1}^{k} n_{i,j}(\mathbf{x}(i) - c_i)$, and for each $i \in \{1, \ldots, k\}$, $\mathbf{x}(i) - c_i \geq 0$. Each d_j may be taken to be the least common multiple of the denominators of the rational coefficients in the original definition. We employ this latter definition, since it is more convenient for working with integer-valued molecular counts.

2.1 Chemical reaction networks

If Λ is a finite set (in this paper, of chemical species), we write $\mathbb{N}^\Lambda$ to denote the set of functions $f : \Lambda \to \mathbb{N}$. 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 $X \in \Lambda$ and $\mathbf{c} \in \mathbb{N}^\Lambda$, we refer to $\mathbf{c}(X)$ as the *count of X in $\mathbf{c}$*. We write $\mathbf{c} \leq \mathbf{c}'$ to denote that $\mathbf{c}(X) \leq \mathbf{c}'(X)$ for all $X \in \Lambda$. 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 $n\mathbf{c}$ for $n \in \mathbb{N}$. If $\Delta \subset \Lambda$, we view a vector $\mathbf{c} \in \mathbb{N}^\Delta$ equivalently as a vector $\mathbf{c} \in \mathbb{N}^\Lambda$ by assuming $\mathbf{c}(X) = 0$ for all $X \in \Lambda \setminus \Delta$.

Given a finite set of chemical species Λ, a *reaction* over Λ is a triple $\alpha = \langle \mathbf{r}, \mathbf{p}, k \rangle \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda \times \mathbb{R}^+$, specifying the stoichiometry of the reactants and products, respectively, and the *rate constant* k. If not specified, assume that $k = 1$ (this is the case for all reactions in this paper), so that the reaction $\alpha = \langle \mathbf{r}, \mathbf{p}, 1 \rangle$ is also represented by the pair $\langle \mathbf{r}, \mathbf{p} \rangle$. For instance, given $\Lambda = \{A, B, C\}$, the reaction $A + 2B \to A + 3C$ is the pair $\langle (1, 2, 0), (1, 0, 3) \rangle$. A *(finite) chemical reaction network (CRN)* is a pair $\mathcal{C} = (\Lambda, R)$, where Λ is a finite set of chemical *species*, and R is a finite set of reactions over Λ. A *configuration* of a CRN $\mathcal{C} = (\Lambda, R)$ is a vector $\mathbf{c} \in \mathbb{N}^\Lambda$. If some current configuration $\mathbf{c}$ is understood from context, we write $\#X$ to denote $\mathbf{c}(X)$.

Given a configuration $\mathbf{c}$ and reaction $\alpha = \langle \mathbf{r}, \mathbf{p} \rangle$, we say that α is *applicable* to $\mathbf{c}$ if $\mathbf{r} \leq \mathbf{c}$ (i.e., $\mathbf{c}$ contains enough of each of the reactants for the reaction to occur). If α is applicable to $\mathbf{c}$, then write $\alpha(\mathbf{c})$ to denote the configuration $\mathbf{c} + \mathbf{p} - \mathbf{r}$ (i.e., the configuration that results from applying reaction α to $\mathbf{c}$). If $\mathbf{c}' = \alpha(\mathbf{c})$ for some reaction $\alpha \in R$, we write $\mathbf{c} \to_\mathcal{C} \mathbf{c}'$, or merely $\mathbf{c} \to \mathbf{c}'$ when $\mathcal{C}$ is clear from context. An *execution* (a.k.a., *execution sequence*) $\mathcal{E}$ is a finite or infinite sequence of one or more configurations $\mathcal{E} = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \ldots)$ such that, for all $i \in \{1, \ldots, |\mathcal{E}| - 1\}$, $\mathbf{c}_{i-1} \to \mathbf{c}_i$. If a finite execution sequence starts with $\mathbf{c}$ and ends with $\mathbf{c}'$, we write $\mathbf{c} \to^*_\mathcal{C} \mathbf{c}'$, or merely $\mathbf{c} \to^* \mathbf{c}'$ when the CRN $\mathcal{C}$ is clear from context. In this case, we say that $\mathbf{c}'$ is *reachable* from $\mathbf{c}$.

Turing machines, for example, have different semantic interpretations depending on the computational task under study (deciding a language, computing a function, etc.). Similarly, in this paper we use CRNs to decide subsets of $\mathbb{N}^k$ (for which we reserve the term "chemical reaction *decider*" or CRD) and to compute functions $f : \mathbb{N}^k \to \mathbb{N}^l$ (for which we reserve the term "chemical reaction *computer*" or CRC). In the next two subsections we define two semantic interpretations of CRNs that correspond to these two tasks. We use the term CRN to refer to either a CRD or CRC when the statement is applicable to either type.

These definitions differ slightly from those of [4], because ours are specialized to "leaderless" CRNs: those that can compute a predicate or function in which no species are present in the initial configuration other than the input species. In the terminology of [4], a CRN with species set Λ and input species set Σ is *leaderless* if it has an *initial context* $\sigma : \Lambda \setminus \Sigma \to \mathbb{N}$ such that $\sigma(S) = 0$ for all $S \in \Lambda \setminus \Sigma$. The definitions below are simplified by assuming this to be true of all CRNs.

We also use the convention of Angluin, Aspnes, and Eisenstat [2] that for a CRD, all species "vote" yes or no, rather than only a subset of species as in [4], since this convention is convenient for proving time bounds.

2.2 Stable decidability of predicates

We now review the definition of stable decidability of predicates introduced by Angluin, Aspnes, and Eisenstat [2].[3] Intuitively, the set of species is partitioned into two sets: those that "vote" yes and those that vote no, and the system stabilizes to an output when a consensus vote is reached (all positive-count species have the same vote) that can no longer be changed (no species voting the other way can ever again be produced). It would be too strong to characterize deterministic correctness by requiring all possible executions to achieve the correct answer; for example, a reversible reaction such as $A \rightleftharpoons B$ could simply be chosen to run back and forth forever, starving any other reactions.

[3] Those authors use the term "stably *compute*", but we reserve the term "compute" to apply to the computation of non-Boolean functions. Also, we omit discussion of the definition of stable computation used in the population protocols literature, which employs a notion of "fair" executions; the definitions are proven equivalent in [4].

In the more refined definition that follows, the determinism of the system is captured in that it is impossible to stabilize to an incorrect answer, and the correct stable output is always reachable.

A *(leaderless) chemical reaction decider* (CRD) is a tuple $\mathcal{D} = (\Lambda, R, \Sigma, \Upsilon)$, where (Λ, R) is a CRN, $\Sigma \subseteq \Lambda$ is the *set of input species*, and $\Upsilon \subseteq \Lambda$ is the set of *yes voters*, with species in $\Lambda \setminus \Upsilon$ referred to as *no voters*. An input to $\mathcal{D}$ will be an *initial configuration* $\mathbf{i} \in \mathbb{N}^{\Sigma}$ (equivalently, $\mathbf{i} \in \mathbb{N}^k$ if we write $\Sigma = \{X_1, \ldots, X_k\}$ and assign X_i to represent the i'th coordinate); that is, only input species are allowed to be non-zero. If we are discussing a CRN understood from context to have a certain initial configuration $\mathbf{i}$, we write $\#_0 X$ to denote $\mathbf{i}(X)$.

We define a global output partial function $\Phi : \mathbb{N}^{\Lambda} \dashrightarrow \{0, 1\}$ as follows. $\Phi(\mathbf{c})$ is undefined if either $\mathbf{c} = \mathbf{0}$, or if there exist $S_0 \in \Lambda \setminus \Upsilon$ and $S_1 \in \Upsilon$ such that $\mathbf{c}(S_0) > 0$ and $\mathbf{c}(S_1) > 0$. Otherwise, either $(\forall S \in \Lambda)(\mathbf{c}(S) > 0 \implies S \in \Upsilon)$ or $(\forall S \in \Lambda)(\mathbf{c}(S) > 0 \implies S \in \Lambda \setminus \Upsilon)$; in the former case, the *output* $\Phi(\mathbf{c})$ of configuration $\mathbf{c}$ is 1, and in the latter case, $\Phi(\mathbf{c}) = 0$.

A configuration $\mathbf{o}$ is *output stable* if $\Phi(\mathbf{o})$ is defined and, for all $\mathbf{c}$ such that $\mathbf{o} \to^* \mathbf{c}$, $\Phi(\mathbf{c}) = \Phi(\mathbf{o})$. We say a CRD $\mathcal{D}$ *stably decides* the predicate $\psi : \mathbb{N}^{\Sigma} \to \{0, 1\}$ if, for any initial configuration $\mathbf{i} \in \mathbb{N}^k$, for all configurations $\mathbf{c} \in \mathbb{N}^{\Lambda}$, $\mathbf{i} \to^* \mathbf{c}$ implies $\mathbf{c} \to^* \mathbf{o}$ such that $\mathbf{o}$ is output stable and $\Phi(\mathbf{o}) = \psi(\mathbf{i})$. Note that this condition implies that no incorrect output stable configuration is reachable from $\mathbf{i}$. We say that $\mathcal{D}$ *stably decides* a set $A \in \mathbb{N}^k$ if it stably decides its indicator function.

The following theorem is due to Angluin, Aspnes, and Eisenstat [2]:

Theorem 2.1 ([2])**.** *A set $A \subseteq \mathbb{N}^k$ is stably decidable by a CRD if and only if it is semilinear.*

The model they use is defined in a slightly different way; the differences (and those differences' lack of significance to the questions we explore) are explained in [4].

2.3 Stable computation of functions

We now define a notion of stable computation of *functions* similar to those above for predicates. Intuitively, the inputs to the function are the initial counts of input species $X_1, \ldots, X_k$, and the outputs are the counts of output species $Y_1, \ldots, Y_l$. The system stabilizes to an output when the counts of the output species can no longer change. Again determinism is captured in that it is impossible to stabilize to an incorrect answer and the correct stable output is always reachable.

A *(leaderless) chemical reaction computer (CRC)* is a tuple $\mathcal{C} = (\Lambda, R, \Sigma, \Gamma)$, where (Λ, R) is a CRN, $\Sigma \subset \Lambda$ is the *set of input species*, $\Gamma \subset \Lambda$ is the *set of output species*, such that $\Sigma \cap \Gamma = \varnothing$. By convention, we let $\Sigma = \{X_1, X_2, \ldots, X_k\}$ and $\Gamma = \{Y_1, Y_2, \ldots, Y_l\}$. We say that a configuration $\mathbf{o}$ is *output stable* if, for every $\mathbf{c}$ such that $\mathbf{o} \to^* \mathbf{c}$ and every $Y_i \in \Gamma$, $\mathbf{o}(Y_i) = \mathbf{c}(Y_i)$ (i.e., the counts of species in Γ will never change if $\mathbf{o}$ is reached). As with CRD's, we require initial configurations $\mathbf{i} \in \mathbb{N}^{\Sigma}$ in which only input species are allowed to be positive. We say that $\mathcal{C}$ *stably computes* a function $f : \mathbb{N}^k \to \mathbb{N}^l$ if for any initial configuration $\mathbf{i} \in \mathbb{N}^{\Sigma}$, $\mathbf{i} \to^* \mathbf{c}$ implies there exists $\mathbf{o}$ such that $\mathbf{c} \to^* \mathbf{o}$ and $\mathbf{o}$ is an output stable configuration with $f(\mathbf{i}) = (\mathbf{o}(Y_1), \mathbf{o}(Y_2), \ldots, \mathbf{o}(Y_l))$. Note that this condition implies that no incorrect output stable configuration is reachable from $\mathbf{i}$.

If a CRN stably decides a predicate or stably computes a function, we say the CRN is *stable* (a.k.a., *deterministic*).

If $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$ is a partial function undefined on some inputs, we say that a CRC $\mathcal{C}$ stably computes f if $\mathcal{C}$ stably computes f on all inputs $\mathbf{x} \in \operatorname{dom} f$, with no constraint on the behavior of $\mathcal{C}$ if it is given an input $\mathbf{x} \notin \operatorname{dom} f$.

2.4 Kinetic model

The following model of stochastic chemical kinetics is widely used in quantitative biology and other fields dealing with chemical reactions between species present in small counts [10]. It ascribes probabilities to execution sequences, and also defines the time of reactions, allowing us to study the computational complexity of the CRN computation in Section 3.

In this paper, the rate constants of all reactions are 1, and we define the kinetic model with this assumption. The rate constants do not affect the definition of stable computation; they only affect the time analysis. Our time analyses remain asymptotically unaffected if the rate constants are changed (although the constants hidden in the big-O notation would change). A reaction is *unimolecular* if it has one reactant and *bimolecular* if it has two reactants. We use no higher-order reactions in this paper.

The kinetics of a CRN is described by a continuous-time Markov process as follows. Given a fixed volume $v \in \mathbb{R}^+$ and current configuration $\mathbf{c}$, the *propensity* of a unimolecular reaction $\alpha : X \to \ldots$ in configuration $\mathbf{c}$ is $\rho(\mathbf{c}, \alpha) = \mathbf{c}(X)$. The propensity of a bimolecular reaction $\alpha : X + Y \to \ldots$, where $X \neq Y$, is $\rho(\mathbf{c}, \alpha) = \frac{\mathbf{c}(X)\mathbf{c}(Y)}{v}$. The propensity of a bimolecular reaction $\alpha : X + X \to \ldots$ is $\rho(\mathbf{c}, \alpha) = \frac{1}{2}\frac{\mathbf{c}(X)(\mathbf{c}(X)-1)}{v}$. The propensity function determines the evolution of the system as follows. The time until the next reaction occurs is an exponential random variable with rate $\rho(\mathbf{c}) = \sum_{\alpha \in R} \rho(\mathbf{c}, \alpha)$ (note that $\rho(\mathbf{c}) = 0$ if no reactions are applicable to $\mathbf{c}$). Therefore, the expected time for the next reaction to occur is $\frac{1}{\rho(\mathbf{c})}$.

The kinetic model is based on the physical assumption that the solution is well-mixed, which is valid if the solution is sufficiently dilute. Thus, we assume the *finite density constraint*, which stipulates that a volume required to execute a CRN must be proportional to the maximum molecular count obtained during execution [16]. In other words, the total concentration (molecular count per volume) is bounded. This realistically constrains the speed of the computation achievable by CRNs. Note, however, that it is problematic to define the kinetic model for CRNs in which the reachable configuration space is unbounded for some start configurations, because this means that arbitrarily large molecular counts are reachable.[4] We apply the kinetic model only to CRNs with configuration spaces that are bounded for each start configuration, choosing the volume to be equal to the reachable configuration with the highest molecular count (in this paper, this will always be within a constant multiplicative factor of the number of input molecules).

It is not difficult to show that if a CRN is stable and has a finite reachable configuration space from any initial configuration $\mathbf{i}$, then under the kinetic model (in fact, for any choice of rate constants), with probability 1 the CRN will eventually reach an output stable configuration.

We require the following lemmas later in our main theorems. Some of these are implicit or explicit in many earlier papers on stochastic CRNs, but we include their proofs for the sake of self-containment.

The lemmas are stated with respect to a certain "initial configuration" $\mathbf{c}$ that may not be the initial configuration of an actual CRN we define. This is because the lemmas are employed to argue about CRNs that are guaranteed to evolve to some configuration $\mathbf{c}$ that satisfies the hypothesis of the lemma, and we use the lemma to bound the time it takes for the CRN to complete a sequence of reactions, starting from $\mathbf{c}$. Therefore terms such as "applicable reaction" refer to being applicable from $\mathbf{c}$ and any configuration reachable from it, although some additional inapplicable reactions may have been applicable prior to reaching the configuration $\mathbf{c}$.

[4] One possibility is to have a "dynamically" growing volume as in [16].

Lemma 2.2. Let $\mathbf{c}$ be a configuration. Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a set of species with the property that, for all configurations reachable from $\mathbf{c}$, every applicable reaction in which any species in $\mathcal{A}$ appears is of the form $A_i \to B_1 + \ldots + B_l$, where each $B_{i'} \notin \mathcal{A}$ for $1 \leq i' \leq l$. Then starting from a configuration $\mathbf{c}$ in which for all $i \in \{1, \ldots, m\}$, $\mathbf{c}(A_i) = O(n)$, the expected time to reach from $\mathbf{c}$ to a configuration in which none of the described reactions can occur is $O(\log n)$.

Proof. Assume the hypothesis. Let $c \in \mathbb{N}$ be the constant such that $\sum_{i=1}^{m} \mathbf{c}(A_i) \leq cn$. After each relevant reaction occurs, this sum is reduced by 1. Therefore no reactions can occur after cn reactions have executed. If $\sum_{i=1}^{m} \#A_i = k$, the expected time for any reaction to occur is $\frac{1}{k}$. By linearity of expectation, the expected time for cn reactions to execute is at most $\sum_{k=1}^{cn} \frac{1}{k} = O(\log n)$. □

Lemma 2.3. Let $\mathbf{c}$ be a configuration. Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a set of species with the property that, for all configurations reachable from $\mathbf{c}$, every applicable reaction in which any species in $\mathcal{A}$ appears is of the form $A_i + A_j \to A_p + B_1 + \ldots + B_l$, where each $B_{i'} \notin \mathcal{A}$ for $1 \leq i' \leq l$, and for all $i, j \in \{1, \ldots, m\}$, there is at least one reaction $A_i + A_j \to \ldots$ of this form. Then starting from a configuration $\mathbf{c}$ in which for all $i \in \{1, \ldots, m\}$, $\mathbf{c}(A_i) = O(n)$, with volume $O(n)$, the expected time to reach a configuration in which none of the described reactions can occur is $O(n)$.

Proof. Assume the hypothesis. Let $c \in \mathbb{N}$ be a constant such that $\sum_{i=1}^{m} \mathbf{c}(A_i) \leq cn$, and let c' be a constant such that the volume is at most $c'n$. After each relevant reaction occurs, this sum is reduced by 1. Therefore no reactions can occur after $cn - 1$ reactions have executed. Now let $\rho(\mathbf{c}, \alpha_{ij})$ be the propensity of the reaction $A_i + A_j \to A_p + B_1 + \ldots + B_l$, which is equal to $\rho(\mathbf{c}, \alpha_{ji})$ as well, and if there is more than one reaction of that form, let $\rho(\mathbf{c}, \alpha_{ij})$ represent the rate of one of those reactions selected arbitrarily. Since A_i can react with A_j for any $i, j \in \{1, \ldots, m\}$, given that $\sum_{i=1}^{m} \#A_i = k$, the time for the next reaction to occur is an exponential random variable with rate equal to the sum of the rates of each possible reaction, i.e.,

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \geq i}}^{m} \rho(\mathbf{c}, \alpha_{ij}) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} \rho(\mathbf{c}, \alpha_{ij}) + \sum_{i=1}^{m} \rho(\mathbf{c}, \alpha_{ii}) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} \frac{\#A_i \#A_j}{c'n} + \sum_{i=1}^{m} \frac{\#A_i(\#A_i - 1)}{2c'n} \\
&= \frac{1}{2c'n} \left[\sum_{i=1}^{m} \sum_{j=1}^{m} \#A_i \#A_j - \sum_{i=1}^{m} \#A_i^2 \right] + \frac{1}{2c'n} \sum_{i=1}^{m} \#A_i(\#A_i - 1) \\
&= \frac{1}{2c'n} \left[\sum_{i=1}^{m} \#A_i \left(\sum_{j=1}^{m} \#A_j \right) - \sum_{i=1}^{m} \#A_i \right] \\
&= \frac{1}{2c'n}(k^2 - k)
\end{aligned}$$

so the expected time for the next reaction to occur is $\frac{2c'n}{k^2-k}$. By linearity of expectation, the expected time for $cn-1$ reactions to execute is at most $\sum_{k=1}^{cn-1} \frac{c'n}{k^2-k} = c'n \sum_{k=1}^{cn-1} (\frac{1}{k-1} - \frac{1}{k}) = c'n(1 - \frac{1}{cn-1}) = O(n)$. □

Lemma 2.4. Let $\mathbf{c}$ be a configuration. Let $\mathcal{C} = \{C_1, \ldots, C_p\}$ and $\mathcal{A} = \{A_1, \ldots, A_m\}$ be two sets of species with the property that, for all configurations reachable from $\mathbf{c}$, every applicable reaction in which any species in $\mathcal{A}$ or $\mathcal{C}$ appears is of the form $C_i + A_j \to C_i + B_1 + \ldots + B_l$, where each $B_{i'} \notin \mathcal{A}$ for $1 \leq i' \leq l$. Then starting from a configuration $\mathbf{c}$ in which for all $i \in \{1, \ldots, p\}$, $\mathbf{c}(C_i) = \Omega(n)$, and for all $j \in \{1, \ldots, m\}$, $\mathbf{c}(A_j) = O(n)$, with volume $O(n)$, the expected time to reach a configuration in which none of the described reactions can occur is $O(\log n)$.

Proof. Assume the hypothesis. Then the counts of each C_i do not decrease. (They may increase if some $B_l \in \mathcal{C}$, but this only strengths the conclusion.) Therefore this is similar to the proof of Lemma 2.2, since for each k, the expected time of each reaction when $\sum_{j=1}^{m} \#A_j = k$ is within a constant of $\frac{1}{k}$. Thus by linearity of expectation, the expected time for cn (i.e., $\sum_{i=1}^{m} \mathbf{c}(A_i) \leq cn$) reactions to occur is at most $\sum_{k=1}^{cn} \frac{1}{k} = O(\log n)$. □

3 Leaderless CRCs can compute semilinear functions

To supply an input vector $\mathbf{x} \in \mathbb{N}^k$ to a CRN, we use an initial configuration with $\mathbf{x}(i)$ molecules of input species X_i. Throughout this section, we let $n = ||\mathbf{x}||_1 = \sum_{i=1}^{k} \mathbf{x}(i)$ denote the initial number of molecules in solution. Since all CRNs we employ have the property that they produce at most a constant multiplicative factor more molecules than are initially present, this implies that the volume required to satisfy the finite density constraint is $O(n)$.

Suppose the CRC $\mathcal{C}$ stably computes a function $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$. We say that $\mathcal{C}$ stably computes f *monotonically* if its output species are not consumed in any reaction.[5]

We show in Lemma 3.1 that affine partial functions can be computed in expected time $O(n)$ by a leaderless CRC. For its use in proving Theorem 3.4, we require that the output molecules be produced monotonically. If we used a direct encoding of the output of the function, this would be impossible for general affine functions. For example, consider the function $f(x_1, x_2) = x_1 - x_2$ where dom $f = \{ (x_1, x_2) \mid x_1 \geq x_2 \}$. By withholding a single copy of X_2 and letting the CRC stabilize to the output value $\#Y = x_1 - x_2 + 1$, then allowing the extra copy of X_2 to interact, the only way to stabilize to the correct output value $x_1 - x_2$ is to consume a copy of the output species Y. Therefore Lemma 3.1 computes f indirectly via an encoding of f's output that allows monotonic production of outputs, encoding the output value $\mathbf{y}(j)$ as the difference between the counts of two monotonically produced species Y_j^P and Y_j^C, a concept formalized by the following definition.

Let $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$ be a partial function. We say that a partial function $\hat{f} : \mathbb{N}^k \dashrightarrow \mathbb{N}^l \times \mathbb{N}^l$ is a *diff-representation* of f if dom f = dom $\hat{f}$ and, for all $\mathbf{x} \in$ dom f, if $(\mathbf{y}_P, \mathbf{y}_C) = \hat{f}(\mathbf{x})$, where $\mathbf{y}_P, \mathbf{y}_C \in \mathbb{N}^l$, then $f(\mathbf{x}) = \mathbf{y}_P - \mathbf{y}_C$, and $\mathbf{y}_P = O(f(\mathbf{x}))$.[6] In other words, $\hat{f}$ represents f as the difference of its two outputs $\mathbf{y}_P$ and $\mathbf{y}_C$, with the larger output $\mathbf{y}_P$ possibly being larger than the original function's output, but is at most by a multiplicative constant larger.

The following lemma is the main technical result required for proving our main theorem, Theorem 3.4. It shows that every affine function can be computed (via a diff-representation) in time $O(n)$ by a leaderless CRC. The example in Figure 1 also clarifies the essence of the leaderless computation of affine functions.

[5] Its output species could potentially be reactants so long as they are catalytic, meaning that the stoichiometry of the species as a product is at least as great as its stoichiometry as a reactant, e.g. if Y is the output species, $X + Y \to Z + Y$ or $A + Y \to Y + Y$.

[6] By $\mathbf{y}_P = O(f(\mathbf{x}))$, we mean that there is a constant c such that $y_P \leq cf(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{N}^k$.

Lemma 3.1. *Let $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$ be an affine partial function with $f(\mathbf{0}) = \mathbf{0}$ if $\mathbf{0} \in \text{dom } f$. Then there is a diff-representation $\hat{f} : \mathbb{N}^k \dashrightarrow \mathbb{N}^l \times \mathbb{N}^l$ of f and a leaderless CRC that monotonically stably computes $\hat{f}$ in expected time $O(n)$.*

Proof. If f is affine, then there exist kl integers $n_{1,1}, \ldots, n_{k,l} \in \mathbb{Z}$ and $2l + k$ nonnegative integers $b_1, \ldots, b_l, c_1, \ldots, c_k, d_1, \ldots, d_l \in \mathbb{N}$ such that, if $\mathbf{y} = f(\mathbf{x})$, then for each $j \in \{1, \ldots, l\}$, $\mathbf{y}(j) = b_j + \frac{1}{d_j} \sum_{i=1}^{k} n_{i,j}(\mathbf{x}(i) - c_i)$, and for each $i \in \{1, \ldots, k\}$, $\mathbf{x}(i) - c_i \geq 0$. Note in particular that since the range of f is $\mathbb{N}^l$, the value $b_j + \frac{1}{d_j} \sum_{i=1}^{k} n_{i,j}(\mathbf{x}(i) - c_i)$ must be an integer.

Define the CRC as follows. It has input species $\Sigma = \{X_1, \ldots, X_k\}$ and output species $\Gamma = \{Y_1^P, \ldots, Y_l^P, Y_1^C, \ldots, Y_l^C\}$.

There are three main components of the CRN, separately handling the c_i offset, the $n_{i,j}/d_j$ coefficient, and the b_j offset.

For a species S that stabilizes to a fixed count depending only on the input configuration, write $\#_\infty S$ to denote the eventual stable count of S (in the case of Y_j^P and Y_j^C, this will be the same as the total amount ever produced, since they are never consumed). The latter two components both make use of Y_j^C molecules to account for production of Y_j^P molecules in excess of $\mathbf{y}(j)$ to ensure that $\#_\infty Y_j^P - \#_\infty Y_j^C = \mathbf{y}(j)$, which establishes that the CRC stably computes a diff-representation of f. It is clear by inspection of the reactions that $\#_\infty Y_j^P = O(\mathbf{y}(j))$.

For all input species X_i $(1 \leq i \leq k)$, add the reaction

$$X_i \to C_{i,1} + B_1 + B_2 + \ldots + B_l + b_1 Y_1^P + b_2 Y_2^P + \ldots b_l Y_l^P \tag{3.1}$$

The first product $C_{i,1}$ will be used to handle the c_i offset, and the remaining products will be used to handle the b_j offsets. By Lemma 2.2, reaction (3.1) takes time $O(\log n)$ to complete.

We now describe the three components of the CRC separately.

c_i **offset:** Reaction (3.1) produces $\mathbf{x}(i)$ copies of $C_{i,1}$. We must reduce this number by c_i, producing $\mathbf{x}(i) - c_i$ copies of X_i', the species that will be used by the next component to handle the $n_{i,j}/d_j$ coefficient. A high-order reaction implementing this is $(c_i+1)C_{i,1} \to c_i C_{i,1} + X_i'$, since that reaction will eventually happen exactly $\mathbf{x}(i) - c_i$ times (stopping when $\#C_{i,1}$ reaches c_i). This is implemented by the following bimolecular reactions.

For each $i \in \{1, \ldots, k\}$ and $m, p \in \{1, \ldots, c_i\}$, if $m + p \leq c_i$, add the reaction

$$C_{i,m} + C_{i,p} \to C_{i,m+p}.$$

If $m + p > c_i$, add the reaction

$$C_{i,m} + C_{i,p} \to C_{i,c_i} + (m + p - c_i)X_i'.$$

By Lemma 2.3, these reactions complete in expected time $O(n)$.

Note that although the final reaction above may have a large number of products, it is straightforward to simulate any such reaction with products $P_1, \ldots, P_\ell$ with reactions having two products only, e.g., the first product is P_1', followed by reactions $P_i' \to P_{i+1}' + P_i$ for each $i \in \{1, \ldots, \ell - 2\}$, and $P_{\ell-1}' \to P_{\ell-1} + P_\ell$.

$n_{i,j}/d_j$ **coefficient:** For each $i \in \{1, \ldots, k\}$, add the reaction

$$X_i' \to X_{i,1} + X_{i,2} + \ldots + X_{i,l}$$

This allows each output to be associated with its own copy of the input. By Lemma 2.2, these reactions complete in expected time $O(\log n)$.

For each $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, l\}$, add the reaction

$$X_{i,j} \to \begin{cases} n_{i,j} D^P_{j,1}, & \text{if } n_{i,j} > 0; \\ (-n_{i,j}) D^C_{j,1}, & \text{if } n_{i,j} < 0. \end{cases}$$

By Lemma 2.2, these reactions complete in expected time $O(\log n)$.

We must now divide $\#D^P_{j,1}$ and $\#D^C_{j,1}$ by d_j. This is accomplished by the high-order reactions $d_j D^P_{j,1} \to Y^P_j$ and $d_j D^C_{j,1} \to Y^C_j$. Similarly to the previous component, we implement these with the following reactions for $d_j \geq 1$.

We first handle the case $d_j > 1$. For each $j \in \{1, \ldots, l\}$ and $m, p \in \{1, \ldots, d_j - 1\}$, if $m + p \leq d_j - 1$, add the reactions

$$\begin{aligned} D^P_{j,m} + D^P_{j,p} &\to D^P_{j,m+p} \\ D^C_{j,m} + D^C_{j,p} &\to D^C_{j,m+p} \end{aligned}$$

If $m + p > c_i$, add the reactions

$$\begin{aligned} D^P_{j,m} + D^P_{j,p} &\to D^P_{j,m+p-d_j} + Y^P_j \\ D^C_{j,m} + D^C_{j,p} &\to D^C_{j,m+p-d_j} + Y^C_j \end{aligned}$$

By Lemma 2.3, these reactions complete in expected time $O(n)$.

When $d_j = 1$, we only have the following unimolecular reactions.

$$\begin{aligned} D^P_{j,1} &\to Y^P_j \\ D^C_{j,1} &\to Y^C_j \end{aligned}$$

By Lemma 2.2, these reactions complete in expected time $O(\log n)$.

These reactions will produce $\frac{1}{d_j} \sum_{n_{i,j}>0} n_{i,j}(\mathbf{x}(i) - c_i)$ copies of Y^P_j and $-\frac{1}{d_j} \sum_{n_{i,j}<0} n_{i,j}(\mathbf{x}(i) - c_i)$ copies of Y^C_j. Therefore, letting $\#_{\text{coef}} Y^P_j$ and $\#_{\text{coef}} Y^C_j$ denote the number of copies of Y^P_j and Y^C_j eventually produced just by this component, it holds that $\#_{\text{coef}} Y^P_j - \#_{\text{coef}} Y^C_j = \frac{1}{d_j} \sum_{i=1}^k n_{i,j}(\mathbf{x}(i) - c_i)$.

b_j **offset:** For each $j \in \{1, \ldots, l\}$, add the reaction

$$B_j + B_j \to B_j + b_j Y^C_j \tag{3.2}$$

By Lemma 2.3, these reactions complete in expected time $O(n)$.

Reaction (3.1) produces b_j copies of Y^P_j for each copy of B_j produced, which is $\sum_{i=1}^k \mathbf{x}(i)$. Reaction (3.2) occurs precisely $(\sum_{i=1}^k \mathbf{x}(i)) - 1$ times. Therefore reaction (3.2) produces precisely b_j fewer copies of Y^C_j than reaction (3.1) produces of Y^P_j. This implies that when all copies of Y^C_j are eventually produced by reaction (3.2), the number of Y^P_j's produced by reaction (3.1) minus the number of Y^C_j's produced by reaction (3.2) is b_j. □

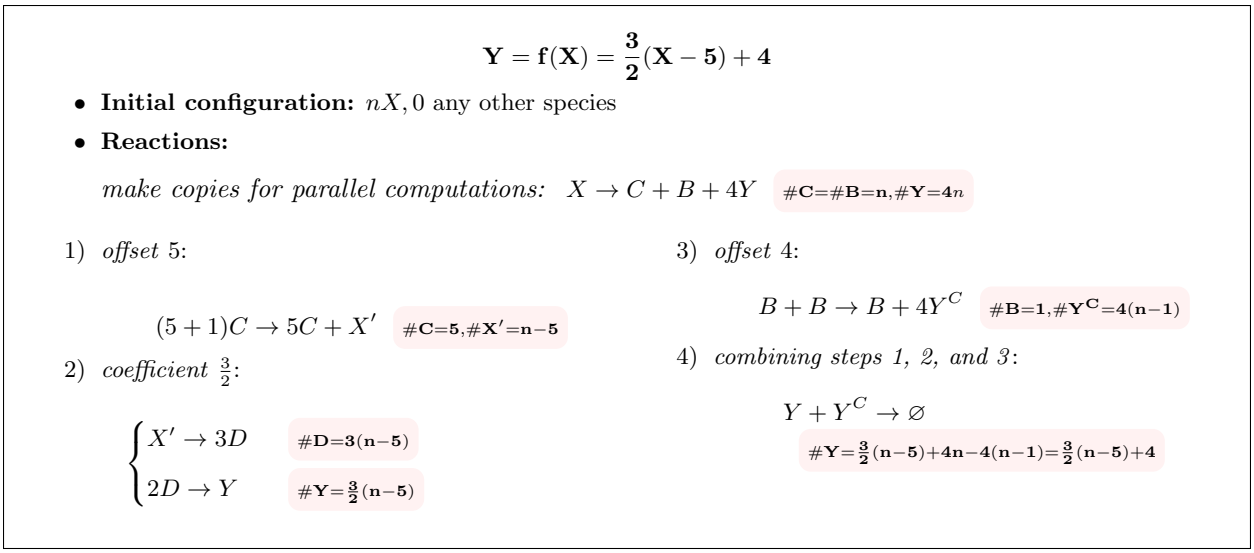

Figure 1: An example consisting of the essential steps of our leaderless construction for the deterministic computation of partial affine functions (3.1).

We require the following lemma, proven in [4].

Lemma 3.2 ([4])**.** *Let $f : \mathbb{N}^k \to \mathbb{N}^l$ be a semilinear function. Then there is a finite set $\{f_1 : \mathbb{N}^k \dashrightarrow \mathbb{N}^l, \ldots, f_m : \mathbb{N}^k \dashrightarrow \mathbb{N}^l\}$ of affine partial functions, where each $\operatorname{dom} f_i$ is a linear set, such that, for each $\mathbf{x} \in \mathbb{N}^k$, if $f_i(\mathbf{x})$ is defined, then $f(\mathbf{x}) = f_i(\mathbf{x})$, and $\bigcup_{i=1}^m \operatorname{dom} f_i = \mathbb{N}^k$.*

We require the following theorem, due to Angluin, Aspnes, and Eisenstat [3, Theorem 5], which states that any semilinear predicate can be decided by a CRD in expected time $O(n)$.

Theorem 3.3 ([3])**.** *Let $\phi : \mathbb{N}^k \to \{0, 1\}$ be a semilinear predicate. Then there is a leaderless CRD $\mathcal{D}$ that stably decides ϕ (so long as some positive number of molecules are initially present), and the expected time to reach an output-stable configuration is $O(n)$.*

The following is the main theorem of this paper. It shows that semilinear functions can be computed by leaderless CRCs in linear expected time.

Theorem 3.4. *Let $f : \mathbb{N}^k \to \mathbb{N}^l$ be a semilinear function with $f(\mathbf{0}) = \mathbf{0}$. Then there is a leaderless CRC that stably computes f in expected time $O(n)$.*

Proof. The CRC will have input species $\Sigma = \{X_1, \ldots, X_k\}$ and output species $\Gamma = \{Y_1, \ldots, Y_l\}$. By Lemma 3.2, there is a finite set $F = \{f_1 : \mathbb{N}^k \dashrightarrow \mathbb{N}^l, \ldots, f_m : \mathbb{N}^k \dashrightarrow \mathbb{N}^l\}$ of affine partial functions, where each $\operatorname{dom} f_i$ is a linear set, such that, for each $\mathbf{x} \in \mathbb{N}^k$, if $f_i(\mathbf{x})$ is defined, then $f(\mathbf{x}) = f_i(\mathbf{x})$. We compute f on input $\mathbf{x}$ as follows. Since each $\operatorname{dom} f_i$ is a linear (and therefore semilinear) set, by Theorem 3.3 we compute each semilinear predicate $\phi_i =$ "$\mathbf{x} \in \operatorname{dom} f_i$ and $(\forall i' \in \{1, \ldots, i-1\})$ $\mathbf{x} \notin \operatorname{dom} f_{i'}$?" by separate parallel CRD's each stabilizing in expected time $O(n)$. (The latter condition ensures that for each $\mathbf{x}$, precisely one of the predicates is true, in case the domains of the partial functions have nonempty intersection.) Here we are relying on the fact that Boolean combinations (union, intersection, complement) of semilinear sets are semilinear [11].

By Lemma 3.1, for each $i \in \{1, \ldots, m\}$, there is a diff-representation $\hat{f}_i$ of f_i that can be stably computed by parallel CRC's. Assume that for each $i \in \{1, \ldots, m\}$ and each $j \in \{1, \ldots, l\}$, the jth pair of outputs $\mathbf{y}_P(j)$ and $\mathbf{y}_C(j)$ of the ith function is represented by species $\hat{Y}^P_{i,j}$ and $\hat{Y}^C_{i,j}$. We interpret each $\hat{Y}^P_{i,j}$ and $\hat{Y}^C_{i,j}$ as an "inactive" version of "active" output species $Y^P_{i,j}$ and $Y^C_{i,j}$.

For each $i \in \{1, \ldots, m\}$, for the CRD $\mathcal{D}_i = (\Lambda, R, \Sigma, \Upsilon)$ computing the predicate ϕ_i, let L^1_i represent any species in Υ, and L^0_i represent any species in $\Lambda \setminus \Upsilon$, and that once $\mathcal{D}_i$ reaches an output stable configuration, where b is the output of $\mathcal{D}_i$. By Theorem 5 of [3], if the total count of L^1_i and L^0_i molecules is $\Omega(n)$ (which can be enforced by adding both L^1_i and L^0_i as products of the initial reactions of the input molecules: $X_i \to L^1_i + L^0_i + \ldots$), then the correct vote can be spread through the population of L^b_i molecules (for $b \in \{0, 1\}$) in time $O(n)$.

Add the following reactions for each $i \in \{1, \ldots, m\}$ and each $j \in \{1, \ldots, l\}$:

$$L^1_i + \hat{Y}^P_{i,j} \to L^1_i + Y^P_{i,j} + Y_j \tag{3.3}$$
$$L^0_i + Y^P_{i,j} \to L^0_i + M_{i,j} \tag{3.4}$$
$$M_{i,j} + Y_j \to \hat{Y}^P_{i,j} \tag{3.5}$$

The latter two reactions implement the reverse direction of the first reaction – using L^0_i as a catalyst instead of L^1_i – using only bimolecular reactions. Also add the reactions

$$L^1_i + \hat{Y}^C_{i,j} \to L^1_i + Y^C_{i,j} \tag{3.6}$$
$$L^0_i + Y^C_{i,j} \to L^0_i + \hat{Y}^C_{i,j} \tag{3.7}$$

and

$$Y^P_{i,j} + Y^C_{i,j} \to K_j \tag{3.8}$$
$$K_j + Y_j \to \varnothing \tag{3.9}$$

That is, a "yes" answer for function i activates the ith output and a "no" answer deactivates the ith output. Eventually each CRD stabilizes so that precisely one i has L^1_i present, and for all $i' \neq i$, $L^0_{i'}$ is present, which takes time $O(n)$ by Theorem 5 of [3]. We now claim that at this point, all outputs for the correct function $\hat{f}_i$ will be activated and all other outputs will be deactivated. The reactions enforce that at any time, $\#Y_j = \#K_j + \sum_{i=1}^m (\#Y^P_{i,j} + \#M_{i,j})$. In particular, $\#Y_j \geq \#K_j$ and $\#Y_j \geq \#M_{i,j}$ at all times, so there will never be a K_j or $M_{i,j}$ molecule that cannot participate in the reaction of which it is a reactant. Eventually $\#Y^P_{i,j}$ and $\#Y^C_{i,j}$ stabilize to 0 for all but one value of i (by reactions (3.4), (3.5), (3.7)), and for this value of i, $\#Y^P_{i,j}$ stabilizes to $\mathbf{y}(j)$ and $\#Y^C_{i,j}$ stabilizes to 0 (by reaction (3.8)). Eventually $\#K_j$ stabilizes to 0 by the last reaction. Eventually $\#M_{i,j}$ stabilizes to 0 since L^0_i is absent for the correct function $\hat{f}_i$. This ensures that $\#Y_j$ stabilizes to $\mathbf{y}(j)$.

It remains to analyze the expected time to stabilization. Let $n = \|\mathbf{x}\|$. By Lemma 3.1, the expected time for each affine function computation to complete is $O(n)$. Since the $\hat{Y}^P_{i,j}$ are produced monotonically, the most $Y^P_{i,j}$ molecules that are ever produced is $\#_\infty \hat{Y}^P_{i,j}$. Since we have m computations in parallel, the expected time for all of them to complete is $O(nm) = O(n)$ (since m depends on f but not n). We must also wait for each predicate computation to complete. By Theorem 3.3, each of these predicates takes expected time $O(n)$ to complete, so all of them complete in expected time $O(mn) = O(n)$.

At this point, the L_1^i leaders must convert inactive output species to active, and $L_0^{i'}$ (for $i' \neq i$) must convert active output species to inactive. By Lemma 2.4, reactions (3.3), (3.4), (3.6), and (3.7) complete in expected time $O(\log n)$. Once this is completed, by Lemma 2.3, reaction (3.5) completes in expected time $O(n)$. Reaction (3.8) completes in expected time $O(n)$ by Lemma 2.3. Once this is done, reaction (3.9) completes in expected time $O(n)$ by Lemma 2.3. □

4 Conclusion

We identify two general directions for future work.

4.1 Time complexity

The clearest shortcoming of our leaderless CRC, compared to the leader-employing CRC of [4], is the time complexity. Our CRC takes expected time $O(n)$ to complete with n input molecules, versus $O(\log^5 n)$ for the CRC of Theorem 4.4 of [4]. However, we do obtain a modest speedup ($O(n)$ versus $O(n \log n)$), compared to the *direct* construction of Theorem 4.1 of [4]. The indirect construction of Theorem 4.1 of [4] relied heavily on the use of a fast, error-prone CRN which computes arbitrary computable functions, and which crucially uses a leader. The major open question is, for each semilinear function $f : \mathbb{N}^k \to \mathbb{N}^l$, is there a leaderless CRC that stably computes f on input of size n in expected time $t(n)$, where t is a sublinear function? This may relate to the question of whether there is a sublinear time CRN that solves the leader election problem, i.e., in volume n with an initial state with n copies of species X and no other species initially present, produce a single copy of a species L. However, it is conceivable that there is a direct way to compute semilinear functions quickly without needing to use a leader election.

If this is not possible for all semilinear functions, another interesting open question is to precisely characterize the class of functions that can be stably computed by a leaderless CRC in polylogarithmic time. For example, the class of linear functions with positive integer coefficients (e.g., $f(x_1, x_2) = 3x_1 + 2x_2$) has this property since they are computable by $O(\log n)$-time unimolecular reactions such as $X_1 \to 3Y, X_2 \to 2Y$. However, most of the CRN programming techniques used to generalize beyond such functions seem to require some bimolecular reaction $A + B \to \ldots$ in which it is possible to have $\#A = \#B = 1$, making the expected time at least n just for this reaction.

4.2 Tolerance to imprecise inputs

Despite the fact that removing the assumption of initial leader species makes the model more realistic, it retains an unrealistic aspect of the model. We have assumed the ability to prepare an initial state with precisely specified counts of input molecules. It is certainly equally as difficult to prepare a solution with 999,999 molecules of X, ensuring that the solution does not contain 1,000,000 moleclues, as to ensure that the solution contains 1 molecule of leader L and not 2. However, it is not clear how to properly formalize the question, "What computations can CRNs do when initial states can only be approximately specified?" If we imagined, for instance, being able to prepare counts only to within k bits of precision for some constant k, then at most 2^k different values of a given input could be specified.

Rather than discussing errors of specification and approximate initial counts, there is an alternative way to formalize the idea that with large amounts of molecules, we lose control over individual counts. This is to use the continuous (mass-action) model, in which the amount of a

species is given by a nonnegative real number indicating its average count per unit volume in an infinite volume solution. Even with the ability to specify a precise initial state (vector of real-valued concentrations), without control over the rates of reactions, only continuous piecewise linear functions can be computed [5]. Because these functions are continuous (in fact, uniformly continuous, i.e., rates of change of the output with respect to input are bounded by a constant), a small error in specifying an initial state provably leads only to a small error in reporting the output. Therefore, continuous CRNs are in a sense already robust to imprecise inputs, merely because they can only compute functions that are naturally "error-tolerant".

Contrast this to the case of the discrete model and the semilinear functions they compute, such as the function $f(n) = n$ if n is even and $f(n) = 0$ otherwise. Here, a small change in the input causes an arbitrarily large change in the correct output, and hence an aribtrarily large error in the reported output if the initial state is specified incorrectly. Thus, given the theoretical ability of discrete CRNs to compute functions lacking the natural robustness of continuous functions to small errors in inputs, it remains an open question to formalize how such a CRN might compute such nonrobust functions in a robust way.

Acknowledgement. We are indebted to Anne Condon for helpful discussions and suggestions.

References

[1] Dana Angluin, James Aspnes, Zoë Diamadi, Michael Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18:235–253, 2006. Preliminary version appeared in PODC 2004.

[2] Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *PODC 2006: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 292–299, New York, NY, USA, 2006. ACM Press.

[3] Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008. Preliminary version appeared in DISC 2006.

[4] Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. In *DNA 2012: Proceedings of the 18th International Meeting on DNA Computing and Molecular Programming*, volume 7433 of *Lecture Notes in Computer Science*, pages 25–42. Springer, 2012.

[5] Ho-Lin Chen, David Doty, and David Soloveichik. Rate-independent computation in mass-action chemical reaction networks. In *ITCS 2014: Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, pages 313–326, 2014.

[6] Anne Condon, Alan Hu, Ján Maňuch, and Chris Thachuk. Less haste, less waste: On recycling and its limits in strand displacement systems. *Journal of the Royal Society Interface*, 2:512–521, 2012. Preliminary version appeared in DNA 2011.

[7] Anne Condon, Bonnie Kirkpatrick, and Ján Maňuch. Reachability bounds for chemical reaction networks and strand displacement systems. In *DNA 2012: 18th International Meeting on DNA Computing and Molecular Programming*, volume 7433, pages 43–57. Springer, 2012.

[8] 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*, pages 543–584. Springer Berlin Heidelberg, 2009.

[9] David Doty. Timing in chemical reaction networks. In *SODA 2014: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 772–784, January 2014.

[10] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[11] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[12] Allen Hjelmfelt, Edward D. Weinberger, and John Ross. Chemical implementation of neural networks and Turing machines. *Proceedings of the National Academy of Sciences*, 88(24):10983–10987, 1991.

[13] Hua Jiang, Marc Riedel, and Keshab Parhi. Digital signal processing with molecular reactions. *IEEE Design and Test of Computers*, 29(3):21–31, 2012.

[14] Richard J. Lipton. The reachability problem requires exponential space. Technical report, Yale University, 1976.

[15] Marcelo O. Magnasco. Chemical kinetics is Turing universal. *Physical Review Letters*, 78(6):1190–1193, 1997.

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

[17] David Soloveichik, Georg Seelig, and Erik Winfree. 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] Chris Thachuk and Anne Condon. Space and energy efficient computation with DNA strand displacement systems. In *DNA 2012: Proceedings of the 18th International Meeting on DNA Computing and Molecular Programming*, pages 135–149, 2012.

[19] Gianluigi Zavattaro and Luca Cardelli. Termination problems in chemical kinetics. *CONCUR 2008-Concurrency Theory*, pages 477–491, 2008.