

國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

表面化學反應網路的模擬能力

On the Simulation Power of Surface Chemical Reaction
Networks

李怡萱

Yi-Xuan Lee

指導教授：陳和麟 博士

Advisor: Ho-Lin Chen, Ph.D.

中華民國 113 年 7 月

July 2024







致謝

謝謝陳和麟老師。

因為太感謝了，需要一個換行的空間來深吸一口氣。從大四當老師的專題生開始，我就常常魯莽行事，paper 東看西看，有時連演算法怎麼做的都還沒搞清楚就跑去咪挺。老師依然非常有耐心的回答我的問題，一步一步引導我說清楚自己的想法，思考每個研究背後的 motivation，以及結果為什麼受到重視。老師在題目選擇上給予非常大的自由，總是問我「你想做的是什麼」。雖然好像是做研究最基本的問題，對我來說卻簡直靈魂拷問。研究過程中我遇到各種對定義的疑問、證明的瓶頸，老師總是能把我雜亂無章的想法化為有條理的思路，任何事情在老師的解釋之下都會變得很簡單。我太笨沒想通的時候，老師也會放慢速度等我想清楚了再繼續。從老師身上真的學到很多事情，非常感謝。也謝謝老師鼓勵我投稿，給我機會出國參加 conference，偶爾在咪挺之中得到老師的認同對我也是莫大的鼓舞。記得某次我事情做不完詢問老師能不能晚幾個小時，老師竟然回信說辛苦了，當下真的眼淚直接溢出來，何其有幸讓我遇到這麼好的老師。謝謝您。

謝謝青芳，我的老師，在無數個我發現自己證明爛掉緊張到不行的深夜陪我講幹話減輕我的焦慮，讓我能夠冷靜下來。通常我修到天亮時你也還醒著，真（可）巧（悲）。除了幹話之外也謝謝你給我很多實用的建議。有太多事情都是因為你在，我才能有試一試的勇氣。

謝謝實驗室的學長姐，尤其是政宏、子瑄，關心彼此論文進度及處理口試的一切事務有你們一起都讓人感到非常安心。也謝謝我的超級說說怪同屆作為實驗室的深夜夥伴及宵夜夥伴（雖然你總是自己偷買），每天認真之餘能跟你分享音樂時事及幹話也是不錯的。



謝謝所有口試委員，呂學一老師、于天立老師，及蔡孟宗老師，撥冗提供寶貴的建議與指教。

謝謝 UCNC 的大家願意與我交流，毫不吝嗇的給予我鼓勵並提出各種想法，謝謝你們讓我有一個很棒很溫馨的會議初體驗。

謝謝爸媽，接受我每天凌晨才回家，早上幾十個鬧鐘都還叫不醒。謝謝你們願意相信並支持我做自己想做的事。

謝謝所有在這段期間聽我發牢騷的朋友，謝謝你們理解我的迷惘。我想特別謝謝其中的三位朋友，謝謝 Sandy 許阿罵，在我大四邊打球邊準備研究所申請時給了我很多支持。謝謝貞諺美眉，每次我快要在實驗室悶成爛梅都是你把我抓出去放風。謝謝庭瑜，在我以為沒機會投稿時接起了我的電話也接住我在努力追趕之後發現還是趕不上的無力，讓我知道這些都沒關係的。

恕我無法一一點名所有幫助過我的人們，你們每個恰到好處的關心與邀約，都讓我充飽了電擁有繼續前進的能量。

謝謝你們。

李怡萱

2024.07.21



摘要

化學反應網路 (CRN) 是一個被廣泛研究的模型，描述混合溶液中分子的相互作用。2014 年，Qian 和 Winfree 提出了表面化學反應網路模型 (sCRN)，將分子置於平面圖的節點上，達到利用空間限制分子間相互作用的效果。在此模型裡，任何分子僅能與相鄰的分子進行反應。許多後續的研究著重在 sCRN 的計算能力並探討分子所能長成的圖案。

為了刻畫 sCRN 所能完成的事情，我們將此模型與其他已被廣泛研究的分子及自我組合系統做了連結。我們證明只要給定相同的初始條件，sCRN、親合力增強磁磚自動機、細胞自動機，及阿米巴機器人都可以互相模擬（經過一些必要的旋轉和翻轉）。此外，我們也考慮了凍結系統之間的模擬，在凍結限制下任意節點的狀態都只能依照某種篇序成長。特別地，我們引入凍結 sCRN，並證明了凍結親合力增強磁磚自動機及凍結細胞自動機可以被凍結 sCRN 模擬（經過必要的旋轉和翻轉）。其中一個我們使用的技巧是即時著色，透過即時著色我們能賦予分子對於方向的共識。

關鍵字：表面化學反應網路，模擬，磁磚自動機，細胞自動機





Abstract

The Chemical Reaction Network (CRN) is a well-studied model that describes the interaction of molecules in well-mixed solutions. In 2014, Qian and Winfree [25] proposed the abstract surface chemical reaction network model (sCRN), which takes advantage of spatial separation by placing molecules on a structured surface, limiting the interaction between molecules. In this model, molecules can only react with their immediate neighbors. Many follow-up works study the computational and pattern-construction power of sCRNs.

In this work, our goal is to describe the power of sCRN by relating the model to other well-studied models in distributed computation. We show that given the same initial configuration, sCRN, affinity-strengthening tile automata, cellular automata, and amoebot can all simulate each other (up to unavoidable rotation and reflection of the pattern). As an extension, we also consider the simulation between freezing systems where the state in any fixed position can only increase according to some partial order. In particular,

we introduce freezing sCRN and show that freezing affinity-strengthening tile automata and the freezing cellular automata can be simulated by freezing sCRN (up to unavoidable rotation and reflection of the pattern). One of our techniques is coloring on-the-fly, which allows all molecules in sCRN to have a global orientation.

Keywords: Surface chemical reaction networks, Simulation, Tile automata, Cellular automata



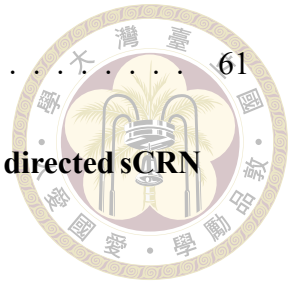
Contents

	Page
Acknowledgements	iii
摘要	v
Abstract	vii
Contents	ix
List of Figures	xiii
List of Tables	xv
Chapter 1 Introduction	1
1.1 Our results	3
Chapter 2 Models	5
2.1 Surface chemical reaction network	5
2.1.1 Surface chemical reaction network with orientation	6
2.1.2 Reachability and termination	7
2.2 Abstract tile assembly model	8
2.3 Cellular automata	10
2.4 Tile automata	12
2.5 Amoebot	14
2.6 Freezing system	17

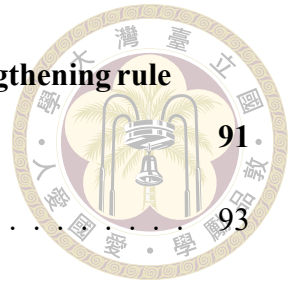


Chapter 3	Simulation	19
3.1	Representation function	19
3.2	Simulation definition	20
3.3	Notations	22
Chapter 4	Simulation of unit-seeded directed sCRN by unit-seeded sCRN (up to rotation and reflection)	23
4.1	Simulation overview	23
4.2	Protocol description	25
4.3	Proof	37
4.3.1	Gamma follows Gamma'	37
4.3.2	Gamma' models Gamma	39
4.3.3	Gamma' and Gamma have equivalent productions	45
4.4	Complexity	46
Chapter 5	Simulate aTAM by unit-seeded directed sCRN	49
5.1	Simulation overview	49
5.2	Protocols	51
5.3	Proof	54
5.4	Complexity	55
Chapter 6	Simulation between unit-seeded TA with affinity-strengthening rule and unit-seeded directed sCRN	57
6.1	Simulation overview	57
6.2	Protocols	58
6.3	Proof	60

6.4	Complexity	61
Chapter 7	Simulation between non-deterministic async-CA and directed sCRN	63
7.1	Simulate non-deterministic async-CA by directed sCRN	63
7.1.1	Simulation overview	64
7.1.2	Protocols	66
7.1.3	Proof	70
7.1.4	Complexity	72
7.2	Simulate directed sCRN by non-deterministic async-CA	72
7.2.1	Simulation overview	72
7.2.2	Protocols	74
7.2.3	Proof	76
7.2.4	Complexity	77
Chapter 8	Simulation between amoebot and clockwise sCRN	79
8.1	Simulating amoebot by clockwise sCRN	79
8.1.1	Simulation overview	79
8.1.2	Encoding particles	81
8.1.3	Observing flags	82
8.1.4	Movements	84
8.1.5	Proof	88
8.1.6	Complexity	89
8.2	Simulate clockwise sCRN by amoebot	90
8.2.1	Simulation overview	90



Chapter 9	Simulate unit-seeded, freezing TA with affinity-strengthening rule by unit-seeded, freezing directed sCRN	91
9.1	Simulation overview	93
9.2	Protocol	94
9.3	Complexity	97
Chapter 10	Simulate freezing async-CA by freezing directed sCRN	99
10.1	Simulation overview	100
10.2	Protocol	102
10.3	Proof	108
10.4	Complexity	114
Chapter 11	Conclusion	115
	References	117





List of Figures

4.1	Coloring the block of the seed by	26
4.2	An example of the simulation of a growing reaction.	28
4.3	3-fuzz simulation.	37
5.1	Tile species and observing species.	50
5.2	A simple example of simulating attachment.	53
7.1	Simulation of async-CA on sCRN.	69
7.2	Simulating sCRN reaction $C + A \rightarrow B + C$ on non-deterministic async-CA	73
8.1	Types of expanded species representing expanded particles	82
8.2	Examples of movements	84
9.1	Non-freezing sequence of reactions	92
10.1	A long locking path.	100
10.2	Simulating local function of freezing async-CA with freezing d-sCRN. . .	107





List of Tables

1.1 Cross-Model Simulation Results.	4
---	---





Chapter 1 Introduction

The Chemical Reaction Network (CRN) is a well-studied model that describes the interaction of molecules in well-mixed solutions. The computational power of CRNs has been studied in various settings, including rate-dependent [15, 19, 27] and rate-independent models [12, 13]. In this study, rather than considering molecules in a well-mixed solution, we study the scenarios in which molecules have designated locations and can only react with other molecules in their vicinity.

In 2014, Qian and Winfree [25] proposed the abstract surface chemical reaction network model (sCRN), which takes advantage of spatial separation by placing molecules on a structured surface, limiting the interaction between molecules. In this model, molecules can only react with their immediate neighbors. Qian and Winfree [25] also proposed an implementation of an arbitrary programmable sCRN using DNA strand displacement. Furthermore, they demonstrate the computational power of sCRN by building continuous active logic circuit and cellular automata in a parallel and scalable way.

A follow-up work by Clamons *et al.* [14] discuss the computational and pattern-creation power of sCRNs. They extend the idea of [25] to achieve local synchronicity on a 2-dimensional square lattice, with an initial pattern providing local orientation. This requires tethering the molecules precisely on the surface. On the other hand, they ask

whether we can achieve complex spatial arrangements from relatively simple initial conditions, which rely on the ability to control molecular interactions precisely. Some of the open questions are solved in [2], which focused on the complexity of deciding the reachability from a given configuration to another given configuration. Brailovskaya et al. [9] showed that feed-forward circuits can be constructed in sCRN using only swap reactions. In this work, our main goal is to study the computational power of sCRN given a uniform surface with no local orientation.

We seek to describe the power of sCRN by relating the model to other well-studied models in distributed computation, such as abstract tile assembly model (aTAM) [1, 18, 26, 29], cellular automata [8, 20, 28], and tile automata [3, 10, 11]. These models have been shown to be Turing universal [10, 16, 29] and can perform a large variety of tasks, including computation and pattern formation (e.g., [3, 8, 24]). We further compare sCRN to the amoebot model, which consists of programmable matter that has the ability to move. Also, as an extension, we restrict ourselves to freezing systems where the state in any fixed position can only increase according to some partial order. In particular, we introduce freezing sCRN and establish its connection to freezing cellular automata [7, 21, 22] and freezing tile automata [11], where similar concepts of freezing have previously been defined and studied in these models. It is also closely related to the notion of k -burnout (where each molecule can only change its state k times) studied within the sCRN model. We use the concept of “simulation” based on the definition given in [11, 23] with some necessary modifications. Intuitively, if two models can simulate each other, they can perform the same tasks in the same way (e.g., create the same pattern in the same ordering).

1.1 Our results



In this work, we show that given the same initial configuration, sCRN, affinity-strengthening tile automata, cellular automata, and amoebot can all simulate each other (up to unavoidable rotation and reflection of the pattern). Moreover, the freezing affinity-strengthening tile automata and the freezing cellular automata can be simulated by freezing sCRN. The results are listed in Table 1.1, where $\mathcal{M}_S, \mathcal{M}_T$ represent different models. We ask whether $\mathcal{M}_S$ can simulate $\mathcal{M}_T$ (denoted by $\mathcal{M}_S \triangleright \mathcal{M}_T$), and vice versa (denoted by $\mathcal{M}_T \triangleright \mathcal{M}_S$).

The main challenge for the simulation is that sCRN does not have predefined directions. Unlike the tile automata model (and all other models considered here), which has a defined orientation, the molecules in sCRN do not have the ability to distinguish their neighbors in different directions. We solve this problem by providing a 2-hop coloring to the surface such that each molecule has four neighbors with different colors. Furthermore, in the simulation of $\mathcal{M}_T$ by $\mathcal{M}_S$, we want the behavior of $\mathcal{M}_S$ to be as close to $\mathcal{M}_T$ as possible. When $\mathcal{M}_T$ enters a terminal configuration, $\mathcal{M}_S$ should also enter a terminal configuration without using too much extra space. Therefore, instead of coloring the whole surface, we carefully perform the coloring on the fly. A molecule is colored only when molecules in its vicinity are about to participate in a reaction. This coloring technique can be used in the simulation of all different models, so we describe this coloring technique separately in Section 4.

In Section 4, we show that unit-seeded sCRN (s-sCRN) can simulate unit-seeded, directed sCRN (s-d-sCRN) up to rotation and reflection, while simulating s-sCRN with s-d-sCRN is trivial. As an extension, we show in Section 9 that freezing s-d-sCRN can

Table 1.1: Cross-Model Simulation Results.

$\mathcal{M}_S$	$\mathcal{M}_T$	$\mathcal{M}_S \triangleright \mathcal{M}_T$	$\mathcal{M}_T \triangleright \mathcal{M}_S$	REF
s-sCRN	s-d-sCRN	✓ (up to rotation and reflection)	✓	Thm 5
s-d-sCRN	aTAM	✓	×	Thm 12
s-d-sCRN	s-as-TA	✓	✓	Thm 13
d-sCRN	async-CA	✓	✓	Thm 15, 16
c-sCRN	amoebot	✓	✓	Thm 18, 19
freezing sCRN	freezing s-as-TA	✓ (up to rotation and reflection)	✓	Cor 22
freezing d-sCRN	freezing async-CA	✓	✓	Thm 24

be simulated by freezing s-sCRN up to rotation and reflection. In Section 5, we show that s-d-sCRN can simulate aTAM, and it is easy to show that aTAM cannot simulate sCRN since the reactions are not reversible. With slight modifications, we show in Section 6 that s-d-sCRN and unit-seeded tile automata with affinity-strengthening rules (s-as-TA) can simulate each other, and the freezing version follows. In Section 7, we show that d-sCRN and asynchronous cellular automata (async-CA) with non-deterministic local function can simulate each other. Then we discuss the freezing version in Section 10 and show that freezing non-deterministic async-CA can be simulated by freezing s-d-sCRN. We also show that clockwise sCRN (c-sCRN) and amoebot can simulate each other. Notice that, when we simulate s-d-sCRN with s-sCRN, we are essentially giving a random global orientation to the surface. When simulating unit-seeded models, the orientation of the seed can be included in the coloring process and thus the terminal configuration of the sCRN is always the same as the system being simulated up to rotation and reflection. On the other hand, cellular automata and amoebots have complicated initial patterns. Therefore, simulating cellular automata and amoebots by s-sCRN (undirected) will inevitably rotate or flip the initial configuration before the simulation starts, unless the direction used in the cellular automata and amoebots is also encoded in the initial configuration of s-sCRN.



Chapter 2 Models

2.1 Surface chemical reaction network

The model of **surface chemical reaction network (sCRN)** was originally proposed by Qian and Winfree [25]. To align the underlying structures of all models and to describe the simulation problem, which is highly relative to the transformation of patterns in each system, we use a slightly different definition given in [2]. In particular, they define the configuration and reachability in sCRN. To make the simulations simpler, we also propose in Section 2.1.1 the directed sCRN as a variation of sCRN. We now give a brief description.

A surface chemical reaction network (Q, R) consists of an underlying *surface* L , a finite set of species Q , and a set of *reactions* R . In general, the surface L is an arbitrary planar graph, but we restrict it to the square lattice (viewed as $\mathbb{Z}^2$) and the triangular lattice with nearest neighbor connectivity in this paper. The *cells* are vertices of L , and each cell u is associated with a species $\sigma(u)$ in Q . For the purpose of defining the simulation between computational models in Section 3.2, we sometimes refer to the species as the “states” of a cell, and Q as the set of “states”. A *configuration* is a mapping from each cell to a state in Q . Every reaction $r \in R$ has one of the following two forms: $(A, B, C, D \in Q)$

- *Unimolecular reaction* $A \rightarrow B$. A species A can transform itself into B . Or,

- *Bimolecular reaction* $A+B \rightarrow C+D$ where A, B, C, D are not necessarily distinct.

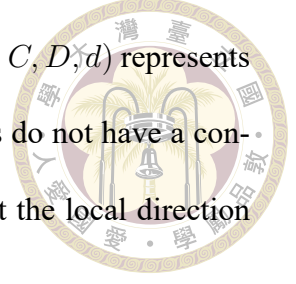
When two species A and B are adjacent, their states can simultaneously be replaced with C and D , respectively. Note that the orientation of A and B does not matter, but the order cannot be changed (species A must change into C , and B must change into D).

In this paper, we consider the sCRN $\Gamma = (Q, S, R)$ with a specified initial configuration S . We further define the **unit-seeded sCRN (s-sCRN)** where the initial configuration S maps every cell to a *blank state* $\mathcal{O} \in Q$ except for a special cell mapped to a *seed state* s . Define $Q^\dagger = Q \setminus \{\mathcal{O}\}$. In s-sCRN, there is no reaction without reactant in $Q^\dagger$. i.e. both $\mathcal{O} \rightarrow A$ and $\mathcal{O} + \mathcal{O} \rightarrow A + B$ are illegal.

2.1.1 Surface chemical reaction network with orientation

For the purpose of our simulation, we introduce and investigate two variants of the sCRN model: the **directed sCRN (d-sCRN)** and the **clockwise sCRN (c-sCRN)**. Directed sCRN assigns each bimolecular reaction a *global* direction $d \in \{N, E, S, W\}$ and use (A, B, C, D, d) to indicate that A and B can simultaneously transform into C and D when B is in the N (North), E (East), S (South), or W (West) direction relative to A . Here, “North” refers to the direction from $(0, 0)$ to $(0, 1)$ in $\mathbb{Z}^2$. This essentially provides species with a global orientation. Notice that $d = S, W$ are actually redundant since bimolecular reactions are symmetric. However, we define these operations to simplify the simulation protocols in Section 4. We use $(A, B, C, D, d)^{-1}$ to represent the reverse reaction (C, D, A, B, d) . For a unimolecular reaction where species A transforms into B , we use the notation $(A, B, \odot)$. Clockwise sCRN, defined on the triangular lattice, assigns each

bimolecular reaction a *local* direction $d \in \{0, \dots, 5\}$ such that (A, B, C, D, d) represents the reaction where B is in the d -th direction as seen from A . Species do not have a consensus on global orientation, but there is a shared understanding that the local direction increases in the clockwise order for all of them.



2.1.2 Reachability and termination

Let $\mathcal{C}$ be the set of all configurations in system Γ . We first define the *one-step reachability*. For any two configurations $\alpha, \beta \in \mathcal{C}$, β is reachable from α in one step if there exists a single cell or a pair of adjacent cells such that performing some reaction $r \in R$ on these cells yields β . Write $\alpha \rightarrow_{\Gamma}^1 \beta$.

Define the relation $\rightarrow_{\Gamma}^{\infty}$ on $\mathcal{C}$ s.t. for any configurations $\alpha, \beta \in \mathcal{C}$, $\alpha \rightarrow_{\Gamma}^{\infty} \beta$ if and only if there exists a (possibly infinite) sequence of reactions $\{r_i\}$ s.t. applying r_i successively to α produces the configuration β . That is, β is reachable from α in one or more steps. Let $\rightarrow_{\Gamma}$ be the reflexive transitive closure of $\rightarrow_{\Gamma}^{\infty}$. That is, the reflexive transitive closure of countable-step sequences.¹ We say that a configuration β is *reachable* from another configuration α if $\alpha \rightarrow_{\Gamma} \beta$.

A configuration α is Γ -*reachable* (or *reachable* when Γ is clear from the context) if $S \rightarrow_{\Gamma} \alpha$, and we denote the set of all Γ -reachable configurations by $\mathcal{A}(\Gamma)$. α is Γ -*terminal* if α is Γ -reachable and there is no configuration $\beta \neq \alpha$ that is reachable from α . The set of Γ -terminal configurations is denoted by $\mathcal{A}_*(\Gamma)$.

¹The common definition of reachability [2, 11] is the transitive closure of one-step reachability. Here, we aim to capture infinite pattern reachability and align with the notion of reachability in cellular automata models. Therefore, we define reachability over infinite steps and ensure that it is transitive.

2.2 Abstract tile assembly model



In his Ph.D. thesis [29], Winfree proposed the **abstract tile assembly model (aTAM)**. This model formalizes the self-assembly of molecules (such as DNA), describing the process wherein simple tiles spontaneously attach to each other to produce complex structures. In this paper, we use the model proposed in his later work with Rothemund [26]. Two differences we make are that we use the definition of assembly in [18] to align with the one used in Section 2.4 when introducing tile automata system, and we use the definition of attachability in [1] to make our description of the protocol in Section 5 more comprehensive. Note that these two definitions are equivalent to the original one given in [29]. We include a brief description here to make this paper self-contained.

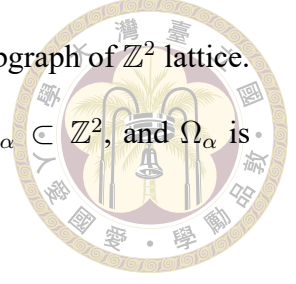
A *tile* is an oriented unit square with the north, east, south, and west sides labeled from some alphabet Σ . For a tile t , define the *state* σ to be the 4-tuple $\sigma(t) = (\sigma_N(t), \sigma_E(t), \sigma_S(t), \sigma_W(t))$, consisting of the labels (also called the *glues*) on its four sides. It is assumed that $\text{null} \in \Sigma$, and we use $\text{null} = (\text{null}, \text{null}, \text{null}, \text{null})$ (abuse of notation) to represent the absence of any other tiles. A *glue strength function* $g : \Sigma \times \Sigma \rightarrow \mathbb{N} \cup \{0\}$ maps a pair of labels to a natural number $\mathbb{N}$ if and only if the labels match each other.

$$g(x, y) = g(y, x) = \begin{cases} \hat{g}(x), & x = y \\ 0, & \text{otherwise} \end{cases}, \quad \forall x, y \in \Sigma, \text{ for some } \hat{g} : \Sigma \rightarrow \mathbb{N}$$

Given a finite set of states Q , it is allowed that an infinite number of tiles of the same state occupying locations in $\mathbb{Z}^2$. A *configuration* is a mapping from $\mathbb{Z}^2$ to Q ($\text{null} \in Q$). Let $Q^\dagger = Q \setminus \{\text{null}\}$, we consider the partial function $\alpha^\dagger$ from $\mathbb{Z}^2$ to $Q^\dagger$ where $\alpha^\dagger(v) = \alpha(v)$

for all $v \in \mathbb{Z}^2$ s.t. $\alpha(v) \neq \text{null}$. We can view the domain of $\alpha^\dagger$ as a subgraph of $\mathbb{Z}^2$ lattice.

We call α an *assembly* if $\alpha^\dagger$ has a connected, non-empty domain $\Omega_\alpha \subset \mathbb{Z}^2$, and Ω_α is called the *shape* of this assembly α .



Let α, β be two configurations s.t. $\alpha^\dagger, \beta^\dagger$ have disjoint domains. Addition of configurations α, β , denoted as $\gamma = \alpha + \beta$ (or $\beta = \gamma - \alpha$) is defined by

$$\gamma = \begin{cases} \alpha(x, y), & \beta(x, y) = \text{null}. \\ \beta(x, y), & \alpha(x, y) = \text{null}. \\ \text{null}, & \alpha(x, y) = \beta(x, y) = \text{null}. \end{cases}$$

The *interaction strength* between two adjacent tiles with their abutting sides labeled x, y is $g(x, y)$. Given a configuration α , define the *binding graph* B_α whose vertices are those tiles in $Q^\dagger$, with an edge of weight w between two vertices if the interaction strength between them is w . For some *temperature* $\tau > 0$, an assembly α is τ -stable if the graph B_α has min-cut $\geq \tau$.

We use $t_{(x_0, y_0)}$ where t is a tile to represent the configuration s.t.

$$t_{(x_0, y_0)}(x, y) = \begin{cases} \sigma(t), & (x, y) = (x_0, y_0). \\ \text{null}, & (x, y) \neq (x_0, y_0). \end{cases}$$

For a τ -stable assembly α , we say the position (x_0, y_0) is *attachable* in α if there exists an assembly β s.t. $\beta = \alpha + t_{(x_0, y_0)}$ and

$$\begin{aligned} &g(\sigma_N(t), \sigma_S(\beta(x_0, y_0 + 1))) + g(\sigma_E(t), \sigma_W(\beta(x_0 + 1, y_0))) \\ &+ g(\sigma_S(t), \sigma_N(\beta(x_0, y_0 - 1))) + g(\sigma_W(t), \sigma_E(\beta(x_0 - 1, y_0))) \geq \tau \end{aligned} \tag{2.1}$$

Notice that Equation(2.1) is equivalent to requiring β to be τ -stable. By the attachment of a single tile at position (x, y) , β is reachable from α in one step. Write $\alpha \xrightarrow{\tau^1} \beta$. The *reachability* $\rightarrow_\Gamma$ is defined the same as in Section 2.1.2. Also, we follow the definition of terminal set $\mathcal{A}_*(\Gamma)$ in Section 2.1.2 with an additional requirement that for any $\alpha \in \mathcal{A}_*(\Gamma)$, α must be τ -stable.

Such a system of aTAM is represented by a quadruple $\Gamma = (Q, S, g, \tau)$, where Q, g, τ are as above and S is the *seed configuration*. In this paper (also as suggested in [26]), we consider only the unit-seeded system where $S = s_{(0,0)}$ is the configuration s.t. $S(0, 0) = s$ for some *seed tile* s and all other locations are null.

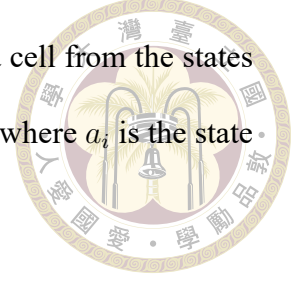
2.3 Cellular automata

The cellular automata was first designed in von Neumann's book [28]. It's a dynamical system that use local interaction to perform complex global behavior. Cellular automata has been studied to model natural phenomena, and it is also computationally universal. In this paper, we use the definition of synchronous cellular automata in the survey [8], and use the definition of asynchronous cellular automata in [20]. We include a brief description here to make this paper self-contained.

A d -dimensional cellular automata, whose underlying topology is $\mathbb{Z}^d$, is specified by a triple $\Gamma = (Q, \mathcal{N}, f)$

- Q is the finite state set.
- $\mathcal{N} = (\vec{v}_1, \dots, \vec{v}_N)$ is the neighborhood vector of N distinct elements of $\mathbb{Z}^d$. Then the *neighbors* of a cell at location $\vec{v} \in \mathbb{Z}^d$ are $\{\vec{v} + \vec{v}_i\}_{i=1}^N$.

- $f : Q^N \rightarrow Q$ is the *local rule* that computes the next state of a cell from the states of its neighbors. i.e. the next state of a cell $\vec{v}$ is $f(a_1, \dots, a_N)$ where a_i is the state of its neighbor $\vec{v} + \vec{v}_i$. All cells use the same rule.



A *configuration* in this system is a mapping $\alpha : \mathbb{Z}^d \rightarrow Q$ that specifies the state of all cells. The set of all possible configurations is denoted as $\mathcal{C}(d, Q) = Q^{\mathbb{Z}^d}$. In this paper, we only consider the 2-dimensional cellular automata with the *von-Neumann* neighborhood, which consists of all $\vec{v}_i$ s.t. $\|\vec{v}_i\|_1 \leq 1$. We fix $\mathcal{N} = (\vec{v}_O, \vec{v}_N, \vec{v}_E, \vec{v}_S, \vec{v}_W)$ throughout this paper, where $\vec{v}_O = (0, 0)$, $\vec{v}_N = (0, 1)$, $\vec{v}_E = (1, 0)$, $\vec{v}_S = (0, -1)$, $\vec{v}_W = (-1, 0)$.

Another way to identify a cellular automata is by its *global transition function* G . We define G in different ways depending on whether the cells are updated simultaneously. A brief introduction is included below.

synchronous CA (CA). All cells are updated simultaneously. For this case, the global function $G : \mathcal{C}(d, Q) \rightarrow \mathcal{C}(d, Q)$ maps configuration α to β ($\beta = G(\alpha)$) if for all $\vec{v} \in \mathbb{Z}^d$, $\beta(\vec{v}) = f(\alpha(\vec{v} + \vec{v}_1), \dots, \alpha(\vec{v} + \vec{v}_N))$.

asynchronous CA (async-CA). The cells follow a *fully asynchronous* updating scheme, where a single cell is chosen at random at each time step. The global function $G : \mathcal{C}(d, Q) \times \mathbb{Z}^d \rightarrow \mathcal{C}(d, Q)$ takes as input a configurations α and a randomly chosen cell $\vec{u} \in \mathbb{Z}^d$, and output the configuration $\beta = G(\alpha)$ s.t.

$$\beta(\vec{v}) = \begin{cases} f(\alpha(\vec{v} + \vec{v}_1), \dots, \alpha(\vec{v} + \vec{v}_N)), & \vec{v} = \vec{u} \\ \alpha(\vec{v}), & \text{otherwise} \end{cases}.$$

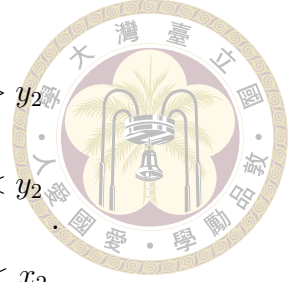
In this paper we consider a specific asynchronous cellular automata called **non-deterministic async-CA** where we allow the local function f to be *nondeterministic*. That is, for any

cell v , let $a_O, a_N, \dots, a_W$ be the states of the neighborhood $\vec{v}, \vec{v} + \vec{v}_N, \dots, \vec{v} + \vec{v}_W$ respectively. Then f maps $(a_O, a_N, a_E, a_S, a_W)$ to a set of possible next states, and one of them is picked non-deterministically at each time. A configuration β is reachable in one step from another configuration α , denoted by $\alpha \rightarrow_\Gamma^1 \beta$, if $\beta = G(\alpha)$. The *reachability* $\rightarrow_\Gamma$ is defined as in Section 2.1.2, and the terminal set $\mathcal{A}_*(\Gamma)$ is a subset of the reachable set that is a fixed point of the global function G . i.e. $\mathcal{A}_*(\Gamma) = \{\alpha \in \mathcal{A}(\Gamma) : G(\alpha) = \alpha\}$.

2.4 Tile automata

The tile automata model, combining *2-handed assembly model* with local state-change rules between pairs of adjacent tiles, is a marriage between tile-based self-assembly and asynchronous cellular automata. It was first proposed by Chalk *et al.* [11] and we follow their model definition. Variants like affinity strengthening and unit-seeded system were considered in [10] and [3] respectively. In this paper, we restrict ourselves to a system with both constraints, and slightly modify the description of single tile attachment to align with the one used in aTAM model introduced in Section 2.2. We include a brief description here to make this paper self-contained.

Similar to how we define a tile in the tile-based self-assembly, here a *tile* t is a unit square located in $\mathbb{Z}^2$, each assigned a *state* $\sigma(t)$ from the finite state set Q . Like in aTAM, we use $\text{null} \in Q$ to denote the absence of any other tile. An *affinity function* $g : Q^2 \times D \rightarrow \mathbb{N}$, where $D = \{\perp, \vdash\}$, represents the *affinity strength* between two states with relative orientation $d \in D$. To describe it explicitly, the affinity strength between two adjacent



$$\text{tiles } t_1, t_2 \text{ located at } (x_1, y_1), (x_2, y_2) \text{ is } \begin{cases} g(\sigma(t_1), \sigma(t_2), \perp), & \text{if } y_1 > y_2 \\ g(\sigma(t_2), \sigma(t_1), \perp), & \text{if } y_1 < y_2 \\ g(\sigma(t_1), \sigma(t_2), \vdash), & \text{if } x_1 < x_2 \\ g(\sigma(t_2), \sigma(t_1), \vdash), & \text{if } x_1 > x_2 \end{cases}$$

And the affinity strength between a null tile and any other tile is 0.

A *configuration* is a mapping α from $\mathbb{Z}^2$ to Q . Let $Q^\dagger = Q \setminus \{\text{null}\}$, we consider the partial function $\alpha^\dagger : \mathbb{Z}^2 \rightarrow Q^\dagger$ like the one defined in Section 2.2. An *assembly* is a configuration α where the domain of $\alpha^\dagger$ is connected. The *shape*, the additive notation “+”, and the induced *binding graph* B_α all follow the definitions in Section 2.2, where the weight w on an edge (t_1, t_2) now equals to the affinity strength between these two vertices. For a *stability threshold* τ , an assembly α is τ -stable if B_α has min-cut $\geq \tau$.

In general, an assembly α could break into two pieces, (say, τ -breakable) if there's a cut in the binding graph with total affinity strength $< \tau$. On the other hand, two τ -stable assembly may combine along a border whose total strength sums to at least τ . A special case is the *single tile attachment*, which is similar to the tile assembly system. For a τ -stable assembly α , we say the position (x_0, y_0) is *attachable* in α if there exists τ -stable assembly β s.t. $\beta = \alpha + t_{(x_0, y_0)}$ for some tile t . In this paper, we only consider the *affinity-strengthening rule* s.t. for each transition rule that takes a state σ_x^1 to σ_x^2 , it must satisfy that the affinity strength with all other states only increase. In other words, for every $\sigma \in Q$, $g(\sigma_x^1, \sigma, d) \leq g(\sigma_x^2, \sigma, d) \forall d \in D$. Notice that with the affinity-strengthening rule, any tile that has attached to a configuration can not fall off. This implies that the whole assembly is unbreakable after any state transition.

Like asynchronous cellular automata, tile automata has a local state changing rule.

The *transition rule* r is a 5-tuple $(\sigma_x^1, \sigma_y^1, \sigma_x^2, \sigma_y^2, d)$ where $\sigma_x^1, \sigma_y^1, \sigma_x^2, \sigma_y^2 \in Q$ and $d \in D$. It means that if states σ_x^1 and σ_y^1 are adjacent with relative orientation d , they can simultaneously turn into state σ_x^2, σ_y^2 respectively. In this paper, we allow the transition rule to be nondeterministic, i.e. there may be two rules $(\sigma_x^1, \sigma_y^1, \sigma_x^2, \sigma_y^2, d)$ and $(\sigma_x^1, \sigma_y^1, \sigma_x^3, \sigma_y^3, d)$ s.t. $(\sigma_x^2, \sigma_y^2) \neq (\sigma_x^3, \sigma_y^3)$.

A **tile automata system (TA)** is a 6-tuple $\Gamma = (Q, S, I, g, R, \tau)$ where Q is a finite set of states, g is an affinity function, S is the initial configuration, R is a set of transition rules, and τ is the stability threshold. We assume that initially, there is a set of tiles whose states belong to the *initial state* $I \subseteq Q$ that could be used in attachment. Any configuration β that is reachable from α in one step (denoted as $\alpha \rightarrow_\Gamma^1 \beta$) if and only if β is formed by applying some transition rule to α or by an attachment of a tile in I . The *reachability* $\rightarrow_\Gamma$ is defined the same as in Section 2.1.2. Also, we follow the definition of terminal set $\mathcal{A}_*(\Gamma)$ in Section 2.1.2 with an additional requirement that for any $\alpha \in \mathcal{A}_*(\Gamma)$, α must be τ -stable.

We restrict ourselves to the **unit-seeded TA with affinity-strengthening rule (s-as-TA)** in this paper. The term “unit-seeded” means that there exists a single seed tile s s.t. $S = s_{(0,0)}$ and only single tile attachment is allowed.

2.5 Amoebot

The amoebot model was originally proposed by Derakhshandeh *et al.* [17]. To simplify the simulation, we use the notions of movement and transition function given by Alumbaugh *et al.* [4], and propose a slightly different definition of configuration. Inspired by amoeba, it is a model of programmable matter where particles perform simple

computation according to local information, and can move via contraction and expansion. We include a brief description here to make this paper self-contained.



The amoebot model is an abstract computational model of *programmable matter* consisting of *particles*, which are simple computational units that can move and bond to others and exchange information by these bonds. The underlying topology is a triangular lattice with nearest neighbor connectivity $G_{\Delta} = (V, E)$. V represents all possible positions of a particle, and E represents all possible movement and information transitions between particles.

Every particle is either *contracted* (occupying a single node) or *expanded* (occupying two adjacent nodes). Particles are *anonymous*, but each edge leaving a particle p is locally labeled so that a particle can uniquely identify each of them: The labeling starts with 0 at an arbitrary outgoing edge leading to a node that is only adjacent to one of the nodes occupied by p , then increases in clockwise order around the particle. Notice that while they have a common clockwise chirality, they may have different *orientations* in $\mathcal{O} = \{0, 1, 2, 3, 4, 5\}$ encoding their offsets for local direction 0 from global direction 0 (to the right). i.e. they don't have consensus in global orientation.

Every particle has a constant-size, shared, local memory which can be read and write by both itself and its neighbor for communications. More formally, we denote an amoebot model by $\Gamma = (\Phi, S, \Sigma, \delta)$. Each particle has a *state* from a finite set Φ . A particle p communicates with an neighboring particle q by placing a *flag* from a finite alphabet Σ on the edges leading to q , so that q can read this flag.

Particles move through *expansions* and *contractions*: A contracted particle can *expand* into an unoccupied adjacent node and become expanded. The *head* of an expanded

particle is the node into which it last expanded, and the other node is the *tail*.² An expanded particle can *contract* to its head, performing a movement toward its head, or contract back to its tail. The direction of the edge labeled 0 remains constant during movement.² Neighboring particles can perform *handover* in one of the two ways:

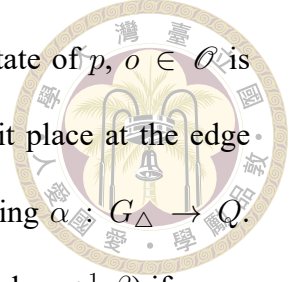
- A contracted particle p *push* an expanded neighbor q by expanding into one of the nodes occupied by q .
- An expanded particle p *pull* a contracted particle q by contracting, forcing q to expand into the currently vacating node.

Define M to be the set of all possible movements, $M = \{\text{idle}\} \cup \{\text{expand}_i : i \in 0, \dots, 5\} \cup \{\text{contract}_i : i \in 0, \dots, 9\} \cup \{\text{handover}_i : i \in 0, \dots, 5\}$. *idle* means that a particle does not move, expand_i and contract_i means expand/ contract toward the local direction i , and handover_i means that a contracted particle p expand toward its i -th direction, forcing an expanded q to contract.

In the execution of an amoebot algorithm, particles progress by performing atomic actions. We may assume a standard asynchronous computation model, i.e. only one particle is activates at a time. Each time a particle p is activate, it acts according to a *transition function* $\delta : \Phi \times \Sigma^{10} \times D \rightarrow \mathcal{P}(\Phi \times \Sigma^{10} \times D \times M)$, where D denote its tail direction of an expanded particle. We use $\varepsilon \in D$ to represents the contracted particle. The transition function takes as input the state of a particle, all the flags it reads, and maps to a set of *turns*. A turn specifies a state to transition into, flags to set, and a movement to execute.

For our simulation, we align the notation of *configuration* with other models. For

²When a particle expands, there could be two choices of edge to be labeled 0, in this case, the edge "away" from the particle is labeled 0.



a particle p , let $\sigma(p) = (\phi, o, d, f_0, \dots, f_9)$, where $\phi \in \Phi$ is the state of p , $o \in \mathcal{O}$ is the orientation of p , $d \in D$ is its tail direction, and f_i is the flag it place at the edge labeled i . Let $Q = \Phi \times \mathcal{O} \times D \times \Sigma^{10}$, a configuration is a mapping $\alpha: G_\Delta \rightarrow Q$. For two configuration α, β , β is *reachable* from α in one step (denoted $\alpha \rightarrow_{\Gamma}^1 \beta$) if α can become β after a single particle activation. Let S be the initial configuration, consisting of contracted particles which form a connected shape. In this paper, we follow the common assumption that there are only finitely many particles in S . According to what we have defined so far, we can also represent an amoebot system by $\Gamma = (Q, S, \delta)$, and we let Q be the new “state set” of the amoebot system throughout the rest of this paper.

2.6 Freezing system

Motivated by the freezing cellular automata model proposed by [22], we say that a system Γ is *freezing* if there exists a partial order $\preceq$ on the state set Q s.t. the state of any given cell can only increase with respect to $\preceq$. That is, for any configurations α, β s.t. $\alpha \rightarrow \beta$, $\alpha(v) \preceq \beta(v)$ for every cell v . In this paper, we'll consider the freezing sCRN, freezing tile automata, and freezing cellular automata. Notice that the definition we use is different from the one proposed by Chalk *et al.* [11] in the tile automata system.³ We chose the partial ordering definition because it implies transitioning into each state only once.

³In the reference, they consider *freezing* tile automata system defined as follows:

- construct a directed graph G s.t. each state $\sigma \in Q$ is a vertex, and for any two states σ_1, σ_2 , (σ_1, σ_2) is an edge if there exists a transition rule in R that turns σ_1 into σ_2 .
- Γ is said to be freezing if and only if G is acyclic.

Roughly speaking, a tile can never revisit the same state twice. Notice that in general, requiring tile to be freezing and requiring cell to be freezing is not equivalent. Tiles could detach, allowing new attachments to a fixed cell. But we only consider a-sa-TA where attached tiles will not fall off. In this case, the two perspectives are the same.





Chapter 3 Simulation

Our simulation definition is adapted from the ones used in [23], [4], and [11]. Let $\mathcal{S}, \mathcal{T} \in \{\text{sCRN}, \text{aTAM}, \text{s-as-TA}, \text{async-CA}, \text{amoebot}\}$ be two system in different models.

3.1 Representation function

Let $Q_{\mathcal{S}}, Q_{\mathcal{T}}$ be the state sets of $\mathcal{S}, \mathcal{T}$ respectively. A *state representation function* $\mathcal{R}$ from $Q_{\mathcal{S}}$ to $Q_{\mathcal{T}}$ is a function which takes as input a state of $\mathcal{S}$, and returns either information about the state of $\mathcal{T}$ or UND (which implies the image of $\mathcal{R}$ on this state is undefined).

Let $\mathbf{v}$ be the coordinates of a node in the underlying lattice L . Let α be a configuration of $\mathcal{S}$, and let $\alpha(\mathbf{v})$ be the state of α on the node $\mathbf{v}$. A *representation function* $\mathcal{R}^*$ from $\mathcal{S}$ to $\mathcal{T}$ takes as input an entire configuration α of $\mathcal{S}$ and apply $\mathcal{R}$ to every state of α , and returns either a corresponding configuration of $\mathcal{T}$, or UND if there is any state contained in α that mapped to UND.

Note that our definition is more relaxed than that in [23] and [4]. In particular we allow UND to be the image of R , which is motivated by the simulation definition in [5]. The relaxation is necessary because when simulating sCRN by non-deterministic async-

CA, it is possible that the sCRN perform a bimolecular reaction and hence two adjacent states A, B change simultaneously. While in the non-deterministic async-CA, it can be one cell performing state transition at a time. Suppose A change its state first, so the other cell B must be mapped to different states before and after A performing state transition. Another way of relaxing this definition to fit in with the limitation of non-deterministic async-CA is to let the representation function $\mathcal{R}$ map a neighborhood of CA to a state in sCRN, but this definition is rather non-intuitive in other models. The details are given in Section 7. Notice that except for the simulation of sCRN by non-deterministic async-CA and the simulation between sCRN and amoebot system, we don't need this relaxation.

3.2 Simulation definition

Roughly speaking, if we say that $\mathcal{S}$ simulates $\mathcal{T}$, we want the pattern of $\mathcal{S}$ to evolve like that in $\mathcal{T}$. Intuitively, we want for any sequence of configuration changes $\alpha \rightarrow \beta$ in $\mathcal{T}$, there exists a sequence of configuration changes $\alpha' \rightarrow \beta'$ s.t. α', β' are mapped to α, β respectively. On the other hand, for any $\alpha' \rightarrow \beta'$ in $\mathcal{T}$, it is also required that there is a sequence of configurations that links the images of α', β' under $\mathcal{R}^*$. i.e. $\mathcal{R}^*(\alpha') \rightarrow \mathcal{R}^*(\beta')$.

Definition 1. We say that $\mathcal{S}$ and $\mathcal{T}$ have equivalent productions (under $\mathcal{R}$), and we write $\mathcal{S} \Leftrightarrow_{\mathcal{R}} \mathcal{T}$ if the following conditions hold:

1. $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}(\mathcal{S})\} = \mathcal{A}(\mathcal{T}) \cup \{UND\}$.
2. $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}_*(\mathcal{S})\} = \mathcal{A}_*(\mathcal{T})$.

Definition 2. We say that $\mathcal{T}$ follows $\mathcal{S}$ (under $\mathcal{R}$) if $\alpha' \rightarrow_{\mathcal{S}} \beta'$ for some $\alpha', \beta' \in \mathcal{A}(\mathcal{S})$ implies $\mathcal{R}^*(\alpha') \rightarrow_{\mathcal{T}} \mathcal{R}^*(\beta')$, provided neither $\mathcal{R}^*(\alpha')$ nor $\mathcal{R}^*(\beta')$ is UND. This is denoted as $\mathcal{T} \dashv_{\mathcal{R}} \mathcal{S}$.

To be more rigorous, for every configuration $\alpha \in \mathcal{T}$, there must exist a reachable set Π in $\mathcal{S}$ which is mapped to α s.t. it is always possible to simulate the next configuration of α from this set. Further more, for any configuration $\alpha'' \in \mathcal{R}^*(\alpha)$ that can only simulate some of the next configuration of α , it is always possible that α'' can be achieved from some $\alpha' \in \Pi$. This can be viewed as a requirement that the two system must be in the same pace in most of the time.

Definition 3. We say that $\mathcal{S}$ models $\mathcal{T}$ (under $\mathcal{R}$) if for every $\alpha \in \mathcal{A}(\mathcal{T})$, there exists a non-empty set $\Pi \subset \mathcal{A}(\mathcal{S})$ such that $\mathcal{R}^*(\alpha') = \alpha \forall \alpha' \in \Pi$, and for every $\beta \in \mathcal{A}(\mathcal{T})$ where $\alpha \rightarrow \beta$, the followings hold:

1. for every $\alpha' \in \Pi$, there exists $\beta' \in \mathcal{A}(\mathcal{S})$ such that $\mathcal{R}^*(\beta') = \beta$ and $\alpha' \rightarrow_{\mathcal{S}} \beta'$.
2. for every $\alpha'' \in \mathcal{A}(\mathcal{S})$ where $\alpha'' \rightarrow_{\mathcal{S}} \beta'$ s.t. $\mathcal{R}^*(\alpha'') = \alpha$ and $\mathcal{R}^*(\beta') = \beta$, there exists $\alpha' \in \Pi$ such that $\alpha' \rightarrow_{\mathcal{S}} \alpha''$.

We denote this relationship by $\mathcal{S} \models_{\mathcal{R}} \mathcal{T}$.

Definition 4. We say that $\mathcal{S}$ simulates $\mathcal{T}$ (under $\mathcal{R}$) if $\mathcal{S} \Leftrightarrow_{\mathcal{R}} \mathcal{T}$, $\mathcal{T} \dashv_{\mathcal{R}} \mathcal{S}$, and $\mathcal{S} \models_{\mathcal{R}} \mathcal{T}$.

c-fuzz We use the concept of *c-fuzz* proposed by [11]. They design the simulation between two tile automata systems, and the idea is that we can have a bounded number of “extra” tiles attached to a configuration without affecting the mapping result. We apply their idea to any unit-seeded systems, in particular the systems of **s-sCRN**, **aTAM**, **a-sa-TA**. Here, with the notion of *c-fuzz*, we are essentially saying that the “extra” space participating in the simulation of any given configuration is bounded. Formally, the simulation of $\mathcal{T}$ by $\mathcal{S}$ with representation function $\mathcal{R}^*$ is said to have *c-fuzz* for some constant c if

the following holds: For any $\alpha' \in \mathcal{A}(S)$ and any $\mathbf{v}$ s.t. $\alpha'(\mathbf{v}) \in Q_S^\dagger$, there exists an intermediate configuration α'' such that α' is produced from α'' in one or more steps, and $\alpha = \mathcal{R}^*(\alpha'')$ satisfies $\alpha(\mathbf{v} + \mathbf{u}) \in Q_{\mathcal{T}}^\dagger$ for some $\|\mathbf{u}\|_\infty \leq c$. Recall that $Q^\dagger$ is the set of states without blank species (in sCRN systems) or null tiles (in tile systems). That is, almost all the space utilized in the simulating system used to be part of some configuration that has been simulated. This definition is slightly different from the one used in [11].¹

3.3 Notations

In the succeeding chapters, we are going to use the s-d-sCRN to simulate other models. Before continuing, we give some notations that will be used frequently.

- For $d = N, E, S, W$, define $d^{-1} = S, W, N, E$ respectively.
- Define $\mathcal{D} = \{N, E, S, W\}$ and $\psi_{\mathcal{D}} = (\psi_N, \psi_E, \psi_S, \psi_W)$.
- Define $\psi_{-d} = \{\psi_N, \psi_E, \psi_S, \psi_W\} \setminus \{\psi_d\}$. That is, we use $-d$ to represent the set of directions without d .
- $\langle \psi', \psi_{-d} \rangle$ means that we place $\{\psi'\} \cup \psi_{-d}$ in the $(\psi_N, \psi_E, \psi_S, \psi_W)$ order.
 - If $d = N$, $\langle \psi', \psi_{-d} \rangle = (\psi', \psi_E, \psi_S, \psi_W)$.
 - If $d = E$, $\langle \psi', \psi_{-d} \rangle = (\psi_N, \psi', \psi_S, \psi_W)$.
 - If $d = S$, $\langle \psi', \psi_{-d} \rangle = (\psi_N, \psi_E, \psi', \psi_W)$.
 - If $d = W$, $\langle \psi', \psi_{-d} \rangle = (\psi_N, \psi_E, \psi_S, \psi')$.

¹If we directly reformulate the definition in their paper in the unit-seeded systems, we would probably get the following: For any $\alpha' \in \mathcal{A}(S)$, let $\alpha = \mathcal{R}^*(\alpha')$. Then for all $\mathbf{v}$ s.t. $\alpha'(\mathbf{v}) \in Q_S^\dagger$, it must be the case that $\alpha(\mathbf{v} + \mathbf{u}) \in Q_{\mathcal{T}}^\dagger$ for some $\mathbf{u}$ s.t. $\|\mathbf{u}\|_\infty \leq c$. But in sCRN, cells can be reverted back to the blank states. Therefore, we care more about all the space that has been used during the simulation process, rather than the space used by the current configuration.



Chapter 4 Simulation of unit-seeded directed sCRN by unit-seeded sCRN (up to rotation and reflection)

In this section, we show that unit-seeded sCRN can simulate unit-seeded directed sCRNs up to rotation and reflection.

Theorem 5. *Given a unit-seeded directed sCRN $\Gamma = (Q, S, R)$, there exists a unit-seeded sCRN $\Gamma' = (Q', S', R')$ which simulates Γ up to rotation and reflection.*

4.1 Simulation overview

Given a s-d-sCRN $\Gamma = (Q, R, S)$, we want to construct a s-sCRN $\Gamma' = (Q', R', S')$ to simulate Γ . For bimolecular reactions with a specified direction d , it is required that each species has a common knowledge of a global orientation. The main idea is inspired by the study in population protocol [6] that enables each node to recognize its two different neighbors using a 2-hop coloring. We partition the plane into blocks of 9 cells, and color a block with ordering numbers $1, \dots, 9$ if and only if a reaction is about to happen on it.



The simulation consists of the following three parts:

- *Determine_Global_Orientation*: In this protocol, we color the seed and the eight cells around it with $\{1, \dots, 9\}$.
- *Growing_Reactions*: If certain reactions transform a cell from a blank species to another species, we first color the eight surrounding cells to establish a clear global orientation for that cell, and then proceed with the reaction.
- *State_Transitions*: Simulate any reactions at a cell who has already known the global orientation.

To be more precise, we define the *block* of a cell u , $\mathcal{B}(u) = \{v : \|v - u\|_\infty \leq 1\}$, to be the set of 9 cells around u . We also use $\mathcal{B}(\rho)$ to denote the block centered at the position of species ρ (abuse of notation). Recall that $Q^\dagger = Q \setminus \{\mathcal{O}\}$. The *growing reactions* are reactions of the form $(A, \mathcal{O}, B, C, d)$ where $A \in Q^\dagger$ and $B, C \in Q$. We denote the set of growing reactions in R by R_g .

Here is a sketch of the idea behind our simulation. Let $S' = S$ where S maps every cell to the blank state $\mathcal{O}$ except for a special cell that is mapped to a seed state, let's say s . In the first protocol *Determine_Global_Orientation*, we color $\mathcal{B}(s)$ with $\{1, \dots, 9\}$ to provide an orientation to the system. In the second protocol *Growing_Reactions*, we ensure that during the simulation process of any growing reaction $(A, \mathcal{O}, B, C, d)$, the state transition can be performed only if a coloring has been applied to $\mathcal{B}(\mathcal{O})$. This coloring aligns with the one provided in the first protocol to maintain the determined orientation for simulating future reactions. The last protocol *State_Transitions* defines the reactions necessary for simulating reactions in a fully colored configuration, which is relatively straightforward. We construct Γ' by adding states and reactions to Q' and R' successively.

The full description of the protocols is provided in Section 4.2 with more details. We give the proof of correctness in Section 4.3 and roughly analyze the complexity in Section 4.4 as well.

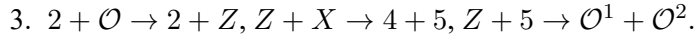
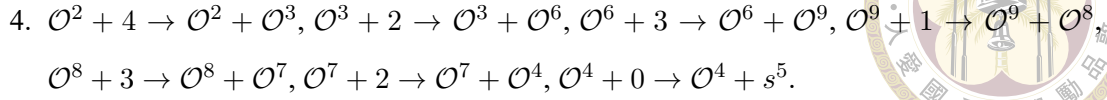


4.2 Protocol description

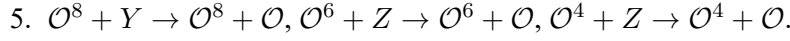
Determine_Global_Orientation This protocol aims to color all cells in the block of the seed. This coloring can be extended to the entire surface and be used as the global orientation. Let $S' = S$ where s is the special seed species in Γ . We color $\mathcal{B}(s)$ with $\{1, \dots, 9\}$ and use $\mathcal{O}^i$ ($i \in \{1, \dots, 9\}$) to denote the colored cells, where the superscripts i indicate the colors they receive.

Initially, we add the reaction $s + \mathcal{O} \rightarrow 0 + 1$ to R' . 0 and 1 then transform all their blank neighbors to X and Y respectively. When X and Y meet each other, they transition to 2 and 3 respectively, which guarantees that they are parallel and adjacent to 0 and 1. Now 2s are on two symmetric sides of 0, and each of them transforms its remaining blank neighbors to Z . The first Z encountered by X reacts with X , performing $Z + X \rightarrow 4 + 5$, while the other Z encountering 5 begins coloring the block of 0 in a clockwise order or counterclockwise order, depending on which side of Z encounters X first. Except for s , which will be transformed into s^5 , other cells receive $\mathcal{O}^i$ where i depends on their positions. After the coloring is complete, we need to eliminate the redundant states Y, Z grown outside $\mathcal{B}(s^5)$. This is done by reactions $\mathcal{O}^8 + Y \rightarrow \mathcal{O}^8 + \mathcal{O}$, $\mathcal{O}^6 + Z \rightarrow \mathcal{O}^6 + \mathcal{O}$, and $\mathcal{O}^4 + Z \rightarrow \mathcal{O}^4 + \mathcal{O}$ in R' . In total, we must include the following reactions in R' :

1. $s + \mathcal{O} \rightarrow 0 + 1$. // Initiate the coloring.
2. $0 + \mathcal{O} \rightarrow 0 + X, 1 + \mathcal{O} \rightarrow 1 + Y, X + Y \rightarrow 2 + 3$.



// s^5 is the new seed with its block colored.



// Eliminate the garbage states.

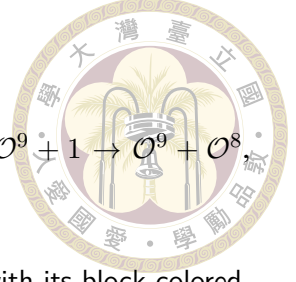


Figure 4.1 shows an evolution when we have this set of reactions.

Notice that the simulation system suffers unavoidable rotation and reflection since any starting reaction $s + \mathcal{O} \rightarrow A + B$ looks the same for s in its four directions, and then any reaction can be perform symmetrically around $\vec{AB}$. Therefore, “simulation up to rotation and reflection” is a necessary relaxation.

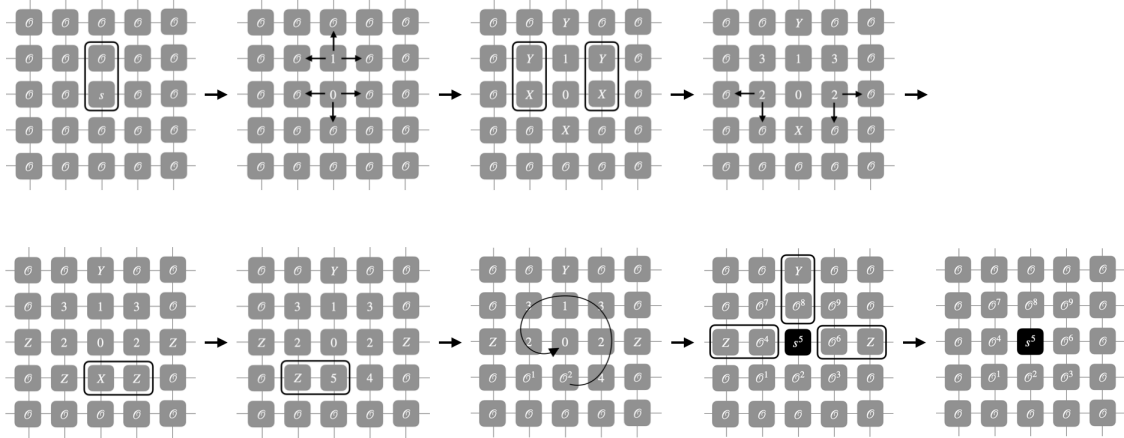


Figure 4.1: Coloring the block of the seed by
Determine_Global_Orientation

Growing_Reactions For any growing reactions of this form $(A, \mathcal{O}, B, C, d)$, we first give a coloring on $\mathcal{B}(\mathcal{O})$ and then perform the reaction. Before giving the protocol, there are some notations we need to define first. We use ψ^i to represent the colored species, where the superscript i is the color of that species. In the simulation, we will have 8 kinds

of species in Q' , which are introduced in detail later:

$$\{\mathcal{O}\}, \{\mathcal{O}^i : i \in [9]\}, \{\sigma^i : \sigma \in Q^\dagger, i \in [9]\}, \{\chi^i : i \in [9]\}, \{\chi_{ij} : i, j \in [9]\},$$

$$\{\chi_i^j : i, j \in [9]\}, \{\chi_i^j p : i, j \in [9]\}, \{\chi_i^j b_0 b_1 : i, j \in [9], b_0, b_1 \in \{0, 1\}\}.$$

Let Q' be the union of them. Among these states, let

$$\Xi^j = \{\sigma^j, \mathcal{O}^j, \chi^j, \chi_i^j, \chi_i^j p, \chi_i^j b_0 b_1 : \sigma \in Q^\dagger, i \in [9], b_0, b_1 \in \{0, 1\}\}$$

be the set of all colored species whose color is j . And let $\Xi = \cup_{j=1}^9 \Xi^j$.

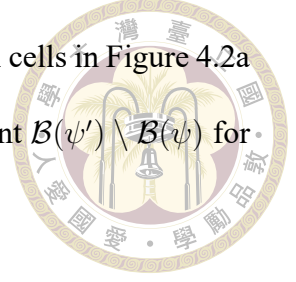
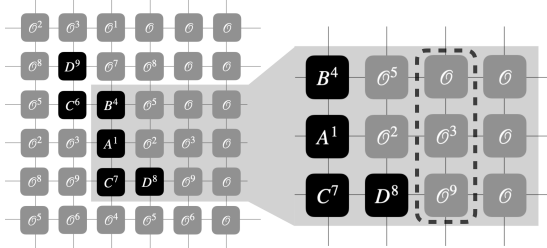
Let $Q_{\text{colored}}^\dagger = \{\sigma^i : \sigma \in Q^\dagger, i \in [9]\}$. Through the representation function $\mathcal{R}$, we map the state $\psi^i \in Q_{\text{colored}}^\dagger$ to ψ for all $i \in [9]$. Except for the states in $Q_{\text{colored}}^\dagger$, all other states are mapped to the blank state $\mathcal{O}$. We say that a configuration $\alpha' \in \Gamma'$ satisfies the *complete coloring property* if the following holds:

For any species $\psi \in Q_{\text{colored}}^\dagger \cup \{\chi^i : i \in [9]\}$, $\mathcal{B}(\psi)$ are completely colored.

Here, “completely colored” means that all the cells in $\mathcal{B}(\psi)$ contain some species in $\{\Xi^i : i \in [9]\}$. It will be demonstrated in the correctness proof that our protocol maintains the complete coloring property.

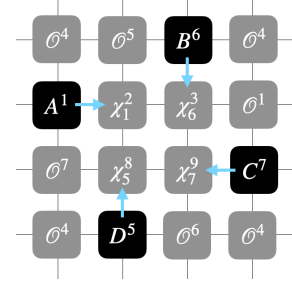
Suppose now we have a configuration $\alpha' \in \Gamma'$ that satisfies the complete coloring property, and there is some growing reaction $(A, \mathcal{O}, B, C, d) \in R$ that can be applied to $\alpha = \mathcal{R}^*(\alpha')$. For example, Figure 4.2a shows a configuration with the complete coloring property. Roughly speaking, if $(A, \mathcal{O}, B, C, E) \in R$, then we want $A^1, \mathcal{O}^2$ to become B^1, C^2 while preserving the complete coloring property at the same time. Existing colors 1, 2 on $A, \mathcal{O}$ help specifying the growing direction. Since $\mathcal{B}(A^1)$ has been colored, we only

need to color $W_{A^1}(\mathcal{O}^2) = \mathcal{B}(\mathcal{O}^2) \setminus \mathcal{B}(A^1)$, which are the three vertical cells in Figure 4.2a enclosed by the dashed lines. We use the notation $W_\psi(\psi')$ to represent $\mathcal{B}(\psi') \setminus \mathcal{B}(\psi)$ for adjacent species $\psi, \psi' \in Q'$.

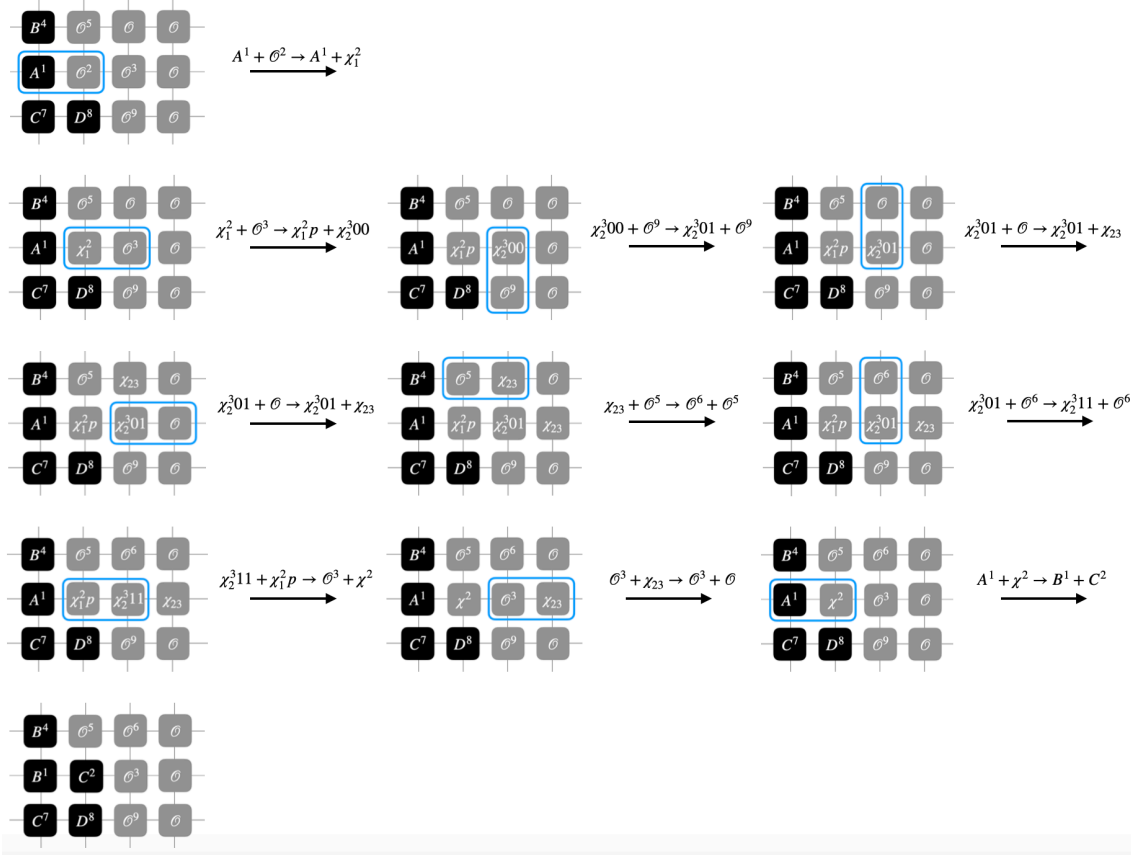


(a) A complete coloring configuration.

The wall centered at $\mathcal{O}^3$ are to be colored.



(b) The undesired deadlock.



(c) A possible sequence of reactions simulating $(A, \mathcal{O}, B, C, d)$.

Figure 4.2: An example of the simulation of a growing reaction.

The first reaction we add to R' is $A^1 + \mathcal{O}^2 \rightarrow A^1 + \chi_1^2$. χ_1^2 serves as a signal to begin the local coloring process. In the remaining part of this section, we'll list several possible patterns around the block of χ_1^2 , and specify the required reactions according to

that situation.

Now suppose that $\mathcal{B}(A^1)$ is completely colored due to the complete coloring property. From χ_1^2 we pass the information of growing direction one cell to the right. Suppose the cell on the east of χ_1^2 is v with state $\psi(v)$.

We categorize the possible values of $\psi(v)$ into 4 classes:

1. $\psi(v) = \sigma^3$ for some $\sigma \in Q^\dagger \cup \{\chi\}$. It means that $\mathcal{B}(v)$ has been colored, which implies $\mathcal{B}(\chi_1^2) \setminus \mathcal{B}(A^1) \subset \mathcal{B}(v)$ has been colored as well. Then we directly transform χ_1^2 to χ^2 , signifying that $\mathcal{B}(\chi_1^2)$ has been colored. Therefore, we add the following reactions to R' :

$$\chi_1^2 + \psi^3 \rightarrow \chi^2 + \psi^3, \text{ for all } \psi \in Q^\dagger \cup \{\chi\}.$$

2. $\psi(v) = \mathcal{O}$ or $\mathcal{O}^3$. We don't know if $\mathcal{B}(v)$ is colored. Therefore, we add the following reactions to R' :

$$\chi_1^2 + \mathcal{O} \rightarrow \chi_1^2 p + \chi_2^3 00.$$

$$\chi_1^2 + \mathcal{O}^3 \rightarrow \chi_1^2 p + \chi_2^3 00.$$

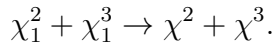
This announces the starting point of coloring $\mathcal{B}(\chi_2^3 00) \setminus \mathcal{B}(\chi_1^2 p)$.

3. $\psi(v) = \chi_6^3$ or χ_9^3 . We need to be cautious about the deadlock in Figure 4.2b. In this case, χ_1^2 continues to color its block. We include the following reactions in R' :

$$\chi_1^2 + \chi_6^3 \rightarrow \chi_1^2 p + \chi_2^3 00.$$

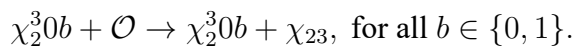
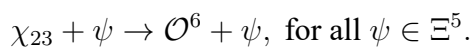
$$\chi_1^2 + \chi_9^3 \rightarrow \chi_1^2 p + \chi_2^3 00.$$

4. $\psi(v) = \chi_1^3$. It means that some species σ^1 (where $\sigma \in Q^+$) located three hops away from A^1 is also attempting to perform a growing reaction from the opposite direction. i.e. they are growing toward each other. Again by the complete coloring assumption, $\mathcal{B}(\chi_1^2) \setminus \mathcal{B}(A^1) \subset \mathcal{B}(\sigma^1)$ and $\mathcal{B}(\chi_1^3) \setminus \mathcal{B}(\sigma^1) \subset \mathcal{B}(A^1)$. Both χ_1^2, χ_1^3 know that their blocks are colored, so they transition simultaneously to χ^2 and χ^3 . Therefore, we add the following reaction to R' :

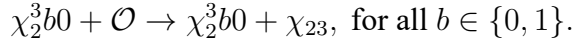
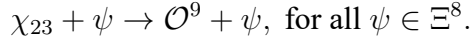


For case 2 and 3, the coloring of $\mathcal{B}(\chi_1^2 p)$ is incomplete. $\chi_2^3 00$ is produced to start coloring the upper and lower cells, referred to as v_N, v_S respectively. $\chi_2^3 00$ will be transformed into $\chi_2^3 b_0 b_1$ for some $b_0, b_1 \in \{0, 1\}$ that records whether v_N, v_S are colored. We call $\chi_2^3 b_0 b_1$ the *coloring species*. Let the states of v_N and v_S be $\psi(v_N)$ and $\psi(v_S)$ respectively. For convenience, we also let the cell on the east of $\chi_2^3 00$ be v_E , whose state is $\psi(v_E)$. There are several patterns for $\psi(v_N), \psi(v_S)$, and $\psi(v_E)$. We discuss them case by case:

1. $\psi(v_N) = \mathcal{O}$. $\chi_2^3 0b$ does not know whether it encounter v_N or v_E , so it transforms the cell into χ_{23} . χ_{23} then starts investigating its neighbors. When encountering $\psi \in \Xi^5$, χ_{23} confirms that it must itself be colored 6. Therefore, we add the following reactions to R' :

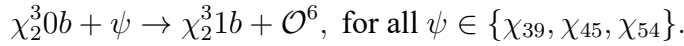


The case $\psi(v_S) = \mathcal{O}$ is similar. We include the following reactions in R' :

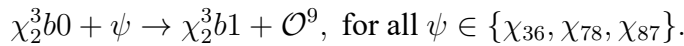


Notice that there might be a redundant state χ_{23} produced at cell v_E , which will be reverted back into $\mathcal{O}$ later.

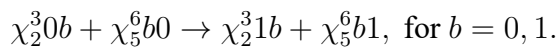
2. $\psi(v_N) \in \{\chi_{39}, \chi_{45}, \chi_{54}\}$. When encountering these states, $\chi_2^3 0b$ knows by the index that they are redundant states from other simulation process of some growing reactions, and that the cell it is investigating is v_N . We can directly transition that cell into $\mathcal{O}^6$. Therefore we add the following reactions to R' :



Similarly, for $\psi(v_S) \in \{\chi_{36}, \chi_{78}, \chi_{87}\}$, we include the following reactions in R' :



3. $\psi(v_N) = \chi_5^6 b 0$. This means that there is some species σ^4 (where $\sigma \in Q^+$) performing a growing reaction toward v_N . Then $\chi_2^3 0b$ and $\chi_5^6 b 0$ could help each other complete their coloring process by transitioning into a colored state. Therefore, we add the following reaction to R' :



For the other direction, $\psi(v_N)$ could be $\chi_4^6 0b$, indicating that a species σ^5 is perform-

ing a growing reaction toward v_N . Similar as above, we must have



$$\chi_2^3 0b + \chi_4^6 0b \rightarrow \chi_2^3 1b + \chi_4^6 1b, \text{ for } b = 0, 1.$$

The case $\psi(v_S) = \chi_8^9 0b$ or $\chi_7^9 b0$ is symmetric. Therefore, we include the following reactions in R' as well:

$$\chi_2^3 b0 + \chi_8^9 0b \rightarrow \chi_2^3 b1 + \chi_8^9 1b, \text{ for } b = 0, 1.$$

$$\chi_2^3 b0 + \chi_7^9 b0 \rightarrow \chi_2^3 b1 + \chi_7^9 b1, \text{ for } b = 0, 1.$$

4. $\psi(v_N) \in \Xi^6 \setminus \{\chi_5^6 b0, \chi_4^6 0b\}$. Then $\psi(v_N)$ is viewed as a colored species and $\chi_2^3 0b$ is transformed into $\chi_2^3 1b$ directly. We add the following reactions to R' :

$$\chi_2^3 0b + \psi \rightarrow \chi_2^3 1b + \psi, \text{ for all } \psi(v_N) \in \Xi^6 \setminus \{\chi_5^6 b0, \chi_4^6 0b\}.$$

Similarly, for $\psi(v_S) \in \Xi^9 \setminus \{\chi_8^9 0b, \chi_7^9 b0\}$, we add the following reactions to R' :

$$\chi_2^3 b0 + \psi \rightarrow \chi_2^3 b1 + \psi, \text{ for all } \psi(v_S) \in \Xi^9 \setminus \{\chi_8^9 0b, \chi_7^9 b0\}.$$

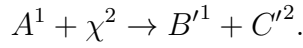
Eventually, $\chi_2^3 00$ will become $\chi_2^3 11$, indicating that both v_N, v_S have been colored.

Then we use a reaction $\chi_2^3 11 + \chi_1^2 p \rightarrow \mathcal{O}^3 + \chi^2$ to announce the termination of the local coloring process. The state transition can now be simulated.

- If $B, C \in Q^\dagger$, then we simply add the reaction $A^1 + \chi^2 \rightarrow B^1 + C^2$ to R' .
- If either B or C is $\mathcal{O}$, then we transform that cell into χ^1 or χ^2 respectively to indicate that the cell has been colored. We do not transform them into $\mathcal{O}^1$ or $\mathcal{O}^2$ because we want to avoid unnecessary re-coloring process.

Therefore, let $B' = \begin{cases} B & \text{if } B \in Q^\dagger \\ \chi & \text{if } B = \mathcal{O} \end{cases}$ and $C' = \begin{cases} C & \text{if } C \in Q^\dagger \\ \chi & \text{if } C = \mathcal{O} \end{cases}$, we add the following.

reaction to R' :



One thing remaining is to eliminate the redundant state χ_{23} produced in the first case.

Therefore, we need the reaction $\mathcal{O}^3 + \chi_{23} \rightarrow \mathcal{O}^3 + \mathcal{O}$.

The above description illustrates the simulation process when a growing reaction is applied to a species A colored by 1 and its eastern neighbor. Figure 4.2c depicts a possible sequence of reactions when simulating a growing reaction $(A, \mathcal{O}, B, C, E)$ in R .

To construct the full reaction set for simulating the growing reaction $(A, \mathcal{O}, B, C, E)$, we apply translations to all the reactions constructed thus far to initiate the growing reaction from any species A colored i , where $i \in [9]$. For other directions $d \in \mathcal{D}$, we simulate the growing reaction $(A, \mathcal{O}, B, C, d)$ by applying an additional rotation. We characterize the compositions of these translations and rotations using permutations.

More precisely, we denote by $\hat{R}$ the reaction set constructed for the growing reaction $(A, \mathcal{O}, B, C, E)$ starting at A^1 . By stating “apply a permutation π to $\hat{R}$ ”, we replace all the subscript i and superscript j in the above simulation process to $\pi(i)$ and $\pi(j)$ respectively.

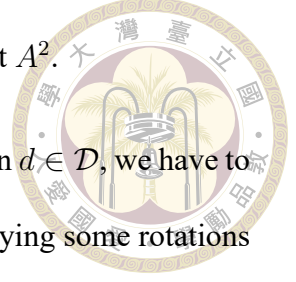
For example, let $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 6 & 9 & 2 & 5 & 8 & 1 & 4 & 7 \end{pmatrix}$. If we apply π to $\hat{R}$, then we get

all the reactions we need for the growing reaction $(A, \mathcal{O}, B, C, N)$ starting at A^3 . Further,

let $\pi' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 1 & 2 & 6 & 4 & 5 & 9 & 7 & 8 \end{pmatrix}$, then applying $\pi' \circ \pi$ to $\hat{R}$ gives us the reactions



needed for simulating the growing reaction toward North that starts at A^2 .



In general, for every growing reaction $(A, \mathcal{O}, B, C, d)$ in direction $d \in \mathcal{D}$, we have to include the reactions on all $A^i, i \in [9]$. Each case can be view as applying some rotations and translations to $\hat{R}$. We represent all the needed permutations by two-line notations.

First we define 4 translations $\pi_r, \pi_l, \pi_u, \pi_d$:

$$\pi_r = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 1 & 5 & 6 & 4 & 8 & 9 & 7 \end{pmatrix}, \pi_l = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 1 & 2 & 6 & 4 & 5 & 9 & 7 & 8 \end{pmatrix},$$

$$\pi_u = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 4 & 5 & 6 & 7 & 8 & 9 & 1 & 2 & 3 \end{pmatrix}, \pi_d = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 8 & 9 & 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}.$$

Let $\Pi_{\text{translation}} = \{id, \pi_r, \pi_l, \pi_u, \pi_d, \pi_u \circ \pi_r, \pi_u \circ \pi_l, \pi_d \circ \pi_r, \pi_d \circ \pi_l\}$. By applying each permutation in $\Pi_{\text{translation}}$ to $\hat{R}$, we could cover all the growing reactions $(A, \mathcal{O}, B, C, E)$ starting at any $A^i \in Q'$ s.t. $i \in [9]$.

We also define 4 rotations $\pi_E, \pi_N, \pi_W, \pi_S$:

$$\pi_E = id = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix}, \pi_N = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 6 & 9 & 2 & 5 & 8 & 1 & 4 & 7 \end{pmatrix},$$

$$\pi_W = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}, \pi_S = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 7 & 4 & 1 & 8 & 5 & 2 & 9 & 6 & 3 \end{pmatrix}.$$

Let $\Pi_{\text{rotation}} = \{\pi_E, \pi_N, \pi_W, \pi_S\}$. By applying the permutations in Π_{rotation} to $\hat{R}$, we could simulate any growing reactions $(A, \mathcal{O}, B, C, d)$ toward different directions.

Let $\Pi_d = \{\pi' \circ \pi_d : \pi' \in \Pi_{\text{translation}}\}, d \in \mathcal{D}$. By applying $\pi \in \Pi_d$ to $\hat{R}$, we can simulate any reactions starting from $A^i, i \in [9]$ toward a specific direction d . The entire

simulation protocol for a growing reaction $(A, \mathcal{O}, C, D, d)$ is described below: For all growing reactions $(A, \mathcal{O}, B, C, d) \in R_g$, add the following reactions to R' for all $\pi \in \Pi_d$:



$$1. A^{\pi(1)} + \mathcal{O}^{\pi(2)} \rightarrow A^{\pi(1)} + \chi_{\pi(1)}^{\pi(2)}. \quad // \text{Initiate the local coloring process.}$$

$$2. \chi_{\pi(1)}^{\pi(2)} + \psi^{\pi(3)} \rightarrow \chi^{\pi(2)} + \psi^{\pi(3)}, \text{ for all } \psi \in Q^{\dagger} \cup \{\chi\}.$$

// $\mathcal{B}(\chi_{\pi(1)}^{\pi(2)})$ has been colored.

$$3. \chi_{\pi(1)}^{\pi(2)} + \mathcal{O}/\mathcal{O}^{\pi(3)} \rightarrow \chi_{\pi(1)}^{\pi(2)}p + \chi_{\pi(2)}^{\pi(3)}00.$$

$$\chi_{\pi(1)}^{\pi(2)} + \chi_{\pi(6)}^{\pi(3)}/\chi_{\pi(9)}^{\pi(3)} \rightarrow \chi_{\pi(1)}^{\pi(2)}p + \chi_{\pi(2)}^{\pi(3)}00.$$

// $\mathcal{B}(\chi_{\pi(1)}^{\pi(2)}p) \setminus \mathcal{B}(A^1)$ is going to be colored.

$$4. \chi_{\pi(1)}^{\pi(2)} + \chi_{\pi(1)}^{\pi(3)} \rightarrow \chi^{\pi(2)} + \chi^{\pi(3)}.$$

// Both blocks $\mathcal{B}(\chi_{\pi(1)}^{\pi(2)}), \mathcal{B}(\chi_{\pi(1)}^{\pi(3)})$ have been colored.

$$5. \chi_{\pi(2)}^{\pi(3)}0b + \mathcal{O} \rightarrow \chi_{\pi(2)}^{\pi(3)}0b + \chi_{23}, \text{ for all } b \in \{0, 1\}.$$

$$\chi_{\pi(2)\pi(3)} + \psi \rightarrow \mathcal{O}^{\pi(6)} + \psi, \text{ for all } \psi \in \Xi^5.$$

$$\chi_{\pi(2)}^{\pi(3)}b0 + \mathcal{O} \rightarrow \chi_{\pi(2)}^{\pi(3)}b0 + \chi_{\pi(2)\pi(3)}, \text{ for all } b \in \{0, 1\}.$$

$$\chi_{\pi(2)\pi(3)} + \psi \rightarrow \mathcal{O}^{\pi(9)} + \psi, \text{ for all } \psi \in \Xi^8.$$

// A coloring species observes the adjacent blank species $\mathcal{O}$.

$$6. \chi_{\pi(2)}^{\pi(3)}0b + \psi \rightarrow \chi_{\pi(2)}^{\pi(3)}1b + \mathcal{O}^{\pi(6)}, \text{ for all } \psi \in \{\chi_{\pi(3)\pi(9)}, \chi_{\pi(4)\pi(5)}, \chi_{\pi(5)\pi(4)}\}.$$

$$\chi_{\pi(2)}^{\pi(3)}b0 + \psi \rightarrow \chi_{\pi(2)}^{\pi(3)}b1 + \mathcal{O}^{\pi(9)}, \text{ for all } \psi \in \{\chi_{\pi(3)\pi(6)}, \chi_{\pi(7)\pi(8)}, \chi_{\pi(8)\pi(7)}\}.$$

// A coloring species meets the redundant states from other growing reactions

$$7. \chi_{\pi(2)}^{\pi(3)}0b + \chi_{\pi(5)}^{\pi(6)}b0 \rightarrow \chi_{\pi(2)}^{\pi(3)}1b + \chi_{\pi(5)}^{\pi(6)}b1, \text{ for } b = 0, 1.$$

$$\chi_{\pi(2)}^{\pi(3)}0b + \chi_{\pi(4)}^{\pi(6)}0b \rightarrow \chi_{\pi(2)}^{\pi(3)}1b + \chi_{\pi(4)}^{\pi(6)}1b, \text{ for } b = 0, 1.$$

$$\chi_{\pi(2)}^{\pi(3)}b0 + \chi_{\pi(8)}^{\pi(9)}0b \rightarrow \chi_{\pi(2)}^{\pi(3)}b1 + \chi_{\pi(8)}^{\pi(9)}1b, \text{ for } b = 0, 1.$$

$$\chi_{\pi(2)}^{\pi(3)}b0 + \chi_{\pi(7)}^{\pi(9)}b0 \rightarrow \chi_{\pi(2)}^{\pi(3)}b1 + \chi_{\pi(7)}^{\pi(9)}b1, \text{ for } b = 0, 1.$$

// Two coloring species observing each other's.

$$8. \chi_{\pi(2)}^{\pi(3)}0b + \psi \rightarrow \chi_{\pi(2)}^{\pi(3)}1b + \psi, \text{ for all } \psi(v_N) \in \Xi^{\pi(6)} \setminus \{\chi_{\pi(5)}^{\pi(6)}b0, \chi_{\pi(4)}^{\pi(6)}0b\}.$$

$$\chi_{\pi(2)}^{\pi(3)}b0 + \psi \rightarrow \chi_{\pi(2)}^{\pi(3)}b1 + \psi, \text{ for all } \psi(v_S) \in \Xi^{\pi(9)} \setminus \{\chi_{\pi(8)}^{\pi(9)}0b, \chi_{\pi(7)}^{\pi(9)}b0\}.$$

// The cell has been colored.

9. $\chi_{\pi(1)}^{\pi(2)}p + \chi_{\pi(2)}^{\pi(3)}11 \rightarrow \chi^{\pi(2)} + \mathcal{O}^{\pi(3)}$. // Announce the termination of local coloring.

10. $\mathcal{O}^{\pi(3)} + \chi_{\pi(2)\pi(3)} \rightarrow \mathcal{O}^{\pi(3)} + \mathcal{O}$.

// Eliminate the redundant states produced by the coloring species.

11. $A^{\pi(1)} + \chi^{\pi(2)} \rightarrow B'^{\pi(1)} + C'^{\pi(2)}$. Where $B' = \begin{cases} B & \text{if } B \in Q^\dagger \\ \chi & \text{if } B = \mathcal{O} \end{cases}$ and $C' = \begin{cases} C & \text{if } C \in Q^\dagger \\ \chi & \text{if } C = \mathcal{O} \end{cases}$.

// Perform the state transition.

As we complete the simulation of growing reactions, we now briefly explain how to simulate the other reactions.

State_Transitions For the reactions that occur on the cells whose blocks have been colored, the simulation is straight forward. For a bimolecular reaction $(A, B, C, D, d) \in R \setminus R_g$, we have to include the reactions on all $A^i, i \in [9]$. Similar to the final step of *Growing_Reactions*, for all $\pi \in \Pi_d$, we add the following reactions to R' :

1. $A^{\pi(1)} + B^{\pi(2)} \rightarrow C'^{\pi(1)} + D'^{\pi(2)}$, where $C' = \begin{cases} C & \text{if } C \in Q^\dagger \\ \chi & \text{if } C = \mathcal{O} \end{cases}$ and $D' = \begin{cases} D & \text{if } D \in Q^\dagger \\ \chi & \text{if } D = \mathcal{O} \end{cases}$.

For a unimolecular reaction $(A, B, \odot)$, the simulation is simple: For all $i \in [9]$, add the following reactions to R' :

1. $A^i \rightarrow B'^i$, where $B' = \begin{cases} B & \text{if } B \in Q^\dagger \\ \chi & \text{if } B = \mathcal{O} \end{cases}$.

Thus far, we have presented the complete simulation of a s-d-sCRN system $\Gamma = (Q, S, R)$ using a s-sCRN system $\Gamma' = (Q', S', R')$ by integrating the aforementioned three parts. Notice that this is a 3-fuzz simulation. The worst case is shown in figure 4.3: A block is colored with no growing reactions happening on it.

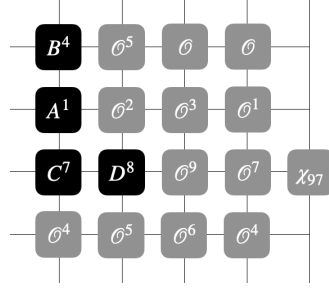


Figure 4.3: 3-fuzz simulation.

The block of O^9 is colored with a garbage species χ_{97} produced.

4.3 Proof

First we give the representation function $\mathcal{R} : Q' \rightarrow Q$. Define $\hat{Q} = \{\sigma^i : \sigma \in Q^\dagger\}$.

Then $\mathcal{R}$ is defined as the following:

- For all $\psi = \sigma^i \in \hat{Q}$, $\mathcal{R}(\psi) = \sigma$.
- $\mathcal{R}(0) = s$.
- For all $\psi \notin \hat{Q} \cup \{0\}$, $\mathcal{R}(\psi) = \mathcal{O}$.

By Definition 4, we need to show that

1. Γ follows Γ' . i.e. $\Gamma \dashv_{\mathcal{R}} \Gamma'$.
2. Γ' models Γ . i.e. $\Gamma' \models_{\mathcal{R}} \Gamma$.
3. Γ' and Γ have equivalent productions. i.e. $\Gamma' \Leftrightarrow_{\mathcal{R}} \Gamma$.

4.3.1 Γ follows Γ'

Lemma 6. $\alpha' \rightarrow^1 \beta'$ for some $\alpha', \beta' \in \mathcal{A}(\Gamma')$ $\implies \mathcal{R}^*(\alpha') \rightarrow \mathcal{R}^*(\beta')$ in finite steps.

Lemma 6 implies $\Gamma \dashv_{\mathcal{R}} \Gamma'$. If we have $\alpha' \rightarrow_{\Gamma'} \beta'$ for some $\alpha', \beta' \in \mathcal{A}(\Gamma')$, there exists a sequence of (possibly infinite) configurations $\{\alpha_i\}$ s.t. $\alpha' = \alpha'_0 \rightarrow^1 \alpha'_1 \rightarrow^1 \dots \rightarrow^1 \beta'$ s.t. $\alpha'_{i-1} \rightarrow^1 \alpha'_i$ for all i . With this lemma we know $\mathcal{R}^*(\alpha'_{i-1}) \rightarrow \mathcal{R}^*(\alpha'_i)$ in finite steps for every i . Therefore, $\mathcal{R}^*(\alpha') \rightarrow_{\Gamma} \mathcal{R}^*(\beta')$, which implies that $\Gamma \dashv_{\mathcal{R}} \Gamma'$.

Proof. of Lemma 6.

Suppose that β' can be produced by applying reactions $r' \in R'$ to α' .

If r' is a unimolecular reaction $A^i \rightarrow B'^i$ for some $A \neq \mathcal{O}$, then there must be a reaction $(A, B, \odot) = (\mathcal{R}^*(A^i), \mathcal{R}^*(B'^i), \odot) \in R$ by our construction. This implies that $\mathcal{R}^*(\alpha') \rightarrow^1 \mathcal{R}^*(\beta')$.

Similarly, if r' is a bimolecular reaction of the form $A^{\pi(1)} + B'^{\pi(2)} \rightarrow C'^{\pi(1)} + D'^{\pi(2)}$ for some $A, B \in Q^\dagger, C, D \in Q$, then there exists a reaction $(A, B, C, D, d) \in R$ where $(A, B, C, D, d) = (\mathcal{R}^*(A^{\pi(1)}), \mathcal{R}^*(B^{\pi(2)}), \mathcal{R}^*(C^{\pi(1)}), \mathcal{R}^*(D^{\pi(2)}), d)$, which implies that $\mathcal{R}^*(\alpha') \rightarrow^1 \mathcal{R}^*(\beta')$.

If r' comes from the simulation process of *Growing_Reactions*, then there exists a growing reaction $(A, \mathcal{O}, B, C, d) \in R_g$, where d corresponds to the permutation π used in r' . In this sub-protocol, a reaction r' induces state transitions in $\mathcal{R}^*(\alpha')$ if and only if $r' \in \{A^{\pi(1)} + \chi^{\pi(2)} \rightarrow B'^{\pi(1)} + C'^{\pi(2)} : A \in Q^\dagger, B', C' \in Q\}$. Since $(\mathcal{R}^*(A^{\pi(1)}), \mathcal{R}^*(\chi^{\pi(2)}), \mathcal{R}^*(B^{\pi(1)}), \mathcal{R}^*(C^{\pi(2)}), d) = (A, \mathcal{O}, B, C, d) \in R_g$, we conclude that $\mathcal{R}^*(\alpha') \rightarrow^1 \mathcal{R}^*(\beta')$. Every other reaction is not going to change the resulting configuration under the mapping $\mathcal{R}^*$, i.e. $\mathcal{R}^*(\alpha') = \mathcal{R}^*(\beta')$.

Otherwise, r' comes from the execution of *Determine_Global_Orientation*. In this case, $\mathcal{R}^*(\alpha') = \mathcal{R}^*(\beta')$ always. □



4.3.2 Γ' models Γ

For every configuration $\alpha \in \mathcal{A}(\Gamma)$, let $\Pi(\alpha) = \mathcal{R}^{*-1}(\alpha) \cap \mathcal{A}(\Gamma')$, which is the set of all reachable configurations in Γ' that are mapped to α . Then it remains to show the following lemma:

Lemma 7. 1. $\Pi(\alpha) \neq \emptyset$.

2. Given any β s.t. $\alpha \rightarrow^1 \beta$, for every $\alpha' \in \Pi(\alpha)$, there exists β' s.t. $\mathcal{R}^*(\beta') = \beta$ and $\alpha' \rightarrow \beta'$ in finite steps.

This indeed implies the original definition of $\Gamma' \models_{\mathcal{R}} \Gamma$. Since every α'' s.t. $\mathcal{R}^*(\alpha'') = \alpha$, α'' is included in $\Pi(\alpha)$. And for any β s.t. $\alpha \rightarrow \beta$, we can find a sequence of (possibly infinite) configurations $\{\alpha_i\}$ s.t. $\alpha = \alpha_1 \rightarrow^1 \alpha_2 \rightarrow^1 \dots \rightarrow^1 \beta$. Then by this lemma, for every $\alpha' \in \Pi(\alpha)$, there exists α'_2 s.t. $\mathcal{R}^*(\alpha'_2) = \alpha_2$ and $\alpha' \rightarrow \alpha'_2$ in finite steps, which implies that $\alpha'_2 \in \Pi(\alpha_2)$. By similar argument, we could show that there exists α'_i s.t. $\mathcal{R}^*(\alpha'_i) = \alpha_i$ and $\alpha'_{i-1} \rightarrow \alpha'_i$ in finite steps for every $i = 3, 4, \dots$. Hence, there exists β' s.t. $\mathcal{R}^*(\beta') = \beta$ and $\alpha' \rightarrow \beta'$.

To prove Lemma 7, we first prove the following lemma.

Lemma 8. For all reachable configuration $\alpha' \in \mathcal{A}(\Gamma')$, α' satisfies the complete coring property.

Notice that Lemma 8 is precisely the property utilized in *Growing_Reactions*.

Proof. Given $\alpha' \in \mathcal{A}(\Gamma')$, we define a set of species $Q_{\text{colored}} = Q_{\text{colored}}^{\dagger} \cup \{\chi^i\}$, and define $V_c = \{u \in \mathbb{Z}^2 : \alpha'(u) \in Q_{\text{colored}}\}$ to be the set of cells that contain some species in Q_{colored} .

Then, requiring α' to satisfy the complete coloring property is equivalent to requiring the following statement to hold for all $u \in V_c$:

At the time u is first occupied by a species in Q_{colored} , $\mathcal{B}(u)$ has already been completely colored.

The reason is that no reaction removes the color from any cell. Therefore, once a block is colored, it remains permanently colored. To prove this statement, we define a strict total order $\prec$ on the set V_c according to the time they were first occupied by a species in Q_{colored} . That is, for all $u, v \in V_c$, $u \prec v$ if and only if v has never been occupied by any species in Q_{colored} at the time u is first occupied by a species in Q_{colored} . The total order has a minimal element, which is the origin $(0, 0)$. This is because the block of $(0, 0)$ is first colored in *Determine_Global_Orientation*, and any reaction in Γ can be simulated only if s^5 has been produced. For every $u \in V_c$, define $V_c^{\prec u}$ to be the set $\{v \in V_c : v \prec u\}$. We complete the proof by the following claim.

Claim 9. For any cell $u \in V_c$, if for every $v \in V_c^{\prec u}$, $\mathcal{B}(v)$ is completely colored, then $\mathcal{B}(u)$ is completely colored.

With this claim, we can see that for every $u \in V_c$, $\mathcal{B}(u)$ must have been completely colored, which implies that α' satisfies the complete coloring property. Otherwise, if there exists some cells that violate the complete coloring property, we can find the minimal cell whose block is not completely colored, contradicting the claim. $\square$

Proof. of Claim 9

Suppose that by applying the reaction $r' \in R'$, cell u transformed into a species in Q_{colored} for the first time. We can observe from our protocol that each species in $Q_{\text{colored}}^\dagger$

appears after either s^5 or some χ^i species. s^5 appears only if its block is colored in the *Determine_Global_Orientation* protocol. For the χ^i species, they must be first produced by the reactions listed in item 2., 4., 9. and 11. in the simulation protocol of growing reactions. We restate them here:

$$2. \chi_{\pi(1)}^{\pi(2)} + \psi^{\pi(3)} \rightarrow \chi^{\pi(2)} + \psi^{\pi(3)}, \text{ for all } \psi \in Q^\dagger \cup \{\chi\}.$$

$$4. \chi_{\pi(1)}^{\pi(2)} + \chi_{\pi(1)}^{\pi(3)} \rightarrow \chi^{\pi(2)} + \chi^{\pi(3)}.$$

$$9. \chi_{\pi(1)}^{\pi(2)}p + \chi_{\pi(2)}^{\pi(3)}11 \rightarrow \chi^{\pi(2)} + \mathcal{O}^{\pi(3)}.$$

$$11. A^{\pi(1)} + \chi^{\pi(2)} \rightarrow B'^{\pi(1)} + C'^{\pi(2)}. \text{ Where } B' = \begin{cases} B, & B \in Q^\dagger \\ \chi, & B = \mathcal{O} \end{cases}, C' = \begin{cases} C, & C \in Q^\dagger \\ \chi, & C = \mathcal{O} \end{cases}.$$

We discuss the four cases reversely.

In item 11., if $C' = \chi$, then it is not the first time the species $\chi^{\pi(2)}$ appears on that cell. If $B' = \chi$, then for all $A \neq s$, $A^{\pi(1)}$ must be transformed from $\chi^{\pi(1)}$; otherwise, $A = s$, then the block of s is already colored. Therefore, the first $\chi^{\pi(2)}$ that appears on a cell, except for the origin, must be the production of the first three reactions. For simplicity, we fix $\pi = id$ in the following description.

In item 9., $\chi_2^3 11$ must have been produced from $\chi_2^3 00$ (item 3.). To understand what happened in between, we have to take a closer look at the behavior of the following 3 kinds of species.

- $\chi_{23} - \chi_{23}$ can only be produced when $\chi_2^3 b_0 b_1$ (for some b_0, b_1 not all 1) reacts with some neighboring blank state $\mathcal{O}$, and from $\chi_2^3 b_0 b_1$ we can trace back to $\chi_2^3 00$, which is activated by χ_1^2 (item 3.). χ_1^2 must be a neighbor of some $A^1 \in Q^\dagger$. Therefore,

the underlying cell of A^1 is in $V_c^{\prec u}$, which implies that $\mathcal{B}(A^1)$ is colored by the assumption. Consequently, any χ_{23} in $\mathcal{B}(\chi_1^2) \setminus \mathcal{B}(A^1)$ will encounter some species in Ξ^5 or Ξ^8 and eventually be transformed into $\mathcal{O}^6, \mathcal{O}^9$ respectively (item 5.). Note that there may be a redundant χ_{23} that appears outside $\mathcal{B}(\chi_1^2 p)$, and it should be transformed back to $\mathcal{O}$ later.

- **$\chi_2^3 b_0 b_1$ and $\chi_1^2 p$** - Let $W = \mathcal{B}(\chi_1^2) \setminus \mathcal{B}(A^1)$. By item 5., $\chi_2^3 b_0 b_1$ transforms $\mathcal{O} \in W$ into χ_{23} , which will eventually become $\mathcal{O}^6$ or $\mathcal{O}^9$. If encountering the species in $\{\chi_{\pi(3)\pi(9)}, \chi_{\pi(4)\pi(5)}, \chi_{\pi(5)\pi(4)}\}$ or $\{\chi_{\pi(3)\pi(6)}, \chi_{\pi(7)\pi(8)}, \chi_{\pi(8)\pi(7)}\}$, $\chi_2^3 b_0 b_1$ determines the appropriate color for each position and assigns them directly to $\mathcal{O}^6$ or $\mathcal{O}^9$ directly. For other χ_{ij} that may appear in its adjacency, they are part of other walls that are being colored, so they eventually transform into $\mathcal{O}^6$ or $\mathcal{O}^9$ as well. Therefore, $\chi_2^3 00$ eventually investigates two colored species on W : one belongs to Ξ^6 and the other belongs to Ξ^9 . Item 7. provides the corresponding reaction for each situation, which always transforms $\chi_2^3 00$ into $\chi_2^3 11$ eventually. $\chi_2^3 11$ serves as a signal that announces the completion of coloring W .

Notice that in our protocol, after χ_1^2 initiate the production of $\chi_2^3 00$ (item 3.), it becomes $\chi_1^2 p$ and then remains inactive, awaiting $\chi_2^3 11$ to transform it into χ^2 (item 9.). Based on the above arguments, $\mathcal{B}(\chi^2)$ is indeed colored. On the other hand, the only reaction that $\chi_2^3 11$ can undergo is also described in item 9.. Therefore, $\chi_2^3 11$ will eventually become $\mathcal{O}^3$, thereby reverting the redundant χ_{23} back to $\mathcal{O}$.

In item 4., we know that χ_1^2 is activated from some species A^1 and χ_1^3 is activated from some species B^1 , then the underlying cells of A^1 and B^1 both belong to $V_c^{\prec u}$. Therefore, $\mathcal{B}(\chi_1^2)$ and $\mathcal{B}(\chi_1^3)$ have both been colored by the assumption. We could transition them

into χ^2, χ^3 simultaneously. In this case, $\mathcal{B}(\chi^2)$ has also been colored.

For the case in item 2., the cell of ψ^3 is also in $V_c^{\prec u}$, so $\mathcal{B}(\psi^3)$ has been completely colored by the assumption. Then by the same argument as in item 4., we know that $\mathcal{B}(\chi^2)$ has been colored.

We've checked all the patterns around u carefully, and the claim follows. $\square$

Now we are ready to prove Lemma 7.

Proof. of Lemma 7.

Recall that for any $\alpha \in \mathcal{A}(\Gamma)$, we let $\Pi(\alpha) = \mathcal{R}^{*-1}(\alpha) \cap \mathcal{A}(\Gamma')$. Then by Lemma 8, for all $\alpha' \in \Pi(\alpha)$, α' satisfies the complete coring property.

Define $\mathcal{A}_0$ to be the set of all possible configurations that could be produced in the *Determine_Global_Orientation* protocol after s^5 appears. That is, a colored block centered at s^5 with possibly some garbage species X, Y, Z . Let $\mathcal{A}(s^5, \Gamma')$ be the set of configurations that are reachable from $\mathcal{A}_0$, which are the reachable configurations after coloring the block of s^5 .

Initially, $\Pi(S) \neq \emptyset$ since $S' \in \Pi(S)$. Now we are going to show that given any β s.t. $\alpha \rightarrow^1 \beta$, for every $\alpha' \in \Pi(\alpha) \cap \mathcal{A}(s^5, \Gamma')$, there is always a sequence of reactions that takes α' to some β' s.t. $\mathcal{R}^*(\beta') = \beta$. With this property, we can also conclude that for every $\alpha \in \mathcal{A}(\Gamma)$ there exists a sequence of reactions that transforms S' to an intermediate configuration $\alpha'' \in \mathcal{A}_0$, and then transforms α'' to some $\alpha' \in \Pi(\alpha)$. Thus $\Pi(\alpha) \neq \emptyset$.

Suppose that β is produced by applying a reaction $r \in R$ to α . If $r \notin R_g$, by the fact that α' is completely colored, we can apply the reactions correspond to r in

State_Transitions to obtain a configuration β' s.t. $\mathcal{R}^*(\beta') = \beta$.

When $r \in R_g$, w.l.o.g. we assume that $r = (A, \mathcal{O}, B, C)$ is applied to a cell v containing A^1 in α' . The cell v' on the East of v now contains a state $\sigma(v')$ colored 2 that is mapped to $\mathcal{O}$. It suffices to show that there exists a sequence of reactions that transforms v' to the state χ^2 . Let $\sigma' = \sigma(v')$. The followings are all the possible values of σ' .

$\sigma' = \chi^2$ - Then it is done.

$\sigma' = \mathcal{O}^2$ - v' will be activated by A^1 and become χ_1^2 .

$\sigma' = \chi_i^2 b_0 b_1$ or χ_{ij} - According to the proof of Lemma 8, σ' will become either $\mathcal{O}^2$ or $\mathcal{O}$ (which is impossible for v' since it has already been colored). It reduces to the previous case.

$\sigma' = \chi_i^2 p$ - According to the proof of Lemma 8, σ' will eventually become χ^2 .

By observation, it simplifies to look at the case $\sigma' = \chi_i^2$. We know that the cells directly above and below v' have been colored, and there is no reaction involving χ_1^2 with any colored species Ξ^5 or Ξ^8 , so we only need to discuss the species ψ on the east of v' .

$\psi = \mathcal{O}, \mathcal{O}^3$ - Then v' transforms into $\chi_1^2 p$. According to the proof of Lemma 8, v' will become χ^2 with its block colored.

$\psi = \chi_i^3 b_0 b_1$ or χ_{ij} - According to the proof of Lemma 8, ψ will become $\mathcal{O}$ or $\mathcal{O}^3$. It reduces to the previous case.

$\psi = \sigma^3$ - $\mathcal{B}(\chi_1^2)$ has been colored, so v' transforms itself into χ^2 .

$\psi = \chi^3$ - According to the proof of Lemma 8, v' transforms itself into χ^2 with its block colored.

$\psi = \chi_i^3 p$ - According to the proof of Lemma 8, ψ will eventually become χ^3 . It reduces to the previous case.

$\psi = \chi_1^3, \chi_9^3, \chi_6^3$ - If $\psi = \chi_1^3$, then both the block of χ_1^2 and χ_1^3 have been colored since α' satisfies the complete coloring property. If $\psi = \chi_9^3, \chi_6^3$, apply the third reaction $\chi_1^2 + \chi_6^3 / \chi_9^3 \rightarrow \chi_1^2 p + \chi_2^3 00$ in our protocol and it reduces to the first case.

We can observe that in all the above cases, we transform v' to χ^2 . Then A^1 together with χ^2 can be transformed into B^1 and C^2 . The proof is complete. $\square$

4.3.3 Γ' and Γ have equivalent productions

First we prove that $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}(\Gamma')\} = \mathcal{A}(\Gamma)$.

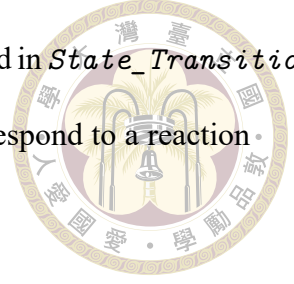
Lemma 10. 1. $\forall \alpha \in \mathcal{A}(\Gamma), \exists \alpha' \in \mathcal{A}(\Gamma') \text{ s.t. } \mathcal{R}^*(\alpha') = \alpha$.

2. $\forall \alpha' \in \mathcal{A}(\Gamma'), \mathcal{R}^*(\alpha') \in \mathcal{A}(\Gamma)$.

The first item implies that $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}(\Gamma')\} \supseteq \mathcal{A}(\Gamma)$, and the second one implies $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}(\Gamma')\} \subseteq \mathcal{A}(\Gamma)$.

Proof. 1. Given α , let $S = \alpha_0 \rightarrow^1 \alpha_1 \rightarrow^1 \dots \rightarrow^1 \alpha$ be any sequence that achieves the configuration α , and assume that α_i is produced by applying reaction r_i to α_{i-1} for all i . Then we construct a sequence of reaction to produce α' . First we use *Determine_Global_Orientation* to get α'_0 . Each r_i corresponds to a sequence of reactions $r'_{i1}, \dots, r'_{im(i)}$ in the simulation. It is obvious that by applying these reaction sequences $(r'_{ik})_{k=1}^{m(i)}$ for $i = 1, \dots, n$, we'll get the resulting configuration α' s.t. $\mathcal{R}^*(\alpha') = \alpha$.

2. The reactions in R' that affect the image of $\mathcal{R}^*$ are those described in *State_Transitions* and item 11. of *Growing_Reactions*. And each of them correspond to a reaction^{*} in R . Therefore, $\mathcal{R}^*(\alpha')$ is achievable in Γ as well.



□

Then it remains to show the following lemma:

Lemma 11. 1. $\alpha \in \mathcal{A}(\Gamma) \setminus \mathcal{A}_*(\Gamma)$ implies $\forall \alpha' \text{ s.t. } \mathcal{R}^*(\alpha') = \alpha, \alpha' \notin \mathcal{A}_*(\Gamma')$.

2. $\forall \alpha \in \mathcal{A}_*(\Gamma), \exists \alpha' \in \mathcal{A}(\Gamma') \text{ s.t. } \mathcal{R}^*(\alpha') = \alpha$.

Proof. 1. $\alpha \in \mathcal{A}(\Gamma) \setminus \mathcal{A}_*(\Gamma)$ means that there exists $r \in R$ s.t. applying r to α produces some $\beta \neq \alpha$. So the statement follows by the analysis in the proof of $\Gamma' \models \Gamma$.

2. $\alpha \in \mathcal{A}(\Gamma)$. By the proof of Lemma 10 item 1, there exists a sequence of reactions $\{r'_i\}$ that produces $\alpha' \in \mathcal{R}^{*-1}(\alpha)$. If there is any reaction sequence that transforms α' to β' s.t. $\mathcal{R}^*(\beta') \neq \mathcal{R}^*(\alpha')$, it corresponds to some reactions in R that transform α to $\beta = \mathcal{R}^*(\beta')$ by the proof of Lemma 10 item 2, Contradicting $\alpha \in \mathcal{A}_*(\Gamma)$.

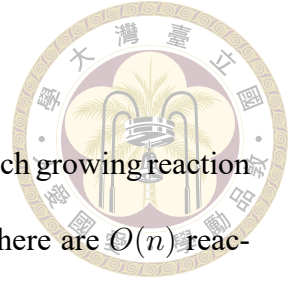
□

4.4 Complexity

Suppose that the unit-seeded directed sCRN system has n species and m reactions. Then the number of species we need in the sCRN system is $|Q'| = 9n + \text{constant} \in O(n)$, see the full description of the state set Q' in Section 4.2. The number of reactions needed is listed below:

- constant in protocol *Determine_Global_Orientation*.
- In *Growing_Reactions*, we have 9 permutations to simulate each growing reaction when the coloring is complete. During the coloring process, there are $O(n)$ reactions when $\chi_{\pi(2)}^{\pi(3)}b_0b_1$ meets some colored species, and $O(n)$ reactions when $x_{\pi(2)\pi(3)}$ recognizes one of its colored neighbor. There are constant kinds of permutations for coloring. As a result there are in total $O(m + n)$ reactions in this part.
- In *State_transitions*, we have 9 permutations to simulate each state transition, which leads to an additional $O(m)$ reaction.

Here $m \in O(n^4)$ since we can choose two reactants and two products to define a reaction. So we need $O(n)$ states and $O(m + n) \in O(n^4)$ reactions for simulation.







Chapter 5 Simulate aTAM by unit-seeded directed sCRN

In this section, we demonstrate that unit-seeded directed sCRN can simulate aTAM.

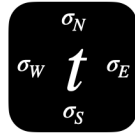
Theorem 12. *Given a system of aTAM $\Gamma = (Q, S, g, \tau)$, there exists a unit-seeded directed sCRN $\Gamma' = (Q', R)$ that simulates Γ .*

5.1 Simulation overview

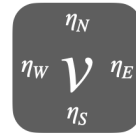
The primary task is to simulate the attachment of a single tile. In particular, we encode each tile t by its state $\sigma(t) = (\sigma_N(t), \sigma_E(t), \sigma_S(t), \sigma_W(t))$. The main challenge arises from the fact that whether a tile is attachable in a position depends on all 4 neighbors. However, in a d-sCRN, the transition of a cell's state is determined by only one of its neighbors. To address this, we introduce a type of special species that are capable of remembering all species around using some auxiliary variables. This process is described in *Observe_Neighbors*. Subsequently, they determine whether there exists any attachable tile in that cell, which is elaborated in *Attachments*.

More precisely, we define two kinds of species: *tile species* and *observing species*, depicted in Figures 5.1a and 5.1b. The set of tile species is of the form $\psi = (\psi_N, \psi_E, \psi_S, \psi_W)$

where ψ corresponds to a tile state in Q . We denote by $\psi(v)$ the tile species located at cell v . For all tile types $\sigma \in Q$, we include all the corresponding tile species $\psi = \sigma$ in Q' . An observing species is denoted by $\eta = (\xi, \eta_N, \eta_E, \eta_S, \eta_W)$, where ξ distinguishes them from the tile species. We use $\eta(v)$ to represent an observing species located at cell v . Any observing species must be adjacent to at least one of the tile species, and $\eta_d(v)$ records the labels facing v in its d direction. That is, let $v = (x_0, y_0)$, then we want $\eta(v)$ to become $(\psi_S(x_0, y_0 + 1), \psi_W(x_0 + 1, y_0), \psi_N(x_0, y_0 - 1), \psi_E(x_0 - 1, y_0))$. Let $\Sigma = \{\iota : \iota \text{ is a label of some } \sigma \in Q\}$ ($\text{null} \in \Sigma$), and let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, then $\eta_D(v) \in \Sigma_\varepsilon^4$. We use ε to represent that the corresponding side has not been observed yet.



(a) Tile species representing tile t .



(b) Observing species.

$$\psi(v) = \sigma(t) = (\sigma_N(t), \sigma_E(t), \sigma_S(t), \sigma_W(t)). \quad \eta(v) = (\eta_N(v), \eta_E(v), \eta_S(v), \eta_W(v)).$$

Figure 5.1: Tile species and observing species.

In the simulation, any tile species ψ could change the neighboring blank species $\mathcal{O}$ to an observing species (ξ, ε_D) . That is, $(\psi, \mathcal{O}, \psi, (\xi, \varepsilon_D), d) \in R'$ for all $d \in \mathcal{D}$. The observing species then start to record the labels in each direction. As soon as it has observed each neighbor at least once, it checks whether there exists a valid tile species that is attachable at cell v . The entire simulation consists of two parts: *Observe_Neighbors* and *Attachments* in Section 5.2. The correctness proof is given in Section 5.3, and a complexity bound is given in Section 5.4.



5.2 Protocols

Observe_Neighbors This protocol let an observing species record each label of its neighbor. If an observing species η has not observed its neighbor v in direction d , then $\eta = (\xi, \langle \varepsilon, \eta_{-d} \rangle)$ for some $\eta_{-d} \in \Sigma_\varepsilon^3$. Let $\psi(v)$ be the state of v , then $\psi(v)$ may be the following three kinds of species:

1. $\psi(v) = \mathcal{O}$. Then η records null in that direction. Therefore, for any $d \in \mathcal{D}$, add the following reactions to R' :

$$((\xi, \langle \varepsilon, \eta_{-d} \rangle), \mathcal{O}, (\xi, \langle \text{null}, \eta_{-d} \rangle), \mathcal{O}, d).$$

2. $\psi(v) = (\xi, \langle \varepsilon, \eta'_{-d-1} \rangle)$ is also an observing species. Then they view each other as a blank species. For any $d \in \mathcal{D}$, add the following reactions to R' :

$$((\xi, \langle \varepsilon, \eta_{-d} \rangle), (\xi, \langle \varepsilon, \eta'_{-d-1} \rangle), (\xi, \langle \text{null}, \eta_{-d} \rangle), (\xi, \langle \text{null}, \eta'_{-d-1} \rangle), d).$$

3. $\psi(v) = \psi_{\mathcal{D}}$ is a tile species. Then η just record the label it sees. For any $d \in \mathcal{D}$, add the following reactions to R' :

$$((\xi, \langle \varepsilon, \eta_{-d} \rangle), \psi, (\xi, \langle \psi_{d-1}, \eta_{-d} \rangle), \psi, d).$$

Otherwise, the observing species η may observe a tile species at v after it has observed a blank species or an observing species at that cell. In this case, we allow η to update its information about v even when it has recorded it as null. Therefore, for any $d \in \mathcal{D}$, we

have the following reactions in R' :

$$((\xi, \langle \text{null}, \eta_{-d} \rangle), \psi, (\xi, \langle \psi_{d-1}, \eta_{-d} \rangle), \psi, d).$$



After the observing species has observed all its neighbors, it perform some “calculation” to see if there is any legal tile species that can attach at its position. The process is described as follows:

Attachments In this protocol, an observing species can check whether a legal attachment exists. An observing species η has recorded all labels it sees in $\eta_{\mathcal{D}} \in \Sigma^4$, for any tile species $\psi = \psi_{\mathcal{D}} \in Q$ satisfies $g(\psi_{\mathcal{N}}, \eta_{\mathcal{N}}) + g(\psi_{\mathcal{E}}, \eta_{\mathcal{E}}) + g(\psi_{\mathcal{S}}, \eta_{\mathcal{S}}) + g(\psi_{\mathcal{W}}, \eta_{\mathcal{W}}) \geq \tau$, we have the following reactions in R' :

$$((\xi, \eta_{\mathcal{N}}, \eta_{\mathcal{E}}, \eta_{\mathcal{S}}, \eta_{\mathcal{W}}), (\psi_{\mathcal{N}}, \psi_{\mathcal{E}}, \psi_{\mathcal{S}}, \psi_{\mathcal{W}}), \odot).$$

Now we summarize the overall simulation protocol.

s-d-sCRN $\triangleright$ aTAM For all $d \in \mathcal{D}$, $\eta_{-d}, \eta'_{-d-1} \in \Sigma_{\varepsilon}^3$, and $\psi = \psi_{\mathcal{D}}$ being a tile species, add the following reactions to R' and all species in Q' :

$$1. (\psi, \mathcal{O}, \psi, (\xi, \varepsilon_{\mathcal{D}}), d).$$

// A tile species turn its neighboring blank species into an observing species.

$$2. ((\xi, \langle \varepsilon, \eta_{-d} \rangle), \mathcal{O}, (\xi, \langle \text{null}, \eta_{-d} \rangle), \mathcal{O}, d).$$

// An observing species meets $\mathcal{O}$.

$$3. ((\xi, \langle \varepsilon, \eta_{-d} \rangle), (\xi, \langle \varepsilon, \eta'_{-d-1} \rangle), (\xi, \langle \text{null}, \eta_{-d} \rangle), (\xi, \langle \text{null}, \eta'_{-d-1} \rangle), d).$$

// Two observing species meet each other.

$$4. ((\xi, \langle \varepsilon, \eta_{-d} \rangle), \psi, (\xi, \langle \psi_{d-1}, \eta_{-d} \rangle), \psi, d).$$

$$((\xi, \langle \text{null}, \eta_{-d} \rangle), \psi, (\xi, \langle \psi_{d-1}, \eta_{-d} \rangle), \psi, d).$$

// An observing species meets a tile species.

$$5. ((\xi, \eta_N, \eta_E, \eta_S, \eta_W), (\psi_N, \psi_E, \psi_S, \psi_W), \odot),$$

$$\text{if } g(\psi_N, \eta_N) + g(\psi_E, \eta_E) + g(\psi_S, \eta_S) + g(\psi_W, \eta_W) \geq \tau. \quad // \text{Tile attachment.}$$

Notice that it is a 1-fuzz simulation, since an observing species may not be able to find any attachable tile species on that position. An example of simulating attachment is provided in Figure 5.2.

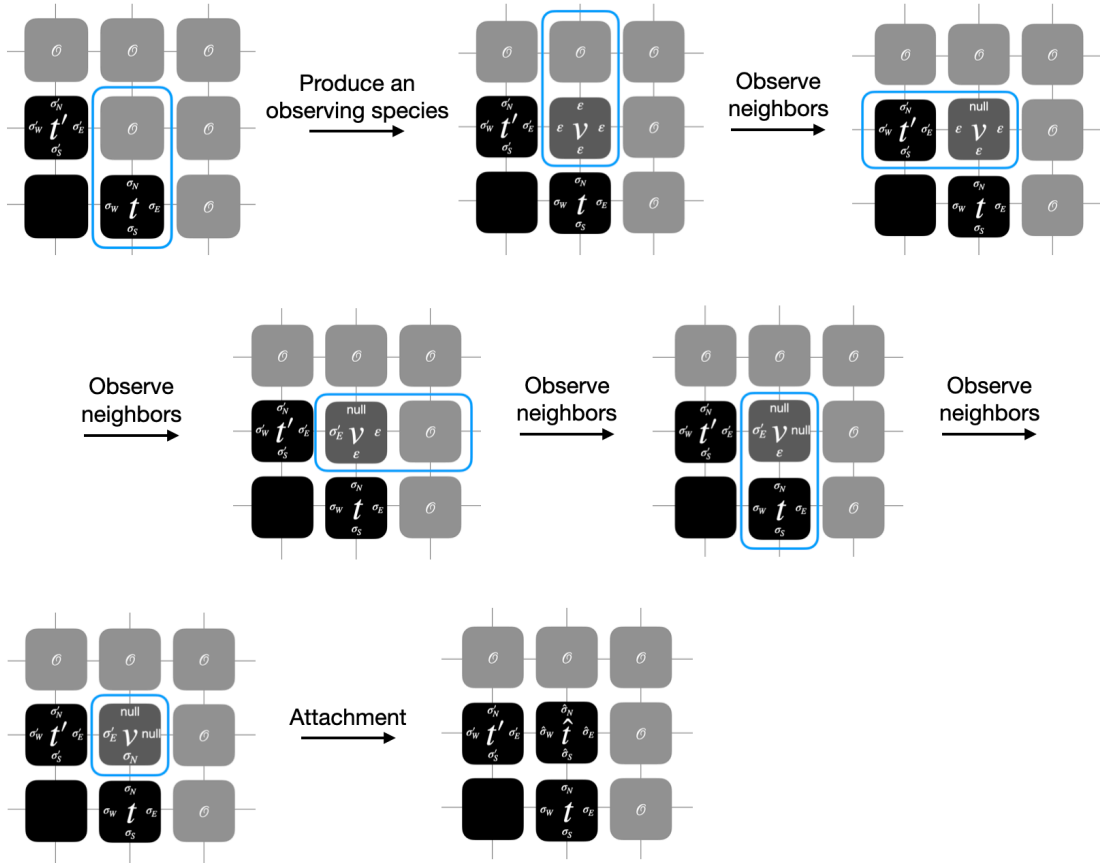


Figure 5.2: A simple example of simulating attachment.

Assuming $g(\hat{\sigma}_W, \sigma'_E) = g(\hat{\sigma}_S, \sigma_N) = 1$ and $\tau = 1$.

An immediate corollary that combine Theorem 5 and Theorem 12 is that aTAM can be simulated by unit-seeded sCRN up to inevitable reflection and rotation, and the simulation is 4-fuzz. In fact, we can make it 3-fuzz if we only allow tile species to initiate the

coloring process. The details are omitted.



5.3 Proof

First, we define the representation function $R : Q' \rightarrow Q$. Let $\hat{Q} = \{\text{tile species}\}$.

Then R is defined as follows:

- For all tile species $\psi \in \hat{Q}$, $\mathcal{R}(\psi) = \psi$.
- For all $\psi \notin \hat{Q}$, $\mathcal{R}(\psi) = \text{null}$.

$\Gamma \dashv_{\mathcal{R}} \Gamma'$ In the entire protocol, the tile species are all produced in the *Attachment* protocol, and each of them appears only if the corresponding tile is attachable to that position. Therefore, for any $\alpha' \rightarrow^1 \beta'$ where $\mathcal{R}^*(\alpha') = \alpha$ and $\mathcal{R}^*(\beta') = \beta$, either $\beta = \alpha$ or $\beta = \alpha + t_{(x_0, y_0)}$ for some tile t attachable at position (x_0, y_0) in the aTAM system Γ .

$\Gamma' \models_{\mathcal{R}} \Gamma$ Let $\Pi(\alpha) = \{\alpha\}$ for all $\alpha \in \mathcal{A}(\Gamma)$. For every $\alpha, \beta \in \mathcal{A}(\Gamma)$ such that $\alpha \rightarrow^1 \beta$ and $\beta = \alpha + t_{(x_0, y_0)}$ for some tile t , we can always produce β from α in system Γ' by placing an observing species in (x_0, y_0) to investigate its neighbors, followed by a unimolecular reaction transforming it into an appropriate tile species. Therefore, from a seed $S' = S$, we can produce any configuration of Γ in Γ' . Then it remains to show that for every $\alpha' \in \Gamma'$ such that $\mathcal{R}^*(\alpha') = \alpha$, $\alpha \rightarrow_{\Gamma'} \alpha'$. Notice that $\alpha' - \alpha$ consists of observing species that retain information about their neighbors. Since the attached tiles cannot fall off, we can create these observing species by sequentially extending the species $(\xi, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$ to the right position and having them collect the labels of a proper subset of their neighbors.

$\Gamma \Leftrightarrow_{\mathcal{R}} \Gamma'$ By the above arguments, it is clear that $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}(\Gamma')\} \neq \mathcal{A}(\Gamma)$. The existence of a valid tile attachment in Γ is equivalent to the existence of a valid transformation from an observing species to a tile species in Γ' , thus we have $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}_*(\Gamma')\} = \mathcal{A}_*(\Gamma)$.

5.4 Complexity

Suppose that the aTAM system has n types of tiles and m attachable combinations of neighborhood. A neighborhood consists of 5 tiles, where one of them can attach to a position surrounded by the other four tiles. Then we have n tile species and $O(n^4)$ observing species (each labels of the observing species records a ε or a label in the aTAM system where number of labels is at most number of types). In terms of reactions, each observing species observes the neighbor in one of the four sides when the corresponding label is still ε or null. So there are $O(n^3 \cdot n) = O(n^4)$ kinds of reactions for *Observe_Neighbors*. There are m combinations of attachable neighborhood, each correspond to an unimolecular reaction in *Attachments*. Notice that m is at most the number of all combinations of neighborhood, which is $O(n^5)$. Therefore, we have in total $O(n^4)$ species and $O(n^4 + m) \in O(n^5)$ reactions. If we use s-sCRN to simulate aTAM, then the complexity bounds are still $O(n^4)$ species and $O(n^4 + m + n) \in O(n^4 + m)$ reactions.





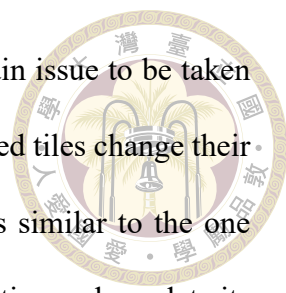
Chapter 6 Simulation between unit-seeded TA with affinity-strengthening rule and unit-seeded directed sCRN

In this section, we prove that unit-seeded directed sCRN and unit-seeded TA with affinity-strengthening rule can simulate each other.

Theorem 13. *Given a unit-seeded tile automata system with affinity strengthening rule $\Gamma = (Q, S, I, g, R, \tau)$, there exists a unit-seeded directed sCRN $\Gamma' = (Q', S', R')$ which simulates Γ .*

6.1 Simulation overview

The configuration changes in the unit-seeded TA result from either state transitions or tile attachments. The simulation of state transitions is straightforward: we simply view them as bimolecular reactions. It is described in Section 6.2, the *State_Transitions*



protocol. The tile attachments are similar to those in aTAM. The main issue to be taken care of is that in the a-sa-TA system, it is allowed that a pair of attached tiles change their states according to the transition rules. Therefore, the simulation is similar to the one used in simulating aTAM, except that the observing species must continuously update its neighbors' information so that it is always possible to record the labels consistent with the current configuration. This is described in *Update_Neighboring_Labels* in Section 6.2. The correctness proof is given in Section 6.3 and the analysis of complexity is given in Section 6.4.

6.2 Protocols

State_Transitions. To simulate the transition rules in R , we simply write each of them as a bimolecular reaction. Thus, we add the following reactions to R' :

$(\psi_1, \psi_2, \psi_3, \psi_4, E)$, for all $(\psi_1, \psi_2, \psi_3, \psi_4, \vdash) \in R$.

$(\psi_1, \psi_2, \psi_3, \psi_4, S)$, for all $(\psi_1, \psi_2, \psi_3, \psi_4, \perp) \in R$.

Update_Neighboring_Labels. To handle the tile attachments, we use the same designing idea as protocol *Observe_Neighbors* in Section 5. This protocol requires the observing species to continuously update their neighbors' labels. There may be tile species performing state transitions after an observing species has recorded its previous state. We slightly modify the *Observe_Neighbors* protocol to make an observing species continuously update its information of neighboring labels on each side, even when all 4 sides have been observed at least once. Notice that now the *tile species* ψ simply encodes a single state in Q . For an observing species η , for all $d \in \mathcal{D}$, all tile species $\psi \neq \eta_d$, we

additionally add the following reactions to R' :

$$((\xi, \eta_N, \eta_E, \eta_S, \eta_W), \psi, (\xi, \langle \psi, \eta_{-d} \rangle), \psi, d).$$



For the entire simulation, the protocol is described below.

s-d-sCRN $\triangleright$ **s-as-TA** For all $d \in \mathcal{D}$, ψ where ψ_i ($i \in [4]$) is a tile species, $\eta_{-d}, \eta'_{-d-1} \in \Sigma_\varepsilon^3$, add the following reactions to R' and all species in Q' :

$$1. (\psi, \mathcal{O}, \psi, (\xi, \varepsilon_{\mathcal{D}}), d).$$

// A tile species turns its neighboring blank species into an observing species.

$$2. ((\xi, \langle \varepsilon, \eta_{-d} \rangle), \mathcal{O}, (\xi, \langle \text{null}, \eta_{-d} \rangle), \mathcal{O}, d).$$

$$((\xi, \langle \varepsilon, \eta_{-d} \rangle), (\xi, \langle \varepsilon, \eta'_{-d-1} \rangle), (\xi, \langle \text{null}, \eta_{-d} \rangle), (\xi, \langle \text{null}, \eta'_{-d-1} \rangle), d).$$

$$((\xi, \langle \varepsilon, \eta_{-d} \rangle), \psi, (\xi, \langle \psi_{d-1}, \eta_{-d} \rangle), \psi, d).$$

$$((\xi, \langle \text{null}, \eta_{-d} \rangle), \psi, (\xi, \langle \psi_{d-1}, \eta_{-d} \rangle), \psi, d).$$

// Behavior of an observing species that adheres to the simulation of aTAM.

$$3. ((\xi, \eta_N, \eta_E, \eta_S, \eta_W), \psi, (\xi, \langle \psi, \eta_{-d} \rangle), \psi, d'). \quad // \text{Updating the neighboring labels.}$$

$$4. ((\xi, \eta_N, \eta_E, \eta_S, \eta_W), (\psi_N, \psi_E, \psi_S, \psi_W), \odot),$$

if $g(\eta_N, \psi_N, \perp) + g(\psi_E, \eta_E, \vdash) + g(\psi_S, \eta_S, \perp) + g(\eta_W, \psi_W, \vdash) \geq \tau$. // Tile attachment.

$$5. (\psi_1, \psi_2, \psi_3, \psi_4, E), \text{ for all } (\psi_1, \psi_2, \psi_3, \psi_4, \vdash) \in R.$$

$$(\psi_1, \psi_2, \psi_3, \psi_4, S), \text{ for all } (\psi_1, \psi_2, \psi_3, \psi_4, \perp) \in R. \quad // \text{State transition.}$$

Similar to the simulation of aTAM, it is a 1-fuzz simulation. And an immediate corollary that combine Theorem 5 and Theorem 13 is that a-sa-TA can be simulated by s-sCRN with inevitable reflection and rotation, and it is a 3-fuzz simulation.

6.3 Proof



First, we give the representation function $R : Q' \rightarrow Q$. Define $\hat{Q} = \{\text{tile species}\}$.

Then R is defined as follows:

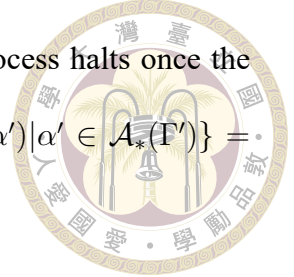
- For all tile species $\psi \in \hat{Q}$, $\mathcal{R}(\psi) = \psi$.
- For all $\psi \notin \hat{Q}$, $\mathcal{R}(\psi) = \text{null}$.

$\Gamma \dashv_{\mathcal{R}} \Gamma'$ A state transition in Γ' corresponds to a bimolecular reaction in Γ' , and the attachments are similar to the proof in the simulation of aTAM. For the cases where an observing species that contains incorrect labels of its neighbors due to state transitions in its neighborhood, it will still transform to a tile species that corresponds to an attachable tile in that position, thanks to the affinity-strengthening condition. Hence $\alpha' \rightarrow_{\Gamma'}^1 \beta' \implies \mathcal{R}^*(\alpha') \rightarrow_{\Gamma}^1 \mathcal{R}^*(\beta')$.

$\Gamma' \models_{\mathcal{R}} \Gamma$ Let $\Pi(\alpha) = \mathcal{A}(\Gamma') \cap \mathcal{R}^{*-1}(\alpha)$. Given α, β s.t. $\beta = \alpha + t_{(x_0, y_0)}$, for any $\alpha' \in \Pi(\alpha)$, the observing species at (x_0, y_0) is always capable of updating all its neighbors with the most current information. Following by an unimolecular reaction, it transforms itself to a tile species corresponding to $t_{(x_0, y_0)}$. The state transitions in the TA system are simulated directly.

$\Gamma \Leftrightarrow_{\mathcal{R}} \Gamma'$ By the above arguments, it is obvious that $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}(\Gamma')\} = \mathcal{A}(\Gamma)$. And the existence of a valid tile attachment in Γ is equivalent to the existence of a valid transformation from an observing species to a tile species in Γ' , allowing the observing species to update the information of its neighbors continually even if there is already a

legal attachment that can be performed. Moreover, the updating process halts once the information is consistent with all its 4 neighbors. So we have $\{\mathcal{R}^*(\alpha') | \alpha' \in \mathcal{A}_*(\Gamma')\} = \mathcal{A}_*(\Gamma)$.



Remark 14. So far, we've proven the correctness of the simulating s-as-TA by s-sCRN. Conversely, Simulating s-sCRN with s-as-TA is quite straightforward. We add a *blank tile* with state $\mathcal{O}$ that is attachable to any other tile. All bimolecular reactions are directly translated into state changes of two adjacent tiles. For unimolecular reactions, we allow the tiles with the corresponding state to perform state changes with any neighboring tiles.

6.4 Complexity

Suppose the TA system has n tiles and m transitions. The state complexity is $O(n^4)$, the same as Section 5.4. The reaction complexity is also similar to Section 5.4, except for additional reactions to update the information of neighbors. Any observing species can update its state when meeting any tile species, this results in $O(n^4 \cdot n) = O(n^5)$ reactions in *Updating_Neighboring_Labels*. Therefore, the reaction complexity is $O(n^5 + m) = O(n^5)$. If we use s-sCRN system to simulate the a-sa-TA system, then the complexity bounds are still $O(n^4)$ states and $O(n^5 + m + n) = O(n^5)$ reactions.





Chapter 7 Simulation between non-deterministic async-CA and directed sCRN

In this section, we show that directed sCRN and non-deterministic async-CA can simulate each other. The simulation of non-deterministic async-CA by directed sCRN is described in Section 7.1, and the simulation of directed sCRN by non-deterministic async-CA is described in Section 7.2. Informally, based on these simulation results, we conclude that directed sCRN and non-deterministic async-CA can accomplish the same tasks.

7.1 Simulate non-deterministic async-CA by directed sCRN

In [14], they suggest a method to emulate the synchronous cellular automata given a coloring initially. Since sCRN is intrinsically asynchronous, we compare sCRN with asynchronous cellular automata through formal simulation.

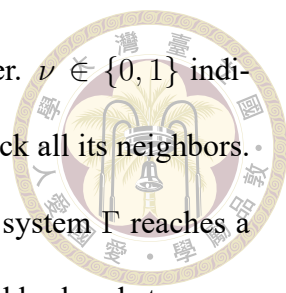
Theorem 15. *Given a non-deterministic, asynchronous CA $\Gamma = (Q, \mathcal{N}, f)$, there exists a directed sCRN $\Gamma' = (Q', S', R')$ which simulates Γ .*

7.1.1 Simulation overview



The local function of cellular automata may depend on the orientation, so it is necessary to simulated it by a d-sCRN rather than an (undirected) sCRN, along with the same initial pattern as in Γ . d-sCRN, which is sCRN with orientation, could be achieved by providing a predefined coloring on the surface at the beginning. To simulate cellular automata, each species must be able to observe and record each of its neighbors, then perform a state transition according to the local function. Similar as the simulation of aTAM in Section 5, we use 4 variables to record neighboring states. One thing to be careful about is that when a species ψ has observed some of its neighbors, we require the those neighbors to keep unchanged until the state transition of ψ is complete. Otherwise, ψ may observe some invalid neighborhood and transition to the wrong state. To avoid this situation, additional 4 variables are introduced to a species to indicate the lock/unlock relation with its neighbors. This is realized in protocol *Observe_and_Lock*. As stated in the *State_transitions* protocol, once a species has locked all its neighbors, it turns into another species that corresponds a different state in the CA system resulting from the local function. Then the species releases all its neighbors, enabling them to engage in reactions with other species. This is described in *Release*.

For further details, we use a species $\psi(v) = (\sigma(v), \sigma_{\mathcal{D}}(v), \kappa_{\mathcal{D}}(v), b, \nu)$ to encode the necessary information for cell v . Here, $\sigma(v)$ is the state of v in the CA system Γ , $\sigma_{\mathcal{D}}(v) = (\sigma_{\mathcal{N}}(v), \sigma_{\mathcal{E}}(v), \sigma_{\mathcal{S}}(v), \sigma_{\mathcal{W}}(v)) \in (Q \cup \{\varepsilon\})^4$ records the states of v 's neighbors, where ε indicates that v has not observed the neighbor on that side yet. $\kappa_{\mathcal{D}}(v) = (\kappa_{\mathcal{N}}(v), \kappa_{\mathcal{E}}(v), \kappa_{\mathcal{S}}(v), \kappa_{\mathcal{W}}(v))$ where $\kappa_d(v) \in \{\varepsilon, 0, 1\}$ for all $d \in \mathcal{D}$ represents whether v is locked by its neighbor in direction d : $\kappa_d(v) = 1$ means v locks that neighbor, 0 means



v is locked by that neighbor, and ε means that no one locks the other. $\nu \in \{0, 1\}$ indicates whether v is a *release species*, where v can do nothing but unlock all its neighbors. To ensure the reaction process ends whenever the cellular automata system Γ reaches a fixed point, we want a cell not to lock its neighbors twice if the neighborhood stays unchanged. Therefore, we set $b = 0$ in the beginning. If the local function f is applied to a neighborhood and no state change occurs for v , we transform v into a *pause species* where $b = 1$. This implies that the activation of v is “paused” until some of its neighbors change. The pause species cannot lock its neighbors. Therefore, if Γ has a reachable fixed point, Γ' is guaranteed to terminate as soon as every cell has executed at most one round of *Observe_and_Lock*.

If the cellular automata system Γ begins with a configuration S , then we set the initial configuration of the sCRN to be S' s.t. $S'(v) = (S(v), \varepsilon_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 0, 0)$ for all $v \in \mathbb{Z}^2$, where $\varepsilon_{\mathcal{D}} = (\varepsilon, \varepsilon, \varepsilon, \varepsilon)$. This means that it has neither observed any of its neighbors nor been locked by them. The basic idea is to have species in the same neighborhood “observe and lock” one another. Roughly speaking, every species tries to lock its neighbors and records their states. Once it has locked all the neighbors successfully, it performs a state transition according to the local function f , and becomes a release species ($\nu = 1$) to unlock all its neighbors. Note that a locked species is not allowed to lock others to avoid deadlocks, and we further allow each lock to be canceled at any time for simplicity. So the protocol mainly consists of two parts: *Observe_and_Lock* and *Release*, stated in Section 7.1.2. The correctness proof is given in Section 7.1.3, and a complexity bound is given in Section 7.1.4.



7.1.2 Protocols

Observe_and_Lock This protocol describes the process for a species to record its neighbors' states and make them unable to change state until being unlocked. In particular, let a pair of species ψ, ψ' be $(\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, \nu, b)$ and $(\sigma', \sigma'_{\mathcal{D}}, \kappa'_{\mathcal{D}}, \nu', b')$ respectively. If ψ , not locked by anyone else, observes an adjacent species ψ' and they are not locked by each other, then we allow ψ to lock ψ' and at the same time record its state. So for any $\sigma, \sigma' \in Q$, $\sigma_{\mathcal{D}}, \sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $\kappa_{-d} \in \{\varepsilon, 1\}^3$, $\kappa'_{-d-1} \in \{\varepsilon, 0, 1\}^3$, $b' \in \{0, 1\}$, and $d \in \mathcal{D}$, we add the following reaction to R' :

$$\begin{aligned} &((\sigma, \langle \sigma_d, \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma'_{d-1}, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), \\ &(\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle 1, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, b'), d) \end{aligned}$$

At any time we allow ψ to unlock ψ' and make it possible to lock other species. For any $\sigma, \sigma' \in Q$, $\sigma_{\mathcal{D}}, \sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $\kappa_{-d}, \kappa'_{-d-1} \in \{\varepsilon, 0, 1\}^3$, $b' \in \{0, 1\}$, and $d \in \mathcal{D}$, we have the following reaction in R' :

$$\begin{aligned} &((\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle 1, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, b'), \\ &(\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), d). \end{aligned}$$

State_Transitions As a species locked all its neighbors, it performs state transition to simulate the local function. Therefore, for all $\sigma, \sigma' \in Q$ and $\sigma_{\mathcal{D}} \in Q^4$ such that $\sigma' = f(\sigma, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{S}}, \sigma_{\mathcal{W}})$, we have the following two cases:

1. If $\sigma' \neq \sigma$, we just apply the local function and turn the species into release state.

Therefore, we add the following reaction to R' :

$$((\sigma, \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 0), (\sigma', \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 1, 0), \odot).$$



2. If $\sigma' = \sigma$, we don't perform any state transition but we turn the species into the pause state, preventing it from locking the same neighborhood again. So we have the following reaction in R' :

$$((\sigma, \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 0), (\sigma', \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 1, 1), \odot).$$

Notice that whenever the pause species has released all its neighbors and investigate any state change among its neighbors, it updates the information and is no longer paused. Therefore, for all $\sigma, \sigma' \in Q$, $\sigma_{\mathcal{D}}, \sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $\kappa_{-d}, \kappa'_{-d-1} \in (\{\varepsilon, 0, 1\})^3$, $b' \in \{0, 1\}$, $d \in \mathcal{D}$, and $\sigma' \neq \sigma_d$, add the following reaction to R' :

$$\begin{aligned} &((\sigma, \langle \sigma_d, \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 1), (\sigma', \langle \sigma'_{d-1}, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, 0), \\ &(\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, 0), d). \end{aligned}$$

we now describe the last part, *Release*.

Release In this protocol, a release species unlocks all its neighbors and exits the release state. We first simulate the unlocking process. For all $\sigma, \sigma' \in Q$, $\sigma_{\mathcal{D}} \in Q^4$, $\sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $\kappa_{-d}, \kappa'_{-d-1} \in (\{\varepsilon, 0, 1\})^3$, $b' \in \{0, 1\}$, and $d \in \mathcal{D}$, we have the following reaction

belong to R' :

$$\begin{aligned} &((\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle 1, \kappa_{-d} \rangle, 1, b), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, b'), \\ &(\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 1, b), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), d) \end{aligned}$$



And then it exits the release state by an unimolecular reaction. Hence, for all $\sigma \in Q$,

$\sigma_{\mathcal{D}} \in Q^4$, $b \in \{0, 1\}$, add the following reaction to R' :

$$((\sigma, \sigma_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 1, b), (\sigma, \sigma_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 0, b), \odot).$$

In conclusion, we summarize the simulation in the following protocol. See Figure 7.1 for a diagram.

d-sCRN $\triangleright$ ***async-CA*** For any $\sigma, \sigma' \in Q$, $\sigma_{\mathcal{D}}, \sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $\kappa_{-d}, \kappa'_{-d-1} \in \{\varepsilon, 0, 1\}^3$, $b' \in \{0, 1\}$, and $d \in \mathcal{D}$, we have the following reactions in R' :

$$\begin{aligned} 1. & ((\sigma, \langle \sigma_d, \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma'_{d-1}, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), \\ & (\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle 1, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, b'), d), \\ & \text{if } \kappa_{-d} \in \{1, \varepsilon\}^3. \end{aligned}$$

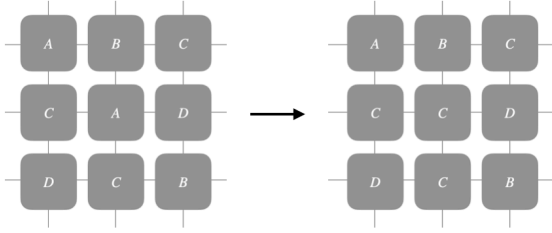
// Lock and observe its neighbors.

$$\begin{aligned} 2. & ((\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle 1, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, b'), \\ & (\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), d). \end{aligned}$$

// Unlock its neighbor whenever it wants.

$$3. ((\sigma, \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 0), (\sigma', \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 1, 0), \odot), \text{ if } \sigma' = f(\sigma, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{S}}, \sigma_{\mathcal{W}}) \text{ and } \sigma \neq \sigma'.$$

// Perform state transition and become release species.

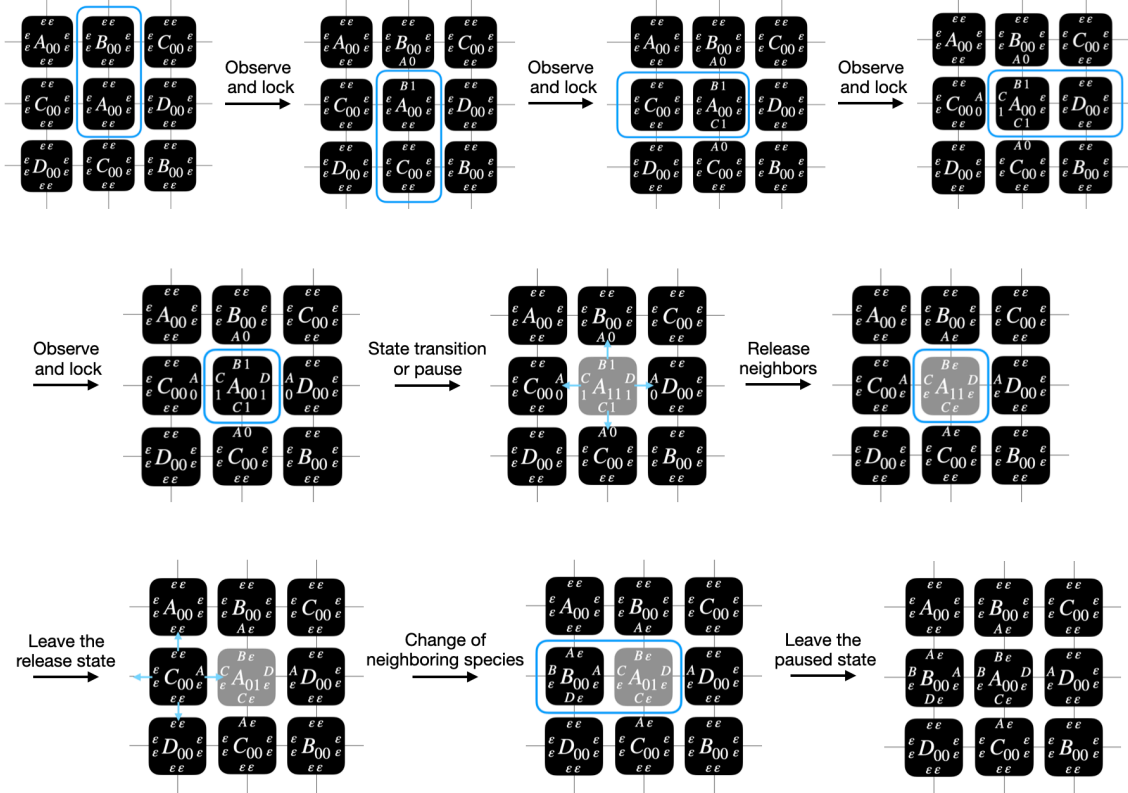


$$\psi(v) = (\sigma(v), \sigma_{\mathcal{D}}(v), \kappa_{\mathcal{D}}(v), \nu, b)$$

$$\begin{matrix} \sigma_N \kappa_N \\ \sigma_W \kappa_W & \sigma_{\nu} b & \sigma_E \kappa_E \\ \sigma_S \kappa_S \end{matrix}$$

(a) Local function $f(A, B, D, C, C) = C$.

(b) A species encoding information of a cell.



(c) Simulating async-CA local function $f(A, B, D, C, C) = A$ by d-sCRN.

Figure 7.1: Simulation of async-CA on sCRN.

4. $((\sigma, \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 0), (\sigma', \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 1, 1), \odot)$, if $\sigma' = f(\sigma, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{S}}, \sigma_{\mathcal{W}})$ and $\sigma' \neq \sigma$.

// Local function maps the neighborhood of σ to itself.

// σ turns itself into the paused release species.

5. $((\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle 1, \kappa_{-d} \rangle, 1, b), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, b'),$

$(\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 1, b), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), d)$, if $\sigma_{\mathcal{D}} \in Q^4$.

// Unlock all its neighbors after applying the local function.

6. $((\sigma, \sigma_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 1, b), (\sigma, \sigma_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 0, b), \odot)$, if $\sigma_{\mathcal{D}} \in Q^4$.

// Exit the release state.

7. $((\sigma, \langle \sigma_d, \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 1), (\sigma', \langle \sigma'_{d-1}, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'),$

$(\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, b'), d)$,

if $\sigma_{\mathcal{D}} \in Q^4$ and $\sigma' \neq \sigma_d$.

// After releasing all its neighbors, some neighbors change their state.

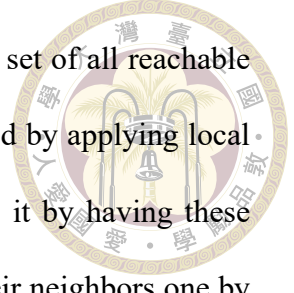
// Then update the information and exit the paused state.

7.1.3 Proof

First, we give the representation function $\mathcal{R} : Q' \rightarrow Q$ as follows:

$$\forall \psi = (\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, b, \nu) \in Q', \mathcal{R}(\psi) = \sigma.$$

$\Gamma \not\vdash_{\mathcal{R}} \Gamma'$ By the same argument as in Section 4.3, Lemma 6, we only need to prove that for any $\alpha', \beta' \in \Gamma'$ such that $\alpha' \rightarrow^1 \beta'$, we have $\mathcal{R}^*(\alpha') \rightarrow^1 \mathcal{R}^*(\beta')$. If it was the reaction “ $((\sigma, \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 0), (\sigma', \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 1, 0), \odot)$ if $\sigma' = f(\sigma, \sigma_{\mathcal{D}})$ and $\sigma \neq \sigma'$ ” in protocol *d-sCRN* $\triangleright$ *async-CA* (item 3) that transformed α' to β' , then $\mathcal{R}(\sigma', \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 1) = \sigma' = f(\sigma, \sigma_{\mathcal{D}}) = f(\mathcal{R}(\sigma, \sigma_{\mathcal{D}}, 1_{\mathcal{D}}, 0, 0), \sigma_{\mathcal{D}})$. This implies $\mathcal{R}^*(\alpha') \rightarrow^1 \mathcal{R}^*(\beta')$. The first component of $\psi \in Q'$ won't change under any other reactions. That is, $\mathcal{R}^*(\alpha') = \mathcal{R}^*(\beta')$.



$\Gamma' \models_{\mathcal{R}} \Gamma$ For any $\alpha \in \mathcal{A}(\Gamma)$, let $\Pi(\alpha) = \mathcal{A}(\Gamma') \cap \mathcal{R}^{*-1}(\alpha)$ be the set of all reachable configurations that are mapped to α . Suppose that α can be produced by applying local function on a sequence of cells in Γ one by one, then we simulate it by having these cells lock their neighbors, perform state transition, and release all their neighbors one by one. It is obvious that the resulting configuration is mapped to α , so $\Pi(\alpha) \neq \emptyset$. Then it suffices to show that for every $\beta \in \mathcal{A}(\Gamma)$ such that $\alpha \rightarrow^1 \beta$ and for every $\alpha' \in \Pi$, there exists a sequence of finitely many reactions in Γ' such that $\alpha' \rightarrow \beta'$ and $\mathcal{R}^*(\alpha') = \alpha, \mathcal{R}^*(\beta') = \beta$. Assume that β can be produced from α by applying local function f at a cell v . If v is occupied by a pause species, then f has no effect on it. Otherwise, species in v 's neighborhood can unlock each other, exit the released state if necessary, and then v can simulate the local function by locking all its neighbors. Therefore, we conclude that $\alpha' \rightarrow \beta'$ for some $\mathcal{R}^*(\beta') = \beta$.

$\Gamma \Leftrightarrow_{\mathcal{R}} \Gamma'$ By the above arguments, it is clear that $\{\mathcal{R}^*(\alpha') : \alpha \in \mathcal{A}(\Gamma')\} = \mathcal{A}(\Gamma)$. For any $\alpha' \in \mathcal{A}_*(\Gamma')$, it must be the case that every cell is in the paused state, i.e. the local function has no effect on the neighborhood of any cell in $\mathcal{R}^*(\alpha')$. So $\mathcal{R}^*(\alpha') \in \mathcal{A}_*(\Gamma)$. On the other hand, for any fixed point $\alpha \in \mathcal{A}_*(\Gamma)$ that is reachable by applying the local function on a sequence of cells, we simulate the transition sequence by having each of the cells successively lock all its neighbors, perform a state transition according to the local function, and then unlock all its neighbors. With an additional round of locking and state transitioning for each cell, species enter the paused state sequentially. Consequently, $\{\mathcal{R}^*(\alpha') : \alpha \in \mathcal{A}_*(\Gamma')\} = \mathcal{A}_*(\Gamma)$ as well.



7.1.4 Complexity

For a CA system with n states and at most m possible outcome of the local function for a given neighborhood, the simulation needs $O(n^5)$ states to record all the combinations of neighborhood. The reaction complexity is therefore $O(mn^5)$ for *State_Transitions* and $O(n^5 \cdot n^5) = O(n^{10})$ for *Observe_and_Lock* and *Release* since any two neighboring cells can update the information of each other. In total there is $O(n^5)$ states and $O((n^5 + m)n^5)$ reactions.

7.2 Simulate directed sCRN by non-deterministic async-CA

Theorem 16. *Given a directed sCRN $\Gamma = (Q, S, R)$, there exists a non-deterministic, asynchronous CA $\Gamma' = (Q', \mathcal{N}, f)$ which simulates Γ .*

7.2.1 Simulation overview

Given a d-sCRN $\Gamma = (Q, S, R)$, we construct a non-deterministic async-CA Γ' with the same initial configuration, where the state of a specific cell is s , and all the other cells are in state $\mathcal{O}$. To simulate a sCRN, we define the local function f by determining the outcome of f for every possible neighborhood. For unimolecular reactions, we let allow a cell to change its state regardless of what its neighbors are. The protocol is given in Section 7.2.2 *Uni_Reactions*. For bimolecular reactions, there exists a fundamental difference between sCRN and cellular automata. In sCRN, two species change their state simultaneously, whereas in cellular automata, typically only one cell changes state at a

time. Therefore, we have to “tie” a pair of adjacent cells together. Let a cell σ non-deterministically pick a neighbor σ' in direction d and “invite” it if there exists a reaction in R that uses σ and σ' as reactants. σ becomes the *invite state* $(\text{inv}, \sigma, \sigma', d)$. After receiving several invitations, σ' non-deterministically selects one of them to “accept” and transitions into an *accept state* $(\text{acc}, \sigma', \varsigma, d')$, which means that it accepts the invitation from a cell with state ς in direction d' . Equivalently, other invitations are viewed as being “rejected”. Then the “invite-accept pair” can change their state according to the reaction respectively. For simplicity we let a cell implicitly lock all of its neighbors as soon as it transitions into an invite state or an accept state. The last step is to tell all the rejected cells to give up and revert to their previous state. The protocol is given in Section 7.2.2 *Bi_Reactions*. See Figure 7.2 for a simple example. The correctness proof is given in Section 7.2.3 and a complexity bound is given in Section 7.2.4.

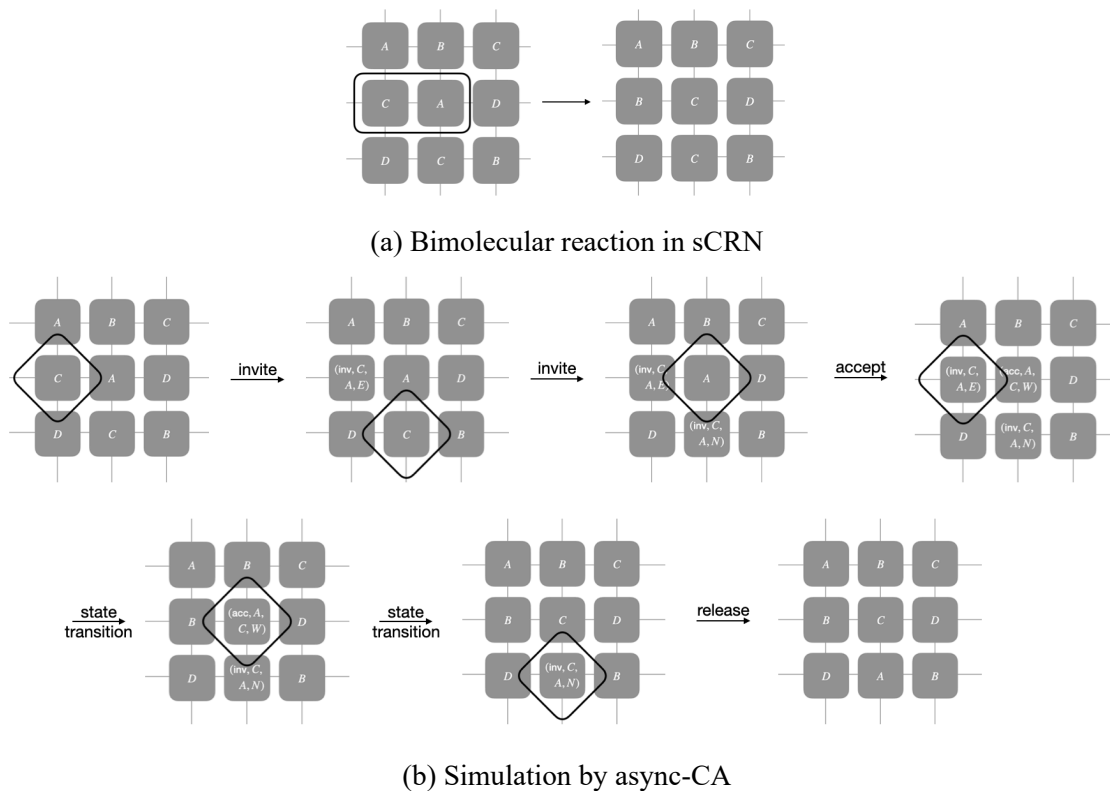


Figure 7.2: Simulating sCRN reaction $C + A \rightarrow B + C$ on non-deterministic async-CA



7.2.2 Protocols

Uni_Reactions This protocol simulates the unimolecular reactions in Γ . First, we let all the states in Q be contained in Q' . For all $\psi, \psi' \in Q$ such that $(\psi, \psi', \odot) \in R$, we add σ' to $f(\psi, \psi_N, \psi_E, \psi_S, \psi_W)$ for every $\psi_D \in Q^4$.

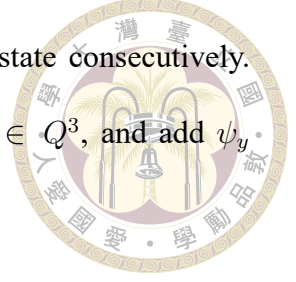
Bi_Reactions To be more precise, for any species $\psi \in Q$ in sCRN, we define the set $nR(\psi, d) = \{\psi' : \exists \psi_x, \psi_y \text{ s.t. } (\psi, \psi', \psi_x, \psi_y, d) \in R\}$ to be all possible species that can react with v in direction d according to R . A cell v non-deterministically picks a direction d . If the neighbor $v' = v + \vec{v}_d$ contains a state representing a species $\psi' \in nR(\psi, d)$, then v changes to an *invite state* $(\text{inv}, \psi, \psi', d)$. On the other hand, v' non-deterministically chooses an invitation to accept, transitions into an *accept state* $(\text{acc}, \psi', \varphi, d')$ (if accepting v then $\varphi = \psi, d' = d^{-1}$), and subsequently the “invite-accept” pair of cells perform their respective state transitions. A state involved in a bimolecular reaction can either invite others or accept the invitations it receives. So we discuss the following two cases:

1. If a state $\psi \in Q$ has a neighbor $\psi_d \in nR(\psi, d)$, ψ could transform into an invite state toward that neighbor. So we add the state $(\text{inv}, \psi, \psi_d, d)$ to $f(\psi, \psi_N, \psi_E, \psi_S, \psi_W)$ for all $\psi_{-d} \in Q^3$.
2. If ψ has been invited by some of its neighbors, it (non-deterministically) picks one of them to accept and transforms into the accept state. Therefore, for any neighbor $\psi_d = (\text{inv}, \psi', \psi, d^{-1})$, we add the state $(\text{acc}, \psi, \psi', d)$ to $f(\psi, \psi_N, \psi_E, \psi_S, \psi_W)$ for all $\psi_{-d} \in (Q \cup \{\text{invite states}\})^3$.

As long as an invite-accept pair $((\text{inv}, \psi, \psi', d), (\text{acc}, \psi', \psi, d^{-1}))$ occurs and there

exists $\psi_x, \psi_y \in Q$ s.t. $(\psi, \psi', \psi_x, \psi_y, d) \in R$, they change their state consecutively.

We add ψ_x to $f((\text{inv}, \psi, \psi', d), \langle (\text{acc}, \psi', \psi, d^{-1}), \psi_{-d} \rangle)$ for all $\psi_{-d} \in Q^3$, and add ψ_y to $f((\text{acc}, \psi', \psi, d^{-1}), \langle \psi_x, \psi'_{-d-1} \rangle)$ for all $\psi'_{-d-1} \in Q^3$.



It remains to deal with those invite states that have not been accepted. If an invite state $(\text{inv}, \psi, \psi', d)$ notices that its neighbor in direction d is not in state ψ' , it can conclude that its invitation has been rejected. So it reverts back to its original state. There may be a situation where ψ' has chosen another state to participate in a bimolecular reaction but remained unchanged itself. In this case, we let $(\text{inv}, \psi, \psi', d)$ to continue inviting ψ' . Therefore, we only need to add the state ψ to $f((\text{inv}, \psi, \psi', d), \psi_N, \psi_E, \psi_S, \psi_W)$ for all $\psi_D \in Q^4$ and $\psi_d \neq \psi'$.

We summarize the above simulation in the following protocol.

async-CA $\triangleright$ d-sCRN For all $\psi, \psi' \in Q$, we construct f by the following protocol:

1. For all $\psi_D \in Q^4$, add the following states to $f(\psi, \psi_N, \psi_E, \psi_S, \psi_W)$:
 - 1.1 σ' , if $(\psi, \psi', \odot) \in R$. // Unimolecular reactions.
 - 1.2 $(\text{inv}, \psi, \psi_d, d)$, if $\psi_d \in nR(\psi, d)$. // Change to invite state.
2. For all $\psi_D \in (Q \cup \{\text{invite states}\})^4$, add the state to $f(\psi, \psi_N, \psi_E, \psi_S, \psi_W)$:
 - 2.1 $(\text{acc}, \psi, \psi', d)$, if $\psi_d = (\text{inv}, \psi', \psi, d^{-1})$. // Change to accept state.
3. For all $(\psi, \psi', \psi_x, \psi_y, d) \in R$:
 - 3.1 Add ψ_x to $f((\text{inv}, \psi, \psi', d), \langle (\text{acc}, \psi', \psi, d^{-1}), \psi_{-d} \rangle)$ for all $\psi_{-d} \in Q^3$.
 - 3.2 Add ψ_y to $f((\text{acc}, \psi', \psi, d^{-1}), \langle \psi_x, \psi'_{-d-1} \rangle)$ for all $\psi'_{-d-1} \in Q^3$.

// Invite-accept pairs performing state transitions to simulate bimolecular reactions.

4. Add ψ to $f((\text{inv}, \psi, \psi', d), \psi_N, \psi_E, \psi_S, \psi_W)$, for all $\psi_D \in Q^4$ and $\psi_d \neq \psi'$.

// Invitation has been rejected.

Remark 17. If we allow the representation function R to map a neighborhood in the cellular automata system to a species in sCRN, then we define R as the following (where $\sigma, \sigma', \sigma_x \in Q$):

- $\mathcal{R}((\text{inv}, \sigma, \sigma', d), \sigma_D) = \sigma$.
- $\mathcal{R}((\text{acc}, \sigma, \sigma', d), \sigma_D) = \sigma$.
- $\mathcal{R}(\sigma_x, \langle (\text{acc}, \sigma', \sigma, d^{-1}), \sigma_{-d} \rangle) = \sigma'$ when the accepted state has not performed a state transition.
- $\mathcal{R}(\sigma, \sigma_D) = \sigma$ for all the remaining cases.

Under this kind of representation function $\mathcal{R}$, we could see that Γ' simulates Γ .

7.2.3 Proof

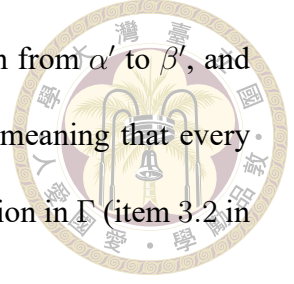
First we give the representation function $\mathcal{R} : Q' \rightarrow Q$.

For all accept state σ_{acc} , $\mathcal{R}(\sigma_{\text{acc}}) = \text{UND}$. For all invite state $\sigma_{\text{inv}} = (\text{inv}, \sigma, \sigma_d, d)$, $\mathcal{R}(\sigma_{\text{inv}}) = \sigma$. Otherwise, $\mathcal{R}(\sigma) = \sigma \forall \sigma \in Q$.

$\Gamma \dashv_{\mathcal{R}} \Gamma'$ For any $\alpha', \beta' \in \Gamma'$ s.t. $\alpha' \rightarrow \beta'$, if $\mathcal{R}^*(\alpha'), \mathcal{R}^*(\beta') \neq \text{UND}$, and $\mathcal{R}^*(\alpha') \neq \mathcal{R}^*(\beta')$, then $\alpha' \rightarrow \beta'$ if and only if at least one of the following holds:

- A unimolecular reaction is simulated directly by applying a local function on a cell (item 1.1 in protocol *async-CA* $\triangleright$ *d-sCRN*).

- Some invite-accept pairs is produced within the transformation from α' to β' , and all the accepted states have undergone their state transitions, meaning that every invite-accept pairs has completely simulate a bimolecular reaction in Γ (item 3.2 in protocol *async-CA* $\triangleright$ *d-sCRN*).



Therefore, for $\alpha' \rightarrow \beta'$ where $\mathcal{R}^*(\alpha') \neq \mathcal{R}^*(\beta')$, $\mathcal{R}^*(\alpha') \rightarrow \mathcal{R}^*(\beta')$ corresponds to a sequence of reactions in the sCRN Γ .

$\Gamma' \models_{\mathcal{R}} \Gamma$ For any $\alpha \in \mathcal{A}(\Gamma)$, let $\Pi = \{\alpha\}$. α is achievable by simulating the unimolecular and bimolecular reactions by item 1.1 and 3.2 respectively. For all α' s.t. $\mathcal{R}^*(\alpha') = \alpha$, there is no accept state in α' , so we can generate all the invite state in an arbitrary order from α since the invite states implicitly lock all of their neighbors. The result follows.

$\Gamma \Leftrightarrow_{\mathcal{R}} \Gamma'$ By the above arguments, it is obvious that $\{\mathcal{R}^*(\alpha') : \alpha \in \mathcal{A}(\Gamma')\} = \mathcal{A}(\Gamma) \cup \{\text{UND}\}$. Notice that if no reactions can be performed on a configuration α , then there exists a configuration α' s.t. $\mathcal{R}^*(\alpha') = \alpha$ where no cells can transform into an invite state, no invite-accept pair exists, and no rejected states occurs, and vice versa. As a result, we have $\{\mathcal{R}^*(\alpha') : \alpha \in \mathcal{A}_*(\Gamma')\} = \mathcal{A}_*(\Gamma)$.

7.2.4 Complexity

For a d-sCRN with n species and m reactions, the simulation needs $O(n^2)$ states for them to invite and accept one another. A fixed neighborhood is associated with at most $O(m)$ possible reactions, so the image size of f for a given neighborhood is $O(m)$.





Chapter 8 Simulation between amoebot and clockwise sCRN

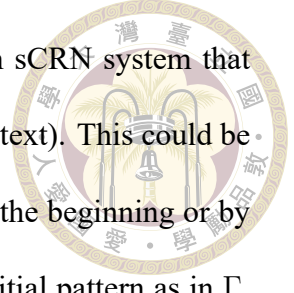
In this section, we show that clockwise sCRN and amoebot can simulate each other. The simulation for amoebot by clockwise sCRN is given in Section 8.1, and the simulation for clockwise sCRN by amoebot is given in Section 8.2

8.1 Simulating amoebot by clockwise sCRN

Theorem 18. *Given an amoebot system $\Gamma = (Q, S, \delta)$, there exists a clockwise sCRN $\Gamma' = (Q', S', R')$ which simulates Γ .*

8.1.1 Simulation overview

In the amoebot model, particles lie on a triangular lattice. This same underlying topology has also been adopted in sCRN [14]. A particle undergoes state transitions and movements based on the flags set by its neighbors. Notice that an algorithm for the amoebot system may result in different asymmetric configurations for clockwise and counter-

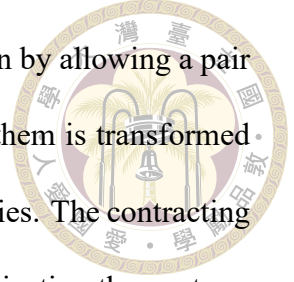


clockwise chirality. Therefore, it is essential to simulate it using an sCRN system that incorporates knowledge of the correct chirality (clockwise in this context). This could be accomplished by providing the surface with a predefined coloring at the beginning or by employing a clockwise sCRN. We now assume that Γ' is given an initial pattern as in Γ . Recall that $Q = \Phi \times \mathcal{O} \times D \times \Sigma^{10}$. The following are two steps to construct the simulation:

- First, we construct a set of states $Q^* = \Phi \times \mathcal{O} \times \mathcal{H} \times \Sigma^{10} \times (\mathcal{O} \cup \{\varepsilon\})$ that can be used to encode each particle in Γ .
- With Q^* , we can construct the reactions R' and state set Q' in Γ' more easily.

We use a single species to represent a contracted particle, which is called the *contracted species*. For an expanded particles that occupies two cells simultaneously, we encode it by a pair of *expanded species*: one *tail species* and one *head species*. They investigate the flags around them and simulate all possible types of movements together. Recall that expansion preserves local orientation. Based on the direction from which a contracted particle expanded when creating the expanded particle, we observe that each expanded particle can be classified into 6 *types*. More details are given in Section 8.1.2.

To simulate a transition rule in the amoebot model, the species in sCRN observe their neighbors' flags first and then perform the corresponding movement. For the observation part, we mainly follow the designing idea in the simulation of cellular automata, which is stated in Section 7.1. We'll briefly describe when to lock or unlock the neighbors and when to enter the pause state for each movement in Section 8.1.3. For the movement part, idle indicates that it is just state transition, and the simulation is straightforward. The protocol is given in *Idle*. For expand_i , it can be simulated by having a contracted species react with a blank species $\mathcal{O}$ in the i direction, while carefully setting the flags in each di-



rection. The protocol is given in *Expansion*. We simulate contraction by allowing a pair of expanded species to react with each other. Subsequently, one of them is transformed into the blank species $\mathcal{O}$, while the other reverts to the contracted species. The contracting process is given in *Contraction*. For handover_i , we simulate by adjusting the contraction and expansion protocols so that the contracted species can remember which of the expanded species it pushed. Subsequently, the expanded and contracted species alternate in performing contraction and expansion operations successively. We define the *switching cell* to be the one occupying by different particles before and after the handover operation. Let v_s be the switching cell. In the simulation, the contracted species transforms v_s to a *prepare species* and transitions itself into a *pushing species*. This makes the pair of expanded species contract out of v_s , leaving v_s with a *waiting species* which waits for the expansion of the pushing species. Details are provided in protocol *Handover*. All the movements are simulated in Section 8.1.4. The correctness proof is given in Section 8.1.5 and the complexity is roughly analyzed in Section 8.1.6.

8.1.2 Encoding particles

Here we explain how to represent a particle in Q by species in Q^* . The labels $f_0, \dots, f_9$ are directly copied from the original particle, so the contracted particles have $f_6 = \dots = f_9 = \varepsilon$. $\mathcal{H} = \{H, T, \varepsilon\}$ indicates whether head or tail the species is, where ε means that it is a contracted species. Recall that in the amoebot system, the initial configuration consists of contracted particles, so each expanded particle is created by some contracted particle performing expand_i , $i \in \mathcal{O}$. The last variable indicate this i , and a contracted species always has an ε there.

We map a contracted particle $\rho_C = (\phi, o, \varepsilon, f_0, \dots, f_5, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$ to a contracted

species $\psi_C = (\phi, o, \varepsilon, f_0, \dots, f_5, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \in Q^*$ naively. And each expanded particle $\rho_E = (\phi, o, d, f_0, \dots, f_9)$ is mapped to a pair of *type- i species*

$$(\psi_T, \psi_H) = ((\phi, o, T, f_0, \dots, f_9, i), (\phi, o, H, f_0, \dots, f_9, i)) \text{ for some } i \in \mathcal{O}.$$

We call ψ_T the tail species, ψ_H the head species, and we also call the corresponding expanded particle the *type- i particle*. Notice that once i and $h \in \mathcal{H}$ are given, the tail direction d is determined as well. See Figure 8.1 for examples of type-0 and type-4 species. The light gray circle represents the tail, and the dark gray circle represents the head. The one with an orange frame means that its local 0 fits the local 0 of the expanded particle. Such a mapping from Q to $Q^* \cup (Q^*)^2$ is denoted as F , where $F(\rho_C) = \psi_C$, $F(\rho_E) = (\psi_T, \psi_H)$.

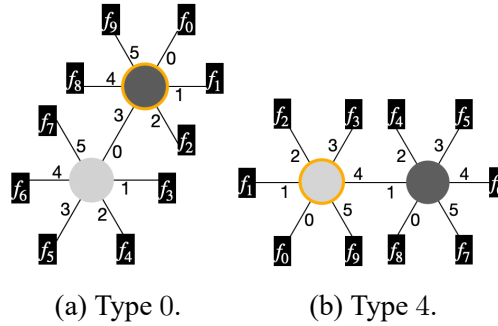
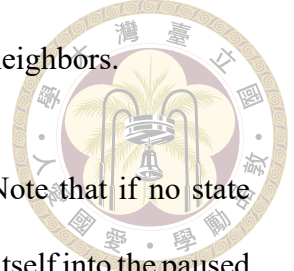


Figure 8.1: Types of expanded species representing expanded particles

8.1.3 Observing flags

Here we explain how to adapt the idea in Section 7.1 to fit the purpose of observing all the flags around a particle. Similarly, the *release species* and the *pause species* are defined. A release species can only unlock its neighbors, and a paused species cannot lock any other species. We allow a species to lock others only if it has not been locked by anyone else, and most of the locks can be cancelled unconditionally except for some cases that we specify in the following description.



For a contracted species ψ_C , it keeps locking and observing its neighbors.

- If all its neighbors have been locked, ψ_C can perform idle. Note that if no state transition is needed to be performed at this point, ψ_C transforms itself into the paused release state.
- If the neighbor in direction i is not locked by any other species, and the rest of its neighbors have all been locked by ψ_C , it can perform either an expansion or a handover toward direction $i \in \mathcal{O}$.
 - If an expansion is performed, two expanded species are produced in the release state.
 - If a handover is performed, a pushing species and a prepare species appear first. A pushing species cannot unlock any neighbors, and the prepare species cannot lock the pushing species or its partner in the expanded species. Then the expanded species contract out of the prepare species as long as all the other neighbors have been locked. This results in a contracted species and a waiting species, where the contracted one is produced in the release state, and the waiting species cannot unlock any species. The last step is to let the pushing species and waiting species simulate the expansion, and both of them become a release state at the same time.

For a pair of expanded species ψ_E , they keep locking and observing all their neighbors except for each other, once the neighbors around them are all locked, they can either idle or perform a contraction.

- If the movement is idle and there is no state transition needed to be performed, then



the species transform themselves into the paused release state.

- If the movement is a contraction, then both the contracted species and blank species resulting from the contraction are produced in the release state.

Following the design when simulating cellular automata, a paused species leaves the pause state if any of its neighbors changes, which includes transitioning into a pushing, prepare, or waiting species. The process described in this section is similar to the one in Section 7.1, so we separate it from the simulation of movements, and we are not giving the technical details or the sketch of proof of this part in this paper. Further, from now on, we assume that every species knows all the flags facing it at any time.

8.1.4 Movements

The goal now remains to simulate any legal movement in Γ . See Figure 8.2 for some examples of these operations.

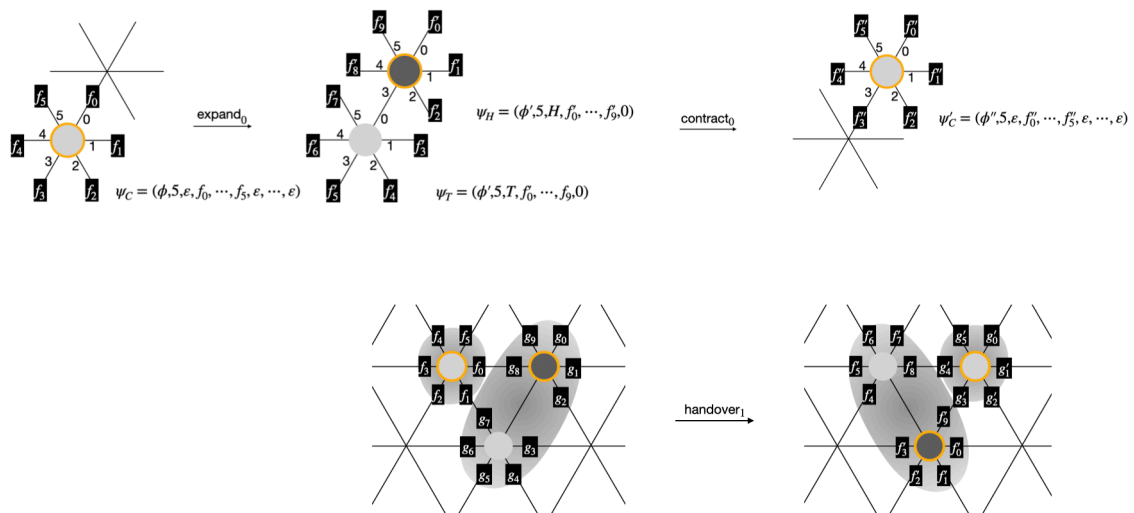


Figure 8.2: Examples of movements

Idle It is just a state transition. Assume that a contracted particle ρ_C becomes ρ'_C by performing idle, and that $\psi_C = F(\rho_C)$, $\psi'_C = F(\rho'_C)$, we add the unimolecular reaction $(\psi_C, \psi'_C, \odot)$ to R' . For an expanded particle ρ_E that becomes ρ'_E , assume that $(\psi_T, \psi_H) = F(\rho_E)$ and $(\psi'_T, \psi'_H) = F(\rho'_E)$, we add the bimolecular reaction $(\psi_T, \psi_H, \psi'_T, \psi'_H, k)$ to R' where k is the type of ρ_E .

Expansion To simulate a contracted particle $\rho_C \in Q$ that expands toward direction i and becomes an expanded particle $\rho_E \in Q$, assuming that $\psi_C = F(\rho_C)$ and $(\psi_T, \psi_H) = F(\rho_E)$, we add $(\psi_C, \mathcal{O}, \psi_T, \psi_H, i)$ to R' .

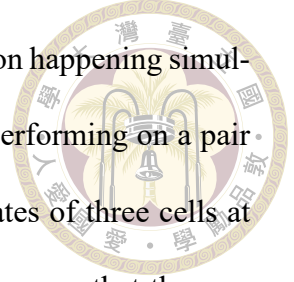
Contraction To simulate a type- k particle performing contract_i , we first see whether it is contracting into the head or the tail.

- Let $C_T = \{(0, 5), (1, 6), (2, 9), (3, 0), (4, 1), (5, 4)\}$. When $(k, i) \in C_T$, a type- k particle performing contract_i is actually contracting into the tail.
- Let $C_H = \{(0, 0), (1, 1), (2, 4), (3, 5), (4, 6), (5, 9)\}$. When $(k, i) \in C_H$, a type- k particle performing contract_i is actually contracting into the head.

Therefore, given an expanded particle $\rho_E \in Q$ which is contracting toward direction i and is turning into a contracted particle $\rho_C \in Q$, we have the following reactions in R' assuming $\psi_C = F(\rho_C)$ and $(\psi_T, \psi_H) = F(\rho_E)$:

$(\psi_T, \psi_H, \psi_C, \mathcal{O}, k)$, for all $(k, i) \in C_T$.

$(\psi_T, \psi_H, \mathcal{O}, \psi_C, k)$, for all $(k, i) \in C_H$.



Handover A handover is equivalent to a contraction and an expansion happening simultaneously. In particular, we view handover_i as $(\text{expand}_i, \text{contract}_j)$ performing on a pair of adjacent particles. In the sCRN, it is impossible to change the states of three cells at the same time, so we use an additional variable $l \in \mathcal{O} \cup \{\varepsilon, w\}$ to ensure that the corresponding contraction and expansion both happen eventually. If $l \in \mathcal{O}$, this means a pushing species intending to push a tail or head species toward the l direction, otherwise, $l = \varepsilon$ represents a prepare species being pushed by a pushing species. When the expanded species has contracted out of the switching cell, l is set to be w that stands for the waiting species.

Assume that it is a contracted particle ρ_C pushing an expanded type- k particle ρ_E toward direction i , and they become ρ'_E and ρ'_C eventually. Let $F(\rho_C) = \psi_C$, $F(\rho'_C) = \psi'_C$, $F(\rho_E) = (\psi_T, \psi_H)$, and $F(\rho'_E) = (\psi'_T, \psi'_H)$. We describe the simulation protocol by the following three parts:

1. We have ρ_C remember i , the direction of the switching cell it is pushing. Let the species located in the switching cell be ψ_h for some $h \in H, T$ (h is fixed given (k, j)), then we have ψ_C become a pushing species and ψ_h become a prepare species (ψ_h, ε) . At this point, (ψ_C, i) and (ψ_h, ε) temporarily “tie” each other. We add the following reaction to R' :

$$(\psi_C, \psi_H, (\psi_C, i), (\psi_H, \varepsilon), i), \text{ for } (k, j) \in C_T.$$

$$(\psi_C, \psi_T, (\psi_C, i), (\psi_T, \varepsilon), i), \text{ for } (k, j) \in C_H.$$

At the same time there may be another contracted particle pushing ρ_E , but these two pushes cannot be performed simultaneously. To avoid this situation we also add the



reverse reactions $(\psi_C, \psi_h, (\psi_C, i), (\psi_h, \varepsilon), i)^{-1}$ to R' .

2. After the prepare species ψ_h appears within the expanded species, the expanded species contracts out of ψ_h , and leaves a mark w on ψ_h , which serves as a signal that tells (ψ_C, i) to perform the expansion. Suppose that ρ_E is a type- k particle, we add the following reactions to R' :

$$(\psi_T, (\psi_H, \varepsilon), \psi'_C, (\psi_H, w), k), \text{ for } (k, j) \in C_T.$$

$$((\psi_T, \varepsilon), \psi_H, (\psi_T, w), \psi'_C, k), \text{ for } (k, j) \in C_H.$$

Note that k is known given ρ_E , which, together with j determines whether head or tail it is contracting into.

3. After (ψ_C, i) observes its neighbor in direction i becoming (ψ_h, w) , it complete the expanding process. So we add the following reaction to R' :

$$((\psi_C, i), (\psi_h, w), \psi'_T, \psi'_H, i).$$

We summarize the simulation of all the movements in the following protocol.

Simulate_Movements_by_c-sCRN For any transition define by δ , we simulate it according to the movement being performed. For all contracted particles $\rho_C, \rho'_C \in Q$, expanded type- k particles $\rho_E, \rho'_E \in Q$, a vector of flags $\vec{f}'' = (f''_0, \dots, f''_9)$ around the particle that is about to move, we assume that $F(\rho_C) = \psi_C$, $F(\rho'_C) = \psi'_C$, $F(\rho_E) = (\psi_T, \psi_H)$, and $F(\rho'_E) = (\psi'_T, \psi'_H)$. For simplicity, let $\delta(\cdot) = \delta(\cdot, \vec{f}'')$. Add the following reactions to R' :

1. $(\psi_C, \psi'_C, \odot)$, if $(\rho'_C, \text{idle}) = \delta(\rho_C)$.

$(\psi_T, \psi_H, \psi'_T, \psi'_H, k), \text{ if } (\rho'_E, \text{idle}) = \delta(\rho_E).$

2. $(\psi_C, \mathcal{O}, \psi_T, \psi_H, i), \text{ if } (\rho_E, \text{expand}_i) = \delta(\rho_C) \text{ for some } i \in \{0, \dots, 5\}.$ // Expansion.

3. $(\psi_T, \psi_H, \psi_C, \mathcal{O}, k), \text{ if } (\rho_C, \text{contract}_i) = \delta(\rho_E) \text{ and } (k, i) \in C_T.$ // Contract into tail.

$(\psi_T, \psi_H, \mathcal{O}, \psi_C, k), \text{ if } (\rho_C, \text{contract}_i) = \delta(\rho_E) \text{ and } (k, i) \in C_H.$ // Contract into head.

4. If $(\hat{\rho}_C, \text{handover}_i) = \delta(\rho_C)$ s.t. $\hat{\rho}_C$ puts the information on its flags that makes ρ_C expand along direction i and become (ρ'_E) , and makes ρ_E contract toward direction j and change to ρ'_C , then we add the following reactions to R' :

$(\psi_C, \psi_H, (\psi_C, i), (\psi_H, \varepsilon)), (\psi_C, \psi_H, (\psi_C, i), (\psi_H, \varepsilon))^{-1}, \text{ for } (k, j) \in C_T.$

$(\psi_C, \psi_T, (\psi_C, i), (\psi_T, \varepsilon)), (\psi_C, \psi_T, (\psi_C, i), (\psi_T, \varepsilon))^{-1}, \text{ for } (k, j) \in C_H.$

$(\psi_T, (\psi_H, \varepsilon), \psi'_C, (\psi_H, w), k), \text{ for } (k, j) \in C_T.$

$((\psi_T, \varepsilon), \psi_H, (\psi_T, w), \psi'_C, k), \text{ for } (k, j) \in C_H.$

$((\psi_C, i), (\psi_h, w), \psi'_T, \psi'_H, i), \text{ for } h \in \{T, H\}.$ // Handover

8.1.5 Proof

First we define the representation function. For any contracted species, it is mapped to the contracted particle it represents, so $R(\psi_C) = F^{-1}(\psi_C)$ for all $\psi_C \in \{\text{contracted species}\}$. For tail and head species in a type- i T-H pair s.t. $F(\rho_E) = (\psi_T, \psi_H)$ for some expanded particle ρ_E , let $\mathcal{R}(\psi_T) = \mathcal{R}(\psi_H) = \rho_E$. This is well-defined since we can decode ρ_E by any one of the ψ_T and ψ_H . For the remaining species, let $\mathcal{R}(\mathcal{O}) = \text{null}$, and $\mathcal{R}((\psi, l)) = \mathcal{R}((\psi, \varepsilon)) = R((\psi, w)) = \text{UND} \forall l \in \mathcal{O}$. The contraction, expansion, or handover, toward each direction, corresponds one-to-one to a protocol in Section 8.1.4.

$\Gamma \dashv_{\mathcal{R}} \Gamma'$ For any $\alpha' \rightarrow \beta'$, it is a sequence consists of contraction, expansion, or a reaction in the protocol simulating handover. If it was the last case, the configuration produced by one step of reaction is not mapped to UND only if it performs “ $((\psi_C, i), (\psi_h, w), \psi'_T, \psi'_H, i)$ ” in protocol *Handover* (item 3), which means that a handover movement has complete.

Therefore, if $\mathcal{R}^*(\alpha'), \mathcal{R}^*(\beta') \neq \text{UND}$, then $\mathcal{R}^*(\alpha') \rightarrow \mathcal{R}^*(\beta')$ since it can be achieved by a sequence of movements.



$\Gamma' \models_{\mathcal{R}} \Gamma$ For any $\alpha \in \mathcal{A}(\Gamma)$, let $\Pi(\alpha) = \{\mathcal{R}^{*-1}(\alpha)\}$. Notice that for any α'' s.t. $\mathcal{R}^*(\alpha'') = \alpha'$ for some $\alpha' \in \{\mathcal{R}^{*-1}(\alpha)\}$, there cannot be any prepare species or waiting species in α'' . So α'' must be a configuration s.t. all sequence of reactions that simulate a single movement is completely performed. Therefore $\alpha \rightarrow \alpha''$.

$\Gamma \Leftrightarrow_{\mathcal{R}} \Gamma'$ By the above explanation, it is clear that $\{\mathcal{R}^*(\alpha') : \alpha \in \mathcal{A}(\Gamma')\} = \mathcal{A}(\Gamma) \cup \{\text{UND}\}$. Notice that the ability of performing a movement in Γ is equivalent to the ability of performing the sub-protocol corresponds to that movement. (Although in the handover operation there might be more than one contracted species pushing the same expanded species, item 2 of protocol *Handover* are reversible reactions, so it will be only one species “accepted” by the expanded species eventually.) Therefore we have $\{\mathcal{R}^*(\alpha') : \alpha \in \mathcal{A}_*(\Gamma')\} = \mathcal{A}_*(\Gamma)$.

8.1.6 Complexity

Suppose an Amoebot system has n kinds of particles (different sets of flags are also taken into account) and at most m outcomes of the transition function δ for each particle with a fixed set of flags it observes. Then we need $O(n)$ species in c-sCRN to encode the particles, and there must be $O(n^2)$ reactions when observing the neighbors. For the movements, we need $O(m)$ reactions to simulate the idle, contract, and expand movements (there may be at most m possible combinations of flags), and we need $O(n^2)$ reactions for handover to lock a pair of contract-expand species before they perform contract and

expand separately. As a result, we have in total $O(n^2 + m)$ reactions for simulation.



8.2 Simulate clockwise sCRN by amoebot

Theorem 19. *Given a clockwise sCRN $\Gamma = (Q, S, R)$, there exists an amoebot system $\Gamma' = (Q', S', \delta)$ which simulate Γ .*

8.2.1 Simulation overview

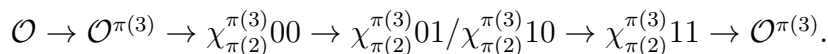
We use only contracted particles, and with the property that each particle can see every neighbors and perform state transition by the information in its neighborhood, we can use the same technique as the simulation of d-sCRN by non-deterministic async-CA (Section 7.2). The complexity to simulate n states and m reactions in c-sCRN systems is $O(n^2)$ states and $O(m)$ image size of the transition function on a fixed particle, where the analysis is also similar to Section 7.2.4. Notice that the underlying lattice does not matter and we don't provide the precise protocol here.



Chapter 9 Simulate unit-seeded, freezing TA with affinity-strengthening rule by unit-seeded, freezing directed sCRN

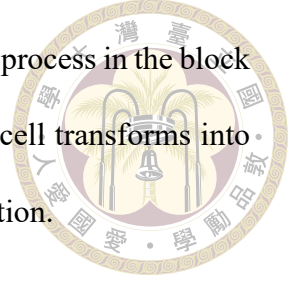
In Section 4, we've shown that in the non-freezing case, the unit-seeded directed sCRN can be simulated by the unit-seeded sCRN up to rotation and reflection. But the simulation is not freezing due to the following three kinds of reactions:

- In the *Determine_Global_Orientation* protocol, we designed the reactions $X/Y/Z \rightarrow \mathcal{O}$ to eliminate the garbage states produced in the coloring process. This may cause the following state transition sequence $\mathcal{O} \rightarrow X/Y/Z \rightarrow \mathcal{O}$ on some cells. These cells revert back into the blank state $\mathcal{O}$, violating the freezing condition.
- In the *Growing_Reactions* protocol, a cell could be repeatedly colored from different directions. In particular, the following reaction sequences are possible:



The blank species is first colored into $\mathcal{O}^{\pi(3)}$, say, by the coloring process in the block

of some species in $\Theta^{\pi(9)}$, and then colored again by the coloring process in the block of some species in $\Theta^{\pi(2)}$. See figure 9.1 for an example. The cell transforms into the colored state more than once and violate the freezing condition.



- Similar to the first point, in the *Growing_Reactions* protocol, we also have the following reaction sequence to eliminate the garbage states:

$$\mathcal{O} \rightarrow \chi_{\pi(i)\pi(j)} \rightarrow \chi_{\pi(2)\pi(3)} \rightarrow \mathcal{O}, \forall (i, j) \in \{(3, 9), (4, 5), (5, 4), (3, 6), (7, 8), (8, 7)\}.$$

These cells revert back into $\mathcal{O}$ more than once and violate the freezing condition.

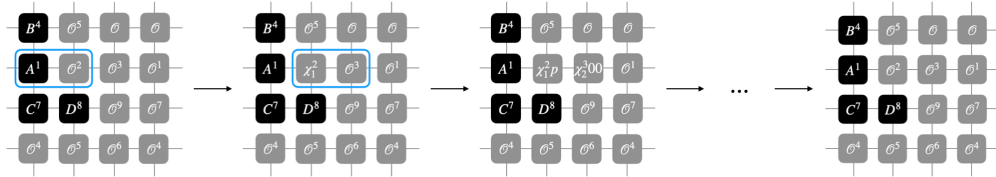


Figure 9.1: Non-freezing sequence of reactions

Essentially, these problems arise due to repeated presence of certain species in a single cell. In fact, each repeated species emerges a constant number of times. Our main idea is to add a counter to each cell that tracks the number of times these repeated species re-emerge.

In this section, we first demonstrate how to adapt the protocol from Section 4 to simulate unit-seeded, freezing directed sCRN using unit-seeded, freezing sCRN up to rotation and reflection. As a corollary, we also show that the unit-seeded, freezing TA with affinity-strengthening rule can be simulated by the unit-seeded, freezing sCRN up to rotation and reflection.

Theorem 20. *Given a unit-seeded, freezing, directed sCRN $\Gamma = (Q, S, R, \preceq_Q)$, there exists a unit-seeded, freezing sCRN $\Gamma' = (Q', S', R', \preceq_{Q'})$ which simulates Γ up to rotation and reflection.*

9.1 Simulation overview



Roughly speaking, given a freezing, directed s-sCRN $\Gamma = (Q, S, R, \preceq_Q)$, we use the protocol in Section 4 to simulate Γ by a non-freezing s-sCRN $\Gamma'' = (Q'', S'', R'')$. Then we show how to modify Γ'' into a freezing s-sCRN $\Gamma' = (Q', S', R', \preceq_{Q'})$ that simulates Γ in Section 9.2.

We've identified every state transition sequence that causes a violation for any partial order at the beginning of this chapter. The violation occurs when a cell transitions into some state twice, and there are 3 kinds of non-freezing sequences of reactions that could potentially occur:

- $\mathcal{O} \rightarrow X/Y/Z \rightarrow \mathcal{O}$. In the *Determine_Global_Orientation* protocol.

- $\mathcal{O} \rightarrow \mathcal{O}^{\pi(3)} \rightarrow \chi_{\pi(2)}^{\pi(3)}00 \rightarrow \chi_{\pi(2)}^{\pi(3)}01/\chi_{\pi(2)}^{\pi(3)}10 \rightarrow \chi_{\pi(2)}^{\pi(3)}11 \rightarrow \mathcal{O}^{\pi(3)}$.

In the *Growing_Reactions* protocol.

- $\mathcal{O} \rightarrow \chi_{\pi(i)\pi(j)} \rightarrow \chi_{\pi(2)\pi(3)} \rightarrow \mathcal{O}, \forall (i, j) \in \{(3, 9), (4, 5), (5, 4), (3, 6), (7, 8), (8, 7)\}$.

In the *Growing_Reactions* protocol.

To deal with these situations, we slightly modify Γ'' and give a new set of states Q' with a corresponding partial order $\preceq_{Q'}$ (write $\preceq$ for simplicity) on Q' . Based on the state set Q'' , the new state set Q' is given by introducing additional states that is treated like the original state in Q'' . More precisely, we introduce a counter $c(v)$ at each cell v . Initially, $c(v) = 0$ for all v . For every reaction that transitions v to $\mathcal{O}$ or $\mathcal{O}^i$ for some $i \in [9]$, we replace it with a reaction that transitions v to $\mathcal{O}(c(v) + 1)$ or $\mathcal{O}^i(c(v) + 1)$, which is essentially adding the counter by 1. We give the protocol and the partial order in

section 9.2, then show that the counter will not exceed a constant number in the proof of Lemma 21. The complexity bound is roughly analyzed in Section 9.3.



9.2 Protocol

In this section, we will construct simulation system Γ' from Γ'' by first giving the set of reactions R' and the corresponding states Q' , then defining a partial order on Q' . We state in Lemma 21 that the counter is guaranteed to be upper bounded by a constant C , so Γ' is indeed a freezing system.

s-sCRN $\triangleright$ *d-s-sCRN* (freezing)

1. For any reaction $A+B \rightarrow C+D$ in R'' , any $c_1, c_2 \leq C$ for some C to be determined, add the following reactions to R' and all the states to Q' :

$$A(c_1) + B(c_2) \rightarrow C(c_1) + D(c_2). \text{ for all } C, D \notin \{\mathcal{O}, \mathcal{O}^i : i \in [9]\},$$

$$A(c_1) + B(c_2) \rightarrow C(c_1) + D(c_2 + 1). \text{ if } D \in \{\mathcal{O}, \mathcal{O}^i : i \in [9]\}.$$

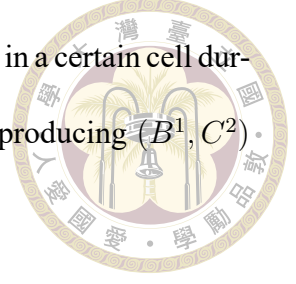
The following lemma is stated

Lemma 21. *In the simulation protocol $s\text{-}s\text{CRN} \triangleright d\text{-}s\text{-}s\text{CRN}$ (freezing), there exists a constant upper bound C on the counter for all cell.*

Before formally define the partial order $\preceq$, we give some observations about $\preceq$:

- For any counter number c, c' , any color $i \in [9]$, we must have $\psi^i(c) \preceq \phi^i(c')$ for all $\psi, \phi \in Q$ s.t. $\psi \preceq_Q \phi$.

- $\preceq$ must be consistent with all the possible orders of state changes in a certain cell during the *Growing_Reactions* process. As an example, when producing (B^1, C^2) from (A^1, O^2) in Γ'' , we can define



1. $O^2(c) \preceq \chi_1^2(c) \preceq \chi_1^2 p(c) \preceq \chi^2(c) \preceq \psi^2(c)$ for all $\psi \in Q$, and
 $O \preceq O^3(c) \preceq \chi_2^3 00(c) \preceq \chi_2^3 b_0 b_1(c) \preceq \chi_2^3 11(c) \preceq O^3(c+1)$, for $b_0 \oplus b_1 = 1$.
2. $\chi_1^2 \preceq \chi_8^2 00$, $\chi_1^2 \preceq \chi_5^2 00$, and $\chi_6^3 \preceq \chi_2^3 00$, $\chi_9^3 \preceq \chi_2^3 00$.
3. $O(c) \preceq \chi_{23}(c) \preceq O(c+1)$.

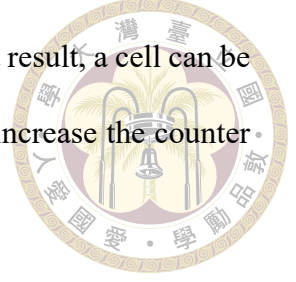
Therefore, we define the partial order $\preceq$ as the following:

1. For any $\psi, \phi \in Q$, any $i \in [9]$ and $c, c' \leq C$, we have $\psi^i(c) \preceq \phi^i(c')$ if $\psi \preceq_Q \phi$.
2. For all $i, j \in [9]$, $\psi \in Q$, we have $O^i(c) \preceq \chi_j^i(c) \preceq \chi_j^i p(c) \preceq \chi^i(c) \preceq \psi^i(c)$, and
 $O(c) \preceq O^i(c) \preceq \chi_j^i 00(c) \preceq \chi_j^i b_0 b_1(c) \preceq \chi_j^i 11(c) \preceq O^i(c+1)$ for $b_0 \oplus b_1 = 1$.
3. $\chi_k^i \preceq \chi_j^i 00$ for all $i, j, k \in [9]$.
4. $O(c) \preceq \chi_{jk}(c) \preceq O(c+1)$ for all $j, k \in [9]$.

proof of Lemma 21. We discuss the three kinds of non-freezing sequences in Γ'' mentioned in section 9.1, and point out when we add the counter by 1 in Γ' .

- Protocol *Determine_Global_Orientation* is only executed once in the simulation, so X, Y, Z could only occurs once as well. This increases the counter by at most 1.
- Each time the reaction $O^{\pi(3)} + \chi_{\pi(2)\pi(3)} \rightarrow O^{\pi(3)} + O$ is performed and cell v is turned back into O , it means that a *wall* of $\mathcal{B}(v)$ is colored. So v will not become

χ_{ij} again because of the local coloring from this direction. As a result, a cell can be turned into state χ_{ij} for some $i, j \in [9]$ at most 4 times. This increase the counter by at most 4.



- Each time the reaction $\chi_{\pi(1)}^{\pi(2)}p + \chi_{\pi(2)}^{\pi(3)}11 \rightarrow \chi^{\pi(2)} + \mathcal{O}^{\pi(3)}$ is executed and cell v is transformed back into $\mathcal{O}^{\pi(3)}$, it means that one of its neighbor transitions into state $\chi^{\pi(2)}$, waiting for an attachment. This can occurs at most once from each direction, thus increase the counter at v by at most 4.

In total, the counter will not exceed 9. Additionally, we got a state set Q'' that is at most 10 times larger than Q' , which is just a constant scale.

□

Notice that when simulating unit-seeded TA with affinity-strengthening rule by unit-seeded directed sCRN in Section 6, we did not utilize any extra repeated states in sCRN. If the s-as-TA system is equipped with a partial order, then following the order in s-d-sCRN system results in a freezing simulation. Specially, for two observing species η and η' , $\eta \preceq \eta'$ if there exists $d \in \mathcal{D}$ s.t. $\eta_d \preceq \eta'_d$ in the tile system and $\eta_{-d} = \eta'_{-d}$. The height of this partial order remains $O(k)$. Details are omitted. As a corollary, we have:

Corollary 22. *Given an freezing, unit-seeded Tile Automata system with affinity strengthening rule $\Gamma = (Q, S, I, g, R, \tau)$, there exists a freezing, unit-seeded directed surface CRN $\Gamma' = (Q', S', R')$ which simulates Γ . Further, there exists a freezing, unit-seeded surface CRN $\Gamma'' = (Q'', S'', R'')$ which simulates Γ up to rotation and reflection as well.*

Remark 23. On the other hand, Simulating freezing s-sCRN with freezing s-as-TA follows directly from *Remark 14*.

9.3 Complexity



Suppose that the freezing s-as-TA system has n states, m transition rules, and the height (the cardinality of the maximum chain) of the partial order is k . The complexity bound follows from the analysis of Section 6.4 with some extra cost due to the counter introduced in Section 9.2. The counter increases at most constant times, which gives a constant scale in the states, reactions, and the height of the partial order. Therefore, we have in total $O(n^4)$ states with a height $O(k)$ partial order, and $O(n^5)$ reactions.





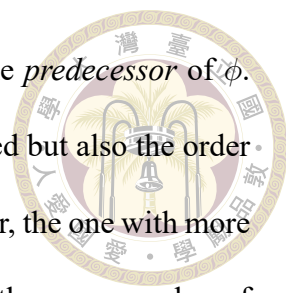
Chapter 10 Simulate freezing

async-CA by freezing

directed sCRN

In Section 7.1, we've shown that in the non-freezing case, the non-deterministic async-CA can be simulated by the directed sCRN given the initial pattern. In the simulation protocol, there are reversible reactions when species are trying to lock and observe their neighbors. More precisely, in Section 7.1.2, all species non-deterministically lock and unlock their neighbors and then proceed with a state transition if some species happen to have locked all its neighbors by chance. It is obviously non-freezing since a cell could revert to a particular state infinitely many times. Therefore, merely adding a counter as described in Section 9 will not work.

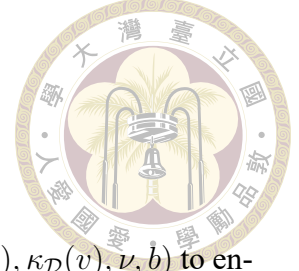
What if we remove all the reversible reactions? Then a cell could possibly be locked indirectly by other species that are arbitrarily far apart. We illustrate this scenario in Figure 10.1. Therefore we need to design a more complicated counter that is influenced only by cells within a constant distance. Our main idea is to add a counter and four variables to a cell that record the order in which its neighbors are locked. By encoding information in these variables, we can control how the counter increments and thus determine the



ϕ the *successor* of ψ if ϕ is locked by ψ . Conversely, ψ is called the *predecessor* of ϕ . Roughly speaking, each species remembers not only who it has locked but also the order in which they became its successor. When two species meet each other, the one with more successors at that moment locks the other one. Otherwise, they have the same number of successors, then they non-deterministically decide which one should be locked. Note that a locked species is not allowed to lock others to avoid deadlocks. By these rules we make sure that the predecessor locks more species than any of its successor. As a result, some species with 4 successors will eventually be generated within a bounded region.

A species with 4 successors (i.e. it has locked all its neighbors) performs state transition according to the local function. Notice that now the state transition is either a state change in the CA system or a transition into a pause species. After the state change is completed, it transitions into a release species, unlocks each of its successors, and transform them into release species as well. We let this “release signal” propagate from predecessors to successors to make the locked species unrestrained again. However, in the freezing system they cannot revert back into the original species, and we resolve this problem by adding a counter to each cell, similar to the method used in section 9.2. Therefore, we’ll first construct a (not necessarily freezing) d-sCRN system (Q'', S', R'') , and then use it to design the freezing d-sCRN system $(Q', S', R', \preceq_{Q'})$. Details of the protocol are described in section 10.2, the correctness proof is given in section 10.3, and the complexity bound is roughly analyzed in Section 10.4.

10.2 Protocol



In Section 7.1, we use a species of this form $\psi(v) = (\sigma(v), \sigma_{\mathcal{D}}(v), \kappa_{\mathcal{D}}(v), \nu, b)$ to encode the information needed for cell v . Here we introduce a new variable $l(v)$ to count the number of v 's successor, so each species has the form $\psi(v) = (\sigma(v), \sigma_{\mathcal{D}}(v), \kappa_{\mathcal{D}}(v), l(v), \nu, b)$. Similar to Section 7.1, $\sigma(v)$ is the state of v in the CA system Γ , $\sigma_{\mathcal{D}}(v) = (\sigma_{\mathcal{N}}(v), \sigma_{\mathcal{E}}(v), \sigma_{\mathcal{S}}(v), \sigma_{\mathcal{W}}(v)) \in (Q \cup \{\varepsilon\})^4$ records the states of v 's neighbors where ε means that v has not observed the neighbor on that side yet. $\kappa_{\mathcal{D}}(v) = (\kappa_{\mathcal{N}}(v), \kappa_{\mathcal{E}}(v), \kappa_{\mathcal{S}}(v), \kappa_{\mathcal{W}}(v)) \in \{\varepsilon, 0, 1, 2, 3, 4\}$ represent the “relation” between v and its neighbors in the corresponding directions, where ε indicates they aren't locked by each other, 0 indicates v is locked, and $1, \dots, 4$ denote the order in which each neighbor became v 's successor. We say that ψ is a *free species* if it is not locked by any other species. $\nu \in \{0, 1\}$ indicates whether v is a *release species*, which initiates the process of unlocking all its neighbors and simultaneously transitions them into release state. To ensure the reaction process terminates when the cellular automata system Γ reaches a fixed point, we want a cell not to re-lock its neighbors if the neighborhood remains unchanged. Set $b = 0$ initially, if the local function f is applied to v ' neighborhood without changing its state, we transform v into a *pause species* where $b = 1$. This implies that the activation of v is “paused” until some of its neighbors change. The pause species cannot lock its neighbors. Therefore, if Γ has a reachable fixed point, Γ' is guaranteed to terminate as soon as every cell has undergone at most one state transition, become a pause species, and released all its successors.

If the cellular automata system Γ begins with a configuration S , then we set the initial configuration of the sCRN to be S' s.t. $S'(v) = (S(v), \varepsilon_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 0, 0, 0)$ for all $v \in \mathbb{Z}^2$, where $\varepsilon_{\mathcal{D}} = (\varepsilon, \varepsilon, \varepsilon, \varepsilon)$. This means that it has neither observed any of its neighbors nor been

locked by them. The protocol consists of three parts: *Lock_and_Observe* (*freezing*), *State_Transitions* and *Release* (*freezing*). First, in *Lock_and_Observe* (*freezing*), we let each of the species interact with its neighbors by comparing the numbers of their successors to decide which one should be locked by the other. So the protocol proceed as follows:

***Lock_and_Observe* (*freezing*)** When a free species $\psi = (\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, l, \nu, b)$ observes another species $\psi' = (\sigma', \sigma'_{\mathcal{D}}, \kappa'_{\mathcal{D}}, l', \nu', b')$, we allow ψ to lock ψ' if ψ' has not been locked by ψ , and ψ has the larger or equal amount of successors than ψ' . So for any $\sigma, \sigma' \in Q$, $\sigma_{\mathcal{D}}, \sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $d \in \mathcal{D}$, $\kappa_{-d} \in \{\varepsilon, 1, \dots, l\}^3$, $\kappa'_{-d-1} \in \{\varepsilon, 0, \dots, l'\}^3$, $l \geq l'$, and $\nu', b' \in \{0, 1\}$, we add the following reaction to R'' :

$$((\sigma, \langle \sigma_d, \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, l, 0, 0), (\sigma', \langle \sigma'_{d-1}, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, l', \nu', b'), \\ (\sigma, \langle \sigma', \sigma_{-d} \rangle, \langle l+1, \kappa_{-d} \rangle, l+1, 0, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, l', \nu', b'), d)$$

Notice that we have $l > l'$ whenever ψ' is locked by ψ .

Once a species ψ has locked all the neighbors successfully ($\kappa_{\mathcal{D}} \in S_4$ where S_4 denotes the set of all permutations on $\{1, 2, 3, 4\}$), it performs a state transition according to the local function f and becomes a release species ($\nu = 1$). We describe this in the following protocol.

State_Transitions For all $\sigma, \varsigma \in Q$, $\sigma_{\mathcal{D}} \in Q^4$ such that $\varsigma = f(\sigma, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{S}}, \sigma_{\mathcal{W}})$, and all $\kappa_{\mathcal{D}} \in S_4$, we have the following two cases:

1. If $\varsigma \neq \sigma$, we just apply the local function and turn the species into release state.

Therefore, we add the following reaction to R'' :

$$((\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, 4, 0, 0), (\varsigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, 4, 1, 0), \odot).$$



2. If $\varsigma = \sigma$, then there is no state transition in the CA system. So we turn the species into the pause state to prevent it from locking the same neighborhood again. Therefore, for all $\kappa_{\mathcal{D}} \in S_4$ we have the following reaction in R'' :

$$((\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, 4, 0, 0), (\varsigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, 4, 1, 1), \odot).$$

A release species, if not locked by any of its neighbors, starts to unlock each successors and at the same time turns them into release species. So the “release signal” propagates to all the species that directly or indirectly locked by ψ and makes them unlocked. The details are provided in protocol *Release (freezing)*.

Release (freezing) In this protocol, we pretend that a species can be unlocked and turn back to the state before being locked for a moment (actually it can't due to the freezing constraint), and this problem will be resolved later.

Now a release species ψ , not locked by anyone else, is going to unlock its successors in the reverse order in which they became its successor, and then exits the release state. For all $\sigma, \sigma', \varsigma \in Q$, $\sigma_{\mathcal{D}} \in Q^4$, $\sigma'_{\mathcal{D}} \in (Q \cup \{\varepsilon\})^4$, $l > l'$, $d \in \mathcal{D}$, $\kappa_{-d} \in (\{\varepsilon, \dots, l-1\})^3$, $\kappa'_{-d-1} \in (\{\varepsilon, 0, \dots, l'\})^3$, and $\nu' \in \{0, 1\}$, there are some cases to discuss with respect to different pairs of (b, b') :

1. If $b' = 0$, i.e. ψ' is not in the paused state. Then we unlock ψ' directly and turn it

into the release state, so we have the following reaction in R'' :

$$\begin{aligned} &((\varsigma, \langle \sigma', \sigma_{-d} \rangle, \langle l, \kappa_{-d} \rangle, l, 1, b), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, l', \nu', b'), \\ &(\varsigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, l-1, 1, b), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, l', 1, b'), d). \end{aligned}$$



2. When $b' = 1$, ψ' is in the paused state.

- When $(b, b') = (1, 1)$, both of them have already in the pause states. It must be the case that ψ' performs state transition earlier than ψ and it has already unlocked all its successors. Subsequently, ψ entered the paused state after the current state transition. This means that $\varsigma = \sigma, l' = 0, \kappa'_D \in \{\varepsilon, 0\}^4$, and none of them can exit the paused state now. We add the following reaction to R'' to unlock ψ' :

$$\begin{aligned} &((\varsigma, \langle \sigma', \sigma_{-d} \rangle, \langle l, \kappa_{-d} \rangle, l, 1, 1), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, l', \nu', 1), \\ &(\varsigma, \langle \sigma', \sigma_{-d} \rangle, \langle \varepsilon, \kappa_{-d} \rangle, l-1, 1, 1), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, l', 1, 1), d). \end{aligned}$$

- When $(b, b') = (0, 1)$. Consider the scenario where a pause species ψ' has released all its successors, and now detects a state change of its neighbor ψ . It updates the information and is no longer paused. Notice that this should happen when ψ is unlocking ψ' after the state transition of ψ . Therefore, for all $\kappa'_{-d-1} \in \{\varepsilon, 0\}^3$, and $\sigma \neq \sigma'_{-d-1}$, add the following reaction to R'' :

$$\begin{aligned} &((\sigma, \langle \sigma_d, \sigma_{-d} \rangle, \langle l, \kappa_{-d} \rangle, l, 1, 0), (\sigma, \langle \sigma'_{-d-1}, \sigma'_{-d-1} \rangle, \langle 0, \kappa'_{-d-1} \rangle, 0, \nu', 1), \\ &(\sigma, \langle \sigma, \sigma_{-d} \rangle, \langle \varepsilon, \kappa'_{-d} \rangle, l-1, 1, 0), (\sigma', \langle \sigma, \sigma'_{-d-1} \rangle, \langle \varepsilon, \kappa'_{-d-1} \rangle, 0, 1, 0), d). \end{aligned}$$

Then, ψ exits the release state by performing an unimolecular reaction when it has no successors. Hence, for all $\sigma \in Q$, $\sigma_{\mathcal{D}} \in Q^4$, $b \in \{0, 1\}$, add the following reactions to R'' :

$$((\sigma, \sigma_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 0, 1, b), (\sigma, \sigma_{\mathcal{D}}, \varepsilon_{\mathcal{D}}, 0, 0, b), \odot).$$

To complete the protocol, it remains to address the scenario that species are unlocked and revert to certain repeated states in the *Release* protocol, violating the freezing constraint. In particular, when a species is unlocked, we need to transform it into a new species that represents the same state in the CA system. For every cell v , we associate a counter $c(v)$. Each time the species $\psi(v)$ at v is unlocked, we add the counter $c(v)$ by 1. If $\psi(v)$ is unlocked, there must be some species performing state transition within a bounded region of diameter at most 7. This is due to the strictly monotonic nature of the locking numbers along a path. In the freezing system, the number of state transitions within a constant distance is at most a constant scale of the height k of the partial order, so every counter will never exceed Mk for some constant M . For any reaction $A + B \rightarrow C + D$ in R'' , any $c_1, c_2 \leq Mk$, we add the following reactions to R' :

$$1. A(c_1) + B(c_2) \rightarrow C(c_1) + D(c_2) \quad // \text{ There is no species being unlocked.}$$

$$2. A(c_1) + B(c_2) \rightarrow C(c_1) + D(c_2 + 1)$$

// if $A + B \rightarrow C + D$ is the reaction that makes A unlock B and produces C, D ,

// i.e. all the reactions in *Release* (*freezing*).

We complete the construction by including all the species to Q' . One can see Figure 10.2 for an example of the simulation process.

The last thing to do is to give a partial order $\preceq$ on Q' . For two species $\psi(c), \psi'(c') \in Q'$ s.t. $\psi = (\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, l, \nu, b)$ and $\psi' = (\sigma', \sigma'_{\mathcal{D}}, \kappa'_{\mathcal{D}}, l', \nu', b')$, then $\preceq$ is defined by com-



paring the following things in order:

1. $c < c'$.
2. $\sigma \preceq_Q \sigma'$ in Q .
3. The number of 0's in $\kappa_{\mathcal{D}}$ is less than the that in $\kappa'_{\mathcal{D}}$.
4. $l < l'$ if ψ, ψ' are both in the release states.
5. $l > l'$ if none of ψ, ψ' is in the release states.
6. $b < b'$.

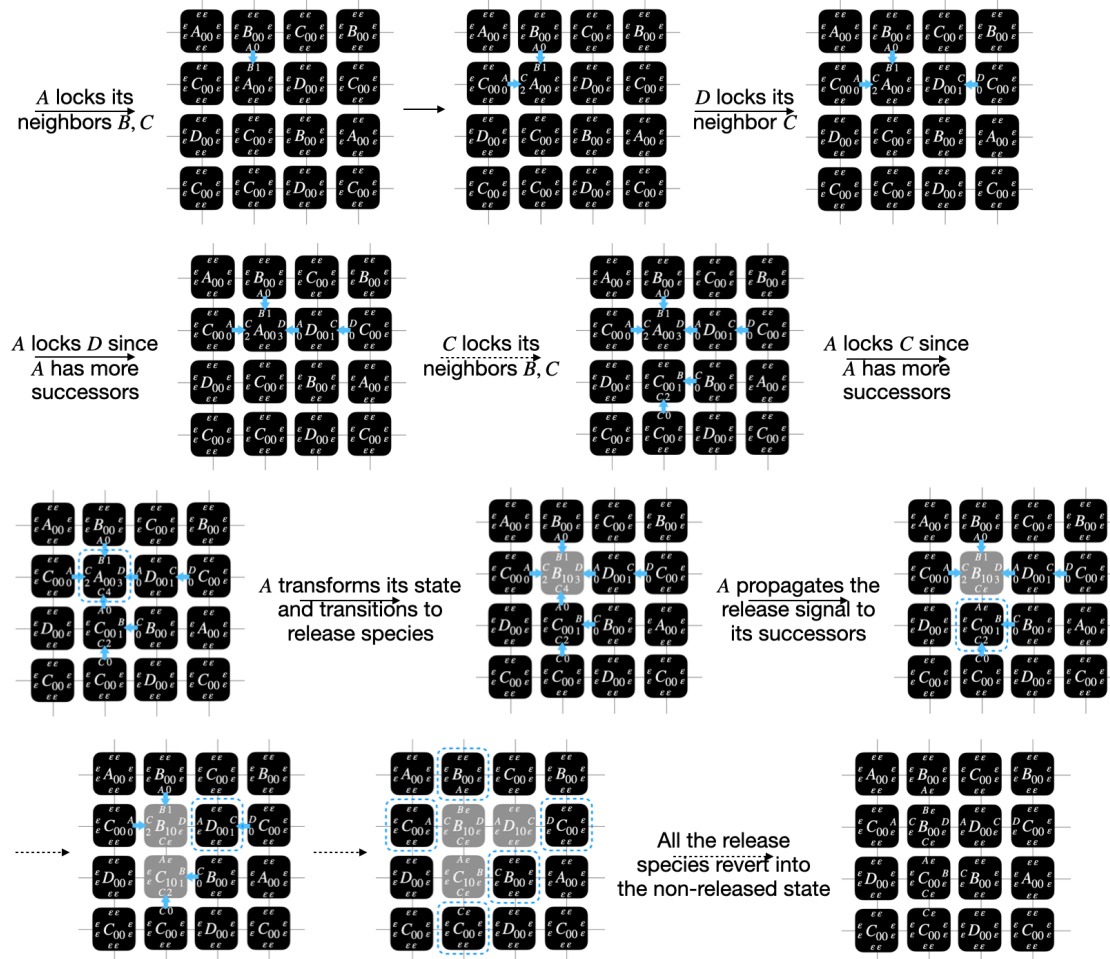


Figure 10.2: Simulating local function of freezing async-CA with freezing d-sCRN.

So far, we have given the complete protocol for constructing a freezing d-sCRN T' that simulates a freezing async-CA Γ . To make the simulation more comprehensive, we summarize the protocol in a high-level manner.



- We have 4 kinds of species: *free species* that is not locked by any other species and is not in the paused or release state; *locked species* that is locked by someone else; *paused species* that, together with its neighborhood forms a fixed point of the local function f , and therefore waiting for some state transition on its neighbors; and *release species* that can unlock its successors when itself is not locked. Notice that a species can simultaneously be a paused species and a locked species, or a release species and a locked species.
- We have 3 kinds of reactions: lock, unlock, and state transition. And the reactions mainly following these 4 rules:
 - A species can lock others only if it is a free species and it has more successors.
 - A species with all its neighbors locked by itself performs a state transition according to the local function.
 - All 4 kinds of species are allowed to be locked by free species.
 - When a species ψ unlocks one of its successors ψ' , ψ always turns ψ' into release species, and increases the counter of ψ' by 1 at the same time.

10.3 Proof

First we give the representation function $\mathcal{R} : Q' \rightarrow Q$. By our construction, for all species $\psi = (\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, l, \nu, b) \in Q'$, ψ is encoding some $\sigma \in Q$. Therefore we define $\mathcal{R}$

s.t. $\mathcal{R}(\psi) = \sigma$. By the definition of simulation between two models, we need to prove that $\Gamma' \Leftrightarrow_{\mathcal{R}} \Gamma$, $\Gamma \dashv \Gamma'$, and $\Gamma' \models_{\mathcal{R}} \Gamma$.



$\Gamma \dashv_{\mathcal{R}} \Gamma'$ We will show that if $\alpha' \rightarrow_{\Gamma'}^1 \beta'$ for some $\alpha', \beta' \in \mathcal{A}(\Gamma')$, then $\mathcal{R}^*(\alpha') = \mathcal{R}^*(\beta')$ or $\mathcal{R}^*(\alpha') \rightarrow_{\Gamma}^1 \mathcal{R}^*(\beta')$.

For any species $\psi(v) = (\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, l, \nu, b)$, suppose that $\psi(v)$ has gone through some reaction in R' and became $\psi'(v) = (\sigma', \sigma'_{\mathcal{D}}, \kappa'_{\mathcal{D}}, l', \nu', b')$, the only reaction that makes $\sigma' \neq \sigma$ is the unimolecular reaction $((\sigma, \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, 4, 0, 0), (\sigma', \sigma_{\mathcal{D}}, \kappa_{\mathcal{D}}, 4, 1, 0), \odot)$ where $\sigma' \neq \sigma$ in the *State_Transitions* protocol. By our construction, this reaction happens only if ψ has locked all its neighbors and $\sigma' = f(\sigma, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{S}}, \sigma_{\mathcal{W}})$. Therefore, the neighbors of $\mathcal{R}(\psi)$ in $\mathcal{R}^*(\alpha')$ are $\sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{S}}, \sigma_{\mathcal{W}}$ and it is possible to achieve $\mathcal{R}^*(\beta')$ by applying the local function on v . This means that $\mathcal{R}^*(\alpha') \rightarrow_{\Gamma} \mathcal{R}^*(\beta')$. In other cases, we always have $\sigma' = \sigma$, so $\mathcal{R}^*(\alpha') = \mathcal{R}^*(\beta')$.

$\Gamma' \models_{\mathcal{R}} \Gamma$ First we need to decide for each configuration $\alpha \in \mathcal{A}(\Gamma)$, which configurations in $\mathcal{A}(\Gamma')$ should be included in Π . For any configuration $\omega \in \mathcal{A}(\Gamma')$, we consider a *locking graph* G_{ω} , which is an infinite directed graph where $V(G_{\omega}) = \mathbb{Z}^2$. $(u, v) \in E(G_{\omega})$ if and only if u is locked by v . For any node $v \in V(G_{\omega})$, we define the *ball* of v , denoted as $\mathcal{B}_{\omega}(v)$, to be a subgraph of G_{ω} with vertex set $\{u : \text{there exists a path from } u \text{ to } v\}$, including all edges on these paths. By our construction, $\mathcal{B}_{\omega}(v)$ is a tree rooted at v . G_{ω} can be covered by the union of the balls of free species, pause species, and release species.

We define $\Pi(\alpha)$ to be the reachable configurations that are mapped to α under $\mathcal{R}^*$, such that except for the cells in the ball of some release species, all other cells cannot lock others. (Note that they cannot be locked either. Otherwise, they could only be locked by



species in the ball of a release species and must therefore belong to that ball as well.) That is, for all $\alpha' \in \Pi(\alpha)$, α' can be covered by only balls of release species and singletons.

Define such configuration set

$$\Lambda = \{ \alpha' : \forall u \in \mathbb{Z}^2, \mathcal{B}_{\alpha'}(u) = \{u\} \text{ or } u \in \mathcal{B}_{\alpha'}(v) \text{ for some } v \text{ s.t. } \alpha(v) \text{ is a release species.} \}.$$

Then we set $\Pi(\alpha) = \mathcal{A}(\Gamma') \cap \mathcal{R}^{*-1}(\alpha) \cap \Lambda$.

Lemma 25. 1. $\Pi(\alpha)$ is non-empty.

2. Given any β s.t. $\alpha \xrightarrow{\Gamma} \beta$, for every $\alpha' \in \Pi(\alpha)$, there exists $\beta' \in \mathcal{A}(\Gamma')$ where $\mathcal{R}^*(\beta') = \beta$ and $\alpha' \rightarrow_{\Gamma'} \beta'$ in finitely many steps.

Proof. Initially, $\Pi(S) \neq \emptyset$ since $S' \in \Pi(S)$. It remains to show that for any $\alpha \in \mathcal{A}(\Gamma)$, if there exists $\alpha' \in \Pi(\alpha)$, then given any β s.t. $\alpha \xrightarrow{\Gamma} \beta$ by applying local function f to some cell v , we can construct a finite sequence of reactions that reaches some $\beta' \in \Pi(\beta)$. Starting at $\alpha' \in \Pi(\alpha)$, $\alpha'(v)$ cannot be a pause species since the neighborhood of v in α is not a fixed point of f . We have all the release species in the neighborhood of v unlock all their successors. Then, all species in this neighborhood can exit the release state by a unimolecular reaction. The configuration remains in $\Pi(\alpha)$. From this configuration, v can lock all its neighbors because v now contains neither a locked species nor a release species. Subsequently, v perform a state transition according to the local function. Now we get a reachable configuration β' that belongs to $\Lambda \cap \mathcal{R}^{*-1}(\beta)$ since the neighborhood of v is in the release ball of v . The proof is complete. $\square$

Lemma 26. Given any β s.t. $\alpha \rightarrow_{\Gamma} \beta$, for every $\alpha'' \in \mathcal{A}(\Gamma') \cap \mathcal{R}^{*-1}(\alpha)$ where $\alpha'' \rightarrow_{\Gamma'} \beta'$ for some $\beta' \in \mathcal{A}(\Gamma') \cap \mathcal{R}^{*-1}(\beta)$, there exists $\alpha' \in \Pi(\alpha)$ s.t. $\alpha' \rightarrow_{\Gamma'} \alpha''$.

Proof. We will show that for every $\alpha'' \in \mathcal{A}(\Gamma') \cap \mathcal{R}^{*-1}(\alpha)$, there exists $\alpha' \in \Pi(\alpha)$ s.t. $\alpha' \rightarrow_{\Gamma'} \alpha''$.

Since $\alpha'' \in \mathcal{A}(\Gamma')$, there exists a set of reactions, say $\mathcal{R}$, that produces α'' from S' by applying them in a specific order. If some of the reactions are applied more than once, we consider each execution as a distinct reaction. This defines a strict total order $\preceq$ on $\mathcal{R}$ where two reactions $r, r' \in \mathcal{R}$ satisfy $r \prec r'$ if and only if r is executed before r' .

Consider the locking graph $G_{\alpha''}$. Let $\mathcal{E}$ be the edge set of $G_{\alpha''}$. Define $\mathcal{E}'$ to be the set of edges that is in the ball of some release species, and $\mathcal{E}''$ the remaining edges. i.e.

$$\mathcal{E}' = \{e : e \in \mathcal{B}_{\alpha''}(v) \text{ for some } v \text{ s.t. } \alpha''(v) \text{ is a release species.}\}, \text{ and } \mathcal{E}'' = \mathcal{E} \setminus \mathcal{E}'.$$

For all $e = (u, v) \in \mathcal{E}''$, we denote by $r_e \in \mathcal{R}$ the corresponding reaction that instructed species v to lock species u . Let $\mathcal{R}'' = \{r_e : e \in \mathcal{E}''\}$ and $\mathcal{R}' = \mathcal{R} \setminus \mathcal{R}''$, where the strict orders on these two sets are directly induced from $\prec$. We are going to show that

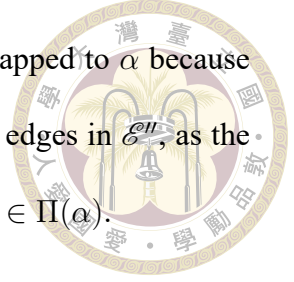
1. A reachable configuration $\alpha' \in \Pi_\alpha$ can be constructed by applying $\mathcal{R}'$ to the initial configuration S' .
2. α'' is reachable from α' by applying $\mathcal{R}''$.

In other words, we can “rearrange” the reaction sequence s.t. $S' \rightarrow \alpha' \rightarrow \alpha''$.

Intuitively, since all the locking reactions in $\mathcal{R}''$ are preserved until the end, we can arrange for those locks to occur after all other reactions are completed without altering the resulting configuration.

For the first part, we will show that $\mathcal{R}'$ is *applicable* to S' , which means that we can successively apply the reactions in $\mathcal{R}'$ to S' following the strict order $\prec$. If it is the

case, we define the resulting configuration to be α' . α' should be mapped to α because all the state transitions are performed. Furthermore, $G_{\alpha'}$ contains no edges in $\mathcal{E}''$, as the corresponding reactions have all been removed. Therefore, we get $\alpha' \in \Pi(\alpha)$.



For every reaction $r' \in \mathcal{R}'$, let $\mathcal{R}_{\prec r'} = \{r \in \mathcal{R}, r \prec r'\}$ be the set of reactions that are executed before r' in the original reaction set $\mathcal{R}$. Define $\mathcal{R}'_{\prec r'}$ and $\mathcal{R}''_{\prec r'}$ similarly. We state the following claim, which we will prove later:

Claim 27. If $\mathcal{R}'_{\prec r'}$ is applicable to S' and results in a configuration $\alpha'_{\prec r'}$, then r' is applicable to $\alpha'_{\prec r'}$.

With this claim, $\mathcal{R}'$ is applicable to S' ; otherwise, there would exist a minimal reaction $r_{\min}$ in $\mathcal{R}'$ that is not applicable to $\alpha'_{\prec r_{\min}}$, contradicting the claim.

For the second part, assume that in the original reaction set $\mathcal{R}$, species $\psi(v)$ in cell v locked a neighboring species $\psi(u)$ with some reaction $r \in \mathcal{R}''$, then $l(v)$ (the number of successors of $\psi(v)$) is larger or equal than $l(u)$ at that time. From then on, $l(u)$ remains unchanged. Moreover, all the edges toward v , including both existing edges and future edges, cannot be removed since v will never transform into a release species (otherwise, v would be in some release ball because it cannot release u after r has been performed). Therefore, $l(v)$ can only increase due to some reactions in $\{r'' \in \mathcal{R}'' : r \prec r''\}$. Based on the above arguments, we conclude that if we apply $\mathcal{R}''$ successively on α' according to $\prec$, the value of l on each pair of reacting species remains the same as in the original sequence. Therefore, $\mathcal{R}''$ is applicable to α' . $\square$

Proof. of Claim 27.

r' affects (at most) two cells u, v . Originally, $\mathcal{R}_{\prec r'} \cup \{r'\}$ is applicable to S' due

to the reachability of α'' . Consider the reactions in $\mathcal{R}''_{\prec r'}$ that affect either u or v . There are at most 7 such reactions since each of them produces a permanent edge around u, v . Let these reactions be $\mathcal{R}''_K = \{r_1, \dots, r_K\}$ where $r_1 \prec \dots \prec r_K$. We'll show that $(\mathcal{R}_{\prec r'} \setminus \mathcal{R}''_K) \cup \{r'\}$ is applicable to S' . Then $(\mathcal{R}_{\prec r'} \setminus \mathcal{R}''_{\prec r'}) \cup \{r'\} = \mathcal{R}'_{\prec r'} \cup \{r'\}$ is applicable to S' because removing other reactions in $\mathcal{R}''_{\prec r'}$ does not affect u, v , and $\mathcal{R}'_{\prec r'}$ is applicable to S' by assumption.

Let $\{e_i : i \in [K]\}$ be the set of edges results from $\{r_i : i \in [K]\}$. WLOG, let e_K be adjacent to u (and possibly to v as well). Suppose that the species in cell u locks another species by reaction r_K . Since the edge e_K is retained in the final configuration α'' and $\alpha''(u)$ is not within the balls of any release species, there must be no release reaction that removes e_K and no state transition on u after r_K . In fact, all u can do after r_K is to lock others or be locked by others, maintaining those locks until reaching α'' . These reactions are all in $\mathcal{R}''$, which means that r' cannot affect u , a contradiction. Therefore, u is locked by another species by reaction r_K . Observe that aside from removing this lock, which is not going to happen, every other reaction related to u after it is locked remains valid before it is locked. These reactions also have the same effect on the encoding variables of u , regardless of whether it is locked or not. As a result, all the reactions between r_K and r' , including r' itself, are still applicable when we remove r_K .

We can repeat the above process, and recursively remove the latest reaction that belongs to $\mathcal{R}''_K$. Eventually, we remove all the reactions in $\mathcal{R}''_K$ and the remaining reactions $(\mathcal{R}_{\prec r'} \setminus \mathcal{R}''_K) \cup \{r'\}$ is still applicable to S' . The proof is complete. $\square$

With Lemma 25 and Lemma 26, The result $\Gamma' \models_{\mathcal{R}} \Gamma$ follows.

$\Gamma \Leftrightarrow_{\mathcal{R}} \Gamma'$ The proof is exactly the same as the the proof regarding (non-freezing) cellular automata in Section 7.1.3.

Remark 28. Although simulating freezing cellular automata by freezing d-sCRN requires careful design on the locking ordering, simulation of freezing d-sCRN with freezing cellular automata can be achieved by slight modification of the protocol given in Section 7.2: The protocol was non-freezing because a cell revert to its original state when its invitation is rejected. Therefore, we associate a counter to each cell and increase the counter by 1 whenever it is rejected. We know that each time it gets rejected, there is a cell 2-hop from it that is accepted and has already undergone a state transition. Thus, each cell can receive a total of $O(k)$ rejections, resulting in a partial ordering of height $O(k)$.

10.4 Complexity

Suppose the freezing CA system has n states, at most m possible outcome of the transition function for a given neighborhood, and the partial order is of height k . Then the number of states we need in the freezing dsCRN system is $O(n^5 \cdot k)$ where recording the information in its neighborhood takes $O(n^5)$ states and increasing the counter results in a $O(k)$ scale in states. The counter also generate a $O(k^2)$ scale in the reaction complexity. Follows the similar argument as in Section 7.1.4, the number of reactions needed is $O((n^5 + m)n^5k^2)$. Take a look at the new partial order defined in Section 10.2, we can see that the height is still $O(k)$: Every increment of a counter corresponds to an increment of the state in the simulated CA system within a region of constant diameter.



Chapter 11 Conclusion

In this work, we construct simulations between sCRN and other well-studied models. Most of these simulations are two-way, except for the straightforward result that aTAM cannot simulate sCRN. We conclude with some questions that may be worth considering in the future:

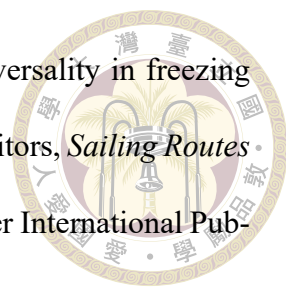
- Although 2-dimensional aTAM cannot simulate sCRN, can we simulate sCRN using 3-dimensional aTAM? Our conjecture is yes for the seeded system. We may simulate the reactions in sCRN layer by layer.
- In the simulation of TA system, we focus solely on the affinity-strengthening system, which is closely related to aTAM. While simulating the general TA system, where a configuration could break into two parts along an arbitrary cut, seems impossible due to the requirement of global information, is it feasible to simulate the restricted system where only single tile detachments are allowed?
- There are many other works studying cross-model simulations in the field of molecular computing, can we compare these models within a unified framework?





References

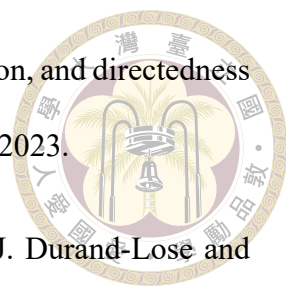
- [1] L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang. Running time and program size for self-assembled squares. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, page 740–748, 2001.
- [2] R. M. Alaniz, J. Brunner, M. Coulombe, E. D. Demaine, J. Diomidova, R. Knobel, T. Gomez, E. Grizzell, J. Lynch, A. Rodriguez, R. Schweller, and T. Wylie. Complexity of reconfiguration in surface chemical reaction networks, 2023.
- [3] R. M. Alaniz, D. Caballero, S. C. Cirlos, T. Gomez, E. Grizzell, A. Rodriguez, R. Schweller, A. Tenorio, and T. Wylie. Building squares with optimal state complexity in restricted active self-assembly. *Journal of Computer and System Sciences*, 138:103462, 2023.
- [4] J. C. Alumbaugh, J. J. Daymude, E. D. Demaine, M. J. Patitz, and A. W. Richa. Simulation of programmable matter systems using active tile-based self-assembly. In *DNA Computing and Molecular Programming*, pages 140–158, 2019.
- [5] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4), 12 2008.
- [6] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4), dec 2008.

- 
- [7] F. Becker, D. Maldonado, N. Ollinger, and G. Theyssier. Universality in freezing cellular automata. In F. Manea, R. G. Miller, and D. Nowotka, editors, *Sailing Routes in the World of Computation*, pages 50–59, Cham, 2018. Springer International Publishing.
- [8] K. Bhattacharjee, N. Naskar, S. Roy, and S. Das. A survey of cellular automata: Types, dynamics, non-uniformity and applications. *Natural Computing*, 19:433 – 461, 06 2020.
- [9] T. Brailovskaya, G. Gowri, S. Yu, and E. Winfree. Reversible computation using swap reactions on a surface. In *DNA Computing and Molecular Programming*, pages 174–196, 2019.
- [10] D. Caballero, T. Gomez, R. Schweller, and T. Wylie. Verification and computation in restricted tile automata. *Natural Computing*, pages 1–19, 2021.
- [11] C. Chalk, A. Luchsinger, E. Martinez, R. Schweller, A. Winslow, and T. Wylie. Freezing simulates non-freezing tile automata. In *DNA Computing and Molecular Programming*, pages 155–172, 2018.
- [12] H. L. Chen, D. Doty, and D. Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014.
- [13] H. L. Chen, D. Doty, D. Soloveichik, and W. Reeves. Rate-independent computation in continuous chemical reaction networks. *Journal of the ACM*, 70(3):1–61, 2023.
- [14] S. Clamons, L. Qian, and E. Winfree. Programming and simulating chemical reaction networks on a surface. *Journal of The Royal Society Interface*, 17:20190790, 2020.
- [15] M. Cook, D. Soloveichik, E. Winfree, and J. Bruck. *Programmability of Chemical*

Reaction Networks, pages 543–584. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.



- [16] A. Dennunzio, E. Formenti, and L. Manzoni. Computing issues of asynchronous ca. *Fundam. Inf.*, 120(2):165–180, 2012.
- [17] Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Amoebot - a new model for programmable matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, page 220–222, New York, NY, USA, 2014. Association for Computing Machinery.
- [18] D. Doty, J. H. Lutz, M. J. Patitz, R. T. Schweller, S. M. Summers, and D. Woods. The tile assembly model is intrinsically universal. In *IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 302–310, 2012.
- [19] F. Fages, G. Le Guludec, O. Bournez, and A. Pouly. Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In J. Feret and H. Koepl, editors, *Computational Methods in Systems Biology*, pages 108–127, Cham, 2017. Springer International Publishing.
- [20] N. Fatès and L. Gerin. Examples of fast and slow convergence of 2d asynchronous cellular systems. In *Cellular Automata*, pages 184–191, 2008.
- [21] E. Goles, D. Maldonado, P. Montealegre, and M. Ríos-Wilson. On the complexity of asynchronous freezing cellular automata. *Information and Computation*, 281:104764, 2021.
- [22] E. Goles, N. Ollinger, and G. Theyssier. Introducing Freezing Cellular Automata. In *Cellular Automata and Discrete Complex Systems, 21st International Workshop (AUTOMATA 2015)*, volume 24 of *TUCS Lecture Notes*, pages 65–73, 2015.

- 
- [23] D. Hader and M. J. Patitz. The impacts of dimensionality, diffusion, and directedness on intrinsic cross-model simulation in tile-based self-assembly, 2023.
- [24] M. J. Patitz. An introduction to tile-based self-assembly. In J. Durand-Lose and N. Jonoska, editors, *Unconventional Computation and Natural Computation*, pages 34–62, 2012.
- [25] L. Qian and E. Winfree. Parallel and scalable computation and spatial dynamics with dna-based chemical reaction networks on a surface. In S. Murata and S. Kobayashi, editors, *DNA Computing and Molecular Programming*, pages 114–131, Cham, 2014. Springer International Publishing.
- [26] P. Rothmund and E. Winfree. The program-size complexity of self-assembled squares. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 459–468, 2000.
- [27] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing: An International Journal*, 7(4):615–633, dec 2008.
- [28] J. von Neumann. Theory of self-reproducing automata, 1966. <https://cba.mit.edu/events/03.11.ASE/docs/VonNeumann.pdf>.
- [29] E. Winfree. Algorithmic self-assembly of dna. In *Proceedings of the International Conference on Microtechnologies in Medicine and Biology*, pages 4–4, 2006.