Deterministic Function Computation with Chemical Reaction Networks[*]

Ho-Lin Chen[†] David Doty[‡] David Soloveichik[§]

Abstract

We study the deterministic computation of functions on tuples of natural numbers by chemical reaction networks (CRNs). CRNs have been shown to be efficiently Turing-universal when allowing for a small probability of error. CRNs that are guaranteed to converge on a correct answer, on the other hand, have been shown to decide only the semilinear predicates.

We introduce the notion of function, rather than predicate, computation by representing the output of a function $f : \mathbb{N}^k \to \mathbb{N}^l$ by a count of some molecular species, i.e., if the CRN starts with $n_1, \ldots, n_k$ molecules of some "input" species $X_1, \ldots, X_k$, the CRN is guaranteed to converge to having $f(n_1, \ldots, n_k)$ molecules of the "output" species $Y_1, \ldots, Y_l$. We show that a function $f : \mathbb{N}^k \to \mathbb{N}^l$ is deterministically computed by a CRN if and only if its graph $\{(\mathbf{x}, \mathbf{y}) \in \mathbb{N}^k \times \mathbb{N}^l \mid f(\mathbf{x}) = \mathbf{y}\}$ is a semilinear set.

Finally, we show that each semilinear function f can be computed on input $\mathbf{x}$ in expected time $O(\text{polylog } \|\mathbf{x}\|_1)$.

1 Introduction

The engineering of complex artificial molecular systems will require a sophisticated understanding of how to *program chemistry*. A natural language for describing abstract chemical systems in a well-mixed solution is that of (finite) chemical reaction networks (CRNs), i.e., finite sets of chemical reactions such as $A + B \to A + C$. When the goal is to model the behavior of individual molecules in a well-mixed solution, CRNs are assigned semantics through *stochastic chemical kinetics* [7], in which reactions occur probabilistically with rate proportional to the product of the molecular count of their reactants and inversely proportional to the volume of the reaction vessel.

Traditionally CRNs have been used as a descriptive language to analyze naturally occurring chemical systems. However, recent investigations of CRNs as a programming language for engineering artificial chemical systems have shown CRNs to have surprisingly powerful computational ability. For example, bounded-space Turing machines can be simulated with an arbitrarily small, non-zero probability of error by a CRN with only a polynomial slowdown [1], and even Turing universal computation is possible with an arbitrarily small, non-zero probability of error over all

[*] The first author was supported by the Molecular Programming Project under NSF grant 0832824, the second and third authors were supported by a Computing Innovation Fellowship under NSF grant 1019343. The third author was supported by NIGMS Systems Biology Center grant P50 GM081879.

[†] National Taiwan University, Taipei, Taiwan, holinc@gmail.com

[‡] California Institute of Technology, Pasadena, CA, USA, ddoty@caltech.edu

[§] University of California, San Francisco, San Francisco, CA, USA, david.soloveichik@ucsf.edu

time [11]. This is surprising since finite CRNs necessarily must represent binary data strings in a unary encoding, since they lack positional information to tell the difference between two molecules of the same species. Other work has investigated the power of CRNs to simulate Boolean circuits [9], digital signal processing [8], the (un)decidability of whether a CRN will reach a state where no further reaction is possible [13], and the robustness of CRNs to tolerate multiple copies of the network running in parallel [6]. Finally, recent work proposes concrete chemical implementations of arbitrary CRN programs, particularly using nucleic-acid strand-displacement cascades as the physical reaction primitive [5, 12].

Angluin, Aspnes and Eisenstat [2] investigated the computational power of deterministic CRNs (under a different name, that of the equivalent distributed computing model known as *population protocols*). Some CRNs, when started in an initial configuration assigning nonnegative integer counts to each of k different input species, are guaranteed to converge on a single "yes" or "no" answer, in the sense that there are two special "voting" species L^1 and L^0 so that eventually either L^1 is present and L^0 absent to indicate "yes", or vice versa to indicate "no." The set of inputs $S \subseteq \mathbb{N}^k$ that cause the system to answer "yes" is then a representation of the decision problem solved by the CRN. Angluin, Aspnes and Eisenstat showed that the input sets S decidable by some CRN are precisely the *semilinear* subsets of $\mathbb{N}^k$ (defined formally in Section 2.2).

We extend these prior investigations of decision problem or predicate computation to study deterministic *function* computation. A function $f : \mathbb{N}^k \to \mathbb{N}^l$ is computed by a CRN $\mathcal{C}$ if the following is true. There are "input" species $X_1, \ldots, X_k$ and "output" species $Y_1, \ldots, Y_l$ such that, if $\mathcal{C}$ is initialized with $n_1, \ldots, n_k$ copies of $X_1, \ldots, X_k$, then it is guaranteed to reach a configuration in which the counts of $Y_1, \ldots, Y_l$ are described by the vector $f(n_1, \ldots, n_k)$, and these counts never again change. For example, the CRN $\mathcal{C}$ with the single reaction $X \to 2Y$ computes the function $f(n) = 2n$ in the sense that, if $\mathcal{C}$ starts in an initial configuration with n copies of X and 0 copies of Y, then $\mathcal{C}$ is guaranteed to stabilize to a configuration with $2n$ copies of Y (and no copies of X). Similarly, the function $f(n) = \lfloor n/2 \rfloor$ is computed by the single reaction $2X \to Y$, in that the final configuration is guaranteed to have exactly $\lfloor n/2 \rfloor$ copies of Y (and 0 or 1 copies of X, depending on whether n is even or odd). A function is said to be *semilinear* if its *graph* $\{(\mathbf{x}, \mathbf{y}) \in \mathbb{N}^k \times \mathbb{N}^l \mid f(\mathbf{x}) = \mathbf{y}\}$ is a semilinear set (see Fig. 1 for the graphs of the two functions just mentioned.) We show that the functions deterministically computable by CRNs are precisely the semilinear functions. This implies, for example, that such functions as $f(n_1, n_2) = n_1 n_2$ or $f(n) = n^2$ are not deterministically computable.

Our result employs the predicate computation characterization of Angluin, Aspnes and Eisenstat [2], together with some nontrivial additional technical machinery. In particular, we introduce the notion of "reducing" the computation of one CRN to that of another, essentially using one CRN as a black box in constructing another. This is more difficult than in standard programming languages since there is in general no way of knowing when a CRN is done computing, or whether it will change its answer in the future.

Having established what functions are deterministically computable by CRNs given unbounded time, we turn our attention to the time required for CRNs to converge to the answer. We show that every semilinear function can be deterministically computed on input $\mathbf{x}$ in expected time $\mathrm{polylog}(\|\mathbf{x}\|)$. This is done by a similar technique used by Angluin, Aspnes, and Eisenstat [2] to show the equivalent result for predicate computation. They run a slow deterministic computation in parallel with a fast randomized computation, allowing the deterministic computation to compare the two answers and update the randomized answer only if it is incorrect, which happens with low probability. However, novel techniques are required since it is not as simple to "nondestructively compare" two integers (so that the counts are only changed if they are unequal) as to compare two Boolean values.

2 Preliminaries

Throughout the paper we use both superscripts and subscripts to index variables to make for easier reading; the superscript never means exponentiation. Apologies in advance.

Given a vector $\mathbf{x} \in \mathbb{Z}^k$, let $\|\mathbf{x}\| = \|\mathbf{x}\|_1 = \sum_{i=1}^{k} |\mathbf{x}_i|$, where $\mathbf{x}_i$ denotes the ith coordinate of $\mathbf{x}$. If $f : \mathbb{Z}^k \to \mathbb{Z}^l$ is a function, define the *graph* of f to be the set $\left\{ \; (\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^k \times \mathbb{Z}^l \; \middle| \; f(\mathbf{x}) = \mathbf{y} \; \right\}$. We say a partial function $f : \mathbb{Z}^k \dashrightarrow \mathbb{Z}^l$ is *affine* if there exist kl rational numbers $a_1^1, \ldots, a_k^l \in \mathbb{Q} \cap [0, \infty)$ and $l + k$ integers $b_1, \ldots, b_l, c_1, \ldots, c_k \in \mathbb{Z}$ such that for each $1 \leq j \leq l$, $\mathbf{y}_j = b_j + \sum_{i=1}^{k} a_i^j (\mathbf{x}_i + c_i)$. In other words, the graph of f, when projected onto the $(k+1)$-dimensional space defined by the k coordinates corresponding to $\mathbf{x}$ and the single coordinate corresponding to $\mathbf{y}_j$, is a subset of a k-dimensional hyperplane.

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 $C \in \mathbb{N}^\Lambda$ as a vector of $|\Lambda|$ nonnegative integers, with each coordinate "labeled" by an element of Λ. Given $X \in \Lambda$ and $C \in \mathbb{N}^\Lambda$, we refer to $C(X)$ as the *count of X in C*. We write $C \leq C'$ to denote that $C(X) \leq C'(X)$ for all $X \in \Lambda$. Given $C, C' \in \mathbb{N}^\Lambda$, we define the vector component-wise operations of addition $C + C'$, subtraction $C - C'$, and scalar multiplication nC for $n \in \mathbb{N}$. If $\Delta \subset \Lambda$, we view a vector $C \in \mathbb{N}^\Delta$ equivalently as a vector $C \in \mathbb{N}^\Lambda$ by assuming $C(X) = 0$ 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 $N = (\Lambda, R)$, where Λ is a finite set of chemical *species*, and R is a finite set of reactions over Λ. A *configuration* of a CRN $N = (\Lambda, R)$ is a vector $C \in \mathbb{N}^\Lambda$. We also write $\#_C X$ to denote $C(X)$, the *count* of species X in configuration C, or simply $\#X$ when C is clear from context.

Given a configuration C and reaction $\alpha = \langle \mathbf{r}, \mathbf{p} \rangle$, we say that α is *applicable* to C if $\mathbf{r} \leq C$ (i.e., C contains enough of each of the reactants for the reaction to occur). If α is applicable to C, then write $\alpha(C)$ to denote the configuration $C + \mathbf{p} - \mathbf{r}$ (i.e., the configuration that results from applying reaction α to C). If $C' = \alpha(C)$ for some reaction $\alpha \in R$, we write $C \to_N C'$, or merely $C \to C'$ when N is clear from context. An *execution* (a.k.a., *execution sequence*) $\mathcal{E}$ is a finite or infinite sequence of one or more configurations $\mathcal{E} = (C_0, C_1, C_2, \ldots)$ such that, for all $i \in \{1, \ldots, |\mathcal{E}| - 1\}$, $C_{i-1} \to C_i$. If a finite execution sequence starts with C and ends with C', we write $C \to_N^* C'$, or merely $C \to^* C'$ when the CRN N is clear from context. In this case, we say that C' is *reachable* from C.

Let $\Delta \subseteq \Lambda$. We say that $P \in \mathbb{N}^\Delta$ is a *partial configuration (with respect to Δ)*. We write $P = C \upharpoonright \Delta$ for any configuration C such that $C(X) = P(X)$ for all $X \in \Delta$, and we say that P is the *restriction of C to Δ*. Say that a partial configuration P with respect to Δ is *reachable* from configuration C' if there is a configuration C reachable from C' and $P = C \upharpoonright \Delta$. In this case, we write $C' \to^* P$. An infinite execution $\mathcal{E} = (C_0, C_1, C_2, \ldots)$ is *fair* if, for all partial configurations P, if P is infinitely often reachable then it is infinitely often reached.[1] In other words, no reachable partial configuration is "starved". This definition of fairness is stricter than that used by Angluin, Aspnes, and Eisenstat [2], which used only full configurations rather than partial configurations. We choose this definition to prevent intuitively unfair executions from vacuously satisfying the definition of "fair" simply because of some species whose count is monotonically increasing with time (preventing any configuration from being infinitely often reachable).

Note that the definition given above, applied to finite executions, deems all of them fair vacuously. We wish to distinguish between finite executions that can be extended by applying another reaction and those that cannot. Say that a configuration is *terminal* if no reaction is applicable to it. We say that a finite execution is *fair* if and only if it ends in a terminal configuration.

2.2 Stable decidability of predicates

We now review the definition of stable decidability of predicates introduced by Angluin, Aspnes, and Eisenstat [2]. Those authors use the term "stably *compute*", but we reserve the term "compute" to apply to the computation of non-Boolean functions. Intuitively, some species "vote" for a yes/no answer, and a CRN N is a stable decider if, for all initial configurations, N is guaranteed (under fair executions) to reach a consensus vote, which is potentially different for different initial configurations but consistent over all fair executions starting from a fixed initial configuration.

A *chemical reaction decider* (CRD) is a tuple $\mathcal{D} = (\Lambda, R, \Sigma, \Upsilon, \phi, \sigma)$, where (Λ, R) is a CRN, $\Sigma \subseteq \Lambda$ is the *set of input species*, $\Upsilon \subseteq \Lambda$ is the set of *voters*, $\phi : \Upsilon \to \{0,1\}$ is the *(Boolean) output function*, and $\sigma \in \mathbb{N}^{\Lambda \setminus \Sigma}$ is the *initial context*. Intuitively, the goal is for the CRD to get all voters to be eventually unanimous and correct (and for at least one to be present). An input to $\mathcal{D}$ will be a vector $I_0 \in \mathbb{N}^\Sigma$. Thus a CRD together with an input vector defines an initial configuration I defined by $I(X) = I_0(X)$ if $X \in \Sigma$, and $I(X) = \sigma(X)$ otherwise. We say that such a configuration is a *valid initial configuration*, i.e., $I \upharpoonright (\Lambda \setminus \Sigma) = \sigma$. If we are discussing a CRN understood from context to have a certain initial configuration I, we write $\#_0 X$ to denote $I(X)$.

We extend ϕ to a partial function $\Phi : \mathbb{N}^\Lambda \dashrightarrow \{0,1\}$ as follows. $\Phi(C)$ is undefined if either $C(X) = 0$ for all $X \in \Upsilon$, or if there exist $X_0, X_1 \in \Upsilon$ such that $C(X_0) > 0$, $C(X_1) > 0$, $\phi(X_0) = 0$ and $\phi(X_1) = 1$. Otherwise, there exists $b \in \{0,1\}$ such that $(\forall X \in \Upsilon)(C(X) > 0 \implies \phi(X) = b)$; in this case, the *output* $\Phi(C)$ of configuration C is b.

A configuration C is *output stable* if $\Phi(C)$ is defined and, for all C' such that $C \to^* C'$, $\Phi(C') = \Phi(C)$. We say that a CRD $\mathcal{D}$ is *stable* if, for any valid initial configuration $I \in \mathbb{N}^\Lambda$, there exists $b \in \{0,1\}$ such that *every* fair execution $\mathcal{E} = (I, C_1, C_2, \ldots)$ contains an output stable configuration C with $\Phi(C) = b$ (i.e., $\mathcal{D}$ always converges to a defined output on input I, and this output is the same for any fair execution starting from I). If $\mathcal{D}$ is stable, then some unique subset $S_0 \subseteq \mathbb{N}^\Sigma$ of all possible initial configurations always converges to output 0 and stays with that output, and the remainder $S_1 = \mathbb{N}^\Sigma \setminus S_0$ always converges to output 1 and stays with that output. We say that $\mathcal{D}$ *stably decides* the set S_1, or that $\mathcal{D}$ *stably decides* the predicate $\psi : \mathbb{N}^\Sigma \to \{0,1\}$ defined by $\psi(I_0) = 1$ if $I_0 \in S_1$ and $\psi(I_0) = 0$ if $I_0 \in S_0$.

[1] i.e. $(\forall \Delta \subseteq \Lambda)(\forall P \in \mathbb{N}^\Delta)[((\exists^\infty i \in \mathbb{N})\ C_i \to^* P) \implies ((\exists^\infty j \in \mathbb{N})\ P = C_j \upharpoonright \Delta)]$.

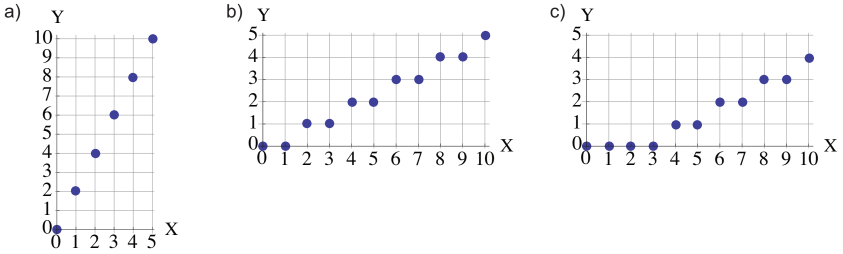

Figure 1: Examples of deterministically computable functions. (a) The graph of the function $f(x) = 2x$ is a semilinear set: $\{\ n_1 \cdot (1,2) \mid n_1 \in \mathbb{N}\ \}$. This function is deterministically computed by the CRC $(\Lambda, R, \Sigma, \Gamma, \sigma)$ where (Λ, R) is the CRN consisting of a single reaction $X \to 2Y$, $\Sigma = \{X\}$, $\Gamma = \{Y\}$, and $\sigma = \{\}$. (b) The graph of the function $f(x) = \lfloor x/2 \rfloor$ is a semilinear set: $\{\ n_1 \cdot (2,1) \mid n_1 \in \mathbb{N}\ \} \cup \{\ (1,0) + n_1 \cdot (2,1) \mid n_1 \in \mathbb{N}\ \}$. This function is deterministically computed by the CRC $(\Lambda, R, \Sigma, \Gamma, \sigma)$ where (Λ, R) is the CRN consisting of a single reaction $2X \to Y$, $\Sigma = \{X\}$, $\Gamma = \{Y\}$, and $\sigma = \{\}$. (c) The graph of the function $f(x) = \max(0, \lfloor x/2 \rfloor - 1)$ is a semilinear set: $\{\ (2,0) + n_1 \cdot (2,1) \mid n_1 \in \mathbb{N}\ \} \cup \{\ (3,0) + n_1 \cdot (2,1) \mid n_1 \in \mathbb{N}\ \}$. This function is deterministically computed by the CRC $(\Lambda, R, \Sigma, \Gamma, \sigma)$ where (Λ, R) is the CRN consisting of reactions $\{2X \to Y,\ K + Y \to \varnothing\}$, $\Sigma = \{X\}$, $\Gamma = \{Y\}$, and $\sigma = \{1K\}$.

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

$$A = \left\{\ \mathbf{b} + n_1\mathbf{u}^1 + \ldots + n_p\mathbf{u}^p\ \middle|\ n_1, \ldots, n_p \in \mathbb{N}\ \right\}.$$

A is *semilinear* if it is a finite union of linear sets.

The following theorem is due to Angluin, Aspenes, 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 definitions of [2] assume that $\Upsilon = \Lambda$ (i.e., every species votes). However, it is not hard to show that we may assume there are only two voting species, L^0 and L^1, so that $\#L^0 > 0$ and $\#L^1 = 0$ means that the CRD is answering "no", and $\#L^0 = 0$ and $\#L^1 > 0$ means that the CRD is answering "yes." This convention will be more convenient in this paper.

2.3 Stable computation of functions

We now define a notion of stable computation of *functions* similar to those above for predicates.[2] Intuitively, the inputs to the function are the initial counts of inputs species $X_1, \ldots, X_k$, and the outputs are the counts of "output" species $Y_1, \ldots, Y_l$, such that the CRN is guaranteed to eventually reach a configuration in which the counts of the output species have the correct values and never change from that point on.

We now formally define what it means for a CRN to stably compute a function. Let $k, l \in \mathbb{Z}^+$. A *chemical reaction computer (CRC)* is a tuple $\mathcal{C} = (\Lambda, R, \Sigma, \Gamma, \sigma)$, 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 $\Sigma \cap \Gamma = \varnothing$, $|\Sigma| = k$, $|\Gamma| = l$, and $\sigma \in \mathbb{N}^{\Lambda \setminus \Sigma}$ is the *initial context.* Write $\Sigma = \{X_1, X_2, \ldots, X_k\}$ and $\Gamma = \{Y_1, Y_2, \ldots, Y_l\}$. We say that a configuration C is *output count stable* if, for every C' such that $C \to^* C'$ and every $X \in \Gamma$, $C(X) = C'(X)$ (i.e., the counts of species in Γ will never change if C is reached). As with CRD's, we require initial configurations I of $\mathcal{D}$ with input $I_0 \in \mathbb{N}^\Sigma$ to

[2] The extension from Boolean predicates to functions described by Aspnes and Ruppert [4] applies only to finite-range functions, where one can choose $|\Lambda| \geq |Y|$ for output range Y.

obey $I(X) = I_0(X)$ if $X \in \Sigma$ and $I(X) = \sigma(X)$ otherwise, calling them *valid initial configurations*. We say that $\mathcal{N}$ *stably computes* $f : \mathbb{N}^k \to \mathbb{N}^l$ if, for every valid initial configuration $I \in \mathbb{N}^\Lambda$, every fair execution $\mathcal{E} = (I, C_1, C_2, \ldots)$ contains an output count stable configuration C such that $f(I(X_1), I(X_2), \ldots, I(X_k)) = (C(Y_1), C(Y_2), \ldots, C(Y_l))$. In other words, the counts of species in Γ are guaranteed to converge to the value of $f(n_1, n_2, \ldots, n_k)$ when started in an initial configuration with n_i copies of X_i for each $i \in \{1, \ldots, k\}$. We say that such a CRC is *count stable*. For any species $A \in \Lambda$, we write $\#_\infty A$ to denote the eventual convergent count of A if $\#A$ is guaranteed to stabilize; otherwise, $\#_\infty A$ is undefined.

Fig. 1 shows example CRCs for $f(x) = 2x$, $f(x) = \lfloor x/2 \rfloor$, and $f(x) = \max(0, \lfloor x/2 \rfloor - 1)$. In sections 3 and 4 we will describe systematic, but much more complex constructions for these and all functions with semilinear graphs.

2.4 Time complexity model

Since all rate constants in this paper are 1, we define time assuming this to be true. 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.

Given a fixed volume v and current configuration C, the *propensity* of a unimolecular reaction $\alpha : X \to \ldots$ in configuration C is $\rho(C, \alpha) = \#_C X$. The propensity of a bimolecular reaction $\alpha : X + Y \to \ldots$, where $X \neq Y$, is $\rho(C, \alpha) = \frac{\#_C X \#_C Y}{v}$. The propensity of a bimolecular reaction $\alpha : X + X \to \ldots$ is $\rho(C, \alpha) = \frac{1}{2} \frac{\#_C X(\#_C X - 1)}{v}$. The propensity function determines the kinetics of the system as follows. The time until the next reaction occurs is an exponential random variable with rate $\rho(C) = \sum_{\alpha \in R} \rho(C, \alpha)$ (note that $\rho(C) = 0$ if no reactions are applicable to C). The probability that next reaction will be a particular α_{next} is $\frac{\rho(C, \alpha_{\text{next}})}{\rho(C)}$.

The kinetic model is based on the physical assumption of well-mixedness valid in a dilute solution. 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 [11].

3 Deterministic function computation

In this section we use Theorem 2.1 to show that only "simple" functions can be stably computed by CRCs. This is done by showing how to reduce the computation of a function by a CRC to the decidability of its graph by a CRD, and vice versa. In this section we do not concern ourselves with time complexity, so the volume is left unspecified.

A function is said to be *definable* in a logical theory if its graph is a set definable in that theory. In particular, since the semilinear sets are precisely those definable in Presburger arithmetic [10], a function is definable in Presburger arithmetic if and only if its graph is semilinear. We call such a function a *semilinear function*.

The next lemma shows that every function computable by a chemical reaction network is semilinear.

Lemma 3.1. *Every function stably computable by a CRC is semilinear.*

Proof. Let $\mathcal{C} = (\Lambda, R, \Sigma, \Gamma, \sigma)$ be the CRC that stably computes $f : \mathbb{N}^k \to \mathbb{N}^l$, with input species $\Sigma = \{X_1, \ldots, X_k\}$ and output species $\Gamma = \{Y_1, \ldots, Y_l\}$. Modify $\mathcal{C}$ to obtain the following CRD

$\mathcal{D} = (\Lambda', R', \Sigma', \Upsilon', \phi', \sigma')$. Let $\mathcal{Y}^C = \{Y_1^C, \ldots, Y_l^C\}$, where each $Y_i^C \notin \Lambda$ are new species. Let $\mathcal{Y}^P = \{Y_1^P, \ldots, Y_l^P\}$, where each $Y_i^P \notin \Lambda$ are new species. Intuitively, $\#Y_i^P$ represents the number of Y_i's produced by $\mathcal{C}$ and $\#Y_i^C$ the number of Y_i's consumed by $\mathcal{C}$. The goal is for C' to stably decide the predicate $f(\#_0 X_1, \ldots, \#_0 X_k) = (\#_0 Y_1^C, \ldots, \#_0 Y_l^C)$. In other words, the initial configuration of C' will be the same as that of C except for some copies of Y_i^C, equal to the purported output of f to be tested by $\mathcal{D}$. Since every predicate stably decidable by a CRD is semilinear (Theorem 2.1), this will prove the lemma.

Let $\Lambda' = \Lambda \cup \mathcal{Y}^C \cup \mathcal{Y}^P \cup \{L^0, L^1\}$. Let $\Sigma' = \Sigma \cup \mathcal{Y}^C$. Let $\Upsilon' = \{L^0, L^1\}$, with $\phi(L^0) = 0$ and $\phi(L^1) = 1$. Let $\sigma'(S) = 0$ for all $S \in \Lambda' \setminus \Sigma'$. Modify R by adding reactions to obtain R' as follows. For each reaction α that consumes a net number n of Y_i molecules, append n products Y_i^C to α. For each reaction α that produces a net number n of Y_i molecules, append n products Y_i^P to α. For example, the reaction $A + 2B + Y_1 + 3Y_3 \to Z + 3Y_1 + 2Y_3$ becomes $A + 2B + Y_1 + 3Y_3 \to Z + 3Y_1 + 2Y_3 + 2Y_1^P + Y_3^C$. Since $\mathcal{C}$ is count-stable, eventually no reactions producing or consuming net copies of Y_i are possible, whence $\mathcal{D}$ as defined so far is count-stable with respect to Y_i^P and Y_i^C as well.

Then add the following additional reactions to R', for each $i \in \{1, \ldots, l\}$,

$$\begin{aligned}
Y_i^P + Y_i^C &\to L^1 && (3.1)\\
Y_i^P + L^1 &\to Y_i^P + L^0 && (3.2)\\
Y_i^C + L^1 &\to Y_i^C + L^0 && (3.3)\\
L^0 + L^1 &\to L^1 && (3.4)
\end{aligned}$$

In the following, we use $\#^{\uparrow}_{\infty} Y_i^P$ to denote the total number of Y_i^P ever produced and $\#^{\uparrow}_{\infty} Y_i^C$ to denote $\#_0 Y_i^C$ plus the total number of Y_i^C's ever produced. Note that, if and only if $f(\#_0 X_1, \ldots, \#_0 X_k) = (\#_0 Y_1^C, \ldots, \#_0 Y_l^C)$, then eventually, for each i, $\#Y_i^P$ and $\#Y_i^C$ stabilize to equal values in the absence of reaction (3.1); in other words, if and only if $\#^{\uparrow}_{\infty} Y_i^P = \#^{\uparrow}_{\infty} Y_i^C$.

Since Y_i^P and Y_i^C are possibly produced but not consumed by reactions other than (3.1), we may think of reaction (3.1) as if it does not occur until $\#Y_i^P$ and $\#Y_i^C$ have stabilized, even though reaction (3.1) may consume some copies of Y_i^P and Y_i^C before all eventual copies have been produced.

Reactions (3.1)-(3.4) ensure that if $\#^{\uparrow}_{\infty} Y_i^P = \#^{\uparrow}_{\infty} Y_i^C$ for all $i \in \{1, \ldots, l\}$, then $\#_\infty L^1 > 0$ and $\#_\infty L^0 = 0$, and if $\#^{\uparrow}_{\infty} Y_i^P \neq \#^{\uparrow}_{\infty} Y_i^C$ for some $i \in \{1, \ldots, l\}$, then $\#_\infty L^1 = 0$ and $\#_\infty L^0 > 0$. To show that this holds, we have two cases for each $i \in \{1, \ldots, l\}$. In the following, we write $f(\#X_1, \ldots, \#X_k)_i$ to denote the value $\#Y_i$ if $f(\#X_1, \ldots, \#X_k) = (\#Y_1, \ldots, \#Y_l)$.

1. $\underline{f(\#_0 X_1, \ldots, \#_0 X_k)_i = \#_0 Y_i^C \text{ for all } i \in \{1, \ldots, l\}}$: Then $\#^{\uparrow}_{\infty} Y_i^P = \#^{\uparrow}_{\infty} Y_i^C$ for all $i \in \{1, \ldots, l\}$, so eventually every Y_i^P and Y_i^C disappears through reaction (3.1). At this point there are some number of L^0's and L^1's remaining. The number of L^1's must be positive since the final execution of reaction (3.1) created a copy of L^1. Since none of reactions (3.1)-(3.3) are possible, $\#L^1$ stays positive forever. After this time, reaction (3.4) eventually removes all copies of L^0.

2. $\underline{f(\#_0 X_1, \ldots, \#_0 X_k)_i \neq \#_0 Y_i^C \text{ for some } i \in \{1, \ldots, l\}}$: Then $\#^{\uparrow}_{\infty} Y_i^P \neq \#^{\uparrow}_{\infty} Y_i^C$ for some $i \in \{1, \ldots, l\}$, so reaction (3.1) ensures that eventually either 1) $\#_\infty Y_i^C = 0$ and $\#_\infty Y_i^P > 0$, or 2) $\#_\infty Y_i^C > 0$ and $\#_\infty Y_i^P = 0$. Eventually reaction (3.1) is not possible for any $j \in \{1, \ldots, l\}$

because either such that $\#_\infty Y_j^P = 0$ or $\#_\infty Y_j^C = 0$, and at that point, no more copies of L^1 are produced. From then on, reaction (3.2) (in case (1)) or reaction (3.3) (in case (2)) ensures that eventually all copies of L^1 are converted to L^0. Reaction (3.4) may convert some copies of L^0 back to L^1 before this happens, but this strictly decreases the quantity $(\#L^1 + \#L^0)$. If this quantity reaches 1 then reaction (3.4) is no longer possible. Thus eventually all existing copies of L^1 are converted to L^0 and reaction (3.4) is no longer possible.

Thus, if $f(\#_0 X_1, \ldots, \#_0 X_k) = (\#_0 Y_1^C, \ldots, \#_0 Y_l^C)$, then $\#_\infty L^1 > 0$ and $\#_\infty L^0 = 0$, and otherwise, $\#_\infty L^1 = 0$ and $\#_\infty L^0 > 0$, showing that $\mathcal{D}$ stably decides the graph of f. □

The next lemma shows the converse of Lemma 3.1.

Lemma 3.2. *Every semilinear function is stably computable by a CRC.*

Proof. Let $f : \mathbb{N}^k \to \mathbb{N}^l$ be a semilinear function, and let

$$F = \left\{ \ (\mathbf{x}, \mathbf{y}) \in \mathbb{N}^k \times \mathbb{N}^l \ \middle| \ f(\mathbf{x}) = \mathbf{y} \ \right\}$$

denote the graph of f. We then consider the set

$$\widehat{F} = \left\{ \ (\mathbf{x}, \mathbf{y}^P, \mathbf{y}^C) \in \mathbb{N}^k \times \mathbb{N}^l \times \mathbb{N}^l \ \middle| \ f(\mathbf{x}) = \mathbf{y}^P - \mathbf{y}^C \ \right\}.$$

Intuitively, $\widehat{F}$ defines the same function as F, but with each output variable expressed as the difference between two other variables. Note that $\widehat{F}$ is not the graph of a function since for each $\mathbf{y} \in \mathbb{N}^l$ there are an infinite number of pairs $(\mathbf{y}^P, \mathbf{y}^C)$ such that $\mathbf{y}^P - \mathbf{y}^C = \mathbf{y}$. However, we only care that $\widehat{F}$ is a semilinear set so long as F is a semilinear set.

Then by Theorem 2.1, $\widehat{F}$ is stably decidable by a CRD $\mathcal{D} = (\Lambda, R, \Sigma, \Upsilon, \phi, \sigma)$, where $\Sigma = \{X_1, \ldots, X_k, Y_1^P, \ldots, Y_l^P, Y_1^C, \ldots, Y_l^C\}$, and we assume that Υ contains only species L^1 and L^0 such that for any output-stable configuration of $\mathcal{D}$, exactly one of $\#L^1$ or $\#L^0$ is positive to indicate a yes or no answer, respectively.

Define the CRC $\mathcal{C} = (\Lambda', R', \Sigma', \Gamma', \sigma')$ as follows. Let $\Sigma' = \{X_1, \ldots, X_k\}$. Let $\Gamma' = \{Y_1, \ldots, Y_l\}$. Let $\Lambda' = \Lambda \cup \Gamma'$. Let $\sigma'(S) = 0$ for all $S \in \Lambda \setminus (\Sigma \cup \{L^0\})$, and let $\sigma'(L^0) = 1$. Intuitively, we will have L^0 change the value of $\mathbf{y}$ (by producing either Y_i^P or Y_i^C molecules), since L^0's presence indicates that $\mathcal{D}$ has not yet decided that the predicate is satisfied. It essentially searches for new values of $\mathbf{y}$ that do satisfy the predicate. This indirect way of representing the value $\mathbf{y}$ is useful because $\mathbf{y}^P$ and $\mathbf{y}^C$ can both be increased monotonically to change $\mathbf{y}$ in either direction. If we wanted to test a lower value of $\mathbf{y}_i$, then this would require consuming a copy of Y_i, but this may not be possible if $\mathcal{D}$ has already consumed all of them.

Let R' be R plus the following reactions for each $1 \leq i \leq l$:

$$L^0 \to L^0 + Y_i^P + Y_i \tag{3.5}$$

$$L^0 + Y_i \to L^0 + Y_i^C \tag{3.6}$$

It is clear that reactions (3.5) and (3.6) enforce that at any time, $\#Y_i$ is equal to the total number of Y_i^P's produced by reaction (3.5) minus the total number of Y_i^C's produced by reaction (3.6) (although some of each of Y_i^P or Y_i^C may have been produced or consumed by other reactions in R).

Suppose that $f(\mathbf{x}) \neq (\#Y_1, \ldots, \#Y_l)$. Then if there are no L^0 molecules present, the counts of Y_i^P and Y_i^C are not changed by reactions (3.5) and (3.6). Therefore only reactions in R proceed, and by the correctness of $\mathcal{D}$, eventually an L^0 molecule is produced (since eventually $\mathcal{D}$ must reach an output-stable configuration answering "no", although L^0 may appear before $\mathcal{D}$ reaches an output-stable configuration, if some L^1 are still present). Once L^0 is present, by the fairness condition (choosing $\Delta = \{Y_1, \ldots. Y_l\}$), eventually the value of $(\#Y_1, \ldots, \#Y_l)$ will change by reaction (3.5) or (3.6). In fact, *every* value of $(\#Y_1, \ldots, \#Y_l)$ is possible to explore by the fairness condition.

Suppose then that $f(\mathbf{x}) = (\#Y_1, \ldots, \#Y_l)$. Perhaps L^0 is present because the reactions in R have not yet reached an output-stable "yes" configuration. Then perhaps the value of $(\#Y_1, \ldots, \#Y_l)$ will change so that $f(\mathbf{x}) \neq (\#Y_1, \ldots, \#Y_l)$. But by the fairness condition, a correct value of $(\#Y_1, \ldots, \#Y_l)$ must be present infinitely many times, so again by the fairness condition, since from such a configuration it is possible to eliminate all L^0 molecules before producing Y_i^P or Y_i^C molecules, this must eventually happen. When all L^0 molecules are gone while $f(\mathbf{x}) = (\#Y_1, \ldots, \#Y_l)$, it is no longer possible to change the value of $(\#Y_1, \ldots, \#Y_l)$, whence $\mathcal{C}$ has reached a count-stable configuration with the correct answer. Therefore $\mathcal{C}$ stably computes f. □

Lemmas 3.1 and 3.2 immediately imply the following theorem.

Theorem 3.3. *A function $f : \mathbb{N}^k \to \mathbb{N}^l$ is stably computable by a CRC if and only if it is semilinear.*

One unsatisfactory aspect of Lemma 3.2 is that we "peek inside the black box" of $\mathcal{D}$ by using the fact that we know it is deciding a semilinear predicate. Lemma 3.1, on the other hand, uses only the fact that $\mathcal{C}$ is computing some function. Although we know that $\mathcal{C}$, being a chemical reaction computer, is only capable of computing semilinear functions, if we imagine that some external powerful "oracle" controlled the reactions of $\mathcal{C}$ to allow it to stably compute a non-semilinear function, then $\mathcal{D}$ would decide that function's graph. Thus Lemma 3.1 is more like the black-box oracle Turing machine reductions employed in computability and complexity theory, which work no matter what mythical device is hypothesized to be responsible for answering the oracle queries.

4 Speed of deterministic function computation

Theorem 3.3 shows that precisely the semilinear functions can be computed without error by a CRC. However, like the predicate-deciding CRDs of Angluin, Aspnes, and Eisenstat [2], the speed is slow, linear-time (in the number of molecules) in the case of one direction and exponential-time in the other direction. Soloveichik, Cook, Winfree, and Bruck [11], and independently, Angluin, Aspnes, and Eisenstat [1] (for bounded-space computation) showed that if a small probability of error is allowed, then arbitrary Turing machines can be simulated with only a polynomial slowdown. The latter authors combined the results of [1] and [2] to show that semilinear functions can be computed without error in expected polylogarithmic time. In this section we show that a similar technique implies that semilinear functions can be computed by CRNs without error in expected polylogarithmic time.

Throughout this section, we use the technique of "running multiple CRNs in parallel" on the same input. To accomplish this it is necessary to split the inputs $X_1, \ldots, X_k$ into separate molecules using a reaction $X_i \to X_i^1 + X_i^2 + \ldots + X_i^p$, which will add only $O(\log n)$ to the time complexity, so that each of the p separate parallel CRNs do not interfere with one another. For brevity we omit stating this formally when the technique is used.

Theorem 4.1. *Let $f : \mathbb{N}^k \to \mathbb{N}^l$ be semilinear. Then there is a CRC $\mathcal{C}$ that stably computes f, and the expected time for $\mathcal{C}$ to reach a count-stable configuration on input $\mathbf{x}$ is $O(\text{polylog } \|\mathbf{x}\|)$.*

Proof. (proof sketch) Our CRC will use the counts of Y_j for each output dimension $\mathbf{y}_j$ as the global output, and begins by running in parallel:

1. A fast, error-prone CRC $\mathcal{F}$ for $\mathbf{y}, \mathbf{b}, \mathbf{c} = f(\mathbf{x})$. It is constructed based on [1]. By [1], for any constant $c > 0$, we may design $\mathcal{F}$ so that it is correct and finishes in time $O(\log^5 n)$ with probability at least $1 - n^{-c}$. We modify it so that upon halting, it copies an "internal" output species $\widehat{Y}_j$ to Y_j (the global output), B_j, and C_j through reactions $H + \widehat{Y}_j \to Y_j + B_j + C_j$ (in asymptotically negligible time). Here, H is some molecule that is guaranteed with high probability not to be present until $\mathcal{F}$ has halted, and to be present in large ($\Omega(n)$) count so that the conversion is fast. In this way we are guaranteed that the amount of Y_j produced by $\mathcal{C}$ is the same as the amounts of B_j and C_j no matter whether its computation is correct or not.

2. A slow, deterministic CRC $\mathcal{S}$ for $\mathbf{y}' = f(\mathbf{x})$. It is constructed as in Lemma 4.2, running in expected $O(n \log n)$ time.

3. A slow, deterministic CRD $\mathcal{D}$ for the semilinear predicate "$\mathbf{b} = f(\mathbf{x})$?". It is constructed as in [3] and runs in expected $O(n \log n)$ time.

Following Angluin, Aspnes, and Eisenstat [1], we construct a "timed trigger" as follows, using a leader molecule, a marker molecule, and $n = O(\|\mathbf{x}\|)$ interfering molecules. The leader fires the trigger if it encounters the marker molecule d times without any intervening reactions with the interfering molecules. This happens rarely enough that with high probability the trigger fires after $\mathcal{F}$ and $\mathcal{D}$ finishes (time analysis is presented below). When the trigger fires, it checks if $\mathcal{D}$ is outputting a "no" (e.g. has a molecule of L_0), and if so, produces a molecule of P_{fix}. This indicates that the output of the fast CRC $\mathcal{F}$ is not to be trusted, and the system should switch from the possible erroneous result of $\mathcal{F}$ to the sure-to-be correct result of $\mathcal{S}$.

Once a P_{fix} is produced, the system converts the output molecules Y'_j of the slow, deterministic CRC $\mathcal{S}$ to the global output Y_j, and kills enough of the global output molecules to remove the ones produced by the fast, error-prone CRC:

$$
\begin{aligned}
P_{\text{fix}} + Y'_j &\to P_{\text{fix}} + Y_j && (4.1)\\
P_{\text{fix}} + C_j &\to P_{\text{fix}} + \overline{Y}_j && (4.2)\\
Y_j + \overline{Y}_j &\to \varnothing. && (4.3)
\end{aligned}
$$

Finally, P_{fix} triggers a process consuming "essential components" of $\mathcal{F}$ in expected $O(\log n)$ time so that afterward, $\mathcal{F}$ cannot produce any output molecules. While this step is not required for correctness, it is necessary for the time analysis in order to ensure that $\mathcal{F}$ does not take too long to output (if $\mathcal{F}$ fails it could produce its output even after $\mathcal{S}$).

First, observe that the output will always eventually converge to the right answer, no matter what happens: If P_{fix} is eventually produced, then the output will eventually be exactly that given by $\mathcal{S}$ which is guaranteed to converge correctly. If P_{fix} is never produced, then the fast, error-prone CRC must produce the correct amount of Y_j — otherwise, $\mathcal{D}$ will detect a problem.

For the expected time analysis, let us first analyze the trigger. The probability that the trigger leader will fire on any particular reaction number is at most n^{-d}. In time n^2, the expected number of leader reactions is $O(n^2)$. Thus, the expected number of firings of the trigger in n^2 time is n^{-d+2}. This implies that the probability that the trigger fires before n^2 time is at most n^{-d+2}. The expected time for the trigger to fire is $O(n^d)$.

We now consider the contribution to the total expected time from 3 cases:

1. $\mathcal{F}$ is correct, and the trigger fires after time n^2. There are two subcases: (a) $\mathcal{F}$ finishes before the trigger fires. Conditional on this, the whole system converges to the correct answer, never to change it again, in expected time $O(\log^5 n)$. This subcase contributes at most $O(\log^5 n)$ to the total expected time. (b) $\mathcal{F}$ finishes after the trigger fires. In this case, we may produce a P_{fix} molecule and have to rely on the slow CRC $\mathcal{S}$. The probability of this case happening is at most n^{-c}. Conditional on this case, the expected time for the trigger to fire is still $O(n^d)$. The whole system converges to the correct answer in expected time $O(n^d)$, because everything else is asymptotically negligible. Thus the contribution of this subcase to the total expectation is at most $O(n^{-c} \cdot n^d) = O(n^{-c+d})$.

2. $\mathcal{F}$ is correct, but the trigger fires before n^2 time. In this case, we may produce a P_{fix} molecule and have to rely on the slow CRC $\mathcal{S}$ for the output. The probability of this case occurring is at most n^{-d+2}. Conditional on this case occurring, the expected time for the whole system to converge to the correct answer can be bounded by $O(n^2)$. Thus the contribution of this subcase to the total expectation is at most $O(n^{-d+2} \cdot n^2) = O(n^{-d+4})$.

3. $\mathcal{F}$ fails. In this case we'll have to rely on the slow CRC $\mathcal{S}$ for the output again. Since this occurs with probability at most n^{-c}, and the conditional expected time for the whole system to converge to the correct answer can be bounded by $O(n^d)$ again, the contribution of this subcase to the total expectation is at most $O(n^{-c} \cdot n^d) = O(n^{-c+d})$.

So the total expected time is bounded by $O(\log^5 n) + O(n^{-c+d}) + O(n^{-d+4}) + O(n^{-c+d}) = O(\log^5 n)$ for $d > 4, c > d$. □

Lemma 4.2. *Let $f : \mathbb{N}^k \to \mathbb{N}^l$ be semilinear. Then there is a CRC $\mathcal{C}$ that stably computes f, and the expected time for $\mathcal{C}$ to reach a count-stable configuration on input $\mathbf{x}$ is $O(\|\mathbf{x}\| \log \|\mathbf{x}\|)$ (where the $O()$ constant depends on f).*

Proof. By Lemma 3.2, there is a CRC $\mathcal{C}_s$ that stably computes f. However, that CRC is too slow to use in this proof. We provide an alternative proof that every semilinear function can be computed by a CRC in expected time $O(\|\mathbf{x}\| \log \|\mathbf{x}\|)$.

By Lemma 4.3, 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 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 dom f_i is a linear (and therefore semilinear) set, we compute each predicate $\phi_i =$ "$\mathbf{x} \in \text{dom } f_i$ and $(\forall 1 \leq i' < i)\ \mathbf{x} \notin \text{dom } f_{i'}$?" by separate parallel CRD's. (The latter condition ensures that for each $\mathbf{x}$, precisely one of the predicates is true.)

By Lemma 4.5, we can compute each f_i by parallel CRC's. Assume that for each $1 \leq i \leq m$ and each $1 \leq j \leq l$, the jth output of the ith function is represented by species $\widehat{Y}_j^i$. Each $\widehat{Y}_j^i$ is an "inactive" version of "active" output species Y_j^i.

For each $1 \leq i \leq m$, we assume that the CRD computing the predicate ϕ_i represents its output by voting species L^i_1 to represent "yes" and L^i_0 to represent "no". Then add the following reactions for each $1 \leq i \leq m$ and each $1 \leq j \leq l$:

$$\begin{aligned} L^i_1 + \widehat{Y}^i_j &\to L^i_1 + Y^i_j + Y_j \\ L^i_0 + Y^i_j &\to M^i_j \\ M^i_j + Y_j &\to L^i_0 + \widehat{Y}^i_j. \end{aligned}$$

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^i_1 present, and for all $i' \neq i$, $L^{i'}_0$ is present. At this point, all outputs for the correct function f_i are activated and all other outputs are deactivated. Since eventually the count of Y^i_j stabilizes to 0 for all but one value of i, this ensures that $\#Y_j$ stabilizes to the correct value of output $\mathbf{y}_j$.

It remains to analyze the expected time to stabilization. Let $n = \|\mathbf{x}\|$. By Lemma 4.5, the expected time for each affine function computation to complete is $O(n \log n)$. Since the $\widehat{Y}^j_i$ are produced monotonically, the most Y^j_i molecules that are ever produced is $\#_\infty \widehat{Y}^j_i$. Since we have m computations in parallel, the expected time for all of them to complete is $O((n \log n)m) = O(n \log n)$ (since m depends on f but not n). We must also wait for each predicate computation to complete. By Theorem 5 of [2], each of these predicates takes expected time $O(n)$ to complete, so all of them complete in expected time $O(nm) = O(n)$.

At this point, the L^i_1 leaders must convert inactive output species to active, and $L^{i'}_0$ (for $i' \neq i$) must convert active output species to inactive. A similar analysis to the proof of Lemma 4.5 shows that each of these requires at most $O(n \log n)$ expected time, therefore they all complete in expected time $O((n \log n)m) = O(n \log n)$. □

Lemma 4.3. *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* 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 $f(\mathbf{x}) = f_i(\mathbf{x})$.*

Proof. Let $F = \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{N}^k \times \mathbb{N}^l \;\middle|\; f(\mathbf{x}) = \mathbf{y} \right\}$ be the graph of f. Since F is semilinear, it is a finite union of linear sets $\{L_1, \ldots, L_n\}$. It suffices to show that each of these linear sets L_m is the graph of an affine partial function. Let L'_m be the $(k+1)$-dimensional projection of L_m onto the coordinates defined by $\mathbf{x}$ and $\mathbf{y}_i$, which is linear because L_m is. Since L'_m is linear, there exist vectors $\mathbf{b}, \mathbf{u}^1, \ldots, \mathbf{u}^p \in \mathbb{N}^{k+1}$ such that $L'_m = \left\{ \mathbf{b} + n_1\mathbf{u}^1 + \ldots + n_p\mathbf{u}^p \;\middle|\; n_1, \ldots, n_p \in \mathbb{N} \right\}$.

It suffices to show that L'_m is a subset of a k-dimensional hyperplane. This is true if at most k of the $\mathbf{u}^1, \ldots, \mathbf{u}^p$ are linearly independent. Suppose not; then there are $k+1$ linearly independent vectors among the list. Assume without loss of generality that they are $\mathbf{u}^1, \ldots, \mathbf{u}^{k+1}$. For each $1 \leq i \leq k+1$, let $\mathbf{v}^i$ be $\mathbf{u}^i$ projected onto the first k coordinates. Since there are $k+1$ vectors and they are k-dimensional, $\mathbf{v}^1, \ldots, \mathbf{v}^{k+1}$ must be linearly dependent. By Lemma 4.4, there exist two lists of natural numbers $N = (n_1, \ldots, n_{k+1})$ and $M = (m_1, \ldots, m_{k+1})$ such that $N \neq M$ and $\sum_{i=1}^{k+1} n_i\mathbf{v}^i = \sum_{i=1}^{k+1} m_i\mathbf{v}^i$. Then the points

$$\mathbf{z}^1 = \mathbf{b} + \sum_{i=1}^{k+1} n_i\mathbf{u}^i \text{ and } \mathbf{z}^2 = \mathbf{b} + \sum_{i=1}^{k+1} m_i\mathbf{u}^i$$

are in L'_m. They must have different y-coordinates, or else we would have $\mathbf{z}^1 = \mathbf{z}^2$ (since their first k coordinates agree), which would contradict the linear independence of $\mathbf{u}^1, \ldots, \mathbf{u}^{k+1}$. Therefore

L'_m does not define the graph of a function since these two identical inputs map to two different outputs, a contradiction. $\square$

Lemma 4.4. *Let $\mathbf{v}^1, \ldots, \mathbf{v}^t \in \mathbb{N}^k$ be linearly dependent vectors. Then there are two lists of natural numbers $N = (n_1, \ldots, n_t) \in \mathbb{N}^t$ and $M = (m_1, \ldots, m_t) \in \mathbb{N}^t$ such that $N \neq M$ and $\sum_{i=1}^t n_i \mathbf{v}^i = \sum_{i=1}^t m_i \mathbf{v}^i$.*

Proof. By the definition of linear dependence, there exist two lists of real numbers $N' = (n'_1, \ldots, n'_t) \in \mathbb{R}^t$ and $M' = (m'_1, \ldots, m'_t) \in \mathbb{R}^t$ such that $N' \neq M'$ and $\sum_{i=1}^t n'_i \mathbf{v}^i = \sum_{i=1}^t m'_i \mathbf{v}^i \in \mathbb{N}^k$ (since all points, integer or not, in the basis of $\mathbf{v}^1, \ldots, \mathbf{v}^t \in \mathbb{N}^k$ can be so expressed). Perhaps some of the coefficients are negative; however, by increasing both n'_i and m'_i by $\min\{n'_i, m'_i\}$ (which changes each sum by the same amount, keeping them equal), we may assume that all coefficients are nonnegative. Furthermore, since the sum $\sum_{i=1}^t n_i \mathbf{v}^i$ is integer-valued, $N', M' \in \mathbb{Q}^t$.

Let L be the least common multiple of the denominators of each n'_i and m'_i when expressed in lowest terms. By multiplying each coefficient by L, we obtain nonnegative integers $N = (n_1, \ldots, n_t) \in \mathbb{N}^t$ and $M = (m_1, \ldots, m_t) \in \mathbb{N}^t$ such that $N \neq M$ and $\sum_{i=1}^t n_i \mathbf{v}^i = \sum_{i=1}^t m_i \mathbf{v}^i$. $\square$

Lemma 4.5. *Let $f : \mathbb{N}^k \dashrightarrow \mathbb{N}^l$ be a partial affine function. Then there is a CRC that computes f on input $\mathbf{x}$ in expected time $O(\|\mathbf{x}\| \log \|\mathbf{x}\|)$, such that the output molecules monotonically increase with time (i.e. none are ever consumed).*

Proof. If $\mathbf{y} = f(\mathbf{x})$, then there exist $kl+l+k$ integers $a_1^1, \ldots, a_k^l \in \mathbb{Q} \cap [0, \infty)$ and $b_1, \ldots, b_l, c_1, \ldots, c_k \in \mathbb{Z}$ such that each $\mathbf{y}_j = b_j + \sum_{i=1}^k a_i^j (\mathbf{x}_i + c_i)$. Define the CRC as follows. It has input species $\Sigma = \{X_1, \ldots, X_k\}$ and output species $\Gamma = \{Y_1, \ldots, Y_l\}$.

Let $b_1^j, \ldots, b_k^j \in \{-|b_j|, \ldots, |b_j|\}$ be integers such that $\sum_{i=1}^k b_i^j = b_j$. Let $c_i^1, \ldots, c_i^l \in \{-|c_i|, \ldots, |c_i|\}$ be integers such that $\sum_{j=1}^l c_i^j = c_i$. For each $1 \leq i \leq k$ and $1 \leq j \leq l$, start with a leader molecule $\widehat{L}^j_{i,c_i^j}$. For each $1 \leq i \leq k$ and $1 \leq j \leq l$, let $\frac{n_i^j}{d_i^j}$ be a_i^j expressed as a fraction such that $d_i^j > c_i$ and, if $b_j < 0$, $n_i^j \geq -b_j$. For each $1 \leq i \leq k$, add the reaction

$$X_i \quad \to \quad X_i^1 + X_i^2 + \ldots + X_i^l$$

For each $1 \leq i \leq k$, $1 \leq j \leq l$ and m such that $\min\{0, c_i^j\} \leq m \leq d_i^j$, add the reactions

$$\widehat{L}^j_{i,m} + X_i^j \quad \to \quad \begin{cases} \widehat{L}^j_{i,m+1}, & \text{if } m < d_i^j; \\ L^j_{i,0} + (n_i^j + b_i^j) Y_j, & \text{otherwise.} \end{cases}$$

$$L^j_{i,m} + X_i^j \quad \to \quad \begin{cases} L^j_{i,m+1}, & \text{if } m < d_i^j; \\ L^j_{i,0} + n_i^j Y_j, & \text{otherwise.} \end{cases}$$

Each initial leader $\widehat{L}$ starts counting (m is the "current count") at an initial value either above or below 0 (depending on the sign of c_i) to account for the initial offset c_i of X_i. Also, each initial leader releases a different amount of Y_j (again depending on the sign of b_j) to account for the initial offset b_j of Y_j. After counting to d_i^j, each initial leader $\widehat{L}$ converts to a normal leader L, which releases n_i^j Y_j molecules for every d_i^j X_i molecules encountered. Therefore (after accounting for initial offsets) each X_i molecule converts into a number of Y_j molecules based on the weighted sum of the a_i^j coefficients. Therefore when all X_i^j molecules are consumed, the number of Y_j molecules is the proper value $\mathbf{y}_j = b_j + \sum_{i=1}^{k} a_i^j(\mathbf{x}_i + c_i)$.

It remains to analyze the expected completion time. Let $n = \|\mathbf{x}\|$. Since the total number of molecules in solution at any time is $O(n)$, the volume required is also $O(n)$. We measure the time to consume all X_i molecules for all i. We start with n such molecules, so the time for all of them to convert is the maximum of n exponential random variables, each with constant expected value, which is $O(\log n)$.

Then all L_i^j molecules must encounter every X_i^j molecule. By a coupon collector argument, this requires at most $O(n \log n)$ time. Therefore the CRC stabilizes in expected time $O(n \log n)$. □

Acknowledgements. We thank Damien Woods and Niranjan Srinivas for many useful discussions.

References

[1] Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, pages 61–75, 2006.

[2] Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *PODC*, pages 292–299, 2006.

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

[4] James Aspnes and Eric Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98–117, 2007.

[5] Luca Cardelli. Strand algebras for DNA computing. *Natural Computing*, 10(1):407–428, 2011.

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

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

[8] Hua Jiang, Marc Riedel, and Keshab Parhi. Digital signal processing with molecular reactions. *IEEE Design and Test of Computers*, 2012. to appear.

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

[10] Mojżesz Presburger. Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen. In *welchem die Addition als einzige Operation hervortritt. Compte Rendus du I. Congrks des Mathematiciens des pays Slavs, Warsaw*, pages 92–101, 1930.

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

[12] 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.

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