Composable Computation in Discrete Chemical Reaction Networks

Eric E. Severson*
eseverson@ucdavis.edu
University of California, Davis
Davis, California

David Haley*
drhaley@ucdavis.edu
University of California, Davis
Davis, California

David Doty*
doty@ucdavis.edu
University of California, Davis
Davis, California

ABSTRACT

We study the composability of discrete chemical reaction networks (CRNs) that *stably compute* (i.e., with probability 0 of error) integer-valued functions $f : \mathbb{N}^d \to \mathbb{N}$. We consider *output-oblivious* CRNs in which the output species is never a reactant (input) to any reaction. The class of output-oblivious CRNs is fundamental, appearing in earlier studies of CRN computation, because it is precisely the class of CRNs that can be composed by simply renaming the output of the upstream CRN to match the input of the downstream CRN.

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

CCS CONCEPTS

• **Theory of computation** → *Distributed computing models*; *Distributed algorithms.*

KEYWORDS

chemical reaction network, population protocol, composable computation, semilinear

ACM Reference Format:
Eric E. Severson, David Haley, and David Doty. 2019. Composable Computation in Discrete Chemical Reaction Networks. In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29-August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3293611.3331615

1 INTRODUCTION

A foundational model of chemistry commonly used in natural sciences is that of chemical reaction networks (CRNs): finite sets of chemical reactions such as $A+B \to A+C$. The model is described as a continuous time, discrete state, Markov process [18]. A configuration of the system is a vector of non-negative integers specifying the molecular counts of the species (e.g., A, B, C), a reaction can occur only when all its reactants are present, and transitions between configurations correspond to reactions (e.g., when the above reaction occurs the count of B is decreased by 1 and the count of C increased by 1). CRNs are widely used to describe natural biochemical systems such as the intricate cellular regulatory networks responsible for the information processing within cells. Looking beyond the *scientific* goal of understanding natural CRNs, to the *engineering* goal of constructing programmable, autonomous smart molecules, artificial CRNs have been implemented using the physical primitive of nucleic-acid strand displacement cascades [8, 11, 22, 23].

Population protocols, a widely-studied model of distributed computing with very limited agents, are a restricted subset of CRNs (those with two reactants and two products in each reaction) that nevertheless capture many of the interesting features of CRNs. The key feature is the inability of agents (molecules) to control their schedule of communication (collisions). The decision problems solvable by population protocols have been studied extensively: they can simulate Turing machines with high probability in polylogarithmic time (with [5] or without [19] an initial leader), whereas requiring probability 0 of error limits the computable predicates to being semilinear [6].

1.1 Function computation

Computation of functions $f : \mathbb{N}^d \to \mathbb{N}$ was discussed briefly in the first population protocols paper [3, Section 3.4], which focused more on Boolean predicate computation, and it was defined formally first for CRNs [10, 15] and later for population protocols [7]. The class of functions stably computable in either model is the same: the semilinear functions [6, 10]. We use the CRN model because it is more natural for describing functions, but our results also apply to the population protocol model.

To represent an input $\mathbf{x} \in \mathbb{N}^d$, we start in a configuration with counts $\mathbf{x}(i)$ of species X_i for each $i \in \{1, \ldots, d\}$, and count 1 of a "leader" species L.[1] A function $f : \mathbb{N}^d \to \mathbb{N}$ is *stably computable* by a CRN if a correct and *stable* configuration $\mathbf{O}$ (i.e., on input $\mathbf{x}$ the count of Y is $f(\mathbf{x})$ in all configurations reachable from $\mathbf{O}$) remains reachable no matter what reactions occur.[2]

See Fig. 1 for examples. It is known that a function $f : \mathbb{N}^d \to \mathbb{N}$ is stably computable by a CRN if and only if it is *semilinear*: intuitively, it is a piecewise affine function. (See Definition 2.5.)

*Authors supported by NSF grants 1619343 and 1844976.

PODC '19, July 29-August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07…$15.00
https://doi.org/10.1145/3293611.3331615

[1] The leader is discussed in Section 1.3. A CRN may ignore its leader, as in Fig. 1.
[2] We use this definition throughout the paper, but we mention here that it is equivalent to two other natural definitions. The first definition is that any fair sequence of reactions *will* take the CRN to such a correct stable configuration, where *fair* means that any configuration that is infinitely often reachable is eventually reached. The second definition is that a correct stable configuration is actually reached with probability 1.

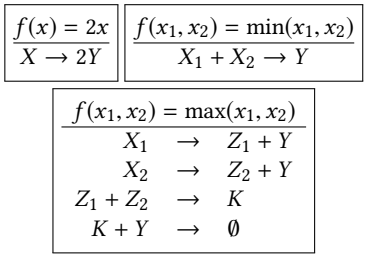


Figure 1: Functions stably computed by CRNs. Note max is computed as $x_1 + x_2 - \min(x_1, x_2)$.

1.2 Composability

Note a key difference between the CRNs for min and max in Fig. 1: the former only produces the output species Y, whereas the latter also contains reactions that consume Y. In one possible sequence of reactions for the max CRN, the inputs can be exhausted through the first two reactions before ever executing the last two reactions. In doing so, the count of Y overshoots its correct value of $\max(x_1, x_2)$ before the excess is consumed by the reaction $K + Y \to \emptyset$.

For this reason that the min CRN is more easily composed with a downstream CRN. For example, the function $2 \cdot \min(x_1, x_2)$ is stably computed by the reactions $X_1 + X_2 \to W$ (computing $w = x_1 + x_2$) and $W \to 2Y$ (computing $y = 2w$), renaming the output of the min CRN to match the input of the multiply-by-2 CRN. However, this approach does not work to compute $2 \cdot \max(x_1, x_2)$; changing Y to W in the four-reaction max CRN and adding the reaction $W \to 2Y$ can erroneously result in up to $2(x_1+x_2)$ copies of Y being produced. Intuitively, the multiply-by-2 reaction $W \to 2Y$ competes with the upstream reaction $K + W \to \emptyset$ from the max CRN.

This motivates us to study the class of functions $f : \mathbb{N}^d \to \mathbb{N}$ stably computable by *output-oblivious* CRNs: those in which the output species Y appears only as a product, never as a reactant. We call such a function *obliviously-computable*. Any obliviously-computable function must be nondecreasing, otherwise reactions could incorrectly overproduce output (see Observation 2.1).

Obliviously-computable functions must also be semilinear, so it is reasonable to conjecture that a function is obliviously-computable if and only if it is semilinear and nondecreasing. In fact, this is true for 1D functions $f : \mathbb{N} \to \mathbb{N}$ (see Section 3). However, in higher dimensions, the function $\max : \mathbb{N}^2 \to \mathbb{N}$ is semilinear and nondecreasing, yet not obliviously-computable; its consumption of output turns out to be unavoidable. Assuming there is no leader, this is simple to prove: Since $\max(1, 0) = 1$, starting with one X_1, a Y can be produced. Similarly, a Y can be produced starting with one X_2. Then with one X_1 and one X_2, these reactions can happen in parallel and produce two Y's, too many since $\max(1, 1) = 1$. It is more involved to prove that even with a leader, it remains impossible to obliviously compute max; see Section 4.[3]

1.3 The role of the leader

Our model includes an initial leader, which is essential for our general constructions (see Sections 3 and 6). The class of stably computable functions is identical whether an initial leader is allowed or not [15], as is the class of stably computable predicates [6].

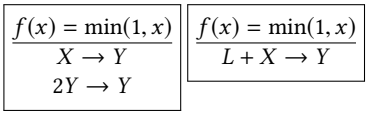


Figure 2: $\min(1, x)$ is stably computed by a *leaderless* non-output-oblivious CRN (left), and an output-oblivious CRN with a single leader L (right).

Interestingly, the class of obliviously-computable functions we study is provably larger when an initial leader is allowed. For example, consider the function $f(x) = \min(1, x)$ (see Fig. 2). f is stably computable with or without a leader, but only the construction with a leader is output-oblivious. Without using a leader, f is not obliviously-computable (see Observation 9.1).

Including the leader gives additional power to the model. This gives more power to our CRN constructions, but makes our impossibility results stronger. Fully classifying the obliviously-computable functions in a leaderless model remains an open question.

1.4 Contribution

Our main result, Theorem 5.2, provides a complete characterization of the class of obliviously-computable functions. It builds off a key definition: a *quilt-affine* function is a nondecreasing function that is the sum of a rational linear function and periodic function (formalized as Definition 5.1). For example, functions such as $\lfloor \frac{3x}{2} \rfloor$ are quilt-affine (see Fig. 3a). Such floored division functions are natural to the discrete CRN model ($\lfloor \frac{3x}{2} \rfloor$ is stably computed by reactions $X \to 3Z$, $2Z \to Y$). Fig. 3b shows a higher-dimensional quilt-affine function, with a "bumpy quilt" structure that motivates the name. Quilt-affine functions are also characterized by nonnegative periodic finite differences, a structure key to showing they are obliviously-computable (see Lemma 6.1).

Theorem 5.2 states that a function $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable if and only if

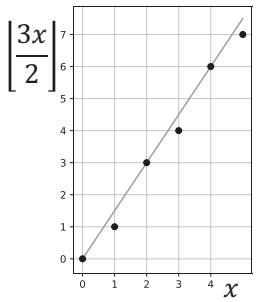
i) [nondecreasing] f is nondecreasing,
ii) [eventually-min] for sufficiently large inputs, f is the minimum of a finite number of quilt-affine functions, and
iii) [recursive] every restriction $f_r : \mathbb{N}^{d-1} \to \mathbb{N}$ obtained by fixing some inputs to a constant value[4] is obliviously-computable (i.e., eventually the minimum of quilt-affine functions).

Condition (ii) characterizes f when all inputs are sufficiently large (greater than some $\mathbf{n} \in \mathbb{N}^d$), whereas condition (iii) characterizes f when some inputs are fixed to smaller values. See Fig. 4a for a representative example of an obliviously-computable $f : \mathbb{N}^2 \to \mathbb{N}$. This pictured function has arbitrary nondecreasing values in the "finite region" where $\mathbf{x} < (4, 4)$, has eventual 1D quilt-affine behavior along the lines $x_1 = 0, 1, 2, 3$ and $x_2 = 0, 1, 2, 3$, and is the minimum of 3 different quilt-affine functions in the "eventual region" where $\mathbf{x} \geq (4, 4)$. This behavior generalizes naturally to higher dimensions.

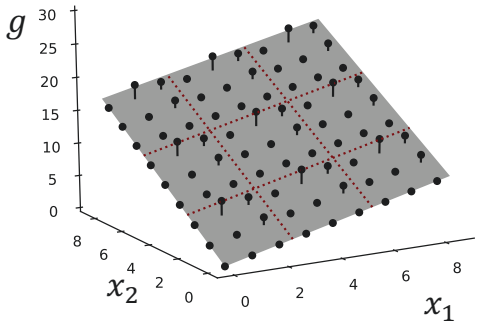
The most technically sophisticated part of our result is the proof that the eventually-min condition (ii) of Theorem 5.2 is necessary; the main ideas of this proof are outlined in Section 7.1.

[3] This result was obtained independently by Chugg, Condon, and Hashemi [12].

[4] Note that Theorem 5.2 defines fixed-input restrictions slightly differently; see Section 5 for an explanation.

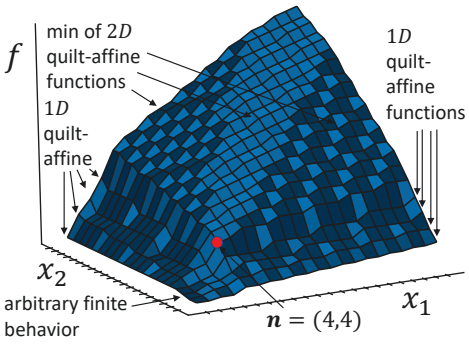


(a) A 1D (single-input) quilt-affine function $\lfloor \frac{3x}{2} \rfloor = \frac{3}{2}x + B(\overline{x} \mod 2)$, where $B(\overline{0}) = 0$ and $B(\overline{1}) = -\frac{1}{2}$.

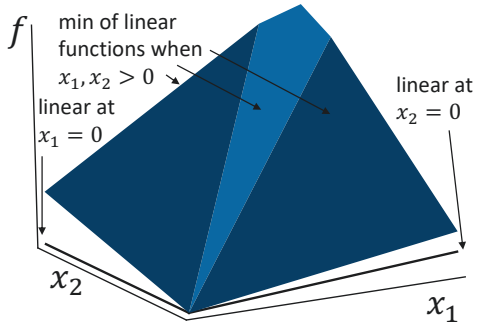


(b) A 2D quilt-affine function $g(\mathbf{x}) = (1, 2) \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod 3)$, where $B(\overline{\mathbf{x}}) = 0$ except when $\overline{\mathbf{x}} \in \{\overline{(1, 2)}, \overline{(2, 2)}, \overline{(2, 1)}\}$.

Figure 3: Examples of 1D and 2D quilt-affine functions.



(a) A 2D function satisfying Theorem 5.2.



(b) The scaling limit gives a 2D real-valued obliviously-computable function from [9].

Figure 4: Example discrete and real-valued obliviously-computable functions.

1.5 Related work

Chalk, Kornerup, Reeves, and Soloveichik [9] showed an analogous result in the *continuous* model of CRNs in which species amounts are given by nonnegative real concentrations. A consequence of their characterization is that any obliviously-computable real-valued function is a minimum of linear functions when all inputs are positive. In Theorem 8.2, we demonstrate that the limit of "scalings" of a function $f : \mathbb{N}^d \to \mathbb{N}$ satisfying our main Theorem 5.2 is in fact a function $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ satisfying the main theorem of [9] (see Fig. 4b). The discrete details lost in the scaling limit constitute precisely the unique challenges of proving Theorem 5.2 that are not handled by [9]. In particular, our function class can contain arbitrary finite behavior and repeated finite irregularities.

Returning to the *discrete* (a.k.a., *stochastic*) CRN model we study, Chugg, Condon, and Hashemi [12] independently investigated the special case of *two-input* functions $f : \mathbb{N}^2 \to \mathbb{N}$ computable by output-oblivious CRNs, obtaining a characterization equivalent to ours when restricted to 2D. Their characterization is phrased much differently, with specially constructed "fissure functions" to describe the function behavior across what we describe as *under-determined regions* (intuitively, thin "1D" regions bounded by parallel lines, where f cannot be described by a unique quilt-affine function, see Section 7). The ideas required to prove the 2D case are sophisticated and far from simple, yet unfortunately, these ideas do not extend straightforwardly to higher dimensions. The planarity of the 2D input space constrains the regions induced by separating hyperplanes (i.e., lines) in a strong way. Furthermore, the fact that there is only one nontrivial integer dimension smaller than 2 implies that the under-determined regions are simpler to reason about than in the case where they can have arbitrary dimension between 1 and d. Finally, even restricted to 2D, a notable aspect of our characterization is expressing f a minimum of quilt-affine functions, which are simple intrinsic building blocks that generalize immediately.

1.6 Other ways of composing computation

In Section 2.3 we show that a for a CRN $\mathcal{C}$ be composable with downstream CRN $\mathcal{D}$ by "concatenation" (renaming $\mathcal{C}$'s output species to match $\mathcal{D}$'s input species and ensuring all other species names are disjoint between $\mathcal{C}$ and $\mathcal{D}$), it is (in a sense) necessary and sufficient for $\mathcal{C}$ to be output-oblivious. There are other ways to compose computations, however.

A common technique (e.g. [17]) is for C to detect when its output has changed and send a *restart* signal to $\mathcal{D}$. However, it is not obvious how to do this with function computation as defined in this paper, where $\mathcal{D}$ changes C's output by consuming it.

Another technique (e.g. [5]) is to set a *termination signal*, which is a sub-CRN that, with high probability, creates a copy of a signal species T, but not before C has converged. T then "activates" the reactions of $\mathcal{D}$, so that $\mathcal{D}$ will not consume the output of C until it is safe to do so. However, this has some positive failure probability. In fact, if we require T to be guaranteed with probability 1 to be produced only after the CRN has converged, only constant functions can be stably computed. Worse yet, in the leaderless case, it is provably impossible to achieve this guarantee even with positive probability [14].

2 PRELIMINARIES

2.1 Notation

$\mathbb{N}$ denotes the set of nonnegative integers. For a set $\mathcal{S}$ (of species), we write $\mathbb{N}^{\mathcal{S}}$ to denote the set of vectors indexed by the elements of $\mathcal{S}$ (equivalently, functions $f : \mathcal{S} \to \mathbb{N}$). Vectors appear in boldface, and we reserve uppercase $\mathbf{A} \in \mathbb{N}^{\mathcal{S}}$ for such vectors indexed by species, and lowercase $\mathbf{a} \in \mathbb{N}^d, \mathbb{Z}^d, \mathbb{Q}^d, \mathbb{R}^d$ for vectors indexed by integers. $\mathbf{A}(S)$ or $\mathbf{a}(i)$ denotes the element indexed by $S \in \mathcal{S}$ or $i \in \{1, \ldots, d\}$. We write $\mathbf{a} \leq \mathbf{b}$ to denote pointwise vector inequality $\mathbf{a}(i) \leq \mathbf{b}(i)$ for all i.

For $p \in \mathbb{N}_+$, $\mathbb{Z}/p\mathbb{Z}$ denotes the additive group of integers modulo p, whose elements are congruence classes. Generalizing to higher dimensions, $\mathbb{Z}^d/p\mathbb{Z}^d$ denotes the additive group of $\mathbb{Z}^d$ modulo p, whose elements are congruence classes. For $\mathbf{x} \in \mathbb{N}^d$ where $d \geq 1$, we write $\overline{\mathbf{x}} \mod p$ to denote the congruence class $\{\mathbf{x} + p\mathbf{z} : \mathbf{z} \in \mathbb{Z}^d\} \in \mathbb{Z}^d/p\mathbb{Z}^d$, also denoted $\overline{\mathbf{x}}$ when p is clear from context.

2.2 Chemical reaction networks

We use the established definitions of stable function computation by (discrete) chemical reaction networks [4, 12]:

A *chemical reaction network* (CRN) $C = (\mathcal{S}, \mathcal{R})$ is defined by a finite set $\mathcal{S}$ of species and a finite set $\mathcal{R}$ of reactions, where a reaction $(\mathbf{R}, \mathbf{P}) \in \mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}$ describes the counts of consumed *reactant* species and produced *product* species.[5] For example, given $\mathcal{S} = \{A, B, C\}$, the reaction $((1, 0, 2), (0, 2, 1))$ would represent $A + 2C \to 2B + C$.

A configuration $\mathbf{C} \in \mathbb{N}^{\mathcal{S}}$ specifies the integer counts of all species. Reaction $(\mathbf{R}, \mathbf{P})$ is *applicable* to $\mathbf{C}$ if $\mathbf{R} \leq \mathbf{C}$, and yields $\mathbf{C}' = \mathbf{C} - \mathbf{R} + \mathbf{P}$, so we write $\mathbf{C} \to \mathbf{C}'$. A configuration $\mathbf{D}$ is *reachable* from $\mathbf{C}$ if there exists a finite sequence of configurations such that $\mathbf{C} \to \mathbf{C}_1 \to \ldots \to \mathbf{C}_n \to \mathbf{D}$; we write $\mathbf{C} \to^* \mathbf{D}$ to denote that $\mathbf{D}$ is reachable from $\mathbf{C}$. Note this reachability relation is additive: if $\mathbf{A} \to^* \mathbf{B}$, then $\mathbf{A} + \mathbf{C} \to^* \mathbf{B} + \mathbf{C}$. This property is key in future proofs to show the reachability of configurations which overproduce output.

To compute a function[6] $f : \mathbb{N}^d \to \mathbb{N}$, the CRN C will include an ordered subset $\{X_1, \ldots, X_d\} \subset \mathcal{S}$ of *input species*, an *output species* Y, and a *leader species* $L \in \mathcal{S}$. (Note that we consider removing the leader in Section 9).

The computation of $f(\mathbf{x})$ will start from an *initial configuration* $\mathbf{I}_{\mathbf{x}}$ encoding the input with $\mathbf{I}_{\mathbf{x}}(X_i) = \mathbf{x}(i)$ for all $i = 1, \ldots, d$, along with a single leader $\mathbf{I}_{\mathbf{x}}(L) = 1$, and count 0 of all other species. A *stable* configuration $\mathbf{C}$ has unchanged output $\mathbf{C}(Y) = \mathbf{D}(Y)$ for any configuration $\mathbf{D}$ reachable from $\mathbf{C}$. The CRN C *stably computes* $f : \mathbb{N}^d \to \mathbb{N}$ if for each initial configuration $\mathbf{I}_{\mathbf{x}}$ encoding any $\mathbf{x} \in \mathbb{N}^d$, and configuration $\mathbf{C}$ reachable from $\mathbf{I}_{\mathbf{x}}$, there is a stable configuration $\mathbf{O}$ reachable from $\mathbf{C}$ with correct output $\mathbf{O}(Y) = f(\mathbf{x})$.

2.3 Composition via output-oblivious CRNs

This section formally defines our notions of "composable computation with CRNs via concatenation of reactions" and "output-oblivious" CRNs that don't consume their output, showing these notions to be essentially equivalent.

A CRN is *output-oblivious* if the output species Y is never a reactant[7]: for any reaction $(\mathbf{R}, \mathbf{P})$, $\mathbf{R}(Y) = 0$. A function $f : \mathbb{N}^d \to \mathbb{N}$ is *obliviously-computable* if f is stably computed by an output-oblivious CRN.

We begin with an easy observation:

Observation 2.1. *An obliviously-computable function $f : \mathbb{N}^d \to \mathbb{N}$ must be nondecreasing.*

Proof. Assume a CRN C (with output species Y) stably computes f, but $f(\mathbf{a}) > f(\mathbf{b})$ for $\mathbf{a} \leq \mathbf{b}$. To stably compute $f(\mathbf{a})$, input configuration $\mathbf{I}_{\mathbf{a}} \to^* \mathbf{O}$ for some configuration with $\mathbf{O}(Y) = f(\mathbf{a})$. However, since $\mathbf{a} \leq \mathbf{b}$, that same sequence of reactions can be applied from the input configuration $\mathbf{I}_{\mathbf{b}} \geq \mathbf{I}_{\mathbf{a}}$. This overproduces Y since $f(\mathbf{a}) > f(\mathbf{b})$. Thus to stably compute $f(\mathbf{b})$, some reaction must consume Y as a reactant, so C cannot be output-oblivious. □

A CRN being output-oblivious was shown in [9] (for *continuous* CRNs) to be equivalent to being "composable via concatenation", meaning renaming the output species of one CRN to match the input of another. This equivalence still holds in our discrete CRN model. This is formalized as Observation 2.2 and Lemma 2.3.

For CRNs C_f stably computing $f : \mathbb{N}^d \to \mathbb{N}$ and C_g stably computing $g : \mathbb{N} \to \mathbb{N}$, define the *concatenated* CRN $C_{g \circ f}$ by combining species and reactions, with C_f's output species as C_g's input species and no other common species, plus a reaction $L \to L^f + L^g$ creating a copy of the leader from each of C_f and C_g.

We first observe that this composition works correctly if the *upstream* CRN C_f is output-oblivious. Intuitively, the reactions from C_g can only affect the reactions from C_f via the common species W, but this output species of C_f is never used as a reactant to stably compute $f(\mathbf{x})$.

Observation 2.2. *If C_f stably computes $f : \mathbb{N}^d \to \mathbb{N}$, C_g stably computes $g : \mathbb{N} \to \mathbb{N}$, and C_f is output-oblivious, then the concatenated CRN $C_{g \circ f}$ stably computes the composition $g \circ f : \mathbb{N}^d \to \mathbb{N}$.*

Note that the downstream CRN C_g need not be output-oblivious, but if two output-oblivious CRNs are composed, then the composition $C_{g \circ f}$ remains output-oblivious. More generally, g can take

[5] We do not limit ourselves to *bimolecular* (two input) reactions, but the higher-order reactions we use can easily be converted to have this form. For example, $3X \to Y$ is equivalent to two reactions $2X \leftrightarrow X_2$ and $X + X_2 \to Y$.

[6] We consider codomain $\mathbb{N}$ without loss of generality, since $f : \mathbb{N}^d \to \mathbb{N}^l$ is stably computable if and only if each output component is stably computable by parallel CRNs.

[7] A more general definition in [12] of *output-monotonic* CRNs just requires no reaction to reduce the count of output species. This can be directly seen to classify the same set of functions, see [21].

any number of inputs from output-oblivious CRNs, which act as modules for arbitrary feedforward composition.

The converse shows that a composable CRN is essentially output-oblivious. If C_f can be correctly composed with any downstream C_g, then C_f must function correctly even if downstream reactions from C_g starve it of the common species W. Thus C_f will still stably compute f if we remove all reactions with output W as a reactant, making it output-oblivious. The proof appears in [21].

Lemma 2.3. *Let C_f stably compute $f : \mathbb{N}^d \to \mathbb{N}$ such that for any C_g stably computing $g : \mathbb{N} \to \mathbb{N}$, the concatenated CRN $C_{g\circ f}$ stably computes the composition $g \circ f : \mathbb{N}^d \to \mathbb{N}$. Then C_f still stably computes f if we remove all reactions using the output species as a reactant, making it output-oblivious.*

2.4 Semilinear functions

The functions stably computable by a CRN were shown in [10], building from work in [4], to be precisely the *semilinear* functions, which are defined based on semilinear sets[8]

Definition 2.4. *A subset $S \subseteq \mathbb{N}^d$ is* semilinear *if S is a finite Boolean combination (union, intersection, complement) of* threshold *sets of the form $\{\mathbf{x} \in \mathbb{N}^d : \mathbf{a} \cdot \mathbf{x} \geq b\}$ for $\mathbf{a} \in \mathbb{Z}^d, b \in \mathbb{Z}$ and* mod *sets of the form $\{\mathbf{x} \in \mathbb{N}^d : \mathbf{a} \cdot \mathbf{x} \equiv b \mod c\}$ for $\mathbf{a} \in \mathbb{Z}^d, b \in \mathbb{Z}, c \in \mathbb{N}_+$.*

A *semilinear function* can be concisely defined as having a semilinear graph, but a more useful equivalent definition comes from Lemma 4.3 of [10][9]:

Definition 2.5 ([10]). *A function $f : \mathbb{N}^d \to \mathbb{N}$ is* semilinear *if f is the finite union of affine partial functions, whose domains are disjoint semilinear subsets of $\mathbb{N}^d$.*

All functions discussed have been semilinear. For example, the function

$$\min(x_1, x_2) = \begin{cases} x_1, & \text{if } x_1 \leq x_2 \\ x_2, & \text{if } x_1 > x_2 \end{cases}$$

is semilinear with affine partial functions on disjoint domains which are defined by a single threshold and thus semilinear.

Similarly, the function

$$\left\lfloor \frac{3x}{2} \right\rfloor = \begin{cases} \frac{3}{2}x, & \text{if } x \text{ is even} \\ \frac{3}{2}x - \frac{1}{2}, & \text{if } x \text{ is odd} \end{cases}$$

is semilinear with affine partial functions on disjoint domains which are defined by parity (a single mod predicate) and thus semilinear. All quilt-affine $f : \mathbb{N}^d \to \mathbb{N}$ are semilinear by the same argument.

Lemma 2.6 ([10]). *A function $f : \mathbb{N}^d \to \mathbb{N}$ is stably computable $\iff$ f is semilinear.*

3 WARM-UP: ONE-DIMENSIONAL CASE

For functions with one-dimensional input, the necessary conditions of being nondecreasing and semilinear are also sufficient.

Theorem 3.1. *$f : \mathbb{N} \to \mathbb{N}$ is obliviously-computable $\iff$ f is semilinear and nondecreasing.*

[8]Semilinear sets have other common equivalent definitions [3]; the above definition is convenient for our proof.

[9]Lemma 4.3 in [10] has domains that are non-disjoint linear sets. We assume the domains are disjoint for convenience, making the domains semilinear sets.

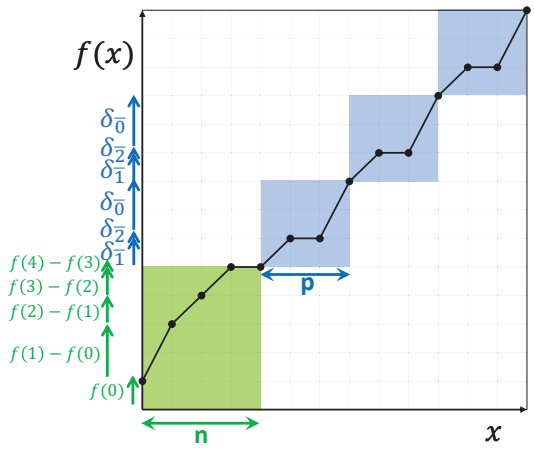


Figure 5: Every semilinear nondecreasing $f : \mathbb{N} \to \mathbb{N}$ is eventually quilt-affine, with periodic finite differences $\delta_{\overline{x}}$.

Intuitively, the proof works as follows. We show semilinear, nondecreasing $f : \mathbb{N} \to \mathbb{N}$ must have the eventually quilt-affine structure in Fig. 5. From this structure, we define a CRN that uses auxiliary leader states to track the value of x (or $\overline{x} \mod p$ once $x \geq n$), while outputting the correct finite differences from adding each input.

Proof. $\implies$: Lemma 2.6 and Observation 2.1.

$\impliedby$: If $f : \mathbb{N} \to \mathbb{N}$ is semilinear and nondecreasing, it will eventually be *quilt-affine* (generalized to higher-dimensional functions as Definition 5.1) and thus have periodic finite differences: for some $n \in \mathbb{N}$, period $p \in \mathbb{N}_+$, and finite differences $\delta_{\overline{0}}, \ldots, \delta_{\overline{p-1}} \in \mathbb{N}$, then for all $x \geq n$, $f(x+1) - f(x) = \delta_{(\overline{x} \mod p)}$ (see Fig. 5).

Because f is semilinear, by Definitions 2.4 and 2.5, it can be represented as a disjoint union of affine partial functions, whose domains are semilinear sets, and thus represented as finite Boolean combinations of threshold $\{x \in \mathbb{N} : x \geq a\}$ and mod $\{x \in \mathbb{N} : x \equiv b \mod c\}$ sets. Now take $n \in \mathbb{N}$ greater than all such a and $p = \mathrm{lcm}(c)$ for all such c. Then for all $x \geq n$, f periodically cycles between affine partial functions. Because f is nondecreasing, these periodically-repeated affine partial functions must all have the same slope. This implies f is eventually quilt-affine, with periodic finite differences for all $x \geq n$ as claimed.

The CRN C to stably compute f uses input species X, output species Y, leader L, and species $L_0, \ldots, L_{n-1}, P_{\overline{0}}, \ldots, P_{\overline{p-1}}$ corresponding to auxiliary "states" of the leader, i.e., exactly one of $L, L_0, \ldots, L_{n-1}, P_{\overline{0}}, \ldots, P_{\overline{p-1}}$ is present at any time. Intuitively the leader tracks how many input X it has seen, where the count past n wraps around mod p, and outputs the correct finite differences. The reactions of C are as follows

$$\begin{aligned} L &\to f(0)Y + L_0 \\ L_i + X &\to [f(i+1) - f(i)]Y + L_{i+1} \qquad \text{for all } i = 0, \ldots, n-2 \\ L_{n-1} + X &\to [f(n) - f(n-1)]Y + P_{\overline{n}} \\ P_{\overline{a}} + X &\to \delta_{\overline{a}} Y + P_{\overline{a+1}} \qquad \text{for all } \overline{a} = \overline{0}, \ldots, \overline{p-1}. \end{aligned}$$

□

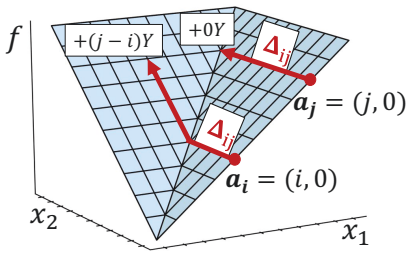


Figure 6: Lemma 4.1 applied to $f = \max(x_1, x_2)$.

In the 1D case, we can also characterize the functions obliviously-computable *without* a leader: they are semilinear and *superadditive*: meaning $f(x) + f(y) \leq f(x + y)$ for all x, y. (Theorem 9.2)

4 IMPOSSIBILITY RESULT

The characterization of obliviously-computable functions as precisely semilinear and nondecreasing from Theorem 3.1 is insufficient in higher dimensions. As an example, consider the function $\max : \mathbb{N}^2 \to \mathbb{N}$, which is both semilinear and nondecreasing. We prove max is not obliviously-computable via a more general lemma:

Lemma 4.1. *Let $f : \mathbb{N}^d \to \mathbb{N}$. If there exists an increasing sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots) \in \mathbb{N}^d$ such that for all $i < j$ there exists some $\mathbf{\Delta}_{ij} \in \mathbb{N}^d$ with $f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j)$, then f is not obliviously-computable.*

Before proving Lemma 4.1, we use it to show max is not obliviously-computable.

For $f = \max(x_1, x_2)$, we let $\mathbf{a}_i = (i, 0)$ and $\mathbf{\Delta}_{ij} = (0, j)$, so for $i < j$, $\max(i, j) - \max(i, 0) = j - i > \max(j, j) - \max(j, 0) = 0$ as desired (see Fig. 6). Adding $\mathbf{\Delta}_{ij}$ input after computing $f(\mathbf{a}_i)$ should produce $j - i$ additional output Y. However, adding $\mathbf{\Delta}_{ij}$ input after computing $f(\mathbf{a}_j)$ should not. Lemma 4.1 uses this to show there exists a reaction sequence that overproduces Y, thus max is not obliviously-computable. We now prove Lemma 4.1.

Proof. Assume toward contradiction an output-oblivious CRN C stably computes f. To stably compute each $f(\mathbf{a}_i)$, the initial configuration $\mathbf{I}_{\mathbf{a}_i} \to^* \mathbf{O}_i$ for some configuration with $\mathbf{O}_i(Y) = f(\mathbf{a}_i)$, giving a sequence of configurations $(\mathbf{O}_i)_{i=1}^{\infty}$. By Dickson's Lemma [13], any sequence of nonnegative integer vectors has a nondecreasing subsequence, so there must be $\mathbf{O}_i \leq \mathbf{O}_j$ for some $i < j$. By assumption there exists $\mathbf{\Delta}_{ij} \in \mathbb{N}^d$ such that

$$f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_j)$$

Now consider the initial configuration $\mathbf{I}_{\mathbf{a}_i + \mathbf{\Delta}_{ij}} \geq \mathbf{I}_{\mathbf{a}_i}$, so define the difference $\mathbf{D} = \mathbf{I}_{\mathbf{a}_i + \mathbf{\Delta}_{ij}} - \mathbf{I}_{\mathbf{a}_i} \in \mathbb{N}^S$. Then the same sequence of reactions $\mathbf{I}_{\mathbf{a}_i} \to^* \mathbf{O}_i$ is applicable to $\mathbf{I}_{\mathbf{a}_i + \mathbf{\Delta}_{ij}}$ reaching configuration $\mathbf{C}_i = \mathbf{O}_i + \mathbf{D}$, with $\mathbf{C}_i(Y) = \mathbf{O}_i(Y) = f(\mathbf{a}_i)$. Then to stably compute $f(\mathbf{a}_i + \mathbf{\Delta}_{ij})$ there must exist a further sequence of reactions α from $\mathbf{C}_i$ that produce an additional $f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i)$ copies of output Y.

By the same argument, from initial configuration $\mathbf{I}_{\mathbf{a}_j + \mathbf{\Delta}_{ij}}$ the configuration $\mathbf{C}_j = \mathbf{O}_j + \mathbf{D}$ is reachable, with $\mathbf{C}_j(Y) = \mathbf{O}_j(Y) = f(\mathbf{a}_j)$. Then since $\mathbf{O}_i \leq \mathbf{O}_j$, we have $\mathbf{C}_i \leq \mathbf{C}_j$, so the same sequence of reactions α is applicable to $\mathbf{C}_j$, reaching some configuration $\mathbf{C}'_j$ with an additional $f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i)$ copies of output Y, so

$$\mathbf{C}'_j(Y) = f(\mathbf{a}_j) + f(\mathbf{a}_i + \mathbf{\Delta}_{ij}) - f(\mathbf{a}_i) > f(\mathbf{a}_j + \mathbf{\Delta}_{ij})$$

Then $\mathbf{I}_{\mathbf{a}_j + \mathbf{\Delta}_{ij}} \to^* \mathbf{C}'_j$ overproduces Y, so the output-oblivious CRN C cannot stably compute $f(\mathbf{a}_j + \mathbf{\Delta}_{ij})$. □

Lemma 4.1 is our main technical tool used to show that a particular semilinear, nondecreasing function is not obliviously-computable, the key challenge in the impossibility direction of Theorem 5.2.

5 MAIN RESULT: FULL-DIMENSIONAL CASE

To formally state our main result, Theorem 5.2, we must first define *quilt-affine* functions as the sum of a linear and periodic function (see Fig. 3b):

Definition 5.1. *A nondecreasing function $g : \mathbb{N}^d \to \mathbb{Z}$ is quilt-affine (with period p) if there exists $\nabla_g \in \mathbb{Q}^d_{\geq 0}$ and $B : \mathbb{Z}^d / p\mathbb{Z}^d \to \mathbb{Q}$ such that $g(\mathbf{x}) = \nabla_g \cdot \mathbf{x} + B(\overline{\mathbf{x}} \mod p)$.*

We call ∇_g the *gradient* of g, and the periodic function B the *periodic offset*. Without loss of generality we have the same period p along all inputs, since p could be the least common multiple of the periods along each input component. Note that $\nabla_g \cdot \mathbf{x}$ and B can each be rational, but the sum $g(x) \in \mathbb{Z}$ will be integer-valued. We allow g to have negative output for technical reasons[10], but in the case that g is quilt-affine with nonnegative output (i.e. $g : \mathbb{N}^d \to \mathbb{N}$), there is a simple output-oblivious CRN construction to stably compute g. The intuitive idea is to use a single leader that reacts with every input species sequentially, tracks the periodic value $\overline{\mathbf{x}} \mod p$, and outputs the correct changes in g (Lemma 6.1).

Our main result has a recursive condition where we fix the input of a function $f : \mathbb{N}^d \to \mathbb{N}$. For each $i = 1, \ldots, d$ and $j \in \mathbb{N}$, define the *fixed-input restriction*[11] $f_{[\mathbf{x}(i) \to j]} : \mathbb{N}^d \to \mathbb{N}$ of f for all $\mathbf{x} \in \mathbb{N}^d$ by $f_{[\mathbf{x}(i) \to j]}(\mathbf{x}) = f(\mathbf{x}(1), \ldots, \mathbf{x}(i-1), j, \mathbf{x}(i+1), \ldots, \mathbf{x}(d))$.

We can now formally state our main result:

Theorem 5.2. *$f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable $\iff$*

i) [nondecreasing] f is nondecreasing,
ii) [eventually-min] there exist quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ such that for all $\mathbf{x} \geq \mathbf{n}$, $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$, and
iii) [recursive] all fixed-input restrictions $f_{[\mathbf{x}(i) \to j]}$ are obliviously-computable.

We first prove that these conditions imply f is obliviously-computable via a general CRN construction in Section 6.

The nondecreasing condition (i) is necessary by Observation 2.1. It is immediate to see the recursive condition (iii) is also necessary:

Observation 5.3. *If $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, then any fixed-input restriction $f_{[\mathbf{x}(i) \to j]} : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable.*

[10] The quilt-affine functions that describe f for large inputs may be negative on inputs close to the origin.
[11] We define $f_{[\mathbf{x}(i) \to j]}$ to have domain $\mathbb{N}^d$ because it is notationally convenient to have the same domain as f, but $f_{[\mathbf{x}(i) \to j]}$ only has relevant input in $d - 1$ of its input components, making condition (iii) recursive.

Proof. Let the output-oblivious CRN C stably compute f. We define the output-oblivious CRN C' to "hardcode" the input $\mathbf{x}(i) = j$ by modifying the reactions of C. Replace all instances of the leader L and input species X_i by L' and X_i' respectively, then add the initial reaction $L \to jX_i' + L'$. It is straightforward to verify that C' stably computes $f_{[\mathbf{x}(i)\to j]}$. □

Then the remaining work (and biggest effort of this paper) is to show the necessity of the eventually-min condition (ii): that every obliviously-computable function can be represented as eventually a minimum of a finite number of quilt-affine functions, which is shown as Theorem 7.1. Its proof relies on f being semilinear, nondecreasing, and not having any "contradiction sequences" to apply Lemma 4.1. Thus the proof of Theorem 7.1 also yields the following alternative characterization to Theorem 5.2:

Theorem 5.4. *$f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable $\iff$ f is semilinear, nondecreasing, and has no sequence $(\mathbf{a}_1, \mathbf{a}_2, \ldots)$ meeting the conditions of Lemma 4.1.*

This gives a "negative characterization" identifying behavior obliviously-computable functions must *avoid*, whereas Theorem 5.2, is a "positive characterization" describing the allowable behavior of such functions. We include Theorem 5.4, though it is less descriptive of the function, because it may be useful in other contexts.

6 CONSTRUCTION

First we show that any quilt-affine function with nonnegative range is stably computed by an output-oblivious CRN:

Lemma 6.1. *Every quilt-affine function $g : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable.*

Proof. Let $g : \mathbb{N}^d \to \mathbb{N}$ be quilt-affine with period p (recalling Definition 5.1). Notice that g has periodic finite differences. For each congruence class $\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$ and input component $i = 1, \ldots, d$, where $\mathbf{e}_i$ is the ith standard basis vector, define

$$\delta^i_{\overline{\mathbf{a}}} = \nabla_g \cdot \mathbf{e}_i + B(\overline{\mathbf{a} + \mathbf{e}_i} \mod p) - B(\overline{\mathbf{a}} \mod p) \in \mathbb{N}.$$

Observe that for all $\mathbf{x} \in \overline{\mathbf{a}}$, $g(\mathbf{x} + \mathbf{e}_i) - g(\mathbf{x}) = \delta^i_{\overline{\mathbf{a}}}$. We now use these periodic finite differences to construct an output-oblivious CRN C to stably compute g.

The CRN C has input species $X_1, \ldots, X_d$, output species Y, leader species L and p^d additional species $L_{\overline{\mathbf{a}}}$ for each $\overline{\mathbf{a}} \in \mathbb{Z}^d/p\mathbb{Z}^d$ coresponding to auxilliary "states" of the leader. The initial reaction $L \to g(\mathbf{0})Y + L_{\overline{\mathbf{0}}}$ is accompanied by dp^d reactions of the form

$$L_{\overline{\mathbf{a}}} + X_i \to \delta^i_{\overline{\mathbf{a}}} Y + L_{\overline{\mathbf{a}+\mathbf{e}_i}}$$

for each $i = 1, \ldots, d$ and $\overline{\mathbf{a}} \in \mathbb{Z}^d/P\mathbb{Z}^d$. This CRN first creates $g(\mathbf{0})$ output, then sequentially outputs all finite differences, and is easily verified to stably compute g. □

We now prove (in Lemma 6.2) one direction of Theorem 5.2: that conditions (i), (ii), and (iii) imply an output-oblivious CRN can stably compute f. Intuitively, by the eventually-min condition (ii) we compute $f(\mathbf{x})$ for $\mathbf{x} \geq \mathbf{n}$ by composing min and quilt-affine functions. If $\mathbf{x} \not\geq \mathbf{n}$, then $\mathbf{x}(i) = j$ for some input i and $j < \mathbf{n}(i)$. By the recursive condition (iii) we compute $f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) = f(\mathbf{x})$ [12]. The key remaining insight is a trick (similar to a proof in [9]) to compose these pieces using minimum and indicator functions.

The proof of Lemma 6.2 then expresses f as such a minimum of finitely many pieces. We justify that f is obliviously-computable by showing that each piece is obliviously-computable, since by Observation 2.2 obliviously-computable functions are closed under composition.

Lemma 6.2. *If $f : \mathbb{N}^d \to \mathbb{N}$ satisfies the conditions of Theorem 5.2, f is obliviously-computable.*

Proof. Assume $f : \mathbb{N}^d \to \mathbb{N}$ satisfies the conditions of Theorem 5.2. Then by eventually-min condition (ii), there exist quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ (without loss of generality assume $\mathbf{n} = (n, \ldots, n)$) such that $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$ for all $\mathbf{x} \geq \mathbf{n}$.

Let $\mathbf{x} \vee \mathbf{n} = (\max(\mathbf{x}(1), n), \ldots, \max(\mathbf{x}(d), n))$ denote the componentwise max of $\mathbf{x}$ and $\mathbf{n}$. Let $\mathbb{1}_{\{\mathbf{x}(i)>j\}} : \mathbb{N}^d \to \{0, 1\}$ denote the indicator function that is $1 \iff$ its input $\mathbf{x}$ obeys $\mathbf{x}(i) > j$. Recall $f_{[\mathbf{x}(i)\to j]}$ is the fixed-input restriction setting input $\mathbf{x}(i) = j$. We claim that f can be expressed as

$$f(\mathbf{x}) = \min\left[f(\mathbf{x} \vee \mathbf{n}), \underbrace{f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot f(\mathbf{x} \vee \mathbf{n})}_{\substack{i=1,\ldots,d \\ j=0,\ldots,n-1}}\right]. \quad (1)$$

We first show $f \geq \min[\ldots]$ since for all $\mathbf{x} \in \mathbb{N}^d$, $f(\mathbf{x})$ is achieved by some term. If $\mathbf{x} \geq \mathbf{n}$, then $f(\mathbf{x}) = f(\mathbf{x} \vee \mathbf{n})$. If $\mathbf{x} \not\geq \mathbf{n}$, there must be $\mathbf{x}(i) = j$ for some $i = 1, \ldots, d$ and $j = 0, \ldots, n-1$, so $f(\mathbf{x}) = f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) = f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot f(\mathbf{x} \vee \mathbf{n})$ since the indicator is 0.

We next show $f \leq \min[\ldots]$ since $f(\mathbf{x}) \leq$ each term for all $\mathbf{x} \in \mathbb{N}^d$. $f(\mathbf{x}) \leq f(\mathbf{x} \vee \mathbf{n})$ since $\mathbf{x} \leq (\mathbf{x} \vee \mathbf{n})$ and f is nondecreasing. When $\mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) = 1$, we then have $f(\mathbf{x}) \leq f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot f(\mathbf{x} \vee \mathbf{n})$. If $\mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) = 0$, then $\mathbf{x}(i) \leq j$ so $f(\mathbf{x}) \leq f_{[\mathbf{x}(i)\to j]}(\mathbf{x})$ since f is nondecreasing. Thus equation 1 holds as claimed.

It remains to show that f is obliviously-computable. From Observation 2.2, output-oblivious CRNs are closed under composition, and equation 1 gives a method to express f as a composition of functions. Thus it suffices to show that each piece is obliviously-computable. Specifically, we show the functions $\min : \mathbb{N}^k \to \mathbb{N}$ (for any k), $f(\mathbf{x} \vee \mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$, $f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) : \mathbb{N}^d \to \mathbb{N}$, and $c(a, b, \mathbf{x}) = a + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot b : \mathbb{N}^{d+2} \to \mathbb{N}$ are each obliviously-computable. Implicit in the composed CRN to stably compute f as the composition from equation 1 is the "fan out" operation where reactions of the form $X_i \to X_i^1, \ldots, X_i^m$ create multiple copies of species X_i to be used as independent inputs to multiple "modules" in this composition.

- $\min : \mathbb{N}^k \to \mathbb{N}$ **is obliviously-computable:**
  Consider the CRN with single reaction $X_1, \ldots, X_k \to Y$, the natural generalization of two-input min from Fig. 1.
- $f(\mathbf{x} \vee \mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$ **is obliviously-computable:**
  By condition (ii), $f(\mathbf{x} \vee \mathbf{n}) = \min_k(g_k(\mathbf{x} \vee \mathbf{n}))$ since $\mathbf{x} \vee \mathbf{n} \geq \mathbf{n}$, so it suffices to show for each quilt-affine $g_k : \mathbb{N}^d \to \mathbb{Z}$ that $g_k(\mathbf{x} \vee \mathbf{n})$ is obliviously-computable.

[12] As a result, this construction is recursive, with an additional input being fixed at each level of the recursion, so the base case is simply a constant function.

By condition (ii), $g_k(\mathbf{x}+\mathbf{n}) \geq f(\mathbf{x}+\mathbf{n}) \geq 0$ since $\mathbf{x}+\mathbf{n} \geq \mathbf{n}$. Then $g_k(\mathbf{x}+\mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$ is still quilt-affine since that property is preserved by translation, but now has guaranteed nonnegative output. Thus by Lemma 6.1, $g_k(\mathbf{x}+\mathbf{n}) : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable.

Letting $(\mathbf{x}-\mathbf{n})_+ = (\max(\mathbf{x}(1)-n, 0), \ldots, \max(\mathbf{x}(d)-n, 0))$, we then show the function $(\mathbf{x}-\mathbf{n})_+ : \mathbb{N}^d \to \mathbb{N}^d$ is obliviously-computable via the CRN with reactions $(n+1)X_i \to nX_i + Y_i$ for each component $i = 1, \ldots, d$.

Finally, because $\mathbf{x} \vee \mathbf{n} = (\mathbf{x}-\mathbf{n})_+ + \mathbf{n}$, we have shown $g_k(\mathbf{x} \vee \mathbf{n}) = g_k((\mathbf{x}-\mathbf{n})_+ + \mathbf{n})$ is obliviously-computable as the composition of obliviously-computable $g_k(\mathbf{x}+\mathbf{n})$ and $(\mathbf{x}-\mathbf{n})_+$.

$f_{[\mathbf{x}(i)\to j]}(\mathbf{x}) : \mathbb{N}^d \to \mathbb{N}$ **is obliviously-computable:**
This is precisely the assumed recursive condition (iii).

$c(a, b, \mathbf{x}) = a + \mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) \cdot b : \mathbb{N}^{d+2} \to \mathbb{N}$ **is obliviously-computable:**
Consider the output-oblivious CRN (with input species $A, B, X_1, \ldots, X_d$ and output species Y) with two reactions $A \to Y$ and $(j+1)X_i + B \to (j+1)X_i + Y$. The A is all converted to Y, and $(j+1)$ copies of input species X_i catalyze the conversion of B to Y, which will only happen when $\mathbb{1}_{\{\mathbf{x}(i)>j\}}(\mathbf{x}) = 1$. Thus this stably computes $c(a, b, \mathbf{x})$ as desired. □

7 OUTPUT-OBLIVIOUS IMPLIES EVENTUALLY MIN OF QUILT-AFFINE FUNCTIONS

To complete the proof of Theorem 5.2, it remains to show the necessity of the eventually-min condition (ii):

Theorem 7.1. *If $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, then there exist quilt-affine $g_1, \ldots, g_m : \mathbb{N}^d \to \mathbb{Z}$ and $\mathbf{n} \in \mathbb{N}^d$ such that for all $\mathbf{x} \geq \mathbf{n}$, $f(\mathbf{x}) = \min_k(g_k(\mathbf{x}))$.*

For the remainder of Section 7, we fix an obliviously-computable $f : \mathbb{N}^d \to \mathbb{N}$, and Section 7 is devoted to finding $g_1, \ldots, g_m$ and $\mathbf{n}$ satisfying Theorem 7.1.

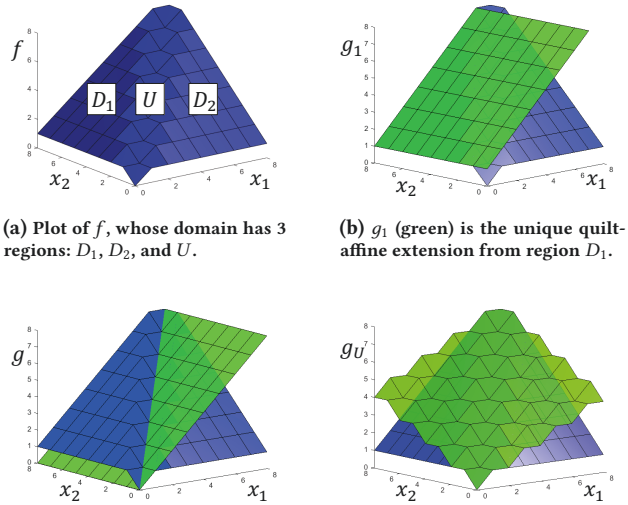
7.1 Proof outline

Since $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, f is semilinear (recall Definition 2.5), and we first consider all threshold sets used to define the semilinear domains of the affine partial functions that define f. Each threshold set defines a hyperplane, and we use these hyperplanes to define *regions* (see Fig. 8a and Fig. 8c). We consider regions as subsets of $\mathbb{R}^d_{\geq 0}$, so they are convex polyhedra with useful geometric properties.[13]

The regions partition [14] the points in the domain $\mathbb{N}^d$. To prove Theorem 7.1, for each region R_k we will identify a quilt-affine function g_k (the *extension* of f from region R) such that $g(\mathbf{x}) = f(\mathbf{x})$ for all integer $\mathbf{x} \in R$. To ensure $f = \min_k(g_k)$, we further require that these quilt-affine extensions *eventually dominate* f (each $g_k(\mathbf{x}) \geq f(\mathbf{x})$ for sufficiently large $\mathbf{x}$). Also, because we only care about sufficiently large $\mathbf{x}$, we need only consider *eventual* regions which are unbounded in all inputs (for example regions 3,4, and 5 in Fig. 8a).

[13] What we consider is a restricted case of a *hyperplane arrangement* [24], with well-studied combinatorial properties.

[14] Without loss of generality, we assume that the hyperplanes do not intersect $\mathbb{N}^d$, so that the partition is well-defined (see Fig. 8a).



(a) Plot of f, whose domain has 3 regions: D_1, D_2, and U.

(b) g_1 (green) is the unique quilt-affine extension from region D_1.

(c) g (green) is a quilt-affine extension from U, but $g < f$ on D_1.

(d) g_U (green) is a quilt-affine extension from U, and $g_U \geq f$.

Figure 7: Obliviously-computable f can be expressed as a min of quilt-affine functions.

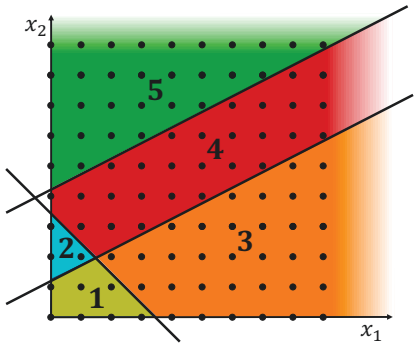
As a simple motivating example, consider the semilinear, nondecreasing function

$$f(x_1, x_2) = \begin{cases} x_1 + 1, & \text{if } x_1 < x_2 \text{ (region } D_1\text{)} \\ x_2 + 1, & \text{if } x_1 > x_2 \text{ (region } D_2\text{)} \\ x_1 & \text{if } x_1 = x_2 \text{ (region } U\text{)} \end{cases}$$

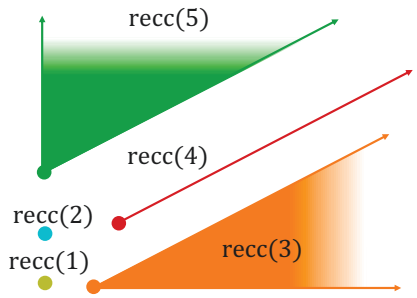
As in Definition 2.5, f is piecewise-affine, with semilinear domains that happen to be only defined by threshold sets. These thresholds then partition the domain into three regions: D_1, D_2, and U (see Fig. 7a). For region D_1, there is a unique quilt-affine extension $g_1(x_1, x_2) = x_1 + 1$ (note an affine function is the special case of a quilt-affine function with period 1). Also, g_1 eventually dominates f as desired, since $g_1(\mathbf{x}) \geq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{N}^2$ (see Fig. 7b). By symmetry, we have the same for region D_2 and its extension $g_2(x_1, x_2) = x_2 + 1$.

These desirable properties follow from D_1 and D_2 being "wide" regions that we define to be *determined* (formalized later). On the other hand, U is a "narrow" region that is *under-determined*. As a result, there is not a unique quilt-affine extension from U. For example, $g(x_1, x_2) = x_1$ is a quilt-affine extension, however, we do not have $g(\mathbf{x}) \geq f(\mathbf{x})$ for all sufficiently large $\mathbf{x}$ (see Fig. 7c).

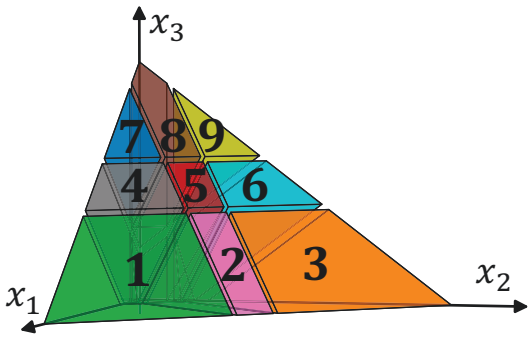
In order to identify a quilt-affine extension from U that does eventually dominate f, we will refer to the unique extensions g_1 and g_2 from regions D_1 and D_2, which are *neighbors* of U (formalized later). We can construct a quilt-affine function with a gradient $(\frac{1}{2}, \frac{1}{2})$ that is the average of the gradients $(1, 0)$ of g_1 and $(0, 1)$ of g_2. In particular, we can let $g_U(x_1, x_2) = \lceil \frac{x_1+x_2}{2} \rceil$ (note this is a quilt-affine function with period 2, see Fig. 7d). We then have $f(\mathbf{x}) = \min[g_1(\mathbf{x}), g_2(\mathbf{x}), g_U(\mathbf{x})]$ for all $\mathbf{x} \geq \mathbf{n} = \mathbf{0}$ as guaranteed by Theorem 7.1.



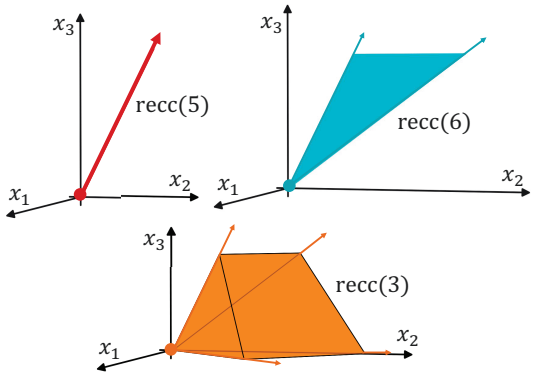
(a) Three threshold hyperplanes creating five regions. Regions 3 and 5 are determined, region 4 is under-determined but still eventual (unbounded in all input).



(b) The recession cones of all five regions. For finite regions, recc(1) = recc(2) = {0}**. Under-determined region 4 has a 1D recession cone, determined regions 3 and 5 have 2D recession cones.**



(c) Two pairs of parallel threshold hyperplanes creating nine eventual regions. Regions 1,3,7,9 are determined. Region 5 is under-determined with 1D recession cone. Regions 2,4,6,8 are under-determined with 2D recession cones.



(d) recc(5) ⊆ recc(6) ⊆ recc(3) **so region 3 is a determined neighbor of under-determined region 5 and under-determined region 6. Also, region 6 is a neighbor of region 5.**

Figure 8: Examples with domains $\mathbb{N}^2$ (top) and $\mathbb{N}^3$ (bottom), with threshold hyperplanes giving regions (left), which are classified by their recession cones (right).

We now describe how we formalize the notion of a *determined region*, *under-determined region*, and *neighbor*, for the general case of domain $\mathbb{N}^d$, where the regions are convex polyhedra in $\mathbb{R}^d$.

To formally define determined regions, we identify the *recession cone* $\text{recc}(R) \subseteq \mathbb{R}^d$ of each region R: the set of vectors along infinite rays in R [20] (see Fig. 8b and Fig. 8d). A determined region D is defined as having a d-dimensional recession cone (see regions 3 and 5 in Fig. 8a and regions 1,3,7,9 in Fig. 8c). For determined regions, we can prove there is a unique quilt-affine extension, which eventually dominates f.

Under-determined regions are then defined as having a recession cone with dimension $< d$ (see regions 1,2,4 in Fig. 8a and regions 2,4,5,6,8 in Fig. 8c). The above arguments do not work for under-determined regions. Instead, identify the *neighbors* of an under-determined region U as regions R with $\text{recc}(U) \subseteq \text{recc}(R)$ (see Fig. 8b and Fig. 8d). We consider the neighbors of U that are determined regions. The possible behavior of f on U is constrained by the unique extensions from these regions, and we can define an extension from U based on an averaging process. Formal definitions and a proof of Theorem 7.1 appear in [21].

8 COMPARISON TO CONTINUOUS CASE

In [9], the authors classified the power of output-oblivious *continuous* CRNs to stably compute real-valued functions $f : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$. We can generalize to also consider such functions by introducing the following natural scaling:

Definition 8.1. *For a function $f : \mathbb{N}^d \to \mathbb{N}$, the* ∞-scaling $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ *is given by* $\hat{f}(\mathbf{x}) = \lim_{c \to \infty} \frac{f(\lfloor c\mathbf{x} \rfloor)}{c}$.

Note this limit may not exist for arbitrary $f : \mathbb{N}^d \to \mathbb{N}$, but it will exist for all obliviously-computable f.

The next theorem shows that in this scaling limit, our output-oblivious function class exactly corresponds to the real-valued function class from [9] (see Fig. 4b). The proof appears in [21].

Theorem 8.2. *If $f : \mathbb{N}^d \to \mathbb{N}$ is obliviously-computable, then the ∞-scaling $\hat{f} : \mathbb{R}^d_{\geq 0} \to \mathbb{R}_{\geq 0}$ is obliviously-computable by a continuous CRN. Furthermore, every function obliviously-computable by a continuous CRN is the ∞-scaling of some function obliviously-computable by a discrete CRN.*

9 LEADERLESS ONE-DIMENSIONAL CASE

In this section we show a characterization of 1D functions $f : \mathbb{N} \to \mathbb{N}$ that are obliviously-computable *without* a leader. The general case for leaderless oblivious computation in higher dimensions remains open.

Note that the following observation applies to any number of dimensions. We say $f : \mathbb{N}^d \to \mathbb{N}$ is *superadditive* if $f(\mathbf{x}) + f(\mathbf{y}) \leq f(\mathbf{x} + \mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{N}^d$.

Observation 9.1. *Every f obliviously-computable by a leaderless CRN is superadditive.*

Proof. Let C be a leaderless CRN stably computing f. We prove the observation by contrapositive. Suppose f is not superadditive. Then there are $\mathbf{x}, \mathbf{z} \in \mathbb{N}^d$ such that $f(\mathbf{x}) + f(\mathbf{z}) > f(\mathbf{x} + \mathbf{z})$. Recall $\mathbf{I_w}$ is the initial configuration of C representing input $\mathbf{w}$. Let $\alpha_\mathbf{x}$ be a sequence of reactions applied to $\mathbf{I_x}$ to produce $f(\mathbf{x})$ copies of Y, and let $\alpha_\mathbf{z}$ be a sequence of reactions applied to $\mathbf{I_z}$ to produce $f(\mathbf{z})$ copies of Y.

Since C is leaderless, $\mathbf{I_{x+z}} = \mathbf{I_x} + \mathbf{I_z}$. Thus we can apply $\alpha_\mathbf{x}$ to $\mathbf{I_{x+z}}$, followed by $\alpha_\mathbf{z}$, producing $f(\mathbf{x}) + f(\mathbf{z})$ copies of Y. Since this is greater than $f(\mathbf{x} + \mathbf{z})$, to stably compute f, C must have a reaction consuming Y, so it is not output-oblivious. Since C was arbitrary, f cannot be obliviously-computable. □

This added condition of superadditivity gives us the 1D leaderless characterization. The proof appears in [21].

Theorem 9.2. *For any $f : \mathbb{N} \to \mathbb{N}$, f is obliviously-computable by a leaderless CRN $\iff$ f is semilinear and superadditive.*

10 CONCLUSION

An obvious question is the computational power of output-oblivious CRNs without an initial leader. A leaderlessly-obliviously-computable function must be superadditive, which is a strictly stronger condition than being nondecreasing. The continuous result [9] had the same restriction of superadditivity, so our "scaling limit" reduction to their function class (Theorem 8.2) shows our main function class is already "almost superadditive." We also showed in the 1D case, $f : \mathbb{N} \to \mathbb{N}$ is leaderlessly-obliviously-computable if and only if f is semilinear and superadditive (Theorem 9.2).

Does adding the additional constraint of superaddivity to our full result (Theorem 5.2) classify leaderlessly-obliviously-computable $f : \mathbb{N}^d \to \mathbb{N}$? If this were true, a proof would require modifying our construction (Section 6) to eliminate the leader L. We successfully modified the 1D construction (Theorem 3.1) to remove the leader in proving Theorem 9.2, but it has been difficult to extend the same ideas to our much more complicated general construction.

An initial leader can also help make computation faster [5, 7, 19]. Many recent results in population protocols have shown time upper and lower bounds for computational tasks such as leader election and function/predicate computation [1, 2, 7, 16, 17, 19]. These techniques, however, are not at all designed to handle the constraint of output-obliviousness. It would be interesting to study how this constraint affects the time required for computation.

Acknowledgements. We thank Anne Condon, Cameron Chalk, Niels Kornerup, Wyatt Reeves, and David Soloveichik for discussing their related work with us and contributing early ideas.

REFERENCES

[1] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in molecular computation. In *SODA 2017: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.
[2] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2221–2239. SIAM, 2018.
[3] 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.
[4] 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.
[5] 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.
[6] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
[7] Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 141:1–141:14, 2017.
[8] Luca Cardelli. Strand algebras for DNA computing. *Natural Computing*, 10(1):407–428, 2011.
[9] Cameron Chalk, Niels Kornerup, Wyatt Reeves, and David Soloveichik. Composable rate-independent computation in continuous chemical reaction networks. In *Computational Methods in Systems Biology*, 2018.
[10] Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. Preliminary version appeared in DNA 2012.
[11] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8(10):755–762, 2013.
[12] Ben Chugg, Anne Condon, and Hooman Hashemi. Output-oblivious stochastic chemical reaction networks. In *OPODIS 2018: Proceedings of the 22nd International Conference on Principles of Distributed Systems*, 2018.
[13] Leonard E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913.
[14] David Doty and Mahsa Eftekhari. Efficient size estimation and impossibility of termination in uniform dense population protocols. In *PODC 2019: Proceedings of the 38th ACM Symposium on Principles of Distributed Computing*, 2019. to appear.
[15] David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. *Natural Computing*, 14(2):213–223, 2015. Preliminary version appeared in DNA 2013.
[16] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018. Special issue of invited papers from DISC 2015.
[17] Leszek Gasieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2653–2667, 2018.
[18] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
[19] Adrian Kosowski and Przemyslaw Uznanski. Brief announcement: Population protocols are fast. In *PODC 2018: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 475–477. ACM, 2018.
[20] Tyson Rockafeller. *Convex Analysis*, chapter 8. Princeton University Press, 1970.
[21] Eric Severson, David Haley, and David Doty. Composable computation in discrete chemical reaction networks. Technical Report 1602.08032, arXiv, 2019. http://arxiv.org/abs/1903.02637.
[22] 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.
[23] Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358(6369):eaal2052, 2017.
[24] Richard Stanley. An introduction to hyperplane arrangments, 2006. URL: http://www.cis.upenn.edu/~cis610/sp06stanley.pdf.