Reverse-Robust Computation with Chemical Reaction Networks

Ravi Kini
Computer Science, University of California, Davis, CA, USA

David Doty
Computer Science, University of California, Davis, CA, USA

Abstract

Chemical reaction networks, or CRNs, are known to stably compute semilinear Boolean-valued predicates and functions, provided that all reactions are irreversible. However, this property does not hold for wet-lab implementations, as all chemical reactions are reversible, even at very slow rates. We study the computational power of CRNs under the *reverse-robust* computation model, where reactions are permitted to occur either in forward or in reverse up to a cutoff point, after which they may only occur in forward. Our main results show that all semilinear predicates and all semilinear functions can be computed reverse-robustly, and in fact, that existing constructions continue to hold under the reverse-robust computational model. A key tool used to prove correctness under the reverse-robust computation model is *invariants*: linear (or linear modulo some m) combinations of the counts of the species that are preserved by all reactions.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases chemical reaction networks, reverse-robust computation, semilinear

Digital Object Identifier 10.4230/LIPIcs.DNA.32.6

Funding *David Doty*: funded by NSF awards 2211793 and 2329909.

1 Introduction

Chemical reaction networks (CRNs) are a fundamental tool for understanding and designing molecular systems. By abstracting chemical reactions into a set of finite, rule-based transformations, CRNs allow us to model the behavior of complex chemical systems. For example, a CRN with the single reaction $X_1 + X_2 \to Y$ produces one molecule of Y whenever an X_1 reacts with an X_2. Interpreting the counts of X_1, X_2 as the vector-valued input and the count of Y as the output, the CRN effectively calculates the function $f(x_1, x_2) = \min(x_1, x_2)$. Suppose the single reaction is instead $X_1 + X_2 \to 2X_2$, where X_2 acts as a catalyst, producing another molecule of X_2 when it reacts with a X_1. Interpreting the presence of X_1 as "no" and the presence of X_2 as "yes", the CRN effectively calculates the predicate $\phi(x_1, x_2) = \text{yes} \Leftrightarrow x_2 > 0$. It is known that precisely the *semilinear* predicates $\phi : \mathbb{N}^k \to \{0, 1\}$ [1] and functions $f : \mathbb{N}^k \to \mathbb{N}$ [2] can be computed *stably*, which roughly means that we can reach a correct state regardless of the reactions that occur or the order in which they occur. More precisely, we require that for every state $\mathbf{c}$ reachable from the initial state encoding the input, there is a correct state $\mathbf{y}$ reachable from $\mathbf{c}$, where $\mathbf{y}$ is also *stable*, meaning that every state reachable from $\mathbf{y}$ has the same output as $\mathbf{y}$. Under the reasonable assumption that only a finite number of states are reachable from the initial state, this simple combinatorial definition based on reachability is equivalent to requiring that the CRN *will* reach a correct, stable state with probability 1.

Existing constructions [1, 2, 3] assume that reactions may not occur in reverse. In reality, it is known that any chemical reaction that can happen in the forward direction, turning reactants into products, can also happen in reverse, turning the products back into the

reactants, even if at a much slower rate.[1] To address this incompatibility, we introduce the notion of *reverse-robust* computation with CRNs, in which reactions are allowed to occur in reverse, but we must always be able to reach a correct state using forward reactions. We formalize this by generalizing the previous definition of stable computation to require that from any state $\mathbf{c}$ reachable from the initial state by any combination of forward and reverse reactions, there is a correct, stable state reachable using only forward reactions. In other words, although the reverse reactions can take the CRN to states potentially unreachable through forward reactions alone, such reactions cannot prevent the CRN from reaching the correct answer. We note that this definition allows reverse reactions only transiently; if they eventually stop, then forward reactions will always take the CRN to a correct, stable state.[2]

To understand the distinction between this model and the stable computation model, consider the CRN $2X \to Y$, which stably computes the function $f(x) = \lfloor x/2 \rfloor$. We then add two reactions involving a third species, Z:

$$2X \to Y \tag{1}$$

$$Z \to Y \tag{2}$$

$$Z \to \varnothing \tag{3}$$

Once again interpreting X as the input and Y as the output, under the stable computation model, since there are no copies of Z present initially and no reaction produces Z, there is no way for reactions (2) and (3) to occur. Consequently, the CRN behaves identically to having only reaction (1), and computes $f(x) = \lfloor x/2 \rfloor$. Consider, however, what happens when reactions are allowed to occur in reverse, under the reverse-robust computation model: when reaction (1) produces a Y, reaction (2) in reverse may convert the Y to a Z, and reaction (3) (forward) may then annihilate that Z. It is then possible to reach a state where no molecules remain, and from which no correct state can be reached by only forward reactions.[3] Reactions (2) and (3) behave like a "trap", only disrupting the computation when reverse reactions are allowed.

We show that it is precisely the semilinear predicates and functions that can be computed reverse-robustly; that is, CRNs are equally as powerful under the reverse-robust computation model as under the stable computation model. Note that the capabilities of reverse-robustly computing CRNs are clearly limited by the established results about the capabilities of stably computing CRNs. We may think of stable computation as requiring that the CRN work against an adversary that may choose which reaction occurs at each intermediate state, only allowing reactions to occur in the forward direction. We can then think of reverse-robust computation as requiring that the CRN work against an adversary that may choose which reaction occurs at each intermediate state, allowing reactions to occur in both the forward and reverse directions. The adversary of the reverse-robust computation model is evidently more powerful than that of the stable computation model; the main challenge is then to

---

[1] The fully general model of CRNs associates to a reaction such as $A \underset{r}{\overset{f}{\rightleftharpoons}} B$ a forward *rate constant* f and reverse rate constant r. Without getting into details of the precise energy model, if $f > r$, this means that B is more energetically stable than A, but to have $r = 0$ (i.e., the reaction is completely irreversible) requires that A be have energy *infinitely* larger than B.

[2] Note that if the definition permitted reverse reactions to occur indefinitely, this would trivially prevent any state from being stable, since one could reverse out of any state possibly all the way back to the initial state. See Section 5 for more discussion.

[3] Note that any state can be reached by a combination of forward and reverse reactions, reversing reaction (3) to produce arbitrary amounts of Z, and the other two reactions to produce target amounts of X and Y from that.

show that against this more powerful adversary, we can compute anything that we could compute against the weaker adversary. In fact, we show that the existing constructions for CRNs that worked against the weaker adversary to compute semilinear predicates and functions continue to work against the stronger adversary. To prove the correctness of those constructions under the reverse-robust computation model, we use "invariants", quantities that are preserved by the reactions of a CRN (and thus, by the reverse of those reactions as well).

The paper is organized as follows. Section 2 formally defines what it means for a CRN to compute reverse-robustly (Definitions 1 and 2) and introduces the concept of an invariant. Sections 3 and 4 contain the main positive results of the paper, and prove that the existing constructions used to decide semilinear sets and compute semilinear functions stably also work under the reverse-robust computation model with minimal or no modification. Appendix A contains the proofs of two technical lemmas used in proving the main results.

2 Preliminaries

2.1 Notation

Let $\mathbb{N}$ denote the nonnegative integers. For any finite set Λ, $\mathbb{N}^\Lambda$ denotes the set of functions $f : \Lambda \to \mathbb{N}$. Equivalently, $\mathbf{c} \in \mathbb{N}^\Lambda$ can be interpreted as a vector of $|\Lambda|$ nonnegative integers, where each element specifies the nonnegative integer count of an element of Λ. $\mathbf{c}(i)$ denotes the i-th coordinate of $\mathbf{c}$, and if $\mathbf{c}$ is indexed by elements of Λ, then $\mathbf{c}(Y)$ denotes the count of species $Y \in \Lambda$. We sometimes use multiset notation for such vectors, e.g., $\{A, 3C\}$ for the vector $\mathbf{c} = (1, 0, 3)$, assuming there are three species A, B, C, indexed in that order. For $\Sigma \subseteq \Lambda$, $\mathbf{i}|\Sigma$ denotes the restriction of $\mathbf{i}$ to Σ.

For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, we write $\mathbf{x} \geqq \mathbf{y}$ to indicate that $\mathbf{x}(i) \geq \mathbf{y}(i)$ for all $1 \leq i \leq n$, $\mathbf{x} \geq \mathbf{y}$ to indicate that $\mathbf{x} \geqq \mathbf{y}$ but $\mathbf{x} \neq \mathbf{y}$, and $\mathbf{x} > \mathbf{y}$ to indicate that $\mathbf{x}(i) > \mathbf{y}(i)$ for all $1 \leq i \leq n$. When $\mathbf{y} = \mathbf{0}$, we refer to $\mathbf{x}$ as nonnegative, semipositive, and positive, respectively. $\leqq, \leq, <$ are defined analogously.

2.2 Chemical reaction networks

A *chemical reaction network* (CRN) is a pair $\mathcal{C} = (\Lambda, R)$ where Λ is a finite set of *chemical species* and R is a finite set of (forward) reactions over Λ, where a *(forward) reaction* is a pair $r = (\mathbf{a}, \mathbf{p}) \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda$ where $\mathbf{a}$ indicates the *reactants* and $\mathbf{p}$ the *products*. For a reaction $r = (\mathbf{a}, \mathbf{p})$, we use $r^\leftarrow = (\mathbf{p}, \mathbf{a})$ to denote the corresponding *reverse reaction*.

A *configuration* $\mathbf{c} \in \mathbb{N}^\Lambda$ assigns nonnegative integer counts to every species $\lambda \in \Lambda$. A reaction $r = (\mathbf{a}, \mathbf{p})$ is *applicable* to a configuration $\mathbf{x}$ if $\mathbf{x} \geqq \mathbf{a}$; applying r to $\mathbf{x}$ then results in the configuration $\mathbf{y} = \mathbf{x} - \mathbf{a} + \mathbf{p}$. We say r is *reverse-applicable* to $\mathbf{x}$ if $\mathbf{x} \geqq \mathbf{p}$. Equivalently, we may say that $r^\leftarrow$ is applicable to $\mathbf{x}$. We say r is *bi-applicable* to $\mathbf{x}$ if it is either applicable or reverse applicable to $\mathbf{x}$. If some reaction r is applicable or reverse-applicable to $\mathbf{x}$ and (reverse-)applying it results in the configuration $\mathbf{y}$, we write $\mathbf{x} \leftrightarrow^1 \mathbf{y}$. If it is known that r was applicable, we write $\mathbf{x} \to^1 \mathbf{y}$. When the specific reaction is relevant, we may also write $\mathbf{x} \to^r \mathbf{y}$ if it is the forward reaction, and $\mathbf{x} \leftrightarrow^{r^\leftarrow} \mathbf{y}$ if it is the reverse. A *bi-execution* $\mathcal{E}$ is a finite or infinite sequence of configurations $(\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \ldots)$ such that for all $j \in \{1, \ldots, |\mathcal{E}| - 1\}$, $\mathbf{c}_{j-1} \leftrightarrow^1 \mathbf{c}_j$. A *(forward-)execution* is defined analogously, instead requiring that $\mathbf{c}_{j-1} \to \mathbf{c}_j$. A configuration $\mathbf{y}$ is *bireachable* from another configuration $\mathbf{x}$ (written as $\mathbf{x} \leftrightarrow \mathbf{y}$) if there exists some finite bi-execution that starts at $\mathbf{x}$ and ends at $\mathbf{y}$, and *reachable* (written as $\mathbf{x} \to \mathbf{y}$) if there exists such a forward-execution. Note that both bireachability and reachability are

additive: if $\mathbf{x} \nrightarrow \mathbf{y}$, for all $\mathbf{z} \in \mathbb{N}^\Lambda$, $\mathbf{x} + \mathbf{z} \nrightarrow \mathbf{y} + \mathbf{z}$, and analogously for when $\mathbf{x} \to \mathbf{y}$. It also follows trivially that both bireachability and reachability are transitive and symmetric; however, bireachability is clearly reflexive, as we can simply reverse-apply the reactions moving backwards through the sequence of configurations, while the same is not true for reachability.

2.3 Stable and reverse-robust computation with CRNs

To perform computation using CRNs, we define a way to interpret the final configuration after letting forward reactions until the result can no longer change by applying forward reactions (characterized below as *stable computation*). We define *reverse-robust computation* analogously, letting both forward and reverse reactions occur until the result can no longer change by applying forward reactions. Computation using CRNs primarily involves the evaluation of *predicates* $\phi : \mathbb{N}^k \to \{0, 1\}$ and *functions* $f : \mathbb{N}^k \to \mathbb{N}$.

The definitions below reference *input species* $\Sigma \subseteq \Lambda$ and an *initial context* $\mathbf{s} \in \mathbb{N}^{\Lambda \setminus \Sigma}$. If $\mathbf{s} = \mathbf{0}$, we say that the CRN is *leaderless*. We may assume without loss of generality that if the initial context is nonzero, it contains a single copy of some *leader species* L. In both cases, we say that $\mathbf{i} \in \mathbb{N}^\Lambda$ is a *valid initial configuration* if $\mathbf{i} = \mathbf{s} + \mathbf{x}$ where $\mathbf{x}(\lambda_j) = 0$ for all $\lambda_j \in \Lambda \setminus \Sigma$; i.e., $\mathbf{i}$ contains only input species in addition to the initial context.

A *chemical reaction decider* (CRD) is a tuple $\mathcal{D} = (\Lambda, R, \Sigma, \Upsilon_1, \Upsilon_0, \mathbf{s})$ where (Λ, R) is a CRN, $\Sigma \subseteq \Lambda$ is the set of input species, Υ_1 is the set of *yes voters*, Υ_0 is the set of *no voters*, and $\mathbf{s} \in \mathbb{N}^{\Lambda \setminus \Sigma}$ is the initial context. In the context of CRDs, we use *output species* to refer to $\Gamma = \Upsilon_0 \cup \Upsilon_1$, i.e. the set of voters. We define a global output partial function $\Phi : \mathbb{N}^\Lambda \to \{0, 1\}$ as follows: $\Phi(\mathbf{c}) = 0$ if $\mathbf{c}(\lambda_j) > 0$ for some $\lambda_j \in \Upsilon_0$ and $\mathbf{c}(\lambda'_j) = 0$ for all $\lambda'_j \in \Upsilon_1$, $\Phi(\mathbf{c}) = 1$ if $\mathbf{c}(\lambda_j) > 0$ for some $\lambda_j \in \Upsilon_1$ and $\mathbf{c}(\lambda'_j) = 0$ for all $\lambda'_j \in \Upsilon_0$, and is undefined otherwise. In other words, a unanimous vote is required for an output. We say a configuration $\mathbf{c}$ is *stable* if for all $\mathbf{c}'$ such that $\mathbf{c} \to \mathbf{c}'$, $\Phi(\mathbf{c}) = \Phi(\mathbf{c}')$. A CRD $\mathcal{D}$ is said to *stably decide* the predicate $\phi : \mathbb{N}^\Sigma \to \{0, 1\}$ if for any valid initial configuration $\mathbf{i} \in \mathbb{N}^\Lambda$, letting $\mathbf{i}_0 = \mathbf{i}|\Sigma$, for all $\mathbf{c} \in \mathbb{N}^\Lambda$ such that $\mathbf{i} \to \mathbf{c}$, there exists some $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$ where $\mathbf{o}$ is stable and $\Phi(\mathbf{o}) = \phi(\mathbf{i}_0)$.

▶ **Definition 1.** *A CRD $\mathcal{D}$ is said to* reverse-robustly decide *the predicate $\phi : \mathbb{N}^\Sigma \to \{0, 1\}$ if for any valid initial configuration $\mathbf{i} \in \mathbb{N}^\Lambda$, letting $\mathbf{i}_0 = \mathbf{i}|\Sigma$, for all $\mathbf{c} \in \mathbb{N}^\Lambda$ such that $\mathbf{i} \nrightarrow \mathbf{c}$, there exists some $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$ where $\mathbf{o}$ is stable and $\Phi(\mathbf{o}) = \phi(\mathbf{i}_0)$.*

A *chemical reaction computer* (CRC) is a tuple $\mathcal{C} = (\Lambda, R, \Sigma, Y, \mathbf{s})$ where (Λ, R) is a CRN, $\Sigma \subseteq \Lambda$ is the set of input species, $Y \in \Lambda$ is the *output species*, and $\mathbf{s} \in \mathbb{N}^{\Lambda \setminus \Sigma}$ is the initial context. We say a configuration $\mathbf{c}$ is *stable* if for all $\mathbf{c}'$ such that $\mathbf{c} \to \mathbf{c}'$, $\mathbf{c}(Y) = \mathbf{c}'(Y)$. A CRC $\mathcal{C}$ is said to *stably compute* the function $f : \mathbb{N}^\Sigma \to \mathbb{N}$ if for any valid initial configuration $\mathbf{i} \in \mathbb{N}^\Lambda$, letting $\mathbf{i}_0 = \mathbf{i}|\Sigma$, for all $\mathbf{c} \in \mathbb{N}^\Lambda$ such that $\mathbf{i} \to \mathbf{c}$, there exists some $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$ where $\mathbf{o}$ is stable and $\mathbf{o}(Y) = f(\mathbf{i}_0)$.

▶ **Definition 2.** *A CRC $\mathcal{C}$ is said to* reverse-robustly compute *the function $f : \mathbb{N}^\Sigma \to \mathbb{N}$ if for any valid initial configuration $\mathbf{i} \in \mathbb{N}^\Lambda$, letting $\mathbf{i}_0 = \mathbf{i}|\Sigma$, for all $\mathbf{c} \in \mathbb{N}^\Lambda$ such that $\mathbf{i} \nrightarrow \mathbf{c}$, there exists some $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$ where $\mathbf{o}$ is stable and $\mathbf{o}(Y) = f(\mathbf{i}_0)$.*

Henceforth, we use CRN when a statement is applicable to either CRCs or CRDs. We can then say that a CRN computes stably if for all valid input configurations $\mathbf{i}$, for all configurations $\mathbf{c}$ such that $\mathbf{i} \to \mathbf{c}$, there exists some stable correct configuration $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$. Analogously, a CRN computes reverse-robustly if for all valid input configurations $\mathbf{i}$, for all configurations $\mathbf{c}$ such that $\mathbf{i} \nrightarrow \mathbf{c}$, there exists some stable correct configuration $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$.

It is known that it is exactly the semilinear predicates that can be stably decided by a CRD, and exactly the semilinear functions that can be stably computed by a CRC.

▶ **Definition 3.** *A set $L \subseteq \mathbb{N}^k$ is* linear *if there are vectors $\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_s$ such that $L = \{\mathbf{b} + n_1\mathbf{p}_1 + \ldots + n_k\mathbf{p}_k \mid n_1, \ldots, n_s \in \mathbb{N}\}$. A set is* semilinear *if it is a finite union of linear sets. A predicate $\phi : \mathbb{N}^d \to \{0,1\}$ is* semilinear *if the set $\phi^{-1}(1)$ is semilinear. A function $f : \mathbb{N}^k \to \mathbb{N}$ is* semilinear *if its graph $\{(x,y) \in \mathbb{N}^{k+1} \mid f(x) = y\}$ is semilinear.*

2.4 "Shuffling" executions

The ability to parallelize computation is a key technique used in designing CRNs that compute stably [2]. By this we mean the common trick of splitting input species X_i into two variants via a reaction $X_i \to X_{i,1} + X_{i,2}$, where $X_{i,j}$ is the i'th input species for CRN C_j, and C_1 and C_2 then run in parallel on their respective copies of the input species. We show that this technique continues to work under the reverse-robust computation model.

We first state two technical lemmas about reachability that will help in showing this. Their proofs are contained in Appendix A. If $\mathbf{c}_0 \to^{r_1} \mathbf{c}_1 \to^{r_2} \mathbf{c}_2$, then we cannot in general commute the reactions since r_2 may not be applicable to $\mathbf{c}_0$. Furthermore, even if r_2 *is* applicable to $\mathbf{c}$ and $\mathbf{c}_0 \to^{r_2} \mathbf{c}_1'$, then r_1 may not be applicable to $\mathbf{c}_1'$. However, if none of the products of r_1 are reactants of r_2, then in fact the reactions can be commuted.

▶ **Lemma 4.** *Let $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ such that $\mathbf{c}_0 \to^{r_1} \mathbf{c}_1 \to^{r_2} \mathbf{c}_2$, where $r_1 = (\mathbf{a}_1, \mathbf{p}_1)$ and $r_2 = (\mathbf{a}_2, \mathbf{p}_2)$. If $\mathbf{p}_1 \cap \mathbf{a}_2 = \varnothing$ (none of the products of r_1 are reactants of r_2), then there is some $\mathbf{c}_1'$ such that $\mathbf{c}_0 \to^{r_2} \mathbf{c}_1' \to^{r_1} \mathbf{c}_2$.*

▶ **Lemma 5.** *Let $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ such that $\mathbf{c}_0 \not\to^{r_1{}^{\leftarrow}} \mathbf{c}_1 \to^{r_2} \mathbf{c}_2$, where $r_1 = (\mathbf{a}_1, \mathbf{p}_1)$ and $r_2 = (\mathbf{a}_2, \mathbf{p}_2)$. If $\mathbf{a}_1 \cap \mathbf{a}_2 = \varnothing$ (none of the products of $r_1{}^{\leftarrow}$ are reactants of r_2), then there is some $\mathbf{c}_1'$ such that $\mathbf{c}_0 \to^{r_2} \mathbf{c}_1' \not\to^{r_1{}^{\leftarrow}} \mathbf{c}_2$.*

Further, when a forward reaction occurs consecutively with its reverse reaction, the pair may intuitively be canceled out, producing a bi-execution where those reactions do not occur. We now show that duplicating the input species allows for the parallel reverse-robust computation of multiple predicates or functions.

▶ **Lemma 6.** *Let $\mathcal{C}_1, \mathcal{C}_2$ be some reverse-robustly computing CRNs with input species $\{X_1, \ldots, X_k\}$. For clarity, rename X_i to $X_{i,1}$ in $\mathcal{C}_1$ and $X_{i,2}$ and $\mathcal{C}_2$. Let $\mathcal{C}$ be the CRN formed by combining the reactions of $\mathcal{C}_1$ and $\mathcal{C}_2$ and adding the reaction $r_{i,0}$, for each $i \in \{1, \ldots, k\}$:*

$$X_i \to X_{i,1} + X_{i,2} \tag{4}$$

Then for all valid input configurations $\mathbf{i}$, for all configurations $\mathbf{c}$ such that $\mathbf{i} \not\to \mathbf{c}$, there exists some stable configuration $\mathbf{o}$ such that $\mathbf{c} \to \mathbf{o}$ and the individual outputs of both $\mathcal{C}_1$ and $\mathcal{C}_2$ are correct.

Proof. Let $\mathbf{i}$ be an arbitrary initial configuration, where $\mathbf{i}(X_i) = x_i$. It is easy to see that if we start with only x_i each of $X_{i,1}$ and $X_{i,2}$ in configuration $\mathbf{i}^*$, the CRNs can be run independently, and are able to reverse-robustly compute their respective functions/predicates in parallel. Then for any $\mathbf{c}^*$ such that $\mathbf{i}^* \not\to \mathbf{c}^*$, there exists some stable $\mathbf{o}^*$ such that $\mathbf{c}^* \to \mathbf{o}^*$ and the individual outputs are correct.

Now let $\mathbf{c}$ such that $\mathbf{i} \not\to \mathbf{c}$. For each i, note that by repeatedly applying $r_{i,0}$, we can reach some $\mathbf{c}'$ such that $\mathbf{c} \to \mathbf{c}'$ and $r_{i,0}$ is no longer applicable; we may then assume that in $\mathbf{c}$, $r_{i,0}$ is no longer applicable, and so X_i is not present.

We first argue that $\mathbf{i} \leftrightarrows \mathbf{c}$ by a bi-execution where $r_{i,0}$ is never applied in reverse for all i. Let $\mathcal{E} = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n)$ be a bi-execution where $r_{i,0}$ is reverse-applied (with $\mathbf{c}_0 = \mathbf{i}$ and $\mathbf{c}_n = \mathbf{c}$) for some i. Note that the only reaction where the products of $r_{i,0}^{\leftarrow}$ are reactants is $r_{i,0}$, and that those are the only reactions in which X_i is involved; this affords us a great deal of flexibility in shuffling the order of reactions.

Note that we can always select j and k where $j < k$ such that $\mathbf{c}_j \leftrightarrows^{r_{i,0}^{\leftarrow}} \mathbf{c}_{j+1}$ and $\mathbf{c}_k \rightarrow^{r_{i,0}} \mathbf{c}_{k+1}$ and $\mathbf{c}_{j+1} \leftrightarrows \mathbf{c}_k$ by a bi-execution where $r_{i,0}$ does not occur in forward or in reverse. This follows from the fact that X_i is not present in $\mathbf{c}$; if $r_{i,0}^{\leftarrow}$ occurred, $r_{i,0}$ must have occurred afterwards, as it is the only way X_i can be consumed.

We then have that:

$$\mathbf{c}_j \leftrightarrows^{r_{i,0}^{\leftarrow}} \mathbf{c}_{j+1} \leftrightarrows \mathbf{c}_k \rightarrow^{r_{i,0}} \mathbf{c}_{k+1}$$

where $\mathbf{c}_{j+1} \leftrightarrows \mathbf{c}_k$ by a bi-execution where $r_{i,0}$ does not occur in forward or in reverse. Since none of the products of $r_{i,0}^{\leftarrow}$ (X_i) are reactants of any of the reactions that occur between $\mathbf{c}_{j+1}$ and $\mathbf{c}_k$, we can "shuffle" $r_{i,0}^{\leftarrow}$ forward using Lemma 5 such that:

$$\mathbf{c}_j \leftrightarrows \mathbf{c}'_{k-1} \leftrightarrows^{r_{i,0}^{\leftarrow}} \mathbf{c}_k \rightarrow^{r_{i,0}} \mathbf{c}_{k+1}$$

where $\mathbf{c}_j \leftrightarrows \mathbf{c}'_{k-1}$ by a bi-execution where $r_{i,0}$ does not occur in forward or in reverse. We may then cancel out $r_{i,0}^{\leftarrow}$ and $r_{i,0}$, and so $\mathbf{c}_j \leftrightarrows \mathbf{c}_{k+1}$ by a bi-execution where $r_{i,0}$ does not occur in forward or in reverse.

By repeatedly canceling out any reverse-applications of $r_{i,0}$ in this manner, we see that $\mathbf{i} \leftrightarrows \mathbf{c}$ by an execution where $r_{i,0}^{\leftarrow}$ never occurs for all i.

We now further argue that $\mathbf{i} \leftrightarrows \mathbf{c}$ by an execution where $r_{i,0}$ is never reverse-applied for all i and $\mathbf{i}^*$ is reached at some point. Since we have an execution that starts at $\mathbf{i}$ and ends at $\mathbf{c}$ where $r_{i,0}^{\leftarrow}$ never occurs for all i, no reactions that have the reactants of $r_{i,0}$ (X_i) as products occur for all i. This allows us to shuffle the applications of $r_{i,0}$ to the beginning of the execution using Lemma 4 and Lemma 5:

$$\mathbf{i} \rightarrow \mathbf{i}' \leftrightarrows \mathbf{c}$$

where $\mathbf{i} \rightarrow \mathbf{i}'$ by a forward-execution where only $r_{i,0}$ are applied and $\mathbf{i}' \leftrightarrows \mathbf{c}$ by a bi-execution where $r_{i,0}$ does not occur in forward or in reverse for all i. It is easy to see that $\mathbf{i}' = \mathbf{i}^*$, and so it follows that there exists some stable $\mathbf{o}$ such that $\mathbf{c} \rightarrow \mathbf{o}$ and the individual outputs are correct. It follows that $\mathbf{c} \rightarrow \mathbf{o}$. Consequently, for all valid input configurations $\mathbf{i}$, for all configurations $\mathbf{c}$ such that $\mathbf{i} \leftrightarrows \mathbf{c}$, there exists some stable configuration $\mathbf{o}$ such that $\mathbf{c} \rightarrow \mathbf{o}$ and the individual outputs of both $\mathcal{C}_1$ and $\mathcal{C}_2$ are correct. ◀

2.5 Invariants

For a given CRN, we define an *invariant* to be a function $I : \mathbb{N}^\Lambda \rightarrow \mathbb{N}$ such that for any valid configurations $\mathbf{c}$ and $\mathbf{c}'$, if $\mathbf{c} \leftrightarrows \mathbf{c}'$, then $I(\mathbf{c}) = I(\mathbf{c}')$. Note that if I is an invariant, so is kI for all $k \in \mathbb{N}$ and if I_1, I_2 are invariants, so is $I_1 + I_2$. This paper uses what we refer to as linear and modular invariants:

- An invariant is *linear* if it is of the form $I_{\text{linear}}(\mathbf{c}) = \sum_{j=1}^{k} w_j \mathbf{c}(\lambda_j)$ for $w_1, \ldots, w_k \in \mathbb{Z}$.
- An invariant is *modular* if it is of the form $I_{\text{modular}}(-\mathbf{a} + \mathbf{p}) = \left(\sum_{j=1}^{k} w_j \mathbf{c}(\lambda_j)\right) \bmod m$ for $w_1, \ldots, w_k \in \mathbb{Z}$ and $m \in \mathbb{N}$. We refer to m as the *modulus* of the invariant.

Further, we refer to a linear function as the *linearization* of a modular invariant if taking the linear function modulo m yields the modular invariant. For example, in the following CRN:

$$2A_1 \to A_2 \tag{5}$$
$$A_1 + A_2 \to \varnothing \tag{6}$$
$$2A_2 \to A_1 \tag{7}$$
$$2B \to B + C \tag{8}$$

Examining reactions (5)-(7), note that when A_i reacts with A_j, the product is $A_{i+j \bmod 3}$. Similarly, examining reaction (8), note that B acts as a catalyst to turn copies of B into copies of C. We then see that $I_1(\mathbf{c}) = \left(\sum_{i=1}^{2} i\mathbf{c}(A_i)\right) \bmod 3$ is a modular invariant and that $I_2(\mathbf{c}) = \mathbf{c}(B) + \mathbf{c}(C)$ is a linear invariant. The linearization of I_1 is $I_{1,\text{lin}} = \sum_{i=1}^{2} i(\mathbf{c}(A_i))$.

▶ **Observation 7.** *To check whether a quantity I is a linear or modular invariant, it is sufficient to check that for all reactions $r = (\mathbf{a}, \mathbf{p})$, $I(\mathbf{a}) - I(\mathbf{p}) = 0$.*

To see why this is true, let $\mathbf{c}$ and $\mathbf{c}'$ be arbitrary configurations such that $\mathbf{c} \nrightarrow \mathbf{c}'$ by a single (forward or reverse) reaction. Then let $r = (\mathbf{a}, \mathbf{p})$ such that $\mathbf{c} - \mathbf{a} + \mathbf{p} = \mathbf{c}'$. If I is a linear invariant:

$$\begin{aligned} I(\mathbf{a}) - I(\mathbf{p}) &= I(\mathbf{a}) - I(\mathbf{p}) + I(\mathbf{c}') - I(\mathbf{c}) \\ &= I(\mathbf{a}) - I(\mathbf{p}) + I(\mathbf{c} - \mathbf{a} + \mathbf{p}) - I(\mathbf{c}) \\ &= I(\mathbf{a}) - I(\mathbf{p}) + I(\mathbf{c}) - I(\mathbf{a}) + I(\mathbf{p}) - I(\mathbf{c}) = 0 \end{aligned}$$

If I is a modular invariant with modulus m:

$$\begin{aligned} I(\mathbf{a}) - I(\mathbf{p}) &= I(\mathbf{a}) - I(\mathbf{p}) + I(\mathbf{c}') - I(\mathbf{c}) \\ &= (I_{\text{lin}}(\mathbf{a}) - I_{\text{lin}}(\mathbf{p}) + I_{\text{lin}}(\mathbf{c}') - I_{\text{lin}}(\mathbf{c})) \bmod m \\ &= (I_{\text{lin}}(\mathbf{a}) - I_{\text{lin}}(\mathbf{p}) + I_{\text{lin}}(\mathbf{c} - \mathbf{a} + \mathbf{p}) - I_{\text{lin}}(\mathbf{c})) \bmod m \\ &= (I_{\text{lin}}(\mathbf{a}) - I_{\text{lin}}(\mathbf{p}) + I_{\text{lin}}(\mathbf{c}) - I_{\text{lin}}(\mathbf{a}) + I_{\text{lin}}(\mathbf{p})) - I_{\text{lin}}(\mathbf{c})) \bmod m = 0 \end{aligned}$$

3 Exactly the semilinear sets are reverse-robustly decidable

In this section, we will show that under the reverse-robust computation model, CRDs have the same computational power as under the stable computation model. The following is the main result of the section:

▶ **Theorem 8.** *Exactly the semilinear sets can be reverse-robustly decided by a CRD.*

Since semilinear sets are Boolean combinations of mod and threshold predicates, we prove this theorem by showing that both mod and threshold sets can be reverse-robustly decided, as well as any Boolean combination of such.

3.1 All mod sets can be reverse-robustly decided

▶ **Lemma 9.** *Every mod set $M = \{(x_1, \ldots, x_k)\} \mid \sum_{i=1}^{n} w_i x_i \equiv c \bmod m\}$ (where $w_i, m, c \in \mathbb{N}$ with $0 \leq w_i, c < m$) is reverse-robustly decidable by a CRD.*

We employ a modified version of the construction from [1]. Let the set of input species be $\Sigma = \{X_1, \ldots, X_k\}$. We then add the following reaction, for each $i \in \{1, \ldots, k\}$:

$$X_i \to Y_{w_i} \tag{9}$$

Then, for each $p, q \in \{0, \ldots, m-1\}$ add the reactions:

$$Y_p + Y_q \to Y_{p+q \bmod m} \tag{10}$$

Let the set of yes voters be $\Gamma_1 = \{Y_c\}$ and the set of no voters be $\Gamma_0 = \{Y_p \mid p \neq c\}$. Intuitively, the CRN does leader election, with the leader accumulating the weighted sum mod m.

Proof. By inspecting the reactions, we see that the CRN has the modular invariant:

$$I_M(\mathbf{c}) = \left(\sum_{i=0}^{k} w_i \mathbf{c}(X_i) + \sum_{p=0}^{m-1} p\mathbf{c}(Y_p) \right) \bmod m$$

For an initial configuration $\mathbf{i}$ where $\mathbf{i}(X_i) = x_i$, the invariant has value:

$$I_M(\mathbf{i}) = \sum_{i=1}^{k} w_i \mathbf{c}(X_i) \bmod m = \sum_{i=1}^{d} w_i x_i \bmod m$$

We now prove that this construction reverse-robustly decides M. Let $\mathbf{i}$ be an arbitrary initial configuration, where $\mathbf{i}(X_i) = x_i$, and let $\mathbf{c}$ such that $\mathbf{i} \not\to \mathbf{c}$. If any X_i are present in $\mathbf{c}$, we first apply reaction (9) repeatedly until X_i is no longer present. We then apply reaction (10) repeatedly until only one Y_{p^*} is present and no more reactions can occur; the resulting configuration $\mathbf{o}$ must then be stable. To see that $\mathbf{o}$ also has the correct output, note that the invariant has value:

$$I_M(\mathbf{o}) = p^* \mathbf{o}(Y_{p^*}) \bmod m = p^*$$

The remaining Y_{p^*} is then a yes voter if and only if $\sum_{i=1}^{d} w_i x_i \equiv c \bmod m$. Since $\mathbf{o}$ is stable with the correct output and $\mathbf{c} \to \mathbf{o}$, the CRD reverse-robustly decides M. ◀

3.2 All threshold sets can be reverse-robustly decided

▶ **Lemma 10.** *Every threshold set $T = \{(x_1, \ldots, x_k)\} \mid \sum_{i=1}^{k} w_i x_i \geq t\}$ (where $w_i, t \in \mathbb{Z}$) is reverse-robustly decidable by a CRD.*

We employ a modified version of the construction from [1]. Let the set of input species be $\Sigma = \{X_1, \ldots, X_k\}$ and define c such that $c = \max\{|w_1|, \ldots, |w_k|, |t|\} + 1$. We then add the following reaction, for each $i \in \{1, \ldots, k\}$:

$$X_i \to Y^L_{w_i} \tag{11}$$

Then, for each $p, q \in \{-c, \ldots, c\}$ and $x \in \{L, F\}$ add the reactions:

$$Y^L_p + Y^x_q \to \begin{cases} Y^L_{-c} + Y^F_{(p+q)+c} & p + q < -c \\ Y^L_{p+q} & -c \leq p + q \leq c \\ Y^L_c + Y^F_{(p+q)-c} & c < p + q \end{cases} \tag{12}$$

Let the set of yes voters be $\Gamma_1 = \{Y^L_p \mid p \geq t\}$ and the set of no voters be $\Gamma_0 = \{Y^L_p \mid p < t\}$. Intuitively, the CRN does leader election, with the leader accumulating the weighted sum, clipped to the range $[-c, c]$ so that the set of species is finite. Any excess is accumulated by one or more of the followers.

Proof. By inspecting the reactions, we see that the CRN has the linear invariant:

$$I_T(\mathbf{c}) = \sum_{i=0}^{k} w_i \mathbf{c}(X_i) + \sum_{p=-c}^{c} p\mathbf{c}(Y_p^L) + \sum_{q=-c}^{c} q\mathbf{c}(Y_q^F)$$

For an initial configuration $\mathbf{i}$ where $\mathbf{i}(X_i) = x_i$, the invariant has value:

$$I_T(\mathbf{i}) = w_i \mathbf{c}(X_i) = \sum_{i=1}^{k} w_i x_i$$

We now prove that this construction reverse-robustly decides T. Let $\mathbf{i}$ be an arbitrary initial configuration, where $\mathbf{i}(X_i) = x_i$, and let $\mathbf{c}$ such that $\mathbf{i} \not\to \mathbf{c}$. If any X_i are present in $\mathbf{c}$, we first apply reaction (11) repeatedly until X_i is no longer present. We then apply reaction (12) repeatedly until only one $Y_{p^*}^L$ is present (possibly in addition to some Y_q^F) and no more reactions can occur; the resulting configuration $\mathbf{o}$ must then be stable. To see that $\mathbf{o}$ also has the correct output, we consider two cases.

In the case that $Y_{p^*}^L$ is the only remaining molecule, the invariant has value:

$$I_T(\mathbf{o}) = p^* \mathbf{o}(Y_{p^*}^L) = p^*$$

The remaining $Y_{p^*}^L$ is then a yes voter if and only if $\sum_{i=1}^{k} w_i x_i \geq t$.

Alternatively, if some Y_q^F are present, we first note that either $p^* = -c$ or $p^* = c$ and further, p^* and all q must have the same sign; otherwise reaction (12) would be applicable. If $p^* = c > 0$, the invariant has value:

$$I_T(\mathbf{o}) = c\mathbf{o}(Y_c^L) + \sum_{q=1}^{c} q\mathbf{o}(Y_{q^*}^F) = c + \sum_{q=1}^{c} q\mathbf{o}(Y_{q^*}^F)$$

The remaining Y_c^L is a yes voter, which is correct since $\sum_{i=1}^{k} w_i x_i \geq t = c + \sum_{q=1}^{c} q\mathbf{o}(Y_{q^*}^F) > c$. By a similar argument, if $p^* = -c$, the remaining Y_{-c}^L is correctly a no voter.

Since $\mathbf{o}$ is stable with the correct output and $\mathbf{c} \to \mathbf{o}$ in all cases, the CRD reverse-robustly decides T. ◀

3.3 Reverse-robustly decidable sets are closed under Boolean operations

▶ **Lemma 11.** *If sets $X_1, X_2 \subseteq \mathbb{N}^d$ are reverse-robustly decided by some CRDs $\mathcal{D}_1$ and $\mathcal{D}_2$, then so are $X_1 \cup X_2$, $X_1 \cap X_2$, and $\overline{X_1}$.*

Proof. To reverse-robustly decide $\overline{X_1}$, swap the yes and no voters. As the reactions themselves remain unchanged, it is trivial to show that this construction is reverse-robust.

To show that $X_1 \cup X_2$ and $X_1 \cap X_2$ are reverse-robustly decidable, consider a construction where both sets are decided in parallel and their votes are recorded in a new voter species. We first add reactions that duplicate the input and produce one of the four new voter species $V_{NN}, V_{NY}, V_{YN}, V_{YY}$:

$$X_i \to X_{i,1} + X_{i,2} + V_{NN} \tag{13}$$

Note that V_{NN} is chosen arbitrarily. $X_{i,1}$ and $X_{i,2}$ then replace X_i in any reactions involving X_i in $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. We then add reactions that store the separate votes in the new voter species. The votes of the $\mathcal{D}_1$ and $\mathcal{D}_2$ are stored in the first and second subscript of the new voter species, respectively. For each $b \in \{Y, N\}$ and for every S_b and T_b that vote in $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, and for each $x \in \{Y, N\}$, we add the reactions:

$$S_b + V_{\overline{b}x} \to S_b + V_{bx} \tag{14}$$
$$T_b + V_{x\overline{b}} \to S_b + V_{xb} \tag{15}$$

To reverse-robustly decide $X_1 \cup X_2$, we let the yes voters be $\Gamma = \{V_{NY}, V_{YN}, V_{YY}\}$ and to reverse-robustly decide $X_1 \cap X_2$, we let the yes voters be $\Gamma = \{V_{YY}\}$.

We now prove that this construction reverse-robustly decides $X_1 \cup X_2$ or $X_1 \cap X_2$. Let $\mathbf{i}$ be an arbitrary initial configuration, where $\mathbf{i}(X_i) = x_i$, and let $\mathbf{c}$ such that $\mathbf{i} \not\mapsto \mathbf{c}$. In the configuration $\mathbf{c}$, the voter species may or may not be have the correct vote–depending on the reactions applied they may end up as any of V_{NN}, V_{NY}, V_{YN}, or V_{YY}.If any X_i are present in $\mathbf{c}$, we first apply reactions (13) repeatedly until X_i is no longer present. We then run $\mathcal{D}_1$ and $\mathcal{D}_2$ until no more of their reactions can occur, then finally do the same for reactions (14) and (15); this resulting configuration $\mathbf{o}$ must then be stable. Since $\mathcal{D}_1$ and $\mathcal{D}_2$ reverse-robustly decide X_1 and X_2 and their voters do not have their counts changed by reactions (14) and (15), by Lemma 6, their individual votes are stable and correct in $\mathbf{o}$. If any of the new voter molecules with the wrong overall vote are present, either (14) and (15) would be applicable; all of the new voter molecules must then have the correct overall vote. Since $\mathbf{o}$ is stable with the correct output, the CRD reverse-robustly decides $X_1 \cup X_2$ or $X_1 \cap X_2$, depending on how the yes and no voters are selected. ◀

4 Exactly the semilinear functions are reverse-robustly computable

In this section, we will show that under the reverse-robust computation model, CRCs have the same computational power as under the stable computation model. The following is the main result of the section:

▶ **Theorem 12.** *Exactly the semilinear functions can be reverse-robustly computed by a CRC.*

4.1 Diff-representations of all partial affine functions can be reverse-robustly computed

We say that a partial function $f : \mathbb{N}^k \to \mathbb{N}$ is *affine* if there exist vectors $a_1, \ldots, a_k \in \mathbb{Q}$, $c_1, \ldots c_k \in \mathbb{N}$, and $b \in \mathbb{N}$ such that when $\mathbf{x}(i) - c_i \geq 0$:

$$f(\mathbf{x}) = b + \sum_{i=1}^{k} a_i(\mathbf{x}(i) - c_i)$$

Note that f is not defined for all $\mathbf{x} \in \mathbb{N}^k$; not only must $\mathbf{x}(i) \geq c_i$ for all i, but $b + \sum_{i=1}^{k} a_i(\mathbf{x}(i) - c_i)$ must be a nonnegative integer. For a partial function f we write $\mathrm{dom} f$ for the domain of f, the set of inputs for which f is defined. For convenience, we can work with integer valued molecule counts by factoring out the reduced denominator d the dot product, where d may be taken to be the least common multiple of the denominators of the rational coefficients in the original definition such that $n_i = d \cdot a_i$:

$$f(\mathbf{x}) = b + \sum_{i=1}^{k} a_i(\mathbf{x}(i) - c_i) \iff f(\mathbf{x}) = b + \frac{1}{d}\sum_{i=1}^{k} n_i(\mathbf{x}(i) - c_i)$$

We say that a partial function $\hat{f} : \mathbb{N}^k \to \mathbb{N}^2$ is a *diff-representation* of f if $\mathrm{dom} f = \mathrm{dom} \hat{f}$ and for all $x \in \mathrm{dom} f$, if $(y_P, y_C) = \hat{f}(\mathbf{x})$, then $f(\mathbf{x}) = y_P - y_C$, and $y_P = O(f(x))$. In other words, $\hat{f}$ represents f as the difference of its two outputs y_P and y_C, the larger output y_P possibly being larger than the output of the original function, but at most a multiplicative constant larger.

▶ **Lemma 13.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be an affine partial function. Then there is a diff-representation $\hat{f} : \mathbb{N}^k \to \mathbb{N}^2$ and a CRC that reverse-robustly computes $\hat{f}$.*

We employ the construction from [3]. Let the set of input species be $\Sigma = \{X_1, \ldots, X_k\}$ and the set of output species be $\Gamma = \{Y^P, Y^C\}$. Then, for each $i \in \{1, \ldots, k\}$, we add the following reaction:

$$X_i \to C_{i,1} + B + bY^P \tag{16}$$

Then, for each $i \in \{1, \ldots, k\}$ and for each $p, q \in \{1, \ldots, c_i\}$, add the reactions:

$$C_{i,p} + C_{i,q} \to \begin{cases} C_{i,p+q} & p+q \le c_i \\ C_{i,c_i} + (p+q-c_i)X_i' & p+q > c_i \end{cases} \tag{17}$$

Then, for each $i \in \{1, \ldots, k\}$, add the reaction:

$$X_i' \to \begin{cases} n_i D_1^P & n_i \ge 0 \\ (-n_i) D_1^C & n_i < 0 \end{cases} \tag{18}$$

Then, for each $p, q \in \{1, \ldots, d-1\}$, add the reactions:

$$D_p^P + D_q^P \to \begin{cases} D_{p+q}^P & p+q \le d-1 \\ D_{p+q-d}^P + Y^P & p+q > d-1 \end{cases} \tag{19}$$

$$D_p^C + D_q^C \to \begin{cases} D_{p+q}^C & p+q \le d-1 \\ D_{p+q-d}^C + Y^C & p+q > d-1 \end{cases} \tag{20}$$

Then, add the reaction:

$$B + B \to B + bY^C \tag{21}$$

Proof. By inspecting the reactions, we see that the CRN has the linear invariant:

$$I_{\hat{f}}(\mathbf{c}) = \sum_{i=1}^{k} n_i \left(\mathbf{c}(X_i) + \sum_{p=1}^{c_i} p\mathbf{c}(C_{i,p}) + \sum_{i=1}^{k} \mathbf{c}(X_i') \right) + \sum_{p=1}^{d-1} p\mathbf{c}(D_p^P) - \sum_{q=1}^{d-1} q\mathbf{c}(D_q^C) + d(\mathbf{c}(Y^P) - \mathbf{c}(Y^C)) - bd\mathbf{c}(B)$$

in addition to the modular invariants:

$$I_P(\mathbf{c}) = \left(\sum_{n_i \ge 0} n_i \left(\mathbf{c}(X_i) + \sum_{p=1}^{c_i} p\mathbf{c}(C_{i,p}) + \sum_{i=1}^{k} \mathbf{c}(X_i') \right) + \sum_{p=1}^{d-1} p\mathbf{c}(D_p^P) \right) \bmod d$$

$$I_C(\mathbf{c}) = \left(\sum_{n_i < 0} n_i \left(\mathbf{c}(X_i) + \sum_{p=1}^{c_i} p\mathbf{c}(C_{i,p}) + \sum_{i=1}^{k} \mathbf{c}(X_i') \right) - \sum_{q=1}^{d-1} q\mathbf{c}(D_q^C) \right) \bmod d$$

For an initial configuration $\mathbf{i}$ where $\mathbf{i}(X_i) = x_i$, the invariants have values:

$$I_{\hat{f}}(\mathbf{i}) = \sum_{i=1}^{k} n_i \mathbf{c}(X_i) = \sum_{i=1}^{k} n_i x_i$$

$$I_P(\mathbf{i}) = \left(\sum_{n_i \ge 0} n_i \mathbf{i}(X_i) \right) \bmod d = \left(\sum_{n_i \ge 0} n_i x_i \right) \bmod d$$

$$I_C(\mathbf{i}) = \left(\sum_{n_i < 0} n_i \mathbf{i}(X_i) \right) \bmod d = \left(\sum_{n_i < 0} n_i x_i \right) \bmod d$$

We now prove that this construction reverse-robustly calculates $\hat{f}$. Let $\mathbf{i}$ be an arbitrary initial configuration, where $\mathbf{i}(X_i) = x_i$ and $x_i - c_i \geq 0$, and let $\mathbf{c}$ such that $\mathbf{i} \nrightarrow \mathbf{c}$. If any X_i are present in $\mathbf{c}$, we first apply reactions (16) repeatedly. We then apply reaction (17) repeatedly until only one C_{i,s_i^*} is present for each i and no more of that reaction can occur. We then do the same for reaction (18) until no more X_i' are present, and then reactions (19) and (20) until only one $D_{p^*}^P$ and $D_{q^*}^C$ are present. Finally, we do the same for reaction (21) until only one B is present; this resulting configuration $\mathbf{o}$ must then be stable. To see that $\mathbf{o}$ also has the correct output, we first consider the following quantity:

$$I_{c,i}(\mathbf{c}) = \mathbf{c}(X_i) + \sum_{p=1}^{c_i} p\mathbf{c}(C_{i,p})$$

In the initial configuration, the quantity has value $I_{c,i}(\mathbf{i}) = \mathbf{i}(X_i) = x_i \geq c_i$. By inspecting reaction (16), as well as reactions (18) through (21), we see that those reactions preserve the quantity. For reaction (17), however, while the quantity is preserved whenever $p + q \leq c_i$, this is not true whenever $p + q > c_i$, as $\sum_{p=1}^{c_i} p\mathbf{c}(C_{i,p})$ changes. Specifically, it decreases when the reaction is applied in forward and increases when applied in reverse. Even so, since C_{i,c_i} is one of the products of the forward reaction, we see that the quantity remains at least c_i for any $\mathbf{c}$ such that $\mathbf{i} \nrightarrow \mathbf{c}$. Since only one C_{i,s_i^*} is present for each i and no X_i are present, the quantity has the value $I_{c,i}(\mathbf{o}) = s_i^*\mathbf{o}(C_{i,s_i^*}) = s_i^*$, and it must be then be the case that $s_i^* = c_i$.

We next consider the modular variants I_P and I_C. In $\mathbf{o}$, I_P has the value:

$$I_P(\mathbf{o}) = \left(\sum_{n_i \geq 0}^{k} n_i c_i \mathbf{c}(C_{i,c_i}) + p^*\mathbf{c}(D_{p^*}^P)\right) \bmod d = \left(\sum_{n_i \geq 0}^{k} n_i c_i + p^*\right) \bmod d$$

For this to be true, $p^* = \sum_{n_i \geq 0} n_i(x_i - c_i) \bmod d$. By a similar argument, $q^* = \sum_{n_i < 0}(-n_i) \cdot (x_i - c_i) \bmod d$. Note that for $f(\mathbf{x})$ to be an integer, $\sum_{i=1}^{k} n_i(\mathbf{x}(i) - c_i) \bmod d = (p^* - q^*) \bmod d = 0$. Since $0 \leq p^*, q^* < d$, it must be the case that $p^* - q^* = 0$. We then see that $I_{\hat{f}}$ has value:

$$\begin{aligned} I_{\hat{f}}(\mathbf{o}) &= \sum_{i=1}^{k} n_i c_i \mathbf{o}(C_{i,c_i}) + p^*\mathbf{o}(D_{p^*}^P) - q^*\mathbf{o}(D_{q^*}^C) + d(\mathbf{o}(Y^P) - \mathbf{o}(Y^C)) - bd\mathbf{o}(B) \\ &= \sum_{i=1}^{k} n_i c_i + p^* - q^* + d(\mathbf{o}(Y^P) - \mathbf{o}(Y^C)) - bd \\ &= \sum_{i=1}^{k} n_i c_i + d(\mathbf{o}(Y^P) - \mathbf{o}(Y^C)) - bd \end{aligned}$$

For this to be true, it must be the case that:

$$\mathbf{o}(Y^P) - \mathbf{o}(Y^C) = b + \frac{1}{d}\sum_{i=1}^{k} n_i(x_i - c_i) = f(\mathbf{x})$$

Since $\mathbf{o}(Y^P) - \mathbf{o}(Y^C) = f(\mathbf{x})$, the CRC reverse-robustly computes $\hat{f}$. ◀

4.2 All semilinear functions can be reverse-robustly computed

We require the following result from [4], which guarantees that any semilinear function can be decomposed into affine partial functions with disjoint domains.

▶ **Lemma 14.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a semilinear function. Then there is a finite set $\{f_1 : \mathbb{N}^k \to \mathbb{N}, \ldots, f_m : \mathbb{N}^d \to \mathbb{N}\}$ of affine partial functions, where each $\mathrm{dom} f_i$ is a linear set, and $\mathrm{dom} f_i \cap \mathrm{dom} f_j = \varnothing$ for all $i \neq j$, such that, for each $x \in \mathbb{N}^k$, if $\mathbf{x} \in \mathrm{dom} f_i$, then $f(\mathbf{x}) = f_i(\mathbf{x})$, and $\bigcup_{i=1}^{m} \mathrm{dom} f_i = \mathbb{N}^k$.*

The next theorem shows that semilinear functions can be reverse-robustly computed.

▶ **Theorem 15.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a semilinear function. Then there is a CRC that reverse-robustly computes f.*

We employ the construction from [3]. Let the set of input species be $\Sigma = \{X_1, \ldots, X_k\}$ and the output species be Y. By Lemma 14, f can be decomposed into partial affine functions $f_1, \ldots, f_m$ with disjoint linear domains, and by Theorem 8 and Lemma 13, each of the partial affine functions and their domains can be computed reverse-robustly. By duplicating X_i, we can reverse-robustly decide the predicate $\mathbf{x} \in \mathrm{dom} f_j$ and compute $\hat{f}_j(\mathbf{x})$ in parallel for each f_j. Let L_j^Y and L_j^N be the yes and no voters, respectively, for the CRD $\mathcal{D}_j$ reverse-robustly deciding if $\mathbf{x} \in \mathrm{dom} f_j$ and let $\hat{Y}_j^P$ and $\hat{Y}_j^C$ be the output species for the CRC $\mathcal{C}_j$ reverse-robustly computing $\hat{f}_j$, using the constructions from the previous sections. We interpret each $\hat{Y}_j^P$ and $\hat{Y}_j^C$ as "inactive" versions of the "active" output species Y_j^P and Y_j^C. Now, we convert the function result of the applicable partial affine function to the global output by adding the following reactions for each $j \in \{1, \ldots m\}$ and for each L_j^Y that votes yes and L_j^N that votes no in $\mathcal{D}_j$:

$$L_j^Y + \hat{Y}_j^P \to L_j^Y + Y_j^P + Y \tag{22}$$
$$L_j^N + Y_j^P + Y \to L_j^N + \hat{Y}_j^P \tag{23}$$

Reaction (22) produces an output copy of species Y and (23) undoes the first reaction. Both are catalyzed by the vote of the i-th predicate result. We also add the reactions:

$$L_j^Y + \hat{Y}_j^C \to L_j^Y + Y_j^C \tag{24}$$
$$L_j^N + Y_j^C \to L_j^Y + \hat{Y}_j^C \tag{25}$$

and

$$Y_j^P + Y_j^C + Y \to \varnothing \tag{26}$$

Proof. We now prove that this construction reverse-robustly decides f. Let $\mathbf{i}$ be an arbitrary initial configuration, where $\mathbf{i}(X_i) = x_i$, and let $\mathbf{c}$ such that $\mathbf{i} \not\hookrightarrow \mathbf{c}$. We first run each $\mathcal{D}_j$ and $\mathcal{C}_j$ in parallel until no more of the inactive output species can be produced. We then run reactions (22) through (26) until no more reactions can occur; this resulting configuration $\mathbf{o}$ must be stable.

To see that $\mathbf{o}$ also has the correct output, since each $\mathcal{D}_j$ reverse-robustly decides $\mathbf{x} \in \mathrm{dom} f_j$ and the counts of their voters are not changed by reactions (22) through (26), by Lemma 6, their individual votes are stable and correct in $\mathbf{o}$. Although each $\mathcal{C}_j$ reverse-robustly computes $\hat{f}_j$, since their output is consumed, we cannot apply Lemma 6. However, let $I_{\hat{f}_j}$ be the linear invariant derived for each $\mathcal{C}_j$. We see that for each $j \in \{1, \ldots, m\}$, appropriately renaming species, constant, and indices, the CRN has the linear invariant:

$$
\begin{aligned}
I_j(\mathbf{c}) &= \sum_{i=1}^{k} n_{i,j}\mathbf{c}(X_i) + I_{\hat{f}_j}(\mathbf{c}) + d_j(\mathbf{c}(Y_j^P) - \mathbf{c}(Y_j^C)) \\
&= \sum_{i=1}^{k} n_{i,j}\left(\mathbf{c}(X_i) + \mathbf{c}(X_{i,j}) + \sum_{p=1}^{c_{i,j}} p\mathbf{c}(C_{i,p,j}) + \sum_{i=1}^{k} \mathbf{c}(X'_{i,j})\right) \\
&+ \sum_{p=1}^{d_j-1} p\mathbf{c}(D_{p,j}^P) - \sum_{q=1}^{d_j-1} q\mathbf{c}(D_{q,j}^C) + d_j(\mathbf{c}(\hat{Y}_j^P) + \mathbf{c}(Y_j^P) - \mathbf{c}(\hat{Y}_j^C) - \mathbf{c}(Y_j^C)) - b_j d_j \mathbf{c}(B_j)
\end{aligned}
$$

By a similar argument to the partial affine case, it follows that:

$$\mathbf{o}(\hat{Y}_j^P) + \mathbf{o}(Y_j^P) - \mathbf{o}(\hat{Y}_j^C) - \mathbf{o}(Y_j^C) = b_j + \frac{1}{d_j}\sum_{i=1}^{k} n_{i,j}(x_i - c_{i,j}) = f_j(\mathbf{x})$$

We further see by inspecting the reactions that the CRN has the additional linear invariant:

$$I_0(\mathbf{c}) = \sum_{j=1}^{m} \mathbf{c}(Y_j^P) - \mathbf{c}(Y)$$

For an initial configuration $\mathbf{i}$ where $\mathbf{i}(X_i) = x_i$, the invariant has value:

$$I_0(\mathbf{i}) = 0$$

We argue that in $\mathbf{o}$, there is exactly one j^* such that $L_{j^*}^Y$ is present, and that $\mathbf{x} \in \mathrm{dom} f_{j^*}$; this follows from each $\mathcal{D}_j$ reverse-robustly deciding their predicate, and their voters deactivating the incorrect output species. It must then be the case that there is exactly one j^* (the same j^*) such that $Y_{j^*}^P$ and $Y_{j^*}^C$ can be present and further, that $\hat{Y}_{j^*}^P$ and $\hat{Y}_{j^*}^C$ are absent, as otherwise they could be activated by the voters of $\mathcal{D}_{j^*}$. It then follows that:

$$\mathbf{o}(Y_{j^*}^P) - \mathbf{o}(Y_{j^*}^C) = f_{j^*}(\mathbf{x})$$

Additionally, the invariant I_0 has the value:

$$I_0(\mathbf{o}) = \mathbf{o}(Y_{j^*}^P) - \mathbf{o}(Y)$$

For this to be true, $\mathbf{o}(Y_{j^*}^P) = \mathbf{o}(Y)$. Note that it is impossible for $Y_{j^*}^P$, $Y_{j^*}^C$, and Y to be simultaneously present, as otherwise reaction (26) could occur. If $\mathbf{o}(Y_{j^*}^P) = \mathbf{o}(Y) > 0$, $\mathbf{o}(Y_{j^*}^C) = 0$. On the other hand, if $\mathbf{o}(Y_{j^*}^P) = \mathbf{o}(Y) = 0$, since $f_{j^*}(\mathbf{x}) \geq 0$, $\mathbf{o}(Y_{j^*}^C) = 0$. In either case, $\mathbf{o}(Y_{j^*}^C) = 0$. Then:

$$\mathbf{o}(Y) = \mathbf{o}(Y_{j^*}^P) = f_{j^*}(\mathbf{x})$$

Since $\mathbf{x} \in \mathrm{dom} f_{j^*}$, $\mathbf{o}(Y) = f(\mathbf{x})$, and the CRC reverse-robustly computes f. ◀

5 Conclusion

We explored the capabilities of chemical reaction networks (CRNs) under the reverse-robust computation model, where reactions are allowed to occur in reverse while maintaining the existence of a reachable correct stable configuration. This model aligns with thermodynamic theory assuming all species have a finite free energy, implying all chemical reactions are reversible, albeit at some possibly very slow rate. We showed that CRNs have the same

computational power under the reverse-robust computation model as they do under the stable computation model. Both semilinear predicates can be decided and semilinear functions can be computed reverse-robustly without an initial leader.

Some key questions remain open: first, are there a set of conditions that are necessary and/or sufficient to show that a CRN is reverse-robust? This paper was able to use existing constructions designed for the stable computation model to prove results for the reverse-robust computation model. In fact, we were unable to design a CRN that works under the former model but not the latter without the presence of a "trap" reaction that cannot occur when only forward reactions are allowed. We thus conjecture that the absence of such reactions is sufficient to prove reverse-robustness (and so, that a stably computing CRN can be made reverse-robust by removing such reactions without affecting its correctness). Second, our model only allows reverse-reactions to occur transiently, as we require that only forward reactions occur in the final path to reach a stable state. Is it possible to formally define some reasonable model of computation in which reverse-reactions are always allowed to occur? Note that such a model cannot require stability, as it is always possible to reverse the path of reactions that led to a state. Instead, we may require "favorability"; although temporary reversal to incorrect states can occur, the correct answer is present "most of the time".

References

1 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/S00446-005-0138-3.

2 Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Nat. Comput.*, 13(4):517–534, 2014. doi:10.1007/S11047-013-9393-6.

3 David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. *Nat. Comput.*, 14(2):213–223, 2015. doi:10.1007/S11047-014-9435-8.

4 David Doty and Ben Heckmann. The computational power of discrete chemical reaction networks with bounded executions. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 – November 1, 2024*, LIPIcs, pages 20:1–20:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.DISC.2024.20.

A Commuting reactions

We restate Lemmas 4 and 5 below for convenience and provide their proofs.

▶ **Lemma 4.** *Let* $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ *such that* $\mathbf{c}_0 \to^{r_1} \mathbf{c}_1 \to^{r_2} \mathbf{c}_2$*, where* $r_1 = (\mathbf{a}_1, \mathbf{p}_1)$ *and* $r_2 = (\mathbf{a}_2, \mathbf{p}_2)$*. If* $\mathbf{p}_1 \cap \mathbf{a}_2 = \varnothing$ *(none of the products of* r_1 *are reactants of* r_2*), then there is some* $\mathbf{c}_1'$ *such that* $\mathbf{c}_0 \to^{r_2} \mathbf{c}_1' \to^{r_1} \mathbf{c}_2$*.*

Proof. Since $\mathbf{c}_0 \to^{r_1} \mathbf{c}_1$ and $\mathbf{c}_1 \to^{r_2} \mathbf{c}_2$, $\mathbf{c}_0 \geqq \mathbf{a}_1$ and $\mathbf{c}_1 \geqq \mathbf{a}_2$. If $\mathbf{p}_1(\lambda) > 0$ (λ is a product of r_1), since we have assumed by hypothesis that is it not a reactant of r_2, then $\mathbf{a}_2(\lambda) = 0$. Consequently, $\mathbf{a}_1(\lambda) + \mathbf{a}_2(\lambda) = \mathbf{a}_1(\lambda)$, and so $\mathbf{c}_0(\lambda) \geq \mathbf{a}_1(\lambda) + \mathbf{a}_2(\lambda)$. On the other hand, if λ is not a product of r_1 ($\mathbf{p}_1(\lambda) = 0$)

$$\begin{aligned}
\mathbf{c}_0(\lambda) &= \mathbf{c}_1(\lambda) + \mathbf{a}_1(\lambda) - \mathbf{p}_1(\lambda) \\
&= \mathbf{c}_1(\lambda) + \mathbf{a}_1(\lambda) && \text{since } \mathbf{p}_1(\lambda) = 0 \\
&\geq \mathbf{a}_2(\lambda) + \mathbf{a}_1(\lambda). && \text{since } \mathbf{c}_1(\lambda) \geq \mathbf{a}_2(\lambda)
\end{aligned}$$

It then follows that $\mathbf{c}_0 \geqq \mathbf{a}_1 + \mathbf{a}_2$ implying $\mathbf{c}_0 \geqq \mathbf{a}_2$ since $\mathbf{a}_1 \geqq \mathbf{0}$. Consequently, r_2 is applicable to $\mathbf{c}_0$; let $\mathbf{c}'_1 = \mathbf{c}_0 - \mathbf{a}_2 + \mathbf{p}_2$ be the resulting configuration. It remains to show r_1 is applicable to $\mathbf{c}'_1$, i.e., $\mathbf{c}'_1 \geqq \mathbf{a}_1$. We have

$$\begin{aligned}
\mathbf{c}'_1 &= \mathbf{c}_0 - \mathbf{a}_2 + \mathbf{p}_2 \\
&\geqq (\mathbf{a}_1 + \mathbf{a}_2) - \mathbf{a}_2 + \mathbf{p}_2 && \text{since } \mathbf{c}_0 \geqq \mathbf{a}_1 + \mathbf{a}_2 \\
&= \mathbf{a}_1 + \mathbf{p}_2 \geqq \mathbf{a}_1. && \text{since } \mathbf{p}_2 \geqq \mathbf{0}
\end{aligned}$$

◀

▶ **Lemma 5.** *Let $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$ such that $\mathbf{c}_0 \upharpoonright\!\!\rightarrow^{r_1{}^{\leftarrow}} \mathbf{c}_1 \rightarrow^{r_2} \mathbf{c}_2$, where $r_1 = (\mathbf{a}_1, \mathbf{p}_1)$ and $r_2 = (\mathbf{a}_2, \mathbf{p}_2)$. If $\mathbf{a}_1 \cap \mathbf{a}_2 = \varnothing$ (none of the products of $r_1{}^{\leftarrow}$ are reactants of r_2), then there is some $\mathbf{c}'_1$ such that $\mathbf{c}_0 \rightarrow^{r_2} \mathbf{c}'_1 \upharpoonright\!\!\rightarrow^{r_1{}^{\leftarrow}} \mathbf{c}_2$.*

Proof. Since $\mathbf{c}_0 \upharpoonright\!\!\rightarrow^{r_1{}^{\leftarrow}} \mathbf{c}_1$ and $\mathbf{c}_1 \rightarrow^{r_2} \mathbf{c}_2$, $\mathbf{c}_0 \geqq \mathbf{p}_1$ and $\mathbf{c}_1 \geqq \mathbf{a}_2$. If $\mathbf{a}_1(\lambda) > 0$ (λ is a product of $r_1{}^{\leftarrow}$), since we have assumed by hypothesis that is it not a reactant of r_2, then $\mathbf{a}_2(\lambda) = 0$. Consequently, $\mathbf{p}_1(\lambda) + \mathbf{a}_2(\lambda) = \mathbf{p}_1(\lambda)$, and so $\mathbf{c}_0(\lambda) \geq \mathbf{p}_1(\lambda) + \mathbf{a}_2(\lambda)$. On the other hand, if λ is not a product of $r_1{}^{\leftarrow}$ ($\mathbf{a}_1(\lambda) = 0$)

$$\begin{aligned}
\mathbf{c}_0(\lambda) &= \mathbf{c}_1(\lambda) + \mathbf{p}_1(\lambda) - \mathbf{a}_1(\lambda) \\
&= \mathbf{c}_1(\lambda) + \mathbf{p}_1(\lambda) && \text{since } \mathbf{a}_1(\lambda) = 0 \\
&\geq \mathbf{a}_2(\lambda) + \mathbf{p}_1(\lambda). && \text{since } \mathbf{c}_1(\lambda) \geq \mathbf{a}_2(\lambda)
\end{aligned}$$

It then follows that $\mathbf{c}_0 \geqq \mathbf{p}_1 + \mathbf{a}_2$ implying $\mathbf{c}_0 \geqq \mathbf{a}_2$ since $\mathbf{a}_1 \geqq \mathbf{0}$. Consequently, r_2 is applicable to $\mathbf{c}_0$; let $\mathbf{c}'_1 = \mathbf{c}_0 - \mathbf{a}_2 + \mathbf{p}_2$ be the resulting configuration. It remains to show $r_1{}^{\leftarrow}$ is applicable to $\mathbf{c}'_1$, i.e., $\mathbf{c}'_1 \geqq \mathbf{p}_1$. We have

$$\begin{aligned}
\mathbf{c}'_1 &= \mathbf{c}_0 - \mathbf{a}_2 + \mathbf{p}_2 \\
&\geqq (\mathbf{p}_1 + \mathbf{a}_2) - \mathbf{a}_2 + \mathbf{p}_2 && \text{since } \mathbf{c}_0 \geqq \mathbf{p}_1 + \mathbf{a}_2 \\
&= \mathbf{p}_1 + \mathbf{p}_2 \geqq \mathbf{p}_1. && \text{since } \mathbf{p}_2 \geqq \mathbf{0}
\end{aligned}$$

◀

We note that for the lemma above, stating that "none of the products of $r_1{}^{\leftarrow}$ are reactants of r_2" is equivalent to stating that none of the reactants of r_1 are reactants of r_2.