

# Composable Rate-Independent Computation in Continuous Chemical Reaction Networks

Cameron Chalk, Niels Kornerup, Wyatt Reeves, David Soloveichik

**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, 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 equilibrium 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. Thus composability severely limits rate-independent computation unless more sophisticated input/output encodings are used.



## 1 INTRODUCTION

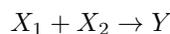
A ubiquitous form of biological information processing occurs in complex chemical regulatory networks in cells. The formalism of chemical reaction networks (CRNs) has been widely used for modelling the interactions underlying such natural chemical computation. More recently CRNs have also become a useful model for designing synthetic molecular cascades can in principle realize arbitrary CRNs, thus motivating the study of CRNs as a programming language [2], [5], [15], [16]. The applications of synthetic chemical computation include reprogramming biological regulatory networks, as well as embedding control modules in environments that are inherently 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 [8]; 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  $2X \rightarrow Y$ . 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

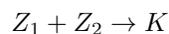
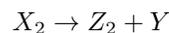
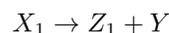
stoichiometry is that it is fundamentally *rate-independent*, requiring no assumptions on the rate law (e.g., that the reaction occurs at a rate proportional to the product of the concentrations of the reactants). This allows the computation to be correct independent of experimental conditions such as 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 [7]. In this paper, we attempt to capture a natural notion of composable rate-independent computation, and study whether composability restricts computational power.

(a)



(b)

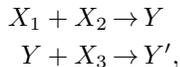


Above, we see two examples of rate-independent computation. Example (a) shows  $y = \min(x_1, x_2)$ . The amount of  $Y$  eventually produced will be the minimum of the initial amounts of  $X_1$  and  $X_2$ , since the reaction will stop as soon as the first reactant runs out. Example (b) shows

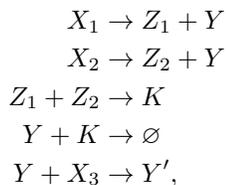
- The authors were with the University of Texas at Austin.
- C. Chalk and D. Soloveichik were supported in part by National Science Foundation grants CCF-1618895 and CCF-1652824.

$y = \max(x_1, x_2)$ . (The last reaction generates waste species that are not used in the computation; thus, we describe the products as an empty set.) 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:



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:



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 min module. 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 be defined by a finite number of linear pieces (with rational coefficients). Note that non-linear continuous functions can be approximated to arbitrary accuracy.<sup>1</sup> We show that requiring the

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.

CRN to be composable restricts the class of computable functions to be superadditive functions; i.e., functions that satisfy: for all input vectors  $\mathbf{a}, \mathbf{b}$ ,  $f(\mathbf{a}) + f(\mathbf{b}) \leq f(\mathbf{a} + \mathbf{b})$ . 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  $[n]$ . Let  $\mathbb{R}_{\geq 0}$  be the set of nonnegative real numbers, and similarly  $\mathbb{R}_{> 0}$  be the set of positive real numbers. If  $\Lambda$  is a finite set (in this paper, of chemical species), we write  $\mathbb{R}^\Lambda$  to denote the set of functions  $f : \Lambda \rightarrow \mathbb{R}$ , and similarly for  $\mathbb{R}_{\geq 0}^\Lambda, \mathbb{N}^\Lambda$ , etc. Equivalently, we view an element  $\mathbf{c} \in \mathbb{R}^\Lambda$  as a vector of  $|\Lambda|$  elements of  $\mathbb{R}$ , each coordinate “labeled” by an element of  $\Lambda$ . Given a function  $f : A \rightarrow B$ , we use  $f|_C$  to denote the restriction of  $f$  to the domain  $C$ . We also use the notation  $\mathbf{c} \upharpoonright \Delta$  to represent  $\mathbf{c}$  projected onto  $\mathbb{R}_{\geq 0}^\Delta$ . Thus,  $\mathbf{c} \upharpoonright \Delta = \mathbf{0}$  iff  $(\forall S \in \Delta) \mathbf{c}(S) = 0$ . If  $\Delta \subseteq \Lambda$ , we view a vector  $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Delta$  equivalently as a vector  $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda$  by assuming  $\mathbf{c}(S) = 0$  for all  $S \in \Lambda \setminus \Delta$ .

### 2.1 Chemical reaction networks

We will start by defining the notation used to describe chemical reactions.

**Definition 1.** Given a finite set of chemical species  $\Lambda$ , a reaction over  $\Lambda$  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.<sup>2</sup>

In this paper, we assume that  $\mathbf{r} \neq \mathbf{0}$ , i.e., we have no reactions of the form  $\emptyset \rightarrow \dots$ . For instance, given  $\Lambda = \{A, B, C\}$ , the reaction  $A + 2B \rightarrow A + 3C$  is the pair  $\langle (1, 2, 0), (1, 0, 3) \rangle$ .

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

**Definition 2.** A (finite) chemical reaction network (CRN) is a pair  $\mathcal{C} = (\Lambda, R)$ , where  $\Lambda$  is a finite set of chemical species, and  $R$  is a finite set of reactions over  $\Lambda$ .

Next we map language about chemical reaction networks to formal definitions and notation.

**Definition 3.** A state of a CRN  $\mathcal{C} = (\Lambda, R)$  is a vector  $\mathbf{c} \in \mathbb{R}_{\geq 0}^{\Lambda}$ .

**Definition 4.** For any  $\mathbf{c} \in \mathbb{R}_{\geq 0}^{\Lambda}$  and any  $S \in \Lambda$ ,  $\mathbf{c}(S)$  is the concentration of  $S$  in  $\mathbf{c}$ .

**Definition 5.** For any  $\mathbf{c} \in \mathbb{R}_{\geq 0}^{\Lambda}$ , the set of species present in  $\mathbf{c}$  (denoted by  $[\mathbf{c}]$ ) is  $\{S \in \Lambda \mid \mathbf{c}(S) > 0\}$ .

**Definition 6.** Given a state  $\mathbf{c}$  and reaction  $\alpha = \langle \mathbf{r}, \mathbf{p} \rangle$ , we say that  $\alpha$  is applicable in  $\mathbf{c}$  if  $[\mathbf{r}] \subseteq [\mathbf{c}]$  (i.e.,  $\mathbf{c}$  contains positive concentration of all of the reactants).

**Definition 7.** A reaction produces (consumes) a species  $S$  if  $S$  appears as a product (reactant).<sup>3</sup>

## 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<sup>4</sup> 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.

**Definition 8.** Let  $\mathcal{C}$  be a CRN defined by  $(\Lambda, R)$ . The linear transformation  $\mathbf{M} : \mathbb{R}^R \rightarrow \mathbb{R}^{\Lambda}$  that maps from the unit vector representing a reaction to the net change in species caused by that reaction is the stoichiometry matrix for  $\mathcal{C}$ .

Note that we can intuitively think of  $\mathbf{M}$  being a matrix where the columns represent the net change in species caused by each reaction. Under this representation, observe that entries in  $\mathbf{M}$  will be negative when more of a reactant is consumed than is produced in a reaction. Observe that the image of  $\mathbf{M}$  represents the possible changes in a state that can occur via the reactions in  $R$ . We will formalize this notion with the next few definitions.

**Definition 9.** For a CRN with the reactions  $R$ , we say that any vector  $\mathbf{u} \in \mathbb{R}_{\geq 0}^R$  is a flux vector. We use  $[\mathbf{u}]$  to denote the set

3. Note that typically a catalyst species is not considered consumed nor produced by a catalytic reaction. For simplicity, as our definition suggests, we say it is both produced and consumed.

4. 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  $A + B \rightarrow A + C$  and  $C + C \rightarrow B$  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.

$\{r \mid \mathbf{u}(r) > 0\}$ . We say that  $\mathbf{u}$  is applicable at a state  $\mathbf{c}$  if every reaction in  $[\mathbf{u}]$  is applicable at  $\mathbf{c}$ .

**Definition 10.** For a CRN with species  $\Lambda$  and stoichiometry matrix  $\mathbf{M}$ , we say a state  $\mathbf{d} \in \mathbb{R}_{\geq 0}^{\Lambda}$  is straight-line reachable from  $\mathbf{c}$ , written  $\mathbf{c} \rightarrow^1 \mathbf{d}$ , or more precisely as  $\mathbf{c} \rightarrow_{\mathbf{u}} \mathbf{d}$ , if there is a applicable flux vector  $\mathbf{u}$  such that  $\mathbf{c} + \mathbf{M}\mathbf{u} = \mathbf{d}$ .

Intuitively, a single segment means running the reactions applicable at  $\mathbf{c}$  at a constant (possibly 0) rate specified by  $\mathbf{u}$  to get from  $\mathbf{c}$  to  $\mathbf{d}$ . Since applying a flux vector can change the set of species present,  $\mathbf{a} \rightarrow^1 \mathbf{b}$  does not imply that  $\mathbf{a}$  and  $\mathbf{b}$  have the same set of applicable reactions. Therefore there can be a state  $\mathbf{c}$  that is straight-line reachable from  $\mathbf{b}$  but not from  $\mathbf{a}$ . This leads us to our next definition.

**Definition 11.** We say state  $\mathbf{d}$  is 1-segment reachable from  $\mathbf{c}$  if it is straight line reachable. We say a state  $\mathbf{d}$  is  $l$ -segment reachable if there is a state  $\mathbf{d}'$  that is  $(l - 1)$ -segment reachable from  $\mathbf{c}$  such that  $\mathbf{d}' \rightarrow^1 \mathbf{d}$ .

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 [9].

**Definition 12.** A state  $\mathbf{d}$  is reachable from  $\mathbf{c}$ , written  $\mathbf{c} \rightarrow \mathbf{d}$ , if  $\exists l \in \mathbb{N}$  such that  $\mathbf{d}$  is  $l$ -segment reachable from  $\mathbf{c}$ . We denote the set of states reachable from  $\mathbf{c}$ , i.e.,  $\{\mathbf{d} \mid \mathbf{c} \rightarrow \mathbf{d}\}$ , as  $\text{Post}(\mathbf{c})$ .

We think of state  $\mathbf{d}$  as being reachable from state  $\mathbf{c}$  if there is a “reasonable” rate law that takes the system from  $\mathbf{c}$  to  $\mathbf{d}$ . As desired, previous work showed that if state  $\mathbf{d}$  is reached from  $\mathbf{c}$  via a mass-action trajectory, it is also segment-reachable.

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

We can now use reachability to formally define rate-independent computation.

**Definition 13.** A chemical reaction computer (CRC) is a tuple  $\mathcal{C} = (\Lambda, R, \Sigma, Y)$ , where  $(\Lambda, R)$  is a CRN,  $\Sigma \subset \Lambda$ , written as  $\Sigma = \{X_1, \dots, X_n\}$ , is the set of input species, and  $Y \in \Lambda \setminus \Sigma$  is the output species.

For simplicity, assume a canonical ordering of  $\Sigma = \{X_1, \dots, X_n\}$  so that a vector  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$  (i.e., an input to  $f$ ) can be viewed equivalently as a state  $\mathbf{x} \in \mathbb{R}_{\geq 0}^{\Sigma}$  of  $\mathcal{C}$  (i.e., an input to  $\mathcal{C}$ ).

**Definition 14.** A state  $\mathbf{o} \in \mathbb{R}_{\geq 0}^{\Lambda}$  is output stable if, for all  $\mathbf{o}'$  such that  $\mathbf{o} \rightarrow \mathbf{o}'$ ,  $\mathbf{o}(Y) = \mathbf{o}'(Y)$ , i.e., once  $\mathbf{o}$  is reached, no reactions can change the concentration of the output species  $Y$ .

**Definition 15.** Let  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  be a function and let  $\mathcal{C}$  be a CRC. We say that  $\mathcal{C}$  stably computes  $f$  if, for all  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$  and all  $\mathbf{c}$  such that  $\mathbf{x} \rightarrow \mathbf{c}$ , there exists an output stable state  $\mathbf{o}$  such that  $\mathbf{c} \rightarrow \mathbf{o}$  and  $\mathbf{o}(Y) = f(\mathbf{x})$ .

We can intuitively justify the above definition of reachability and stable computation as capturing the class of computation that is independent of the rate law. The output stable states are exactly those in which the output cannot be

changed by a rate law chosen by an adversary. If a chemical reaction network does not stably compute a function, then some rate law can take the system to a state from which an output stable state is not reachable (including by mass-action by Lemma 1).

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

Also note that initial states contain only the input species  $\Sigma$ ; other species must have initial concentration 0. Section 5 discusses how allowing some initial concentration of non-input species affects computation.

### 2.3 Composability

In this section we define the composition of CRCs and formally relate composability to a CRC not using its output species as a reactant (output-obliviousness). We show that output-oblivious CRCs are composable, and that any composable CRC can be reduced (simply by removing reactions) to an output-oblivious form.

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

**Definition 16.** 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  $\mathcal{C}_2$ .

We say two CRCs are composable if they stably compute the composition of their functions when composed:

**Definition 17.** A CRC  $\mathcal{C}_1$  which stably computes  $f_1$  is composable if  $\forall \mathcal{C}_2$  stably computing  $f_2$ ,  $\mathcal{C}_{2 \circ 1}$  stably computes  $f_2 \circ f_1$ .

We want to relate composability to the property that a CRC does not use its output species as a reactant:

**Definition 18.** We call a CRC  $(\Lambda, R, \Sigma, Y)$  output-oblivious if  $Y$  does not appear as a reactant in  $R$ .

For simplicity, we focus on single-input, single-output CRCs, but these results easily generalize to multiple input and output settings.

For the proof that the output-oblivious condition is sufficient to guarantee composability, we formalize the idea that the composed CRCs act independently, and do not interfere with each other's execution. In Lemmas 2 and 3 we show how this independence can be used to "reorder" the sequence of reactions of a CRC in way that preserves the concentrations in the final state. In Lemma 4, we take an output-oblivious CRC, compose another CRC downstream, and reorder any sequence of reactions of the composed CRC into a sequence which we can easily argue must have stably computed as expected.

**Definition 19.** A flux vector  $\mathbf{u}$  produces (consumes) a species  $S$  if there is an  $\mathbf{r} \in [\mathbf{u}]$  such that  $S$  is a product (reactant) of  $\mathbf{r}$ .

**Definition 20.** A flux vector  $\mathbf{u}_1$  is independent of a flux vector  $\mathbf{u}_2$  if  $\mathbf{u}_1$  does not consume any species that are produced or consumed by  $\mathbf{u}_2$ .

**Lemma 2.** In a CRC  $\mathcal{C} = (\Lambda, R)$  if flux vector  $\mathbf{u}_1$  is independent of flux vector  $\mathbf{u}_2$  then:

- 1) If  $\mathbf{a} \xrightarrow{\mathbf{u}_2} \mathbf{b} \xrightarrow{\mathbf{u}_1} \mathbf{c}$ , then  $\mathbf{a} \xrightarrow{\mathbf{u}_1 + \mathbf{u}_2} \mathbf{c}$ .
- 2) If  $\mathbf{a} \xrightarrow{\mathbf{u}_1 + \mathbf{u}_2} \mathbf{c}$ , then there is a state  $\mathbf{b}$  such that  $\mathbf{a} \xrightarrow{\mathbf{u}_1} \mathbf{b} \xrightarrow{\mathbf{u}_2} \mathbf{c}$ .
- 3) If  $\mathbf{a} \xrightarrow{\mathbf{u}_1} \mathbf{b}$  and  $\mathbf{a} \xrightarrow{\mathbf{u}_2} \mathbf{c}$ , then there is a state  $\mathbf{d}$  such that  $\mathbf{c} \xrightarrow{\mathbf{u}_1} \mathbf{d}$ .

*Proof.* For 1, since  $\mathbf{u}_1$  is independent of  $\mathbf{u}_2$ , we know that  $\mathbf{u}_2$  cannot produce any of the species necessary to make  $\mathbf{u}_1$  applicable. Since  $\mathbf{u}_1$  was applicable at  $\mathbf{b}$ , we know that  $\mathbf{u}_1$  must be applicable at  $\mathbf{a}$ , so  $\mathbf{a} \xrightarrow{\mathbf{u}_1 + \mathbf{u}_2} \mathbf{c}$ .

For 2, we need to show that  $\mathbf{b} \in \mathbb{R}_{\geq 0}^\Lambda$  (has nonnegative concentrations) and that  $\mathbf{u}_2$  is applicable at  $\mathbf{b}$ . Consider any species  $S \in \Lambda$  such that  $S$  is not produced in  $\mathbf{u}_2$ . Since  $\mathbf{b} = \mathbf{c} - M\mathbf{u}_2$  and  $\mathbf{c}(S) \geq 0$ , we know that  $S$  has nonnegative concentration at  $\mathbf{b}$ . Now consider any species  $S \in \Lambda$  such that  $S$  is produced in  $\mathbf{u}_2$ . Since  $\mathbf{u}_2$  produces  $S$ , we know that  $\mathbf{u}_1$  must not consume it because  $\mathbf{u}_1$  is independent of  $\mathbf{u}_2$ . Since  $\mathbf{b} = \mathbf{a} + M\mathbf{u}_1$ , we can conclude that  $S$  must have nonnegative concentration at  $\mathbf{b}$ . Therefore we can conclude that  $\mathbf{b} \in \mathbb{R}_{\geq 0}^\Lambda$ . To see that  $\mathbf{u}_2$  is applicable at  $\mathbf{b}$ , first observe that  $\mathbf{u}_2$  is applicable at  $\mathbf{a}$ . Then, since  $\mathbf{u}_1$  is independent of  $\mathbf{u}_2$ , it follows that  $\mathbf{u}_2$  must also be applicable at  $\mathbf{b}$ .

For 3, we want to show that  $\mathbf{u}_1$  is still applicable at  $\mathbf{c}$  and that  $\mathbf{d} = \mathbf{c} + M\mathbf{u}_1$  is in  $\mathbb{R}_{\geq 0}^\Lambda$  (has nonnegative concentrations).  $\mathbf{u}_1$  is still applicable at  $\mathbf{c}$  since  $\mathbf{u}_1$  does not consume species involved in  $\mathbf{u}_2$ . If  $\mathbf{d}$  had negative concentration on species  $S$ , that species must have been consumed by  $\mathbf{u}_1$  since  $\mathbf{c}$  has nonnegative concentrations. Since  $\mathbf{u}_2$  does not consume any species consumed by  $\mathbf{u}_1$  and  $\mathbf{c} = \mathbf{a} + M\mathbf{u}_1$ , then negative  $\mathbf{d}(S)$  implies negative  $\mathbf{c}(S)$ , which contradicts that  $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda$ , so  $\mathbf{d} \in \mathbb{R}_{\geq 0}^\Lambda$ .  $\square$

**Lemma 3.** Given two output-oblivious CRCs  $\mathcal{C}_1$  and  $\mathcal{C}_2$ , consider the composition CRC  $\mathcal{C}_{2 \circ 1}$ . If  $\mathbf{c} \rightarrow \mathbf{d}$  then there is  $\mathbf{b}$  such that  $\mathbf{c} \rightarrow \mathbf{b} \rightarrow \mathbf{d}$ , where  $\mathbf{c} \rightarrow \mathbf{b}$  only uses reactions from  $\mathcal{C}_1$  and  $\mathbf{b} \rightarrow \mathbf{d}$  only uses reactions from  $\mathcal{C}_2$ .

*Proof.* Let  $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  be the flux vectors such that  $\mathbf{c} \xrightarrow{\mathbf{v}_1} \mathbf{c}_1 \xrightarrow{\mathbf{v}_2} \dots \xrightarrow{\mathbf{v}_n} \mathbf{d}$ . We can write  $\mathbf{v}_i = \mathbf{u}_{1,i} + \mathbf{u}_{2,i}$ , where  $\mathbf{u}_{1,i}$  corresponds to the reactions in  $\mathcal{C}_1$  and  $\mathbf{u}_{2,i}$  corresponds to the reactions in  $\mathcal{C}_2$ . Since  $\mathcal{C}_1$  is output-oblivious, we know that every  $\mathbf{u}_{1,i}$  is independent of every  $\mathbf{u}_{2,j}$  and thus we can apply Lemma 2 item 2 to see that  $\mathbf{c} \xrightarrow{\mathbf{u}_{1,1}} \mathbf{b}_1 \xrightarrow{\mathbf{u}_{2,1}} \mathbf{c}_1 \xrightarrow{\mathbf{u}_{1,2}} \mathbf{b}_2 \dots \xrightarrow{\mathbf{u}_{2,n}} \mathbf{d}$ . By repeatedly applying Lemma 2 items 1 and 2, we can then rearrange the sequence of reactions so that each  $\mathbf{u}_{1,i}$  precedes each  $\mathbf{u}_{2,j}$  to get  $\mathbf{c} \xrightarrow{\mathbf{u}_{1,1}} \mathbf{b}_1 \xrightarrow{\mathbf{u}_{1,2}} \dots \xrightarrow{\mathbf{u}_{1,n}} \mathbf{b} \xrightarrow{\mathbf{u}_{2,1}} \dots \xrightarrow{\mathbf{u}_{2,n}} \mathbf{d}$ .  $\square$

**Lemma 4.** Output-oblivious CRCs are composable.

*Proof.* Consider the composition  $\mathcal{C}_{2 \circ 1} = (\Lambda, R, \Sigma, Y)$  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  $x \in \mathbb{R}_{\geq 0}^\Sigma$ . Consider some state  $\mathbf{c}$  reached from  $\mathbf{x}$  in  $\mathcal{C}_{2 \circ 1}$ . Let  $\mathbf{b}$  be as in Lemma 3, so  $\mathbf{x} \xrightarrow{\mathbf{u}_{1,1}} \dots \xrightarrow{\mathbf{u}_{1,n}} \mathbf{b} \xrightarrow{\mathbf{u}_{2,1}} \dots \xrightarrow{\mathbf{u}_{2,n}} \mathbf{c}$ , where  $\mathbf{r}_i = (\mathbf{u}_{i,1}, \dots, \mathbf{u}_{i,n})$  is a sequence of flux vectors

with  $\bigcup_j [\mathbf{u}_{i,j}] \subseteq R_i$ . Since  $\mathcal{C}_1$  stably computes  $f_1$ , we know that there is some  $\mathcal{C}_1$ -output stable state  $\mathbf{o}_1$  reachable from  $\mathbf{b}$  using a series of flux vectors  $\mathbf{r} = (\mathbf{u}_1, \mathbf{u}_2 \dots \mathbf{u}_k)$  such that  $\bigcup_i [\mathbf{u}_i] \subseteq R_1$ . Since  $\mathbf{r}_2$  only uses reactions from  $\mathcal{C}_2$  and  $\mathcal{C}_1$  is output-oblivious, every flux vector in  $\mathbf{r}$  is independent of every flux vector in  $\mathbf{r}_2$ , so by Lemma 2 item 3 we know the sequence of flux vectors  $\mathbf{r}$  is applicable starting at  $\mathbf{c}$ . Let  $\mathbf{a}$  be such that  $\mathbf{c} \rightarrow_{\mathbf{u}_1} \dots \rightarrow_{\mathbf{u}_k} \mathbf{a}$ . Then applying Lemma 2 items 1 and 2 repeatedly to the flux vectors in  $\mathbf{r}$  and  $\mathbf{r}_2$ , we see that  $\mathbf{o}_1 \rightarrow_{\mathbf{u}_{2,1}} \dots \rightarrow_{\mathbf{u}_{2,n}} \mathbf{a}$ . Since  $\mathcal{C}_2$  stably computes  $f_2$ , since  $\mathbf{o}_1(Y_1) = f(x)$ , and since  $\mathbf{a}$  is reachable from  $\mathbf{o}_1$  only using reactions in  $\mathcal{C}_2$ , there must be some  $\mathbf{o}_2$  that is  $\mathcal{C}_2$ -output stable such that  $\mathbf{a} \rightarrow \mathbf{o}_2$  and  $\mathbf{o}_2(Y_2) = f_2 \circ f_1(x)$ . We know that  $\mathbf{o}_2$  is reachable from  $\mathbf{c}$  since  $\mathbf{c} \rightarrow \mathbf{a} \rightarrow \mathbf{o}_2$ . Finally, since  $\mathbf{o}_1$  is  $\mathcal{C}_1$ -output stable, reactions from  $\mathcal{C}_1$  cannot change the concentrations of species in  $\mathbf{o}_2 \upharpoonright \Lambda_2$ , so if  $\mathbf{o}_2 \rightarrow \mathbf{y}$ , then restricting to  $\mathcal{C}_2$  we find  $\mathbf{o}_2 \upharpoonright \Lambda_2 \rightarrow_{\mathcal{C}_2} \mathbf{y} \upharpoonright \Lambda_2$ . Since  $\mathbf{o}_2$  is  $\mathcal{C}_2$ -output stable we see that  $\mathbf{y}(Y_2) = \mathbf{o}_2(Y_2)$ , so  $\mathbf{o}_2$  is  $\mathcal{C}_{2 \circ 1}$ -output stable.  $\square$

Next we show that the output-oblivious condition is effectively necessary for composition. Technically, there are CRCs which are not output-oblivious but are composable. However, we show that for such CRCs, we can remove reactions until they are output-oblivious, resulting in a CRC which is still composable and computes the same function. Thus, characterizing what is computable by output-oblivious CRCs does characterize the class of functions computable by composable CRCs.

To prove this, we will want to reach a state that has used up its capacity to produce more of some species  $S$ . Intuitively this can be done by producing the maximum amount of  $S$  possible by the CRC. However, in general this notion is ill-defined.<sup>5</sup> Lemma 5 proves that there is a state with maximal amount of  $S$ . In other words, from any state we can always reach a state where afterwards it is impossible to increase the amount of  $S$ . The proof of Lemma 5 is left to the appendix.

**Lemma 5.** *For any state  $\mathbf{c}$  and any species  $S$ , if the amount of  $S$  present in any state reachable from  $\mathbf{c}$  is bounded above, there is a state  $\mathbf{d}$  reachable from  $\mathbf{c}$  such that for any state  $\mathbf{a}$  reachable from  $\mathbf{d}$ , we know that  $\mathbf{a}(S) \leq \mathbf{d}(S)$ .*

**Lemma 6.** *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.* Let  $\mathcal{C}_1 = (\Lambda_1, R_1, \Sigma_1, Y_1)$  be a composable CRC stably computing some function  $f$ . Let  $\mathcal{C}_0 \subseteq \mathcal{C}_1$  be the CRN obtained by removing all of the reactions that consume  $Y_1$  from  $\mathcal{C}_1$ . We would like to show that  $\mathcal{C}_0$  stably computes  $f$ . Suppose we compose  $\mathcal{C}_1$  with  $\mathcal{C}_2$  consisting of only the reaction  $Y_1 \rightarrow Y_2$  with input species  $Y_1$  and output species  $Y_2$ . Since  $\mathcal{C}_1$  is composable and  $\mathcal{C}_2$  stably computes the identity function, the resulting CRN  $\mathcal{C}_{2 \circ 1}$  must stably compute  $f$ . For any input vector  $\mathbf{x}$  consider a state  $\mathbf{c}$  reachable from  $\mathbf{x}$ .

<sup>5</sup> In the CRN  $X \rightarrow Z, X + Z \rightarrow S + Z$ , from an initial state  $\mathbf{x}$  consisting of only species  $X$ , note that  $\max_{\mathbf{c} \mid \mathbf{x} \rightarrow \mathbf{c}} \mathbf{c}(S)$  is not well-defined. To see this, note that some  $\epsilon$  amount of  $X$  must convert to  $Z$  to catalyze the second reaction; we can always choose a smaller  $\epsilon'$  to produce more  $S$ .

Assume that  $\mathcal{C}_0$  could reach a state  $\mathbf{d}$  from  $\mathbf{c}$  where  $\mathbf{d}(Y_1) > f(\mathbf{x})$ . Then  $\mathcal{C}_1$  would not be composable because  $\mathcal{C}_{2 \circ 1}$  can also reach  $\mathbf{d}$  and then applying the reaction in  $\mathcal{C}_2$  to convert all  $Y_1$  into  $Y_2$  gives us a state  $\mathbf{d}'$  with  $\mathbf{d}'(Y_2) > f(\mathbf{x})$ . Since there is no reaction in  $\mathcal{C}_{2 \circ 1}$  that consumes  $Y_2$  there is no output stable state reachable from  $\mathbf{d}'$  that computes  $f(\mathbf{x})$ . Therefore there is no state  $\mathbf{d}$  such that  $\mathbf{d}(Y_1) \geq f(\mathbf{x})$  and  $\mathbf{d}$  is reachable from  $\mathbf{c}$  via reactions of  $\mathcal{C}_0$ .

Since this implies that the amount of  $Y_1$  in any state reachable from  $\mathbf{c}$  is bounded, we can apply Lemma 5 to say that there is a state  $\mathbf{d}$  such that  $\mathbf{c} \rightarrow \mathbf{d}$  and for any state  $\mathbf{a}$  reachable from  $\mathbf{d}$  we know  $\mathbf{a}(Y_1) \leq \mathbf{d}(Y_1)$ . Since  $\mathcal{C}_0$  has no reactions that consume  $Y_1$ , this is an output stable state of  $\mathcal{C}_0$ . Now, consider the state  $\mathbf{b}$  in  $\mathcal{C}_{2 \circ 1}$  obtained by converting all  $Y_1$  in  $\mathbf{d}$  into  $Y_2$ . Observe that if there were a way to produce  $Y_1$  from  $\mathbf{b}$ , then there would be a state in  $\mathcal{C}_0$  reachable from  $\mathbf{d}$  that contained more  $Y_1$ . Since there are no reactions in  $\mathcal{C}_{2 \circ 1}$  that consume  $Y_2$  and no reactions that produce  $Y_1$ , we know that  $\mathbf{b}$  is an output stable state. Since  $\mathcal{C}_{2 \circ 1}$  stably computes  $f(\mathbf{x})$ , we know  $\mathbf{b}(Y_2) = f(\mathbf{x})$ . Thus we can conclude that  $\mathbf{d}(Y_1) = f(\mathbf{x})$  and  $\mathcal{C}_0$  stably computes  $f(\mathbf{x})$ .  $\square$

To allow composition of multiple downstream CRCs, we can use the reaction  $Y \rightarrow Y_1 + \dots + Y_n$  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 as *superadditive*, *positive-continuous*, and *piecewise rational linear*.

**Definition 21.** *A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^l$  is superadditive iff  $\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^n, f(\mathbf{a}) + f(\mathbf{b}) \leq f(\mathbf{a} + \mathbf{b})$ .*

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 7.** *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 10 there is no composable CRN which stably computes max.  $\square$

**Definition 22.** *A function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}^l$  is positive-continuous if for all  $U \subseteq [n]$ ,  $f$  is continuous on the domain  $D_U = \{x \in \mathbb{R}_{\geq 0}^n \mid (\forall i \in [n]), x(i) > 0 \iff i \in U\}$ . I.e.,  $f$  is continuous on any subset  $D \subset \mathbb{R}_{\geq 0}^n$  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 unnecessary in this work, particularly because the additional constraint of positive-continuity gives enough restriction.

**Definition 23.** A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is rational linear if there exists  $a_1, \dots, a_n \in \mathbb{Q}$  such that  $f(x) = \sum_{i=1}^n a_i x(i)$ . A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is piecewise rational linear if there is a finite set of partial rational linear functions  $f_1, \dots, f_p : \mathbb{R}^n \rightarrow \mathbb{R}$  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  $f_1, \dots, f_p$  the components of  $f$ .

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

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

The function is superadditive since for all input vectors  $\mathbf{a} = (a_1, a_2, a_3)$ ,  $\mathbf{b} = (b_1, b_2, b_3)$ , 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  $f(\mathbf{a}) + f(\mathbf{b}) = \min(a_1, a_2) + b_1 + b_2 \leq a_1 + a_2 + b_1 + b_2 = f(\mathbf{a} + \mathbf{b})$ . 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  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  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 8.** [Proven in [4]] If a function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  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 10 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.

**Lemma 9.** Given states  $\mathbf{a}, \mathbf{b}, \mathbf{c}$ , if  $\mathbf{a} \rightarrow \mathbf{b}$  then  $\mathbf{a} + \mathbf{c} \rightarrow \mathbf{b} + \mathbf{c}$ .

*Proof.* Adding species cannot prevent reactions from occurring. Thus, we can consider the series of reactions where  $\mathbf{c}$  doesn't react to reach the state  $\mathbf{b} + \mathbf{c}$  from the state  $\mathbf{a} + \mathbf{c}$ .  $\square$

We now utilize this lemma to prove that composable computable functions must be superadditive.

**Lemma 10.** If a function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  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  $\mathbf{x}_1, \mathbf{x}_2$ ,  $\exists \mathbf{o}_1, \mathbf{o}_2$  such that  $\mathbf{x}_1 \rightarrow \mathbf{o}_1$  with  $\mathbf{o}_1(Y) = f(\mathbf{x}_1)$  and  $\mathbf{x}_2 \rightarrow \mathbf{o}_2$  with  $\mathbf{o}_2(Y) = f(\mathbf{x}_2)$ . Consider  $\mathcal{C}$  on input  $\mathbf{x}_1 + \mathbf{x}_2$ . By Lemma 9,  $\mathbf{x}_1 + \mathbf{x}_2 \rightarrow \mathbf{o}_1 + \mathbf{x}_2$ , and again by Lemma 9,  $\mathbf{o}_1 + \mathbf{x}_2 \rightarrow \mathbf{o}_1 + \mathbf{o}_2$ . Looking at the concentration of output species  $Y$ , we have  $(\mathbf{o}_1 + \mathbf{o}_2)(Y) = f(\mathbf{x}_1) + f(\mathbf{x}_2)$ . Since  $\mathcal{C}$  stably computes  $f$ , there exists an output stable state  $\mathbf{o}'$  reachable from initial state  $\mathbf{x}_1 + \mathbf{x}_2$  and reachable from state  $\mathbf{o}_1 + \mathbf{o}_2$ , with  $\mathbf{o}'(Y) = f(\mathbf{x}_1 + \mathbf{x}_2)$ . Since  $\mathcal{C}$  is composable, we can assume that species  $Y$  does not appear as a reactant without loss of generality by Lemma 6. Thus the concentration of  $Y$  in any state reachable from state  $\mathbf{o}_1 + \mathbf{o}_2$  cannot be reduced from  $f(\mathbf{x}_1) + f(\mathbf{x}_2)$ , implying  $\mathbf{o}'(Y) = f(\mathbf{x}_1 + \mathbf{x}_2) \geq f(\mathbf{x}_1) + f(\mathbf{x}_2)$ . This holds for all input states  $\mathbf{x}_1, \mathbf{x}_2$ , and thus  $f$  is superadditive.  $\square$

**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 [11] 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  $Y^+ + Y^- \rightarrow \emptyset$  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 10, 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 24.** An open ray  $\ell$  in  $\mathbb{R}^n$  from the origin through a point  $\mathbf{x}$  is the set  $\ell = \{\mathbf{y} \in \mathbb{R}^n \mid \mathbf{y} = t \cdot \mathbf{x}, t \in \mathbb{R}_{>0}\}$ . Note that  $t$  is strictly positive, so the origin is not contained in  $\ell$ .

**Definition 25.** We call a subset  $D \subseteq \mathbb{R}^n$  a cone if for all  $\mathbf{x} \in \mathbb{R}^n$ , we know that  $\mathbf{x} \in D$  implies the open ray from the origin through  $\mathbf{x}$  is contained in  $D$ .

**Lemma 11.** Suppose we are given a continuous piecewise rational linear function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ . 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 a proof in the appendix.

**Lemma 12.** *Any superadditive continuous piecewise rational linear function  $f : \mathbb{R}_{>0}^n \rightarrow \mathbb{R}_{\geq 0}$  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 11, 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  $\mathbf{x} \in \mathbb{R}_{>0}^n$  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  $|\mathbf{x}|$ , so there exist points  $\mathbf{y}, \mathbf{z} \in D_j$  such that  $\mathbf{y} + \mathbf{x} = \mathbf{z}$ . By the superadditivity of  $f$ , the linearity of  $g_j$ , and the fact that  $\mathbf{y}, \mathbf{z} \in D_j$ , we see:

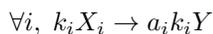
$$g_j(\mathbf{y}) + f(\mathbf{x}) = f(\mathbf{y}) + f(\mathbf{x}) \leq f(\mathbf{z}) = g_j(\mathbf{x} + \mathbf{y}) = g_j(\mathbf{y}) + g_j(\mathbf{x})$$

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

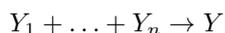
Lemma 12 is particularly useful for us since, as seen in the introduction, CRCs computing min are easy to construct, and rational linear functions are relatively straightforward as well. The next lemma gives details on constructing a CRC to compute  $f$  by piecing together CRCs which compute the components (rational linear functions) of  $f$  and then computing the min across their outputs. However, since Lemma 12 as given applies to continuous functions with domain  $\mathbb{R}_{>0}^n$ , so does this lemma; we handle the domain  $\mathbb{R}_{\geq 0}^n$  later on.

**Lemma 13.** *We can construct a composable CRC that stably computes any superadditive continuous piecewise rational linear function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ .*

*Proof.* By Lemma 12, 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  $g(\mathbf{x}) = a_1x_1 + a_2x_2 + \dots + a_nx_n$  is stably computed by the reactions



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  $X_1 \rightarrow X_1^1 + \dots + X_1^5$  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



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 12, 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.  $\square$

The above construction only handles the domain  $\mathbb{R}_{>0}^n$ , where we know our functions are continuous by positive-continuity. However, when extended to the domain  $\mathbb{R}_{\geq 0}^n$ , 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 15 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 14.** *Consider any superadditive positive-continuous piecewise rational linear function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ . Write  $N = [n]$ , and for each  $S \subseteq N$ , let  $g_S(\mathbf{x})$  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  $\mathbf{x} \in D_S$  we know  $g_S(\mathbf{x}) \leq g_T(\mathbf{x})$ .*

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

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

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

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

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

A naive approach might be the following: for each subdomain  $D_S$ , the function is continuous, so compute it by CRC according to Lemma 13, 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  $P_T$  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  $g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x})$ , 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 14 says that  $g_S(\mathbf{x}) \geq g_I(\mathbf{x})$  (thus the min of these is  $g_I(\mathbf{x})$ ) or (2) the summation term added to  $g_S(\mathbf{x})$  contains at least  $g_I(\mathbf{x})$ , and since  $g_S(\mathbf{x}) + g_I(\mathbf{x}) \geq g_I(\mathbf{x})$ , the min of these is  $g_I(\mathbf{x})$ . Thus taking the min for all  $S$  of  $g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x})$  is exactly  $g_I(\mathbf{x})$ , where  $I$  is the correct set of initially present input species.

**Lemma 15.** Consider any superadditive positive-continuous piecewise rational linear function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ . Write  $N = [n]$ , and for each  $S \subseteq N$ , let  $g_S(\mathbf{x})$  be the superadditive continuous piecewise rational linear function that is equal to  $f$  on  $D_S$ . Then,  $f(\mathbf{x}) = \min_{S \subseteq N} [g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x})]$ .

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

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

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

Case 1:  $S \not\supseteq I$

In this case,

$$\begin{aligned} h_S(\mathbf{x}) &= g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x}) \\ &\geq g_S(\mathbf{x}) + P_I(\mathbf{x})g_I(\mathbf{x}) \\ &\geq P_I(\mathbf{x})g_I(\mathbf{x}). \end{aligned}$$

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

Case 2:  $S \supseteq I$

By Lemma 14,  $g_S(\mathbf{x}) \geq g_I(\mathbf{x})$ . As a result,

$$\begin{aligned} h_S(\mathbf{x}) &= g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x}) \\ &\geq g_S(\mathbf{x}) \geq g_I(\mathbf{x}) \\ &= f(\mathbf{x}). \end{aligned}$$

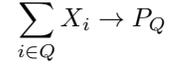
Since for all  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$ , we know  $h_S(\mathbf{x}) \geq f(\mathbf{x})$  for all  $S \subseteq N$  and  $h_I(\mathbf{x}) = f(\mathbf{x})$  for some  $I \subseteq N$ , it follows that  $f(\mathbf{x}) = \min_{S \subseteq N} h_S(\mathbf{x})$ .  $\square$

Lemma 16 takes the above Lemma 15 along with the construction which stably computes on strictly continuous

domains from Lemma 13 to construct a CRC which stably computes on positive-continuous domains.

**Lemma 16.** Given any superadditive positive-continuous piecewise rational linear function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ , 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 15) which are computable by output-oblivious CRCs and are thus composable by Lemma 4. By Lemma 15, we know that  $f(\mathbf{x}) = \min_{S \subseteq N} [g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x})]$ . The first subroutine copies the input species, e.g.  $X_1 \rightarrow X_1^1 + \dots + X_1^5$ , 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:

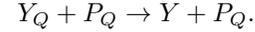


noting that although the predicate  $P_Q(x)$  is defined to be 0 or 1, it is sufficient in this construction for the concentration of the species representing  $P_Q(x)$  to be zero or nonzero. This CRC is output-oblivious.

We can also compute each  $g_Q(\mathbf{x})$  (via Lemma 13) 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:



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.,  $X_1 \rightarrow Y, X_2 \rightarrow Y$  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  $\min_{S \subseteq N} [g_S(\mathbf{x}) + \sum_{K \not\subseteq S} P_K(\mathbf{x})g_K(\mathbf{x})]$ , which is equal to  $f(\mathbf{x})$  by Lemma 15.  $\square$

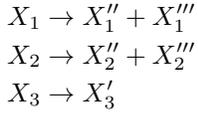
**Corollary 2.** Given any superadditive positive-continuous piecewise rational linear function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$ , there exists a composable CRC with reactions with at most two reactants and at most two products which stably computes  $f$ .

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  $X_1 + \dots + X_n \rightarrow Y_1 + \dots + Y_n$  can be decomposed into the reactions  $X_1 + X_2 \rightarrow X_{12}, X_{12} + X_3 \rightarrow X_{123}, \dots, X_{123\dots n-1} + X_n \rightarrow Y_{12\dots n-1} + Y_n, Y_{12\dots n-1} \rightarrow Y_{12\dots n-2} + Y_{n-1}, \dots, Y_{12} \rightarrow Y_1 + Y_2$ . We can verify that each affected module stably computes correctly with these expanded systems of reactions, and remains composable.

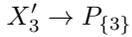
## 4 EXAMPLE

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.

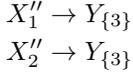
Making copies of input:



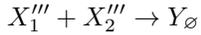
Using  $X_3'$  to make  $P_3$ , which catalyzes reactions for the domain  $X_3 > 0$ :



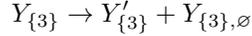
Computing the sum in  $Y_{\{3\}}$ :



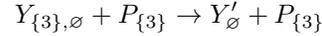
Computing the min in  $Y_{\emptyset}$ :



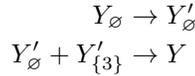
Making a copy of  $Y_{\{3\}}$  for use in increasing  $Y_{\emptyset}'$ :



Increase  $Y_{\emptyset}'$  so that it will not be the min when  $x_3$  is present:



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



## 5 FUNCTIONS COMPUTABLE BY COMPOSABLE CRNS WITH INITIAL CONTEXT

So far, our CRCs 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.

**Definition 27.** A CRC with initial context is denoted  $\mathcal{C}^{I,\mathbf{i}}$  =  $(\Lambda, R, \Sigma, Y, I, \mathbf{i})$  with  $\Lambda, R, \Sigma$ , and  $Y$  defined as in Definition 13, and the initial context species  $I \subset \Lambda \setminus (\Sigma \cup Y)$  and initial context concentrations  $\mathbf{i} \in \mathbb{R}_{\geq 0}^I$ .  $\mathcal{C}^{I,\mathbf{i}}$  stably computes  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  if, for all  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$  and all  $\mathbf{c}$  such that  $\mathbf{x} + \mathbf{i} \rightarrow \mathbf{c}$ , there exists an output stable state  $\mathbf{o}$  such that  $\mathbf{c} \rightarrow \mathbf{o}$  and  $\mathbf{o}(Y) = f(\mathbf{x})$ .

We will show that such CRCs can compute exactly what we could already compute when we set one of our inputs to one. Note that this results in a larger class of computable functions than superadditive positive-continuous piecewise rational linear. For example, we can now compute the non superadditive function  $f(x_1) = 1$ . To formalize this idea, we will also need the following definition.

**Definition 28.** Consider a piecewise affine (linear with an offset) function  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  defined by  $g_i$  on domain  $D_i$ . The linear extension  $f' : \mathbb{R}_{\geq 0}^n \times \mathbb{R}_{> 0} \rightarrow \mathbb{R}_{\geq 0}$  of  $f$  is constructed as follows: for all  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n, \gamma \in \mathbb{R}_{> 0}$  with  $\mathbf{x}/\gamma \in D_i$ , let  $f'(\mathbf{x}, \gamma) = g_i(\mathbf{x}) - g_i(\mathbf{0}) + g_i(\mathbf{0})\gamma$ .

For example, given the piecewise affine function

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

its linear extension would be

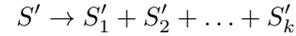
$$f'(x_1, \gamma) = \begin{cases} \gamma & x_1 > \gamma \\ x_1 & x_1 \leq \gamma \end{cases}.$$

Observe that  $f'$  is just the minimum of  $x_1$  and  $\gamma$ . Intuitively the linear extension replaces all constant terms with a linear term in the new variable  $\gamma$ . We show that initial context for composable CRCs allows only functions whose linear extensions are superadditive, positive-continuous, piecewise rational linear. Intuitively, this gives us the same class of functions that we would have without initial context when we set one of our inputs to the constant value one.

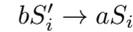
As defined, we allow an arbitrary number of initial context species with differing initial concentrations, but we will focus on the single species case with an initial concentration of one. This is well motivated: we show one initial context species with concentration one is equivalent in stable computing power to having any number of species with nonnegative rational initial concentrations.

**Lemma 17.** Given a CRC with initial context  $\mathcal{C}^{I,\mathbf{i}}$  with  $\mathbf{i} \in \mathbb{Q}_{\geq 0}^I$  (rational initial concentrations) which stably computes  $f$ , there exists a CRC  $\mathcal{C}^{I',\mathbf{i}'}$  with  $I' = \{S'\}$  and  $\mathbf{i}'(S') = 1$  (one initial species with concentration one) which stably computes  $f$ .

*Proof.* Let  $q_i = \frac{a_i}{b_i}$  for  $a_i, b_i \in \mathbb{Z}_{\geq 0}$  be the initial (rational) concentrations of the initial context in  $\mathcal{C}^{I,\mathbf{i}}$ . Observe that the CRN:



can be used to produce  $k$  species with concentrations equal to  $S'$ 's initial concentration. Then the CRN:



for each  $i$  produces a concentration of  $q_i$  for species  $S_i$ .  $\square$

While this schema cannot be used to generate initial context with irrational concentrations, continued fractions can be used to approximate irrational numbers as rational numbers with arbitrarily small error. Thus our restriction to one initial species with a initial concentration one is reasonable to consider for stable computation in this model. To characterize the functions stably computable with initial context, we first prove some lemmas. Recall  $\text{Post}(\mathbf{c})$  is the set of states reachable from  $\mathbf{c}$ , i.e.,  $\{\mathbf{d} \mid \mathbf{c} \rightarrow \mathbf{d}\}$ .

**Lemma 18.** Given a CRN  $\{\Lambda, R\}$ , for any  $\gamma \in \mathbb{R}_{\geq 0}$  and  $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda$ ,  $\text{Post}(\gamma\mathbf{c}) = \gamma\text{Post}(\mathbf{c})$ .

*Proof.* If  $\mathbf{c} \rightarrow \mathbf{d}$ , then for any  $k \in \mathbb{R}_{\geq 0}, k\mathbf{c} \rightarrow k\mathbf{d}$ . This can be verified by taking the straight line segments to get from  $\mathbf{c}$  to  $\mathbf{d}$  and scaling them by  $k$ .

When  $\gamma = 0$ , this lemma is trivial, as the only state reachable from the zero state is the zero state and zero times any state yields the zero state. Thus we only need to consider the case where  $\gamma > 0$ .

Let  $\mathbf{v} \in \text{Post}(\gamma\mathbf{c})$ . By the definition of  $\text{Post}$  this implies that  $\gamma\mathbf{c} \rightarrow \mathbf{v}$ . This implies that  $\mathbf{c} \rightarrow \frac{1}{\gamma}\mathbf{v}$ . Therefore  $\frac{1}{\gamma}\mathbf{v} \in \text{Post}(\mathbf{c})$  and  $\mathbf{v} \in \gamma\text{Post}(\mathbf{c})$ .

Let  $\mathbf{v} \in \gamma\text{Post}(\mathbf{c})$ . By the definition of  $\text{Post}$  this implies that  $\mathbf{c} \rightarrow \frac{1}{\gamma}\mathbf{v}$ . This implies that  $\gamma\mathbf{c} \rightarrow \mathbf{v}$ . Which implies that  $\mathbf{v} \in \text{Post}(\gamma\mathbf{c})$ .  $\square$

**Lemma 19.** Let  $\mathcal{C}^{I,i}$  with  $I = \{S_1\}$  and  $\mathbf{i}(S_1) = q_1$  stably compute  $f$ . Then  $\mathcal{C}^{I,\gamma\mathbf{i}}$  for  $\gamma \in \mathbb{R}_{>0}$  stably computes some function  $g$ .

*Proof.* Consider running  $\mathcal{C}$  with an initial concentration of  $q_1' = \gamma q_1$  for the species  $S_1$ . Observe that the initial state  $\mathbf{x}' = \frac{1}{\gamma}\mathbf{x}$ , where  $\mathbf{x}$  is a state that will stably compute  $f$  under the definition of  $\mathcal{C}$ . By Lemma 18, we know that the set of reachable states from  $\mathbf{x}'$  is equal to the set of reachable states from  $\mathbf{x}$  scaled by  $\gamma$ . Thus consider some state  $\mathbf{c}'$  reached from  $\mathbf{x}'$ . Observe that there exists a state  $\mathbf{c}$  reachable from  $\mathbf{x}$  such that  $\mathbf{c}' = \gamma\mathbf{c}$ . Consider some output stable state  $\mathbf{o}$  reachable from  $\mathbf{c}$ . Observe that by Lemma 18 the state  $\mathbf{o}' = \gamma\mathbf{o}$  must be reachable from  $\mathbf{c}'$ . Likewise by Lemma 18 we know that  $\mathbf{o}'$  must be an output stable state. Thus, we know that  $\mathcal{C}$  must stably compute some function regardless of the initial value for  $S_1$ .  $\square$

We thus know that scaling the value of the initial context retains the fact that  $\mathcal{C}$  stably computes a function in the region where that species has a positive concentration. We can then view a single species of initial context as an additional input to  $\mathcal{C}$  and claim that this CRN stably computes some function, using lemmas from the case of no initial context to prove properties of that function.

**Lemma 20.** Let  $\mathcal{C}^{I,i}$  be output-oblivious with  $I = \{S_1\}$  and  $\mathbf{i}(S_1) = 1$ .  $\mathcal{C}^{I,i}$  stably computes  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  only if  $f$  is a piecewise affine function whose linear extension is superadditive positive-continuous piecewise rational linear.

*Proof.* Intuitively, we treat the initial context as an input species. For  $\mathcal{C}^{I,i} = (\Lambda, R, \Sigma, Y, I, \mathbf{i})$ , let  $\mathcal{C}' = (\Lambda, R, \Sigma \cup I, Y)$ . By assumption,  $\mathcal{C}^{I,i}$  is output-oblivious, so  $\mathcal{C}'$  is output-oblivious. Further, by Lemma 19,  $\mathcal{C}'$  stably computes some function for all (positive) initial concentrations of  $S_1$ . So  $\mathcal{C}'$  must stably compute a superadditive, positive-continuous, piecewise rational linear function  $f'$  on all domains with positive (nonzero) initial concentration of  $S_1$ . Since  $\mathcal{C}^{I,i}$  and  $\mathcal{C}'$  share the same CRN, output species, and input species other than  $S_i$ , we know that for any  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$ ,  $f'(\mathbf{x}, 1) = f(\mathbf{x})$ . All that remains is to show that  $f'$  is the linear extension of  $f$ . Pick any  $\gamma \in \mathbb{R}_{>0}$ ,  $\mathbf{x} \in \mathbb{R}_{\geq 0}^n$  such that  $\mathbf{x}/\gamma \in D_i$ . Let  $g_i$  be the affine function used by  $f$  on  $D_i$ . Since  $f'(\mathbf{x}/\gamma, 1) = f(\mathbf{x}/\gamma)$  and the domains of  $f'$  are cones by Lemma 11,  $f'(\mathbf{x}, \gamma) = \gamma g_i(\mathbf{x}/\gamma)$ . Expanding out the right hand side gives us that  $\gamma f'(\mathbf{x}/\gamma, 1) = g_i(\mathbf{x}) - g_i(\mathbf{0}) + \gamma g_i(\mathbf{0})$ . By Definition 28  $f'$  is the linear extension of  $f$ .  $\square$

**Theorem 2.** Let  $\mathcal{C}^{I,i}$  be output-oblivious with  $\mathbf{i} \in Q_{\geq 0}^I$ . Then  $\mathcal{C}^{I,i}$  stably computes  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$  if and only if  $f$  is a rational affine function whose linear extension is super-additive, positive-continuous, piecewise rational linear.

*Proof.* Lemma 20 gives us the necessary conditions for composable CRCs with initial context. Now all we need to do is show that there is a composable CRC for any rational affine function  $f$  whose linear extension is superadditive, positive-continuous, piecewise rational linear. Since the linear extension of  $f$  is superadditive, positive-continuous, piecewise rational linear, Lemma 16 gives a CRC that computes this function. Observe that changing the input in this CRC representing the linear extension ( $\gamma$ ) to the initial context species will give us a new CRC that computes  $f$ .  $\square$

## 6 DISCUSSION

Instead of continuous concentrations of species, one may consider discrete counts. This changes which functions are stably computed by CRNs. Without the composability constraint, [3] shows in the discrete model that a function  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  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 composable computable functions must be superadditive (Lemma 10) holds for the discrete model as well. Additionally, there exist functions which are superadditive and semilinear but are not computable in the discrete model by a composable CRN. For example (the proof is omitted):

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

so the class of composable computable functions is slightly more restricted.

Three recent works characterized output-oblivious computation in the discrete model. With initial context, the characterization for functions on two inputs ( $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ ) was given by [6], and subsequently extended to arbitrarily many inputs [14]. More recently, [10] showed that without initial context, computable functions must be superadditive in addition to the constraints presented in [14].

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  $t \rightarrow \infty$ ) 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, [12] 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] Ben Chugg, Hooman Hashemi, and Anne Condon. Output-oblivious stochastic chemical reaction networks. In *22nd International Conference on Principles of Distributed Systems, OPODIS 2018, December 17-19, 2018, Hong Kong, China*, pages 21:1–21:16, 2018.
- [7] Domitilla Del Vecchio, Alexander J Ninfa, and Eduardo D Sontag. Modular cell biology: retroactivity and insulation. *Molecular systems biology*, 4(1):161, 2008.
- [8] 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.
- [9] 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.
- [10] Hooman Hashemi, Ben Chugg, and Anne Condon. Composable computation in leaderless, discrete chemical reaction networks. In *26th International Conference on DNA Computing and Molecular Programming (DNA 26)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- [11] Sergei Ovchinnikov. Max-min representation of piecewise linear functions. *Contributions to Algebra and Geometry*, 43(1):297–302, 2002.
- [12] 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.
- [13] Alexander Schrijver. *Theory of Linear and Integer Programming*, pages 85–88. John Wiley & Sons, Inc., New York, NY, USA, 1986.
- [14] Eric E Severson, David Haley, and David Doty. Composable computation in discrete chemical reaction networks. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 14–23, 2019.
- [15] 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.
- [16] Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358(6369):eaal2052, 2017.

## 7 APPENDIX

Most definitions and lemmas in this section work towards proving Lemma 5. We also include a proof of Lemma 11.

**Definition 29.** A polyhedron is a subset of  $\mathbb{R}^n$  of the form  $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$  for some  $m \times n$  matrix  $A$  and some vector  $\mathbf{b} \in \mathbb{R}^m$ .

**Definition 30.** A convex polytope is the convex hull of a finite set of points in  $\mathbb{R}^n$ .

**Definition 31.** A polyhedral cone is a set of the form  $\{\mathbf{x} = \lambda_1 \mathbf{x}_1 + \dots + \lambda_n \mathbf{x}_n \mid \lambda_1, \dots, \lambda_n \geq 0\}$  for some finite set of points  $\{x_1, \dots, x_n\}$  in  $\mathbb{R}^n$

The following lemma comes from a previous work. Note that this sum is the Minkowski sum.

**Lemma 21.** [Proven in [13]] A subset  $P \subseteq \mathbb{R}^n$  is a polyhedron if and only if  $P = Q + C$  for some convex polytope  $Q$  and some polyhedral cone  $C$ .

**Lemma 22.** For a given state  $\mathbf{c}$  of a CRN  $\mathcal{C}$ , the set of states  $\{\mathbf{d} \mid \mathbf{c} \rightarrow^1 \mathbf{d}\}$  that are straight-line reachable from  $\mathbf{c}$  is a polyhedron.

*Proof.* If  $m = |R|$  is the number of reactions in  $\mathcal{C}$  and  $n = |\Lambda|$  is the number of species in  $\mathcal{C}$ , then the stoichiometry matrix can be thought of as a linear transformation from the reaction space  $\mathbb{R}^m$  to the species space  $\mathbb{R}^n$ . Let  $\mathcal{R}_{\mathbf{c}}$  be the set of basis vectors corresponding to reactions applicable at  $\mathbf{c}$ . Then since  $M$  is a linear transformation, it sends the polyhedral cone defined by  $\mathcal{R}_{\mathbf{c}}$  to a polyhedral cone  $C'$  in  $\mathbb{R}^n$ . By Lemma 21, the set  $\mathbf{c} + C'$  is a polyhedron in  $\mathbb{R}^n$ , and since the set of states that are straight-line reachable from  $\mathbf{c}$  is the intersection of this polyhedron with the set of all vectors with nonnegative components, it is also a polyhedron.  $\square$

**Definition 32.** The set of possible species produced from a state  $\mathbf{c}$  is

$$\mathcal{P}(\mathbf{c}) = \bigcup_{\mathbf{d} \in \text{Post}(\mathbf{c})} [\mathbf{d}]$$

**Lemma 23.** For a CRN the set of reachable states is closed under convex combination.

*Proof.* Consider some state  $\mathbf{c}$  and states  $S = \{\alpha_1, \dots, \alpha_k\}$  reachable from  $\mathbf{c}$ . Let  $\mathbf{d} = \sum_{i=0}^k a_i \alpha_i$  where  $\forall i a_i > 0$  and  $\sum_{i=0}^k a_i = 1$ . By lemma 18 we know that  $\forall i a_i \alpha_i$  is reachable from  $a_i \mathbf{c}$ . Since  $\sum_{i=0}^k a_i \mathbf{c} = \mathbf{c}$ , we know that  $\mathbf{c} \rightarrow \sum_{i=0}^k a_i \alpha_i$ , which is equal to  $\mathbf{d}$ .  $\square$

**Lemma 24.** Given a state  $\mathbf{c}$ , there is a state  $\mathbf{d}$  reachable from  $\mathbf{c}$  such that  $\mathcal{P}(\mathbf{c}) = [\mathbf{d}]$ . For such a state,  $\mathcal{P}(\mathbf{d}) = \mathcal{P}(\mathbf{c})$ .

*Proof.* If  $\mathbf{c}$  is the zero vector, observe that  $\mathcal{P}(\mathbf{c}) = [\mathbf{c}]$ , so setting  $\mathbf{d} = \mathbf{c}$  makes this hold. Otherwise, for each species  $S \in \mathcal{P}(\mathbf{c})$ , there is some state  $\mathbf{d}_S$  reachable from  $\mathbf{c}$  with  $S \in [\mathbf{d}_S]$ . Then the state  $\mathbf{d} = \frac{1}{|\mathcal{P}(\mathbf{c})|} \sum_{S \in \mathcal{P}(\mathbf{c})} \mathbf{d}_S$  is reachable from  $\mathbf{c}$  by lemma 23. Since  $\mathbf{d}$  contains a positive contribution from each  $\mathbf{d}_S$ ,  $\mathcal{P}(\mathbf{c}) = [\mathbf{d}]$ . Since  $\mathbf{c} \rightarrow \mathbf{d}$  we know that  $\mathcal{P}(\mathbf{d}) \subseteq \mathcal{P}(\mathbf{c})$ . Since  $\mathcal{P}(\mathbf{c}) = [\mathbf{d}]$  and  $[\mathbf{d}] \subseteq \mathcal{P}(\mathbf{d})$  we know that  $\mathcal{P}(\mathbf{c}) \subseteq \mathcal{P}(\mathbf{d})$ . Thus we can conclude that  $\mathcal{P}(\mathbf{d}) = \mathcal{P}(\mathbf{c})$ .  $\square$

**Lemma 25.** If  $\mathbf{c}$  is a state such that  $\mathcal{P}(\mathbf{c}) = [\mathbf{c}]$ , then any state  $\mathbf{d}$  that is reachable from  $\mathbf{c}$  is straight-line reachable from  $\mathbf{c}$ .

*Proof.* Since  $\mathcal{P}(\mathbf{c}) = [\mathbf{c}]$ , by the definition of  $\mathcal{P}(\mathbf{c})$  we know that the set of applicable reactions from  $\mathbf{c}$  is a superset of those applicable from any state reachable from  $\mathbf{c}$ . Thus we can take the sum of all the straight-line segments used to reach  $\mathbf{d}$  from  $\mathbf{c}$  and apply them all in a single straight-line segment to get  $\mathbf{c} \rightarrow^1 \mathbf{d}$ .  $\square$

**Lemma 5.** For any state  $\mathbf{c}$  and any species  $S$ , if the amount of  $S$  present in any state reachable from  $\mathbf{c}$  is bounded above, there is a state  $\mathbf{d}$  reachable from  $\mathbf{c}$  such that for any state  $\mathbf{a}$  reachable from  $\mathbf{d}$ , we know that  $\mathbf{a}(S) \leq \mathbf{d}(S)$ .

*Proof.* By Lemma 24, there is some  $\mathbf{c}_1$  reachable from  $\mathbf{c}$  such that  $[\mathbf{c}_1] = \mathcal{P}(\mathbf{c})$ . By Lemma 22, we know that the states that are straight-line reachable from  $\mathbf{c}_1$  are a polyhedron  $P$ . The linear map  $\mathbb{R}^n \rightarrow \mathbb{R}$  sending  $\mathbf{x} \mapsto \mathbf{x}(S)$  maps  $P$  to some polyhedral subset of  $\mathbb{R}$ —in particular this is a closed subset. Since we assume that the image of this map is bounded above, we know that this subset attains its maximum  $M$ , so there is some  $\mathbf{d} \in P$  with  $\mathbf{d}(S) = M$ . Any state  $\mathbf{a}$  that is reachable from  $\mathbf{d}$  is also reachable from  $\mathbf{c}_1$ , so by Lemma 25 it is contained in  $P$ . As a result,  $\mathbf{a}(S) \leq M = \mathbf{d}(S)$ .  $\square$

**Lemma 11.** Suppose we are given a continuous piecewise rational linear function  $f : \mathbb{R}_{>0}^n \rightarrow \mathbb{R}_{\geq 0}$ . Then we can choose domains for  $f$  which are cones which contain an open ball of non-zero radius.

*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  $\mathbf{x} \in D_k$ . Since the open ray  $\ell_{\mathbf{x}}$  from the origin passing through  $\mathbf{x}$  is contained in  $\mathbb{R}_{>0}^n$ , it is covered by the domains in  $\mathcal{D}$ . If we write any point  $\mathbf{y} \in \ell_{\mathbf{x}}$  in the form  $t \cdot \mathbf{x}$ , then, for each  $i$ , the restriction of  $g_i$  to  $D_i \cap \ell_{\mathbf{x}}$  is of the form  $g_i(t \cdot \mathbf{x}) = \alpha_i t$  for some  $\alpha_i \in \mathbb{R}$ . Since  $\mathbf{x} \in D_k \cap \ell_{\mathbf{x}}$ , we know that  $f(1 \cdot \mathbf{x}) = \alpha_k \cdot 1 = \alpha_k$ .

Now suppose that for some  $s \in \mathbb{R}_{>0}$  we know that  $f(s \cdot \mathbf{x}) \neq \alpha_k s$ . First consider the case where  $s > 1$ . Then define the set  $A = \{t \in [1, s] \mid f(t \cdot \mathbf{x}) = \alpha_k t\}$  and define the set  $B = \{t \in [1, s] \mid f(t \cdot \mathbf{x}) \neq \alpha_k t\}$ . 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  $\lim_{j \rightarrow \infty} t_j = t'$ . As a result, from the continuity of  $f$ , we see that:

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

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  $\lim_{j \rightarrow \infty} s_j = t'$ . 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  $\lim_{j' \rightarrow \infty} s_{j'} = t'$ , so by the continuity of  $f$  and the fact that  $s_{j'} \in D_{k'}$ , we see that:

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

Since  $t' > 0$ , this implies that  $\alpha_{k'} = \alpha_k$ , so that  $f(s_{j'} \cdot \mathbf{x}) = \alpha_k s_{j'}$ . 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  $f(s \cdot \mathbf{x}) \neq \alpha_k s$ , must be false. A similar argument, using the infimum instead of the supremum, shows that there can be no  $s < 1$  such that  $f(s \cdot \mathbf{x}) \neq \alpha_k s$ . As a result, for every point  $t \in \ell_{\mathbf{x}}$ , we know  $f(t \cdot \mathbf{x}) = \alpha_k t$ . In other words,  $f|_{\ell_{\mathbf{x}}} = g_k|_{\ell_{\mathbf{x}}}$ , so we can replace  $D_k$  with  $D_k \cup \ell_{\mathbf{x}}$  without issue. Doing this for every  $\mathbf{x} \in D_k$ , 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  $\mathbf{x} \in D_i$  not in the interior of  $D_i$ . Then  $\mathbf{x}$  is in the closure of the complement of  $D_i$ , so there exists a sequence  $\{\mathbf{x}_k\}_{k=1}^{\infty}$  of points in the complement of  $D_i$  such that  $\lim_{k \rightarrow \infty} \mathbf{x}_k = \mathbf{x}$ . Since the complement of  $D_i$  is covered by the  $D_j \in \mathcal{D}$ , where  $j \neq i$ , we know that each  $\mathbf{x}_k$  lies in one of the  $D_j$ . Since there are only finitely many  $D_j$  but infinitely many  $\mathbf{x}_k$ , we know that infinitely many  $\mathbf{x}_k$  must lie in at least one of the  $D_j$ . As a result,  $\mathbf{x}$  is in the closure of this  $D_j$ , and since  $D_j$  is closed, we see that  $\mathbf{x} \in D_j$ . 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.  $\square$