Reachability in Interactive Chemical Reaction Networks*

Alberto Avila-Jimenez, Bin Fu, Elise Grizzell, Robert Schweller, and Tim Wylie

University of Texas Rio Grande Valley

Abstract. This paper studies the effects of interactivity on molecular computation, specifically in the Step Chemical Reaction Networks model (Step CRNs), by adding the ability for a user to interact with the system by selecting which species to add at each step, or by having some control over which reactions execute. The two proposed variants are Interactive CRNs and Randomized Interactive CRNs. We show that in Interactive CRNs, even when restricted to *void* (deletion-only) rules of relatively small size, if a user can decide which species to add at each step based on the configuration, reachability is PSPACE-complete when bounded and EXPTIME-hard when unbounded. In Randomized Interactive CRNs, we prove that reachability with void rules is PSPACE-complete.

1 Introduction

Molecular computing is an important topic that has generated a lot of interest and research within the past few decades. This has led to a host of theoretical models based on different laboratory techniques such as DNA Tiles and Origami. A popular model based solely on the interaction of species is Chemical Reaction Networks [4], which is fundamental in describing distributed processes and is equivalent to both Vector Addition Systems [12] and Petri-nets [14].

In general, many molecular theoretical models, including Chemical Reaction Networks (CRNs), are not designed to allow for the addition of more inputs later. Instead, most work under the operating assumption of designing a computation for the model and letting it run until complete with no intermediate interaction. Obviously, this is limiting for many computational tasks and would require additional systems to run experiments under different parameters.

The Step Chemical Reaction Network (Step CRN) model was introduced to address this limitation by allowing staged addition of chemicals [2, 3]. It was shown that even with void rules, threshold circuits can be simulated. The model, however, still requires all added species for every step to be fixed as input, and thus does not allow flexible interaction with the computation.

In this paper, we investigate interaction with molecular computation in two generalizations of the Step CRN. First, we explore reachability when a user

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

is allowed to choose which sets of species to add at each step. These Interactive Chemical Reaction Networks, even with void rules, are extremely powerful. Then, we include randomization to the interaction process. The Randomized Interactive CRNs introduce randomized elements by allowing random selection of species additions, a significant step toward realistic biochemical computation scenarios. We demonstrate that reachability remains PSPACE-hard within this randomized context.

1.1 Previous Work

CRNs and Step CRNs. Chemical Reaction Networks (CRNs) are a well-studied field of theoretical chemistry. CRNs abstract chemical interactions through chemical *species* and a set of reaction *rules* that dictate how the species interact. Models that are similar to CRNs include Vector Addition Systems (VASs) [12] and Petri-nets [14]. The traditional CRN model studies an environment in which all chemicals that will be used in the system are added to a single container simultaneously. Step Chemical Reaction Networks [2] are a natural extension of CRNs, aligning more closely with actual laboratory procedures. Step CRNs with void rules have been used to simulate threshold circuits[2] and study reachability questions [9], proving NP-completeness for traditional Step CRNs.

Void Rules. Reachability with void rules in CRNs was first studied in [1]. Previous studies have included void rules as a part of their systems; it is even possible to program such rules in CRN++, but they were never studied exclusively. Void rules can also be considered a special case of reaction extinction [18].

Mixing Systems. Another generalization of CRNs related to the step model is I/O CRNs [8], where additional inputs can be added at timed intervals. Still, those inputs are read-only in the system (used exclusively as catalysts). Step CRNs generalize I/O CRNs as the inputs are not read-only and are rate-independent, unlike I/O CRNs. Staged systems have been explored in many self-assembly models [5–7, 13].

1.2 Our Contributions.

This paper introduces two laboratory-motivated generalizations of the Step CRN: the Interactive CRN model and the Randomized Interactive CRN model. We start in Section 2 by defining these two models and the reachability problem. In Section 3, we first show PSPACE membership of reachability in Interactive CRNs. Then, in Section 4, we present two different reductions from 2-Player Constraint Logic to prove hardness for the reachability problem. Later in Section 5, we show the hardness of reachability in the Randomized Interactive CRN by simulating Interactive Proofs. Table 1 gives an overview of the main results.

Interactive CRNs		
Void Rules	**Complexity**	**Ref.**
(10,8)	PSPACE-Complete	Thm.2
(13,10)	EXPTIME-hard	Thm.3
Randomized Interactive CRNs		
Void Rules	**Complexity**	**Ref.**
(2,0)	PSPACE-hard	Thm.4
(3,0)	PSPACE-hard	Thm.5

Table 1: Summary of results: reachability hardness in two different Step CRN generalization models using only void rules.

2 Preliminaries

2.1 Chemical Reaction Networks

Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|\Lambda|}\}$ denote some ordered alphabet of *species*. A configuration over Λ is a length-$|\Lambda|$ vector of non-negative integers that denotes the number of copies of each present species. A *rule* or *reaction* has two multisets, the first containing one or more *reactant* (species), used for creating the resulting *product* (species) contained in the second multiset. Each rule is represented as an ordered pair of configuration vectors $R = (\overrightarrow{R_r}, \overrightarrow{R_p})$. $\overrightarrow{R_r}$ contains the minimum counts of each reactant species necessary for reaction R 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 $\overrightarrow{R_p}$ has the count of each species *produced* by the *application* of rule R, effectively replacing vector $\overrightarrow{R_r}$. The species corresponding to the non-zero elements of $\overrightarrow{R_r}$ and $\overrightarrow{R_p}$ are termed *reactants* and *products* of R, respectively.

The *application* vector of R is $\overrightarrow{R_a} = \overrightarrow{R_p} - \overrightarrow{R_r}$, which shows the net change in species counts after applying rule R once. For a configuration $\overrightarrow{C}$ and rule R, we say R is applicable to $\overrightarrow{C}$ if the count of species i in configuration $\overrightarrow{C}$, $\overrightarrow{C}[i] \geq \overrightarrow{R_r}[i]$ for all $1 \leq i \leq |\Lambda|$, and we define the *application* of R to $\overrightarrow{C}$ as the configuration $\overrightarrow{C'} = \overrightarrow{C} + \overrightarrow{R_a}$. For a set of rules Γ, a configuration $\overrightarrow{C}$, and rule $R \in \Gamma$ applicable to $\overrightarrow{C}$ that produces $\overrightarrow{C} = \overrightarrow{C} + \overrightarrow{R_a}$, we say $\overrightarrow{C} \rightarrow^1_\Gamma \overrightarrow{C}'$, a relation denoting that $\overrightarrow{C}$ can transition to $\overrightarrow{C}'$ by way of a single rule application from Γ. We further use the notation $\overrightarrow{C} \rightarrow^*_\Gamma \overrightarrow{C'}$ to signify the transitive closure of $\rightarrow^1_\Gamma$ and say $\overrightarrow{C}'$ is *reachable* from $\overrightarrow{C}$ under Γ, i.e., $\overrightarrow{C'}$ can be reached by applying a sequence of applicable rules from Γ to the initial configuration $\overrightarrow{C}$. A configuration is *terminal* if no applicable rules exist. Here, we use the following notation to depict a rule $R = (\overrightarrow{R_r}, \overrightarrow{R_p})$: $\sum_{i=1}^{|\Lambda|} \overrightarrow{R_r}[i]s_i \rightarrow \sum_{i=1}^{|\Lambda|} \overrightarrow{R_p}[i]s_i$.

Using this notation, a rule turning two copies of species H and one copy of species O into one copy of species W would be written as $2H + O \rightarrow W$.

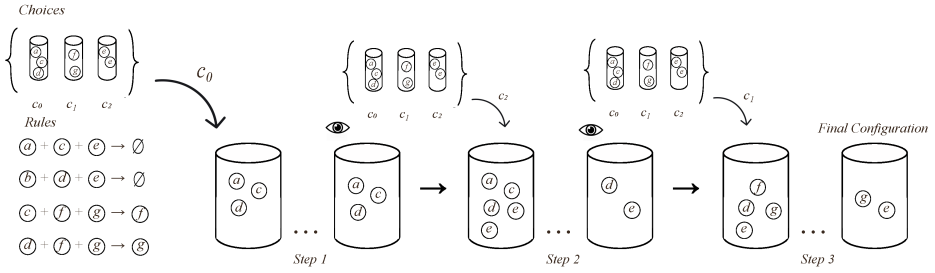


Fig. 1: An example Interactive CRN system. c_j is added once all the possible rules have been applied (i.e., the CRN has reached a terminal configuration).

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

Definition 2 (Void rules). *A rule $R = (\overrightarrow{R_r}, \overrightarrow{R_p})$ is a* void *rule if $\overrightarrow{R_a} = \overrightarrow{R_p} - \overrightarrow{R_r}$ has no positive entries and at least one negative entry. A void rule is a rule whose product multiset is a strict sub-multiset of the reactant. In catalytic void rules, such as $(2, 1)$ rules, one or more reactants remain, and one or more is deleted after the rule is applied.*

Definition 3. *The* size/volume *of a configuration vector $\overrightarrow{C}$ is* $\texttt{volume}(\overrightarrow{C}) = \sum \overrightarrow{C}[i]$.

Definition 4 (size-(i,j) rules). *A rule $R = (\overrightarrow{R_r}, \overrightarrow{R_p})$ is said to be a size-(i,j) rule if $(i,j) = (\texttt{volume}(\overrightarrow{R_r}), \texttt{volume}(\overrightarrow{R_p}))$.*

2.2 Interactive CRNs

An Interactive CRN is a generalization of a Step CRN. Step CRNs define a number of copies of each species to add at each step. This can be viewed as a single "choice". Interactive CRNs offer a fixed set of k choices $\{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c_{k-1}}\}$, where, like Step CRNs, each $\overrightarrow{c_j}$ is a sequence of length-$|\Lambda|$ vectors of non-negative integers denoting how many copies of each species type are present within that choice. In this model, whenever a reaction from Γ applies to the current configuration $\overrightarrow{A}$, the system executes that reaction in the usual fashion, producing $\overrightarrow{B} = \overrightarrow{A} - \overrightarrow{R_r} + \overrightarrow{R_p}$. When the configuration is terminal (no reaction can fire), the experimenter observes the current state of the system and selects one of the choices to add. The process of adding $\overrightarrow{c_j}$ and waiting for the CRN to reach a terminal configuration is denoted as a step. See Figure 1.

Definition 5 (Interactive CRN). *An Interactive CRN is a tuple $C_{\text{int}} = \big((\Lambda, \Gamma), \{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c}_{k-1}\}\big)$, where Λ is a finite species set, $\Gamma \subseteq \mathbb{N}^\Lambda \times \mathbb{N}^\Lambda$ is the reaction set, and $\{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c}_{k-1}\}$ is a finite set of vectors $\overrightarrow{c_j} \in \mathbb{N}^\Lambda$. The*

one-step transition $\xrightarrow{C_{\text{int}}}$ *on configurations* $\overrightarrow{A}, \overrightarrow{B} \in \mathbb{N}^{\Lambda}$ *is defined by* $\overrightarrow{A} \xrightarrow{C_{\text{int}}} \overrightarrow{B}$ *if there exists* $(\overrightarrow{R_r}, \overrightarrow{R_p}) \in \Gamma$ *with* $\overrightarrow{R_r} \leq \overrightarrow{A}$ *and* $\overrightarrow{B} = \overrightarrow{A} - \overrightarrow{R_r} + \overrightarrow{R_p}$, *or, when no reaction in* Γ *applies to* $\overrightarrow{A}$, *if there is some* $\overrightarrow{c_j} \in \{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c}_{k-1}\}$ *with* $\overrightarrow{B} = \overrightarrow{A} + \overrightarrow{c_j}$.

Note that a *step* is counted only when the terminal configuration is different from the initial. If a choice vector is added and subsequent the reactions yield the same configuration, it it is a *null step* and does not increase step count.

Given an Interactive CRN, we define all configurations that are terminal as $TERM$. A *strategy* is a polynomial time computable function $\sigma(\overrightarrow{X}) \in \{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c}_{k-1}\}, \forall \overrightarrow{X} \in TERM$ that computes which $\overrightarrow{c_j}$ to add given a terminal configuration $\overrightarrow{X}$. Let $REACH_1[\sigma]$ be the set of reachable configurations of C_{int} with initial configuration $\overrightarrow{A}$ at step 1 plus $\sigma(\overrightarrow{A})$, and let $TERM_1[\sigma]$ be the subset of reachable configurations that are terminal. Define $REACH_2[\sigma]$ to be the union of all reachable configurations from each possible starting configuration $\overrightarrow{C} \in TERM_1[\sigma]$ plus $\sigma(\overrightarrow{C})$. Let $TERM_2[\sigma]$ be the subset of the configurations in $REACH_2[\sigma]$ that are terminal. Similarly, define $REACH_i[\sigma]$ to be the union of all reachable sets attained by using some terminal configuration $\overrightarrow{X} \in TERM_{i-1}[\sigma]$ plus $\sigma(\overrightarrow{X})$, and let $TERM_i[\sigma]$ denote the subset of these configurations that are terminal.

Definition 6 (Interactive CRN Forced Reachability problem). *Let* m *be an integer. Given start configuration* $\overrightarrow{A}$, *target configuration* $\overrightarrow{B}$, *and* $C_{\text{int}} = \big((\Lambda, \Gamma), \{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c}_{k-1}\}\big)$, *determine whether there exists a strategy* σ *that ensures* $TERM_m[\sigma] = \{\overrightarrow{B}\}$. *In short, is there a strategy* σ *that picks which choice* c_j *to add at every step and guarantees* $\overrightarrow{B}$ *is the only terminal configuration reachable after following* σ *for* m *steps?*

3 PSPACE Membership of Interactive CRNs

Theorem 1. *Let* $C_{\text{int}} = \big((\Lambda, \Gamma), \{\overrightarrow{c_0}, \overrightarrow{c_1} \ldots, \overrightarrow{c}_{k-1}\}\big)$ *be an Interactive CRN. Then the Forced Interactive reachability problem for* $TERM_m[\sigma]$ *is in space* $(m + v_0 + \max_i\{|c_i|\})^{O(1)}$, *where initial volume is* v_0 *and* $|c_i|$ *is the sum of elements in the vector* c_i.

Proof. We describe a polynomial-space algorithm based on a depth-first search traversal of the configuration tree. Consider a start configuration $\overrightarrow{A}$, a target $\overrightarrow{B}$ and some integer m. From $\overrightarrow{A}$, the system evolves by applying reactions or by adding some $\overrightarrow{c_j}$ chosen by σ. Each step adds a bounded number of species, The volume is bounded by $v_0 + m \times \max_i\{|c_i|\}$ where v_0 is the initial volume, and c_i is the sum of elements in $\overrightarrow{c_j}$. Since we only use void rules, all sequences of rule applications can happen in polynomial time. We construct a configuration tree with polynomial depth bounded by the step count m and the volume v. Each node represents a configuration, and every node has a child node for a

$\overrightarrow{c_j}$ addition or a rule application. We now run depth-first search on this tree, we store only the current configuration using polynomial space. The algorithm checks if $\overrightarrow{B}$ is the only terminal configuration after exactly m steps. Since depth and volume are polynomially bounded, we have a polynomial space algorithm.

4 Hardness for Interactive CRNs

In this section, we prove hardness for the Interactive CRN Forced Reachability problem using void rules by reductions from bounded and unbounded two-player constraint logic (2CL). We interpret the IA-CRN dynamics as a game between the experimenter and the CRN. Bounded and unbounded 2CL using AND, OR, CHOICE, FANOUT, and VARIABLE vertices were shown to be PSPACE-complete (bounded) and EXPTIME-hard (unbounded) by Hearn in 2006 [10].

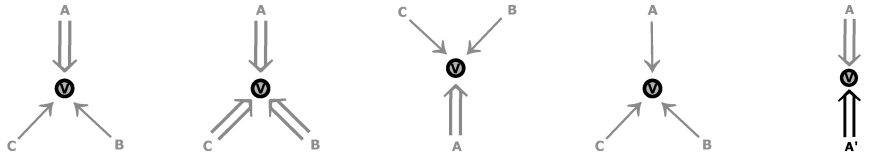


Fig. 2: Bounded 2 player constraint logic vertex variations. Left to right: (a) AND vertex (b) OR vertex (c) FANOUT vertex (d) CHOICE vertex (e) VARIABLE vertex

4.1 Bounded Systems

Bounded 2CL An instance of Bounded 2CL, defined by [10], consists of a directed graph $G(V, E)$ with edge weights $\in \{1, 2\}$, vertices of degree 3, and a minimum inflow constraint of 2 for each vertex. Graphs are constructed using the vertex gadgets detailed below and shown in Figure 2. Players alternate flipping edges on their respective vertices while maintaining vertex inflow constraints. The only vertices where both players have edges that they may flip are the variable vertices. Every edge can be flipped at most once. The decision problem is: given a target edge $e \in E$, does a sequence of legal edge flips exist allowing player 1 to flip edge e?

Vertex gadgets. Bounded 2CL uses five degree-3 vertex types (Fig. 2). *AND*: Figure 2a. One weight-2 edge and two weight-1 edges; the weight-2 edge can flip outward only when both weight-1 edges point inward. *OR*: Figure 2b. Three weight-2 edges; any inward-pointing edge satisfies the inflow constraint, allowing

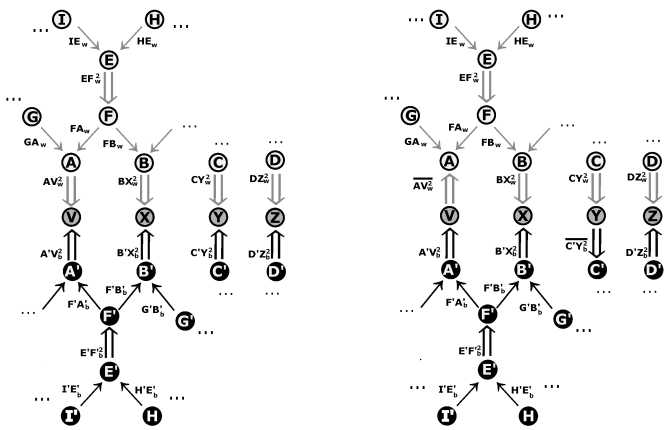
either of the other two to flip. *FANOUT*: Figure 2c. One weight-2 "input" edge and two weight-1 "output" edges; with the input edge directed inward, both outputs may point outward simultaneously. *CHOICE*: Figure 2d. Three weight-1 edges; at most one may point outward at any time. *VARIABLE*: Figure 2e. The only vertex where both players influence the inflow of the. It consists of two weight-2 edges, one controlled by each player; whichever edge flips first locks the other in place, since flipping both would violate the inflow constraint.

We transform an instance of 2CL into an instance of Interactive CRN Forced reachability. An edge reversal, or flip, for both players is represented by a single rule where the experimenter (white) adds the species representing the desired flip. If it is a valid move, the rule flips that edge, and the system (black) nondeterministically selects a valid response. The experimenter's objective is to reach a configuration where the species corresponding to the winning edge indicates it has been reversed.

Theorem 2. *The Interactive CRN Forced Reachability Problem is PSPACE-complete using only void rules of at most size (10,8) with a polynomially bounded number of steps.*

Proof. Given a constraint graph $G = (V, E)$ with edges partitioned into black (B) and white (W), we encode the instance into an IA-CRN as follows. Each edge is represented as a species named by its endpoints: an edge from vertex A to vertex B becomes species AB, while its flipped version is $\overline{AB}$. Weight-two edges include a superscript and each edge species has a subscript indicating which player it belongs to (see Figure 3a). At every step, the white player adds exactly one choice $\overrightarrow{c_j} \in \{\overrightarrow{c_0}, \ldots, \overrightarrow{c_{k-1}}\}$, where $k = |W|$. Each $\overrightarrow{c_j}$ consists of exactly one flipped white edge (the edge the white player attempts to reverse), one copy of every flipped black edge, and $|B| - 1$ copies of a cleanup species DEL_b. The idea is that when a choice is added, the flipped white edge species will consume the unflipped version. Similarly, one out of all of the flipped black edge species will consume the unflipped version. The rest of flipped black edge species that were added will be cleaned up by the cleanup species DEL_b after the reaction representing the edge reversals executes.

We construct the rules Γ as follows. Consider a white edge $e_w \in W$ directed into vertex A. To determine if e_w can be flipped (reversed), we first identify all other edges surrounding A. The different combinations of flipped and unflipped versions of these surrounding edges form local configurations $\overrightarrow{C_{e_w}}$. However, we only retain local configurations that satisfy the constraint when e_w is flipped. For example, in Figure 2a, if edge AV^2 is flipping away from vertex V, and the other edges directed a V are CV and BV, then the local configurations are $\{(CV, BV), (\overline{CV}, BV), (CV, \overline{BV}), (\overline{CV}, \overline{BV})\}$. Since these edges are weight-1, only the configuration $\overrightarrow{C_{AV}} = \{(CV, BV)\}$ would be retained, since this is the only scenario where reversing BA mantains the constraint. Similarly, for each black edge $e_b \in B$, we construct local configurations $\overrightarrow{C_{e_b}}$, retaining only valid configurations that satisfy vertex constraints. Every local configuration $\overrightarrow{w} \in \overrightarrow{C_{e_w}}$ and $\overrightarrow{b} \in \overrightarrow{C_{e_b}}$ will also include the species of the edge that is being reversed and



(a) (b)

Fig. 3: (a) A fragment of a constraint graph and the species used to represent its initial state. (b) The same graph after one rule application, where player 1 selects vertex V and player 2 selects vertex Y. Edge EF_w^2 cannot be reversed unless white player selects vertices V and X first. Ellipsis (. . .) imply omitted portions of the full graph.

its reversed version as a prefix. Finally, we form rules by combining each valid local configuration $\overrightarrow{w}$ for white edges with each valid configuration $\overrightarrow{b}$ for black edges, provided the moves remain valid when the white player's flip occurs first. Each rule γ_i thus has reactants $\overrightarrow{w} + \overrightarrow{b}$. The unflipped edge species are consumed, so the rest of the reactants are used as catalysts and remain as the product. An example reaction is shown in Figure 4.

We show our rule construction explicitly using the VARIABLE and AND vertices. The OR, CHOICE, and FANOUT vertices follow similarly by modifying inflow conditions. Full details for these vertices are straightforward and can be found in the appendix. Vertex labels and edge directions correspond to those in Fig. 3.

VARIABLE vertex. See VARIABLE vertex V in Figure 3a . Initially, AV_w^2 and $A'V_b^2$ are present. For the rule flip AV_w^2, $\overrightarrow{w}$ will have AV_w^2, $\overline{AV_w^2}$ and $A'V_b^2$. These represent the edge that will be flipped, the flipped version of that edge, and the inflow check to vertex V. The second half $\overrightarrow{b}$ will include the same for another variable edge. In a valid flip, the only element consumed will be the species representing the original edges that were flipped by white and black. The complete $(6, 4)$ void reaction for a turn on a VARIABLE vertex is shown in Figure 4

AND vertex. Let F be an AND vertex with single edges FA_w, FB_W and double edge EF_w^2. To reverse EF_w^2, we have local configurations $\overrightarrow{C_{EF_w^2}} = \{(\overline{FA_w}, \overline{FB_w})\}$

$$\overbrace{AV_w^2 + \overline{AV_w^2} + A'V_b^2}^{\vec{w}} + \underbrace{C'Y_b^2 + \overline{C'Y_b^2} + CY_w^2}_{\vec{b}} \rightarrow \overline{AV_w^2} + A'V_b^2 + \overline{C'Y_b^2} + CY_w^2$$

Fig. 4: $(6, 4)$ void rule to play a turn on a VARIABLE vertex. Player 1 selects vertex V and reverses edge AV_w^2. Player 2 selects Vertex Y and reverses edge $C'Y_b^2$. The graph after this turn is shown in Figure 3b.

that make up $\overrightarrow{w}$, the first half of the rule. The second half $\overrightarrow{b}$ consists of a valid move for Player 2. For simplicity in this example, we show a mirrored reversal of the player 1 move in a different vertex F'. Two copies of any reversed edge of the black player are needed in the reactantsto avoid the black player's reversed edge species added in $\overrightarrow{c_j}$ from triggering this reaction in an invalid configuration. It is important to note that after the VARIABLE vertex selections, black moves are irrelevant. We still account for them to preserve turn order. The completed $(10, 8)$ void reaction for a turn on an AND vertex is as follows: Player 1 reverses edge EF_w^2. Player 2 reverses edge $E'F_b'^2$,

$$EF_w^2 + \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + E'F_b'^2 + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'} \rightarrow \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'}$$

After a reaction representing a turn executes, the reversed black edge species added in $\overrightarrow{c_j}$ will still be in the system. The species DEL_b then cleans up those leftover black edge species, leaving the system with only the species representing the current state of the graph. These rules are:

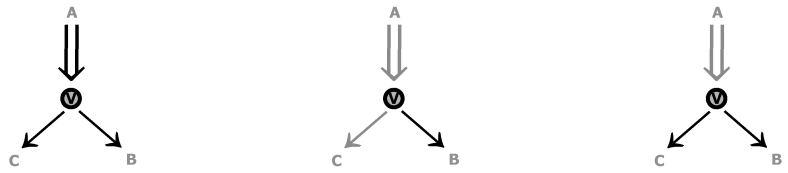
$$(1)\ 2\overline{A'V_b^2} + DEL_b \rightarrow \overline{A'V_b^2}$$
$$(2)\ A'V_b^2 + \overline{A'V_b^2} + DEL_b \rightarrow A'V_b^2$$
$$(3)\ BX_w^2 + \overline{BX_w^2} \rightarrow BX_w^2, (4)\ 2\overline{BX_w^2} \rightarrow \overline{BX_w^2}$$

First, a (3,1) void rule to clean up unused black player flipped edges. This rule consumes one of two reversed black edge species for the case when the current state of the graph has $A'V_b^2$ in reversed state. Second is a (2,1) void rule for the case where the current state of the graph has $A'V_b^2$ in its initial state. Rules (3) and (4) will delete a white move and return the system to its original state before the choice. (3) takes the case for an invalid white flip. (4) is for the case of trying to flip an already flipped edge.

Using this construction, the IA-CRN reaches a terminal configuration containing the species corresponding to the reversed winning edge if and only if the white player has a winning strategy in the constraint logic game. This establishes the Interactive CRN Forced Reachability problem is PSPACE-hard. By Theorem 1, it follows that the problem is PSPACE-complete.

4.2 Unbounded systems

In unbounded two-player constraint logic, edges may be flipped and unflipped indefinitely. The graph representation remains similar to the bounded case, but



(a) (b) (c)

Fig. 5: Unbounded 2-player constraint logic vertex variations. Left to right: (a) Black player AND vertex (b) Mixed AND vertex case 1 (c) Mixed AND vertex case 2.

each black player edge is tracked using two species to represent its current direction. To allow reversibility, both flipped and unflipped versions of black edges are included in each choice vector. As a result, rule size increases, with valid rules deleting both versions of a black edge, reaching a maximum size of (17, 6).

Unbounded 2CL An instance of unbounded 2-player Constraint Logic[10] is similar to that in Section 4.1, consisting of a directed graph with edge weights in $1, 2$ and degree-3 vertices enforcing minimum inflow constraints. Unlike the bounded case, edges may be flipped multiple times. While the reduction is less direct, it reuses the vertex gadgets from Section 4.1, with modifications, particularly to the AND vertex, which now accommodates mixed white and black edges, as shown in Figure 5.

Theorem 3. *The Interactive CRN reachability problem is EXPTIME-hard, even when restricted to void rules of at most size (13,10) and allowing an unbounded number of steps.*

Proof. Given a constraint graph $G = (V, E)$ with E partitioned into B and W, we encode the instance into the IA-CRN the same as Theorem 2. After every step, the white player adds exactly one choice $\overrightarrow{c_j} \in \{\overrightarrow{c_0}, \ldots, \overrightarrow{c_{k-1}}\}$, where $k = 2|W|$. one for every e_w and one for its reverse$\overrightarrow{e_w}$. Each $\overrightarrow{c_j}$ consists of exactly one white edge, one copy of every flipped black edge, one copy of every reversed black edge, and $|B| - 1$ copies of a cleanup species DEL_b. The rules are constructed the similar to Theorem 2. They keep a similar reactant structure of $\overrightarrow{w} + \overrightarrow{b}$. Note that now, they can have mixed edges, so $\overrightarrow{w}$ wont be exclusively white edges and the same for $\overrightarrow{b}$. For every black edge used as a validation catalyst in $\overrightarrow{w}$ and $\overrightarrow{b}$ will need 2 copies in its reactants. Having two of every species for every black edge ensures the rule matches the current state (one copy already in the configuration and one from the choice), and prevents the extra black edges in $\overrightarrow{c_j}$ from serving as catalysts. The rule keeps the flipped orientations and the existing DEL_b cleanup reactions then remove these duplicates, restoring a single valid edge species.

We explicitly show the unbounded construction with two representative gadgets. First, we revisit the AND vertex used in Theorem 2 and show how a

weight-2 white edge can be flipped forward and back under our new "two-copy" convention. Then, we present the mixed-edge AND vertex in Figure 5c. This case produces the largest rule in the reduction. All other mixed-edge variants from Figure 5 can be found in the appendix.

Black AND Vertex. The rules for a black AND vertex follow almost the same as the AND vertex in Theorem 2. The only change is that the current state of the black edge that is being flipped requires 2 copies in the reactants. In the following rule, white is flipping edge EF_w^2 and black player is flipping $E'F_b'^2$. Note that now, two copies of $E'F_b'^2$ are required to verify the current state of the edge. See vertex F and F' in Figure 3 for reference.

$$EF_w^2 + \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + 2E'F_b'^2 + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'} \rightarrow \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'}$$

Rules to reverse an edge again are symmetrical. To reverse edge $\overline{EF_w^2}$ and $\overline{E'F_b'^2}$ again, we must now ensure the valid local configurations of vertices E and E' are present. These now include edges $IE_w, HE_w, I'E_b', H'E_b'$. The $(11, 8)$ reaction is the following:

$$\overline{EF_w^2} + EF_w^2 + IE_w + HE_w + 2\overline{E'F_b'^2} + E'F_b'^2 + 2I'E_b' + 2H'E_b' \rightarrow EF_w^2 + IE_w + HE_w + E'F_b'^2 + 2I'E_b' + 2H'E_b'$$

Mixed AND Vertex. The largest rule in the reduction is triggered in the scenario where both players attempt to flip edges from an AND vertex with two black single-weight edges and one white weight-2 edge. In this case, all of the validation catalysts require to have a count of two since they are all black edges. Let V be an AND vertex with edges AV_w^2, VC_b, VB_b Let X be a black AND vertex with edges $A'X_b^2, VC_b', VB_b'$. A $(13, 10)$ rule to flip both $AV_w^2, A'X_b^2$ in a single turn is as follows: Player 1 reverses edge AV_w^2. Player 2 reverses edge $A'X_b^2$.

$$AV_w^2 + \overline{AV_w^2} + 2\overline{VC_b} + 2\overline{VB_b} + 2A'X_b^2 + \overline{A'X_b^2} + 2\overline{VC_b'} + 2\overline{VB_b'} \rightarrow \overline{AV_w^2} + 2\overline{VC_b} + 2\overline{VB_b} + \overline{A'X_b^2} + 2\overline{VC_b'} + 2\overline{VB_b'}+$$

Similar to Theorem 2, after each turn rule executes the system enters a cleanup phase that uses DEL_b species to delete every unused edge left by the choice vector:

$$(1)\ 2\,\overline{A'V_b^2} + A'V_b^2 + DEL_b \rightarrow \overline{A'V_b^2} \quad (2)\ 2\,A'V_b^2 + \overline{A'V_b^2} + DEL_b \rightarrow A'V_b^2$$
$$(3)\ BX_w^2 + \overline{BX_w^2} + \overrightarrow{w} \rightarrow BX_w^2 + \overrightarrow{w} \quad (4)\ BX_w^2 + \overline{BX_w^2} + \overrightarrow{\overrightarrow{w}} \rightarrow \overline{BX_w^2} + \overrightarrow{\overrightarrow{w}}$$
$$(5)\ 2\,BX_w^2 \rightarrow BX_w^2,\ \ 2\,\overline{BX_w^2} \rightarrow \overline{BX_w^2}$$

(1) and (2) are $(3, 1)$ void reactions that discard the unused black edges introduced by the choice vector. Each consumes one DEL_b and preserves the edge's current state. (3) and (4) discard an invalid white edge flip. Recall $\overrightarrow{w}$ (respectively $\overrightarrow{\overrightarrow{w}}$) represents a valid local configuration, so the stray copy is deleted while the valid orientation is retained. (5) handles the redundant case in which the choice vector adds a duplicate of the already-present white orientation.

With this construction, the IA-CRN reaches the target terminal configuration iff White has a winning strategy in the unbounded 2CL. Hence, Interactive CRN forced reachability with no bound in m is EXPTIME-hard.

5 Hardness for Randomized Interactive CRNs

In this section, we prove hardness for the reachability problem for Randomized Interactive CRNs defined in Definition 7 that allows some random number of copies of species to be added each step. Using this approach, we can transform the classical result IP=PSPACE to the PSPACE-hardness of the reachability problem of Randomized Interactive CRN with only $(2, 0)$ rules.

Definition 7 (Randomized Interactive CRN). *A Randomized Interactive CRN is a tuple* $C_{\text{rand}} = \big((\Lambda, \Gamma), h, m, n\big)$, *where* Λ *is a finite species set,* $\Gamma \subseteq \mathbb{N}^{\Lambda} \times \mathbb{N}^{\Lambda}$ *is the reaction set,* m *is the length of random bits in one step,* n *is the input-length for a problem (to solve with randomized interactive CRN). A series of interaction steps* S_0, R_0, S_1, $R_1, \ldots, S_{k-1}, R_{k-1}, S_k$ *with* $k \leq h$ *is allowed in CRN computation, where (1) each* S_i *is a species vector in* $\mathbb{N}^{\Lambda}$ *and depends on* $S_0, R_0, \ldots, S_{i-1}, R_{i-1}$, *and (2)* R_i *in* $\{0,1\}^{\Lambda}$ *encodes a sequence of* m *random bits. If* A *is the configuration right before a step* R_i, *it enters the configuration* $B = A + R_i$, *and then enters a terminal configuration* A' *after applying reactions.*

In Randomized Interactive CRNs, instead of adding a defined number of species at each step, the model allows the addition of species to encode some random bits. In this model, reactions fire in the usual fashion whenever they are enabled. The experimenter adds the next step whenever the system reaches a terminal configuration.

We define the reachability of Randomized Interactive CRNs below. A terminal configuration is a "Yes" configuration if it contains at least one copy of species "y".

Definition 8. *Let* $C_{\text{rand}} = \big((\Lambda, \Gamma), h, m, n\big)$ *be a randomized interactive CRN. We define: A terminal configuration is "Yes" if it contains at least one copy of species "y".*
A configuration "Yes" is fully reachable by C_{rand} *if for any* R_0, $R_1, \ldots$, R_{k-1}, *with* $k \leq h$ *there is a series of steps* S_0, R_0, S_1, R_1, $\ldots, S_{k-1}, R_{k-1}, S_k$ *and it enters "Yes" configuration.*
A configuration "Yes" is highly unreachable by C_{rand} *if a sequence of steps* S_0, R_0, S_1, R_1,*... does not enter* Yes *configuration with probability at least* $3/4$.
The reachability problem for C_{rand} *is to determine if Yes configuration is fully reachable.*
The randomized interactive CRN C_{rand} *is of a polynomial size if* $|\Lambda| + |\Gamma| + h + m \leq p(n)$ *for some polynomial* $p(.)$.

5.1 Interactive Proof Systems

An IP consists of a Prover **P** (unbounded computational power) and a Verifier **V** (probabilistic polynomial-time machine) that exchange messages. The dialogue protocol proceeds as follows. First, P asserts a statement S. Then P and V engage in k rounds of interaction: in round i, P sends proof π_i and V responds with challenge c_i. Finally, V either accepts or rejects S based on the

transcript $(\pi_1, c_1, \ldots, \pi_k, c_k)$. Shamir [16](later simplified by [17]) showed that IP = PSPACE by constructing an IP for the TQBF problem via arithmetization. The quantified boolean formula is converted to a polynomial P over some prime field $\mathbb{F}_p$, and in each round the prover sends a one-variable polynomial that the verifier checks with a random number in the field. The protocol runs for n rounds, one per quantified variable. In the final round, the verifier plugs in the last random value, evaluates the degree-1 polynomial directly, and halts with accept or reject.

Lemma 1. *In the TQBF interactive-proof protocol, after reducing each quantified polynomial modulo $x^2 - x$, the verifier's check in any round consists of evaluating a one-variable polynomial of degree at most 1 over $\mathbb{F}_p$. Furthermore, the verifier uses a random number in the range $[0, 2^i - 1]$, where i is the integer with $2^i \leq p < 2^{i+1}$.*

Proof. It is easy to see that the polynomial P the Verifier has to evaluate could be of a large degree as the size of the formula increases. In [17], the authors add the operation RxP after the arithmetization of each of the quantifiers. This operation reduces the degree of the polynomial $P \mod x^2 - x$. This means that any x^l where $l > 1$ is changed to x. This leaves P as a polynomial of degree ≤ 1 while keeping the correct boolean values. For two different polynomials $p_1(x)$ and $p_2(x)$ of degrees at most k over F_p, with probability at most $\frac{2k}{p}$, $p_1(x) = p_2(x)$ for a random number x in $[0, 2^i - 1]$. This is because the number of integers in the interval $[0, 2^i - 1]$ is at least $\frac{p}{2}$. Therefore, if p is selected to be at least double the size of the old choice, we can satisfy the same probability upper bound for a failure case in the proof system.

Using a random number in $[0, 2^i - 1]$ instead of $[0, p - 1]$ is easy to implement in CRNs. For an m bits random 0, 1-sequence $b_1 b_2 \ldots b_m$, we may use a copy of species $x_{i,0}$ to represent the case $b_i = 0$, and a copy of species $x_{i,1}$ to represent the case $b_i = 1$.

5.2 Circuits and circuit simulation

TC^0 is a class of circuits with constant depth and polynomial size using only AND, OR, NOT, and MAJORITY gates. It is known that TC circuits can do arithmetic. See Lemma 3 in the Appendix for a brief explanation. Reif and Tate (1992) [15] show multiplication and addition, and Hesse, Allender, and Barrington (2002) [11] show division. We use this class of circuits as a verifier in the protocol for TQBF. Step CRNs are known to simulate threshold circuits. See Lemma 4 in the Appendix for a brief explanation. Anderson et al. (2024) [2] show Step CRNs with (3,0) rules can simulate threshold circuits. We directly extend this to the Randomized Interactive CRN.

We show that $(3, 0)$ rules are PSPACE-hard in Theorem 4. We extend that proof and PSPACE-hardness to $(2, 0)$ rules in Theorem 5. These proofs rely on Lemma 2 from Anderson et. al. [3].

Lemma 2. *A Boolean formula with G gates and D depth can be computed by a $(2, 0)$ Step CRN with $\mathcal{O}(G)$ species, $\mathcal{O}(D)$ steps, and $\mathcal{O}(G)$ volume. Anderson et al. [3].*

Theorem 4. *The reachability problem in polynomial size steps of Randomized Interactive CRNs with $(3, 0)$ rules is PSPACE-hard.*

Proof. We show that the PSPACE-hard problem TQBF can be transformed into the reachability problem of Randomized Interactive CRNs with $(3, 0)$ rules. We let each S_i encode the message from the prover in round i, and R_i encode the random elements in $[0, 2^i - 1] \subseteq [0, p - 1]$ from the verifier in round i. For an instance F for TQBF, by Lemmas 1, 3, and 4, a polynomial size randomized interactive step CRN C_{rand}, which only has $(3, 0)$ rules, can be generated in polynomial time. We have $F \in TQBF$ if and only if the yes configuration of C_{rand} is fully reachable.

Theorem 5. *The reachability problem in polynomial size steps of Randomized Interactive CRNs with $(2, 0)$ rules is PSPACE-hard.*

Proof. We show that the PSPACE-hard problem TQBF can be transformed into the reachability problem of Interactive Randomized CRNs with $(2, 0)$ rules. Each threshold gate can be simulated by an NC_1 circuit of polynomial size and $O(\log n)$ depth. A NC_1 circuit can be transformed into a polynomial-size boolean formula by the classical result $NC_1 = BF$, where BF represents the class of decision problems with polynomial-size boolean formulas. We let each S_i encode the message from the prover in round i, and R_i encode the random elements in $[0, 2^i - 1] \subseteq [0, p - 1]$ from the verifier in round i. For an instance F for TQBF, by Lemmas 1, 3, and 2, a polynomial size randomized interactive CRN C_{rand}, which only has $(2, 0)$ rules, can be generated in polynomial time. We have $F \in TQBF$ if and only if the yes configuration of C_{rand} is fully reachable.

6 Discussion and Future Work

In this paper, we introduce Interactive CRN computation in both deterministic and randomized models. The hardness of reachability problems is studied for IA-CRNs with bounded-size void rules. Some interesting directions remain open. *Rule size*: our reductions use void rules of size $(12, 6)$ and $(17, 6)$, narrowing these parameters, or pinpointing the exact threshold where hardness emerges, would sharpen the model. *Plain reachability*: We analyzed the forced variant (requiring a unique terminal state). Determining the complexity of ordinary reachability for IA-CRNs, which we conjecture is still in PSPACE, is a natural next target.

References

1. Alaniz, R.M., Fu, B., Gomez, T., Grizzell, E., Rodriguez, A., Schweller, R., Wylie, T.: Reachability in restricted chemical reaction networks (2022)

2. Anderson, R., Avila, A., Fu, B., et al.: Computing threshold circuits with void reactions in step chemical reaction networks. In: International Conference on Machines, Computations, and Universality. pp. 52–71. Springer (2024)
3. Anderson, R., Fu, B., Massie, A., et al.: Computing threshold circuits with bimolecular void reactions in step chemical reaction networks. In: International Conference on Unconventional Computation and Natural Computation. pp. 253–268. Springer (2024)
4. Aris, R.: Prolegomena to the rational analysis of systems of chemical reactions. Archive for Rational Mechanics and Analysis **19**(2), 81–99 (jan 1965). https://doi.org/10.1007/BF00282276
5. Chalk, C., Martinez, E., Schweller, R., Vega, L., Winslow, A., Wylie, T.: Optimal staged self-assembly of general shapes. Algorithmica **80**, 1383–1409 (2018)
6. Cirlos, S.C., Gomez, T., Grizzell, E., Rodriguez, A., Schweller, R., Wylie, T.: Simulation of multiple stages in single bin active tile self-assembly. In: International Conference on Unconventional Computation and Natural Computation. pp. 155–170. Springer (2023)
7. Demaine, E.D., Eisenstat, S., Ishaque, M., Winslow, A.: One-dimensional staged self-assembly. Natural Computing **12**(2), 247–258 (2013)
8. Ellis, S.J., Klinge, T.H., Lathrop, J.I.: Robust chemical circuits. Biosystems **186**, 103983 (2019)
9. Fu, B., Gomez, T., Knobel, R., et al.: Brief announcement: Reachability in deletion-only chemical reaction networks. In: Proc. of the Symposium on Algorithmic Foundations of Dynamic Networks. SAND (2025)
10. Hearn, R.A.: Games, Puzzles, and Computation. Ph.D. thesis, Massachusetts Institute of Technology (May 2006)
11. Hesse, W., Allender, E., Mix Barrington, D.A.: Uniform constant-depth threshold circuits for division and iterated multiplication. Journal of Computer and System Sciences **65**(4), 695–716 (2002). https://doi.org/10.1016/S0022-0000(02)00025-9, special Issue on Complexity 2001
12. Karp, R.M., Miller, R.E.: Parallel program schemata. Journal of Computer and System Sciences **3**(2), 147–195 (1969). https://doi.org/10.1016/S0022-0000(69)80011-5
13. Mo, D., Stefanovic, D.: Iterative self-assembly with dynamic strength transformation and temperature control. In: DNA Computing and Molecular Programming: 19th International Conference, DNA 19, Tempe, AZ, USA, September 22-27, 2013. Proceedings 19. pp. 147–159. Springer (2013)
14. Petri, C.A.: Communication with automata (1966)
15. Reif, J.H., Tate, S.R.: On threshold circuits and polynomial computation. SIAM Journal on Computing **21**(5), 896–908 (1992). https://doi.org/10.1137/0221053
16. Shamir, A.: IP = PSPACE. J. ACM **39**(4), 869–877 (Oct 1992). https://doi.org/10.1145/146585.146609
17. Shen, A.: IP = PSPACE: simplified proof. J. ACM **39**(4), 878–880 (Oct 1992). https://doi.org/10.1145/146585.146613
18. Winfree, E.: Chemical reaction networks and stochastic local search. In: DNA Computing and Molecular Programming: 25th International Conference, DNA 25, Seattle, WA, USA, August 5–9, 2019, Proceedings 25. pp. 1–20. Springer (2019)

A Hardness for Interactive CRNs

Theorem 2. *The Interactive CRN Forced Reachability Problem is PSPACE-complete using only void rules of at most size (10,8) with a polynomially bounded number of steps.*

Proof. Given a constraint graph $G = (V, E)$ with edges partitioned into black (B) and white (W), we encode the instance into an IA-CRN as follows. Each edge is represented as a species named by its endpoints: an edge from vertex A to vertex B becomes species AB, while its flipped version is $\overline{AB}$. Weight-two edges include a superscript and each edge species has a subscript indicating which player it belongs to (see Figure 3a). At every step, the white player adds exactly one choice $\overrightarrow{c_j} \in \{\overrightarrow{c_0}, \ldots, \overrightarrow{c_{k-1}}\}$, where $k = |W|$. Each $\overrightarrow{c_j}$ consists of exactly one flipped white edge (the edge the white player attempts to reverse), one copy of every flipped black edge, and $|B| - 1$ copies of a cleanup species DEL_b. The idea is that when a choice is added, the flipped white edge species will consume the unflipped version. Similarly, one out of all of the flipped black edge species will consume the unflipped version. The rest of flipped black edge species that were added will be cleaned up by the cleanup species DEL_b after the reaction representing the edge reversals executes.

We construct the rules Γ as follows. Consider a white edge $e_w \in W$ directed into vertex A. To determine if e_w can be flipped (reversed), we first identify all other edges surrounding A. The different combinations of flipped and unflipped versions of these surrounding edges form local configurations $\overrightarrow{C_{e_w}}$. However, we only retain local configurations that satisfy the constraint when e_w is flipped. For example, in Figure 2a, if edge AV^2 is flipping away from vertex V, and the other edges directed a V are CV and BV, then the local configurations are $\{(CV, BV), (\overline{CV}, BV), (CV, \overline{BV}), (\overline{CV}, \overline{BV})\}$. Since these edges are weight-1, only the configuration $\overrightarrow{C_{AV}} = \{(CV, BV)\}$ would be retained, since this is the only scenario where reversing BA mantains the constraint. Similarly, for each black edge $e_b \in B$, we construct local configurations $\overrightarrow{C_{e_b}}$, retaining only valid configurations that satisfy vertex constraints. Every local configuration $\overrightarrow{w} \in \overrightarrow{C_{e_w}}$ and $\overrightarrow{b} \in \overrightarrow{C_{e_b}}$ will also include the species of the edge that is being reversed and its reversed version as a prefix. Finally, we form rules by combining each valid local configuration $\overrightarrow{w}$ for white edges with each valid configuration $\overrightarrow{b}$ for black edges, provided the moves remain valid when the white player's flip occurs first. Each rule γ_i thus has reactants $\overrightarrow{w} + \overrightarrow{b}$. The unflipped edge species are consumed, so the rest of the reactants are used as catalysts and remain as the product. An example reaction is shown in Figure 4. Vertex labels and edge directions correspond to those in Fig. 3.

VARIABLE vertex. See VARIABLE vertex V in Figure 3a . Initially, AV_w^2 and $A'V_b^2$ are present. For the rule flip AV_w^2, $\overrightarrow{w}$ will have AV_w^2, $\overline{AV_w^2}$ and $A'V_b^2$. These represent the edge that will be flipped, the flipped version of that edge, and the inflow check to vertex V. The second half $\overrightarrow{b}$ will include the same for

$$\overbrace{AV_w^2 + \overline{AV_w^2} + A'V_b^2}^{\overrightarrow{w}} + \underbrace{C'Y_b^2 + \overline{C'Y_b^2} + CY_w^2}_{\overrightarrow{b}} \rightarrow \overline{AV_w^2} + A'V_b^2 + \overline{C'Y_b^2} + CY_w^2$$

Fig. 6: $(6, 4)$ void rule to play a turn on a VARIABLE vertex. Player 1 selects vertex V and reverses edge AV_w^2. Player 2 selects Vertex Y and reverses edge $C'Y_b^2$. The graph after this turn is shown in Figure 3b.

another variable edge. In a valid flip, the only element consumed will be the species representing the original edges that were flipped by white and black. The complete $(6, 4)$ void reaction for a turn on a VARIABLE vertex is shown in Figure 4

AND vertex. Let F be an AND vertex with single edges FA_w, FB_W and double edge EF_w^2. To reverse EF_w^2, we have local configurations $\overrightarrow{C_{EF_w^2}} = \{(\overline{FA_w}, \overline{FB_w})\}$ that make up $\overrightarrow{w}$, the first half of the rule. The second half $\overrightarrow{b}$ consists of a valid move for Player 2. For simplicity in this example, we show a mirrored reversal of the player 1 move in a different vertex F'. Two copies of any reversed edge of the black player are needed in the reactants to avoid the black player's reversed edge species added in $\overrightarrow{c_j}$ from triggering this reaction in an invalid configuration. It is important to note that after the VARIABLE vertex selections, black moves are irrelevant. We still account for them to preserve turn order. The completed $(10, 8)$ void reaction for a turn on an AND vertex is as follows: Player 1 reverses edge EF_w^2. Player 2 reverses edge $E'F_b'^2$,

$$EF_w^2 + \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + E'F_b'^2 + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'} \rightarrow \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'}$$

OR vertex. Use the system in Figure 3 for reference. Now, let F be an OR vertex with double edges FA_w^2, FB_W^2, and EF_w^2. Vector $\overrightarrow{w}$ will contain EF_W^2, $\overline{EF_w^2}$, and one of $\overline{FA_W^2}, \overline{FB_w^2}$. Note that only one is needed since one double edge satisfies the inflow constraint. There will exist two reactions to reverse the output edge of an OR vertex. Vector $\overrightarrow{b}$ is formed the same as was for the AND vertex. The two $(7, 5)$ void reaction rules for a turn on an OR vertex are as follows: Player 1 reverses edge EF_w^2. Player 2 reverses edge $E'F_b'^2$. The first rule uses edges FA_w^2 and $F'A_b'^2$ to ensure the inflow constraint is met. The second one uses FB_w^2 and $F'B_b'^2$,

$$EF_w^2 + \overline{EF_w^2} + \overline{FA_w^2} + E'F_b'^2 + \overline{E'F_b'^2} + 2\overline{F'A_b'^2} \rightarrow \overline{EF_w^2} + \overline{FA_w^2} + \overline{E'F_b'^2} + 2\overline{F'A_b'^2}$$
$$EF_w^2 + \overline{EF_w^2} + \overline{FB_w^2} + E'F_b'^2 + \overline{E'F_b'^2} + 2\overline{F'B_b'^2} \rightarrow \overline{EF_w^2} + \overline{FB_w^2} + \overline{E'F_b'^2} + 2\overline{F'B_b'^2}$$

CHOICE vertex. In Figure 2d. Let V be a CHOICE vertex with single edges AV_w, CV_w, BV_w. Vector $\overrightarrow{w}$ will consist of $AV_w, \overline{AV_w}$, and the two edges that ensure the inflow constraint is met, CV_w, BV_w. Vector $\overrightarrow{b}$ is constructed the

same way as the OR and AND vertices. The completed $(8, 6)$ reaction for a turn on a CHOICE vertex is as follows: Player 1 reverses edge AV_w. Player 2 reverses edge $A'X_b$. Note that a choice vertex will also have more variations, as different edges can satisfy the constraint.

$$AV_w + \overline{AV_w} + CV_w + BV_w + A'X_b + \overline{A'X_b} + C'X'_b + B'X'_b \rightarrow \\ \overline{AV_w} + CV_w + BV_w + \overline{A'X_b} + C'X'_b + B'X'_b$$

FANOUT vertex. Using Figure 3. The two rules for each output single edge will need the double edge to ensure the constraint is met. Let A be a FANOUT vertex with single edges GA_w, FA_w and double edge AV_w^2. Vector $\overrightarrow{w}$ consists of one of GA_w, FA_w, whichever one is being reversed along with its reversed version, and $\overline{AV_w^2}$. Let B' be a FANOUT vertex with edges $G'B'_b, F'B'_b, B'X_b^2$. Vector $\overrightarrow{b}$ is constructed the same way as in the previous vertices. The two $(7, 5)$ void reactions for a turn on a FANOUT vertex are as follows: Player 1 reverses edge GA_w, FA_w. Player 2 reverses edge $G'B'_b, F'B'_b$,

$$GA_w + \overline{GA_w} + \overline{AV_w^2} + G'B'_b + \overline{G'B'_b} + 2\overline{B'X_b^2} \rightarrow \overline{GA_w} + \overline{AV_w^2} + \overline{G'B'_b} + \overline{B'X_b^2} \\ FA_w + \overline{FA_w} + \overline{AV_w^2} + F'B'_b + 2\overline{F'B'_b} + 2\overline{B'X_b^2} \rightarrow \overline{FA_w} + \overline{AV_w^2} + \overline{F'B'_b} + 2\overline{B'X_b^2}$$

After a reaction representing a turn executes, the reversed black edge species added in $\overrightarrow{c_j}$ will still be in the system. The species DEL_b then cleans up those leftover black edge species, leaving the system with only the species representing the current state of the graph. These rules are:

$$(1)\ 2\overline{A'V_b^2} + DEL_b \rightarrow \overline{A'V_b^2} \\ (2)\ A'V_b^2 + \overline{A'V_b^2} + DEL_b \rightarrow A'V_b^2 \\ (3)\ BX_w^2 + \overline{BX_w^2} \rightarrow BX_w^2, (4)\ 2\overline{BX_w^2} \rightarrow \overline{BX_w^2}$$

First, a (3,1) void rule to clean up unused black player flipped edges. This rule consumes one of two reversed black edge species for the case when the current state of the graph has $A'V_b^2$ in reversed state. Second is a (2,1) void rule for the case where the current state of the graph has $A'V_b^2$ in its initial state. Rules (3) and (4) will delete a white move and return the system to its original state before the choice. (3) takes the case for an invalid white flip. (4) is for the case of trying to flip an already flipped edge.

Using this construction, the IA-CRN reaches a terminal configuration containing the species corresponding to the reversed winning edge if and only if the white player has a winning strategy in the constraint logic game. This establishes the Interactive CRN Forced Reachability problem is PSPACE-hard. By Theorem 1, it follows that the problem is PSPACE-complete.

Theorem 3. *The Interactive CRN reachability problem is EXPTIME-hard, even when restricted to void rules of at most size (13,10) and allowing an unbounded number of steps.*

Proof. Given a constraint graph $G = (V, E)$ with E partitioned into B and W, we encode the instance into the IA-CRN the same as Theorem 2. After every step, the white player adds exactly one choice $\overrightarrow{c_j} \in \{\overrightarrow{c_0}, \ldots, \overrightarrow{c_{k-1}}\}$, where $k = 2|W|$.

one for every e_w and one for its reverse$\overline{e_w}$. Each $\overrightarrow{c_j}$ consists of exactly one white edge, one copy of every flipped black edge, one copy of every reversed black edge, and $|B| - 1$ copies of a cleanup species DEL_b. The rules are constructed the similar to Theorem 2. They keep a similar reactant structure of $\overrightarrow{w} + \overrightarrow{b}$. Note that now, they can have mixed edges, so $\overrightarrow{w}$ wont be exclusively white edges and the same for $\overrightarrow{b}$. For every black edge used as a validation catalyst in $\overrightarrow{w}$ and $\overrightarrow{b}$ will need 2 copies in its reactants. Having two of every species for every black edge ensures the rule matches the current state (one copy already in the configuration and one from the choice), and prevents the extra black edges in $\overrightarrow{c_j}$ from serving as catalysts. The rule keeps the flipped orientations and the existing DEL_b cleanup reactions then remove these duplicates, restoring a single valid edge species.

We explicitly show the unbounded construction with two representative gadgets. First, we revisit the AND vertex used in Theorem 2 and show how a weight-2 white edge can be flipped forward and back under our new "two-copy" convention. Then, we present the mixed-edge AND vertex in Figure 5c, where there are two single edges for black, and one double for white. This case produces the largest rule in the reduction. All other mixed-edge variants (single/weight-2 combinations and color permutations) can be found in the appendix.

Black AND Vertex. The rules for a black AND vertex follow almost the same as the AND vertex in Theorem 2. The only change is that the current state of the black edge that is being flipped requires 2 copies in the reactants. In the following rule, white is flipping edge EF_w^2 and black player is flipping $E'F_b'^2$. Note that now, two copies of $E'F_b'^2$ are required to verify the current state of the edge. See vertex F and F' in Figure 3 for reference.

$$EF_w^2 + \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + 2E'F_b'^2 + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'} \rightarrow \overline{EF_w^2} + \overline{FA_w} + \overline{FB_w} + \overline{E'F_b'^2} + 2\overline{F'A_b'} + 2\overline{F'B_b'}$$

Rules to reverse an edge again are symmetrical. To reverse edge $\overline{EF_w^2}$ and $\overline{E'F_b'^2}$ again, we must now ensure the valid local configurations of vertices E and E' are present. These now include edges $IE_w, HE_w, I'E_b', H'E_b'$. The $(11, 8)$ reaction is the following:

$$\overline{EF_w^2} + EF_w^2 + IE_w + HE_w + 2\overline{E'F_b'^2} + E'F_b'^2 + 2I'E_b' + 2H'E_b' \rightarrow EF_w^2 + IE_w + HE_w + E'F_b'^2 + 2I'E_b' + 2H'E_b'$$

Mixed AND Vertex. The largest rule in the reduction is triggered in the scenario where both players attempt to flip edges from an AND vertex with two black single-weight edges and one white weight-2 edge. In this case, all of the validation catalysts require to have a count of two since they are all black edges. Let V be an AND vertex with edges AV_w^2, VC_b, VB_b Let X be a black AND vertex with edges $A'X_b^2, XC_b', XB_b'$. A $(13, 10)$ rule to flip both $AV_w^2, A'X_b^2$ in a single turn is as follows: Player 1 reverses edge AV_w^2. Player 2 reverses edge $A'X_b^2$.

$$AV_w^2 + \overline{AV_w^2} + 2\overline{VC_b} + 2\overline{VB_b} + 2A'X_b^2 + \overline{A'X_b^2} + 2\overline{XC_b'} + 2\overline{XB_b'} \rightarrow \overline{AV_w^2} + 2\overline{VC_b} + 2\overline{VB_b} + \overline{A'X_b^2} + 2\overline{XC_b'} + 2\overline{XB_b'} +$$

B/W AND Vertex Case 1. There will be instances where the AND vertex will be half white edges and half black edges. Both players must make valid moves when they act on the same vertex. The white player's move happens first, so a rule where a single black edge is flipped into the vertex and then a white double edge is flipped away from the same vertex would not be valid within the same turn. This means we should only account for the white double edge pointing out once both black single edges are pointing in. Let V be an AND vertex with both single black edges VC'_b and VB'_b, and a white double edge AV^2_w. Let X be an AND vertex with both single white edges XC_w and XB_w, and a white double edge $A'X^2_b$. A $(13, 6)$ reaction for a turn to flip AV^2_w and $A'X^2_b$ is as follows:

$$AV^2_w + \overline{AV^2_w} + 2\overline{VC'_b} + 2\overline{VB'_b} + 2A'X^2_b + \overline{A'X^2_b} + \overline{XC_w} + \overline{XB_w} \rightarrow \overline{AV^2_w} + 2\overline{VC'_b} + 2\overline{VB'_b} + \overline{A'X^2_b} + \overline{XC_w} + \overline{XB_w}$$

Player 1 reverses edge AV^2_w. Player 2 reverses edge $A'X^2_b$. Note that $\overrightarrow{w}$ uses 2 black edges as catalysts and $\overrightarrow{b}$ uses 2 white edges as catalysts.

B/W AND Vertex Case 2. Another case is when each single edge in the AND vertex belongs to a different player. One white and one black single edge means both players can affect the same vertex in a single turn. This will only happen when both single edges are being flipped away from their common vertex. Let V be an AND vertex with one white single edge VC_w, one black single edge VB_b, and one white double edge AV^2_w. A $(6, 3)$ reaction for a turn to flip $\overline{VC_w}, \overline{VB_b}$ on a mixed AND vertex is as follows:

$$\overline{VC_w} + VC_w + AV^2_w + 2\overline{VB_b} + VB_b \rightarrow VC_w + AV^2_w + VB_b$$

Here, Player 1 reverses edge $\overline{VC_w}$. Player 2 reverses edge $\overline{VB_b}$.

Similar to Theorem 2, after each turn rule executes the system enters a cleanup phase that uses DEL_b species to delete every unused edge left by the choice vector:

$$(1)\ 2\,\overline{A'V^2_b} + A'V^2_b + DEL_b \rightarrow \overline{A'V^2_b} \quad (2)\ 2\,A'V^2_b + \overline{A'V^2_b} + DEL_b \rightarrow A'V^2_b$$
$$(3)\ BX^2_w + \overline{BX^2_w} + \overrightarrow{w} \rightarrow BX^2_w + \overrightarrow{w} \quad (4)\ BX^2_w + \overline{BX^2_w} + \overrightarrow{\overrightarrow{w}} \rightarrow \overline{BX^2_w} + \overrightarrow{\overrightarrow{w}}$$
$$(5)\ 2\,BX^2_w \rightarrow BX^2_w,\ \ 2\,\overline{BX^2_w} \rightarrow \overline{BX^2_w}$$

(1) and (2) are $(3, 1)$ void reactions that discard the unused black edges introduced by the choice vector. Each consumes one DEL_b and preserves the edge's current state. (3) and (4) discard an invalid white edge flip. Recall $\overrightarrow{w}$ (respectively $\overrightarrow{\overrightarrow{w}}$) represents a valid local configuration, so the stray copy is deleted while the valid orientation is retained. (5) handles the redundant case in which the choice vector adds a duplicate of the already-present white orientation.

With this construction, the IA-CRN reaches the target terminal configuration iff White has a winning strategy in the unbounded 2CL. Hence, Interactive CRN forced reachability with no bound in m is EXPTIME-hard.

B Randomized Interactive CRNs

Lemma 3. *Polynomial operations, which are based on $+, -, *$ in the field modulo a prime p (F_p), can be implemented using Threshold circuits with constant depth and polynomial size[15, 11].*

Proof. Let p be a fixed prime number. Polynomial operations over the finite field $\mathbb{Z}_p$, such as addition, multiplication, and division, can be executed using constant-depth, polynomial-size threshold circuits. Addition and multiplication of polynomials involve coefficient-wise operations and can be performed in TC^0 circuits [15]. Evaluating a polynomial at a specific point is to solve additions and multiplications in $\mathbb{Z}_p$, all of which are computable in TC^0 [15]. In [11], the authors then show division is also achievable in TC^0. Therefore, all fundamental polynomial operations modulo p are within TC^0 of constant depth and polynomial size.

Lemma 4. *Threshold circuits with G gates and D depth can be computed by a $(3, 0)$ Step CRN with $\mathcal{O}(G)$ species, $\mathcal{O}(D)$ steps, and $\mathcal{O}(G)$ volume. [2].*

Proof. In [2], the authors show that a step-CRN with only (2,0) void can simulate threshold formulas of size G and depth D under linear resource bounds. In particular, each formula gate is represented by a constant number of species, all gates at the same depth fire in one CRN step, and only $O(G)$ total volume is ever used. Then [3] refines the construction to handle threshold circuits (allowing unbounded fan-out) by suitably adjusting volume to account for copying across gates. They prove that the same $(2, 0)$-step CRN and rules suffice to compute a circuit of size G and depth D with only $O(G)$ species, in $O(D)$ steps, and using $O(G)$ volume when fan-out is bounded, or $O(G\, F_{\text{out}}^{D})$ in the general case.