Composable Rate-Independent Computation in Continuous Chemical Reaction Networks

Cameron Chalk*, Niels Kornerup, Wyatt Reeves, David Soloveichik*

The University of Texas at Austin

Abstract. Biological regulatory networks depend upon chemical interactions to process information. Engineering such molecular computing systems is a major challenge for synthetic biology and related fields. The chemical reaction network (CRN) model idealizes chemical interactions, abstracting away specifics of the molecular implementation, and allowing rigorous reasoning about the computational power of chemical kinetics. Here we focus on function computation with CRNs, where we think of the initial concentrations of some species as the input and the eventual steady-state concentration of another species as the output. Specifically, we are concerned with CRNs that are rate-independent (the computation must be correct independent of the reaction rate law) and composable ($f \circ g$ can be computed by concatenating the CRNs computing f and g). Rate independence and composability are important engineering desiderata, permitting implementations that violate mass-action kinetics, or even "well-mixedness", and allowing the systematic construction of complex computation via modular design. We show that to construct composable rate-independent CRNs, it is necessary and sufficient to ensure that the output species of a module is not a reactant in any reaction within the module. We then exactly characterize the functions computable by such CRNs as superadditive, positive-continuous, and piecewise rational linear. Our results show that composability severely limits rate-independent computation unless more sophisticated input/output encodings are used.

1 Introduction

A ubiquitous form of biological information processing occurs in complex chemical regulatory networks in cells. The formalism of chemical reaction networks (CRNs) has been widely used for modelling the interactions underlying such natural chemical computation. More recently CRNs have also become a useful model for designing synthetic molecular computation. In particular, DNA strand displacement cascades can in principle realize arbitrary CRNs, thus motivating the study of CRNs as a programming language [2, 5, 11]. The applications of synthetic chemical computation include reprogramming biological regulatory networks, as well as embedding control modules in environments that are inherently

* These authors' work was supported in part by National Science Foundation grants CCF-1618895 and CCF-1652824.

incompatible with traditional electronic controllers for biochemical, nanotechnological, or medical applications.

The study of information processing within biological CRNs, as well the engineering of CRN functionality in artificial systems, motivates the exploration of the computational power of CRNs. In general, CRNs are capable of Turing universal computation [7]; however, we are often interested in restricted classes of CRNs which may have certain desired properties. Previous work distinguished two programmable features of CRNs: the stoichiometry of the reactions and the rate laws governing the reaction speeds [4]. As an example of computation by stoichiometry alone, consider the reaction $2X \to Y$. We can think of the concentrations of species X and Y to be the input and output, respectively. Then this reaction effectively computes $f(X) = \frac{X}{2}$, as in the limit of time going to infinity, the system converges to producing one unit of Y for every two units of X initially present. The reason we are interested in computation via stoichiometry is that it is fundamentally *rate-independent*, requiring no assumptions on the rate law (e.g., that the reaction occurs at a rate proportional to the product of the concentrations of the reactants). This allows the computation to be correct independent of experimental conditions such as temperature, chemical background, or whether or not the solution is well-mixed.

Computation does not happen in isolation. In an embedded chemical controller, inputs would be produced by other chemical systems, and outputs would affect downstream chemical processes. Composition is easy in some systems (e.g. digital electronic circuits can be composed by wiring the outputs of one to the inputs of the other). However, in other contexts composition presents a host of problems. For example, the effect termed retroactivity, which results in insufficient isolation of modules, has been the subject of much research in synthetic biology [6]. In this paper, we attempt to capture a natural notion of composable rate-independent computation, and study whether composability restricts computational power.

$$X_1 + X_2 \to Y$$

(a)

$$\begin{aligned} X_1 &\to Z_1 + Y \\ X_2 &\to Z_2 + Y \\ Z_1 + Z_2 &\to K \\ Y + K &\to \varnothing \end{aligned}$$

(b)

Above, we see two examples of rate-independent computation. Example **(a)** shows $y = \min(x_1, x_2)$. The amount of Y eventually produced will be the minimum of the initial amounts of X_1 and X_2, since the reaction will stop as soon as the first reactant runs out. Example **(b)** shows $y = \max(x_1, x_2)$. The amount of Y eventually produced in reactions 1 and 2 is the sum of the initial amounts of X_1 and X_2. The amount of K eventually produced in reaction 3 is the minimum of the initial amounts of X_1 and X_2. Reaction 4 subtracts the minimum from the sum, yielding the maximum.

Now consider how rate-independent computation can be naturally composed. Suppose we want to compute $\min(\min(x_1, x_2), x_3)$. It is easy to see that simple concatenation of two min modules (with proper renaming of the species) correctly computes this function:

$$\begin{aligned} X_1 + X_2 &\to Y \\ Y + X_3 &\to Y' \end{aligned}$$

where Y' represents the output of the composed computation. In contrast, suppose we want to compute $\min(\max(x_1, x_2), x_3)$. Concatenating the modules yields:

$$\begin{aligned} X_1 &\to Z_1 + Y \\ X_2 &\to Z_2 + Y \\ Z_1 + Z_2 &\to K \\ Y + K &\to \varnothing \\ Y + X_3 &\to Y' \end{aligned}$$

where Y' represents the output of the composed computation. Observe that depending on the relative rates of reactions 4 and 5, the eventual value of Y' will vary between $\min(\max(x_1, x_2), x_3)$ and $\min(x_1+x_2, x_3)$, and the composition does not compute in a rate-independent manner.

Why is min composable, but max not? The problem arose because the output of the max module (Y) is consumed in both the max module and in the downstream module (min). This creates a competition between the consumption of the output within its own module and the downstream module.

Towards modularity, we assume the two CRNs to be composed do not share any species apart from the interface between them (i.e., a species Y representing the output of the first network is used as the species representing the input to the second network, and otherwise the two sets of species are disjoint). We prove that to construct composable rate-independent modules in this manner, it is necessary and sufficient to ensure that the output species of a module is not a reactant in any reaction of that module. We then exactly characterize the computational power of composable rate-independent computation.

Previously it was shown that without the composability restriction, rate-independent CRNs can compute arbitrary positive-continuous, piecewise rational linear functions [4]. Positive-continuity means that the only discontinuities occur when some input goes from 0 to positive, and piecewise rational linear means that the function can by defined by a finite number of linear pieces (with rational coefficients). Note that non-linear continuous functions can be approximated to arbitrary accuracy.[1] We show that requiring the CRN to be composable restricts

[1] To approximate arbitrary continuous non-linear functions, piecewise linear functions are not sufficient, but rather we need piecewise affine functions (linear functions with offset). However, affine functions can be computed if we use an additional input fixed at 1.

the class of computable functions to be superadditive functions; i.e., functions that satisfy: for all input vectors $\boldsymbol{a}, \boldsymbol{b}$, $f(\boldsymbol{a}) + f(\boldsymbol{b}) \leq f(\boldsymbol{a} + \boldsymbol{b})$. This strongly restricts computational power: for example, subtraction or max cannot be computed or approximated in any reasonable sense. In the positive direction, we show that any superadditive, positive-continuous, piecewise rational linear function can be computed by composable CRNs in a rate-independent manner. Our proof is constructive, and we further show that unimolecular and bimolecular reactions are sufficient.

We note that different input and output encodings can change the computational power of rate-independent, composable CRNs. For example, in the so-called *dual-rail* convention, input and output values are represented by differences in concentrations of two species (e.g., the output is equal to the concentration of species Y^+ minus the concentration of Y^-). Dual-rail simplifies composition—instead of consuming the output species to decrease the output value, a dual-rail CRN can produce Y^-—at the cost of greater system complexity. Dual-rail CRNs can compute the full class of continuous, piecewise rational linear functions while satisfying rate-independence and composability [4]. Note, however, that the dual-rail convention moves the non-superadditive subtraction operation to "outside" the system, and converting from a dual-rail output to a direct output must break composability.

2 Preliminaries

Let $\mathbb{N}$ and $\mathbb{R}$ denote the set of nonnegative integers and the set of real numbers, respectively. The set of the first n positive integers is denoted by $[n]$. If $x \in \mathbb{R}$, let $\mathbb{R}_{\geq x} = \{\ x' \in \mathbb{R} \mid x' \geq x\ \}$, and similarly for $\mathbb{R}_{>x}$. If Λ is a finite set (in this paper, of chemical species), we write $\mathbb{R}^\Lambda$ to denote the set of functions $f : \Lambda \to \mathbb{R}$, and similarly for $\mathbb{R}^\Lambda_{\geq 0}$, $\mathbb{N}^\Lambda$, etc. Equivalently, we view an element $\boldsymbol{c} \in A^\Lambda$ as a vector of $|\Lambda|$ elements of A, each coordinate "labeled" by an element of Λ. Given a function $f : A \to B$, we use $f|_C$ to denote the restriction of f to the domain C. We also use the notation $\boldsymbol{c} \upharpoonright \Delta$ to represent $\boldsymbol{c}$ projected onto $\mathbb{R}^\Delta_{\geq 0}$. Thus, $\boldsymbol{c} \upharpoonright \Delta = \mathbf{0}$ iff $(\forall S \in \Delta)\ \boldsymbol{c}(S) = 0$.

2.1 Chemical reaction networks

Given $S \in \Lambda$ and $\boldsymbol{c} \in \mathbb{R}^\Lambda_{\geq 0}$, we refer to $\boldsymbol{c}(S)$ as the *concentration of S in $\boldsymbol{c}$*. For any $\boldsymbol{c} \in \mathbb{R}^\Lambda_{\geq 0}$, let $[\boldsymbol{c}] = \{S \in \Lambda \mid \boldsymbol{c}(S) > 0\}$, the set of species *present* in $\boldsymbol{c}$. If $\Delta \subseteq \Lambda$, we view a vector $\boldsymbol{c} \in \mathbb{R}^\Delta_{\geq 0}$ equivalently as a vector $\boldsymbol{c} \in \mathbb{R}^\Lambda_{\geq 0}$ by assuming $\boldsymbol{c}(S) = 0$ for all $S \in \Lambda \setminus \Delta$.

Given a finite set of chemical species Λ, a *reaction* over Λ is a pair $\alpha = \langle \mathbf{r}, \mathbf{p} \rangle \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda$, specifying the stoichiometry of the reactants and products, respectively.[2] In this paper, we assume that $\mathbf{r} \neq \mathbf{0}$, i.e., we have no reactions of

[2] As we are studying CRNs whose output is independent of the reaction rates, we leave the rate constants out of the definition.

the form $\varnothing \to \ldots$. For instance, given $\Lambda = \{A, B, C\}$, the reaction $A + 2B \to A+3C$ is the pair $\langle (1,2,0), (1,0,3) \rangle$. A *(finite) chemical reaction network (CRN)* is a pair $\mathcal{C} = (\Lambda, R)$, where Λ is a finite set of chemical *species*, and R is a finite set of reactions over Λ. A *state* of a CRN $\mathcal{C} = (\Lambda, R)$ is a vector $\boldsymbol{c} \in \mathbb{R}^{\Lambda}_{\geq 0}$. Given a state $\boldsymbol{c}$ and reaction $\alpha = \langle \mathbf{r}, \mathbf{p} \rangle$, we say that α is *applicable* in $\boldsymbol{c}$ if $[\mathbf{r}] \subseteq [\boldsymbol{c}]$ (i.e., $\boldsymbol{c}$ contains positive concentration of all of the reactants).

2.2 Reachability and stable computation

We now follow [4] in defining rate-independent computation in terms of reachability between states (this treatment is in turn based on the notion of "stable computation" in distributed computing [1]). Intuitively, we say a state is "reachable" if some rate law can take the system to this state. For computation to be rate-independent, since unknown rate laws might take the system to any reachable state, the system must be able to reach the correct output from any such reachable state.

To define the notion of reachability, a key insight of [4] allows one to think of reachability via a sequence of straight line segments. This may be unintuitive, since mass-action[3] and other rate laws trace out smooth curves. However, a number of properties are shown which support straight-line reachability as an interpretation which includes mass-action reachability as well as reachability under other rate laws.

Let $m = |R|$ be the number of reactions in CRN $\mathcal{C}$, and let $n = |\Lambda|$ be the number of species in $\mathcal{C}$. The $n \times m$ *reaction stoichiometry matrix* $\boldsymbol{M}$ is such that $\boldsymbol{M}(i,j)$ is the net amount of the i'th species that is produced by the j'th reaction (negative if the species is consumed). We say state $\boldsymbol{d}$ is *straight-line reachable* from $\boldsymbol{c}$, written $\boldsymbol{c} \to^1 \boldsymbol{d}$, if $(\exists \boldsymbol{u} \in \mathbb{R}^m_{\geq 0})$ $\boldsymbol{c} + \boldsymbol{M}\boldsymbol{u} = \boldsymbol{d}$ and $\boldsymbol{u}(j) > 0$ only if reaction j is applicable at $\boldsymbol{c}$. Intuitively, a single segment means running the reactions applicable at $\boldsymbol{c}$ at a constant (possibly 0) rate to get from $\boldsymbol{c}$ to $\boldsymbol{d}$. We say state $\boldsymbol{d}$ is *l-segment reachable*, if $(\exists \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{l+1})$ $\boldsymbol{c} = \boldsymbol{b}_1 \to^1 \boldsymbol{b}_2 \to^1 \boldsymbol{b}_3 \to^1 \ldots \to^1 \boldsymbol{b}_{l+1} = \boldsymbol{d}$. Generalizing to an arbitrary number of segments, we obtain our general notion of reachability below. Note that by the definition of straight-line reachability, only applicable reactions occur in each segment. The definition of reachability is closely related to exploring the "stoichiometric compatibility class" of the initial state [8].

Definition 1. *State $\boldsymbol{d}$ is* reachable *from $\boldsymbol{c}$, written $\boldsymbol{c} \to \boldsymbol{d}$, if $\exists l \in \mathbb{N}$ such that $\boldsymbol{d}$ is l-segment reachable from $\boldsymbol{c}$ for some $l \in \mathbb{N}$.*

We think of state $\boldsymbol{d}$ as being reachable from state $\boldsymbol{c}$ if there is a "reasonable" rate law that takes the system from $\boldsymbol{c}$ to $\boldsymbol{d}$. Not surprisingly, previous work

[3] Although the formal definition of mass-action kinetics is outside the scope of this paper, we remind the reader that a CRN with rate constants on each reaction define a system of ODEs under mass-action kinetics. For example, the two reactions $A+B \to A+C$ and $C+C \to B$ correspond to the following ODEs: $\dot{a} = 0$, $\dot{b} = k_2 c^2 - k_1 ab$, and $\dot{c} = k_1 ab - 2k_2 c^2$, where a, b, and c are the concentrations of species A, B, and C over time and k_1, k_2 are the rate constants of the reactions.

showed that if state $\boldsymbol{d}$ is reached from $\boldsymbol{c}$ via a mass-action trajectory, it is also segment-reachable.

Lemma 1 (Proven in [4]). *If $\boldsymbol{d}$ is mass-action reachable from $\boldsymbol{c}$, then $\boldsymbol{c} \to \boldsymbol{d}$.*

We can now use reachability to formally define rate-independent computation. Formally, a *chemical reaction computer (CRC)* is a tuple $\mathcal{C} = (\Lambda, R, \Sigma, Y)$, where (Λ, R) is a CRN, $\Sigma \subset \Lambda$, written as $\Sigma = \{X_1, \ldots, X_n\}$, is the *set of input species*, and $Y \in \Lambda \setminus \Sigma$ is the *output species*. For simplicity, assume a canonical ordering of $\Sigma = \{X_1, \ldots, X_n\}$ so that a vector $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ (i.e., an input to f) can be viewed equivalently as a state $\boldsymbol{x} \in \mathbb{R}^{\Sigma}_{\geq 0}$ of $\mathcal{C}$ (i.e., an input to $\mathcal{C}$).

Definition 2. *A state $\boldsymbol{o} \in \mathbb{R}^{\Lambda}_{\geq 0}$ is* output stable *if, for all $\boldsymbol{o}'$ such that $\boldsymbol{o} \to \boldsymbol{o}'$, $\boldsymbol{o}(Y) = \boldsymbol{o}'(Y)$, i.e., once $\boldsymbol{o}$ is reached, no reactions can change the concentration of the output species Y.*

Definition 3. *Let $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a function and let $\mathcal{C}$ be a CRC. We say that $\mathcal{C}$* stably computes *f if, for all $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ and all $\boldsymbol{c}$ such that $\boldsymbol{x} \to \boldsymbol{c}$, there exists an output stable state $\boldsymbol{o}$ such that $\boldsymbol{c} \to \boldsymbol{o}$ and $\boldsymbol{o}(Y) = f(\boldsymbol{x})$.*

The results herein extend easily to functions $f : \mathbb{R}^n \to \mathbb{R}^l$, i.e., whose output is a vector of l real numbers. This is because such a function is equivalently l separate functions $f_i : \mathbb{R}^n \to \mathbb{R}$.

Also note that initial states contain only the input species Σ; other species must have initial concentration 0. We briefly discuss in the conclusion how allowing some initial concentration of non-input species affects computation.

2.3 Composability

We call a CRC $\mathcal{C} = (\Lambda, R, \Sigma, Y)$ *output oblivious* if the output species Y does not appear as a reactant. We now show that an output oblivious CRC is composable. For simplicity, in this section we focus on single-input, single-output CRCs, but our results can be easily generalized to multiple input and output settings.

First, we define the composition of two CRCs as the concatenation of their chemical reactions, such that the output species of the first is the input species of the second:

Definition 4. *Given two CRCs $\mathcal{C}_1 = (\Lambda_1, R_1, \Sigma_1, Y_1)$ and $\mathcal{C}_2 = (\Lambda_2, R_2, \Sigma_2, Y_2)$, consider $\mathcal{C}_2' = (\Lambda_2', R_2', \Sigma_2', Y_2')$ constructed by renaming species of $\mathcal{C}_2$ such that $\Lambda_1 \cap \Lambda_2' = \{Y_1\}$ and $Y_1 \in \Sigma_2'$. The* composition *of $\mathcal{C}_1$ and $\mathcal{C}_2$ is the CRC $\mathcal{C}_{2\circ 1} = (\Lambda_1 \cup \Lambda_2', R_1 \cup R_2', \Sigma_1 \cup \Sigma_2' \setminus \{Y_1\}, Y_2')$. In other words, the composition is constructed by concatenating $\mathcal{C}_1$ and $\mathcal{C}_2$ such that their only interface is the output species of $\mathcal{C}_1$, used as the input for $\mathcal{C}_2$.*

Definition 5. *A CRC $\mathcal{C}_1$ which stably computes f_1 is* composable *iff $\forall \mathcal{C}_2$ stably computing f_2, $\mathcal{C}_{2\circ 1}$ stably computes $f_2 \circ f_1$.*

We first show that output oblivious CRCs are composable. Second, we show that if a CRC is composable then any reactions using the output species as a reactant can be removed without affecting functionality.

Lemma 2. *Output oblivious CRCs are composable.*

Proof. Consider the composition $\mathcal{C}_{2\circ1}$ of two CRCs $\mathcal{C}_1 = (\Lambda_1, R_1, \Sigma_1, Y_1)$ and $\mathcal{C}_2 = (\Lambda_2, R_2, \Sigma_2, Y_2)$ that stably compute f_1 and f_2 respectively, and consider an input $x \in \mathbb{R}_{\geq 0}$. Consider some state $\boldsymbol{c}$ reached by $\mathcal{C}_{2\circ1}$. We want to show that from $\boldsymbol{c}$, we can reach an output stable configuration $\boldsymbol{o}$ s.t. $\boldsymbol{o}(Y_2) = f_2 \circ f_1(x)$. From $\boldsymbol{c}$, first produce the maximal amount of Y_1 possible from the reactions in $\mathcal{C}_1$. Since Y_1 is the only species shared between Λ_1 and Λ_2 and Y_1 is not a reactant in any reaction of R_1, reactions from R_2 do not inhibit reactions from R_1. Thus, from $\boldsymbol{c}$, there is a state $\boldsymbol{c'}$ that is reachable where the reactions in $\mathcal{C}_1$ have produced in total $f_1(x)$ of Y_1. Again since Y_1 is the only species shared between $\mathcal{C}_1$ and $\mathcal{C}_2$, we can now consider $\boldsymbol{c'} \restriction \Lambda_2$, as $\mathcal{C}_1$ cannot produce any more Y_1. Observe that $\boldsymbol{c'} \restriction \Lambda_2$ must have also been reachable in $\mathcal{C}_2$ with an input of $f_1(x)$. We know this is true because if we undid all of the performed reactions in R_2, we would end up with a state where the only compound present in Λ_2 with a positive concentration is $f_1(x)$ of Y_1. Thus, by the definition of stable computation, there is an output stable state $\boldsymbol{o}$ s.t. $\boldsymbol{c'} \restriction \Lambda_2 \to \boldsymbol{o}$ and $\boldsymbol{o}(Y_2) = f_2(f_1(x)) = f_2 \circ f_1(x)$. Since the reactions in $\mathcal{C}_1$ can no longer affect the output of $\mathcal{C}_{2\circ1}$, this is also an output stable state reachable in $\mathcal{C}_{2\circ1}$. Thus, $\mathcal{C}_{2\circ1}$ stably computes $f_2 \circ f_1(x)$. □

Lemma 3. *If a CRC $\mathcal{C}$ stably computes f and is composable, then we can remove all reactions where the output species appears as a reactant, and the resulting output oblivious CRC will still stably compute f.*

Proof. Assume that $\mathcal{C}_1$ is a composable CRC stably computing f with the output species Y_1. Suppose we compose it with $\mathcal{C}_2$, which contains a single reaction $Y_1 \to Y_2$ with Y_2 as the output species of $\mathcal{C}_2$. Since $\mathcal{C}_2$ stably computes the identity function, the resulting CRN $\mathcal{C}_{2\circ1}$ must stably compute f. By the definition of stable computation, from any reachable state $\boldsymbol{c}$ in $\mathcal{C}_{2\circ1}$, we know that there is an output stable state $\boldsymbol{o}$ such that $\boldsymbol{c} \to \boldsymbol{o}$ and $\boldsymbol{o}(Y_2) = f(\boldsymbol{x})$. Thus we can consider any state $\boldsymbol{c'}$ reachable by running the reactions in $\mathcal{C}_1$ that don't use Y_1 as a reactant until no series of those reactions can further increase the concentration of Y_1. From $\boldsymbol{c'}$, run the reaction in $\mathcal{C}_2$ to convert all Y_1 into Y_2 to reach the state $\boldsymbol{c}$. Since we have produced a maximal amount of Y_1 possible in $\mathcal{C}_1$ without using the reactions that involve Y_1 as a reactant, there are no more reactions that can run to produce Y_1. Since the reactions output molecule Y_1 cannot occur (since it has a concentration of zero), we know that $\boldsymbol{c}$ must be an output stable state. Thus we know that $\boldsymbol{c}(Y_2) = f(\boldsymbol{x})$. This means that any trajectory of $\mathcal{C}_1$ that produces a maximal amount of Y_1 without using reactions that involve Y_1 as a reactant must produce an amount of Y_1 equal to $f(\boldsymbol{x})$. Therefore removing all such reactions from $\mathcal{C}_1$ gives us a CRN $\mathcal{C}_1'$ that also stably computes f. □

To allow composition of multiple downstream CRCs, we can use the reaction $Y \to Y_1 + \ldots + Y_n$ to generate n "copies" of the output species Y, such that each downstream module uses a different copy as input. Additionally, if the downstream module is output oblivious, then the composition is also output oblivious and thus the composition is composable. These observations allow complex compositions of modules, and will be used in our constructions in Section 3.2.

3 Functions Computable by Composable CRNs

Here we give a complete characterization of the functions computable by composable CRNs. First, we define exactly our notions of *superadditive*, *positive-continuous*, and *piecewise rational linear*.

Definition 6. *A function $f : \mathbb{R}^n \to \mathbb{R}^l$ is* superadditive *iff $\forall \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n, f(\boldsymbol{a}) + f(\boldsymbol{b}) \leq f(\boldsymbol{a} + \boldsymbol{b})$.*

Note that superadditivity implies monotonicity in our case, since the functions computed must be nonnegative. As an example, we show that the max function is not superadditive:

Lemma 4. *The function $\max(x_1, x_2)$ is not superadditive.*

Proof. Pick any $x_1, x_2 > 0$. Observe that $\max(x_1, 0) + \max(0, x_2) = x_1 + x_2$. But since x_1 and x_2 are both positive, we know that $x_1 + x_2 > \max(x_1, x_2)$. Thus max is not superadditive and by Lemma 6 there is no composable CRN which stably computes max. □

Definition 7. *A function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^l$ is* positive-continuous *if for all $U \subseteq [n]$, f is continuous on the domain $D_U = \{\ x \in \mathbb{R}^n_{\geq 0} \mid (\forall i \in [n]),\ x(i) > 0 \iff i \in U\}$. I.e., f is continuous on any subset $D \subset \mathbb{R}^n_{\geq 0}$ that does not have any coordinate $i \in [n]$ that takes both zero and positive values in D.*

Next we give our definition of piecewise rational linear. One may (and typically does) consider a restriction on the domains selected for the pieces, however this restriction is unneccesary in this work, particularly because the additional constraint of positive-continuity gives enough restriction.

Definition 8. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is* rational linear *if there exists $a_1, \ldots, a_n \in \mathbb{Q}$ such that $f(x) = \sum_{i=1}^n a_i x(i)$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is* piecewise rational linear *if there is a finite set of partial rational linear functions $f_1, \ldots, f_p : \mathbb{R}^n \to \mathbb{R}$ with $\bigcup_{j=1}^p \text{dom } f_j = \mathbb{R}^n$, such that for all $j \in [p]$ and all $x \in \text{dom } f_j$, $f(x) = f_j(x)$. We call $f_1, \ldots, f_p$ the* components *of f.*

The following is an example of a superadditive, positive-continuous, piecewise rational linear function:

$$f(\boldsymbol{x}) = \begin{cases} x_1 + x_2 & x_3 > 0 \\ \min(x_1, x_2) & x_3 = 0 \end{cases} \tag{1}$$

The function is superadditive since for all input vectors $\boldsymbol{a} = (a_1, a_2, a_3)$, $\boldsymbol{b} = (b_1, b_2, b_3)$, there are three cases: **(1)** $a_3 = b_3 = 0$, in which case both input vectors compute min which is a superadditive function; **(2)** $a_3, b_3 \neq 0$, in which case both input vectors compute $x_1 + x_2$, which is a superadditive function; **(3)** without loss of generality, $a_3 = 0$ and $b_3 \neq 0$, in which case $f(\boldsymbol{a}) + f(\boldsymbol{b}) = \min(a_1, a_2) + b_1 + b_2 \leq a_1 + a_2 + b_1 + b_2 = f(\boldsymbol{a} + \boldsymbol{b})$. The function is positive-continuous, since the only points of discontinuity are when x_3 changes from zero to positive. The function is piecewise rational linear, since min is piecewise rational linear.

Theorem 1. *A function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ is computable by a composable CRC if and only if it is superadditive positive-continuous piecewise rational linear.*

We prove each direction of the theorem independently in Sections 3.1 and 3.2.

3.1 Computable Functions are Superadditive Positive-Continuous Piecewise Rational Linear

Here, we prove that a stably computable function must be superadditive positive-continuous piecewise rational linear. The constraints of positive-continuity and piecewise rational linearity stem from previous work:

Lemma 5 (Proven in [4]). *If a function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ is stably computable by a CRC, then f is positive-continuous piecewise rational linear.*

In addition to the constraints in the above lemma, we show in Lemma 6 that a function must be superadditive if it is stably computed by a CRC. To prove this, we first note a useful property of reachability in CRNs.

Claim. Given states $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, if $\boldsymbol{a} \to \boldsymbol{b}$ then $\boldsymbol{a} + \boldsymbol{c} \to \boldsymbol{b} + \boldsymbol{c}$.

This claim comes from the fact that adding species cannot prevent reactions from occurring. Thus, we can consider the series of reactions where $\boldsymbol{c}$ doesn't react to reach the state $\boldsymbol{b} + \boldsymbol{c}$ from the state $\boldsymbol{a} + \boldsymbol{c}$. We now utilize this claim to prove that composably computable functions must be superadditive.

Lemma 6. *If a function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ is stably computable by a composable CRC, then f is superadditive.*

Proof. Assume $\mathcal{C}$ stably computes f. By definition of $\mathcal{C}$ stably computing f, $\forall$ initial states $\boldsymbol{x_1}, \boldsymbol{x_2}$, $\exists$ $\boldsymbol{o}_1, \boldsymbol{o}_2$ such that $\boldsymbol{x}_1 \to \boldsymbol{o}_1$ with $\boldsymbol{o}_1(Y) = f(\boldsymbol{x}_1)$ and $\boldsymbol{x}_2 \to \boldsymbol{o}_2$ with $\boldsymbol{o}_2(Y) = f(\boldsymbol{x}_2)$. Consider $\mathcal{C}$ on input $\boldsymbol{x}_1 + \boldsymbol{x}_2$. By the claim, $\boldsymbol{x}_1 + \boldsymbol{x}_2 \to \boldsymbol{o}_1 + \boldsymbol{x}_2$, and again by the claim, $\boldsymbol{o}_1 + \boldsymbol{x}_2 \to \boldsymbol{o}_1 + \boldsymbol{o}_2$. Looking at the concentration of output species Y, we have $(\boldsymbol{o}_1 + \boldsymbol{o}_2)(Y) = f(\boldsymbol{x}_1) + f(\boldsymbol{x}_2)$. Since $\mathcal{C}$ stably computes f, there exists an output stable state $\boldsymbol{o'}$ reachable from initial state $\boldsymbol{x}_1 + \boldsymbol{x}_2$ and reachable from state $\boldsymbol{o}_1 + \boldsymbol{o}_2$, with $\boldsymbol{o'}(Y) = f(\boldsymbol{x}_1 + \boldsymbol{x}_2)$. Since $\mathcal{C}$ is composable, species Y does not appear as a reactant and thus its concentration in any state reachable from state $\boldsymbol{o}_1 + \boldsymbol{o}_2$ cannot be reduced from $f(\boldsymbol{x}_1) + f(\boldsymbol{x}_2)$, implying $\boldsymbol{o'}(Y) = f(\boldsymbol{x}_1 + \boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + f(\boldsymbol{x}_2)$. This holds for all input states $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and thus f is superadditive. □

Corollary 1. *No composable CRC computes $f(x_1, x_2) = \max(x_1, x_2)$.*

3.2 Superadditive Positive-Continuous Piecewise Rational Linear Functions are Computable

It was shown in [9] that every piecewise linear function can be written as a max of mins of linear functions. This fact was exploited in [4] to construct a CRN that dual-rail computed continuous piecewise rational linear functions. To directly compute a positive-continuous piecewise rational linear function, dual-rail networks were used to compute the function on each domain, take the appropriate max of mins, and then the reaction $Y^+ + Y^- \to \varnothing$ was used to convert the dual-rail output into a direct output where the output species is Y^+. However, this technique is not usable in our case: by Corollary 1, we cannot compute the max function, and the technique of converting dual-rail output to a direct output is not output oblivious. In fact, computing $f(Y^+, Y^-) = Y^+ - Y^-$ is not superadditive, and so by Lemma 6, there is no composable CRC which computes this conversion.

Since our functions are positive-continuous, we first consider domains where the function is continuous, and show that it can be computed by composing rational linear functions with min. Since rational linear functions and min can be computed without using the output species as a reactant, we achieve composability. We then extend this argument to handle discontinuities between domains.

Definition 9. *An* open ray ℓ *in* $\mathbb{R}^n$ *from the origin through a point* $\boldsymbol{x}$ *is the set* $\ell = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} = t \cdot \boldsymbol{x},\ t \in \mathbb{R}_{>0}\}$*. Note that* t *is strictly positive, so the origin is not contained in* ℓ*.*

Definition 10. *We call a subset* $D \subseteq \mathbb{R}^n$ *a* cone *if for all* $\boldsymbol{x} \in \mathbb{R}^n$*, we know that* $\boldsymbol{x} \in D$ *implies the open ray from the origin through* $\boldsymbol{x}$ *is contained in* D*.*

Lemma 7. *Suppose we are given a continuous piecewise rational linear function* $f : \mathbb{R}^n_{>0} \to \mathbb{R}_{\geq 0}$*. Then we can choose domains for* f *which are cones which contain an open ball of non-zero radius.*

Intuitively, we can consider any open ray from the origin and look at the domains for f along this ray. If the ray traveled through different domains, then there must be boundary points where the function switches domains. But we know that f is continuous, so the domains must agree on their boundaries. Since there is only one line that passes through the origin and any given point, the domains must share the same linear function to be continuous. Thus we can place the ray into one domain corresponding to its linear function. Applying this argument to all rays gives these domains as cones. This argument is formalized in the below proof.

Proof. Since f is piecewise rational linear, we can pick a finite set of domains $\mathcal{D} = \{D_i\}_{i=1}^N$ for f, such that $f|_{D_i} = g_i|_{D_i}$, where g_i is a rational linear function. Fix a domain D_k, and consider any point $\boldsymbol{x} \in D_k$. Since the open ray $\ell_{\boldsymbol{x}}$ from the origin passing through $\boldsymbol{x}$ is contained in $\mathbb{R}^n_{>0}$, it is covered by the domains in $\mathcal{D}$. If we write any point $\boldsymbol{y} \in \ell_{\boldsymbol{x}}$ in the form $t \cdot \boldsymbol{x}$, then, for each i, the restriction

of g_i to $D_i \cap \ell_{\boldsymbol{x}}$ is of the form $g_i(t \cdot \boldsymbol{x}) = \alpha_i t$ for some $\alpha_i \in \mathbb{R}$. Since $\boldsymbol{x} \in D_k \cap \ell_{\boldsymbol{x}}$, we know that $f(1 \cdot \boldsymbol{x}) = \alpha_k \cdot 1 = \alpha_k$

Now suppose that for some $s \in \mathbb{R}_{>0}$ we know that $f(s \cdot \boldsymbol{x}) \neq \alpha_k s$. First consider the case where $s > 1$. Then define the set $A = \{t \in [1, s] \mid f(t \cdot \boldsymbol{x}) = \alpha_k t\}$ and define the set $B = \{t \in [1, s] \mid f(t \cdot \boldsymbol{x}) \neq \alpha_k t\}$. We know that A is non-empty since $1 \in A$, so $\sup A$ exists - call it t'. From the standard properties of the supremum, we know that there exists a sequence of points $\{t_j\}_{j=1}^{\infty}$ such that $t_j \in A$ for all j and $\lim_{j\to\infty} t_j = t'$. As a result, from the continuity of f, we see that:

$$f(t' \cdot \boldsymbol{x}) = \lim_{j\to\infty} f(t_j \cdot \boldsymbol{x}) = \lim_{j\to\infty} \alpha_k t_j = \alpha_k t'$$

So $t' \in A$. However, by assumption, $s \in B$, so that $t' < s$. Since t' is an upper bound on A, it must then be the case that $(t', s] \subseteq B$, so that there exists a sequence of points $\{s_j\}_{j=1}^{\infty}$ such that $s_j \in B$ for all j and $\lim_{j\to\infty} s_j = t'$. Since there are only finitely many domains in $\mathcal{D}$, but infinitely many s_j, by the pigeonhole principle infinitely many of the s_j must be contained in a single domain $D_{k'}$. Now write the subsequence of points contained in $D_{k'}$ as $\{s_{j'}\}_{j'=1}^{\infty}$. We still know that $\lim_{j'\to\infty} s_{j'} = t'$, so by the continuity of f and the fact that $s_{j'} \in D_{k'}$, we see that:

$$\alpha_k t' = f(t' \cdot \boldsymbol{x}) = \lim_{j'\to\infty} f(s_{j'} \cdot \boldsymbol{x}) = \lim_{j'\to\infty} \alpha_{k'} s_{j'} = \alpha_{k'} t'$$

Since $t' > 0$, this implies that $\alpha_{k'} = \alpha_k$, so that $f(s_{j'} \cdot \boldsymbol{x}) = \alpha_k s_{j'}$. However, this contradicts the fact that we were able to choose $s_{j'} \in B$. As a result, our assumption, that there is some $s > 1$ such that $f(s \cdot \boldsymbol{x}) \neq \alpha_k s$, must be false. A similar argument, using the infimum instead of the supremum, shows that there can be no $s < 1$ such that $f(s \cdot \boldsymbol{x}) \neq \alpha_k s$. As a result, for every point $t \in \ell_{\boldsymbol{x}}$, we know $f(t \cdot \boldsymbol{x}) = \alpha_k t$. In other words, $f|_{\ell_{\boldsymbol{x}}} = g_k|_{\ell_{\boldsymbol{x}}}$, so we can replace D_k with $D_k \cup \ell_{\boldsymbol{x}}$ without issue. Doing this for every $\boldsymbol{x} \in D_k$, we can replace D_k with a cone. By enlarging every domain in $\mathcal{D}$ in this way, we can choose domains for f which are cones.

Since f is continuous, we can replace each $D_i \in \mathcal{D}$ by its closure, which is again a cone. Suppose that for any $D_i \in \mathcal{D}$, there is a point $\boldsymbol{x} \in D_i$ is not in the interior of D_i. Then $\boldsymbol{x}$ is in the closure of the complement of D_i, so there exists a sequence $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ of points in the complement of D_i such that $\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}$. Since the complement of D_i is covered by the $D_j \in \mathcal{D}$, where $j \neq i$, we know that each $\boldsymbol{x}_k$ lies in one of the D_j. Since there are only finitely many D_j but infinitely many $\boldsymbol{x}_k$, we know that infinitely many $\boldsymbol{x}_k$ must lie in at least one of the D_j. As a result, $\boldsymbol{x}$ is in the closure of this D_j, and since D_j is closed, we see that $\boldsymbol{x} \in D_j$. Because of this, if D_i has no interior points, then it is completely contained in the other D_j, so we can remove it from the set of domains. After doing this for every D_i which contains no interior points, we can ensure that the domains we have chosen for f all contain an open ball of non-zero radius. □

Lemma 8. *Any superadditive continuous piecewise rational linear function $f : \mathbb{R}^n_{>0} \to \mathbb{R}_{\geq 0}$ can be written as the minimum of a finite number of rational linear functions g_i.*

Proof. Since f is a continuous piecewise rational linear function, by Lemma 7, we can choose domains $\{D_i\}_{i=1}^N$ for f which are cones and contain an open ball of non-zero radius, such that $f|_{D_i} = g_i|_{D_i}$, where g_i is a rational linear function. Now pick any $\boldsymbol{x} \in \mathbb{R}^n_{>0}$ and any g_j. Then because D_j is a cone containing an open ball of finite radius, it contains open balls with arbitrarily large radii. In particular, it contains a ball with radius greater than $|\boldsymbol{x}|$, so there exist points $\boldsymbol{y}, \boldsymbol{z} \in D_j$ such that $\boldsymbol{y} + \boldsymbol{x} = \boldsymbol{z}$. By the superadditivity of f, the linearity of g_j, and the fact that $\boldsymbol{y}, \boldsymbol{z} \in D_j$, we see:

$$g_j(\boldsymbol{y}) + f(\boldsymbol{x}) = f(\boldsymbol{y}) + f(\boldsymbol{x}) \leq f(\boldsymbol{z}) = g_j(\boldsymbol{x} + \boldsymbol{y}) = g_j(\boldsymbol{y}) + g_j(\boldsymbol{x})$$

So that $f(\boldsymbol{x}) \leq g_j(\boldsymbol{x})$. Since this is true for all g_j, and since we know that $f(\boldsymbol{x}) = g_i(\boldsymbol{x})$ for some i, we see that $f(\boldsymbol{x}) = \min_i g_i(\boldsymbol{x})$, as desired. □

Lemma 8 is particularly useful for us since, as seen in the introduction, CRCs computing min are easy to construct, and rational linear functions are relatively straightforward as well. The next lemma gives details on constructing a CRC to compute f by piecing together CRCs which compute the components (rational linear functions) of f and then computing the min across their outputs. However, since Lemma 8 as given applies to continuous functions with domain $\mathbb{R}^n_{>0}$, so does this lemma; we handle the domain $\mathbb{R}^n_{\geq 0}$ later on.

Lemma 9. *We can construct a composable CRC that stably computes any superadditive continuous piecewise rational linear function $f : \mathbb{R}^n_{>0} \to \mathbb{R}_{\geq 0}$.*

Proof. By Lemma 8, we know that f can be written as the minimum of a finite number of rational linear functions g_i. Observe that a general rational linear function $g(\boldsymbol{x}) = a_1 x_1 + a_2 x_2 + \ldots a_n x_n$ is stably computed by the reactions

$$\forall i,\ k_i X_i \to a_i k_i Y$$

where k_i is a positive integer such that $k_i a_i$ is also a positive integer. Since f is the minimum of a number of g_i's, we can make a chemical reaction network where we compute each g_i using a copy of the input species, calling the output Y_i (the reaction $X_1 \to X_1^1 + \ldots + X_1^5$ produces five species with concentrations equal to X_1's initial concentration, effectively copying the input species so that the input may be a reactant in several modules without those modules competing). Next, we use the chemical reaction

$$Y_1 + \ldots + Y_n \to Y$$

to get the minimum of the Y_i's. Since each Y_i obtains the count of the corresponding g_i, this CRN will produce the minimum of the g_i's quantity of Y's. Thus, according to Lemma 8, the described CRC stably computes f. Note that each sub-CRC described in this construction is output oblivious, and thus composable, so the composition of these modules maintains correctness. □

The above construction only handles the domain $\mathbb{R}^n_{>0}$, where we know our functions are continuous by positive-continuity. However, when extended to the domain $\mathbb{R}^n_{\geq 0}$, positive-continuity of our functions allows discontinuity where inputs change from zero to positive. The challenge, then, is to compute the superadditive *continuous* piecewise rational linear function corresponding to which inputs are nonzero.

Surprisingly, Lemma 11 below shows that we can express a superadditive positive-continuous piecewise rational linear function as a min of superadditive *continuous piecewise rational linear functions*. The first step towards this expression is to see that, given two subspaces of inputs wherein the species present in one subspace A are a superset of the species present in a subspace B, the function as defined on the subspace A must be greater than the function as defined on the subspace B; otherwise, the function would disobey monotonicity and thus superadditivity, as proven below:

Lemma 10. *Consider any superadditive positive-continuous piecewise rational linear function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$. Write $N = [n]$, and for each $S \subseteq N$, let $g_S(\boldsymbol{x})$ be the superadditive continuous piecewise rational linear function that is equal to f on D_S. If $S, T \subseteq N$ and $S \subseteq T$, then for all $\boldsymbol{x} \in D_S$ we know $g_S(\boldsymbol{x}) \leq g_T(\boldsymbol{x})$.*

Proof. Write $\boldsymbol{e}_i$ for the vector of length 1 pointing in the positive direction of the ith coordinate axis. Define the vector $\boldsymbol{v} = \sum_{i \in T \setminus S} \boldsymbol{e}_i$. Then for any $\boldsymbol{x} \in D_S$ and any $\epsilon \in \mathbb{R}_{>0}$, we know that $\boldsymbol{x} + \epsilon \boldsymbol{v} \in D_T$. Since f is superadditive, it is also monotonic. Suppose that $g_T(\boldsymbol{x}) < g_S(\boldsymbol{x})$. Because g_T is continuous, taking $\delta = g_S(\boldsymbol{x}) - g_T(\boldsymbol{x}) > 0$, there is some small enough $\epsilon > 0$ such that

$$f(\boldsymbol{x} + \epsilon \boldsymbol{v}) = g_T(\boldsymbol{x} + \epsilon \boldsymbol{v}) < g_T(\boldsymbol{x}) + \delta = g_S(\boldsymbol{x}) = f(\boldsymbol{x})$$

contradicting the monotonicity of f. Our assumption must be false, so $g_S(\boldsymbol{x}) \leq g_T(\boldsymbol{x})$. □

Next we define a predicate for each subset of inputs which is true if all inputs in that subset are nonzero. Intuitively, in the CRC construction to follow, this predicate is used by the CRC to determine which inputs are present:

Definition 11. *For any set $S \subseteq [n]$, define the S-*predicate *$P_S : \mathbb{R}^n_{\geq 0} \to \{0, 1\}$ to be the function given by:*

$$P_S(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x}(i) > 0 \ \forall i \in S \\ 0 & \text{otherwise} \end{cases}$$

A naïve approach might be the following: for each subdomain D_S, the function is continuous, so compute it by CRC according to Lemma 9, producing an output Y_S. Then compute the P_S predicate by CRC, and if the predicate is true (e.g., a species representing P_S has nonzero concentration), use that species to catalyze a reaction which changes the Y_S to Y, the final output of the system. However, note that if T is a subset of S, P_S and and P_T are both true, so this technique will overproduce Y.

The following technique solves this issue by identifying a min which can be taken over the intermediate outputs Y_S. In particular, for each S, we compute $g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x})$, and then take the min of these terms. When S corresponds to the set of input species with initially nonzero concentrations, then the summation term in this expression is 0, since $P_K(x) = 0$ for all $K \not\subseteq S$. When S does not correspond to the set of input species with initially nonzero concentration, then either **(1)** it is a superset of the correct set I, in which case Lemma 10 says that $g_S(x) \geq g_I(x)$ (thus the min of these is $g_I(x)$) or **(2)** the summation term added to $g_S(x)$ contains at least $g_I(x)$, and since $g_S(x) + g_I(x) \geq g_I(x)$, the min of these is $g_I(x)$. Thus taking the min for all S of $g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x})$ is exactly $g_I(x)$, where I is the correct set of initially present input species.

Lemma 11. *Consider any superadditive positive-continuous piecewise rational linear function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$. Write $N = [n]$, and for each $S \subseteq N$, let $g_S(\boldsymbol{x})$ be the superadditive continuous piecewise rational linear function that is equal to f on D_S. Then, $f(\boldsymbol{x}) = \min_{S \subseteq N} [g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x})]$.*

Proof. For $S \subseteq N$, let $h_S : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ be given by

$$h_S(\boldsymbol{x}) = g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x})$$

We want to show that $f(\boldsymbol{x}) = \min_{S \subseteq N} h_S(\boldsymbol{x})$. To do this, fix $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ and define the set $I = \{i \in N \mid \boldsymbol{x}(i) > 0\}$. First, let's show that $h_I(\boldsymbol{x}) = f(\boldsymbol{x})$. By the definition of I, for all $K \not\subseteq I$, we know $P_K(\boldsymbol{x}) = 0$. Thus, $\sum_{K \not\subseteq I} P_K(\boldsymbol{x}) g_K(\boldsymbol{x}) = 0$, so $h_I(\boldsymbol{x}) = g_I(\boldsymbol{x}) = f(\boldsymbol{x})$. Now we must show that $h_S(\boldsymbol{x}) \geq f(\boldsymbol{x})$ for all $S \subseteq N$. There are two cases to consider:

Case 1: $S \not\supseteq I$
In this case,

$$h_S(\boldsymbol{x}) = g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x}) \geq g_S(\boldsymbol{x}) + P_I(\boldsymbol{x}) g_I(\boldsymbol{x}) \geq P_I(\boldsymbol{x}) g_I(\boldsymbol{x})$$

By the definition of I, we know $P_I(\boldsymbol{x}) = 1$, so $P_I(\boldsymbol{x}) g_I(\boldsymbol{x}) = g_I(\boldsymbol{x}) = f(\boldsymbol{x})$. Thus we get that $h_S(\boldsymbol{x}) \geq f(\boldsymbol{x})$.

Case 2: $S \supseteq I$
By Lemma 10, $g_S(\boldsymbol{x}) \geq g_I(\boldsymbol{x})$. As a result,

$$h_S(\boldsymbol{x}) = g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x}) \geq g_S(\boldsymbol{x}) \geq g_I(\boldsymbol{x}) = f(\boldsymbol{x})$$

Since for all $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$, we know $h_S(\boldsymbol{x}) \geq f(\boldsymbol{x})$ for all $S \subseteq N$ and $h_I(\boldsymbol{x}) = f(\boldsymbol{x})$ for some $I \subseteq N$, it follows that $f(\boldsymbol{x}) = \min_{S \subseteq N} h_S(\boldsymbol{x})$. □

Lemma 12 takes the above Lemma 11 along with the construction which stably computes on strictly continuous domains from Lemma 9 to construct a CRC which stably computes on positive-continuous domains.

Lemma 12. *Given any superadditive positive-continuous piecewise rational linear function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$, there exists a composable CRC which stably computes f.*

Proof. The proof follows by identifying that the function can be expressed as a composition of functions (via Lemma 11) which are computable by output oblivious CRCs and are thus composable by Lemma 2. By Lemma 11, we know that $f(\boldsymbol{x}) = \min_{S \subseteq N}[g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x})]$. The first subroutine copies the input species, e.g. $X_1 \to X_1^1 + \ldots + X_1^5$, in order for each sub-CRC to not compete for input species. This copying is output oblivious. Then for any $Q \subseteq [n]$, $P_Q(x)$ is computed using one set of copies via the reaction:

$$\sum_{i \in Q} X_i \to P_Q$$

noting that although the predicate $P_Q(x)$ is defined to be 0 or 1, it is sufficient in this construction for the concentration of the species representing $P_Q(x)$ to be zero or nonzero. This CRC is output oblivious.

We can also compute each $g_Q(\boldsymbol{x})$ (via Lemma 9) using copies of the input molecules. This construction is output oblivious. To compute $P_Q(x)g_Q(x)$ given the concentration species P_Q as nonzero iff $P_Q(x) = 1$ as shown above, we simply compute the following (assuming Y_Q is the output of the module computing $g_Q(x)$):

$$f(P_Q, Y_Q) = \begin{cases} Y_Q & P_Q \neq 0 \\ 0 & P_Q = 0 \end{cases}$$

which is computed by this output oblivious CRC:

$$Y_Q + P_Q \to Y + P_Q$$

The CRC computing min is output oblivious, as seen in the introduction. The CRC computing the sum of its inputs is output oblivious (e.g., $X_1 \to Y, X_2 \to Y$ computes $X_1 + X_2$). Since each CRC shown is output oblivious and thus composable, we can compose the modules described to construct a CRC stably computing $\min_{S \subseteq N}[g_S(\boldsymbol{x}) + \sum_{K \not\subseteq S} P_K(\boldsymbol{x}) g_K(\boldsymbol{x})]$, which is equal to $f(\boldsymbol{x})$ by Lemma 11. □

Corollary 2. *Given any superadditive positive-continuous piecewise rational linear function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$, there exists a composable CRC with reactions with at most two reactants and at most two products which stably computes f.*

To deduce this corollary, note that the reactions with more than two reactants and/or products are used to compute the following functions: computation of

a rational linear function, copying inputs, min, and predicate computation. We can decompose such reactions into a set of bimolecular reactions. For example, a reaction $X_1 + \ldots + X_n \to Y_1 + \ldots + Y_n$ can be decomposed into the reactions $X_1 + X_2 \to X_{12},\ X_{12} + X_3 \to X_{123}, \ldots, X_{123\ldots n-1} + X_n \to Y_{12\ldots n-1} + Y_n$, $Y_{12\ldots n-1} \to Y_{12\ldots n-2} + Y_{n-1}, \ldots, Y_{12} \to Y_1 + Y_2$. We can verify that each affected module stably computes correctly with these expanded systems of reactions, and remains composable.

4 Example

In this section, we demonstrate the construction presented in the previous section through an example. Consider the function shown in Equation 1 in Section 3. As shown in that section, the function is superadditive, positive-continuous, and piecewise rational linear. Thus, we can apply our construction to generate a composable CRN stably computing this function. Note that while this CRN is generated from our methodology, we have removed irrelevant species and reactions.

Making copies of input:

$$X_1 \to X_1'' + X_1'''$$
$$X_2 \to X_2'' + X_2'''$$
$$X_3 \to X_3'$$

Using X_3' to make P_3, which catalyzes reactions for the domain $X_3 > 0$:

$$X_3' \to P_{\{3\}}$$

Computing the sum in $Y_{\{3\}}$:

$$X_1'' \to Y_{\{3\}}$$
$$X_2'' \to Y_{\{3\}}$$

Computing the min in $Y_\varnothing$:

$$X_1''' + X_2''' \to Y_\varnothing$$

Making a copy of $Y_{\{3\}}$ for use in increasing $Y_\varnothing'$:

$$Y_{\{3\}} \to Y_{\{3\}}' + Y_{\{3\},\varnothing}$$

Increase $Y_\varnothing'$ so that it won't be the min when x_3 is present:

$$Y_{\{3\},\varnothing} + P_{\{3\}} \to Y_\varnothing' + P_{\{3\}}$$

Rename $Y_\varnothing$ to $Y_\varnothing'$ so that it will be summed with the term created by the previous reaction:

$$Y_\varnothing \to Y_\varnothing'$$
$$Y_\varnothing' + Y_{\{3\}}' \to Y$$

5 Future Work

Instead of continuous concentrations of species, one may consider discrete counts. This changes which functions are stably computed by CRNs. Without our composability constraint, [3] shows in the discrete model that a function $f : \mathbb{N}^n \to \mathbb{N}$ is stably computable by a direct CRN if and only if it is semilinear; i.e., its graph $\{(x, y) \in \mathbb{N}^n \times \mathbb{N} \mid f(x) = y\}$ is a semilinear subset of $\mathbb{N}^n \times \mathbb{N}$. The proof that composably computable functions must be superadditive (Lemma 6) holds for the discrete model as well. So, it is now known that functions computable

in the discrete model by a direct, composable CRN must be superadditive and semilinear. Surprisingly, however, there exists a function which is superadditive and semilinear which is not computable by such CRNs (the proof is non-trivial and is omitted):

$$f(x_1, x_2) = \begin{cases} x_1 - 1 & x_1 > x_2 \\ x_1 & x_1 \leq x_2. \end{cases}$$

Therefore the exact characterization of the class of computable functions for the discrete, composable case remains an open question.

In our model of a chemical reaction computer, we restrict the concentrations of non-input species in the initial state to be zero. One may consider some (constant) initial concentration of non-input species, called *initial context*, and how that may affect computation. In the non-composable case, this allows the components of the piecewise functions to be rational *affine* functions of the form $f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i \boldsymbol{x}(i) + b$, where the additional b concentration of output species is produced from the initial concentration of non-output species. However, in the composable case, a function as simple as $f(x) = x - 1$ is not computable, even though it is superadditive and affine. Additionally, since $f(x) = x + 1$ is not superadditive, it also cannot be computed. Uncovering what, if any, additional power is gained from initial context in the composable case remains open.

Our negative and positive results are proven with respect to stable computation, which formalizes our intuitive notion of rate-independent computation. It is possible to strengthen our positive results to further show that our CRNs converge (as time $t \to \infty$) to the correct output from any reachable state under *mass-action kinetics* (proof omitted). It is interesting to characterize the exact class of rate laws that guarantee similar convergence.

Apart from the dual-rail convention discussed in the introduction, other input/output conventions for computation by CRNs have been studied. For example, [10] considers *fractional encoding* in the context of rate-dependent computation. As shown by dual-rail, different input and output conventions can affect the class of functions stably computable by CRNs. While using any superadditive positive continuous piecewise rational linear output convention gives us no extra computational power—since the construction in this paper shows how to compute such an output convention directly—it is unclear how these conventions change the power of rate-independent CRNs in general.

Finally we can ask what insights the study of composition of rate-independent modules gives for the more general case of rate-dependent computation. Is there a similar tradeoff between ease of composition and expressiveness for other classes of CRNs?

References

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

2. Luca Cardelli. Strand algebras for DNA computing. *Natural Computing*, 10(1):407–428, 2011.
3. Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural computing*, 13(4):517–534, 2014.
4. Ho-Lin Chen, David Doty, and David Soloveichik. Rate-independent computation in continuous chemical reaction networks. In *Proceedings of the 5th conference on Innovations in Theoretical Computer Science*, pages 313–326. ACM, 2014.
5. 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, 2013.
6. Domitilla Del Vecchio, Alexander J Ninfa, and Eduardo D Sontag. Modular cell biology: retroactivity and insulation. *Molecular systems biology*, 4(1):161, 2008.
7. François Fages, Guillaume Le Guludec, Olivier Bournez, and Amaury Pouly. Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In *International Conference on Computational Methods in Systems Biology*, pages 108–127. Springer, 2017.
8. Martin Feinberg and Friedrich JM Horn. Dynamics of open chemical systems and the algebraic structure of the underlying reaction network. *Chemical Engineering Science*, 29(3):775–787, 1974.
9. Sergei Ovchinnikov. Max-min representation of piecewise linear functions. *Contributions to Algebra and Geometry*, 43(1):297–302, 2002.
10. Sayed Ahmad Salehi, Xingyi Liu, Marc D. Riedel, and Keshab K. Parhi. Computing mathematical functions using dna via fractional coding. *Scientific Reports*, 8(8312), 2018.
11. David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010.