Abstract

There are many systems in biology that use chemical interactions for decision making. Such molecular computing systems which operate via chemical interactions may be useful to engineers in cases where traditional computing systems cannot be used. The chemical reaction network (CRN) model formalizes the notion of chemical interactions, allowing us to perform rigorous reasoning about the computational power of chemical interactions. In this model we think of a series of chemical reactions as computing a function, where the initial concentration of a subset of the species are the inputs and the final concentration of another species is the output. In this undergraduate thesis (which is intended to be a more intuitive overview of the attached paper), we precisely characterise the set of functions computable by output oblivious CRNs (CRNs that do not use the output species as a reactant) that are rate-independent (they always produces the correct output independent of the reaction rates) as super-additive, positive-continuous, and piecewise rational linear.

1 Introduction

In biology, cells often use chemical regulatory networks to process information, such as to reach a consensus on gene expression. Chemical reaction networks (CRNs) are widely used for modelling this computation. CRNs also provide us with a model under which we can talk about harnessing the power of molecular computation in domains such as nanotechnology, which are fundamentally incompatible with silicon based computing due to size constraints. In 2013 Yuan-Jyue Chen et. al. showed that DNA reactions are capable of simulating arbitrary CRNs and in 2017 François Fages et. al. showed that CRNs are Turing universal under certain assumptions [3, 5].

Despite the general computing power of CRNs, it is often more useful to consider special subsets of them. Normally there are two core programmable aspects to CRNs: the reactions and their rate laws. In the laboratory, rate laws are the
harder of these parameters to control, as they often rely on species being well mixed and are highly temperature dependent. As such, we want to explore the computational power of CRNs regardless of their rate laws, as done in [2]. For example, the reaction

\[ X \rightarrow Y \]

will always turn all of molecule \( X \) into molecule \( Y \) and thus is rate independent. On the other hand, the following CRN is not rate independent:

\[ X \rightarrow Y \]
\[ X \rightarrow Z \]

since the amounts of \( Y \) and \( Z \) that are created depends on the rates of these reactions.

If you want to use a chemical reaction network as a computer to compute a function, you might want to construct modules that compute useful subfunctions. These modules could then be abstracted away, like is often done with circuits in more traditional computing. However, unlike in traditional circuits, composition presents a problem in a chemical computer. A module that computes a function might use its output as a reactant in one of its reactions. If this molecule is also used by another downstream computation, then these modules can interfere with one another, which results in a loss of rate independence. In fact, this is a known issue in actual biological systems. In this paper, we will show how composability restricts the computational power of CRNs in the rate-independent setting. Note that this thesis is meant to be a higher level presentation of the work that I did with Cameron Chalk, Wyatt Reeves, and Dr. David Soloveichik and that the full paper that I wrote with these individuals is included as an appendix here for reference. I will refer to the full paper for details in this thesis.

2 Definitions

2.1 Chemical Reaction Networks

A Chemical Reaction Network (CRN) is defined by a set of chemical species and their interactions. For example, in chemistry you might see a chemical reaction network where the species are hydrogen, carbon, and oxygen that looks like this:

\[ 2H_2 + O_2 \rightarrow 2H_2O \]
\[ 6CO_2 + 6H_2O \rightarrow C_6H_{12}O_6 + 6O_2 \]

For the purposes of this paper, we do not care about the exact chemical processes that are occurring, so we can instead abstract this CRN to the following:

\[ 2A + B \rightarrow 2C \]
Here we have two reactions: one where species $A$ and $B$ are inputs to produce $C$ and another where $D$ and $C$ are inputs to produce $E$ and $B$. We can think about the state of a system under the influence of a CRN as a vector indexed by the species present in that CRN. If we have the state:

<table>
<thead>
<tr>
<th>Species</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
</tbody>
</table>

and then fire the first reaction with concentration $\epsilon_1$, then we will end up in the state

<table>
<thead>
<tr>
<th>Species</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$1 - 2\epsilon_1$</td>
</tr>
<tr>
<td>B</td>
<td>$1 - \epsilon_1$</td>
</tr>
<tr>
<td>C</td>
<td>$2\epsilon_1$</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
</tbody>
</table>

Now if we fire the second reaction with concentration $\epsilon_2$, then we get the state

<table>
<thead>
<tr>
<th>Species</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$1 - 2\epsilon_1$</td>
</tr>
<tr>
<td>B</td>
<td>$1 - \epsilon_1 + 6\epsilon_2$</td>
</tr>
<tr>
<td>C</td>
<td>$2\epsilon_1 - 6\epsilon_2$</td>
</tr>
<tr>
<td>D</td>
<td>$1 - 6\epsilon_2$</td>
</tr>
<tr>
<td>E</td>
<td>$\epsilon_2$</td>
</tr>
</tbody>
</table>

Note that in-order to agree with chemistry, 'firing' a reaction must require that the resulting state has no negative concentrations. It is also possible to talk about firing multiple reactions at the same time with different concentrations, as long as all species keep non-negative concentrations (parallelism). While in actual chemistry all of the reactions occur simultaneously according to rate laws, for the purposes of this paper we will be proving behavior about CRNs regardless of the used rate law. Thus it is okay to think of reactions occurring in sequential steps as we did above. We say that one state of a CRN is reachable from another state if there is a valid way to fire the reactions and end up in that state.

## 2.2 Stable Computation

In order to give meaning to a computer defined by a chemical reaction network, we need to add some special labels to species. For the purposes of this paper, any species labeled $X_i$ will be an input species and the species labeled $Y$ will be
the output species. We think of our CRN as behaving like a relation on $\mathbb{R}^n \to \mathbb{R}$ (note that this easily generalises to relations on $\mathbb{R}^n \to \mathbb{R}^m$), where the initial concentrations of the input species is the "input" and the final concentration of the output species is the "output." But this begs the question: what does it mean for a CRN to have a final concentration? For example, the CRN

$$\emptyset \to Y$$

will infinitely produce $Y$ and thus never obtain a final concentration of the output species. To avoid this problem, we will be restricting ourselves to output stable CRNs. These are CRNs that obey the following rule: From any reachable state, there exists a way to fire the reactions and end in a state where the output concentration will never change (a fixed point). We will call such a state terminal. Note that while the amount of output species never changes in a terminal state, the other species are allowed to change concentrations. We will say that a CRN stably computes a function $f$ if and only if the concentration of the output species is $f(x_1, \ldots, x_n)$ in all reachable terminal states where the initial concentrations of the input species $X_i$ was $x_i$ and the initial concentration of all other species was zero. For future convenience, we will call a CRN with input species $x_1 \ldots x_n$ and output species $Y$ that stably computes some function $f$ a Chemical Reaction Computer (CRC) for $f$.

### 2.2.1 Examples

Here are some examples of CRCs for functions:

<table>
<thead>
<tr>
<th>$f(x_1, x_2)$</th>
<th>$\min(x_1, x_2)$</th>
<th>$x_1 + x_2$</th>
<th>$x_1/2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC</td>
<td>$X_1 + X_2 \to Y$</td>
<td>$X_1 \to Y'$</td>
<td>$2X_1 \to Y$</td>
</tr>
</tbody>
</table>

Intuitively the first reaction must stop when either $X_1$ or $X_2$ reaches a concentration of zero, causing it to compute the function $\min$. The second reaction independently turns all $X_1$ into $Y$ and $X_2$ into $Y$, which will result in a $x_1 + x_2$ concentration of $Y$. The last example takes two $X_1$ to produce $Y_2$, causing it to compute half the input. Note that the last example will compute $x_1/2$ in our model rather than $\lfloor x_1/2 \rfloor$ because we are working with concentrations rather than exact molecular counts, making this model continuous rather than discrete. Sometimes we can combine the ideas in smaller CRCs to compute more complicated functions. For example, here is a CRC for the function $\max(x_1, x_2)$:

$$X_1 \to X_1' + Y$$

$$X_2 \to X_2' + Y$$

$$X_1' + X_2' \to M$$

$$Y + M \to \emptyset$$

This CRN uses the fact that the third reaction will compute the min of $X_1$ and $X_2$ and that $\max(x_1, x_2) = x_1 + x_2 - \min(x_1, x_2)$. 


2.3 Output Oblivious Computation

Looking again at our CRC for max, observe that we used Y as a reactant in one of our reactions. At a first glance, this doesn’t seem like a major problem, as we will eventually end up with the correct amount of Y. However, there are a couple of issues with this design. First of all, if we wanted to use this CRN as a module to compute max in a larger CRN, then some other reaction could compete with the last one, resulting in an incorrect count for Y. Similarly, we will get the wrong answer if we filter the Y out of solution as it is produced, which is a common technique when doing this in the lab. This leads us to the idea of an Output Oblivious CRC. We will call a CRC (with output Y) output oblivious if it contains no essential reactions that use Y as a reactant (by essential, we mean that the CRN will still be a CRC for the same function if we remove those reactions). It turns out that Output Oblivious CRNs are exactly the ones that can be used as modules in larger CRNs without issues. See section 2.3 in the paper for more details about this (note that the paper uses a slightly different definition of output-oblivious, but this one is simpler for our purposes).

At a first glance, output-obliviousness might not look like a strong restriction. We could turn any CRC into a CRN that does not use Y by relabeling Y as Z and then adding the reaction Z \rightarrow Y into our reaction set. Unfortunately this trick will not work in general, since we will lose rate independence when we add this reaction if the original CRC was not output-oblivious. If the CRC was output-oblivious, then we just composed it with the identity function!

3 Functions Computable by Output Oblivious CRNs

3.1 Restrictions on Computation by Output Oblivious CRCs

While the CRC described above for max is not output oblivious, one might wonder if such a CRC can exist. In fact, it is pretty easy to show that there is no rate independent output oblivious CRC for the max function. If such a CRC existed, then there must be a way to fire all of the X_1 into Y without using X_2, since \text{max}(x_1, 0) = x_1. By symmetry there is also a way to fire all of the X_2 into Y without using X_1. Therefore, there is a reachable state that contains an \(x_1 + x_2\) concentration of Y. Since output oblivious CRNs cannot decrease the concentration of Y, there will be no terminal state reachable from this point with the correct amount of Y. It turns out that we can generalize this argument a bit: output oblivious CRNs cannot compute any function \(f\) such that there exists \(x_1, \ldots, x_n, x'_1, \ldots, x'_n\) where \(f(x_1, \ldots, x_n) + f(x'_1, \ldots, x'_n) > f(x_1 + x'_1, \ldots, x_n + x'_n)\). The proof behaves the same way: Since there are independent pathways to convert the first input into \(f(x_1, \ldots, x_n)\) and the second input into \(f(x'_1, \ldots, x'_n)\), we can take both of these pathways to convert the combined input into a concentration of \(f(x_1, \ldots, x_n) + f(x'_1, \ldots, x'_n)\) for Y,
which is more than we should be able to produce while computing $f$ with an output oblivious CRC. We will call this restriction on $f$ superadditivity [1].

From previous works, we know that real valued rate independent CRCs can compute exactly positive-continuous piecewise rational linear functions [2]. Positive-continuous means that $f$ must be continuous on domains where the same set of input species have non-zero concentrations and piecewise rational linear means that $f$ can be described as a collection of linear functions (not affine) with rational coefficients over finite disjoint domains. Combining these requirements gives us our class of functions: superadditive, positive-continuous, piecewise rational linear.

### 3.1.1 Example functions

What exactly do superadditive, positive-continuous, piecewise rational linear functions look like? We already showed that max is not in this class of functions, but min is. Here are some plots of example superadditive, positive-continuous, piecewise rational linear functions:

| $f(x) = \min(x_1, x_2)$ | $f(x) = \begin{cases} 
3x_2 & x_1 > 2x_2 \\
3x_1 & x_2 > 2x_1 \\
x_1 + x_2 & \text{otherwise}
\end{cases}$ | $f(x) = \begin{cases} 
2x_1 & 2x_1 < x_2 \\
x_2 & 2x_1 \geq x_2
\end{cases}$ |

Note that all of the above functions are continuous. Notice that each of these functions is defined as linear functions over conic subdomains. In fact, it is not hard to prove that continuous piecewise rational linear functions always have conic subdomains that contain non-empty balls (see lemma 11 in the paper for details). Functions such as

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

also fit our criteria. While they are not continuous, they can be thought of as multiple continuous functions joined together at the boundaries where inputs switch from zero to positive in value.

### 3.2 Main result

Our main theorem is that the class of functions computable by rate-independent output-oblivious CRNs is exactly the set of superadditive, positive-continuous, piecewise rational linear functions.
3.3 CRC Construction

3.3.1 Properties of continuous piecewise rational linear functions

Now that we have shown that our functions must be superadditive, positive-continuous, piecewise rational linear, we want to show that any such function is computable by an output-oblivious CRC. We will do this by showing that any superadditive, continuous, piecewise rational linear function can be expressed as the minimum of rational linear functions, coming up with a construction for such functions, and then extending the logic to positive-continuous functions. We will start with an observation about continuous piecewise rational linear functions: they can be expressed in a way such that each domain is a cone that contains a non-empty ball over the non-zero inputs in that domain.

Since we know that our domains \( D_1, \ldots, D_m \) of the piecewise function are cones that contain balls of non-zero radius, we know that each domain contains a ball of arbitrary radius. We will say that \( f(\vec{x}) = g_i(\vec{x}) \) if \( \vec{x} \) lies within \( D_i \). We know that each \( g_i \) will be a linear function by definition.

We will sketch the proof that for any input \( \vec{x} \), \( f(\vec{x}) = \min_{i \in [n]} g_i(\vec{x}) \). Without loss of generality, assume that \( \vec{x} \in D_x \). Pick any \( D_y \neq D_x \). Since \( D_y \) is a cone that contains balls of arbitrary radii, it contains a ball of radius \( \|\vec{x}\| \) and thus there is a vector \( \vec{y} \) such that \( \vec{y} \in D_y \) and \( \vec{x} + \vec{y} \in D_y \). By superadditivity, we know that

\[
f(\vec{x}) + f(\vec{y}) \leq f(\vec{x} + \vec{y})
\]

Observing that \( \vec{x} \in D_x \) and \( \vec{y}, \vec{x} + \vec{y} \in D_y \), we can rewrite this as

\[
g_x(\vec{x}) + g_y(\vec{y}) \leq g_y(\vec{x} + \vec{y})
\]

Since \( g_y \) is a linear function, we get that

\[
g_x(\vec{x}) + g_y(\vec{y}) \leq g_y(\vec{x}) + g_y(\vec{y})
\]

and thus that

\[
g_x(\vec{x}) \leq g_y(\vec{x})
\]

Because \( g_x(\vec{x}) \leq g_y(\vec{x}) \) for all \( y \in [n] \), we can conclude that \( f(x) = \min_{i \in [n]} g_i(x) \). Thus any superadditive continuous piecewise rational linear function is the min of the rational linear functions it is composed of.

3.3.2 Construction of continuous piecewise rational linear functions

We can compute any superadditive continuous piecewise rational linear function if we can come up with output oblivious CRNs for the minimum of \( n \) inputs, and for arbitrary rational-linear functions. We can compute min on \( n \) inputs via the following CRC:

\[
X_1 + X_2 + \ldots + X_n \rightarrow Y
\]
Likewise we can construct the rational linear function \((a_1/k_1)x_1 + (a_2/k_2)x_2 + \ldots (a_n/k_n)x_n\) defined with \(a_i, k_i \in \mathbb{Z}_{\geq 0}\) via the CRC:

\[
\forall i \in [n] \; k_i X_i \rightarrow a_i k_i Y
\]

3.3.3 Example superadditive, continuous, piecewise rational linear construction

Let us construct a CRC for the superadditive function

\[
f(x) = \begin{cases} 
3x_2 & x_1 > 2x_2 \\
3x_1 & x_2 > 2x_1 \\
x_1 + x_2 & \text{otherwise}
\end{cases}
\]

Above we proved that this function is actually just \(\min(3x_1, 3x_2, x_1 + x_2)\). Thus the following CRC will compute this function:

\[
\begin{align*}
X_1 & \rightarrow X'_1 + X''_1 \\
X_2 & \rightarrow X'_2 + X''_2 \\
X'_1 & \rightarrow 3Y_1 \\
X'_2 & \rightarrow 3Y_2 \\
X''_1 & \rightarrow Y_3 \\
X''_2 & \rightarrow Y_3 \\
Y_1 + Y_2 + Y_3 & \rightarrow Y
\end{align*}
\]

Here, reactions (1) and (2) serve to copy the value of \(X_1\) and \(X_2\) so that they can be used by multiple equations. Reactions (3) and (4) compute the first two sub-functions and reactions (5) and (6) work together to compute the third sub-function. Finally, reaction (7) computes the minimum of these sub-functions.

3.3.4 Properties of positive-continuous piecewise rational linear functions

Positive continuity creates a bit of an issue for our construction. The proof that the correct value is the min of all components functions breaks, since we no longer have balls of non-zero radii for each domain. Fortunately we can think of a positive-continuous function as a collection of continuous functions on different sets of inputs with positive values. For each \(S \subseteq [n]\), let \(g_S\) be the continuous piecewise rational linear function that is used by \(f\) when \(S\) is the set of positive inputs. Observe that if \(S \subseteq T\), then for any input \(x\), \(g_S(x) \leq g_T(x)\) due to superadditivity. Let us define the boolean function \(P_S(\vec{x}) = 1\) only if for each \(i \in S \setminus \vec{x}\) has a positive value in that index and 0 otherwise. We will want to show that:

\[
f(\vec{x}) = \min_{S \subseteq [n]} \left[ g_S(\vec{x}) + \sum_{T \not\subseteq S} P_T(\vec{x}) g_T(\vec{x}) \right]
\]
Assume that $K \subseteq [n]$ is largest cardinality set such that $P_K(\vec{x}) = 1$.

We will enumerate over the possible $S \subseteq [n]$

If $S = K$, then $g_S(\vec{x}) + \sum_{T \subseteq S} P_T(\vec{x})g_T(\vec{x}) = g_S(\vec{x})$.

If $K \subset S$, then from above we know that $g_K(\vec{x}) \leq g_S(\vec{x})$ and thus this will not be the smallest term in the minimum.

If $K \not\subseteq S$, then $g_S(\vec{x}) + \sum_{T \not\subseteq S} P_T(\vec{x})g_T(\vec{x}) \geq P_K(\vec{x})g_K(\vec{x}) = g_K(\vec{x})$. Thus for all all subsets $S$ of $[n]$, $g_S(\vec{x}) + \sum_{T \not\subseteq S} P_T(\vec{x})g_T(\vec{x}) \geq f(\vec{x})$.

### 3.3.5 Construction of positive-continuous piecewise rational linear functions

In section 3.3.2 we showed how to compute the $g_S$ for each continuous piecewise rational linear function that makes up $f$. Now we will show how these can be combined using the representation in 3.3.4 as an output-oblivious CRC. Again we can use the reaction

$$X_1 + X_2 + \ldots X_m \rightarrow Y$$

to compute a minimum and the CRC

$$\forall i \in [n] X_i \rightarrow Y$$

to compute a sum. All that remains is to show that we can compute the product $P_T(\vec{x})g_T(\vec{x})$ from a CRC that computes $g_T$. If we relabel the output species of the CRC for $g_T$ as $Y_T$, then we can add the reactions

$$\sum_{i \in T} X_i \rightarrow P_T$$

$$Y_T + P_T \rightarrow Y + P_T$$

Since this reaction can only occur when each $i \in T$ has a positive value and we can only turn $Y_1$ into $Y$ when it is present, it will behave like multiplying by the boolean function $P_T$.

This concludes the proof that there is a rate-independent output-oblivious CRC for any superadditive, positive-continuous, piecewise rational linear function.

### 3.3.6 Example positive continuous construction

We will demonstrate how to use the ideas in this section to construct a CRC for the function:

$$f(\vec{x}) = \begin{cases} 
  x_1 + x_2 & x_3 > 0 \\
  \min(x_1, x_2) & x_3 = 0.
\end{cases}$$
This function is superadditive, positive-continuous, and piecewise rational linear. Thus, we can apply our construction to generate a CRC for this function.

\[ X_1 \rightarrow X_1' + X_1'' \]  
\[ X_2 \rightarrow X_2' + X_2'' \]  
\[ X_3 \rightarrow P_{X_3>0} \]  
\[ X_1' \rightarrow Y_{X_3>0} + Y_{X_3>0}' \]  
\[ X_2' \rightarrow Y_{X_3>0} + Y_{X_3>0}' \]  
\[ X_1'' + X_2'' \rightarrow Y_{X_3=0} \]  
\[ Y'_{X_3>0} + P_{X_3>0} \rightarrow Y_{X_3=0} + P_{X_3>0} \]  
\[ Y_{X_3=0} + Y'_{X_3>0} \rightarrow Y \]

At the high level, \( X_1' \) and \( X_1'' \) are used as copies of the original \( X_1 \) for computation of both the sum and the min. \( X_3 \) is relabeled as \( P_{X_3>0} \) for clarity, since we only use the input for the predicate. Reactions (4) and (5) will compute the sum in the variables \( Y_{X_3>0} \) and \( Y'_{X_3>0} \), as we will need a copy of this value later. Reaction (6) computes the min in \( Y_{X_3=0} \). Reaction (7) adds the sum to the min when \( x_3 > 0 \), following the logic described in the above section. Finally, reaction (8) computes the min of the two values in our output variable. Note that while this is a relatively simple example, the ideas used here can be extended to any superadditive, positive-continuous, piecewise rational linear function.

4 Work Since this Research

Since this work has been published, there has been progress made in understanding output-oblivious computation in other chemical models. In 2018 Ben Chugg, Anne Condon, and Hooman Hashemi showed that output-oblivious computable functions \( \mathbb{N}^2 \rightarrow \mathbb{N} \) in the discrete model (where exact molecular counts are used for the inputs and outputs) must be monotonic and either grid-affine (the domains can be congruence classes) or the minimum of fissure functions \[4\]. This is a natural extension of our result to this model, and it was shown by Severson, Haley, and Doty in 2019 that output oblivious functions \( \mathbb{N}^d \rightarrow \mathbb{N} \) in the discrete model can compute exactly monotonic quilt-affine functions (a quilt-affine function is a linear function with a periodic offset) \[7\]. Neither of these results give output-oblivious CRCs much more power than they have in the continuous model.

Both of the papers listed above make use of a leader - some species with a fixed initial quantity of one. In the extended version of our paper (the one attached to this one), we extended our logic to add initial context (the continuous equivalent to a leader). While writing this thesis, I discovered a bug in this proof, which I plan to address in the near future.
5 Future Work

We have shown that output-oblivious computation is severely restricted in both the discrete and continuous models. It turns out that these restrictions can be mitigated by using different output conventions. In the dual rail model of computation, you have two copies of each input and output and the difference between these values gives you the actual value. It is easy to see that in the dual rail model output obliviousness does not affect the class of computable functions (functions that consume Y can instead produce -Y). Another unexplored output convention is fractional encoding, where the ratio of two species gives the associated value. It is unknown what the output-oblivious restriction does to fractionally encoded CRCs [6].

Finally, it would be interesting to explore output obliviousness in rate dependent systems. It would be interesting to see if the ideas in this paper translate and if they do, how actual biological systems get around the issues created by non-output oblivious computation.

References


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 computation. In particular, DNA strand displacement cascades can in principle realize arbitrary CRNs, thus motivating the study of CRNs as a programming language [2], [5], [13], [14]. 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.

\[ X_1 + X_2 \rightarrow Y \]  
\[ X_1 \rightarrow Z_1 + Y \]  
\[ X_2 \rightarrow Z_2 + Y \]  
\[ Z_1 + Z_2 \rightarrow K \]  
\[ Y + K \rightarrow \emptyset. \]

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...
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 \( f : \Lambda \to \mathbb{R} \), and similarly for \( \mathbb{R}_{\geq 0}^\Lambda \), \( \mathbb{N}^\Lambda \), etc. Equivalently, we view an element \( \mathbf{c} \in A^\Lambda \) as a vector of \( |\Lambda| \) elements of \( A \), each coordinate “labeled” by an element of \( \Lambda \). Given a function \( f : A \to B \), we use \( f|_C \) to denote the restriction of \( f \) to the domain \( C \). We also use the notation \( \mathbf{c} \downarrow \Delta \) to represent \( \mathbf{c} \) projected onto \( \mathbb{R}_{\geq 0}^\Delta \). Thus, \( \mathbf{c} \downarrow \Delta = 0 \iff (\forall S \in \Delta) \mathbf{c}(S) = 0 \). If \( \Delta \subseteq \Lambda \), we view a vector \( \mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda \) equivalently as a vector \( \mathbf{c} \in \mathbb{R}_{\geq 0}^\Delta \) 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 \( \alpha = (\mathbf{r}, \mathbf{p}) \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda \), specifying the stoichiometry of the reactants and products, respectively.

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

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.

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 \( C = (\Lambda, R) \), 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 \( C = (\Lambda, R) \) is a vector \( c \in \mathbb{R}^\Lambda_{\geq 0} \).

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

Definition 5. For any \( c \in \mathbb{R}^\Lambda_{\geq 0} \), the set of species present in \( c \) (denoted by \( \{|c|\} \)) is \( \{S \in \Lambda | c(S) > 0\} \).

Definition 6. Given a state \( c \) and reaction \( \alpha = (r, p) \), we say that \( \alpha \) is applicable in \( c \) if \( |r| \subseteq |c| \) (i.e., \( c \) contains positive concentration of all of the reactants).

Definition 7. A reaction produces (consumes) a species \( S \) if \( S \) appears as a product (reactant).

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\(^4\) 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 \( C \) be a CRN defined by \( (\Lambda, R) \). The linear transformation \( M : \mathbb{R}^\Lambda \to \mathbb{R}^\Lambda \) that maps from the unit vector representing a reaction to the net change in species caused by that reaction is the stoichiometry matrix for \( C \).

Note that we can intuitively think of \( M \) being a matrix where the columns represent the net change in species caused by each reaction. Under this representation, observe that entries in \( M \) will be negative when more of a reactant is consumed than is produced in a reaction. Observe that the image of \( M \) 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 \( u \in \mathbb{R}^\Lambda_{\geq 0} \) is a flux vector. We use \( [u] \) to denote the set \( \{r | u(r) > 0\} \). We say that \( u \) is applicable at a state \( c \) if every reaction in \([u]\) is applicable at \( c \).

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

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

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

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 \( d \) is reachable from \( c \), written \( c \to d \), if, for all \( \exists \alpha \subseteq \Lambda \) such that \( d \) is 1-segment reachable from \( c \). We denote the set of states reachable from \( c \), i.e., \( \{d | c \to d\} \), as \( \text{Post}(c) \).

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

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

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

Definition 13. A chemical reaction computer (CRC) is a tuple \( C = (\Lambda, R, \Sigma, Y) \), where \( (\Lambda, R) \) is a CRN, \( \Sigma \subseteq \Lambda \), written as \( \Sigma = \{X_1, \ldots, X_n\} \), is the set of input species, and \( Y \in \Lambda \setminus \Sigma \) is the output species. For simplicity, assume a canonical ordering of \( \Sigma = \{X_1, \ldots, X_n\} \) so that a vector \( x \in \mathbb{R}^\Sigma_{\geq 0} \) (i.e., an input to \( f \)) can be viewed equivalently as a state \( x \in \mathbb{R}^\Lambda_{\geq 0} \) of \( C \) (i.e., an input to \( C \)).

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

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

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
Definition 17. We say two CRCs are composable if they stably compute the function $f$ that their only interface is the output species of $C_1$ and $C_2$, where the concatenation of their chemical reactions, such that the output species of the first is the input species of the second: $f : R^n \rightarrow R^l$, i.e., whose output is a vector of $l$ real numbers. This is because such a function is equivalently $l$ separate functions $f_i : R^n \rightarrow R$.

Also note that initial states contain only the input species $\Sigma_1$; 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 $C_1 = (A_1, R_1, \Sigma_1, Y_1)$ and $C_2 = (A_2, R_2, \Sigma_2, Y_2)$, consider $C'_2 = (A'_2, R'_2, \Sigma'_2, Y'_2)$ constructed by renaming species of $C_2$ such that $A_1 \cap A'_2 = \{Y_1\}$ and $Y_1 \in \Sigma'_2$. The composition of $C_1$ and $C_2$ is the CRC $C_{21} = (A_1 \cup A'_2, R_1 \cup R'_2, \Sigma_1 \cup \Sigma'_2 \setminus \{Y_1\}, Y'_2)$. In other words, the composition is constructed by concatenating $C_1$ and $C_2$ such that their only interface is the output species of $C_1$, used as the input for $C_2$.

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

Definition 17. A CRC $C_1$ which stably computes $f_1$ is composable if $\forall C_2$ stably computing $f_2$, $C_{21}$ 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 $u$ produces (consumes) a species $S$ if there is an $r \in [u]$ such that $S$ is a product (reactant) of $r$. A flux vector $u_1$ is independent of a flux vector $u_2$ if $u_1$ does not consume any species that are produced or consumed by $u_2$.

Lemma 2. In a CRC $C = (\Lambda, R)$ if flux vector $u_1$ is independent of flux vector $u_2$ then:

1) If $a \rightarrow_{u_2} b \rightarrow u_1$, then $a \rightarrow_{u_1 + u_2} c$.
2) If $a \rightarrow_{u_1 + u_2} c$, then there is a state $b$ such that $a \rightarrow_{u_1} b \rightarrow u_2 c$.
3) If $a \rightarrow_{u_1} b$ and $a \rightarrow_{u_2} c$, then there is a state $d$ such that $c \rightarrow_{u_1} d$.

Proof. For 1, since $u_1$ is independent of $u_2$, we know that $u_2$ cannot produce any of the species necessary to make $u_1$ applicable. Since $u_1$ was applicable at $b$, we know that $u_1$ must be applicable at $a$, so $a \rightarrow_{u_1 + u_2} c$.

For 2, we need to show that $b \in R^d_{\leq 0}$ has nonnegative concentrations) and that $u_2$ is applicable at $b$. Consider any species $S \in \Lambda$ such that $S$ is not produced in $u_2$. Since $b = c - Mu_2$ and $c(S) \geq 0$, we know that $S$ has nonnegative concentration at $b$. Now consider any species $S \in \Lambda$ such that $S$ is produced in $u_2$. Since $u_2$ produces $S$, we know that $u_1$ must not consume it because $u_1$ is independent of $u_2$. Since $b = a + Mu_1$, we can conclude that $S$ must have nonnegative concentration at $b$. Therefore we can conclude that $b \in R^d_{\leq 0}$. To see that $u_2$ is applicable at $b$, first observe that $u_2$ is applicable at $a$, since $u_1$ is independent of $u_2$, it follows that $u_2$ must also be applicable at $b$.

For 3, we want to show that $u_1$ is still applicable at $c$ and that $d = c - Mu_1$ is in $R^d_{\geq 0}$ has nonnegative concentrations). $u_1$ is still applicable at $c$ since $u_1$ does not consume species involved in $u_2$. If $d$ had negative concentration on species $S$, that species must have been consumed by $u_1$ since $c$ has nonnegative concentrations. Since $u_2$ does not consume any species consumed by $u_1$ and $c = a + Mu_1$, then negative $d(S)$ implies negative $c(S)$, which contradicts that $c \in R^d_{\geq 0}$, so $d \in R^d_{\geq 0}$.

Lemma 3. Given two output-oblivious CRCs $C_1$ and $C_2$, consider the composition CRC $C_{21}$. If $c \rightarrow d$ then there is a $b$ such that $c \rightarrow b \rightarrow d$, where $c \rightarrow b$ only uses reactions from $C_1$ and $b \rightarrow d$ only uses reactions from $C_2$.

Proof. Let $\{v_1, \ldots, v_n\}$ be the flux vectors such that $c \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n$. We can write $v_i = u_{1,i} + u_{2,i}$, where $u_{1,i}$ corresponds to the reactions in $C_1$ and $u_{2,i}$ corresponds to the reactions in $C_2$. Since $C_1$ is output-oblivious, we know that every $u_{1,i}$ is independent of every $u_{2,j}$ and thus we can apply Lemma 2 item 2 to see that $c \rightarrow_{u_{1,i}} b_1 \rightarrow_{u_{2,i}} c_1 \rightarrow_{u_{1,j}} b_2 \rightarrow_{u_{2,j}} d$. By repeatedly applying Lemma 2 items 1 and 2, we can then rearrange the sequence of reactions so that each $u_{1,i}$ precedes each $u_{2,j}$ to get $c \rightarrow_{u_{1,i}} b_1 \rightarrow_{u_{2,i}} \cdots \rightarrow_{u_{1,n}} b \rightarrow_{u_{2,i}} \cdots \rightarrow_{u_{2,n}} d$.

Lemma 4. Output-oblivious CRCs are composable.

Proof. Consider the composition $C_{21} = (\Lambda, R, \Sigma, Y)$ of two CRCs $C_1 = (\Lambda_1, R_1, \Sigma_1, Y_1)$ and $C_2 = (\Lambda_2, R_2, \Sigma_2, Y_2)$ that stably compute $f_1$ and $f_2$ respectively, and consider an input $x \in R^d_{\leq 0}$. Consider some state $c$ reached from $x$ in $C_{21}$. Let $b$ be as in Lemma 3, so $x \rightarrow_{u_{1,i}} \cdots \rightarrow_{u_{1,n}} b \rightarrow_{u_{2,i}} \cdots \rightarrow_{u_{2,n}} c$, where $r_i = (u_{1,i}, \ldots, u_{i,n})$ is a sequence of flux vectors...
with \( \bigcup_j \{ u_{1,j} \} \subseteq R_1 \). Since \( C_1 \) stably computes \( f_1 \), we know that there is some \( C_1 \)-output stable state \( o_1 \) reachable from \( b \) using a series of flux vectors \( r = (u_1, u_2 \ldots u_{n}) \) such that \( \bigcup_j \{ u_{1,j} \} \subseteq R_1 \). Since \( r \) only uses reactions from \( C_2 \) and \( C_1 \) is output-oblivious, every flux vector in \( r \) is independent of every flux vector in \( r_2 \), so by Lemma 2 item 3 we know the sequence of flux vectors \( r \) is applicable starting at \( c \). Let \( a \) be such that \( c \rightarrow u_{1} \rightarrow u_{2} \rightarrow u_{n} \). Then applying Lemma 2 items 1 and 2 repeatedly to the flux vectors in \( r \) and \( r_2 \), we see that \( o_1 \rightarrow u_{1} \rightarrow \cdots \rightarrow u_{n} \). Since \( C_2 \) stably computes \( f_2 \), since \( o_1(Y_1) = f(x) \), and since \( a \) is reachable from \( o_1 \) only using reactions in \( C_2 \), there must be some \( o_2 \) that is \( C_2 \)-output stable such that \( a \rightarrow o_2 \) and \( o_2(Y_2) = f_2 \circ f_1(x) \). We know that \( o_2 \) is reachable from \( c \) since \( c \rightarrow a \rightarrow o_2 \). Finally, since \( o_1 \) is \( C_1 \)-output stable, reactions from \( C_1 \) cannot change the concentrations of species in \( o_2 \) before \( f_2 \), so if \( o_2 \rightarrow y \), then restricting to \( C_2 \) we find \( o_2 \rightarrow \Lambda_2 \rightarrow c_y \rightarrow \Lambda_2 \). Since \( o_2 \) is \( C_2 \)-output stable we see that \( y(Y_2) = o_2(Y_2) \), so \( o_2 \) is \( C_2 \)-output stable.

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.

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 \( c \) and any species \( S \), if the amount of \( S \) present in any state reachable from \( c \) is bounded above, there is a state \( d \) reachable from \( c \) such that for any state \( a \) reachable from \( d \), we know that \( a(S) \leq d(S) \).

**Lemma 6.** If a CRC \( C \) 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 \( C_1 = (\Lambda_1, R_1, \Sigma_1, Y_1) \) be a composable CRC stably computing some function \( f \). Let \( C_0 \subseteq C_1 \) be the CRN obtained by removing all of the reactions that consume \( Y_1 \) from \( C_1 \). We would like to show that \( C_0 \) stably computes \( f \). Suppose we compose \( C_1 \) with \( C_2 \) consisting of only the reaction \( Y_1 \rightarrow Y_2 \) with input species \( Y_1 \) and output species \( Y_2 \). Since \( C_1 \) is composable and \( C_2 \) stably computes the identity function, the resulting CRN \( C_{201} \) must stably compute \( f \). For any input vector \( x \) consider a state \( c \) reachable from \( x \).

Assume that \( C_0 \) could reach a state \( d \) from \( c \) where \( d(Y_1) > f(x) \). Then \( C_1 \) would not be composable because \( C_{201} \) can also reach \( d \) and then applying the reaction in \( C_2 \) to convert all \( Y_1 \) into \( Y_2 \) gives us a state \( d' \) with \( d'(Y_2) > f(x) \). Since there is no reaction in \( C_{201} \) that consumes \( Y_2 \) there is no output stable state reachable from \( d' \) that computes \( f(x) \). Therefore there is no state \( d \) such that \( d(Y_1) \geq f(x) \) and \( d \) is reachable from \( c \) via reactions of \( C_0 \).

Since this implies that the amount of \( Y_1 \) in any state reachable from \( c \) is bounded, we can apply Lemma 5 to say that there is a state \( d \) such that \( c \rightarrow d \) and for any state \( a \) reachable from \( d \) we know \( a(Y_1) \leq d(Y_1) \). Since \( C_0 \) has no reactions that consume \( Y_1 \), this is an output stable state of \( C_0 \). Now, consider the state \( b \) in \( C_{201} \) obtained by converting all \( Y_1 \) in \( d \) into \( Y_2 \). Observe that if there were a way to produce \( Y_1 \) from \( b \), then there would be a state in \( C_0 \) reachable from \( d \) that contained more \( Y_1 \). Since there are no reactions in \( C_{201} \) that consume \( Y_2 \) and no reactions that produce \( Y_1 \), we know that \( b \) is an output stable state. Since \( C_{201} \) stably computes \( f(x) \), we know \( b(Y_2) = f(x) \). Thus we can conclude that \( d(Y_1) = f(x) \) and \( C_0 \) stably computes \( f(x) \).

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

### 3 Functions Computable by Composable CRNs

Here we give a complete characterization of the functions computable by composable CRNs as superadditive, positive-continuous, and piecewise rational linear.

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

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 \( \max \).

**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 \( D_U = \{ x \in \mathbb{R}_{\geq 0}^n | (\forall i \in [n]), x(i) > 0 \iff i \in U \} \). 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 $a_1, \ldots, a_n \in \mathbb{Q}$ 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 $f_1, \ldots, f_p : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 $f_1, \ldots, f_p$ the components of $f$.

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

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

The function is superadditive since for all input vectors $a = (a_1, a_2, a_3)$, $b = (b_1, b_2, b_3)$, 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 $f(a) + f(b) = \min(a_1, a_2) + b_1 + b_2 \leq a_1 + a_2 + b_1 + b_2 = f(a + b)$. The function is positive-continuous, since the only points of discontinuity are when $x_3$ changes from zero to positive. The function is piecewise rational linear, since $\min$ is piecewise rational linear.

**Theorem 1.** A function $f : \mathbb{R}^n_+ \rightarrow \mathbb{R}_{\geq 0}$ 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 $f : \mathbb{R}^n_+ \rightarrow \mathbb{R}_{\geq 0}$ 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 $a$, $b$, $c$, if $a \rightarrow b$ then $a + c \rightarrow b + c$.

**Proof.** Adding species cannot prevent reactions from occurring. Thus, we can consider the series of reactions where $c$ doesn’t react to reach the state $b + c$ from the state $a + c$. □

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

**Lemma 10.** If a function $f : \mathbb{R}^n_+ \rightarrow \mathbb{R}_{\geq 0}$ is stably computable by a composable CRC, then $f$ is superadditive.

**Proof.** Assume $C$ stably computes $f$. By definition of $C$ stably computing $f$, $\forall$ initial states $x_1, x_2, \exists o_1, o_2$ such that $x_1 \rightarrow o_1$ with $o_1(Y) = f(x_1)$ and $x_2 \rightarrow o_2$ with $o_2(Y) = f(x_2)$. Consider $C$ on input $x_1 + x_2$. By Lemma 9, $x_1 + x_2 \rightarrow o_1 + o_2$, and again by Lemma 9, $o_1 + o_2 \rightarrow o_1 + o_2$. Looking at the concentration of output species $Y$, we have $(o_1 + o_2)(Y) = f(x_1) + f(x_2)$. Since $C$ stably computes $f$, there exists an output stable state $o'$ reachable from initial state $x_1 + x_2$ and reachable from state $o_1 + o_2$, with $o'(Y) = f(x_1 + x_2)$. Since $C$ 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 $o_1 + o_2$ cannot be reduced from $f(x_1) + f(x_2)$, implying $o'(Y) = f(x_1 + x_2) \geq f(x_1) + f(x_2)$. This holds for all input states $x_1, x_2$, and thus $f$ is superadditive. □

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

### 3.2 Superadditive Positive-Continuous Piecewise Rational Linear Functions are Computable

It was shown in [10] 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 this 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 compositability. 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 $x$ is the set $\ell = \{ y \in \mathbb{R}^n \mid y \cdot x = t \cdot x, t \in \mathbb{R}_{>0} \}$. 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 $x \in \mathbb{R}^n$, we know that $x \in D$ implies the open ray from the origin through $x$ is contained in $D$.

**Lemma 11.** Suppose we are given a continuous piecewise rational linear function $f : \mathbb{R}^n_+ \rightarrow \mathbb{R}_{\geq 0}$. 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 \( f : \mathbb{R}^n_{>0} \to \mathbb{R}_{\geq 0} \) can be written as the minimum of a finite number of rational linear functions \( g_i \).

**Proof.** Since \( f \) is a continuous piecewise rational linear function, by Lemma 11, we can choose domains \( \{ D_j \}_{j=1}^N \) for \( f \) which are cones and contain an open ball of non-zero radius, such that \( f|_{D_j} = g_j|_{D_j} \), where \( g_j \) is a rational linear function. Now pick any \( x \in \mathbb{R}^n_{>0} \) and any \( g_j \). Then because \( D_j \) is a cone containing an open ball of finite radius, it contains open balls with arbitrarily large radii. In particular, it contains a ball with radius greater than \( |x| \), so there exist points \( y, z \in D_j \) such that \( y + x = z \). By the superadditivity of \( f \), the linearity of \( g_j \), and the fact that \( y, z \in D_j \), we see:

\[
g_j(y) + f(x) = f(y) + f(x) \leq f(z) = g_j(x+y) = g_j(y) + g_i(x)
\]

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

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 \( \mathbb{R}^n_{\geq 0} \), so does this lemma; we handle the domain \( \mathbb{R}^n_{>0} \) later on.

**Lemma 13.** We can construct a composable CRC that stably computes any superadditive continuous piecewise rational linear function \( f : \mathbb{R}^n_{>0} \to \mathbb{R}_{\geq 0} \).

**Proof.** By Lemma 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 \( g(x) = a_1x_1 + a_2x_2 + \ldots + a_nx_n \) is stably computed by the reactions

\[
\forall i, \; k_ix_i \to a_i k_i Y
\]

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

\[
Y_1 + \ldots + Y_n \to Y
\]

to get the minimum of the \( Y_i \)'s. Since each \( Y_i \) obtains the count of the corresponding \( g_i \), this CRN will produce the minimum of the \( g_i \)'s quantity of \( Y \)'s. Thus, according to Lemma 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.

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

Surprisingly, Lemma 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 \( f : \mathbb{R}^n_{>0} \to \mathbb{R}_{\geq 0} \). Write \( N = [n] \), and for each \( S \subseteq N \), let \( g_S(x) \) 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 \( x \in D_S \) we know \( g_S(x) \leq g_T(x) \).

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

\[
f(x + \epsilon v) = g_T(x + \epsilon v) < g_T(x) + \delta = g_S(x) = f(x)
\]

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

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

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

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

A naïve approach might be the following: for each subspace \( D_S \), 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 \( g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x) \), 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 \( g_S(x) \geq g_I(x) \) (thus the min of these is \( g_I(x) \)) or (2) the summation term added to \( g_S(x) \) contains at least \( g_I(x) \), and since \( g_S(x) + g_I(x) \geq g_I(x) \), the min of these is \( g_I(x) \). Thus taking the min for all \( S \) of \( g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x) \) is exactly \( g_I(x) \), 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 \( g_S(x) \) be the superadditive continuous piecewise rational linear function that is equal to \( f \) on \( D_S \). Then, \( f(x) = \min_{S \subseteq N} [g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x)] \).

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

\[
h_S(x) = g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x) \]

We want to show that \( f(x) = \min_{S \subseteq N} h_S(x) \). To do this, fix \( x \in \mathbb{R}_{\geq 0}^n \) and define the set \( I = \{ i \in N | x(i) > 0 \} \).

First, let’s show that \( h_I(x) = f(x) \). By the definition of \( I \), for all \( K \not\subseteq I \), we know \( P_K(x) = 0 \). Thus, \( \sum_{K \subseteq I} P_K(x)g_K(x) = 0 \), so \( h_I(x) = g_I(x) = f(x) \). Now we must show that \( h_S(x) \geq f(x) \) for all \( S \subseteq N \). There are two cases to consider:

**Case 1: \( S \not\supseteq I \)**

In this case,

\[
h_S(x) = g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x) \geq g_S(x) + P_I(x)g_I(x) \geq P_I(x)g_I(x) \]

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

**Case 2: \( S \supseteq I \)**

By Lemma 14, \( g_S(x) \geq g_I(x) \). As a result,

\[
h_S(x) = g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x) \geq g_S(x) \geq g_I(x) = f(x) \]

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

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 \( f(x) = \min_{S \subseteq N} [g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x)] \). The first subroutine copies the input species, e.g. \( X_1 \rightarrow X_1^1 + \ldots + X_1^n \), in order for each sub-CRC to not compute for input species. This copying is output-oblivious. Then for any \( Q \subseteq [n] \), \( P_Q(x) \) is computed using one set of copies via the reaction:

\[
\sum_{i \in Q} X_i \rightarrow P_Q \]

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

We can also compute each \( g_Q(x) \) (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:

\[
Y_Q + P_Q \rightarrow Y + P_Q.
\]

The CRC computing min is output-oblivious, as seen in the introduction. The CRC computing the sum of its inputs is output-oblivious (e.g., \( X_1 \rightarrow Y, X_2 \rightarrow Y \) 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 \( \min_{S \subseteq N} [g_S(x) + \sum_{K \subseteq S} P_K(x)g_K(x)] \), which is equal to \( f(x) \) by Lemma 15. 

**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 \( X_1 + \ldots + X_n \rightarrow Y_1 + \ldots + Y_n \) can be decomposed into the reactions \( X_1 + X_2 \rightarrow X_{12}, X_{12} + X_3 \rightarrow X_{123}, \ldots, X_{123...n-1} + X_n \rightarrow Y_{12...n-1} + Y_n, Y_{12...n-1} \rightarrow Y_{12...n-2} + Y_{n-1}, \ldots, Y_{12} \rightarrow Y_1 + Y_2 \). 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 their reactions.

Making copies of input:
\[ X_1 \rightarrow X'_1 + X''_1 \]
\[ X_2 \rightarrow X'_2 + X''_2 \]
\[ X_3 \rightarrow X'_3 \]

Using \( X'_3 \) to make \( P_3 \), which catalyzes reactions for the domain \( X_3 > 0 \):
\[ X'_3 \rightarrow P_3 \]

Computing the sum in \( Y_3 \):
\[ X'_3 \rightarrow Y_3 \]
\[ X''_2 \rightarrow Y_3 \]

Computing the min in \( Y_\emptyset \):
\[ X''_1 + X''_2 \rightarrow Y_\emptyset \]

Making a copy of \( Y_\{3\} \) for use in increasing \( Y_\emptyset \):
\[ Y_\{3\} \rightarrow Y'_\{3\} + Y_\{3\}.\emptyset \]

Increase \( Y'_\emptyset \) so that it will not be the min when \( x_3 \) is present:
\[ Y_\{3\}.\emptyset + P_3 \rightarrow Y'_\emptyset + P_3 \]

Rename \( Y_{\emptyset} \) to \( Y'_\emptyset \) so that it will be summed with the term created by the previous reaction:
\[ Y_{\emptyset} \rightarrow Y'_\emptyset \]
\[ Y'_\emptyset + Y'_\{3\} \rightarrow Y \]

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 that may affect computation.

Definition 27. A CRN with initial context, denoted \( C^{I,1} \), is a CRN \((\Lambda, R, \Sigma, Y, I, 1)\) 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 \( I \in \mathbb{R}_{\geq 0}^I \). \( C^{I,1} \) stably computes \( f : \mathbb{R}_{\geq 0}^I \rightarrow \mathbb{R}_{\geq 0} \) if, for all \( x \in \mathbb{R}_{\geq 0}^I \) and all \( c \) such that \( x + I \rightarrow c \), there exists an output stable state \( o \) such that \( c \rightarrow o \) and \( o(Y) = f(x) \).

We show that initial context for composable CRNs allows only functions which are a min of rational affine functions (in contrast to, without initial context, functions which are a min of rational linear functions).

Note that such functions are not superadditive (e.g., \( f(x_1) = 1 \) is rational affine but not superadditive), so we cannot characterize the class of functions as superadditive positive-continuous piecewise rational affine. Additionally, they are more restricted than positive-continuous piecewise rational affine functions (without superadditivity). Thus, we leave the characterization stated as a min of rational affine functions.

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 CRN \( C \) with initial context \( C^{I,1} \) with \( I \in \mathbb{R}_{\geq 0}^I \) (rational initial concentrations) which stably computes \( f \), there exists a CRN \( C^{I,1} \) with \( I' = \{S'\} \) and \( I'(S') = 1 \) (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 \( C^{I,1} \). Observe that the CRN:
\[ S' \rightarrow S'_1 + S'_2 + \ldots + S'_n \]
can be used to produce \( k \) species with concentrations equal to \( S' \)’s initial concentration. Then the CRN:
\[ bS'_1 \rightarrow aS_i \]
for each \( i \) produces a concentration of \( q_i \) for species \( S_i \).

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 Post\(c\) is the set of states reachable from \( c \), i.e., \( \{d \mid c \rightarrow d\} \).

Lemma 18. Given a CRN \( \{A, R\} \) for any \( r \in \mathbb{R}_{\geq 0} \) and \( c \in \mathbb{R}_{\geq 0}^A \), Post\(rc\) = \( r\text{Post}(c) \).

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

When \( r = 0 \), 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 \( r > 0 \).

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

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

Lemma 19. Let \( C^{I,1} \) with \( I = \{S_1\} \) and \( i(S_1) = q_1 \) stably compute \( f \). Then \( C^{I,\gamma,1} \) for \( \gamma \in \mathbb{R}_{>0} \) stably computes some function \( g \).

Proof. Consider running \( C \) with an initial concentration of \( q'_1 = \gamma q_1 \) for the species \( S_1 \). Observe that the initial state \( x' = \frac{1}{\gamma}x \), where \( x \) is a state that will stably compute \( f \) under the definition of \( C \). By Lemma 18, we know that the set of reachable states from \( x' \) is equal to the set of reachable states from \( x \) scaled by \( \gamma \). Thus consider some state \( c' \) reached from \( x' \). Observe that there exists a state \( c \) reachable from \( x \) such that \( c' = \gamma c \). Consider some output stable state \( o \) reachable from \( c \). Observe that by Lemma 18 the state \( o' = \gamma o \) must be reachable from \( c' \). Similarly by Lemma 18 we know that \( o' \) must be an output stable state. Thus, we know that \( C \) 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 \( C \) 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 \( C \) and claim that this CRN stably computes some function, using lemmas from the case of no initial context to prove properties of that function.

**Definition 28.** A function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is rational affine if there exists \( a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{Q} \) such that \( f(x) = \sum_{i=1}^{n} a_i x(i) + b_i \).

**Lemma 20.** Let \( C_{I, i}^{1, 1} \) be output-oblivious with \( I = \{ S_1 \} \) and \( i(S_1) = 1 \). \( C_{I, i}^{1, 1} \) stably computes \( f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0} \) only if \( f \) is a min of rational affine functions.

**Proof.** Intuitively, we treat the initial context as an input species. For \( C_{I, i}^{1, 1} = (\Lambda, R, \Sigma, Y, I, i) \), let \( C' = (\Lambda, R, \Sigma \cup I, Y) \). By assumption, \( C_{I, i}^{1, 1} \) is output-oblivious, so \( C' \) is output-oblivious. Further, by Lemma 19, \( C' \) stably computes some function for all (positive) initial concentrations of \( S_1 \). So \( C' \) must stably compute a superadditive-positive continuous piecewise rational linear function \( f' \) on the domain with positive (nonzero) initial concentration of \( S_1 \). Therefore by Lemma 12, \( f' \) is a min of rational linear functions:

\[
f'(x_1, \ldots, x_n, s_1) = \min_{j=1, \ldots, m} \left( \sum_{i=1}^{n} a_{ij} x(i) + b_j s_1 \right).
\]

As a result, the function \( f' \) restricted to the domain with the input value represented by \( S_1 \) equal to one is a min of rational affine functions:

\[
f'(x_1, \ldots, x_n, 1) = \min_{j=1, \ldots, m} \left( \sum_{i=1}^{n} a_{ij} x(i) + b_j \right).
\]

So \( C' \) with initial concentration of \( S_1 \) equal to one stably computes a min of rational affine functions. Note that \( C' \) with this input restriction is exactly the CRC \( C_{I, i}^{1, 1} \). Then \( C_{I, i}^{1, 1} \) also computes a min of rational affine functions.

**Theorem 2.** Let \( C_{I, i}^{1, 1} \) be output-oblivious with \( i \in Q_{I, i} \). Then \( C_{I, i}^{1, 1} \) stably computes \( f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0} \) if and only if \( f \) is a min of rational affine functions.

**Proof.** To compute a rational affine function \( \sum_{i=1}^{n} a_i x(i) + b_i \), we produce \( a_i x(i) \) of species \( Y \) as in the case with no initial context (Lemma 13) and then produce \( \sum_{i=1}^{n} b_i \) of \( Y \) via the initial context: \( n \) different initial species \( S_i \) with initial concentrations equal to each \( b_i \), and the reaction \( S_i \rightarrow Y \). This is done composably, so we can compute each rational affine function and then compute the min, resulting in \( f \).

To show that a computed function must be a min of rational affine functions, by Lemma 17, there exists a CRC \( C_{I', i'}^{1, 1} \) with \( I' = \{ S' \} \) and \( i'(S') = 1 \) (one initial species with concentration one) which stably computes the same \( f \). By Lemma 20, \( C_{I', i'}^{1, 1} \) must compute a min of rational affine functions.

\[
f : \mathbb{N}^n \rightarrow \mathbb{N} \text{ is stably computable by a direct CRN if and only if it is semilinear; i.e., its graph } \{(x, y) \in \mathbb{N}^n \times \mathbb{N} \mid f(x) = y\} \text{ is a semilinear subset of } \mathbb{N}^n \times \mathbb{N}. \text{ The proof that composable computable functions must be superadditive (Lemma 10) holds for the discrete model as well. Additionally, there exist functions which are superadditive and semilinear but are not computable in the discrete model by a composable CRN. For example (the proof is omitted):}
\]

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

so the class of composable computable functions is slightly more restricted. A characterization for functions on two inputs \( f : \mathbb{N}^2 \rightarrow \mathbb{N} \), allowing initial context, is given by [6]. Currently, no similar characterization has been proven for functions on more than two inputs.

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, [11] 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


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 $\{x \mid Ax \leq b\}$ for some $m \times n$ matrix $A$ and some vector $b \in \mathbb{R}^m$.

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 $\{x = \lambda_1 x_1 + \ldots + \lambda_n x_n \mid \lambda_1, \ldots, \lambda_n \geq 0\}$ for some finite set of points $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^n$.

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

Lemma 21. [Proven in [12]] 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 $c$ of a CRN $C$, the set of states $\{d \mid c \rightarrow^1 d\}$ that are straight-line reachable from $c$ is a polyhedron.

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

Definition 32. The set of possible species produced from a state $c$ is $\mathcal{P}(c) = \bigcup_{d \in \text{Post}(c)} \{d\}$

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

Proof. Consider some state $c$ and states $S = \{\alpha_1, \ldots, \alpha_k\}$ reachable from $c$. Let $d = \sum_{i=0}^k a_i \alpha_i$ 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 $a_i c$. Since $\sum_{i=0}^k a_i c = c$, we know that $c \rightarrow \sum_{i=0}^k a_i\alpha_i$, which is equal to $d$.

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

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

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

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

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

Proof. By Lemma 24, there is some $c_1$ reachable from $c$ such that $[c_1] = \mathcal{P}(c)$. By Lemma 22, we know that the states that are straight-line reachable from $c_1$ are a polyhedron $P$. The linear map $\mathbb{R}^n \rightarrow R$ sending $x \rightarrow (x(S))$ 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 $d \in P$ with $d(S) = M$. Any state $a$ that is reachable from $d$ is also reachable from $c_1$, so by Lemma 25 it is contained in $P$. As a result, $a(S) \leq M = d(S)$.

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.

Proof. Since $f$ is piecewise rational linear, we can pick a finite set of domains $D = \{D_i\}_{i=1}^n$ for 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 $x \in D_k$. Since the open ray $\ell_x$ from the origin passing through $x$ is contained in $\mathbb{R}_{>0}^n$, it is covered by the domains in $D$. If we write any point $y \in \ell_x$ in the form $t \cdot x$, then, for each $i$, the restriction of $g_i$ to $D_i \cap \ell_x$ is of the form $g_i(t \cdot x) = a_i t$ for some $a_i \in \mathbb{R}$. Since $x \in D_k \cap \ell_x$, we know that $f(1 \cdot x) = a_k \cdot 1 = a_k$.

Now suppose that for some $s \in \mathbb{R}_{\geq 0}$ we know that $f(s \cdot x) \neq a_k s$. First consider the case where $s > 1$. Then define the set $A = \{t \in [1, s] \mid f(t \cdot x) = a_k t\}$ and define the set $B = \{s \in [1, s] \mid f(t \cdot x) \neq a_k t\}$. 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:

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

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 $D$, and 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:

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

Since \( f \) is continuous, we can replace each \( D_i \in D \) by its closure, which is again a cone. Suppose that for any \( D_i \in D \), there is a point \( x \in D_i \) not in the interior of \( D_i \). Then \( x \) is in the closure of the complement of \( D_i \), so there exists a sequence \( \{ x_k \}_{k=1}^{\infty} \) of points in the complement of \( D_i \) such that \( \lim_{k \to \infty} x_k = x \). Since the complement of \( D_i \) is covered by the \( D_j \in D \), where \( j \neq i \), we know that each \( x_k \) lies in one of the \( D_j \). Since there are only finitely many \( D_j \) but infinitely many \( x_k \), we know that infinitely many \( x_k \) must lie in at least one of the \( D_j \). As a result, \( x \) is in the closure of this \( D_j \), and since \( D_j \) is closed, we see that \( x \in D_j \). 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. \( \square \)