Reachability in Deletion-Only Chemical Reaction Networks

Bin Fu
University of Texas Rio Grande Valley, Edinburg, TX, USA

Ryan Knobel
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

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

Timothy Gomez
Massachusetts Institute of Technology, Cambridge, MA, USA

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

Marco Rodriguez
Massachusetts Institute of Technology, Cambridge, MA, USA

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

Abstract

For general discrete Chemical Reaction Networks (CRNs), the fundamental problem of reachability – the question of whether a target configuration can be produced from a given initial configuration – was recently shown to be Ackermann-complete. However, many open questions remain about which features of the CRN model drive this complexity. We study a restricted class of CRNs with *void rules*, reactions that only decrease species counts. We further examine this regime in the motivated model of step CRNs, which allow additional species to be introduced in discrete stages. With and without steps, we characterize the complexity of the reachability problem for CRNs with void rules. We show that, without steps, reachability remains polynomial-time solvable for bimolecular systems but becomes NP-complete for larger reactions. Conversely, with just a single step, reachability becomes NP-complete even for bimolecular systems. Our results provide a nearly complete classification of void-rule reachability problems into tractable and intractable cases, with only a single exception.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Problems, reductions and completeness

Keywords and phrases CRN, Chemical Reaction Network, Reachability, Void Reactions

Digital Object Identifier 10.4230/LIPIcs.DNA.31.3

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

1 Introduction

Background. In molecular programming, Chemical Reaction Networks [5, 6] have become a staple model for abstracting molecular interactions. The model consists of a set of chemical species (formal symbols) as well as a set of reactions that dictate how these species interact. As an example, the reaction $A + B \to C + D$ describes chemical species A and B reacting to form species C and D. While chemical kinetics are commonly modeled continuously using ordinary differential equations, this approximation breaks down for systems with relatively small volumes where species are present in very low amounts. Such systems are better modeled as discrete Chemical Reaction Networks (which we will hereby be referring to simply as CRNs), where the system state consists of non-negative integer counts of each species and

Table 1 Summary of reachability results. The notation $(k, k-1)^+$ means one or more void rules of the form $(k, k-1)$ for $k \geq 2$. The notation $(k \geq 3, g \leq k-2)$ means void rules with at least three reactants that consume at least two species. The notation $\rightarrow$ NPC signifies that the step CRN result follows directly from the basic CRN case.

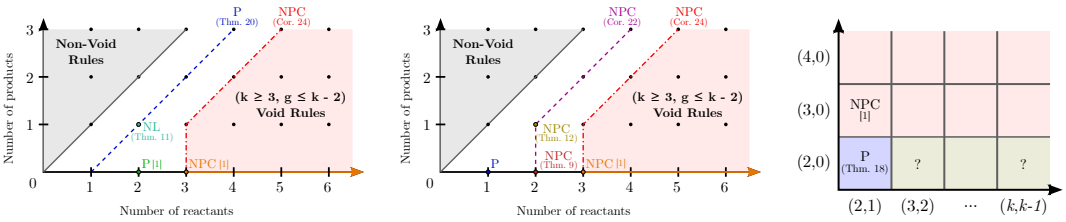
Reachability Results				
	Basic CRNs (1-step)		**2-Step CRNs**	
Void Rules	**Complexity**	**Ref.**	**Complexity**	**Ref.**
$(2,1)$	NL	Thm. 11	NPC	Thm. 12
$(k, k-1)^+$	$O(\lvert\Lambda\rvert^2\lvert\Gamma\rvert)$	Thm. 21	NPC	Cor. 22
$(2,0)$	$O(\lvert\Lambda\rvert^2\lvert\Gamma\rvert \log(\lvert\Lambda\rvert))$	[1]	NPC	Thm. 9
$(2,0)$, $(2,1)$	$O(\lvert\Lambda\rvert^2\lvert\Gamma\rvert \log(\lvert\Lambda\rvert))$	Thm. 18	NPC	Thm. 9
$(k \geq 3, g \leq k-2)$	NPC	Cor. 24	$\rightarrow$ NPC	-

state transitions occur stochastically as a continuous time Markov process [13]. It turns out that this model of chemistry has deep connections to several other well-studied mathematical objects. In fact, CRNs are equivalent [9] to Vector Addition Systems (VASs) [16] and Petri-Nets [23] which were independently introduced to represent and analyze concurrent and distributed processes. This underlying object also appears in the form of commutative semigroups when only fully reversible reactions are considered [17, 22], and as Population Protocols [4] when reactions must have exactly two input and two output elements.

Perhaps the most fundamental problem within these models is that of *reachability* which asks: given an initial configuration and target configuration for a particular system, can the target configuration be reached from the initial configuration by following some sequence of legal transitions [20, 21, 25]? Although interest in this problem dates back to the 1970s and 1980s, it was only recently resolved with a flurry of new results over the past few years [10, 11, 12, 18, 19], which conclude that reachability is Ackermann-complete for these models. Over the course of this 40+ year-long quest to determine the complexity of reachability, scientists also began to discover the centrality of this problem. Several other important problems from seemingly unrelated areas have been reduced to reachability, from system liveness to language emptiness problems and more [14, 26], emphasizing the significance of succinctly classifying reachability.

Despite the closure of this problem for general CRNs, and in fact due to how complicated these systems can be, there is a natural motivation to explore reachability for more restricted systems. Many papers have emerged that investigate reachability in restricted versions of the model [7, 8, 15, 26, 31], but one of the most elementary restrictions is that of [1] and [2]. There, the authors consider deletion-only systems (called *void rules*) where reactions only ever consume species and reduce the size of the system. The interaction rules for these systems can be thought to convert some chemical reactants into inert waste species, or cause system agents to "go offline" and become inactive. Similar work has also investigated reachability in a slightly different version of size-reducing chemical reaction networks whose stoichiometric matrices are totally unimodular [28, 29]. This limited class of systems permits tractable and intractable problems, placing it along an interesting complexity boundary.

Related Work. In [1], the authors studied reachability for this restricted class of CRNs that use (deletion-only) *void rules* (e.g., $A + B \rightarrow A$ or $A + B \rightarrow \emptyset$). Under this restriction, they prove NP-completeness for reachability using void rules of size (3,0) (3 reactants, no



(a) Void Rules in CRNs (1-step). **(b)** Void Rules with 2-step CRNs. **(c)** Mixing Void Rules.

Figure 1 Visual representations showing how the results from Table 1 fit together. (a) A plot depicting the complete characterization of reachability complexity for basic CRNs with uniform-type void rules. (b) A plot depicting the complete characterization of uniform-type step results with only a single additional step. (c) Void rule systems with mixed-type rules in basic CRNs.

products) and provide a polynomial time algorithm for reachability using void rules of size (2,0) (2 reactants, no products) for unary inputs. In this paper, we continue this line of work by considering reachability in CRNs with void rules of varying numbers of reactants and products. Since void rules arguably constitute the simplest type of reaction, one may wonder what computationally interesting behavior can be achieved with them – and this exact idea is investigated in [2].

The authors of [2] and [3] combine void rules with an experimentally motivated model extension called *step* CRNs. Ideally, experimental scientists use the CRN model to design molecular systems and predict the molecular interactions and possible end-states of those systems. However, contrary to the CRN model in which all the molecules are ready to interact from the start, many experimental designs rely on a progressive addition of molecules that allow some interactions to occur first before moving forward with the experiment. Furthermore, the ability of reactions to add molecules that interact with the rest of the system is precisely what makes experimental applications of theoretical concepts difficult to realize with minimal error at a practical scale [27, 32]. Thus, the step CRN model was introduced which incorporates a sequence of discrete steps, where, at each step, new species are added to the existing CRN and react until no further reactions can occur. This augmentation more closely reflects a laboratory setting in which chemicals are incrementally added in a test tube and left to interact. The work of [2] and [3] shows that, surprisingly, extremely simple void rules are capable of simulating threshold formulas and threshold circuits when steps are added. In this paper, we show that adding even a single step drastically changes the computational complexity of the reachability problem, moving from P (and even NL) to NP-complete in many cases.

Our Contributions and Paper Organization. In this paper we present several results for the various types of deletion-only interaction rules. We show that bimolecular deletion-only systems are solvable in polynomial time, with or without catalysts for basic (*single-step*) CRNs, while they are NP-complete even with one additional step. On the other hand, we show that larger void rules (with $k \geq 3$ reactants and $g \leq k-2$ products) are NP-complete even for basic CRNs unless all but one reactant is a catalyst, in which case we show the problem is in P. These results are outlined in Table 1, and we visualize the complexity landscape in Figures 1a, 1b, and 1c. When taken together, these results yield the following theorems, which provide a nearly complete characterization of reachability with void rules in CRNs with and without steps.

▶ **Theorem.** *Reachability for basic CRNs uniformly using size* $(k \geq 3, g \leq k-2)$ *void rules is NP-complete, and is in P when uniformly using any other size void rule.*

▶ **Theorem.** *Reachability for 2-step CRNs with only void rules that use exactly one reactant is in P, and is NP-complete otherwise.*

▶ **Theorem.** *Reachability for basic CRNs that use a combination of void rule types is in P for combinations of (2,0) + (2,1) rules as well as* $(k, k-1)^+$ *rules, and is NP-complete for any combination that uses* $(k \geq 3, g \leq k-2)$ *rules.*

We begin the paper by defining void rules, Step Chemical Reaction Networks, and the reachability problem in Section 2. Sections 3-6 establish the complexity of reachability based on the size of rules used by a given system: Section 3 shows membership in NP for all deletion-only systems. Section 4 considers bimolecular rule sets that are either all catalytic or non-catalytic, and shows membership in P in either case. It also shows that, in contrast, the problem becomes NP-complete with the inclusion of a second step. Section 5 expands this to bimolecular systems with mixed catalytic and non-catalytic rules, and Section 6 considers larger size rules which are polynomially solvable if all but a single reactant serve as catalysts, and NP-complete otherwise.

2 Preliminaries

2.1 Chemical Reaction Networks

Basics. Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|\Lambda|}\}$ denote some ordered alphabet of *species*. A configuration $\vec{C} \in \mathbb{N}^\Lambda$ is a length-$|\Lambda|$ vector of non-negative integers where $\vec{C}[i]$ denotes the number of copies of species λ_i. For a species $\lambda_i \in \Lambda$, we denote the configuration consisting of a single copy of λ_i and no other species as $\vec{\lambda}_i$. A *rule* or *reaction* is represented as an ordered pair $\gamma = (\vec{R}, \vec{P}) \in \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda$. $\vec{R}$ contains the minimum counts of each *reactant* species necessary for reaction γ to occur, where reactant species are either *consumed* by the rule in some count or leveraged as *catalysts* (not consumed); in some cases a combination of the two. The *product* vector $\vec{P}$ has the count of each species *produced* by the *application* of rule γ, effectively replacing vector $\vec{R}$. The species corresponding to the non-zero elements of $\vec{R}$ and $\vec{P}$ are termed *reactants* and *products* of γ, respectively.

The *application* vector of γ is $\vec{P} - \vec{R}$, which shows the net change in species counts after applying rule γ once. For a configuration $\vec{C}$ and rule γ, we say γ is applicable to $\vec{C}$ if $\vec{C}[i] \geq \vec{R}[i]$ for all $i \in \Lambda$, and we define the *application* of γ to $\vec{C}$ as the configuration $\vec{C}' = \vec{C} + \vec{P} - \vec{R}$. For a set of rules Γ, a configuration $\vec{C}$, and rule $\gamma \in \Gamma$ applicable to $\vec{C}$ that produces $\vec{C}' = \vec{C} + \vec{P} - \vec{R}$, we say $\vec{C} \to_\Gamma^1 \vec{C}'$, a relation denoting that $\vec{C}$ can transition to $\vec{C}'$ by way of a single rule application from Γ. We further use the notation $\vec{C} \to_\Gamma^* \vec{C}'$ to signify the transitive closure of $\to_\Gamma^1$ and say $\vec{C}'$ is *reachable* from $\vec{C}$ under Γ, i.e., $\vec{C}'$ can be reached by applying a sequence of applicable rules from Γ to initial configuration $\vec{C}$. Here, we use the following notation to depict a rule $(\vec{R}, \vec{P})$: $\vec{R}[1]\lambda_1 + \cdots + \vec{R}[|\Lambda|]\lambda_{|\Lambda|} \to \vec{P}[1]\lambda_1 + \cdots + \vec{P}[|\Lambda|]\lambda_{|\Lambda|}$. For example, a rule turning two copies of species H and one copy of species O into one copy of species W would be written as $2H + O \to W$.

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

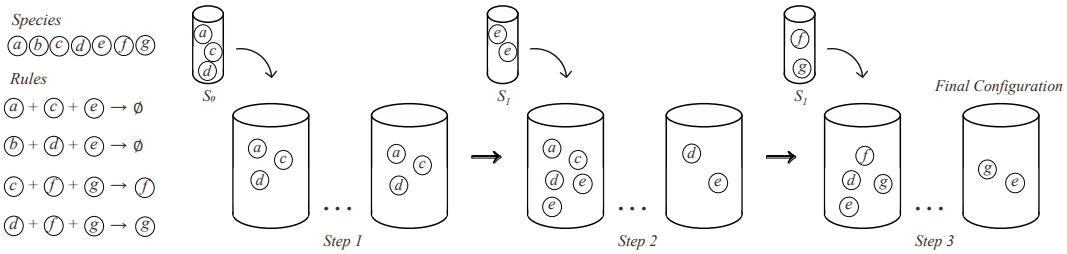


Figure 2 An example step CRN system. The test tubes show the species added at each step and the system with those elements added. The CRN species and void rule-set are shown on the left.

A configuration is called *terminal* with respect to a CRN (Λ, Γ) if no rule γ can be applied to it. An initial configuration $\vec{A}$ and CRN (Λ, Γ) is said to be *bounded* if a terminal configuration is guaranteed to be reached within some finite number of rule applications starting from $\vec{A}$. We denote the set of reachable configurations of a CRN as $REACH_{\vec{A},\Lambda,\Gamma}$. We define the subset of reachable configurations that are terminal as $TERM_{\vec{A},\Lambda,\Gamma}$.

2.2 Void Rules

A *void* rule is any rule that does not create any new copies of any species types (only deletes). Thus, a void rule 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 rules and rule size are as follows.

▶ **Definition 2** (Void rules). *A rule* $(\vec{R}, \vec{P})$ *is a* void *rule if* $\vec{P} - \vec{R}$ *has no positive entries and at least one negative entry. There are two classes of void rules, catalytic and true void. In catalytic void rules, one or more reactants remain and one or more reactant is deleted after the rule is applied. In true void rules, there are no products remaining.*

▶ **Definition 3.** *The* size/volume *of a configuration* $\vec{C}$ *is* $\texttt{volume}(\vec{C}) = \sum \vec{C}[i]$. *When* $\vec{C}$ *is an initial configuration of a CRN, we refer to* $\Sigma_{\vec{C}} = \texttt{volume}(\vec{C})$.

▶ **Definition 4** (size-(i,j) rules). *A rule* $(\vec{R}, \vec{P})$ *is said to be a size-*(i,j) *rule if* $(i,j) = (\texttt{volume}(\vec{R}), \texttt{volume}(\vec{P}))$. *A reaction is trimolecular if* $i = 3$, *bimolecular if* $i = 2$, *and unimolecular if* $i = 1$.

2.3 Step CRNs

A step CRN is an augmentation of a basic CRN in which a sequence of additional copies of some system species are added after a terminal configuration is reached. Formally, a step CRN of k steps is an ordered pair $((\Lambda, \Gamma), (\vec{S}_0, \vec{S}_1, \vec{S}_2, \ldots \vec{S}_{k-1}))$, where the first element of the pair 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* $\vec{C}_i$ for a step CRN as a valid configuration $\vec{C}$ over (Λ, Γ) along with an integer $i \in \{0, \ldots, k-1\}$ that denotes the configuration's step. We denote the initial volume of step CRN as $\Sigma_{\vec{S}_0} = \texttt{volume}(\vec{S}_0)$ and total volume as $\Sigma_T = \sum_{i=0}^{k-1} \texttt{volume}(\Sigma_{\vec{S}_i})$. Figure 2 illustrates a simple step CRN system.

Given a step CRN, we define the set of reachable configurations after each sequential step. To start off, let REACH_1 be the set of reachable configurations of (Λ, Γ) with initial configuration $\vec{S}_0$, which we refer to as the set of configurations reachable *after step 1*. Let

TERM_1 be the subset of configurations in REACH_1 that are terminal. Note that after a single step we have a normal CRN, i.e., 1-step CRNs are just normal CRNs with initial configuration $\vec{S}_0$. For the second step, we consider any configuration in TERM_1 combined with $\vec{S}_1$ as a possible starting configuration and define REACH_2 to be the union of all reachable configurations from each possible starting configuration attained by adding $\vec{S}_1$ to a configuration in TERM_1. We then define TERM_2 as the subset of configurations in REACH_2 that are terminal. Similarly, define REACH_i to be the union of all reachable sets attained by using initial configuration $\vec{S}_{i-1}$ plus any element of TERM_{i-1}, and let TERM_i denote the subset of these configurations that are terminal. The set of reachable configurations for a k-step CRN is the set REACH_k, and the set of terminal configurations is TERM_k. A classical CRN can be represented as a step CRN with $k = 1$ steps and an initial configuration of $\vec{A} = \vec{S}_0$.

Note that our definitions assume only the terminal configurations of a given step are passed on to seed the subsequent step. This makes sense if we assume we are dealing with *bounded* systems, as this represents simply waiting long enough for all configurations to reach a terminal state before proceeding to the next step. In this paper, we only consider bounded void rule systems; we leave more general definitions to be discussed in future work.

2.4 Reachability

The computational problem studied in this paper is *reachability*. Informally, reachability asks if a given initial configuration $\vec{A}$ can be turned into a target configuration $\vec{B}$ by applying a sequence of rules from the given CRN $\mathcal{C}$. The precise problem statement is as follows.

▶ **Definition 5** (Reachability Problem). *Given an initial configuration $\vec{A}$, a destination (target) configuration $\vec{B}$, and a step CRN $\mathcal{C}_\mathcal{S} = ((\Lambda, \Gamma), (\vec{S}_0 = \vec{A}, \vec{S}_1, \vec{S}_2, \ldots \vec{S}_{k-1}))$, determine if $\vec{B} \in REACH_k$, i.e., is configuration $\vec{B}$ reachable for the given step CRN. In the case of basic CRNs, this simplifies to: given configurations $\vec{A}$ and $\vec{B}$, and basic CRN $\mathcal{C} = (\Lambda, \Gamma)$, determine if $\vec{B} \in REACH_{\vec{A},\Lambda,\Gamma}$.*

3 Membership in NP for void rule systems

We initiate our study of deletion-only systems by observing that reachability stays within the class NP with only void rules, even with step-CRNs. This is straightforward to see in the case of polynomial bounded volume (unary encoded species counts) as each rule reduces the system volume by at least 1. But in the case of binary encoded species counts, the argument is more subtle as such deletion sequences could be exponentially long. To deal with this issue, we use the following *rearrangement* lemma that states that any order of void rule applications can be rearranged such that all applications of a given rule occur in a contiguous sequence. Given this lemma, any sequence of void rule applications can be rearranged into a sequence that can be encoded and verified in polynomial time.

▶ **Lemma 6** (Rearrangement Lemma). *For any sequence of applicable void rules A, there exists a sequence B that is a permutation of A such that all applications of a given rule type occur contiguously.*

Proof. Consider a sequence of applicable void rules A that is not contiguous. Suppose rule x occurs at positions i and j in A, $i < j - 1$, and there is at least one non-x rule in between them. Construct a new sequence A' by shifting the x rule at position i up to position $j - 1$, and shifting all rules in between down one position. This new sequence must be applicable

as the only rule that moved to a higher index in the sequence is of type x, and we know that x is still applicable at position $j-1$ since x is known to be applicable at position j. As this swapping preserves the applicability of the sequence while reducing the number of non-contiguous blocks of one rule type in the sequence, we can repeat this process of swapping rule positions until the sequence is contiguous. ◀

This lemma implies the existence of a polynomial-time verifiable certificate for "yes" instances of the reachability problem for step-CRNs, giving us membership in NP.

▶ **Theorem 7.** *The reachability problem for step-CRNs with void rules is in NP.*

Proof. As a certificate, we utilize a contiguous sequence of applicable rules for each step of the CRN, which must exist by Lemma 6. This sequence, while potentially exponential in length, can be encoded with a sequence of rule types accompanied by a count on the number of applications of each rule type. The result of such a sequence can be computed in polynomial time and therefore can serve as a certificate for the reachability problem. ◀

4 Bimolecular Rules of Uniform-type: With or Without Catalysts

In this section, we focus on bimolecular systems with either all size-$(2,0)$ rules (non-catalytic) or $(2,1)$ rules (catalytic). Recently, size-$(2,0)$ rule reachability in a single step was proven to be polynomial [1]. For size-$(2,1)$ 1-step systems, we present polynomial-time algorithms for reachability. In contrast, we show that in either scenario the problem becomes NP-complete with the addition of a second step. Later in Section 5, we consider the scenario of CRNs that use both $(2,0)$ and $(2,1)$ rules together.

4.1 Bimolecular Void Rules Without a Catalyst: $(2,0)$

In [1], the authors proved that reachability in a CRN system with only size-$(2,0)$ rules is in P by reducing from the perfect b-matching problem, which is a generalization of matching. This takes the form of a traditional matching when all b-values are 1, and an uncapacitated b-matching occurs when all edge capacities are assigned $u(e) = \infty$.

▶ **Theorem 8** ([1]). *Reachability for basic CRNs with binary encoded species with only rules of size (2, 0) is solvable in* $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$ *time.*

We now look at the problem with only one additional step, and show that it becomes NP-complete by reducing from the graph 3-colorability (3-COL) problem. Given an instance $\langle G \rangle$, where $G = (V, E)$ is an undirected graph, 3-COL asks if each vertex of G can be assigned one of three colors such that no adjacent vertices share the same color. An instance of 3-COL $\langle G \rangle$ can be converted into a (2,0) 2-step CRN $\mathcal{C}_\mathcal{S}$ as follows.

Species. For each vertex $v \in V$, we create the species v, v_R, v_G, and v_B. v_C represents an assignment of color $C \in \{R, G, B\}$ to vertex v; the v species will be used to represent assigning only one color to vertex v through specific reactions. We also create the species X to verify that a corresponding color assignment of G in $\mathcal{C}_\mathcal{S}$ has no adjacent vertices that share a color.

Steps and Rules. In step one (or $\overrightarrow{S}_0$), for each $v \in V$, we add two copies of v and one copy of v_R, v_G, and v_B. We also create the *assignment* rule $v + v_C \to \emptyset$ for each $C \in \{R, G, B\}$. Two of the three assignment rules created for v are applied to its respective species copies, consuming all v copies and two of the three copies of v_C. The remaining v_C copy then corresponds to assigning vertex v the color C. Additionally, for each edge $(i, j) \in E$ and $C \in \{R, G, B\}$, we create the *edge* rule $i_C + j_C \to \emptyset$. If the remaining copy of both i_C and j_C share a color C, they will be deleted by one of the edge rules.

In the second step ($\overrightarrow{S}_1$), we introduce $|V|$ copies of the species X. We also construct the *verification* rule $v_C + X \to \emptyset$ for each $v \in V$ and $C \in \{R, G, B\}$. All existing copies of v_C will be consumed by a X species. Thus, any remaining copies of X in the terminal configuration indicates that some v_C copies were deleted by an edge rule.

▶ **Theorem 9.** *Reachability for 2-step CRNs with only rules of size-*$(2, 0)$ *is NP-complete, even for unary encoded species counts.*

Proof. We reduce from the graph 3-colorability problem. Given an instance of 3-COL $\langle G \rangle$, we convert G into a 2-step $(2, 0)$ CRN $\mathcal{C}_\mathcal{S}$, following the construction outlined above, and set $\overrightarrow{A} = \overrightarrow{S}_0$ and $\overrightarrow{B}$ to the *empty configuration* $\overrightarrow{0}$.

Forward Direction. Assume there exists a color assignment in G where no adjacent vertices share a color. By the construction of $\mathcal{C}_\mathcal{S}$, a sequence of assignment rules can be applied in $\overrightarrow{A}$ that results in a configuration of one copy of v_C for each vertex that matches the color assignment in G. Denote this new configuration $\overrightarrow{S}'_0$. Since no adjacent vertices share a color in G, no edge rule will be applied in $\overrightarrow{S}'_0$, keeping the count of the v_C copies to $|V|$. $\mathcal{C}_\mathcal{S}$ then transitions to $\overrightarrow{S_1}$, introducing the $|V|$ X copies. Since $|V|$ V_C copies were preserved, all v_C and X copies are deleted by verification reactions to reach the final configuration $\overrightarrow{0} = \overrightarrow{B}$.

Reverse Direction. Assume there exists a sequence of applicable rules in $\mathcal{C}_\mathcal{S}$ that reaches $\overrightarrow{B}$ from $\overrightarrow{A}$. First, removing all v species can only be accomplished by applying a sequence of assignment rules. The resulting configuration $\overrightarrow{S}'_0$ is $|V|$ v_C copies. Assume no edge rule can be applied in $\overrightarrow{S}'_0$. By the construction of $\mathcal{C}_\mathcal{S}$, this implies that the matching color assignment in G also does not have adjacent vertices sharing colors. We then add $|V|$ X copies at the second step, resulting in the deletion of all v_C and X copies in the system by the verification rules, resulting in a final terminal configuration of $\overrightarrow{0}$. If an edge rule was applied in $\overrightarrow{S}'_0$, at least two v_C copies were removed, causing the final count of X to be greater than 0. Note that applying an edge rule *before* an assignment rule also guarantees $\overrightarrow{0}$ cannot be reached, as either 1) the assignment rules for the affected v_C copies are then applied, removing those copies and consequentially preventing some X copies from being deleted, or 2) more edge rules are applied on the affected v_C copies, which prevents all v copies from being deleted. Therefore, the only way for $\mathcal{C}_\mathcal{S}$ to reach $\overrightarrow{0} = \overrightarrow{B}$ is for a color assignment to exist on all vertices in G where no adjacent vertices share a color.

Theorem 7 shows reachability with void step CRN systems to be in NP. ◀

4.2 Bimolecular Void Rules with Catalyst: $(2, 1)$

In this section, we first show that reachability for size-$(2, 1)$ void rules resides in the class NL.

▶ **Lemma 10.** *Let the* implication *graph G of a CRN (Λ, Γ) with size-$(2, 1)$ void rules be the graph where each node is a species λ_i and each reaction $\lambda_i + \lambda_j \to \lambda_j$ implies a directed edge from λ_i to λ_j. A configuration $\overrightarrow{B}$ is reachable from $\overrightarrow{A}$ if and only if for each species λ_i there exists a path to a node λ_r that holds one of the following properties:*

1. $\overrightarrow{A}[r] = \overrightarrow{B}[r] > 0$,
2. $\lambda_r + \lambda_j \to \lambda_j \in \Gamma$ *where* $\overrightarrow{B}[j] \geq 1$, *or*
3. $\overrightarrow{B}[r] \geq 1$ *and* $\lambda_r + \lambda_r \to \lambda_r \in \Gamma$

Proof. We will refer to a node which satisfies one of these conditions as a *root node*. A path from species λ_i to a root node λ_r means that we can delete enough copies of λ_i to reach the target configuration. We will prove this recursively, inducing over the length of the path from λ_i to λ_r, to create a reaction sequence through this process with our base case being the end of the sequence.

For our base case, any node $\lambda_i = \lambda_r$ can reach the target amount if it satisfies a condition in the Lemma statement. In Case 1, the number of species in the starting configuration is already the target amount so the claim is trivially true. In Case 2, we may use copies of the species λ_j to delete λ_i using the leftover species in the target configuration. In Case 3, the species may delete itself to reach the target amount.

For our inductive case, assume that there exists a reachable configuration such that any species λ_k with a shortest path of length $\leq l$ to a node λ_r can be reduced to the target amount $\overrightarrow{B}[k]$. For a species λ_i with a shortest path of length $l+1$, there exists an edge to a species λ_k with length l. We can use the reaction $\lambda_i + \lambda_k \to \lambda_k$ to decrease λ_i to $\overrightarrow{B}[i]$ copies at the start of the current sequence. It remains to prove that removing these copies does not affect anything later in the sequence. If the closest node λ_r falls under case 1 or 3 then removing λ_i does not affect it. If the closest node is case 2 and $\lambda_i = \lambda_j$, the species λ_i is the one used to remove λ_r; thus, the condition $\overrightarrow{B}[j] \geq 1$ means that we leave at least a single copy in the configuration that can be used to delete λ_r.

If a node does not have a path to a root node, then it either does not have any outgoing edges, or all of its outgoing edges are part of some cycle where all nodes along each cycle have a target count of 0. As a result, if the node does not fall into case 1, there is no way of reducing the respective species to its target count without leaving some other species unsatisfied. ◀

▶ **Theorem 11.** *Reachability for basic CRNs with size (2, 1) void rules is in NL.*

Proof. We will show that we can decide whether a node has no path to a root in log space; thus reachability is in coNL = NL. We non-deterministically check a species λ_i, then for every node λ_j reachable from λ_i in the implication graph, we check if λ_j satisfies any of the conditions in Lemma 10. If we do not find such a node λ_j, then we reject.

If any node does not have a path to a root node, then some branch of this algorithm will reject. Checking each path can be done in NL as this is a directed graph. Checking if a node is a root node can be done in log space as it only involves edge queries and queries to the target configuration. ◀

We now show that adding an extra step turns the problem NP-complete, as with $(2, 0)$ CRN systems. Here, we reduce from the classic 3SAT problem. Given an instance of 3SAT $\langle \Phi \rangle$, we construct a $(2, 1)$ 2-step CRN $\mathcal{C}_\mathcal{S}$ as follows.

Species. For each variable x_i, we create a pair of species T_i and F_i, which represents assigning x_i a value of true or false, respectively. Additionally, for each clause c_j, we create the species C_j. The presence of a copy of C_j in a configuration of $\mathcal{C}_\mathcal{S}$ indicates c_j has yet to be satisfied by one of its assigned variables. Finally, we create the species X for "clean-up" procedures.

Steps and Rules. In the first step ($\overrightarrow{S}_0$), for each variable x_i, we introduce one copy of T_i and F_i. We also create a pair of *assignment* rules, one to represent assigning true to x_i ($T_i + F_i \to T_i$) and one for assigning false ($T_i + F_i \to F_i$). One of two assignment reactions will be applied to the copies of T_i and F_i; the non-deleted copy represents assigning x_i the corresponding boolean value.

In the second step ($\overrightarrow{S}_1$), we add a single copy of a clause species C_j for each clause c_j and a single copy of X. Additionally, given a clause c_j and a variable of the clause $x_k \in (x_a, x_b, x_c)$, 3 separate *verification* rules created of the form $C_j + T_k \to T_k$ (for non-negated variables) or $C_j + F_k \to F_k$ (for negated variables). If T_k/F_k is still present in $\mathcal{C}_\mathcal{S}$, then C_j will be consumed by that species. Finally, for each variable x_i, we create the *cleaning* rules $X + T_i \to X$ and $X + F_i \to X$ to consume all present copies of T_i and F_i. Any C_j species remaining in $\mathcal{C}_\mathcal{S}$ after the application of the cleaning rules indicates that it could not be deleted by a verification rule.

▶ **Theorem 12.** *Reachability for 2-step CRNs with only rules of size (2, 1) is NP-complete, even for unary encoded species counts.*

Proof. We reduce from 3SAT. Given an instance of 3SAT $\langle\Phi\rangle$, we convert Φ into a 2-step $(2,1)$ CRN $\mathcal{C}_\mathcal{S}$ via the reduction from above. Let $\overrightarrow{A} = \overrightarrow{S}_0$ and $\overrightarrow{B}$ be a single copy of X ($\vec{X}$).

Forward Direction. Assume there exists an assignment of variables that satisfies Φ to true. A sequence of assignment reactions can be then performed in $\overrightarrow{S}_0$ that results in a configuration with only one T_i/F_i copy for each variable that matches the variable assignment. In the second step, by the construction of $\mathcal{C}_\mathcal{S}$, since the assignment satisfies Φ, each introduced copy of C_j can be deleted with a verification reaction. Finally, the added X copy deletes all remaining literal species. The final configuration of $\mathcal{C}_\mathcal{S}$ is then $\vec{X}$.

Reverse Direction. Assume there exists a sequence of applicable rules in $\mathcal{C}_\mathcal{S}$ that reaches $\overrightarrow{B}$ from $\overrightarrow{A}$. First, a sequence of assignment reactions can be performed in $\overrightarrow{S}_0$ that consumes one of the T_i and F_i copy for each variable. We then add one copy of each C_j species and one copy of X in the second step. Assume all C_j copies can be consumed by a verification reaction. By the construction of $\mathcal{C}_\mathcal{S}$, this implies that there exists an assignment of variables in Φ that evaluates the formula to true. The X copy can delete the remaining X_i/F_i copies with cleanup rules to reach the final configuration $\vec{X}$. If a C_j copy could not be removed, this implies that the variable assignment couldn't satisfy the corresponding clause. Therefore, the only way for $\mathcal{C}_\mathcal{S}$ to reach $\vec{X} = \overrightarrow{B}$ is for an assignment of variables in Φ to exist that satisfies all clauses of the formula.

Theorem 7 shows reachability with void step CRN systems to be in NP. ◀

5 Bimolecular Rules of Mixed Type: (2,0) and (2,1)

In this section, we show that the reachability problem for general single-step bimolecular void rules systems that include a mix of non-catalytic $(2,0)$ rules and catalytic $(2,1)$ rules is in P. We show this via a reduction to the perfect b-matching problem in an undirected graph.

▶ **Definition 13** (Perfect b-matching Problem)**.** *Given a graph $G = (V, E)$, $u : e \in E \to \mathbb{N} \cup \{\infty\}$ to be edge capacities, and $b : v \to \mathbb{N}$ to be the number of matchings a vertex can take, does there exists an assignment to the edges $f : e \to \mathbb{N}$ such that $f(e) \leq u(e)$ and $\sum_{e \in \delta(v)} f(e) = b(v)$ for all $v \in V$?*

Unlike reducing from reachability to perfect b-matching with only $(2,0)$ rules in [1], the inclusion of catalyst rules requires a substantially more involved reduction. We therefore begin with a brief overview of our reduction and then describe each step in greater detail. Finally, we follow with a proof of correctness and a runtime analysis for the entire result.

Overview

Our reduction transforms an instance of CRN reachability $\langle \mathcal{C}, \overrightarrow{A}, \overrightarrow{B} \rangle$ into an instance of the perfect b-matching problem. The key idea is to identify which species involved in catalytic $(2,1)$ rules can be fully deleted using just catalytic rules and which must be further deleted using non-catalytic $(2,0)$ rules. If this distinction were known in advance, it would be straightforward to construct a corresponding graph and solve reachability via a matching instance.

To infer this structure, we first construct a directed graph T, which we refer to as the *catalytic deletion graph*, where each vertex corresponds to a species λ_i and each directed edge (λ_i, λ_j) represents a $(2,1)$ rule $\lambda_i + \lambda_j \to \lambda_j$ that catalytically deletes λ_i using λ_j. We then compute the strongly connected components (SCC's) of T and build a condensation graph H whose nodes each represent a component of T. The structure of H allows us to identify which catalytic species *must* be completely deleted via a non-catalytic $(2,0)$ rule.

Using this information, we construct an undirected graph G whose nodes effectively represent species that must be deleted via $(2,0)$ or $(2,1)$ rules and whose edges represent those corresponding rules. We then formulate a perfect b-matching instance on G where a perfect matching exists exactly when there exists a valid sequence of $(2,0)$ and $(2,1)$ rules that complements the catalytic deletions to reach the target configuration $\overrightarrow{B}$.

Creating Catalytic Deletion Graph T

Given a CRN $\mathcal{C} = (\Lambda, \Gamma)$, we construct the directed graph $T = (V, E)$ as follows. For each catalyst void rule $\lambda_i + \lambda_j \to \lambda_j \in \Gamma$, if $\overrightarrow{A}[i], \overrightarrow{A}[j] > 0$, we create vertices λ_i and λ_j and the directed edge (λ_i, λ_j).

The intuition of T is that an edge from λ_i to λ_j represents the species λ_i being deleted by the catalyst species λ_j. It then follows that a species whose respective vertex in T has an out-degree of 0 can only be deleted by a $(2,0)$ rule. We label these vertices as *mandatory vertices*. We also consider cycles in T in which each vertex (species) only has an out-going edge to another vertex in the cycle. If the count of all represented species in the cycle in $\overrightarrow{B}$ is zero, then regardless of the application of rules corresponding to the edges of the cycle, there is guaranteed to be at least one remaining species left that can only be completely removed by a $(2,0)$ rule. We label these cycles as *mandatory cycles*.

▶ **Definition 14** (Mandatory Vertices). *A vertex in T with an out-degree of 0.*

▶ **Definition 15** (Mandatory Cycles). *A cycle in T in which each vertex 1) only has an out-going edge to another vertex in the cycle, and 2) has a corresponding species count of 0 in $\overrightarrow{B}$.*

Creating SCC Condensation Graph H

Given a directed graph $T = (V, E)$, we construct the directed graph $H = (V', E')$ as follows. First, we run Tarjan's Strongly Connected Component Algorithm on T, which returns a partition of T's vertices of strongly connected components $C = \{c_1, c_2, \cdots, c_n\}$ [30]. For each component $c_i \in C$, we create the vertex c_i. For each directed edge from c_i to another component c_j, we create the directed edge (c_i, c_j).

▶ **Observation 16.** *A vertex in H represents a mandatory vertex if it is not a condensed component of T and it has an out-degree of 0.*

▶ **Observation 17.** *A vertex in H represents a mandatory cycle if it is a condensed component of T in which all corresponding species have final counts of 0, and it has an out-degree of 0.*

Creating b-matching Instance Graph G

Given a CRN $\mathcal{C} = (\Lambda, \Gamma)$, configurations $\overrightarrow{A}$ and $\overrightarrow{B}$, and directed graphs $T = (V, E)$ and $H = (V', E')$, we create an instance of the perfect b-matching problem with the graph $G = (V'', E'')$ as follows. Let the *difference configuration* $\overrightarrow{D} = \overrightarrow{A} - \overrightarrow{B}$.

Creating V'' and $b(\cdot)$. For each species $\lambda_i \in \Lambda$, if $\overrightarrow{D}[i] > 0$, we create the vertices λ_{i1} and λ_{i2} and set both $b(\lambda_{i1})$ and $b(\lambda_{i2})$ to $\overrightarrow{D}[i]$. These vertices represent the number of copies of λ_i that must be removed from $\overrightarrow{A}$ by the void rules. Additionally, if $\overrightarrow{B}[i] > 0$, we create the vertices $\overline{s_i}_1$ and $\overline{s_i}_2$ and set both $b(\overline{s_i}_1)$ and $b(\overline{s_i}_2)$ to $\overrightarrow{B}[i]$. These vertices exist just to "set aside" the final configuration for matchings, hence the bar labels.

We now consider species that can be deleted by catalyst void rules. For each catalytic rule $\lambda_i + \lambda_j \to \lambda_j \in \Gamma$, if $\overrightarrow{A}[i], \overrightarrow{A}[j] > 0$ *and* its corresponding edge in T is not part of a cycle, we create the vertices λ_{i1}' and λ_{i2}', if not already created, and set both $b(v_{i,1}')$ and $b(v_{i,2}')$ to $\overrightarrow{D}[i]$. Additionally, given a vertex of H $c_i \in V'$, if c_i represents a condensed cycle $\{\lambda_1, \cdots, \lambda_n\}$, we create the vertices c_{i1}' and c_{i2}'. If the cycle is mandatory, we assign $b(c_{i1}')$ and $b(c_{i2}')$ the value $(\sum_{\lambda_i \in c_i} \overrightarrow{D}[i]) - 1$; else they are assigned $(\sum_{\lambda_i'' \in c_i} \overrightarrow{D}[i])$, where λ_i'' is a *non-mandatory* vertex of c_i. These vertices represent a choice to delete a species λ_i using a catalytic species.

Let the vertices with b-values from $\overrightarrow{D}$ be the sub-graph G_D, and the vertices with b-values from $\overrightarrow{B}$ be the sub-graph G_B.

Creating E'' and $u(\cdot)$. For each $(2,0)$ rule $\lambda_i + \lambda_j \to \emptyset \in \Gamma$, if the vertices for both species were created in G, we create the edges $(\lambda_{i1}, \lambda_{j1})$ and $(\lambda_{i2}, \lambda_{j2})$. Performing a matching on these edges corresponds to deleting λ_i and λ_j by a $(2,0)$ rule.

For each $(2,1)$ rule $\lambda_i + \lambda_j \to \lambda_j \in \Gamma$, if λ_{i1}' and λ_{i1}' were created in G and the rule is not part of a cycle in T, we create the edges $(\lambda_{i1}, \lambda_{i1}')$ and $(\lambda_{i2}, \lambda_{i2}')$. For each vertex of T that represents a condensed cycle $c_i = \{\lambda_1, \cdots, \lambda_n\}$, if the cycle is mandatory, we create the edges (λ_{k1}, c_{i1}') and (λ_{k2}, c_{i2}') for all $s_k \in c_i$. Otherwise, we only create (λ_{k1}, c_{i1}') and (λ_{k2}, c_{i2}') for the *non-mandatory* vertices of c_i. Matching the edges represents deleting a species λ_i with a catalyst rule.

We finally create the following edges: for all $\bar{s}_{i1}$ and $\bar{s}_{i2}$ vertices, create the edge $(\bar{s}_{i1}, \bar{s}_{i2})$, for all λ_{i1}' and λ_{i2}' vertices, create the edge $(\lambda_{i1}', \lambda_{i2}')$, and for all c_{i1}' and c_{i2}' vertices, create the edge (c_{i1}', c_{i2}'). Matching on these edges does not represent a rule application, but rather ensures a perfect b-matching can be performed on these vertices even if they were not perfectly matched by other (2,1) edges. For all edges $e \in E''$, assign $u(e) = \infty$.

Result

The overall effect is that G has a perfect b-matching exactly when configuration $\overrightarrow{B}$ is reachable from configuration $\overrightarrow{A}$ which yields our main theorem from this section.

▶ **Theorem 18.** *Reachability in void rule systems with (2,0) and (2,1) rules is solvable in $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$ time.*

Due to space, the full proof can be found in Appendix A.

6 Larger Void Rules

Our next results involve CRNs with reactions that require more than two reactants. If a system's rules have all but one reactant serving as catalysts (i.e., $(k, k-1)$ void rules), then reachability remains polynomial-time solvable. In contrast, reachability for systems with any other form of void rule (with 3 or more reactants) becomes NP-complete.

6.1 Mostly-Catalytic Large Void Rules of Mixed-type: $(\mathbf{k}, \mathbf{k-1})^+$

We provide a polynomial-time dynamic programming algorithm to decide reachability for CRNs that use mostly-catalytic void rules of the form $(k, k-1)$. We further argue that reachability remains in P, even for CRNs that use a combination of various size $(k, k-1)$. For simplicity, we refer to void rules of sizes $(k_1, k_1-1), \ldots, (k_b, k_b-1)$, where all $k_i \in \mathbb{N}$, as $(k, k-1)^+$, meaning there is one or more rule of this type.

▶ **Lemma 19.** *Reachability for basic CRNs with only void rules of size $(k, k-1)$ requires at most $|\Lambda|$ distinct rules.*

Proof. For simplicity, let $n = |\Lambda|$. Assume there exists a sequence of reactions $a_1 r_1, \ldots, a_{n+1} r_{n+1}$ for a CRN $\mathcal{C}$ with set of species Λ and set of rules Γ that takes some initial configuration $\overrightarrow{A}$ to configuration $\overrightarrow{B}$, where $a_1, \ldots, a_{n+1}$ are positive integers (denoting how many times to apply each rule) and $r_1, \ldots, r_{n+1}$ are rules in Γ. There must then exist some species s that gets consumed by 2 rules r_i and r_j, where $i < j$. Let s^i, s^x and s^f denote the initial, intermediate and final counts of species s, where r_i reduces s from s^i to s^x and r_j reduces s from s^x to s^f. Since $s^i > s^x$, any rule r_l, where $l > i$, that uses s with count s^x can also use s with count s^i. Thus, rule r_i is not needed. ◀

▶ **Theorem 20.** *Reachability for basic CRNs with only void rules of size $(k, k-1)$ is solvable in $O(|\Lambda|^2|\Gamma|)$.*

Proof. Let Γ denote the set of rules. We can use a dynamic programming approach to solve the problem. Construct an $|\Lambda| \times (|\Lambda|+1)$ table $D(s, j)$ of boolean entries, where each row represents a different species. Reduce the count of each species to $max(k, s^f)$, where s^f represents the final count of species s, if there exists a rule that can do so. Starting from the first column $j = 0$, place a 1 if the respective species is already in its final count. Then, for each entry $D(s, j)$, place a 1 if $D(s, j-1)$ is a 1 or if there exists a reaction $\gamma \in \Gamma$ that reduces s to its final count, where all the reactants of γ have either reached their final counts or will not prevent the reaction from occurring once they do. If column $|\Lambda|+1$ contains all 1's, then reachability is possible. Otherwise, it is not.

By Lemma 19, a solution to the problem requires at most $|\Lambda|$ unique reactions, where each reaction directly reduces each species from its initial counts to its final counts. Thus, finding a solution to the problem takes at most $|\Lambda|$ steps since we are implicitly selecting at least one reaction per column. This results in $|\Lambda|+1$ columns, with the first column representing the initial configuration.

Since every rule is a catalytic void rule, there must exist an ordering of reactions such that some reactions can occur first without impeding other species from getting reduced to their final counts. A bottom-up approach can be used to find this ordering, starting with the reactions that can be put off until later and working up to the reactions that must occur first. In table D, this ordering is implicitly represented between columns, with the reactions between the rightmost columns being the ones we do first. Any species with final count

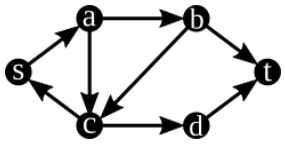


Figure 3 Directed graph used in the example $(3, 1)$ reduction in Section 6.2.

greater than k is reduced before the algorithm is run if there exists an applicable rule, as reducing this species count does not prevent any other rule from occurring. Filling D takes at most $O(|\Lambda|^2|\Gamma|)$ steps. Hence, reachability is solvable in polynomial time. ◀

▶ **Theorem 21.** *Reachability for basic CRNs with only void rules of size* $(k, k-1)^+$ *is solvable in* $O(|\Lambda|^2|\Gamma|)$.

Proof. This follows from Theorem 20. Since we are considering a rule set Γ where the size of each rule $\gamma \in \Gamma$ is $\leq k$, reducing each initial species count to $\max(k, s^f)$ guarantees that any r that needs s will be able to occur. The algorithm remains the same. ◀

▶ **Corollary 22.** *Reachability for 2-step CRNs with only void rules of size* $(k, k-1)^+$ *is NP-complete, even for unary encoded species counts.*

Proof. Follows from Theorem 12. ◀

6.2 Large Void Rules of Uniform-type: $(\mathbf{k \geq 3, g \leq k-2})$

Here we show that reachability for CRNs using any void rules with at least three reactants, and which remove at least two species, becomes NP-complete. To achieve this result, we show that reachability is NP-complete for CRNs using only $(3, 1)$ void rules via a reduction from the Hamiltonian path problem for directed graphs. Since it was previously shown that reachability is NP-complete for CRNs using only $(3, 0)$ rules [1], this implies Corollary 24.

We reduce from the Hamiltonian path problem in directed graphs. For simplicity, we reduce from the variation where each vertex has max in-degree and out-degree of two, which is still NP-complete [24]. Given an instance of the problem $\langle G, s, t\rangle$, we outline the species and the reactions for the reduction, and show correctness in the proof.

Species. We create a species for every path through a vertex (unless it starts at t or goes back to s). For example, looking at Figure 3, for vertex b, we create two species $S_{a,b,c}$ and $S_{a,b,t}$, where the notation denotes the vertex it came from, the vertex itself, and the vertex it goes to. With a maximum in/out degree of two, for each vertex, there are at most 4 paths through the vertex. In general, for each vertex $j \in V$, we create the species $S_{i,j,k}$ where $(i, j), (j, k) \in E$ and $k \neq s$, $i \neq t$. For s, t, we just have S_s and S_t. We also create a species P_i for each $i \in V$ where $i \neq s$. Thus, we have $n - 1$ of these species.

Reactions. There are two main types of reactions: those that pick which path through a vertex and those that walk the Hamiltonian path.

- Since there are at most 4 species per vertex, we will create rules that create a tournament to choose one of the species for each vertex. Assuming 3 species for ease of explanation, we create the rules $S_1 + S_2 + S_3 \to S_1$, $S_1 + S_2 + S_3 \to S_2$, and $S_1 + S_2 + S_3 \to S_3$. With only two species, we can create a dummy species that is not used as a catalyst. With 4 species, we augment the case of 3 species with another reaction that must occur with a dummy species (S_d) and the previous choice. For each choice S_i with $i \in \{1, 2, 3\}$, we create the rules $S_i + S_4 + S_d \to S_i$ and $S_i + S_4 + S_d \to S_4$.

- We create rules that walk the Hamiltonian path with valid connecting species and a "fuel" species (P_i) so that we can only visit a vertex one time. We create each of the rules $S_{i,j,k} + S_{j,k,l} + P_k \to S_{j,k,l}$ where $i, j, k, l \in V$ and $i \neq t, l \neq s$. This rule indicates a walk from j to k along a valid edge and removes the P_k token to mark the vertex as visited.

Example. For Figure 3, we give the full reduction as follows. The final configuration $\vec{S_t}$ (just a single copy of S_t) is only reachable if there is a Hamiltonian path.

- The main species are S_s, $S_{s,a,b}$, $S_{s,a,c}$, $S_{a,b,c}$, $S_{a,b,t}$, $S_{a,c,d}$, $S_{b,c,d}$, $S_{c,d,t}$, S_t, P_a, P_b, P_c, P_d, and P_t. We also create a single copy each of dummy species D_a, D_b, D_c.
- The tournament reactions for each vertex are (d has only one path)
  - a) $S_{s,a,b} + S_{s,a,c} + D_a \to S_{s,a,b}$ and $S_{s,a,b} + S_{s,a,c} + D_a \to S_{s,a,c}$,
  - b) $S_{a,b,c} + S_{a,b,t} + D_b \to S_{a,b,c}$ and $S_{a,b,c} + S_{a,b,t} + D_b \to S_{a,b,t}$,
  - c) $S_{a,c,d} + S_{b,c,d} + D_c \to S_{a,c,d}$ and $S_{a,c,d} + S_{b,c,d} + D_c \to S_{b,c,d}$.
- The walking reactions for each vertex are
  - a) $S_s + S_{s,a,b} + P_a \to S_{s,a,b}$ and $S_s + S_{s,a,c} + P_a \to S_{s,a,c}$,
  - b) $S_{s,a,b} + S_{a,b,c} + P_b \to S_{a,b,c}$ and $S_{s,a,b} + S_{a,b,t} + P_b \to S_{a,b,t}$,
  - c) $S_{s,a,c} + S_{a,c,d} + P_c \to S_{a,c,d}$ and $S_{a,b,c} + S_{b,c,d} + P_c \to S_{b,c,d}$,
  - d) $S_{a,c,d} + S_{c,d,t} + P_d \to S_{c,d,t}$ and $S_{b,c,d} + S_{c,d,t} + P_d \to S_{c,d,t}$,
  - t) $S_{a,b,t} + S_t + P_t \to S_t$ and $S_{c,d,t} + S_t + P_t \to S_t$.

▶ **Theorem 23.** *Reachability for CRNs with only void rules of size* $(3, 1)$ *is NP-complete, even for unary encoded species counts.*

Proof. We reduce from the directed Hamiltonian path problem. Given an instance $\langle G, s, t \rangle$, we convert this to an instance of the reachability problem, as outlined above, for CRN $\mathcal{C}$ and target configuration $\vec{S_t}$. We show that H is true iff the configuration $\vec{S_t}$ is reachable in $\mathcal{C}$.

Forward Direction. Assume there exists a Hamiltonian path in G from s to t. Then it is possible that the tournament for every vertex correctly produces the species that represents the Hamiltonian path through the vertex. If this does occur, all species have been removed from the system except $|V|$ S species for the path and $|V| - 1$ P species. Then, the walking reactions can occur successively by destroying the previous path vertex and the "fuel" species, which will only leave one copy of S_t.

Reverse Direction. Assume there exists a sequence of applicable rules in $\mathcal{C}_\mathcal{S}$ that reaches $\overrightarrow{B}$ from $\overrightarrow{A}$. The only way to remove an S species is through the tournament and the walking reactions. The tournament will always leave at least one S for each vertex, meaning the walking reactions must be used to delete the other $|V| - 1$. Along with this, the P vertices ensure that each vertex can only be visited once. Thus, the walk can not occur before the tournament and take multiple paths to reach a vertex. Thus, reaching a configuration with only S_t ensures that a walk through the graph occurred starting at S_s, ending at S_t, and that every vertex was visited.

All void CRN systems are in NP with the certificate being the sequence of rules to apply, and the number of times to apply them [1]. ◀

▶ **Corollary 24.** *Reachability for CRNs with only void rules of size* $(k \geq 3, g \leq k - 2)$ $(k, g \in \mathbb{N})$ *is NP-complete, even for unary encoded species counts.*

Proof. For $g \geq 1$, this follows from Theorem 23. For $g = 0$, this follows from the fact that reachability for $(3, 0)$ rules is NP-complete [1]. ◀

7 Primary Results

We restate the primary results formally with the corresponding proofs. Although not discussed, for completeness, we also include the following lemma.

▶ **Lemma 25.** *Reachability for step CRNs (including basic CRNs) uniformly using size* $(1, 0)$ *void rules is in P.*

Proof. Simply decrease each species to the desired count. Since each step must become terminal, all species in rules will be removed before the subsequent step. Thus, there must exist a step that adds counts greater than or equal to the target counts. Then treat that step as a basic CRN. If no such step exists, the target configuration is not reachable. ◀

The collection of results presented, as a whole, yields the following main theorems of this work that characterize void rules within CRNs and step CRNs.

▶ **Theorem 26.** *Reachability for basic CRNs uniformly using size* $(k \geq 3, g \leq k-2)$ *void rules is NP-complete, and is in P when uniformly using any other size void rule.*

Proof. This follows from [1], Theorems 11, 20, Corollary 24, and Lemma 25. ◀

▶ **Theorem 27.** *Reachability for 2-step CRNs with only void rules that use exactly one reactant is in P, and is NP-complete otherwise.*

Proof. This follows from [1], Theorems 9, 12, Corollaries 22, 24, and Lemma 25. ◀

▶ **Theorem 28.** *Reachability for basic CRNs that use a combination of void rule types is in P for combinations of (2,0) + (2,1) rules as well as* $(k, k-1)^+$ *rules, and is NP-complete for any combination that uses* $(k \geq 3, g \leq k-2)$ *rules.*

Proof. This follows from [1], Theorems 18, 21, and 24. ◀

8 Conclusion and Future Work

This paper presents a nearly complete classification of the computational complexity of reachability for CRNs and step CRNs that consist of deletion-only rules. We provide polynomial-time algorithms for most combinations of void rules in basic CRNs and show NP-completeness for rules of size greater than $(k, g \leq k-2)$ for $k \geq 3$. Additionally, we prove that with the addition of a single step, these problems become NP-complete. We include some natural open directions to explore:

- **Mixed-size Void Rule Systems.** What is the complexity of reachability when you consider void rules of size $(2, 0)$ and $(k, k-1)$ together? This combination of void rule types is the missing piece that would complete the picture for the entire complexity landscape of void rule reachability.
- **Staged CRNs.** The step CRN model augments the basic CRN model with steps that add species once reactions are completed. A generalization of this model could have multiple "stages", where CRNs are left to react and the results of these stages are combined. How do stages affect reachability?
- **Model Variants.** What is the complexity of reachability in deletion-only extensions of CRNs, petri-nets, and vector addition systems?

References

1 Robert M. Alaniz, Bin Fu, Timothy Gomez, Elise Grizzell, Andrew Rodriguez, Robert Schweller, and Tim Wylie. Reachability in restricted chemical reaction networks, 2022. doi:10.48550/arXiv.2211.12603.

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 2024)*, 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*, pages 253–268. Springer, 2024. doi:10.1007/978-3-031-63742-1_18.

4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, March 2006. doi:10.1007/s00446-005-0138-3.

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 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, January 1968. doi:10.1007/BF00251438.

7 Michael Blondin, Matthias Englert, Alain Finkel, Stefan Göller, Christoph Haase, Ranko Lazić, Pierre Mckenzie, and Patrick Totzke. The reachability problem for two-dimensional vector addition systems with states. *Journal of the ACM (JACM)*, 68(5):1–43, 2021. doi:10.1145/3464794.

8 Adam Case, Jack H Lutz, and Donald M Stull. Reachability problems for continuous chemical reaction networks. *Natural Computing*, 17(2):223–230, 2018. doi:10.1007/S11047-017-9641-2.

9 Matthew Cook, David Soloveichik, Erik Winfree, and Jehoshua Bruck. *Programmability of Chemical Reaction Networks*, pages 543–584. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-540-88869-7_27.

10 Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. *Journal of the ACM (JACM)*, 68(1):1–28, 2020. doi:10.1145/3422822.

11 Wojciech Czerwiński, Sławomir Lasota, and Łukasz Orlikowski. Improved Lower Bounds for Reachability in Vector Addition Systems. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 128:1–128:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.128.

12 Wojciech Czerwiński and Łukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd Annual Symposium on Foundations of Computer Science*, FOCS'21, pages 1229–1240. IEEE, 2021.

13 Daniel T Gillespie. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58(1):35–55, 2007.

14 Michel Henri Théodore Hack. *Decidability questions for Petri Nets.* PhD thesis, Massachusetts Institute of Technology, 1976.

15 John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979. doi:10.1016/0304-3975(79)90041-0.

16 Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969. doi:10.1016/S0022-0000(69)80011-5.

17 Ulla Koppenhagen and Ernst W Mayr. Optimal algorithms for the coverability, the subword, the containment, and the equivalence problems for commutative semigroups. *Information and Computation*, 158(2):98–124, 2000. doi:10.1006/INCO.1999.2812.

18 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd Annual Symposium on Foundations of Computer Science*, FOCS'21. IEEE, 2021.

19 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785796.

20 Richard J. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.

21 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 238–246, New York, NY, USA, 1981. Association for Computing Machinery. doi:10.1145/800076.802477.

22 Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.

23 Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik an der Universität Bonn, 1962.

24 Ján Plesník. The np-completeness of the hamiltonian cycle problem in planar diagraphs with degree bound two. *Information Processing Letters*, 8(4):199–201, 1979. doi:10.1016/0020-0190(79)90023-1.

25 George S Sacerdote and Richard L Tenney. The decidability of the reachability problem for vector addition systems (preliminary version). In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 61–76, 1977. doi:10.1145/800105.803396.

26 Sylvain Schmitz. The complexity of reachability in vector addition systems. *ACM SigLog News*, 2016. doi:10.1145/2893582.2893585.

27 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *natural computing*, 7(4):615–633, 2008. doi:10.1007/S11047-008-9067-Y.

28 Gergely Szlobodnyik and Gábor Szederkényi. Polynomial time reachability analysis in discrete state chemical reaction networks obeying conservation laws. *MATCH-Communications in Mathematical and in Computer Chemistry*, 89(1):175–196, 2023.

29 Gergely Szlobodnyik, Gábor Szederkényi, and Matthew D Johnston. Reachability analysis of subconservative discrete chemical reaction networks. *MATCH-Communications in Mathematical and in Computer Chemistry*, 81(3):705–736, 2019.

30 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

31 Chris Thachuk and Anne Condon. Space and energy efficient computation with dna strand displacement systems. In *International Workshop on DNA-Based Computers*, 2012.

32 Boya Wang, Chris Thachuk, Andrew D Ellington, Erik Winfree, and David Soloveichik. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences*, 115(52):E12182–E12191, 2018.

A Proof Details for (2,0) and (2,1) Mixed rules

Here we show the full details for the proof of correctness and runtime for the Lemmas and Theorems from Section 5.

A.1 Proof of Correctness

▶ **Lemma 29.** *A species represented by a mandatory vertex must use* $(2, 0)$ *void rules to be completely deleted, and mandatory cycles must have at least one species represented by a mandatory vertex to completely delete the cycle.*

Proof. We first consider the case of trying to delete a non-zero amount of a species λ_i represented by a mandatory vertex. We thus know that λ_i cannot be deleted by any catalytic rules. Thus, it can only be deleted with $(2,0)$ void rules, which means we must have as many $(2,0)$ matchings with λ_i as the number of deletions required to remove the species.

To show that completely deleting a mandatory cycle requires at least one species to be represented by a mandatory vertex, recall that the definition of a mandatory cycle is a cycle of $(2,1)$ void rules where the only catalytic rules that can delete each vertex is within the cycle. This means to completely delete the species in the cycle only using the rules of the cycle will always leave at least one species in the cycle with a non-zero amount of copies. Since we can always perform catalytic rules in the cycle such that every species in the cycle has a count of 1, then no matter what order the catalytic rules are applied in the cycle, there is always at least one species λ_i with a count of one. Thus, to completely delete all species of the cycle, a matching from a $(2,0)$ rule must be performed on λ_i. ◀

▶ **Lemma 30.** *A perfect b-matching in G implies we have a set of rules that, if applied, can delete $\overrightarrow{D}$ and only $\overrightarrow{D}$.*

Proof. We represent configurations $\overrightarrow{D}$ and $\overrightarrow{B}$ in disjoint subgraphs in G as G_D and G_B, respectively. If a perfect matching exists, then $\sum_{e\in\delta(v)} f(e) = b(v)$ for all $v \in V''$, which holds for all vertices in G_D. Since every edge represents a deletion either from a $(2,0)$ void rule or a $(2,1)$ catalytic void rule, this implies that if all rules are applied that are represented in $\delta(v)$, this would delete at least v because $\sum_{e\in\delta(v)} f(e) = b(v)$, and $b(v)$ is equal to the count of v in G_D. Since this is true for all $v \in V''$, we can delete the entirety of G_D and completely remove only D if we have a perfect matching. ◀

▶ **Lemma 31.** *A perfect b-matching in G contains (2,0) void rule matchings for all species represented by mandatory vertices in G_D.*

Proof. If a species λ_i is represented in G_D, there exists a non-zero count of λ_i that must be deleted to reach $\overrightarrow{B}$. By definition, a species represented by a mandatory vertex can only be completely deleted by a $(2,0)$ rule. Thus, if we have a perfect matching, we have matched all of the species represented by mandatory vertices through these $(2,0)$ void rule edges. ◀

▶ **Lemma 32.** *If there exists a perfect b-matching in G, then there exists a vertex in each mandatory cycle that is matched to a $(2,0)$ rule.*

Proof. A perfect matching contains the property that for all vertices, $\sum_{e\in\delta(v)} f(e) = b(v)$. Let c_j be a collection of vertices that compose a mandatory cycle. By the construction of G, for all $\lambda_i \in c_j$, there exists an edge from λ_{i1} to c_{j1}' or λ_{i2} to c_{j2}', where c_{jk}' is the node representing catalytic rules among the vertices in the cycle. The vertex c_{jk}' has b-value $b(c_{jk}') = (\sum_{\lambda_{ik}\in c_j} b(v)) - 1$, which means there is at most $(\sum_{\lambda_{ik}\in c_j} b(v)) - 1$ matchings with the vertices in the cycle. This implies at least one matching must occur without the catalytic rules represented by c_{jk}', and since the matching is perfect, this matching must exists. Furthermore, since this a mandatory cycle, no species in this cycle can be deleted through a catalytic rule not represented in the cycle. Thus, there is at least one $(2,0)$ rule matching with some vertex in this cycle when we have a perfect matching. ◀

▶ **Theorem 33.** *G has a perfect b-matching if and only if $\overrightarrow{B}$ is reachable from $\overrightarrow{A}$.*

Proof.

Forward Direction. Assume there exists a sequence of applicable rules in $\mathcal{C}$ that reaches $\overrightarrow{B}$ from $\overrightarrow{A}$. It thus follows that all species with non-zero counts in $\overrightarrow{D}$ can be completely deleted by following this sequence. Then, by the construction of G, a perfect b-matching can be performed in the graph. Recall that our graph G has two disjoint subgraphs G_B and G_D. In our construction, G_B will always form a perfect matching since we have two vertices with equal b-values attached by one edge of infinite capacity. Thus, these vertices can always match with each other, and we only need to show how to create a perfect matching in G_D. Every vertex in G_D must be perfectly matched to create a perfect b-matching, and we can think of matching as a deletion of the deleted species for $(2,1)$ void rules and deletion of both species in $(2,0)$ void rules. We then make the assignment $f((\lambda_{i1}, \lambda_{j1})) = f((\lambda_{i2}, \lambda_{j2}))$ on each edge the number of times we used that respective $(2,0)$ or $(2,1)$ rule, which perfectly matches all $v_i \in V''$. However, if our catalytic vertices λ_{i1}' are not perfectly matched through its respective $(\lambda_{i1}, \lambda_{i1}')$ edge, this means that we deleted λ_{i1} using more than the catalytic rule corresponding with λ_{i1}', which means we have excess λ_{i1}'. We thus use the edge $(\lambda_{i1}', \lambda_{i2}')$ so that any excess matchings can be assigned along that edge, and thus create a perfect b-matching in G.

Reverse Direction. Assume there exists a perfect b-matching in G. Then $\overrightarrow{B}$ is reachable from $\overrightarrow{A}$ if there exists a valid sequence of $(2,0)$ and $(2,1)$ void rules to get from $\overrightarrow{A}$ to $\overrightarrow{B}$. Recall that a perfect b-matching means that we have an assignment of edges such that $\sum_{e \in \delta(v)} f(e) = b(v)$ for every $v \in V''$. Through Lemma 30, this perfect matching implies we can delete G_D and only G_D. Now we must show that there exists a valid sequence of rules that can be applied to delete G_D. This is done through Lemmas 31 and 32, where we show that all mandatory vertices and mandatory cycles have $(2,0)$ void rules to match with. This means we can apply all $(2,1)$ void rules the correct number of times ($f(e)$ times) without worrying about deleting a catalytic species needed for a later catalytic rule. We can then execute all $(2,0)$ void rules their respective $f(e)$ times to successfully delete $\overrightarrow{D}$, and thus reach $\overrightarrow{B}$ from $\overrightarrow{A}$. ◀

A.2 Runtime Analysis

We first construct T, which creates $O(|\Lambda|)$ vertices and $O(|\Gamma|)$ edges since it creates a vertex for each unique species involved in a $(2,1)$ rule and an edge for each unique $(2,1)$ rule. Thus, this step takes $O(|V| + |E|)$ time, where $V = O(|\Lambda|)$ and $E = O(|\Gamma|)$. We then run Tarjan's Strongly Connected Component Algorithm on T to produce graph H, which has a runtime of $O(|V| + |E|)$ [30]. Finally, we create the graph G and functions $u(\cdot)$ and $b(\cdot)$. It takes $O(|\Lambda|)$ time to create G_D, and since we create at most 6 vertices per species when creating the vertex set for G, this takes $O(|\Lambda|)$ time. Additionally, since we create one edge for every valid void rule in our CRN and an edge for every species in our final configuration, we create $O(|\Lambda| + |\Gamma|)$ edges. We then assign a b-value $b(\cdot)$ for each vertex in G, and since assigning a b-value to a vertex requires a look-up in $\overrightarrow{D}$ that takes $O(1)$ time, we assign all the b-values in $O(|\Lambda|)$ time. Finally, we assign all edge capacities $u(\cdot)$ to infinity, which takes $O(1)$ time per edge with a runtime of $O(|\Lambda| + |\Gamma|)$. This results in an overall runtime of $O(|\Lambda| + |\Gamma|)$ for the construction of G. Finally, the runtime of the maximum b-matching algorithm is proven to be strictly polynomial with a runtime of $O(|V|^2 \log(|V|)(|E| + |V| \log(|V|)))$. This results in a total polynomial runtime for the algorithm of $O(|V|^2 \log(|V|)(|E| + |V| \log(|V|)))$.

▶ **Theorem 34.** *The reachability problem in CRNs with only* $(2,0)$ *and* $(2,1)$ *rules is solvable in* $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$ *time.*

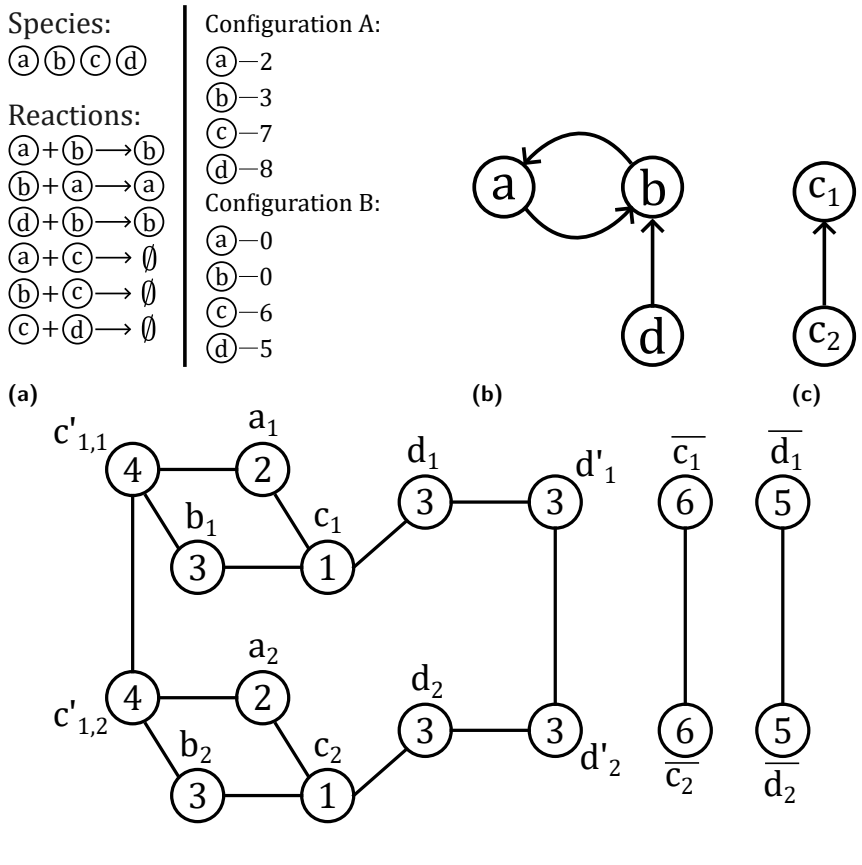


Figure 4 Transforming an instance of the $(2,0)$ and $(2,1)$ CRN reachability problem into an instance of the perfect b-matching problem, as described in Section 5. (a) An instance of $(2,0),(2,1)$ CRN reachability. (b) The corresponding catalytic deletion graph T. Note that species a and b form a mandatory cycle. (c) The corresponding SCC condensation graph H. c_1 represents the sole mandatory cycle in T, and c_2 only the vertex d. (d) The corresponding b-matching instance graph G.

Proof. Subsection 5 shows how to convert any instance of the $(2,0)$ and $(2,1)$ reachability problem into an instance of the perfect b-matching problem in $O(|\Lambda|^2 \log(|\Lambda|)(|\Gamma| + |\Lambda| \log(|\Lambda|)))$ time (Subsection A.2), with Subsection (A.1) proving its correctness. ◀