Polynomial Simulations of CRN Models with Trimolecular Void Step-Cycle CRNs *

Austin Luchsinger, Aiden Massie, Robert Schweller, Evan Tomai, and Tim Wylie

University of Texas Rio Grande Valley

Abstract. We investigate the computational power of Step-Cycle Chemical Reaction Networks (CRNs) when restricted to void reactions of size at most $(3, 1)$. Step-Cycle CRNs extend the previously introduced step CRN model by repeatedly cycling through a fixed sequence of species additions and reaction phases. We show that even under the severe constraint of trimolecular void rules — which can only delete or preserve species — the model retains full computational power. In particular, we prove that (3,1) void Step-Cycle CRNs can polynomially simulate (1) any general CRN, (2) any general Step CRN, and (3) any general Step-Cycle CRN. Ultimately, these results demonstrate that the Step-Cycle model retains its complete expressive power even when restricted to $(3, 1)$-size void rules.

Keywords: Chemical Reaction Networks · Simulations · Petri-nets · Vector Addition Systems.

1 Introduction

Chemical Reaction Networks (CRNs) [5,6] are a well-established model of abstract molecular computing. CRNs transform complex real-world chemical dynamics into a simpler system consisting of molecular *species*, which change through applications of *reactions*. CRNs have been shown to be computationally equivalent to other important models of distributed systems, including Petri-nets [20] and Vector Addition Systems [17], as shown in [11,15].

There is a long history of proposed extensions to the models of distributed systems mentioned above. These extensions include inhibiting transitions by the presence of certain species [1,9,13,16], establishing a priority of firing some transitions over others [16,21,22], and enabling parallel firings of transitions [8,23]. It is known that each of these extensions allows the systems to detect when a species is absent (not possible in the traditional CRNs), which immediately unlocks Turing-universal computation [19,7].

This paper focuses on one such extension, the Step-Cycle CRN model, which reflects cyclic protocols often seen in molecular systems and laboratory practices.

* This research was supported in part by National Science Foundation Grant CCF-2329918.

In particular, we explore the expressive power of such systems when restricted to simple *void* rules that only ever remove species. The following paragraphs provide background on void rules, step-cycles, polynomial simulation, and our main contributions.

Void Reactions. Perhaps the simplest form of reactions in CRNs are *void* reactions, which do not generate new species. Void reactions can be categorized into two types: *true void* reactions that only delete reactant species, and *catalyst void* reactions that may preserve some reactant species. While conceptually simple and relevant to practical implementation, restricting systems to only use void rules can drastically limit computational power. For example, the reachability problem (which asks if a target configuration $\overrightarrow{B}$ can be reached from some initial configuration $\overrightarrow{A}$) is known to be Ackermann-complete for general CRNs [12,18], but becomes polynomial-time solvable when restricted to true void rules of size $(2, 0)$ [2].

Step-Cycle CRNs. Of particular interest to this paper is a recently introduced extension for CRNs known as the *Step* CRN model [3]. Motivated by real-world laboratory practices, the Step CRN augments a traditional CRN by introducing new copies of species into a configuration once no more reactions can be applied to it. The authors of [3,4] demonstrated that Step CRNs, even when restricted to true void reactions of small sizes such as $(2, 0)$ or $(3, 0)$, can compute threshold circuits. In [14], it was then proven that the addition of only one step strengthens the CRN reachability problem with only $(2, 0)$ rules to NP-complete.

The *Step-Cycle* CRN is a simple modification to the Step CRN model in which the system continually loops through its steps. The cyclic nature of execution makes this model particularly relevant for automating recurring operations in wet lab experiments and other time-structured settings. All of the results of this paper involve Step-Cycle CRNs that are restricted to using void rules of at most size (3,1).

Polynomial Simulation of CRNs. Although the aforementioned CRN extensions have been shown to be Turing universal, universality alone does not capture how efficiently these models can simulate one another. For this reason, recent work [7] has introduced *polynomial simulation* as a framework for comparing models not just in power, but in practical encoding complexity.

Under this framework, one CRN system T' simulates another T if configurations of T can be represented as configurations of T', and the dynamics of T are faithfully reproduced by bounded sequences of transitions in T'. A simulation is *polynomially efficient* if the number of species, rule size, volume growth, and transition steps remain polynomially bounded with respect to the simulated system. This approach provides a more nuanced comparison between CRN models. It allows us to reason about simulation overhead and efficiency, rather than treating all universal models as equally expressive. In this work, we use this framework to analyze the power of Step-Cycle CRNs that are restricted to using void rules of at most size (3,1).

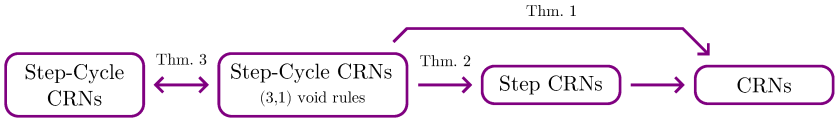


Fig. 1: Our simulation results (and corresponding theorems) with $(3, 1)$ void Step-Cycle CRNs. An arrow pointing from a system T' to another system T represents T' simulating T under polynomial simulation. Note that other than the $(3, 1)$ void rule system, every system in the hierarchy is a generalization from left to right.

Our Contributions. In this paper, we show that $(3, 1)$ void rule step-cycle CRNs can perform polynomial simulation of:

1. general CRNs,
2. general Step CRNs, and
3. general Step-Cycle CRNs.

Thus, even when restricting this experimentally motivated step-cycle model to some of the simplest void-rule reactions, full expressive power is maintained. These results and the corresponding theorems are visualized in Figure 1.

2 Preliminaries

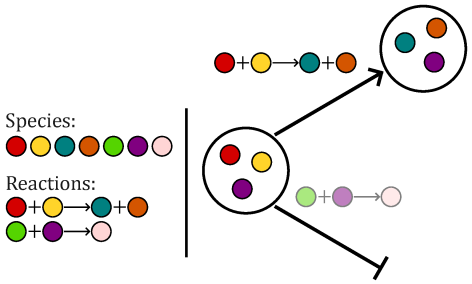
We first describe the CRN model, as well as its extensions discussed in this paper and void reactions, then describe the framework of polynomial simulation.

2.1 Chemical Reaction Network Models

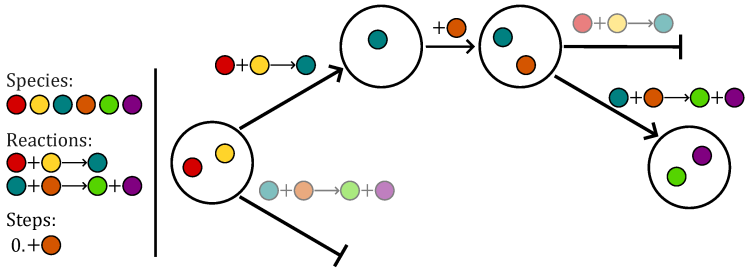
We formally define the CRN models discussed in this paper as follows. Example systems for each CRN model are shown in Figure 2.

CRNs A *chemical reaction network* (CRN) $\mathcal{C} = (\Lambda, \Gamma)$ is defined by a finite set of species Λ, and a finite set of reactions Γ where each reaction is a pair $(\overrightarrow{R}, \overrightarrow{P}) \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda$, sometimes written $\overrightarrow{R} \longrightarrow \overrightarrow{P}$, that denotes the *reactant* species consumed by the reaction and the *product* species generated by the reaction. For example, given $\Lambda = \{a, b, c\}$, the reaction $((2, 0, 0), (0, 1, 1))$ represents $2a \longrightarrow b + c$.

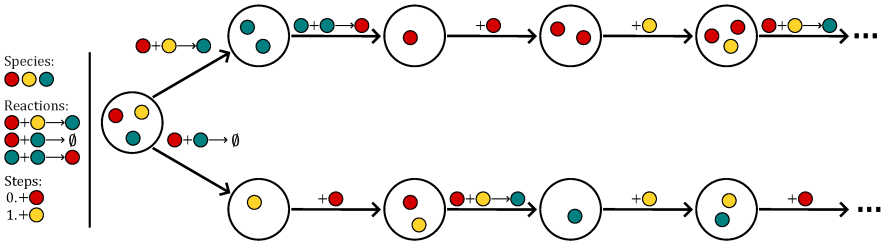
A *configuration* $\overrightarrow{C} \in \mathbb{N}^\Lambda$ of a CRN assigns integer counts to every species $\lambda \in \Lambda$, and we use notation $\overrightarrow{C}[\lambda]$ to denote that count. For a species $\lambda \in \Lambda$, we denote the configuration consisting of a single copy of λ and no other species as $\overrightarrow{\lambda}$. It is often useful to reference the set of species whose counts are not zero in a given configuration. In such cases, the notation $\{\overrightarrow{C}\}$ is used. Formally, $\{\overrightarrow{C}\} = \{\lambda \in \Lambda \mid \overrightarrow{C}[\lambda] > 0\}$, and when convenient and clear from the context,



(a) CRN Example



(b) Step CRN Example



(c) Step-Cycle CRN Example

Fig. 2: Example systems for the CRN models discussed in this paper. The blurred portions of the figure represent invalid reaction sequences. We also use colors for the alphabet of the species set of a system. (a) CRN model. (b) Step CRN model with 1 step. Note that the step species are only added once the system reaches a terminal state. (c) Step-Cycle CRN model with 2 steps. Note how the system adds the species of Step 0 again even after the addition of the species of Step 1.

we further use $\{\overrightarrow{C}\}$ to denote the configuration (vector) representation in which each element has a single copy. Finally, let $|\overrightarrow{C}| = \sum_{\lambda \in \Lambda} C[\lambda]$ denote the total number of copies of all species in a configuration, sometimes referred to as the *volume* of $\overrightarrow{C}$.

A reaction $(\overrightarrow{R}, \overrightarrow{P})$ is said to be *applicable* in configuration $\overrightarrow{C}$ if $\overrightarrow{R} \leq \overrightarrow{C}$; in other words, a reaction is applicable if $\overrightarrow{C}$ has at least as many copies of each

species as $\overrightarrow{R}$. If the reaction $(\overrightarrow{R}, \overrightarrow{P})$ is applicable, it results in configuration $\overrightarrow{C'} = \overrightarrow{C} - \overrightarrow{R} + \overrightarrow{P}$ if it occurs, and we write $\overrightarrow{C} \to \overrightarrow{C'}$. If there exists a finite sequence of configurations such that $\overrightarrow{C} \to \overrightarrow{C}_1 \to \ldots \to \overrightarrow{C}_n \to \overrightarrow{D}$, then we say that $\overrightarrow{D}$ is *reachable* from $\overrightarrow{C}$ and we write $\overrightarrow{C} \rightsquigarrow \overrightarrow{D}$. A configuration is said to be *terminal* if no reactions are applicable. We denote the set of reachable configurations of a CRN as $REACH_{\mathcal{C}}$. We define the subset of reachable configurations that are terminal as $TERM_{\mathcal{C}}$. When considering computation in CRNs, we use the notion of output-stable computation from [10].

Definition 1 (Discrete Chemical Reaction Network). *A discrete chemical reaction network (CRN) is an ordered pair (Λ, Γ) where Λ is an ordered alphabet of species, and Γ is a set of rules over Λ.*

Definition 2 (CRN Dynamics). *For a CRN (Λ, Γ) and configurations $\overrightarrow{A}$ and $\overrightarrow{B}$, we say that $\overrightarrow{A} \to_{crn}^{(\Lambda,\Gamma)} \overrightarrow{B}$ if there exists a rule $(\overrightarrow{R}, \overrightarrow{P}) \in \Gamma$ such that $\overrightarrow{R} \leq \overrightarrow{A}$, and $\overrightarrow{A} - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B}$.*

Step CRNs A step CRN is an augmentation of a basic CRN in which additional copies of species may be added to the system once it reaches a terminal configuration within a *step*. Once a step has passed (i.e. all copies of the new species are added), the system transitions to the next step if it exists. Formally, a step CRN of k steps is defined as $((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \ldots, \overrightarrow{S}_{k-1}))$, where the first element is a normal CRN (Λ, Γ) and the second is a sequence of length-$|\Lambda|$ vectors of non-negative integers denoting how many copies of each species type to add after each step. We define a *step-configuration* $\overrightarrow{C}_i$ for a step CRN as a valid configuration $\overrightarrow{C}$ over (Λ, Γ) along with an integer $i \in \{0, \ldots, k-1\}$ that denotes the configuration's step.

Definition 3 (Step Dynamics). *For a step CRN $((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \ldots, \overrightarrow{S}_{k-1}))$ and step-configurations $\overrightarrow{A}_i$ and $\overrightarrow{B}_j$, we say that $\overrightarrow{A}_i \to_s^{(\Lambda,\Gamma)} \overrightarrow{B}_j$ if either*

1. *$i = j$, there exists a rule $(\overrightarrow{R}, \overrightarrow{P}) \in \Gamma$ s.t. $\overrightarrow{R} \leq \overrightarrow{A}_i$, and $\overrightarrow{A}_i - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B}_j$, or*
2. *$i + 1 = j$, $\overrightarrow{A}_i$ is terminal, and $\overrightarrow{A}_i + \overrightarrow{S}_i = \overrightarrow{B}_j$.*

Step-Cycle CRNs A *step-cycle CRN* is a step CRN that infinitely repeats steps 0 through $k-1$: that is, once $\overrightarrow{S}_{k-1}$ is added to the terminal configuration in the $(k-1)^{th}$ step, the resulting configuration is treated as the new initial configuration for the step CRN. A *step-cycle CRN* of k steps is again formally defined as an ordered pair $((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \ldots, \overrightarrow{S}_{k-1}))$. We also define a *step-configuration* $\overrightarrow{C}_i$ for a step-cycle CRN as a valid configuration $\overrightarrow{C}$ over (Λ, Γ) where $i \in \{0, \ldots, k-1\}$.

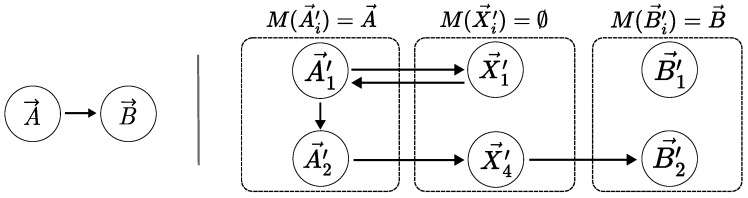


Fig. 3: (Left) A system T with states $\overrightarrow{A}$ and $\overrightarrow{B}$, and transition $\overrightarrow{A} \rightarrow_T \overrightarrow{B}$. (Right) A state-space diagram for system T' that simulates T. Here, each arrow represents some transition $\overrightarrow{X} \rightarrow_{T'} \overrightarrow{Y}$ under the dynamics of T'. Observe how T follows T' and T' models T.

Definition 4 (Step-Cycle Dynamics). *For a step-cycle CRN* $((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \ldots, \overrightarrow{S}_{k-1}))$ *and step-configurations* $\overrightarrow{A}_i$ *and* $\overrightarrow{B}_j$*, we say that* $\overrightarrow{A}_i \rightarrow_{sc}^{(\Lambda,\Gamma)} \overrightarrow{B}_j$ *if either*

1. $i = j$*, there exists a rule* $(\overrightarrow{R}, \overrightarrow{P}) \in \Gamma$ *s.t.* $\overrightarrow{R} \leq \overrightarrow{A}_i$*, and* $\overrightarrow{A}_i - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B}_j$*, or*
2. $(i+1) \mod k = j$*,* $\overrightarrow{A}_i$ *is terminal, and* $\overrightarrow{A}_i + \overrightarrow{S}_i = \overrightarrow{B}_j$*.*

Void Reactions A *void* reaction is any rule that does not create any new copies of any species type, and can only delete or preserve existing species copies. Thus, a void reaction either has no products or has products that are a subset of its reactants (in which case these products are termed *catalysts*). The formal definitions of void reactions and rule size are as follows.

Definition 5 (Void Reactions). *A reaction* $(\overrightarrow{R}, \overrightarrow{P})$ *is a* void *reaction if* $\overrightarrow{P} - \overrightarrow{R}$ *has no positive entries and at least one negative entry. There are two classes of void reactions, catalytic and true void. In catalytic void reactions, one or more reactants remain and one or more reactant is deleted after the rule is applied. In true void reactions, there are no products remaining.*

Definition 6 (Size-(i,j) Reactions). *A reaction* $(\overrightarrow{R}, \overrightarrow{P})$ *is said to be a size-*(i,j) *reaction if* $(i,j) = (|\overrightarrow{R}|, |\overrightarrow{P}|)$*. A reaction is trimolecular if* $i = 3$*, bimolecular if* $i = 2$*, and unimolecular if* $i = 1$*.*

2.2 Simulation

To define the concept of one system $T' = (\Lambda', \Gamma')$ simulating another system $T = (\Lambda, \Gamma)$ we introduce *configuration maps* and *representative configurations*. A configuration map is a polynomial-time computable function $M : \text{configs}_{T'} \rightarrow \text{configs}_T \bigcup \emptyset$ mapping at least one element in $\text{configs}_{T'}$ to each element in configs_T, and the *representative* configurations for a configuration $\overrightarrow{C} \in \text{configs}_T$ are $[\![\overrightarrow{C}]\!] = \{\overrightarrow{C'} \mid \overrightarrow{C} = M(\overrightarrow{C'})\}$, also computable in polynomial

time.[1] We say a configuration map M is *valid* if for all $\overrightarrow{C'}, \overrightarrow{D'} \in \text{configs}_{T'}$ where $M(C') \neq \emptyset$ and $M(\overrightarrow{D'}) = \emptyset$, if $\overrightarrow{C'} \rightsquigarrow \overrightarrow{D'}$ then $\overrightarrow{D'} \notin TERM_{\mathcal{T}'}$. Finally, we define the concept of *macro transitionable* for representative configurations $\overrightarrow{A'}$ and $\overrightarrow{B'}$, denoted $\overrightarrow{A'} \Rightarrow_{T'} \overrightarrow{B'}$, to mean that $\overrightarrow{B'}$ is reachable from $\overrightarrow{A'}$ in system T' through a sequence of k intermediate configurations $\langle \overrightarrow{A'}, \overrightarrow{X'_1}, \ldots, \overrightarrow{X'_k}, \overrightarrow{B'} \rangle$ such that $M(\overrightarrow{A'}) \neq \emptyset$, $M(\overrightarrow{B'}) \neq \emptyset$, and each $M(\overrightarrow{X'_i}) \in \{M(A'), \emptyset\}$. Note that k can be zero, resulting in the sequence $\langle \overrightarrow{A'}, \overrightarrow{B'} \rangle$.

Intuition. Each representative configuration set $[\![\overrightarrow{C}]\!]$ is a particular collection (of at least 1 configuration) that adheres to the strictest modeling of system T within system T'. That is, any $\overrightarrow{C'} \in [\![\overrightarrow{C}]\!]$ must be able to grow into anything that $\overrightarrow{C} = M(\overrightarrow{C'})$ can grow into, and no $\overrightarrow{C'} \in [\![\overrightarrow{C}]\!]$ may grow into something that $\overrightarrow{C} = M(\overrightarrow{C'})$ cannot grow into.

For a given configuration map and collection of representative configurations, we define the concepts of *following* and *modeling*, followed by our definition of *simulation*.

Definition 7 (It Follows). *We say system T follows system T' if whenever $\overrightarrow{A'} \Rightarrow_{T'} \overrightarrow{B'}$ and $M(\overrightarrow{A'}) \neq M(\overrightarrow{B'})$, then $M(\overrightarrow{A'}) \rightarrow_T M(\overrightarrow{B'})$.*

Definition 8 (Models). *We say system T' models system T if $\overrightarrow{A} \rightarrow_T \overrightarrow{B}$ implies that $\forall \overrightarrow{A'} \in [\![\overrightarrow{A}]\!]$, $\exists \overrightarrow{B'} \in [\![\overrightarrow{B}]\!]$ such that $\overrightarrow{A'} \Rightarrow_{T'} \overrightarrow{B'}$.*

Definition 9 (Simulation). *A system T' simulates a system T if there exists a valid, polynomial time computable function $M : \text{configs}_{T'} \rightarrow \text{configs}_T$ and polynomial time computable set $[\![\overrightarrow{C}]\!] = \{\overrightarrow{C'} \mid M(\overrightarrow{C'}) = \overrightarrow{C}\}$ for each $\overrightarrow{C} \in \text{configs}_T$, such that:*

1. *T follows T'.*
2. *T' models T.*

Polynomial Simulation. We say a simulation is *polynomial efficient* if the simulating system adheres to the following:

polynomial species and rules. The number of species and rules is at most polynomial in the number of species and rules of the simulated system.

polynomial rule size. The maximum rule size (number of products plus number of reactants) is polynomial in the maximum rule size of the simulated system.

polynomial transition sequences. For all B such that $A \rightarrow_T B$, the expected number of transitions taken to perform a macro transition from

[1] We write $M(\overrightarrow{C'}) = \emptyset$ to mean that the mapping is undefined, which is not the same as $M(\overrightarrow{C'}) = \overrightarrow{0}$.

$M(A) \Rightarrow_{T'} M(B)$, conditioned that $M(A)$ does macro transition to $M(B)$, has expected number of transitions polynomial in the number of rules and species of the simulated system based on a uniform sampling of applicable rules.

polynomial volume. Each $\overrightarrow{C'} \in [\![\overrightarrow{C}]\!]$ has a volume that is polynomially bounded by the volume of C, and for any macro transition $A' \Rightarrow_{T'} B'$, any intermediate configuration within this macrotransition has volume polynomially bounded in the volume of $M(A')$ and $M(B')$.

3 Simulation Results

Here, we show our simulation results achieved by void Step-Cycle CRNs with rule size at most $(3, 1)$. Due to space constraints, detailed explanations of the constructions are presented in this section, while the complete formal simulation proofs are left for the full version of this paper.

3.1 Table Notation

Here, we provide an overview of notation used for the added species and reactions illustrated in our tables in this section.

In detailing the added species created for the simulating system, we may describe that we add a species using the universal quantifier, such as for each reaction γ_i in Γ or each step $\overrightarrow{S}_l \in \mathcal{C}_{\mathcal{S}}$. If the added species shares the same index i/l, then this indicates that we create a unique species for each reaction, each of which we add one copy of. If the species does not have an index, this means that we add $|\Gamma|$ copies of that species instead. We may also further indicate that we add species for each individual species in the reactants and products (indicated by $\mathcal{R}_i$ and $\mathcal{P}_i$ respectively) for each reaction, or for each step species for each step-configuration. A similar case is made for the reactions created for the simulating system.

For example, if the added species of a step is described as: $\forall \gamma_i \in \Gamma : \overrightarrow{x}, \overrightarrow{G}_i$, then the step introduces $|\Gamma|$ copies of $\overrightarrow{x}$, but a single copy for each of $\overrightarrow{G}_1, \cdots, \overrightarrow{G}_{|\Gamma|}$. Similarly, if the reaction is described as $\forall \gamma_i \in \Gamma : \overrightarrow{G} + \overrightarrow{G}_i + \overrightarrow{g}_i \rightarrow \emptyset$, then we create a reaction set consisting of $\overrightarrow{G} + \overrightarrow{G}_1 + \overrightarrow{g}_1 \rightarrow \emptyset, \cdots, \overrightarrow{G} + \overrightarrow{G}_{|\Gamma|} + \overrightarrow{g}_{|\Gamma|} \rightarrow \emptyset$.

Theorem 1. *Step-Cycle CRNs can simulate any given CRN under polynomial simulation, even when restricted to at most size-*$(3, 1)$ *void rules.*

Proof. Given a CRN $\mathcal{C} = (\Lambda, \Gamma)$ and a configuration $\overrightarrow{C}$ of $\mathcal{C}$, we construct a Step-Cycle CRN $\mathcal{C}'_{\mathcal{SC}} = (\Lambda', \Gamma', (\overrightarrow{S}'_0, \ldots, \overrightarrow{S}'_8))$ and a configuration $\overrightarrow{C'}$ of $\mathcal{C}'_{\mathcal{SC}}$ that simulates $\mathcal{C}$ over $\overrightarrow{C}$ as follows.

Species and Configuration. We first add the species from $\mathcal{C}$ with no changes. For the new set of species, we first create $G, G_1, \cdots, G_{|\Gamma|}$, and $g_1, \cdots, g_{|\Gamma|}$. G

functions as a global species that randomly selects a rule $\gamma_i \in \Gamma$ to attempt to apply a rule in the set of species for all $\lambda_i \in \Lambda$. However, a single copy of both G_i and g_i must be present for G to select γ_i. $G'_1, \cdots, G'_{|\Gamma|}$ are also made; these species are used to ensure that, if γ_i was found to not be applicable, it will not be selected again until another separate reaction is successfully applied. We also construct the species $E^1_1, \ldots, E^{|\mathcal{R}_{|\Gamma|}|}_{|\Gamma|}$ for use in determining if the count of any reactant species $r_j \in \mathcal{R}_i$ is zero. a_y and a_n are created to verify if applying γ_i is either valid or invalid, respectively. We also create the species c and w for assistance in the above processes. $R^1_1, \ldots, R^{|\mathcal{R}_{|\Gamma|}|}_{|\Gamma|}$ and $P^1_1, \ldots, P^{|\mathcal{P}_{|\Gamma|}|}_{|\Gamma|}$ are made for use in the actual application of γ_i. The pair of species y_1 and y_2 are used to perform "clean-up" operations for excessive species. Finally, we create x and z. The former determines if all reactions have been selected without any successful rule application; if this is true, then z ensures that G can no longer select a reaction, effectively indicating a terminal configuration for $\mathcal{C}'_{\mathcal{SC}}$. The final species set of $\mathcal{C}'_{\mathcal{SC}}$ is then $\Lambda' = \Lambda \cup \{G, G_1, \cdots, G_{|\Gamma|}, g_1, \cdots, g_{|\Gamma|}, G'_1, \cdots, G'_{|\Gamma|}, E^1_1, \ldots, E^{|\mathcal{R}_{|\Gamma|}|}_{|\Gamma|}, R^1_1, \ldots, R^{|\mathcal{R}_{|\Gamma|}|}_{|\Gamma|}, P^1_1, \ldots, P^{|\mathcal{P}_{|\Gamma|}|}_{|\Gamma|}, a_y, a_n, c, w, x, y_1, y_2, z\}$.

Given a configuration of $\mathcal{C}$ $\overrightarrow{C}$, let the configuration of $\mathcal{C}'_{\mathcal{SC}}$ be $\overrightarrow{C}' = \overrightarrow{C} \cup \{\overrightarrow{G}_1, \cdots, \overrightarrow{G}_{|\Gamma|}, \overrightarrow{g}_1, \cdots, \overrightarrow{g}_{|\Gamma|}\}$.

Steps and Reactions. We first note that we construct the steps and reactions of $\mathcal{C}'_{\mathcal{SC}}$ such that a specific subset of the reactions will only be applicable within specific steps. Therefore, our descriptions here will mainly describe the steps in order, along with any *relevant rules* that are applicable only in those steps. The full set of added species and reactions for each step is illustrated in Table 1

The first steps (0-4) involve selecting a random reaction and determining the validity of applying it within the set of species for all $\lambda_i \in \Lambda$. In Step 0, we only add one copy of $\overrightarrow{G}$. Then the only initially applicable reaction is an instance of Reaction 1, which represents $\overrightarrow{G}$ selecting a random reaction $\gamma_i \in \Gamma$ to try simulating. We then move to Step 1, where we add a single copy of $\overrightarrow{E}^i_j$ for each individual reactant r_j for each reaction in Γ. If a $\overrightarrow{E}^i_j$ species represents a reactant that is *not* in γ_i, it will be filtered out by Reaction 2. In Step 2, we add a single copy of $\overrightarrow{w}$. $\overrightarrow{w}$ performs two operations in this step: it first deletes all $\overrightarrow{g}_i$ copies with Reaction 3, and then checks if each reactant species in γ_i is present in the configuration by Reaction 4. If the check is completely successful, then no $\overrightarrow{E}^j_i$ copy will be present after this step, indicating that γ_i can be applied, and at least one otherwise. We then progress to Step 3 and add a single copy of $\overrightarrow{a}_y$. If any $\overrightarrow{E}^j_i$ copy is still present, then $\overrightarrow{a}_y$ will be deleted (Reaction 5), but is preserved otherwise. Finally, in Step 4, we add a copy of $\overrightarrow{a}_n$. If $\overrightarrow{a}_y$ is present, then $\overrightarrow{a}_n$ is removed through Reaction 6. Else, $\overrightarrow{a}_n$ remains in the configuration.

Step 5 will apply γ_i if allowed. Here, we add one copy of species representing each individual reactant and product for each reaction in Γ, both in their "normal" form and in their $\overrightarrow{R}^j_i$ or $\overrightarrow{P}^j_i$ form. Regarding reactions, we first perform a similar procedure to Step 1, in which each reactant r_j and product p_j not from

Step	Additions		Relevant Rules	
0		$\overrightarrow{G}$	$\forall \gamma_i \in \Gamma$:	1. $\overrightarrow{G} + \overrightarrow{G}_i + \overrightarrow{g}_i \to \emptyset$ 24. $\overrightarrow{z} + \overrightarrow{G} \to \overrightarrow{z}$
1	$\forall \gamma_i \in \Gamma, \forall r_j \in \mathcal{R}_i$:	$\overrightarrow{E}_i^j$	$\forall \gamma_i \in \Gamma, \forall r_j \in \mathcal{R}_i$:	2. $\overrightarrow{G}_i + \overrightarrow{E}_i^j \to \overrightarrow{G}_i$
2		$\overrightarrow{w}$	$\forall \gamma_i \in \Gamma, \forall r_j \in \mathcal{R}_i$:	3. $\overrightarrow{w} + \overrightarrow{g}_i \to \overrightarrow{w}$ 4. $\overrightarrow{w} + \overrightarrow{r}_j + \overrightarrow{E}_i^j \to \overrightarrow{w}$
3		$\overrightarrow{a}_y$	$\forall \gamma_i \in \Gamma, \forall r_j \in \mathcal{R}_i$:	5. $\overrightarrow{E}_i^j + \overrightarrow{a}_y + \overrightarrow{w} \to \overrightarrow{E}_i^j$ 25. $\overrightarrow{a}_y + \overrightarrow{w} + \overrightarrow{z} \to \emptyset$
4		$\overrightarrow{a}_n$		6. $\overrightarrow{a}_y + \overrightarrow{a}_n + \overrightarrow{w} \to \overrightarrow{a}_y$
5	$\forall \gamma_i \in \Gamma$:	$\forall r_j \in \mathcal{R}_i$: $\overrightarrow{r}_j, \overrightarrow{R}_i^j$ $\forall p_j \in \mathcal{P}_i$: $\overrightarrow{p}_j, \overrightarrow{P}_i^j$	$\forall \gamma_i \in \Gamma$:	$\forall r_j \in \mathcal{R}_i$: 7. $\overrightarrow{G}_i + \overrightarrow{r}_j + \overrightarrow{R}_i^j \to \overrightarrow{G}_i$ 8. $\overrightarrow{E}_i^j + \overrightarrow{r}_j + \overrightarrow{R}_i^j \to \emptyset$ 9. $\overrightarrow{a}_y + \overrightarrow{r}_j + \overrightarrow{R}_i^j \to \overrightarrow{a}_y$ $\forall p_j \in \mathcal{P}_i$: 10. $\overrightarrow{G}_i + \overrightarrow{p}_j + \overrightarrow{P}_i^j \to \overrightarrow{G}_i$ 11. $\overrightarrow{a}_n + \overrightarrow{p}_j + \overrightarrow{P}_i^j \to \overrightarrow{a}_n$
6	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{x}, \overrightarrow{G}'_i$	$\forall \gamma_i \in \Gamma$:	12. $\overrightarrow{x} + \overrightarrow{G}'_i + \overrightarrow{G}_i \to \emptyset$
7	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{c}$	$\forall \gamma_i \in \Gamma$:	13. $\overrightarrow{a}_y + \overrightarrow{c} + \overrightarrow{G}'_i \to \overrightarrow{a}_y$
8	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{g_i}$ $\overrightarrow{y}_1$	$\forall \gamma_i \in \Gamma$:	$\forall r_j \in \mathcal{R}_i$: 14. $\overrightarrow{y}_1 + \overrightarrow{E}_i^j \to \overrightarrow{y}_1$ 15. $\overrightarrow{y}_1 + \overrightarrow{R}_i^j \to \overrightarrow{y}_1$ $\forall p_j \in \mathcal{P}_i$: 16. $\overrightarrow{y}_1 + \overrightarrow{P}_i^j \to \overrightarrow{y}_1$ 17. $\overrightarrow{G}'_i + \overrightarrow{g}_i \to \overrightarrow{G}'_i$ 18. $\overrightarrow{y}_1 + \overrightarrow{x} \to \overrightarrow{y}_1$ 19. $\overrightarrow{y}_1 + \overrightarrow{a}_y \to \overrightarrow{y}_1$ 20. $\overrightarrow{y}_1 + \overrightarrow{a}_n \to \overrightarrow{y}_1$ 21. $\overrightarrow{y}_1 + \overrightarrow{c} \to \overrightarrow{y}_1$
9	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{y}_2, \overrightarrow{z}$ $\overrightarrow{G}_i$	$\forall \gamma_i \in \Gamma$:	22. $\overrightarrow{y}_1 + \overrightarrow{y}_2 \to \emptyset$ 23. $\overrightarrow{z} + \overrightarrow{g}_i \to \overrightarrow{g}_i$
Return to Step 0.				

Table 1: Steps and reactions for a $(3, 1)$ void Step-Cycle CRN $\mathcal{C}'_{\mathcal{SC}}$ simulating a CRN $\mathcal{C}$.

γ_i is filtered out through Reactions 7 and 10. Then, if γ_i is found to *not* be applicable by the presence of $\overrightarrow{a}_n$, then we delete all product species p_i and $\overrightarrow{P}_i^j$ with Reaction 11. Additionally, we use Reaction 8 to delete all reactants whose corresponding $\overrightarrow{E}_i^j$ is still present (Reaction 9). The effect is that any reactants of γ_i that *were* present in $\overrightarrow{C'}$ in Step 0 are restored back; those that aren't are removed. Alternatively, if γ_i can be applied by the presence of $\overrightarrow{a}_y$, then the added reactant species r_i and $\overrightarrow{R}_i^j$ will be deleted once again (Reaction 9) and the product species are left unchanged.

The final steps (6-9) constitute of "clean-up" operations, as well as ensuring that our reaction, if not applied, cannot be chosen again until another reaction is successfully applied. In step 6, we add a copy of $\overrightarrow{G}'_i$ for each reaction, as well as $|\Gamma|$ copies of $\overrightarrow{x}$. Here, $\overrightarrow{x}$ and G'_i will delete all present copies of $\overrightarrow{G}_i$ through

Reaction 12. Note that if $\mathcal{C}'_{\mathcal{SC}}$ selects a reaction γ_i from Step 0, then at least one copy of $\overrightarrow{x}$, and thus a copy of $\overrightarrow{G}'_i$, is guaranteed to remain. Then in Step 7, $|\Gamma|$ copies of $\overrightarrow{c}$ are introduced. Only if $\overrightarrow{a}_y$ is present will it delete a copy of $\overrightarrow{c}$ and the $\overrightarrow{G}'_i$ copy if γ_i is selected. Moving to Step 8, we add a copy of $\overrightarrow{g}_i$ for each reaction and one copy of $\overrightarrow{y}_1$. $\overrightarrow{y}_1$ will remove any remaining excessive species with Reactions 14-16 and 18-21. In Reaction 17, $\overrightarrow{g}_i$ will be deleted if the corresponding $\overrightarrow{G}'_i$ species is still present; this ensures that γ_i cannot be selected again in Step 0 until another reaction is applied due to the missing $\overrightarrow{g}_i$ species required for Reaction 1. Finally, in Step 9, we add a copy of $\overrightarrow{y}_2$, one copy of $\overrightarrow{z}$, and a copy of $\overrightarrow{G}_i$ for each reaction. $\overrightarrow{y}_1$ and $\overrightarrow{y}_2$ will delete themselves with Reaction 22. Additionally, if at least one copy of $\overrightarrow{g}_i$ exists, then $\overrightarrow{z}$ will be deleted (Reaction 23). If $\overrightarrow{z}$ remains present after this step, it indicates that we have exhausted all of our reaction choices and none of them can be applied.

We define the step-configuration mapping as follows. If a single copy of $\overrightarrow{G}$ is present in the step-configuration of $\mathcal{C}_{\mathcal{SC}}$ $\overrightarrow{C}'_i$, then we map $\overrightarrow{C}'_i$ to a configuration of $\mathcal{C}$ in which the number of all species $\lambda_i \in \Lambda$ in $\overrightarrow{C}$ is equivalent to the count of all species $\lambda_i \in \Lambda$ in $\overrightarrow{C}'_i$. We note that, by the construction of $\mathcal{C}_{\mathcal{SC}}$, $\overrightarrow{G}$ can only be present in the first step configuration $\overrightarrow{C}'_0$, so any configurations with a defined mapping can be simply referred to as $\overrightarrow{C'}$.

$$M(\overrightarrow{C_i}') = \begin{cases} \sum_{\lambda' \in \Lambda} \overrightarrow{C_i}'[\lambda'] \cdot \overrightarrow{\lambda}' & \text{if } \overrightarrow{G} \in \{\overrightarrow{C_i}'\} \\ \emptyset & \text{otherwise.} \end{cases}$$

Polynomial Simulation. Let n and m be the maximum number of reactants and products of any reaction in Γ, respectively. $\mathcal{C}'_{\mathcal{SC}}$ has $\mathcal{O}(|\Lambda| + |\Gamma| \cdot (n+m))$ species and $\mathcal{O}(|\Gamma|(n+m))$ reactions. $\mathcal{C}'_{\mathcal{SC}}$ has a maximum constant rule size of $(3,1)$. The sequence of transitions is at most $\mathcal{O}(|\Gamma|(n+m))$. The volume of any representative configuration differs from the original volume by only by a polynomial amount $\mathcal{O}(|\Gamma|)$. The volume of any configuration along a macrotransition will differ from the original volume by at most $\mathcal{O}(|\Gamma|(n+m))$.

Theorem 2. *Step-Cycle CRNs can simulate any given Step CRN under polynomial simulation, even when restricted to at most* $(3,1)$ *void rules.*

Proof. Given a Step CRN $\mathcal{C}_{\mathcal{S}} = (\Lambda, \Gamma, (\overrightarrow{S}_0, \ldots, \overrightarrow{S}_k))$ and a configuration of $\mathcal{C}_{\mathcal{S}}$ $\overrightarrow{C}_S$, we construct a Step-Cycle CRN $\mathcal{C}'_{\mathcal{SC}} = (\Lambda, \Gamma, (\overrightarrow{S}'_0, \ldots, \overrightarrow{S}'_{12}))$ and a configuration of $\mathcal{C}'_{\mathcal{SC}}$ $\overrightarrow{C'}$ that simulates $\mathcal{C}_{\mathcal{S}}$ over $\overrightarrow{C}_S$ as follows.

First, we create the same species as described Theorem 1. We now construct new species exclusively for the step functionality of $\mathcal{C}_{\mathcal{S}}$. First, for each step $\overrightarrow{S}_l \in \mathcal{C}_{\mathcal{S}}$, we construct the species S_l, d_l, and E_{l+1}. S_l is used to record the current step-configuration the system is simulating, which is represented by the *missing* S_l copy. Additionally, E_{l+1} will help perform the transition from step $\overrightarrow{S}_l$ to $\overrightarrow{S}_{l+1}$ when needed while d_l removes any excessive species involved simulating

Step	Additions		Relevant Rules		
Steps 0-6 follow from Table 1.					
7	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{c}$	$\forall \gamma_i \in \Gamma$:		13. $\overrightarrow{a}_y + \overrightarrow{c} + \overrightarrow{G'}_i \to \overrightarrow{a}_y$
					14. $\overrightarrow{S}_l + \overrightarrow{s}_j + \overrightarrow{S}^j_l \to \overrightarrow{S}_l$
	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S} : \forall s_j \in \overrightarrow{S}_l$:	$\overrightarrow{s}_j, \overrightarrow{S}^j_l, \overrightarrow{E}_{l+1}, \overrightarrow{b}$	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$:		15. $\overrightarrow{S}_l + \overrightarrow{E}_{l+1} + \overrightarrow{b} \to \overrightarrow{S}_l$
					16. $\overrightarrow{x} + \overrightarrow{s}_j + \overrightarrow{S}^j_l \to \overrightarrow{x}$
					17. $\overrightarrow{x} + \overrightarrow{E}_l + \overrightarrow{b} \to \overrightarrow{x}$
8			$\forall \gamma_i \in \Gamma$:	$\forall r_j \in \mathcal{R}_i$:	18. $\overrightarrow{y}_1 + \overrightarrow{E}^j_i \to \overrightarrow{y}$
					19. $\overrightarrow{y}_1 + \overrightarrow{R}^j_i \to \overrightarrow{y}_1$
		$\overrightarrow{y}$		$\forall p_j \in \mathcal{P}_i$:	20. $\overrightarrow{y}_1 + \overrightarrow{P}^j_i \to \overrightarrow{y}_1$
	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{g}_i$			21. $\overrightarrow{G'}_i + \overrightarrow{g}_i \to \overrightarrow{G'}_i$
					22. $\overrightarrow{y}_1 + \overrightarrow{a}_y \to \overrightarrow{y}_1$
					23. $\overrightarrow{y}_1 + \overrightarrow{a}_n \to \overrightarrow{y}_1$
					36. $\overrightarrow{y}_1 + \overrightarrow{b} \to \overrightarrow{y}_1$
					37. $\overrightarrow{y}_1 + \overrightarrow{c} \to \overrightarrow{y}_1$
			$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$:		26. $\overrightarrow{y}_1 + \overrightarrow{S}^j_l \to \overrightarrow{y}_1$
9	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$:	$\overrightarrow{S}_l$	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$:		27. $\overrightarrow{S}_l + \overrightarrow{S}_l + \overrightarrow{E}_l \to \emptyset$
10	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$:	$\overrightarrow{d}_l$	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$		28. $\overrightarrow{S}_l + \overrightarrow{S}_l + \overrightarrow{d}_l \to \overrightarrow{S}_l$
					29. $\overrightarrow{S}_l + \overrightarrow{d}_l + \overrightarrow{x} \to \overrightarrow{x}$
11		$\overrightarrow{y}_2$	$\forall \overrightarrow{S}_l \in \mathcal{C}_\mathcal{S}$:		30. $\overrightarrow{y}_2 + \overrightarrow{d}_l \to \overrightarrow{y}_2$
					31. $\overrightarrow{y}_2 + \overrightarrow{x} \to \overrightarrow{y}_2$
	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{g}_i$			32. $\overrightarrow{y}_2 + \overrightarrow{y}_1 \to \overrightarrow{y}_2$
			$\forall \gamma_i \in \Gamma$:		33. $\overrightarrow{G'}_i + \overrightarrow{g}_i \to \overrightarrow{G'}_i$
12		$\overrightarrow{y}_2, \overrightarrow{z}$			34. $\overrightarrow{y}_2 + \overrightarrow{y}_2 \to \emptyset$
	$\forall \gamma_i \in \Gamma$:	$\overrightarrow{G}_i$			35. $\overrightarrow{z} + \overrightarrow{g}_i \to \overrightarrow{g}_i$
Return to Step 0.					

Table 2: Steps and reactions for a (3,1) rule size void Step-Cycle CRN $\mathcal{C}'_{\mathcal{SC}}$ simulating a Step-CRN $\mathcal{C}_\mathcal{S}$.

the steps. Furthermore, for each species $s_j \in \overrightarrow{S}_l$, we create the species S^j_l to ensure only the step species will be affected by reactions related to the simulated steps. Finally, the species b is made to assist in smaller tasks for $\mathcal{C}'_{\mathcal{SC}}$.

The final species set of $\mathcal{C}'_{\mathcal{SC}}$ is then $\Lambda' = \Lambda \cup \{G, G_1, \cdots, G_{|\Gamma|}, g_1, \cdots, g_{|\Gamma|}, G'_1, \cdots, G'_{|\Gamma|}, E^1_1, \cdots, E^{|\mathcal{R}_{|\Gamma|}|}_{|\Gamma|}, R^1_1, \cdots, R^{|\mathcal{R}_{|\Gamma|}|}_{|\Gamma|}, P^1_1, \cdots, P^{|\mathcal{P}_{|\Gamma|}|}_{|\Gamma|}, \overrightarrow{S}_0, \cdots, \overrightarrow{S}_k, \overrightarrow{S}^1_0, \cdots, \overrightarrow{S}^{|\overrightarrow{S}_k|}_k, \overrightarrow{E}_1, \cdots, \overrightarrow{E}_{k+1}, a_y, a_n, b, c, w, x, y_1, y_2, z\}$.

Constructing the configuration is the same as from Theorem 1, with the only new addition of the species $S_1, \cdots, S_k$; S_0 is intentionally excluded to represent $\mathcal{C}'_{\mathcal{SC}}$ starting at Step 0. Then, let the configuration of $\mathcal{C}'_{\mathcal{SC}}$ be $\overrightarrow{C} \cup \{\overrightarrow{G}_1, \cdots, \overrightarrow{G}_{|\Gamma|}, \overrightarrow{g}_1, \cdots, \overrightarrow{g}_{|\Gamma|}, \overrightarrow{S}_1, \cdots, \overrightarrow{S}_k\}$.

Steps and Reactions. Steps 0-6 and its respective relevant reactions are constructed as described from Table 1, with the exceptions of modifying Reaction

24 to $\overrightarrow{z} + \overrightarrow{G} \to \emptyset$ and removing Reaction 25 to ensure that we can "reset" our reaction choices back to normal once a new step has been reached. The full set of added species and reactions for steps 7-12 are presented in Table 2.

Step 7 adds the step species from $\overrightarrow{S}_l$ if allowed. This is determined by the configuration of $\mathcal{C}'_{\mathcal{SC}}$ following Step 6. Recall that, if no copy of $\overrightarrow{x}$ remains present following Step 6, then this indicates no reaction γ_i was selected, and the simulated system $\mathcal{C}_{\mathcal{S}}$ must be at a terminal step-configuration.

We add $|\Gamma|$ copies of $\overrightarrow{c}$ and a copy of $\overrightarrow{S}^j_l, \overrightarrow{E}_{l+1}$ and $\overrightarrow{b}$ for each step-configuration $\overrightarrow{S_l} \in \mathcal{C}_{\mathcal{S}}$ and a copy of $\overrightarrow{s}_j$ for all step species. If a copy of $\overrightarrow{x}$ remains present in Step 7, all copies of $\overrightarrow{s}_j$, $\overrightarrow{S}^j_l$ and $\overrightarrow{E}_{l+1}$ copies are removed by Reactions 16 and 17, preventing any step species from being added. Else, each present $\overrightarrow{S}_l$ copy removes the E_{l+1} copy and their respective step species in its "normal" ($\overrightarrow{s}_j$) and $\overrightarrow{S}^j_l$ form. This leaves one copy for all step species in $\overrightarrow{S}_l$ and $\overrightarrow{E}_{l+1}$. Additionally, $\overrightarrow{a}_y$ must be present, and thus Reaction 13 removes all $\overrightarrow{G}'_i$ copies, allowing $\overrightarrow{G}$ to select reactions when introduced again.

Steps 8-9 performs some initial clean-up operations and the transition from $\overrightarrow{S}_l$ to $\overrightarrow{S}_{l+1}$ if required. First, Step 8 introduces $\overrightarrow{y}_1$ and $|\Gamma|$ copies of $\overrightarrow{g}_i$ and largely functions similar to Step 7 from the construction of 1. The only changes are that $\overrightarrow{y}_1$ does not remove $\overrightarrow{x}$ and now removes any remaining copies of $\overrightarrow{b}$, $\overrightarrow{c}$, and $\overrightarrow{S}^j_l$ by Reactions 36, 37, and 26 respectively. In Step 9, we add a copy of $\overrightarrow{S}_l$ for each step. In the case that the step-configuration $\overrightarrow{S}_l$ has become terminal in $\mathcal{C}'_{\mathcal{SC}}$, a copy of each of $\overrightarrow{S}_l$ and E_{l+1} must be present. It then removes $\overrightarrow{S}_{l+1}$ by Reaction 27, representing preparation for the next step.

The final steps (10-12) perform the final set of "clean-up" operations. Step 10 adds one copy of $\overrightarrow{d}_l$ for each step-configuration. If $\mathcal{C}'_{\mathcal{SC}}$ has performed a transition to a new configuration, then $\overrightarrow{d}_l$ will remove the excessive $\overrightarrow{S}_l$ copies added in Step 9 by Reaction 28. Else, then a remaining copy of $\overrightarrow{x}$ will remove the $\overrightarrow{d}_l$ copy and extra copy of $\overrightarrow{S}_l$ with Reaction 29. Step 11 and 12 essentially follow the same function as Steps 8 and 9 of the construction in Theorem 1; they perform the last "clean-up" reactions.

We define the step-configuration mapping as follows. Because we are simulating a CRN system with step-configurations, we must map both the count of all species $\lambda_i \in \Lambda$ and the step-configuration $\overrightarrow{S}_i$. If a single copy of $\overrightarrow{G}$ is present in the step-configuration of $\mathcal{C}_{\mathcal{SC}}$ $\overrightarrow{C}'_i$, then we map $\overrightarrow{C}'_i$ to a step-configuration of $\mathcal{C}_{\mathcal{S}}$ in which the number of all species $\lambda_i \in \Lambda$ in $\overrightarrow{C}$ is equivalent to the count of all species $\lambda_i \in \Lambda$ in $\overrightarrow{C}'_i$. Additionally, the specific step of $\mathcal{C}_{\mathcal{S}}$ is represented by the one missing copy from all of $\overrightarrow{S}_0, \cdots, \overrightarrow{S}_k$. A special configuration mapping is present for the final step configuration $\overrightarrow{S}_{k-1}$. If a copy for all of $\overrightarrow{S}_0, \cdots, \overrightarrow{S}_k$

is present in $\overrightarrow{C}'_i$, then $\overrightarrow{C}'_i$ only maps to the step-configuration $\overrightarrow{S}_{k-1}$.

$$M(\overrightarrow{C_i}') = \begin{cases} (\sum_{\lambda' \in \Lambda} \overrightarrow{C_i}'[\lambda'] \cdot \overrightarrow{\lambda}', i) & \text{if } \overrightarrow{G} \in \{\overrightarrow{C_i}\}' \wedge \overrightarrow{S}_i \notin \{\overrightarrow{C'}\} \\ (\sum_{\lambda' \in \Lambda} \overrightarrow{C}'_{k-1}[\lambda'] \cdot \overrightarrow{\lambda}', k-1) & \text{if } \overrightarrow{G} \in \{\overrightarrow{C_i}\}' \wedge \overrightarrow{S}_0, \cdots, \overrightarrow{S}_{k-1} \in \{\overrightarrow{C'}\} \\ \emptyset & \text{otherwise.} \end{cases}$$

Polynomial Simulation. Let a be the number of steps in the simulated CRN, n be the maximum number of reactants of a reaction in Γ, m be the maximum number of products of a reaction in Γ, and p be the maximum number of introduced copies in a step in $\mathcal{C}'_{\mathcal{SC}}$. $\mathcal{C}'_{\mathcal{SC}}$ uses $\mathcal{O}(|\Lambda| + |\Gamma|(n+m))$ species and $\mathcal{O}(a \cdot p + |\Gamma|(n+m))$ reactions with maximum constant rule size of $(3,1)$. The worst case transition sequence has $\mathcal{O}(|\Gamma|)$ length. Any configuration of ${\mathcal{C}_{\mathcal{SC}}}'$ has at most an $\mathcal{O}(|\Gamma| + a)$ blowup in volume.

Theorem 3. *Step-Cycle CRNs can simulate any given Step-Cycle CRN under polynomial simulation, even when restricted to at most* $(3,1)$ *void rules.*

Proof. Given a Step-Cycle CRN $\mathcal{C}_{\mathcal{SC}} = (\Lambda, \Gamma, (\overrightarrow{S}_0, \ldots, \overrightarrow{S}_k))$ and a configuration of $\mathcal{C}_{\mathcal{SC}}$ $\overrightarrow{C}_{SC}$, we construct a Step-Cycle CRN $\mathcal{C}'_{\mathcal{SC}} = (\Lambda, \Gamma, (\overrightarrow{S}'_0, \ldots, \overrightarrow{S}'_{12}))$ and a configuration of $\mathcal{C}'_{\mathcal{SC}}$ $\overrightarrow{C'}$ that simulates $\mathcal{C}_{\mathcal{SC}}$ over $\overrightarrow{C}_{SC}$ as follows.

Species and Configuration. First, we create the same species made in Theorem 2. We then include E_0 to allow $\mathcal{C}'_{\mathcal{SC}}$ to simulate returning to $\overrightarrow{S}_0$ following the additions of all species in $\overrightarrow{S}_k$.

The configuration is constructed the same as in Theorem 2, which is $\overrightarrow{C} \cup \{\overrightarrow{G}_1, \cdots, \overrightarrow{G}_{|\Gamma|}, \overrightarrow{g}_1, \cdots, \overrightarrow{g}_{|\Gamma|}, \overrightarrow{S}_1, \cdots, \overrightarrow{S}_k\}$.

Steps and Reactions. The steps and reactions are again constructed nearly the same as in Theorem 2. The only new modification is the specific variant of Reaction 15 $\overrightarrow{S}_k + \overrightarrow{E}_{k+1} + b \to \emptyset$ from Step 7 is changed to $\overrightarrow{S}_k + \overrightarrow{E}_0 + b \to \emptyset$. The effect of this new change is that, upon $\mathcal{C}'_{\mathcal{SC}}$ adding the step species of $\overrightarrow{S}_k$, the $\overrightarrow{E}_0$ copy is not removed. Therefore, in Step 9, when an additional copy of $\overrightarrow{S}_0$ is added, Reaction 27 is executed, leaving no $\overrightarrow{S}_0$ copies remaining. This indicates $\overrightarrow{S}_0$ as the next step-configuration to simulate; $\overrightarrow{E}_1$ then becomes the next deletion species to survive, eventually triggering Reaction 27 on $\overrightarrow{S}_1$.

The configuration mapping is structured the same as in Theorem 2, although we remove the special mapping case for the final step:

$$M(\overrightarrow{C_i}') = \begin{cases} (\sum_{\lambda' \in \Lambda} \overrightarrow{C_i}'[\lambda'] \cdot \overrightarrow{\lambda}', i) & \text{if } \overrightarrow{G} \in \{\overrightarrow{C_i}\}' \wedge \overrightarrow{S}_i \notin \{\overrightarrow{C'}\} \\ \emptyset & \text{otherwise.} \end{cases}$$

Corollary 1. *Step-Cycle CRNs are Turing universal, even when restricted to at most* $(3,1)$ *void rules.*

Proof. Follows from Theorem 3 and the fact that general Step-Cycle CRNs are Turing universal [7].

4 Conclusion

We have demonstrated that the Step-Cycle CRN model retains a vast computational power, even when restricted to using only trimolecular void reactions, through simulating CRNs, step CRNs, and Step-Cycle CRNs under polynomial simulation. Nevertheless, there are still several open questions to pursue.

For example, can other extended CRN models—such as those with inhibitors, priorities, or synchrony—also achieve polynomial simulation using only small void reactions? Furthermore, can a Step-Cycle CRN, under the same rule restriction, simulate other models such as inhibitory CRNs [9], coarse-rate CRNs, or instruction-parallel CRNS? How do these instruction-parallel CRNs compare to the ability to add a large number of species in a single "step"?

Also, our simulation definition allows for nondeterministic simulation of deterministic systems. While *polynomial efficient* simulation requires that all macro transitions are expected to complete in a polynomial number of steps, this does permit cyclic reaction sequences that may never complete the macro transition. The question of whether (3,1) void rule Step-Cycle CRNs can deterministically simulate general Step-Cycle CRNs remains open.

References

1. Tilak Agerwala. Complete model for representing the coordination of asynchronous processes. Technical report, Johns Hopkins Univ., Baltimore, Md.(USA), 1974.
2. Robert M. Alaniz, Bin Fu, Timothy Gomez, Elise Grizzell, Andrew Rodriguez, Marco Rodriguez, Robert Schweller, and Tim Wylie. Reachability in restricted chemical reaction networks, 2022. `arXiv:2211.12603`.
3. Rachel Anderson, Alberto Avila, Bin Fu, Timothy Gomez, Elise Grizzell, Aiden Massie, Gourab Mukhopadhyay, Adrian Salinas, Robert Schweller, Evan Tomai, and Tim Wylie. Computing threshold circuits with void reactions in step chemical reaction networks. In *10th conference on Machines, Computations and Universality*, MCU'24, 2024.
4. Rachel Anderson, Bin Fu, Aiden Massie, Gourab Mukhopadhyay, Adrian Salinas, Robert Schweller, Evan Tomai, and Tim Wylie. Computing threshold circuits with bimolecular void reactions in step chemical reaction networks. In *International Conference on Unconventional Computation and Natural Computation*, UCNC'24, pages 253–268. Springer, 2024.
5. Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions. *Archive for Rational Mechanics and Analysis*, 19(2):81–99, jan 1965. `doi:10.1007/BF00282276`.
6. Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions ii. some addenda. *Archive for Rational Mechanics and Analysis*, 27(5):356–364, jan 1968. `doi:10.1007/BF00251438`.
7. Divya Bajaj, Jose Luis Castellanos, Ryan Knobel, Austin Luchsinger, Aiden Massie, Adrian Salinas, Pablo Santos, Ramiro Santos, Robert Schweller, and Tim Wylie. Polynomial equivalence of extended chemical reaction models. *Under Submission*, 2025.

8. Bernard Berthomieu and Dmitry A. Zaitsev. Sleptsov nets are turing-complete. *Theoretical Computer Science*, 986:114346, 2024. URL: `https://www.sciencedirect.com/science/article/pii/S030439752300659X`, `doi:10.1016/j.tcs.2023.114346`.
9. Kim Calabrese and David Doty. Rate-independent continuous inhibitory chemical reaction networks are turing-universal. In *International Conference on Unconventional Computation and Natural Computation*, pages 104–118. Springer, 2024.
10. Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural computing*, 13:517–534, 2014.
11. Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. Programmability of chemical reaction networks. In *Algorithmic bioprocesses*, pages 543–584. Springer, 2009.
12. Wojciech Czerwiński and Łukasz Orlikowski. Reachability in vector addition systems is ackermann-complete. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1229–1240. IEEE, 2022.
13. Javier Esparza and Mogens Nielsen. Decidability issues for petri nets–a survey. *arXiv preprint arXiv:2411.01592*, 2024.
14. Bin Fu, Timothy Gomez, Ryan Knobel, Austin Luchsinger, Aiden Massie, Marco Rodriguez, Adrian Salinas, Robert Schweller, and Tim Wylie. Brief announcement: Reachability in deletion-only chemical reaction networks. In *Proc. of the Symposium on Algorithmic Foundations of Dynamic Networks*, SAND, 2025.
15. Michel Henri Théodore Hack. *Decidability questions for Petri Nets.* PhD thesis, Massachusetts Institute of Technology, 1976.
16. Michel Henri Theódore Hack. Petri net language. Technical report, Massachusetts Institute of Technology, 1976.
17. Richard M Karp and Raymond E Miller. Parallel program schemata: A mathematical model for parallel computation. In *8th Annual Symposium on Switching and Automata Theory (SWAT 1967)*, pages 55–61. IEEE, 1967.
18. Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1241–1252. IEEE, 2022.
19. James Lyle Peterson. Petri net theory and the modeling of systems, 1981.
20. Carl Adam Petri. Communication with automata. 1966.
21. Phillip Senum and Marc Riedel. Rate-independent constructs for chemical computation. *PloS one*, 6(6):e21414, 2011.
22. Adam Shea, Brian Fett, Marc D Riedel, and Keshab Parhi. Writing and compiling code into biochemistry. In *Biocomputing 2010*, pages 456–464. World Scientific, 2010.
23. Pamela Bridgers Thomas. Petri net: a modeling tool for the coordination of asynchronous processes. Technical report, Tennessee Univ., Knoxville (USA); Oak Ridge Gaseous Diffusion Plant, Tenn.(USA), 1976.