# Polynomial Equivalence of Extended Chemical Reaction Models

**Divya Bajaj** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Jose-Luis Castellanos** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Ryan Knobel** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Austin Luchsinger** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Aiden Massie** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Adrian Salinas** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Pablo Santos** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Ramiro Santos** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Robert Schweller** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

**Tim Wylie** ✉
University of Texas Rio Grande Valley, Edinburg, TX, USA

## Abstract

The ability to detect whether a species (or dimension) is zero in Chemical Reaction Networks (CRN), Vector Addition Systems, or Petri Nets is known to increase the power of these models – making them capable of universal computation. While this ability may appear in many forms, such as extending the models to allow transitions to be inhibited, prioritized, or synchronized, we present an extension that directly performs this zero checking. We introduce a new *void genesis* CRN variant with a simple design that merely increments the count of a specific species when any other species' count goes to zero. As with previous extensions, we show that the model is Turing Universal. We then analyze several other studied CRN variants and show that they are all equivalent through a *polynomial* simulation with the void genesis model, which does not merely follow from Turing-universality. Thus, inhibitor species, reactions that occur at different rates, being allowed to run reactions in parallel, or even being allowed to continually add more volume to the CRN, does not add additional *simulation power* beyond simply detecting if a species count becomes zero.

**2012 ACM Subject Classification** Theory of computation → Models of computation

**Keywords and phrases** Chemical Reaction Networks, Simulations, Petri-nets, Vector Addition Systems, Bi-simulation, Turing-universality, Inhibitors

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2025.7

**Related Version** *Full Version*: `https://doi.org/10.48550/arXiv.2509.15584`

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

## 1 Introduction

**Background.** Chemical Reaction Networks [5], Vector Addition Systems [18], and Petri nets [23] are three formalisms that arose in different disciplines to study distributed/concurrent systems. Despite their distinct origins, these models are mathematically equivalent in expressive power [8, 15]. While these models are capable of very complex behavior – e.g., deciding if one configuration is reachable from another via a sequence of transitions was recently shown to be Ackermann-complete [9, 19] – they fall short of Turing-universality.

Even though these base models are not capable of universal computation, they are each right on the cusp of doing so. It is well known that extending the models in any way that allows "checking for zero" immediately results in Turing-universality [1, 16, 22]. This has been shown for model extensions allowing transition inhibition [1, 7, 11, 16], transition prioritization [16, 24, 25], synchronous (parallel) transitions [6, 28], or even continual volume increase [2, 3]. These results are typically established by showing how each extended model can simulate a register (counter) machine [22].

However, Turing equivalence alone is a blunt instrument. It tells us that models can emulate one another, but not how efficiently this can be done. We often care not just that two models are Turing-complete, but if they can be easily transformed into one another. For instance, Hack gave one such transformation between Inhibitory and Priority Petri nets [16], but discussed their equivalence in terms of languages. For this paper, we focus on the concept of simulation to draw comparisons between models.

In the literature, several notions of simulation have been proposed to capture structural or behavioral equivalence between systems. While concepts like strong/weak bisimulation [4, 10, 12, 17, 21] and pathway decomposition [26, 27] have been used to compare the expressiveness of different systems, there is typically a tradeoff between reasonably preserving dynamics and maintaining structural correspondence. Furthermore, efficiency of simulation is often either implicitly included or sometimes omitted altogether. Part of our work aims to introduce a more wieldy definition of simulation that explicitly accounts for efficiency. We give a more detailed discussion on simulation later in the paper.

**Our Contributions.** In this work, we make two main contributions:

1. We define a general-purpose notion of polynomial efficient simulation that is intended to capture a broader notion of simulation while still remaining reasonable. Our definition ensures that the simulation respects a polynomial correspondence in both time (dynamics) and space (structure). This allows us to formally compare the simulation efficiency of different CRN variants.
2. We introduce a new model, which we call Void Genesis (VG), that makes zero-testing explicit: it creates a designated species whenever a tracked species reaches zero. We use this model as a unifying model that captures the essence of zero-testing in a clean and modular way. We show that the Void Genesis model is polynomially equivalent to other known Turing-universal extensions of CRNs.

Since all of these extended models can be simulated by VG (and vice versa) with only polynomial overhead, our results establish a hub of polynomial simulation equivalence among them. This yields a stronger and more precise understanding of the relationships between these CRN variants and highlights VG simulation as a useful proof technique for adding models to this hub. Figure 1 summarizes the simulation relationships we establish here.

**Organization.** Given the number of models and results, we have arranged the paper in a way that systematically builds up understanding and techniques. Section 2 covers the formal definitions of the models and a simulation of one model with another. Section 3 provides minimum working examples in the models. The polynomial equivalence between models is proven through a series of simulations in Section 4. In Section 5, we discuss our results and some open problems.

## 2 Preliminaries

In Section 2.1, we define the extended chemical reaction network models considered in this paper with examples, and in Section 2.2 we define the concept of inter-model simulation.

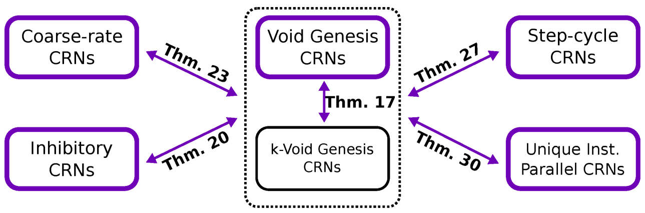


**Figure 1** CRN Model variants and their connections to Void Genesis CRNs. Theorems 1-5 each show a polynomial equivalence between two CRN variants. With the Void Genesis model as a central hub, the implication of these results is a polynomial equivalence between all models.

## 2.1 Models

Here, we define the six chemical reaction networks considered in this paper: the basic chemical reaction network model (Section 2.1.1), the Void Genesis model (Section 2.1.2), the Inhibitory CRN model (Section 2.1.3), the Coarse-Rate CRN model (Section 2.1.4), the Step-Cycle CRN model (Section 2.1.5), and the Unique Instruction Parallel model (Section 2.1.6).

### 2.1.1 Chemical Reaction Networks

A *chemical reaction network* (CRN) $\mathcal{C} = (\Lambda, \Gamma)$ is defined by a finite set of species $\Lambda$, and a finite set of reactions $\Gamma$ 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 $2a \longrightarrow b + c$; 2 $a$ species are removed, and 1 new $b$ and $c$ species are created.

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 $\lambda$ and no other species as $\vec{\lambda}$. 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, 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 $\overrightarrow{C} \rightarrow^{(\Lambda,\Gamma)}_{crn} \overrightarrow{C'}$, or simply $\overrightarrow{C} \rightarrow \overrightarrow{C'}$ when the model and CRN are clear from context. We use the same notation as configuration vectors to denote the size ($|\overrightarrow{R}|$ and $|\overrightarrow{P}|$) and explicit content ($\{\overrightarrow{R}\}$ and $\{\overrightarrow{P}\}$) of reactants and products for a reaction.

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

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

If there exists a finite sequence of configurations such that $\overrightarrow{C} \rightarrow \overrightarrow{C}_1 \rightarrow \ldots \rightarrow \overrightarrow{C}_n \rightarrow \overrightarrow{D}$, 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 also define an *initial configuration* for a CRN as its starting configuration. A *CRN System* $T$ is then defined as a pair of a CRN model and its initial configuration. The following sections define extensions of the basic CRN model by way of defining modified dynamics.

### 2.1.2 Void Genesis CRNs

A *Void Genesis CRN* $\mathcal{C}_{\mathcal{VG}} = ((\Lambda, \Gamma), z)$ is a basic CRN with a *zero* species $z \in \Lambda$ whose count is incremented whenever the count of any species other than $z$ goes to zero. See Figure 2d for an example.

▶ **Definition 3** (Void-Genesis Dynamics). *For a Void-Genesis CRN* $((\Lambda, \Gamma), z \in \Lambda)$ *and configurations* $\overrightarrow{A}$, $\overrightarrow{B_t}$ *and* $\overrightarrow{B}$, *we say that* $\overrightarrow{A} \rightarrow^{(\Lambda,\Gamma)}_{\mathcal{C}_{\mathcal{VG}}} \overrightarrow{B}$ *if there exists a rule* $(\overrightarrow{R}, \overrightarrow{P}) \in \Gamma$ *such that* $\overrightarrow{R} \leq \overrightarrow{A}$, $\overrightarrow{A} - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B_t}$, *and* $\overrightarrow{B} = \overrightarrow{B_t} + n \cdot \vec{z}$ *where* $n$ *is the cardinality of* $\{\lambda \in \Lambda \setminus \{z\} \mid \overrightarrow{A}[\lambda] \neq 0$ *and* $\overrightarrow{B_t}[\lambda] = 0\}$.

It is straightforward to show that the Void Genesis model is Turing-universal via simulating a register machine, and we do so in Section 4.7.

### 2.1.3 Inhibitory CRNs

A reaction $\gamma$ is said to be *inhibited* by a species $\lambda$ when the reaction $\gamma$ may only be applied if $\lambda$ is absent in the system. We define an inhibitor mapping $\mathcal{I} : \Gamma \rightarrow \mathbb{P}(\Lambda)$ that maps a reaction to a subset of species that inhibit the reaction. An *Inhibitory CRN* $\mathcal{C}_{\mathcal{IC}} = ((\Lambda, \Gamma), \mathcal{I})$ as defined by [7] is then a basic CRN along with the mapping $\mathcal{I}$. See Figure 2c for an example.

▶ **Definition 4** (Inhibitory Dynamics). *For a Inhibitory CRN* $((\Lambda, \Gamma), \mathcal{I})$ *and configurations* $\overrightarrow{A}$ *and* $\overrightarrow{B}$, *we say that* $\overrightarrow{A} \rightarrow^{(\Lambda,\Gamma)}_{\mathcal{C}_{\mathcal{IC}}} \overrightarrow{B}$ *if there exists a rule* $\gamma = (\overrightarrow{R}, \overrightarrow{P}) \in \Gamma$ *such that* $\overrightarrow{R} \leq \overrightarrow{A}$, $\overrightarrow{A} - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B}$, *and* $A[\lambda] = 0, \forall \lambda \in \mathcal{I}(\gamma)$.

### 2.1.4 Coarse-Rate CRNs

A *Coarse-Rate CRN* $\mathcal{C}_{CR} = ((\Lambda, \Gamma), rank)$ as introduced by [25] is a basic CRN along with a function $rank : \Gamma \rightarrow \mathbb{N}$. We define a set of reactions $\Gamma^l$ as the set of all reactions $\gamma$ where $rank(\gamma) = l$. The set of reactions $\Gamma$ is then defined as an ordered partition set given by $\{\Gamma^1, \Gamma^2, \ldots, \Gamma^n\}$. Any applicable reaction $\gamma^\ell_i$ may only be applied if no reaction $\gamma^k_j \in \Gamma^k, \forall k \in [\ell + 1, n]$ is applicable. We use $(\overrightarrow{R}, \overrightarrow{P})^\ell$ to denote a reaction $\gamma^\ell \in \Gamma^\ell$. In the context of this paper we focus on models with rank at most 2. For clarity, we will refer to reactions with rank 2 as *fast* reactions, and the ones with rank 1 as *slow* reactions. See Figure 2b for an example.

▶ **Definition 5** (Coarse-Rate Dynamics). *For a Coarse-Rate CRN* $((\Lambda, \Gamma), rank)$ *and configurations* $\overrightarrow{A}$ *and* $\overrightarrow{B}$, *we say that* $\overrightarrow{A} \rightarrow^{(\Lambda,\Gamma)}_{\mathcal{C}_{\mathcal{CR}}} \overrightarrow{B}$ *if there exists a rule* $(\overrightarrow{R}, \overrightarrow{P})^\ell \in \Gamma$ *such that* $\overrightarrow{R} \leq \overrightarrow{A}$, $\overrightarrow{A} - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B}$, *and* $\nexists$ *an applicable reaction* $\gamma^{k>\ell}$.

### 2.1.5 Step-Cycle CRNs

A *step-cycle CRN* is a step CRN [2] 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. More formally, a *step-cycle CRN* of $k$ steps is an ordered pair $((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \ldots, \overrightarrow{S}_{k-1}))$, where the first element is a normal CRN $(\Lambda, \Gamma)$ 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-cycle CRN as a valid configuration $\overrightarrow{C}$ over $(\Lambda, \Gamma)$ along with an integer $i \in \{0, \dots, k-1\}$ that denotes the configuration's step.

▶ **Definition 6** (Step-Cycle Dynamics). *For a step-cycle CRN* $((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \dots, \overrightarrow{S}_{k-1}))$ *and step-configurations* $\overrightarrow{A}_i$ *and* $\overrightarrow{B}_j$*, we say that* $\overrightarrow{A}_i \rightarrow^{(\Lambda,\Gamma)}_{\mathcal{SC}} \overrightarrow{B}_j$ *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* $\overrightarrow{A}_i + \overrightarrow{S}_i = \overrightarrow{B}_j$*.*

### 2.1.6 Unique Instruction Parallel CRNs

The Unique Instruction Parallel model modifies the dynamics of a normal CRN $(\Lambda, \Gamma)$ by applying a maximal set of compatible rules as a single transition. In this paper, we restrict this maximal set to contain only one application of any given rule, leaving the study of more relaxed parallel models for future work.

▶ **Definition 7** (Plausibly Parallel Rules). *A multiset of* $n$ *(not necessarily distinct) rules* $\{(\overrightarrow{R}_1, \overrightarrow{P}_1), \dots, (\overrightarrow{R}_n, \overrightarrow{P}_n)\}$ *are plausibly parallel for a configuration* $\overrightarrow{C}$ *over* $\Lambda$ *if the vector* $\overrightarrow{R} = \sum_{i=1}^{n} \overrightarrow{R}_i$ *is such that* $\overrightarrow{R} \leq \overrightarrow{C}$*.*

▶ **Definition 8** (Unique-Instruction Plausibly Parallel). *A plausibly parallel multiset is said to be Unique-Instruction if it contains at most one copy of any given rule (i.e., it is a set). It is considered unique-instruction maximal if it is not a proper subset of any other unique-instruction plausibly parallel set.*

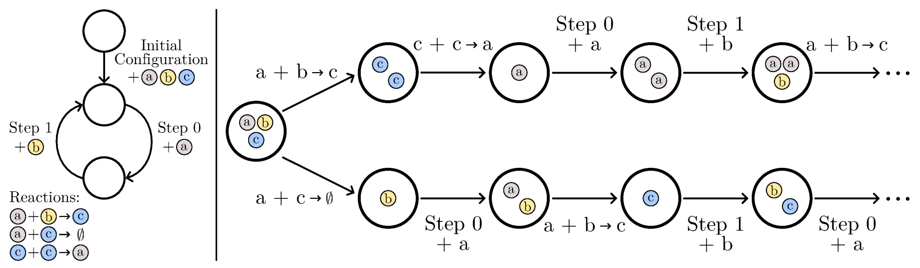
▶ **Definition 9** (Unique-Instruction Parallel Dynamics). *For a CRN* $(\Lambda, \Gamma)$ *and configurations* $\overrightarrow{A}$ *and* $\overrightarrow{B}$*, we say that* $\overrightarrow{A} \rightarrow^{(\Lambda,\Gamma)}_{\mathcal{UI}} \overrightarrow{B}$ *if there exists a unique-instruction maximal plausibly parallel set* $\{(\overrightarrow{R}_1, \overrightarrow{P}_1), \dots, (\overrightarrow{R}_k, \overrightarrow{P}_k)\}$ *for configuration* $\overrightarrow{A}$ *and rule set* $\Gamma$ *such that* $\overrightarrow{B} = \overrightarrow{A} - \sum_{i=1}^{k} \overrightarrow{R}_i + \sum_{i=1}^{k} \overrightarrow{P}_i$*.*

## 2.2 Simulation

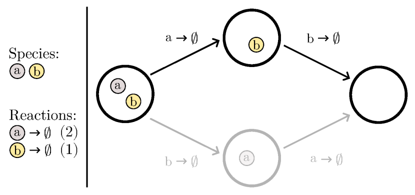
By way of Petri nets, discrete CRNs have seen various model extensions. To meaningfully compare the computational capabilities of these variants, we turn to the notion of simulation, which serves as a tool to compare the relative expressive power of each model. However, existing definitions in the literature vary in scope and applicability. Some emphasize strict structural correspondence while others focus purely on dynamic behavior. Thus, it is worthwhile to discuss why we formulate our own definition of simulation and equivalence.

Borrowed from classical process theory, (strong) *bisimulation* [21, 12, 4] is perhaps the most stringent form of equivalence. It requires that for every state and transition in one system, there exists a matching state and transition in the other, and vice versa. This strong bidirectional constraint means bisimulation ensures both behavioral and structural fidelity, and notably also implicitly guarantees efficiency.

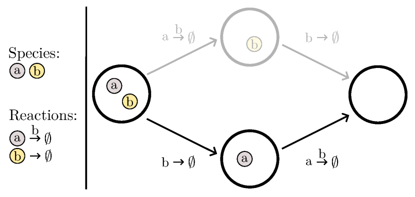
Weak bisimulation [10, 17], on the other hand, relaxes the strict step-by-step matching of bisimulation. Instead, a transition in one system may correspond to a *macrotransition* in the other: a sequence of transitions possibly allowing "hidden" or "silent" intermediate steps. This makes bisimulation more flexible and applicable to realistic implementations, but it is not without its own limitations. Pathway decomposition, as presented in [26, 27], takes a different approach. Rather than comparing reactions directly, they identify a CRN's



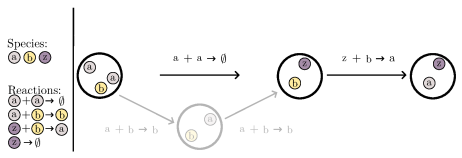
**(a)** Step-Cycle Example.



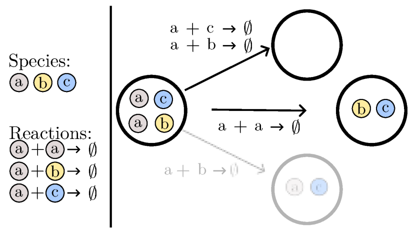
**(b)** Coarse-Rate Example.



**(c)** iCRN Example.



**(d)** Void Genesis Example.



**(e)** Unique Instruction Parallel Example.

**Figure 2** Example systems for the 5 CRN models. The blurred portions of the figure represent invalid reaction sequences. (a) Step-cycle with 2 steps. Species are added when configurations reach a terminal state. (b) Coarse-rate. The numbers next to each reaction denote the rank. The bottom reaction sequence is invalid as the reaction with rank 2 must occur first. (c) Inhibitory. The top reaction sequence is invalid as there exists a $b$ in the initial configuration, which is an inhibitor for the rule $a \to \emptyset$. (d) Void Genesis. Once the $a$ species reaches a count of 0, the $z$ species is created. In the bottom reaction sequence, the $z$ species reacts with the $b$ species, and another $z$ species is created. (e) Unique-Instruction Parallel. The bottom reaction sequence is invalid as the rule $a + b \to \emptyset$ is not a maximal set.

*formal basis* – a set of "indivisible" formal pathways that collectively define its behavior. This allows pathway decomposition to capture phenomena such as *delayed choice*, in which nondeterministic behavior is distributed across multiple steps.

Our framework seeks to strike a middle ground. We define simulation in terms of a configuration map and macrotransitions, retaining the core idea of weak bisimulation but allowing for a more general structural correspondence between systems. Since this correspondence is so general, we explicitly include a measure of efficiency with our definition. We define polynomial efficient simulation that permits abstraction and internal nondeterminism, like weak bisimulation and pathway decomposition, but captures both dynamic behavior and bounded structural transformation. We say two systems are polynomially equivalent if they can each simulate each other via our definition. This is analogous to bisimilarity [21], but in the context of efficient, weak simulation.

**Simulation Definition.** To define the concept of one CRN system $T'$ simulating another CRN system $T$ we introduce *configuration maps* and *representative configurations*. A configuration map is a polynomial-time computable function $M : \text{configs}_{T'} \to \text{configs}_T \bigcup \{\perp\}$

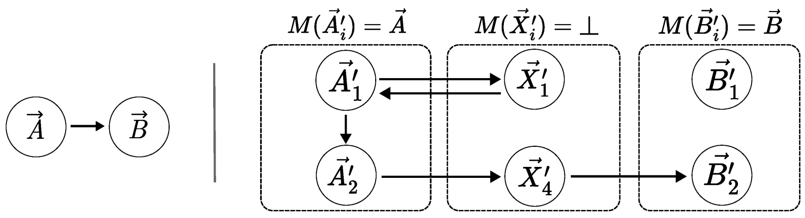


**Figure 3** (Left) A system $T$ with states $\overrightarrow{A}$ and $\overrightarrow{B}$, and transition $\overrightarrow{A} \rightarrow_T \overrightarrow{B}$. (Right) A state-space diagram for system $T'$ that simulates $T$. Here, each arrow represents some transition $\overrightarrow{X} \rightarrow_{T'}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

mapping at least one element in configs$_{T'}$ to each element in configs$_T$, and the *representative* configurations for a configuration $\overrightarrow{C} \in$ configs$_T$ are $[\![\overrightarrow{C}]\!] = \{\overrightarrow{C'} \mid \overrightarrow{C} = M(\overrightarrow{C'})\}$, also computable in polynomial time.[1] Finally, we define the concept of *single-step transition*, $\overrightarrow{A} \rightarrow_T \overrightarrow{B}$, to mean that $\overrightarrow{A}$ transitions to $\overrightarrow{B}$ in system $T$. And, the concept of *macro transitionable*, $\overrightarrow{A'} \Rightarrow_{T'} \overrightarrow{B'}$, 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 $M(\overrightarrow{A'}) \neq \perp$, $M(\overrightarrow{B'}) \neq \perp$, and each $M(\overrightarrow{X'_i}) \in \{M(\overrightarrow{A'}), \perp\}$. 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 10** (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 11** (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 12** (Simulation). *A system $T'$ simulates a system $T$ if there exists a polynomial time computable function $M$ : configs$_{T'} \rightarrow$ configs$_T$ and polynomial time computable set $[\![\overrightarrow{C}]\!] = \{\overrightarrow{C'} \mid M(\overrightarrow{C'}) = \overrightarrow{C}\}$ for each $\overrightarrow{C} \in$ 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.

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

**polynomial transition sequences.** For all $B$ such that $A \rightarrow_T B$, the expected number of transitions taken to perform a macro transition from $M(A) \Rightarrow_{T'} M(B)$, 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 $M(B')$.

▶ **Theorem 13** (Transitivity). *Given three CRN systems $T_1, T_2$ and $T_3$ such that $T_2$ simulates $T_1$ under polynomial simulation, and $T_3$ simulates $T_2$ under polynomial simulation, then $T_3$ simulates $T_1$ under polynomial simulation.*

**Proof.** Given three CRN systems $T_1$, $T_2$ and $T_3$ such that $T_2$ simulates $T_1$ under polynomial simulation, and $T_3$ simulates $T_2$ under polynomial simulation. Let $M_{21}$ : $\text{configs}_{T_2} \rightarrow \text{configs}_{T_1}$ and $M_{32}$ : $\text{configs}_{T_3} \rightarrow \text{configs}_{T_2}$ be the polynomial-time computable configuration mappings. We define a configuration mapping function $M_{31}$ : $\text{configs}_{T_3} \rightarrow \text{configs}_{T_1}$ by composing functions $M_{21}$ and $M_{32}$ as follows.

$$M_{31}(\overrightarrow{C_3}) = \begin{cases} (M_{21} \circ M_{32})(\overrightarrow{C_3}) & \text{if } M_{32}(\overrightarrow{C_3}) \neq \bot \land M_{21}(M_{32}(\overrightarrow{C_3})) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

And the set of representative configurations for a configuration $\overrightarrow{C_1} \in \text{configs}_{T_1}$ as $[\![\overrightarrow{C_1}]\!] = \{\overrightarrow{C_3} \mid \overrightarrow{C_1} = M_{31}(\overrightarrow{C_3})\}$. If the configuration $\overrightarrow{C_1}$ has a non-empty set of representative configurations in $T_2$, and each of those configurations have representative configurations in $T_3$, then this set will contain all such configurations. Therefore, if both $M_{21}$ and subsequently $M_{32}$ are defined, this set will be non-empty.

We now show that $T_3$ simulates $T_1$ by proving that $T_1$ follows $T_3$ and $T_3$ models $T_1$ for the configuration mapping and representative configurations defined above. We then show that the simulation is polynomial efficient.

***T*₁ follows *T*₃.** $T_1$ follows $T_3$ if for any two configurations $\overrightarrow{A_3}$ and $\overrightarrow{B_3}$ in $T_3$ where $M_{31}(\overrightarrow{A_3})$ and $M_{31}(\overrightarrow{B_3})$ are defined, such that $\overrightarrow{A_3} \Rightarrow_{T_3} \overrightarrow{B_3}$, and $M_{31}(\overrightarrow{A_3}) \neq M_{31}(\overrightarrow{B_3})$, then $M_{31}(\overrightarrow{A_3}) \rightarrow_{T_1} M_{31}(\overrightarrow{B_3})$. For these configurations if $M_{32}(\overrightarrow{A_3}) \neq M_{32}(\overrightarrow{B_3})$ then $M_{32}(\overrightarrow{A_3}) \rightarrow_{T_2} M_{32}(\overrightarrow{B_3})$. This is true because $T_2$ follows $T_3$.
Because $T_1$ follows $T_2$, for any two configurations $\overrightarrow{A_2}$ and $\overrightarrow{B_2}$ in $T_2$, if $\overrightarrow{A_2} \Rightarrow_{T_2} \overrightarrow{B_2}$, and $M_{21}(\overrightarrow{A_2}) \neq M_{21}(\overrightarrow{B_2})$, then $M_{21}(\overrightarrow{A_2}) \rightarrow_{T_1} M_{21}(\overrightarrow{B_2})$. A single-step transition is simply a macro-transition with one step. Therefore, if $M_{21}(M_{32}(\overrightarrow{A_3})) \neq M_{21}(M_{32}(\overrightarrow{B_3}))$, then $M_{21}(M_{32}(\overrightarrow{A_3})) \rightarrow_{T_1} M_{21}(M_{32}(\overrightarrow{B_2}))$. When $M_{31}$ is defined we can infer based on the definition of $M_{31}$ that, if $M_{31}(\overrightarrow{A_3}) \neq M_{31}(\overrightarrow{B_3})$ then $M_{31}(\overrightarrow{A_3}) \rightarrow_{T_1} M_{31}(\overrightarrow{B_3})$. Therefore, $T_1$ follows $T_3$.

***T*₃ models *T*₁.** $T_3$ models $T_1$ if for any two configurations $\overrightarrow{A_1}$ and $\overrightarrow{B_1}$ in $T_1$ such that $\overrightarrow{A_1} \rightarrow_{T_1} \overrightarrow{B_1}$, $\forall \overrightarrow{A_3} \in [\![\overrightarrow{A_1}]\!]$, $\exists \overrightarrow{B_3} \in [\![\overrightarrow{B_1}]\!]$ under $M_{31}$ such that $\overrightarrow{A_3} \Rightarrow_{T_3} \overrightarrow{B_3}$. For all such configurations $\overrightarrow{A_1}$ and $\overrightarrow{B_1}$ in $T_1$, $\forall \overrightarrow{A_2} \in [\![\overrightarrow{A_1}]\!], \exists \overrightarrow{B_2} \in [\![\overrightarrow{B_1}]\!]$ under $M_{21}$ such that $\overrightarrow{A_2} \Rightarrow_{T_2} \overrightarrow{B_2}$ because $T_2$ models $T_1$. This macro transition $\overrightarrow{A_2} \Rightarrow_{T_2} \overrightarrow{B_2}$ is represented as a sequence of single-step transitions $\overrightarrow{X_2}^i \rightarrow_{T_2} \overrightarrow{X_2}^{i+1}$ where $M_{21}(\overrightarrow{X_2}^i)$, $M_{21}(\overrightarrow{X_2}^{i+1}) \in \{\overrightarrow{A_1}, \bot\}$ for $0 \leq i < k$ as shown in Figure 4. Furthermore, $\forall \overrightarrow{X_3}^j \in [\![\overrightarrow{X_2}^i]\!]$, $\exists \overrightarrow{X_3}^\ell \in [\![\overrightarrow{X_2}^{i+1}]\!]$ under $M_{32}$ such that $\overrightarrow{X_3}^j \Rightarrow_{T_3} \overrightarrow{X_3}^\ell$ because $T_3$ models $T_2$.

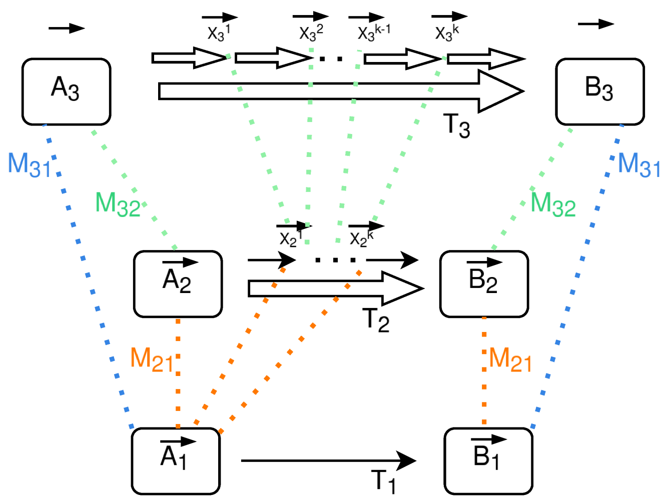


**Figure 4** A single-step transition $\overrightarrow{A_1} \rightarrow_{T_1} \overrightarrow{B_1}$ is simulated using a sequence of transitions starting at $\overrightarrow{A_2}$ through $\overrightarrow{B_2}$ in $T_2$. Each of these single-step is represented using a sequence of transitions in $T_3$.

Because $M_{32}(\overrightarrow{X_3}^j) = \overrightarrow{X_2}^i$ and $M_{32}(\overrightarrow{X_3}^\ell) = \overrightarrow{X_2}^{i+1}$ and $M_{21}(\overrightarrow{X_2}^i)$, $M_{21}(\overrightarrow{X_2}^{i+1}) \in \{\overrightarrow{A_1}, \perp\}$, therefore, $M_{31}(\overrightarrow{X_3}^j)$, $M_{31}(\overrightarrow{X_3}^\ell) \in \{\overrightarrow{A_1}, \perp\}$ for $0 \leq j < k$ and $j < \ell \leq k$. As shown in Figure 4, the sequence of macro-transitions $\overrightarrow{X_3}^j \Rightarrow_{T_3} \overrightarrow{X_3}^\ell$ can be represented as a single macro-transition $\langle \overrightarrow{A_3}, \overrightarrow{X_3}^1, \ldots, \overrightarrow{X_3}^k, \overrightarrow{B_3} \rangle$, where $M_{31}(\overrightarrow{X_3}^j) \in \{\overrightarrow{A_1}, \perp\}$ for $0 \leq j \leq k$. Also, $M_{32}(\overrightarrow{A_3}) = \overrightarrow{A_2}$ and $M_{32}(\overrightarrow{B_3}) = \overrightarrow{B_2}$. We know that $M_{21}(\overrightarrow{A_2}) = \overrightarrow{A_1}$ and $M_{21}(\overrightarrow{B_2}) = \overrightarrow{B_1}$, therefore, $M_{31}(\overrightarrow{A_3}) = \overrightarrow{A_1}$ and $M_{31}(\overrightarrow{B_3}) = \overrightarrow{B_1}$ when both $M_{21}$ and $M_{32}$ are defined. Hence, such a macro-transition $\overrightarrow{A_3} \Rightarrow_{T_3} \overrightarrow{B_3}$ models the single-step transition $\overrightarrow{A_1} \rightarrow_{T_1} \overrightarrow{B_1}$ when $\overrightarrow{A_3} \in [\![\overrightarrow{A_1}]\!]$ and $\overrightarrow{B_3} \in [\![\overrightarrow{B_1}]\!]$. Therefore, $T_3$ models $T_1$.

**Polynomial Simulation.** Given that $T_2$ simulates $T_1$ under polynomial simulation, and $T_3$ simulates $T_2$ under polynomial simulation, we now show that $T_3$ simulates $T_1$ under polynomial simulation. Let $\Lambda_i$ and $\Gamma_i$ be the set of species and reactions for the model $T_i$ for $1 \leq i \leq 3$.

1. **polynomial species and rules:** We know that $|\Lambda_2| = \mathcal{O}(|\Lambda_1|) = |\Lambda_1|^{c_1}$ and $|\Lambda_3| = \mathcal{O}(|\Lambda_2|) = |\Lambda_2|^{c_2}$, therefore, $|\Lambda_3| = |\Lambda_1|^{c_1 \cdot c_2}$. Similarly, $|\Gamma_3| = |\Gamma_1|^{c_1 \cdot c_2}$. Therefore, $|\Lambda_3| = \mathcal{O}(|\Lambda_1|)$ and $|\Gamma_3| = \mathcal{O}(|\Gamma_1|)$.
2. **polynomial rule size:** Let $\mathcal{R}_i$ be the largest rule in the model $T_i$. We know that $|\mathcal{R}_2| = \mathcal{O}(|\mathcal{R}_1|)$ and $|\mathcal{R}_3| = \mathcal{O}(|\mathcal{R}_2|)$, therefore, $|\mathcal{R}_3| = \mathcal{O}(|\mathcal{R}_1|)$.
3. **polynomial transition sequences:** The length of the transition sequence for a macro-transition in $T_3$ is bounded by $\mathcal{O}(|\Lambda_2| + |\Gamma_2|)$. Given that $|\Lambda_2| = |\Lambda_1|^{c_1}$ and $|\Gamma_2| = |\Gamma_1|^{c_1}$, therefore, each macro-transition in $T_3$ simulating a single-step in $T_1$ uses $\mathcal{O}(|\Lambda_1| + |\Gamma_1|)$ single-step transitions.
4. **polynomial volume:** For a macro-transition $\overrightarrow{A_2} \Rightarrow_{T_2} \overrightarrow{B_2}$ representing a transition $\overrightarrow{A_1} \rightarrow_{T_1} \overrightarrow{B_1}$, the volume of representative configurations $\overrightarrow{A_2}$ and $\overrightarrow{B_2}$ is polynomial in the volume of configurations $\overrightarrow{A_1}$ and $\overrightarrow{B_1}$ respectively. Similarly, for a macro-transition $\overrightarrow{A_3} \Rightarrow_{T_3} \overrightarrow{B_3}$ as shown in Figure 4, the volume of $\overrightarrow{A_3}$ and $\overrightarrow{B_3}$ is polynomial in volume of $\overrightarrow{A_2}$ and $\overrightarrow{B_2}$ respectively. Therefore, the volume of representative configurations $\overrightarrow{A_3}$ and $\overrightarrow{B_3}$ is polynomial in the volume of configurations $\overrightarrow{A_1}$ and $\overrightarrow{B_1}$, where $\overrightarrow{A_3} \Rightarrow_{T_3} \overrightarrow{B_3}$ models $\overrightarrow{A_1} \rightarrow_{T_1} \overrightarrow{B_1}$. The volume of all intermediate configurations are bound by $\overrightarrow{A_3}$ and $\overrightarrow{B_3}$. ◀

## 3 Detecting Zero: Examples in Each Model

The power each extended CRN model has over the basic CRN model stems from the ability to detect when a species has reached a count of zero. Thus, the focus of this section is twofold: to detail how each model is capable of detecting zero and to provide minimum working examples (MWEs) for ease of comprehension. For all examples, we want to check if species $s$ is in the system. The idea is that there is a species $c_s$ that turns to $n_s$ if there are no $s$'s and $y_s$ if there are $s$'s in the system while nothing else in the system changes.

**Void Genesis.** Detecting zero is explicitly done by the model. For the simplest version of the model, we assume we have no $z$ species. To check whether $s$ is zero, simply use either as a catalyst (examples on the left side). For this to work, you must maintain a single $z$. Thus, when producing any species $s$, continue to check with $s$ as a catalyst and $z$ as a reactant. As an example, the rule $x \rightarrow s$ would be modified to be implemented with the two rules on the right side.

| | |
|---|---|
| $c_s + s \rightarrow y_s + s$ | $x + s \rightarrow 2s$ |
| $s_s + z \rightarrow n_s + z$ | $x + z \rightarrow s$ |

**Inhibitory CRNs.** We can use an inhibited rule and a catalyst to detect whether a species $s$ exists or not.

| | |
|---|---|
| $c_s \xrightarrow{s} n_s$ | runs when $s$ does not exist |
| $c_s + s \rightarrow y_s + s$ | runs when $s$ does exist |

**Course-Rate CRNs.** Detecting a species $s$ only requires that there is a fast rule that uses it and a slow rule that does not.

| | |
|---|---|
| Fast Reaction | $c_s + s \rightarrow y_s + s$ (will always execute first) |
| Slow Reaction | $c_s \rightarrow n_s$ (will only execute if no $s$ exists in the system) |

**Step-Cycle CRNs.** Detecting zero is simple by using $s$ in a reaction and then going to another step. Since each step reaches a terminal configuration, it must not exist if the reaction did not occur.

| Step | Description | Add | Rules |
|---|---|---|---|
| 1 | Add something that only reacts with $s$ | a single $c_s$ | $s + c_s \rightarrow y$ |
| 2 | Check if it reacted | a single $w$ | $w + y \rightarrow y_s + s$ ($s$ exists)<br>$w + c_s \rightarrow n_s$ ($s$ not exists) |

**Unique-Instruction Parallel CRNs.** The parallel model UI takes advantage of how rules are applied to force reactions to run in a specific order depending on whether the count of a certain species is zero or not. Since we are limiting the possible rules that can run by our species selection, detecting zero can be done in this manner. The intuition is to run two independent rules, followed by a rule that uses the output of both. The "Round" indicates that all rules in this round would execute before the next round due to the maximal selection.

| Round | Description | Rules |
|---|---|---|
| 1 | Create a timing/clock species ($t_i$) and a species to use $s$ with. | $c_s \rightarrow r_c + t_1$ |
| 2 | Try to use $s$ and use the timer $t$ in another rule. | $r_c + s \rightarrow r_s$ (can only run if $s$ exists)<br>$t_1 \rightarrow t_2$ (runs even if the other rule can not) |
| 3 | Now we use the timer to see if the other rule ran | $t_2 + r_s \rightarrow y_s + s$ (there is an $s$ in the system)<br>$t_2 + r_c \rightarrow n_s$ (there are no $s$'s in the system) |

## 4 Equivalence

This section shows equivalence between the CRN models, represented as the purple bi-directional arrows in Figure 1. We first introduce a more general Void Genesis model, along with some useful notation and techniques, before presenting our equivalence results. A full version of the paper, including construction details and formal proofs, is available on arXiv.

### 4.1 Equivalence Preliminaries

#### 4.1.1 $k$-Void Genesis

Due to the complexity of some of the simulations, we first provide a more general version of the VG model that makes simulation easier. We will prove that the standard Void Genesis model can still simulate the more general model with at most a polynomial blow-up in rules, species, and expected rule applications. Essentially, the only difference is that rather than a single zero species $z$, there can be a different zero species for every species.

Formally, a *$k$-Void Genesis CRN* $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda, \Gamma), Z_\emptyset)$ is a CRN with a partial mapping function $Z_\emptyset : \Lambda_1 \to \Lambda_2$, such that $\Lambda_1 \cup \Lambda_2 = \Lambda$ and $\Lambda_1 \cap \Lambda_2 = \emptyset$, that indicates which species is created whenever another species count goes to zero (if mapped). The partition creates a distinction between *normal* species and the special *zero-counting* species, which eliminates chaining effects that could arise with zero-species being created from the elimination of other zero-species. For convenience, we use the notation $Z_\emptyset(\lambda) \to z_\lambda$ to indicate that if the count of species $\lambda$ in the system goes to zero, a $z_\lambda$ species is created.

▶ **Definition 14** ($k$-Void-Genesis Dynamics). *For a $k$-Void-Genesis CRN $((\Lambda, \Gamma), Z_\emptyset)$ and configurations $\overrightarrow{A}$, $\overrightarrow{B_t}$ and $\overrightarrow{B}$, we say that $\overrightarrow{A} \to^{(\Lambda,\Gamma)}_{\mathcal{C}_{\mathcal{KVG}}} \overrightarrow{B}$ if there exists a rule $(\overrightarrow{R}, \overrightarrow{P}) \in \Gamma$ such that $\overrightarrow{R} \leq \overrightarrow{A}$, $\overrightarrow{A} - \overrightarrow{R} + \overrightarrow{P} = \overrightarrow{B_t}$, and $\overrightarrow{B} = \overrightarrow{B_t} + \overrightarrow{Z}$ where $\overrightarrow{Z} = \sum_{\lambda \in C} \vec{z}_\lambda$, and $C = \{\lambda \in \Lambda | \overrightarrow{A}[\lambda] \neq 0$ and $\overrightarrow{B_t}[\lambda] = 0\}$.*

#### 4.1.2 Notation and Techniques

**Notation.** To simplify the proofs and for consistency, we use the following notation for rules. For rule $i$, we have $\mathcal{G}_i = (\overrightarrow{\mathcal{R}_i}, \overrightarrow{\mathcal{P}_i})$. We make use of the $\{\mathcal{R}\}$ and $|\mathcal{R}|$ notation defined in the preliminaries, as well the difference between a configuration with many species $\overrightarrow{X}$ versus a configuration with only a single species $\vec{X}$ by the over arrow used.

One technique that is used in our simulations is to maintain a single global leader species $G$ that selects which rule $\mathcal{G}_i$ to execute, and then a sequence of rules to either sequentially consume the reactants or sequentially break down the product of the other models' rules to execute zero-checking before managing zero species.

**Sequential Reactants.** Some of the results also benefit from processing the reactants of a rule one at a time as well. This is needed when consuming a species might result in its count as zero and some action needs to be taken. In order to verify that all reactants exists (and thus the rule could be executed), we use all reactants as a catalyst with a global leader species $G$. Table 1a describes the necessary rules for consuming reactants sequentially. For rule $i$, this creates $R_i^1$ that will begin the chain of consuming one species at a time. We create a $C_i^j$ species in order to check whether $r_j$ is now zero or not. This step can be skipped if no check is needed. We create the species $R_i^j$ sequentially, which is then consumed if $r_j$ is in the system. Systems that use this technique will have one copy of $G$ in the initial configuration and no $R$ or $C$ species. For shorthand, we will represent this series of rules as $\vec{G} + \overrightarrow{\mathcal{R}_i} \dashrightarrow \vec{G} + \overrightarrow{\mathcal{P}_i}$.

**Table 1** An overview of two techniques. (a) A procedure to consume the reactants of some rule sequentially in order to test whether a rule can be applied and whether zero species are (or need to be) created. This is abbreviated for rule $i$ as $\overrightarrow{\mathcal{R}_i} \dashrightarrow \overrightarrow{\mathcal{P}_i}$. (b) Creating products sequentially by consuming all reactants and then using counter species to create each product with a different rule until all have been created. This may be needed to decrease the number of rule combinations while handling zero species. Note that the zero species $z_{p_j}$ is consumed when $p_j$ is created. If $p_j$ still exists, then it is used as a catalyst. This process is denoted for some rule $i$ as $\overrightarrow{\mathcal{R}_i} \rightarrowtail\!\!\rightarrowtail\!\!\rightarrow \overrightarrow{\mathcal{P}_i}$.

| | |
|---|---|
| | $\vec{G} + \overrightarrow{\mathcal{R}_i} \to \vec{R}_i^1 + \overrightarrow{\mathcal{R}_i}$ |
| $\forall r_j \in \mathcal{R}_i :$ | $\vec{R}_i^j + \vec{r}_j \to \vec{C}_i^j$ (check for zero) |
| | $\vec{C}_i^j + \vec{z}_{r_j} \to \vec{R}_i^{j+1} + \vec{z}_{r_j}$ ($r_j$ is zero) |
| | $\vec{C}_i^j + \vec{r}_j \to \vec{R}_i^{j+1} + \vec{r}_j$ ($r_j$ is not zero) |
| | $\vec{R}_i^{\lvert\mathcal{R}_i\rvert+1} \to \vec{G} + \overrightarrow{\mathcal{P}_i}$ |

**(a)** Using reactants sequentially $\overrightarrow{\mathcal{R}_i} \dashrightarrow \overrightarrow{\mathcal{P}_i}$.

| | |
|---|---|
| | $\vec{G} + \overrightarrow{\mathcal{R}_i} \to \vec{P}_i^1$ |
| $\forall p_j \in \mathcal{P}_i :$ | $\vec{z}_{p_j} + \vec{P}_i^j \to \vec{p}_j + \vec{P}_i^{j+1}$ (remove zero) |
| | $\vec{p}_j + \vec{P}_i^j \to \vec{p}_j + \vec{p}_j + \vec{P}_i^{j+1}$ (no zero) |
| | $\vec{P}_i^{\lvert\mathcal{P}_i\rvert+1} \to \vec{G}$ |

**(b)** Creating products sequentially $\overrightarrow{\mathcal{R}_i} \rightarrowtail\!\!\rightarrowtail\!\!\rightarrow \overrightarrow{\mathcal{P}_i}$.

**Sequential Products.** This technique only allows a single $G$ species to exist in the system at a time. Table 1b describes the necessary rules where a rule $\mathcal{G}_i = (\overrightarrow{\mathcal{R}_i}, \overrightarrow{\mathcal{P}_i})$ is chosen if the reactants exist in the system. If this is the case, then each product is created sequentially by having the species $P_i^1, \ldots, P_i^{|\mathcal{P}_i|}$ in the system (one at a time, starting from $P_i^1$), then consuming each species $P_i^j$ to create the product $p_j$ and the next $P_i^{j+1}$ until all products have been produced. If species $P_i^j$ exists in the system, it means that for rule $\mathcal{G}_i$, the first $j-1$ products have been created, and either the next $p_j$ will be created or the process will end (return $G$) if there are no more products.

In a simulating system, this requires an additional number of rules and species on the order of the largest number of products in any rule. The final rule returns the single leader species. Note that we make use of a zero species $z_{p_j}$ for every product species $p_j$, i.e., $Z_\emptyset(p_j) = z_{p_j}$. For shorthand, we will represent this set of rules for rule $i$ as $\overrightarrow{\mathcal{R}_i} \rightarrowtail\!\!\rightarrowtail\!\!\rightarrow \overrightarrow{\mathcal{P}_i}$. Thus, the entire set of rules and species from Table 1b, that initiate with a global leader $G$, would be given in shorthand as $\vec{G} + \overrightarrow{\mathcal{R}_i} \rightarrowtail\!\!\rightarrowtail\!\!\rightarrow \vec{G} + \overrightarrow{\mathcal{P}_i}$.

## 4.2 $k$-Void Genesis equivalence with Void Genesis

▶ **Lemma 15.** *The $k$-Void Genesis model can simulate the Void Genesis model.*

**Construction.** Given a Void Genesis model $\mathcal{C}_{\mathcal{VG}} = ((\Lambda, \Gamma), z)$, let $Z_\emptyset(\lambda) = z$, $\forall \lambda \in \Lambda \setminus \{z\}$. Then the $k$-Void Genesis model $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda, \Gamma), Z_\emptyset)$ is equivalent.

▶ **Lemma 16.** *The Void Genesis model can simulate the $k$-Void Genesis model.*

**Construction.** This result is straightforward using the methods to sequentially consume reactants as previously defined. We slightly modify them with specifics as shown in Table 2. Whenever we consume a reactant $r_j$, we check if the single $z$ species exists and if it does, create the specific $z_{r_j}$ species if $r_j$ is mapped in $Z_\emptyset$.

Given a $k$-Void Genesis model $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda, \Gamma), Z_\emptyset)$, we create a VG CRN $\mathcal{C}_{\mathcal{VG}} = ((\Lambda', \Gamma'), z)$. We let $\Lambda' = \Lambda \cup \{G\} \cup \{z_\lambda : \lambda \in \Lambda \setminus \{z\}\} \cup S_R$ where $S_R = \{R_i^j : 1 \leq i \leq |\Gamma|, 1 \leq j \leq |\mathcal{R}_i| + 1\}$. We can simulate having specific $z_j$ species for all $j$ species by keeping the $z$ species count at 0, and checking whether we consumed the last $j$.

**Table 2** (a) The rules for the simulation of $k$-Void Genesis by Void Genesis. For each applicable rule, we create a set of rules that sequentially consume each reactant $r_j$, and if its count is now zero, it creates the $z_{r_j}$ species (unless it is unmapped, then $z$ is simply consumed). Only one of the zero check rules is added based on the mapping, which is why they are grouped. (b) The reactions that simulate the rule $\vec{a} + 2\vec{b} \to 2\vec{a} + \vec{c}$ from the $k$-VG model within the VG model.

| | |
|---|---|
| | $\vec{G} + \overrightarrow{\mathcal{R}_i} \to \vec{R}_i^1 + \overrightarrow{\mathcal{R}_i}$ |
| $\forall r_j \in \mathcal{R}_i :$ | $\vec{R}_i^j + \vec{r}_j \to \vec{C}_i^j$ (check for zero) |
| | $\begin{cases} \vec{C}_i^j + \vec{z} \to \vec{R}_i^{j+1} \text{(unmapped species)} \\ \vec{C}_i^j + \vec{z} \to \vec{R}_i^{j+1} + \vec{z}_{r_j} \text{(zero species)} \end{cases}$ |
| | $\vec{C}_i^j + \vec{r}_j \to \vec{R}_i^{j+1} + \vec{r}_j$ (count not zero) |
| | $\vec{R}_i^{\lvert\mathcal{R}_i\rvert+1} \to \vec{G} + \overrightarrow{\mathcal{P}_i}$ |

**(a)** Zero checking reactants $\overrightarrow{\mathcal{R}_i} \dashrightarrow \overrightarrow{\mathcal{P}_i}$.

| $\vec{G} + \vec{a} + 2\vec{b} \to \vec{R}_1^1 + \vec{a} + 2\vec{b}$ | |
|---|---|
| $\vec{R}_1^1 + \vec{a} \to \vec{C}_1^1$ | $\vec{C}_1^2 + \vec{b} \to \vec{R}_1^3 + \vec{b}$ |
| $\vec{C}_1^1 + \vec{z} \to \vec{R}_1^2 + \vec{z}_a$ | $\vec{R}_1^3 + \vec{b} \to \vec{C}_1^3$ |
| $\vec{C}_1^1 + \vec{a} \to \vec{R}_1^2 + \vec{a}$ | $\vec{C}_1^3 + \vec{z} \to \vec{R}_1^4 + \vec{z}_b$ |
| $\vec{R}_1^2 + \vec{b} \to \vec{C}_1^2$ | $\vec{C}_1^3 + \vec{b} \to \vec{R}_1^4 + \vec{b}$ |
| $\vec{C}_1^2 + \vec{z} \to \vec{R}_1^3 + \vec{z}_b$ | $\vec{R}_1^4 \to \vec{G} + 2\vec{a} + \vec{c}$ |

**(b)** Example reaction set.

**Table 3** (a) Reactions for an Inhibitory CRN to simulate any given $k$-VG CRN. (b) Reactions for a $k$-VG CRN to simulate any given Inhibitory CRN.

| | |
|---|---|
| $\forall \gamma_i \in \Gamma :$ | 1. $\overrightarrow{\mathcal{R}}_i \xrightarrow{I} \overrightarrow{e}_{\{\mathcal{R}_i\}} + \overrightarrow{\mathcal{P}}_i + \lvert\{\mathcal{R}_i\}\rvert \cdot \vec{I}$ |
| $\forall \lambda_i \in \Lambda_1 :$ | 2. $\vec{e}_{\lambda_i} + \vec{I} + \vec{\lambda}_i \to \vec{\lambda}_i$ |
| | 3. $\vec{e}_{\lambda_i} + \vec{I} \xrightarrow{\lambda_i} \vec{z}_{\lambda_i}$ |

**(a)** iCRN simulating $k$-VG.

| | |
|---|---|
| $\forall \gamma_i \in \Gamma :$ | 1. $\vec{G} + \overrightarrow{\mathcal{R}_i} + \overrightarrow{Z_i} \to \vec{G}_i + \overrightarrow{\mathcal{R}_i} + \overrightarrow{Z_i}$ |
| | 2. $\vec{G}_i + \overrightarrow{\mathcal{R}_i} \twoheadrightarrow\!\!\to \vec{G} + \overrightarrow{\mathcal{P}_i}$ |

**(b)** $k$-VG simulating iCRN.

▶ **Theorem 17.** *The Void Genesis model is equivalent under polynomial simulation to the $k$-Void Genesis model.*

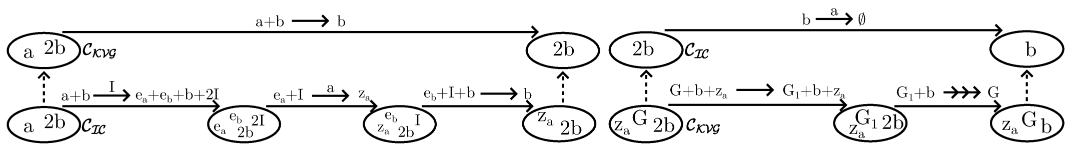
## 4.3 Inhibitory CRNs

▶ **Lemma 18.** *Inhibitory CRNs can simulate any given $k$-VG CRN under polynomial simulation.*

**Construction.** Given a $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda, \Gamma), Z_\emptyset)$, we construct an Inhibitory CRN $\mathcal{C}_{\mathcal{IC}} = ((\Lambda', \Gamma'), \mathcal{I})$. We let $\Lambda' = \Lambda \cup \{I, e_{\lambda_1}, \cdots, e_{\lambda_{|\Lambda_1|}}\}$ where $I$ and $e_{\lambda_1}, \ldots, e_{\lambda_{|\Lambda_1|}}$ check if any species of $\Lambda_1$ used in a simulated reaction have a resulting count of zero. Recall that $\Lambda_1$ is the set of non-zero species in $\Lambda$. Each reaction in $\Gamma$ $\gamma_i$ is simulated using Reaction 1 from Table 3$a$. Reactions 2 and 3 check if any reactants of $\gamma_i$ reached a count of zero following $\gamma_i$'s simulation. If so, then a corresponding zero species is produced. Figure 5a shows an example of an Inhibitory CRN simulating a $k$-VG CRN with $\Lambda = \{a, b\}$ and a reaction that consumes the species $a$.

▶ **Lemma 19.** *$k$-VG CRNs can simulate any Inhibitory CRN under polynomial simulation.*

**Construction.** Given an Inhibitory CRN $\mathcal{C}_{\mathcal{IC}} = ((\Lambda, \Gamma), \mathcal{I})$, we construct a $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda', \Gamma'), Z_\emptyset)$. We let $\Lambda' = \Lambda \cup \{G, G_1, \ldots, G_{|\Gamma|}, P_i^1, \ldots, P_{|\Gamma|}^{|\mathcal{P}_{|\Gamma|}|+1}, z_{\lambda_1}, \ldots, z_{\lambda_{|\Lambda|}}\}$ where the global species $G$ is consumed to produce $G_i$ when selecting a reaction $\gamma_i \in \Gamma$. The $z$ species are the zero-species, and the species $P_i^1 \ldots P_i^{|\mathcal{P}_i|+1}$ are produced for the sequential product generation as described in Table 1b. Let $Z_i$ represent the set of $z$ species corresponding to inhibitors of $\gamma_i$. Each reaction $\gamma_i$ in $\Gamma$ is represented by the two reactions given in Table 3$b$, where Reaction 1 checks if $\gamma_i$ is applicable (indicated by the presence of reactants of $\gamma_i$ and $Z_i$) and Reaction 2 applies the reaction by generating products sequentially.



a+b → b
a 2b
$\mathcal{C}_{\mathcal{KVG}}$
2b
a+b $\xrightarrow{I}$ e_a+e_b+b+2I
e_a+I $\xrightarrow{a}$ z_a
e_b+I+b → b
a 2b
$\mathcal{C}_{\mathcal{IC}}$
e_a e_b 2I 2b
z_a e_b I 2b
z_a 2b
b $\xrightarrow{a}$ ∅
2b
$\mathcal{C}_{\mathcal{IC}}$
b
G+b+z_a → G_1+b+z_a
G_1+b ⇛ G
z_a G 2b
$\mathcal{C}_{\mathcal{KVG}}$
z_a G_1 2b
z_a G b

**(a)** $\mathcal{C}_{\mathcal{IC}} \to \mathcal{C}_{\mathcal{KVG}}$ **(b)** $\mathcal{C}_{\mathcal{KVG}} \to \mathcal{C}_{\mathcal{IC}}$

**Figure 5** (a) A rule application in the *simulated* $k$-void genesis CRN $\mathcal{C}_{\mathcal{KVG}}$ and the equivalent rule application sequence in the *simulating* inhibitory CRN $\mathcal{C}_{\mathcal{IC}}$. The dashed arrows represent mapping a configuration of $\mathcal{C}_{\mathcal{IC}}$ to a configuration of $\mathcal{C}_{\mathcal{KVG}}$. (b) The other direction.

**Table 4** (a) Reactions for a Coarse-Rate CRN to simulate any given $k$-VG CRN. (b) Reactions for a $k$-VG CRN to simulate any given Coarse-Rate CRN.

| | |
|---|---|
| | **Fast Reactions** (Rank 2) |
| $\forall\ \gamma_i \in \Gamma:$ | 1. $\vec{G} + \overrightarrow{\mathcal{R}_i} \to \vec{x} + \overrightarrow{e}_{\{\mathcal{R}_i\}} + \overrightarrow{\mathcal{P}_i}$ |
| $\forall \lambda_i \in \Lambda_1:$ | 2. $\vec{x} + \vec{e}_{\lambda_i} + \vec{\lambda_i} \to \vec{x} + \vec{\lambda_i}$ |
| | 3. $\vec{y} + \vec{e}_{\lambda_i} \to \vec{y} + \vec{z}_{\lambda_i}$ |
| | **Slow Reactions** (Rank 1) |
| | 4. $\vec{x} \to \vec{y}$ |
| | 5. $\vec{y} \to \vec{G}$ |

**(a)** Coarse-Rate CRN simulating $k$-VG CRN.

| | | |
|---|---|---|
| | 1. | $\vec{G} + \vec{g}_i + \overrightarrow{\mathcal{R}_i} \to \vec{r}_1 + \overrightarrow{\mathcal{P}_i}$ |
| $\forall \gamma_i^2 \in \Gamma^2,$ | 1$b$. | $\vec{G} + \vec{g}_i + \vec{z}_k \to \vec{G} + \vec{z}_k$ |
| $z_k \in \{\mathcal{R}_i\}$: | 2. | $\vec{r}_j + \vec{g}_j \to \vec{r}_{j+1} + \vec{g}_j$ |
| | 3. | $\vec{r}_j + \vec{z}_{g_j} \to \vec{r}_{j+1} + \vec{g}_j$ |
| | 4. | $\vec{r}_{\vert\Gamma^2\vert+1} \to \vec{G}$ |
| | 5. | $\vec{G} + \vec{z}_{g_1} + \ldots + \vec{z}_{g_{\vert\Gamma^2\vert}} \to \vec{s}$ |
| | 6. | $\vec{t} \to \vec{G} + \vec{g}_1 + \ldots + \vec{g}_{\vert\Gamma^2\vert}$ |
| $\forall \gamma_i^1 \in \Gamma^1$: | 7. | $\vec{s} + \overrightarrow{\mathcal{R}_i} \rightarrow\!\!\!\rightarrow\!\!\!\rightarrow \vec{t} + \overrightarrow{\mathcal{P}_i}$ |

**(b)** $k$-VG CRN simulating Coarse-Rate CRN.

Figure 5$b$ shows an example of a $k$-VG CRN simulating an Inhibitory CRN with $\Lambda = \{a, b\}$ and a reaction that consumes $b$ if $a$ is absent.

▶ **Theorem 20.** *The Void Genesis model is equivalent under polynomial simulation to the Inhibitory model.*

## 4.4 Coarse-Rate CRNs

▶ **Lemma 21.** *Coarse-Rate CRNs can simulate any given $k$-VG CRN under polynomial simulation.*

**Construction.** Given a $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda, \Gamma), Z_\emptyset)$, we construct a Course-Rate CRN $\mathcal{C}_{\mathcal{CR}} = ((\Lambda', \Gamma'), rank)$. Let $\Lambda' = \Lambda \cup \{x, y, G, e_{\lambda_1}, \cdots, e_{\lambda_{|\Lambda_1|}}\}$ where $G$ is used to select a reaction and species $x$ and $y$ check if any non-zero species in $\Lambda$ have a count of zero. Additionally, $e_{\lambda_1}, \cdots, e_{\lambda_{|\Lambda_1|}}$ are used to check the counts of the $\lambda_i \in \Lambda_1$ that are relevant to the simulated reaction, where $\Lambda_1$ is the set of non-zero species in $\Lambda$. As shown in Table 4$a$, each reaction $\gamma_i \in \Gamma$ is simulated using 3 fast reactions which check if any involved reactant reached a count of zero and generates the corresponding zero species if so, along with applying the original reaction. Finally the 2 slow reactions use up any $x$ or $y$ species in the system to produce $G$. This allows the system to apply another reaction in $\Gamma$. Figure 6 shows an example of a Coarse-Rate CRN simulating a $k$-VG CRN with $\Lambda = \{a, b\}$ and a reaction that consumes $a$, hence, producing $z_a$.

▶ **Lemma 22.** *$k$-VG CRNs can simulate any given Coarse-Rate CRN under polynomial simulation.*

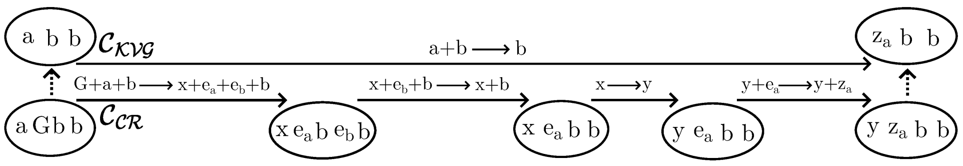


**Figure 6** A rule application in the *simulated* $k$-void genesis CRN $\mathcal{C}_{\mathcal{KVG}}$ and the equivalent rule application sequence in the *simulating* Coarse-Rate CRN $\mathcal{C}_{\mathcal{CR}}$.

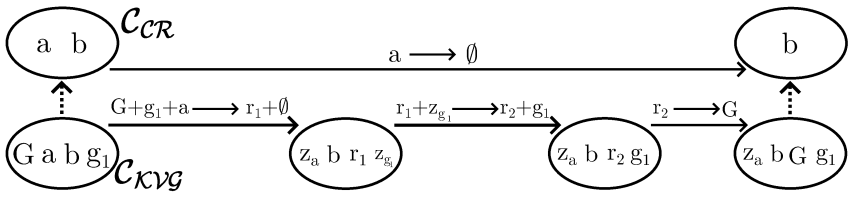


**Figure 7** A rule application in the *simulated* Coarse-Rate CRN $\mathcal{C}_{\mathcal{CR}}$ and the equivalent rule application sequence in the *simulating* $k$-void genesis CRN $\mathcal{C}_{\mathcal{KVG}}$.

**Construction.** Given a Coarse-Rate CRN $\mathcal{C}_{\mathcal{CR}} = ((\Lambda, \Gamma), rank)$, we construct a $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda', \Gamma'), Z_\emptyset)$. Let $\Lambda' = \Lambda \cup \{G, g_1, \cdots, g_{|\Gamma^2|}, z_{g_1}, \cdots, z_{g_{|\Gamma^2|}}, z_{\lambda_1}, \cdots, z_{\lambda_{|\Lambda|}}, P_1^1, \cdots, P_{|\Gamma^1|}^{|\mathcal{P}_{|\Gamma^1|}|+1}, s, t\}$ as follows. $G$ and a random $g_i$ species is consumed to select a random fast reaction $\gamma_i^2 \in \Gamma^2$ as illustrated in Table 4b. We additionally add $P_1^1, \cdots, P_{|\Lambda|}^{|\mathcal{P}_{|\Lambda|}|+1}$ for generating the products of any slow reaction. We add species $s$ to signify that no fast reactions are applicable and species $t$ to signify that a slow reaction has been applied. Finally, we add a zero species for all $g_i$ species. Figure 7 shows an example of a $k$-VG CRN simulating a Course-Rate CRN with $\Lambda = \{a, b\}$ and a fast reaction that deletes $a$ and a slow reaction that deletes $b$.

▶ **Theorem 23.** *The Void Genesis model is equivalent under polynomial simulation to the Course-Rate model.*

## 4.5 Step-Cycle CRNs

▶ **Lemma 24.** *Step-Cycle CRNs can simulate any given k-VG CRN under polynomial simulation.*

**Construction.** Given a $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda, \Gamma), Z_\emptyset)$, we construct a Step-Cycle CRN $\mathcal{C}_{\mathcal{SC}} = ((\Lambda', \Gamma'), \overrightarrow{S}_0)$. We let $\Lambda' = \Lambda \cup \{G, y, x, w, e_{\lambda_1}, \ldots, e_{\lambda_{|\Lambda_1|}}, z_{\lambda_1}, \ldots, z_{\lambda_{|\Lambda_2|}}\}$ and $\overrightarrow{S}_0 = \vec{y}$. The $G$ species is used to select a reaction, and $y$ is used to check if any non-zero species in $\Lambda$ have a count of zero. Finally, $e_{\lambda_1}, \cdots, e_{\lambda_{|\Lambda_1|}}$ are used to check the counts of the $\lambda_i \in \Lambda_1$ that are relevant to the simulated reaction, where $\Lambda_1$ is the set of non-zero species in $\Lambda$. Each reaction $\gamma_i \in \Gamma$ is simulated by Reaction 1 as given in Table 5a. Reactions 2 consumes $e$ species for any reactant whose count does not reach zero. Once the system is terminal, a $y$ is added so as to generate $z$ species corresponding to the remaining reactants using Reaction 3. Another $y$ is added to enable Reaction 4. Once no reaction is applicable in $\Gamma$, another $y$ is added to enable Reaction 5, which allows Reaction 6 to execute infinitely. Finally, the system reaches a terminal configuration and prevents the volume from growing infinitely. Figure 8 shows an example of a Step-Cycle CRN simulating a $k$-VG CRN with $\Lambda = \{a, b\}$ and a reaction that consumes the species $a$.

**Table 5** (a) Reactions for a Step-Cycle CRN to simulate any given k-VG CRN. (b) Reactions for a k-VG CRN to simulate any given Step-Cycle CRN.

| | |
|---|---|
| $\forall \gamma_i \in \Gamma :$ | $1.\vec{G} + \overrightarrow{\mathcal{R}_i} \rightarrow \overrightarrow{e}_{\{\mathcal{R}_i\}} + \overrightarrow{\mathcal{P}_i}$ |
| $\forall \lambda_i \in \Lambda_1 :$ | $2.\vec{\lambda}_i + \vec{e}_{\lambda_i} \rightarrow \vec{\lambda}_i$ |
| | $3.\vec{y} + \vec{e}_{\lambda_i} \rightarrow \vec{y} + \vec{z}_{\lambda_i}$ |
| | $4.\vec{y} + \vec{y} \rightarrow \vec{G}$ |
| | $5.\vec{G} + \vec{y} \leftrightarrow \vec{G} + \vec{w}$ |
| | $6.\vec{w} \leftrightarrow x$ |

**(a)** Step-Cycle CRN sim. $k$-VG CRN.

| | |
|---|---|
| $\forall \gamma_i \in \Gamma,\ z_k \in \{\mathcal{R}_i\}:$ | $1.\vec{G} + \vec{g}_i \dashrightarrow \vec{r}_1 + \overrightarrow{\mathcal{P}_i}$ |
| | $2.\vec{r}_j + \vec{g}_j \rightarrow \vec{r}_{j+1} + \vec{g}_j$ |
| | $3.\vec{r}_j + \vec{z}_{g_j} \rightarrow \vec{r}_{j+1} + \vec{g}_j$ |
| | $4.\vec{r}_{\lvert\Gamma\rvert+1} \rightarrow \vec{G}$ |
| | $5.\vec{G} + \vec{z}_{g_1} + \ldots + \vec{z}_{g_{\lvert\Gamma\rvert}} \rightarrow \vec{s}$ |
| | $6.\vec{t} \rightarrow \vec{G} + \vec{g}_1 + \ldots + \vec{g}_{\lvert\Gamma\rvert}$ |
| $\forall S_i \in S \setminus \{S_{k-1}\} :$ | $7.\vec{s} + \vec{s}_i \twoheadrightarrow \vec{t} + \vec{s}_{i+1} + \overrightarrow{S}_i$ |
| | $8.\vec{s} + \vec{s}_{k-1} \twoheadrightarrow \vec{t} + \vec{s}_0 + \overrightarrow{S}_{k-1}$ |

**(b)** $k$-VG CRN simulating a Step-Cycle CRN.

**Table 6** (a) Checking reactants sequentially $\overrightarrow{\mathcal{R}_i} \dashrightarrow \overrightarrow{\mathcal{P}_i}$ (b) Undoing reaction selection if not enough reactants exist for rule $i$.

| | |
|---|---|
| | $\vec{G} + \vec{g}_i \rightarrow \vec{R}_1^1 + \vec{z}_{g_i}$ |
| $\forall \lambda_j \in \mathcal{R}_i :$ | $\vec{R}_i^k + \vec{\lambda}_j \rightarrow \vec{R}_i^{k+1} + \vec{\lambda}'_j$ |
| | $\vec{R}_i^{\lvert\mathcal{R}_i\rvert+1} + \overrightarrow{\mathcal{R}_i}' \twoheadrightarrow \vec{r}_1 + \overrightarrow{\mathcal{P}}_i$ |

**(a)** Check reactants.

| | |
|---|---|
| $\forall \lambda_j \in \mathcal{R}_i :$ | $\vec{R}_i^k + \vec{z}_{\lambda_j} \rightarrow \vec{R}_i^{k^-} + \vec{z}_{\lambda_j}$ |
| | $\vec{R}_i^{k^-} + \vec{z}_{\lambda_j} \rightarrow \vec{R}_i^{k-1^-} + \vec{\lambda}_j$ |
| | $\vec{R}_i^{k^-} + \vec{\lambda}'_j \rightarrow \vec{R}_i^{k^-} + \vec{\lambda}_j$ |
| | $\vec{R}_1^{1^-} \rightarrow \vec{G}$ |

**(b)** Undo reactants.

▶ **Lemma 25.** *$k$-VG CRNs can simulate any given Step-Cycle CRN under polynomial simulation.*

**Construction.** Given a Step-Cycle CRN $\mathcal{C}_{\mathcal{SC}} = ((\Lambda, \Gamma), (\overrightarrow{S}_0, \overrightarrow{S}_1, \ldots, \overrightarrow{S}_{k-1}))$, we construct a $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda', \Gamma'), Z_\emptyset)$. We let $\Lambda' = \Lambda \cup \{G, \lambda'_1, \ldots, \lambda'_{\lvert\Lambda\rvert}, z_{\lambda_1}, \ldots, z_{\lambda_{\lvert\Lambda\rvert}}, r_1, \ldots, r_{\lvert\Gamma\rvert+1}, \quad s, s_0, \ldots, s_{k-1}, t\} \cup \{g_i, z_{g_i}, R_i^j, R_i^{j^-}, P_i^l : 1 \leq i \leq \lvert\Gamma\rvert, 1 \leq j \leq \lvert\mathcal{R}_i\rvert + 1, 1 \leq l \leq \lvert\mathcal{P}_i\rvert + 1\}$. The $G$ species is used to select a reaction represented by $g_i$. The reactants are checked sequentially, converting each reactant into $\lambda'_i$. Species $r_i$ reintroduce each consumed $g_i$ into the system. Species $s_i$ represent step $i$, with species $s$ and $t$ used to transition between steps. Reaction 1 in Table 5b attempts to apply $\gamma_i$ by checking if enough of each reactant exists, as shown in Table 6. If it is successful, then it simulates the reaction and begins the process of adding $g_1, \ldots, g_{\lvert\Gamma\rvert}$ back into the system (which is carried out in reactions 2, 3, and 4). If a species in $\gamma_i$ does not exist in the system, then the reactants are reintroduced into the system as shown in Table 6, which also removes $g_i$ from the system, producing $z_{g_i}$. If there are no executable reactions, then reaction 5 is executed to produce an $s$ species. This $s$ species then reacts with the current "step" of the system, denoted by species $s_i$. Reaction 7 or 8 then runs, introducing the species corresponding to the step. These reactions also introduce the species $t$, which then reintroduces $G$ along with each $g_i$ through reaction 6. Figure 9 shows an example of a $k$-VG CRN simulating a Step-Cycle CRN with $\Lambda = \{a, b\}$ and reaction $a + b \rightarrow b$.

▶ **Theorem 26.** *The Void Genesis model is equivalent under polynomial simulation to the Step-Cycle model.*

**Extension to deletion-only rules.** Given the recent result in [20], these results extend to give the following corollary.

▶ **Corollary 27.** *Even when restricted to at most $(3, 1)$ void rules, the Step-Cycle model is equivalent under polynomial simulation to the Void Genesis model.*

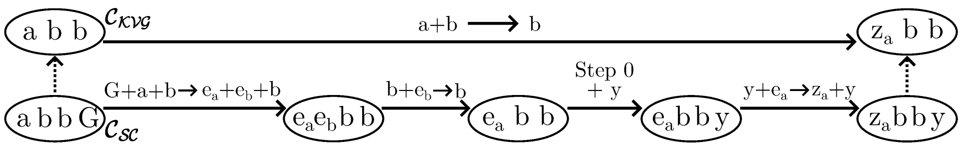


**Figure 8** A rule application in the *simulated* $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}}$ and the equivalent rule application sequence in the *simulating* Step-Cycle CRN $\mathcal{C}_{\mathcal{SC}}$.

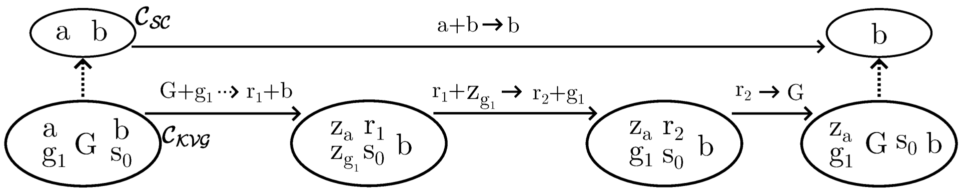

$\mathcal{C}_{\mathcal{SC}}$
a b
a+b → b
b
a $g_1$ G b $s_0$
$\mathcal{C}_{\mathcal{KVG}}$
G+$g_1$ → $r_1$+b
$z_a$ $r_1$ $z_{g_1}$ $s_0$ b
$r_1$+$z_{g_1}$ → $r_2$+$g_1$
$z_a$ $r_2$ $g_1$ $s_0$ b
$r_2$ → G
$z_a$ $g_1$ G $s_0$ b

**Figure 9** A rule application in the *simulated* Step-Cycle CRN $\mathcal{C}_{\mathcal{SC}}$ and the equivalent rule application sequence in the *simulating* $k$-VG CRN $\mathcal{C}_{\mathcal{KVG}}$.

**Table 7** (a) Reactions for a UI parallel CRN to simulate any given VG CRN. Here, $\overrightarrow{e_{\{\mathcal{R}_i\}}}$ is an $e_{\lambda_j}$ species created for each of the $j$ different reactants in $\mathcal{R}_i$ (only one $e$ is created if multiple copies of that species are used). (b) Reactions for a kVG CRN to simulate any given UI parallel CRN.

| | |
|---|---|
| $\forall \gamma_i \in \Gamma :$ | $1.\vec{G}+\overrightarrow{\mathcal{R}_i} \to \vec{G_i}+\overrightarrow{e_{\{\mathcal{R}_i\}}}+\overrightarrow{\mathcal{P}_i}$ |
| $\forall \lambda_j \in \Lambda_1 :$ | $2.\vec{e}_{\lambda_j} \to \vec{r}_j^1 + \vec{t}_j^1$ |
| | $3.\vec{r}_j^1 + \vec{\lambda}_i \to \vec{r}_j^2$<br>$\vec{t}_j^1 \to \vec{t}_2$ |
| | $4.\vec{t}_2 + \vec{r}_j^2 \to \vec{\lambda}_i + \vec{E}$<br>$\vec{t}_2 + \vec{r}_j^1 \to \vec{z} + \vec{E}$ |
| | $5.\vec{G}_i + \lvert\{\mathcal{R}_i\}\rvert \cdot \vec{E} \to \vec{G}$ |

**(a)** UI parallel CRN simulating a VG CRN.

| | |
|---|---|
| $\forall \gamma_i \in \Gamma,$<br>$\forall r_j \in \mathcal{R}_i :$ | $\vec{G}_i + \vec{z}_{r_j} \to \vec{N}_i + \vec{X}_i + \vec{z}_{r_j}$ (can not run) |
| seq.<br>reacts.<br><br>$j = \lvert\mathcal{R}_i\rvert$ | $\vec{G}_i + \vec{r}_1 \to \vec{R}_i^2$<br>$\vec{R}_i^j + \vec{r}_j \to \vec{R}_i^{j+1}$<br>$\vec{R}_i^j + \vec{z}_{r_j} \to \vec{N}_i + \vec{X}_i + \vec{r}_1 + \ldots + \vec{r}_{j-1} + \vec{z}_{r_j}$<br>$\vec{R}_i^j + \vec{r}_j \to \vec{I}_i + \vec{X}_i$ (rule selected) |
| | $X_1 + \ldots + X_{\lvert\Gamma\rvert} \to \vec{F}$ (rules selected) |
| $\forall \gamma_i \in \Gamma$ | $\vec{I}_i + \vec{F} \to \vec{F}_i + \vec{F} + \vec{\mathcal{P}}_i$ (output)<br>$\vec{N}_i + \vec{F} \to \vec{F}_i + \vec{F}$ (not run) |
| reset | $\vec{F} + \vec{F}_1 + \ldots + \vec{F}_{\lvert\Gamma\rvert} \to \vec{G}_i + \ldots + \vec{G}_{\lvert\Gamma\rvert}$ |

**(b)** $k$-VG simulating UI parallel CRN.

## 4.6 Unique-Instruction Parallel Model

▶ **Lemma 28.** *UI parallel CRNs can simulate any given VG CRN.*

**Construction.** Given a Void-Genesis CRN $\mathcal{C}_{\mathcal{VG}} = ((\Lambda, \Gamma), z)$, we construct the UI parallel CRN $\mathcal{C}_{\mathcal{UI}} = (\Lambda', \Gamma')$. We create $\Lambda' = \Lambda \cup \{G, E, t^2\} \cup \{e_{\lambda_j}, t_j^1, r_j^1, r_j^2, G_i : i \in \{1, \ldots, |\Gamma|\}, j \in \{1, \ldots, |\{\mathcal{R}_i\}|\}\}$. The global species $G$ selects a rule $\gamma_i \in \Gamma$ non-deterministically, and produces $G_i$ to disallow Reaction 1 from running again prematurely, a set of species $e_\lambda$, and the products of $\gamma_i$, as shown in Table 7*a*. The $e_\lambda$ species then creates a timer species $t_j^i$, and a checker species $r_j^1$, which is used to check if a species exists in the system. In Reaction 3, the rules run in parallel, so the timer species goes down, and at the same time, we check if any checker species have incremented. Reaction 4 either reintroduces the consumed species or creates the zero species $z$, based on the checker species. Both reactions produce the species $E$, which is later used in Reaction 5 to reintroduce the $G$ species, allowing the system to run another reaction. Figure 10 shows an example of a UI CRN simulating a VG CRN with $\Lambda = \{a, b\}$ and the rule $a + b \to b$. .

▶ **Lemma 29.** *$k$-VG CRNs can simulate any given UI Parallel CRN.*

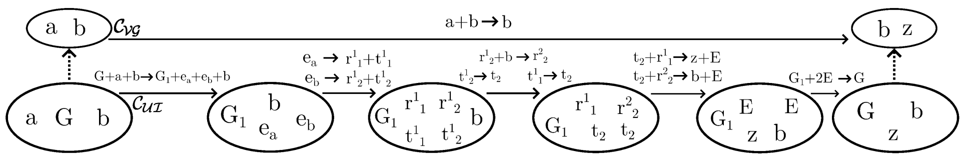


**Figure 10** A rule application in the *simulated* void genesis CRN $\mathcal{C}_{\mathcal{VG}}$ and the equivalent rule application sequence in the *simulating* Unique Instruction CRN $\mathcal{C}_{\mathcal{UI}}$.

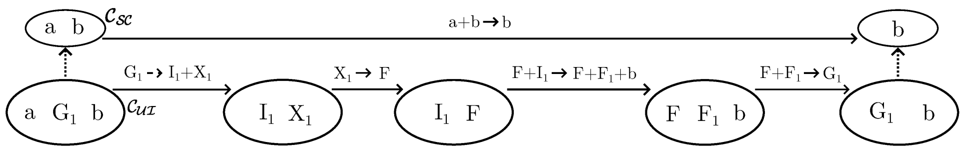


**Figure 11** A rule application in the *simulated* Unique Instruction CRN $\mathcal{C}_{\mathcal{UI}}$ and the equivalent rule application sequence in the *simulating* $k$-void genesis CRN $\mathcal{C}_{\mathcal{KVG}}$.

**Construction.** Given a UI parallel CRN $\mathcal{C}_{\mathcal{UI}} = (\Lambda, \Gamma)$, we construct a $k$-Void-Genesis CRN $\mathcal{C}_{\mathcal{KVG}} = ((\Lambda', \Gamma'), Z_\emptyset)$ as described. Let $\Lambda' = \Lambda \cup \{F\} \cup \{G_i, N_i, I_i, F_i, R_i^j : i \in \{1, \ldots, |\Gamma|\}, j \in \{1, \ldots, |\mathcal{R}_i|\}\} \cup \{z_{r_j} : r_j \in \mathcal{R}_i s.t. 1 \leq i \leq |\Gamma|\}$. For each reaction $\gamma_i \in \Gamma$, a $G_i$ species exists which attempts to sequentially consume the reactants of $\gamma_i$. If this is successful, it is indicated by the production of an $I_i$ species. If some reactant is missing, it returns any previously consumed reactants and creates an $N_i$ species instead. Both procedures create an $X_i$ species, and when a copy of $X_i$ exists for each reaction, this indicates a maximal set of simulated reactions has been chosen and the $F$ species is created. The $N_i$ or $I_i$ species then turn into a $F_i$ species, as well as produce the products of $\gamma_i$ if $I_i$ was converted. Once a $F_i$ species exists for each reaction, they combine to reset the rule selection process by creating all the $G_i$ species again. The full reaction set is shown in Table 7b, and Figure 11 shows an example of a $k$-VG CRN simulating a UI CRN with $\Lambda = \{a, b\}$ and reaction $a + b \to b$.

▶ **Theorem 30.** *The Void Genesis model is equivalent under polynomial simulation to the Unique-Instruction parallel model.*

## 4.7 Register Machine

In this section, we show that Void Genesis CRNs are Turing Universal by constructing a simulation of a standard register machine.

▶ **Theorem 31.** *$k$-VG CRNs are Turing Universal.*

**Proof.** Given a Register Machine (RM) with $s_1, \ldots, s_n$ states–each with either a $inc(r_l, s_j)$ or $dec(r_l, s_j, s_k)$ instruction–and $r_1, \ldots, r_m$ registers, we construct the $k$-VG CRN as follows. Each register and state will be represented by a species, and the instructions will be encoded in the rules. The initial configurations should have one copy of $s_1$ (assume $s_1$ is the RM's starting state) and the register species' counts should be equal to the registers they represent.

Table 8 shows what rules should be made for each state of a given RM. Because $k$-VG CRNs can simulate any given RM, $k$-VG CRNs are Turing Universal. ◀

Although it can be inferred from this result and the simulation equivalence results that all the models studied here are Turing Universal, we do not give formal proofs of this due to space.

**Table 8** Rules for a Void Genesis CRN to simulate a given Register Machine.

| Instruction | Relevant Rules | Instruction | Relevant Rules |
|---|---|---|---|
| | $\vec{s}_j + \vec{r}_i \to \vec{s}_k + \vec{r}_i + \vec{r}_i$ | | $\vec{s}_j + \vec{r}_i \to \vec{s}_k$ |
| $s_j : inc(r_i, s_k)$ | $\vec{s}_j + \vec{z}_{r_i} \to \vec{s}_k + \vec{r}_i$ | $s_j : dec(r_i, s_k, s_l)$ | $\vec{s}_j + \vec{z}_{r_i} \to \vec{s}_l + \vec{z}_{r_i}$ |
| | | | $Z_\emptyset(r_i) = z_{r_i}$ |

## 5 Conclusion

In this paper, we demonstrate equivalence through polynomial simulation between 5 natural extensions to the CRN model. We centralize these simulations around the Void Genesis CRN model, as this model's ability to detect zero is one of the simplest augmentations to a regular CRN. We then show that Void Genesis CRNs are Turing Universal, implying that Step-Cycle CRNs, Inhibitory CRNs, Parallel CRNs, and Coarse-Rate CRNs are also Turing Universal. While this work is complete in proving equivalence between these models, there are still several interesting open problems to consider (some of which are shown in Figure 1):

- We aim to explore constrained versions of our simulation definition that recover existing notions as special cases. Does restricting the configuration map to be consistent with an underlying species-species mapping immediately results in weak bisimulation? If that underlying map is a total bijective function, does that yield strong bisimulation?
- For iCRNs, a rule is inhibited by the existence of one or more species. Our definition effectively uses a logical `OR` (inhibition is only false when all inhibitor counts are zero). A natural extension is to consider inhibition functions using other logic (e.g., AND – a reaction is inhibited only when all of its inhibitors are present).
- Coarse-Rate CRNs are limited to 2 ranks for reactions. A natural generalization of this model is to allow for $k$ different ranks ($k$-rate CRNs). What is the relationship between Coarse-Rate CRNs and $k$-rate CRNs?
- Even when limited to only void reactions (rules where no species are created), step CRNs are able to compute threshold circuits [2, 3]. Does this suggest that Step-Cycle CRNs, even when limited to only void reactions (void Step-Cycle), are still Turing Universal?
- Are there more efficient ways to simulate these augmented CRNs using VG CRNs?
- What is the complexity of reachability in restricted instances of each model, such as [13, 14]? As mentioned, we know the complexity with step-cycles [20], but deletion-only rules have not been explored in detail in the other models.

## References

1 Tilak Agerwala. Complete model for representing the coordination of asynchronous processes. Technical report, Johns Hopkins Univ., Baltimore, Md.(USA), 1974.

2 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.

3 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. `doi:10.1007/978-3-031-63742-1_18`.

4 Marco Antoniotti, Carla Piazza, Alberto Policriti, Marta Simeoni, and Bud Mishra. Taming the complexity of biochemical models through bisimulation and collapsing: theory and practice. *Theoretical Computer Science*, 325(1):45–67, 2004. `doi:10.1016/J.TCS.2004.03.064`.

5 Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions. *Archive for Rational Mechanics and Analysis*, 19(2):81–99, January 1965. doi:10.1007/BF00282276.

6 Bernard Berthomieu and Dmitry A. Zaitsev. Sleptsov nets are turing-complete. *Theoretical Computer Science*, 986:114346, 2024. doi:10.1016/j.tcs.2023.114346.

7 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. doi:10.1007/978-3-031-63742-1_8.

8 Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. Programmability of chemical reaction networks. In *Algorithmic bioprocesses*, pages 543–584. Springer, 2009. doi:10.1007/978-3-540-88869-7_27.

9 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.

10 Qing Dong. A bisimulation approach to verification of molecular implementations of formal chemical reaction networks. *Master's thesis, Stony Brook University*, 2012.

11 Javier Esparza and Mogens Nielsen. Decidability issues for petri nets–a survey. *arXiv preprint arXiv:2411.01592*, 2024. doi:10.48550/arXiv.2411.01592.

12 Jean-Claude Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13(2-3):219–236, 1990. doi:10.1016/0167-6423(90)90071-K.

13 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.

14 Bin Fu, Timothy Gomez, Ryan Knobel, Austin Luchsinger, Aiden Massie, Marco Rodriguez, Adrian Salinas, Robert Schweller, and Tim Wylie. Reachability in deletion-only chemical reaction networks. In *Proc. of the International Conference on DNA Computing and Molecular Programming*, DNA, 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 Robert Johnson, Qing Dong, and Erik Winfree. Verifying chemical reaction network implementations: a bisimulation approach. *Theoretical Computer Science*, 765:3–46, 2019. doi:10.1016/J.TCS.2018.01.002.

18 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. doi:10.1109/FOCS.1967.27.

19 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.

20 Austin Luchsinger, Aiden Massie, Robert Schweller, Evan Tomai, and Tim Wylie. Polynomial simulations of crn models with trimolecular void step-cycle crns. In *Proc. of the 22th International Conference on Unconventional Computing and Natural Computing*, UCNC, 2025.

21 Robin Milner. *Communication and concurrency*, volume 84. Prentice hall Englewood Cliffs, 1989.

22 James Lyle Peterson. Petri net theory and the modeling of systems, 1981.

23 Carl Adam Petri. *Communication with automata*. PhD thesis, Technische Universitat Darmstadt, 1966.

24 Phillip Senum and Marc Riedel. Rate-independent constructs for chemical computation. *PloS one*, 6(6):e21414, 2011.

25 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. URL: http://psb.stanford.edu/psb-online/proceedings/psb10/shea.pdf.

26 Seung Woo Shin. *Compiling and verifying DNA-based chemical reaction network implementations*. PhD thesis, California Institute of Technology, 2012.

27 Seung Woo Shin, Chris Thachuk, and Erik Winfree. Verifying chemical reaction network implementations: a pathway decomposition approach. *Theoretical Computer Science*, 765:67–96, 2019. doi:10.1016/J.TCS.2017.10.011.

28 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.