國立臺灣大學電機資訊學院電機工程研究所

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

對輸出視而不見的化學反應網路的速度

The speed of output-oblivious chemical reaction network

吳昱融

Yu-Rong Wu

指導教授: 陳和麟 博士

Advisor: Ho-Lin Chen Ph.D.

中華民國 111 年 1 月

January, 2022

doi:10.6342/NTU202300354

Acknowledgements

這篇論文可以順利完成，首要感謝的就是我的指導教授陳和麟老師，在研究的過程中給予我非常大的彈性，可以對於自己喜歡的方向自由發揮，並且在碰到瓶頸時能夠適當的給予幫助，每當在閱讀論文時遇到有不太能夠理解的地方，在和老師討論過後都能夠讓我對於瓶頸的地方有更深刻的理解、突破，讓研究得以順利的進行下去。對於論文架構老師也給予了非常多的建議，讓其他人在閱讀時能夠快速地了解我的研究成果。

就讀碩士的這兩年多的時間裡，可能是我整個求學過程中最開心的一段時間，老師在研究進度上不會給予壓力，能夠讓自己適當的花費一些時間在其他有興趣的事物上。雖然不會給予壓力，但在指導學生方面卻是盡心盡力，每個禮拜的個人 meeting 和團體 meeting 都要花費老師大量的時間，並且每次 meeting 時都會認真的聆聽，同時給予珍貴的意見。在學生生涯的最後一段路上，非常開心且幸運能夠遇到這樣的老師。

最後要感謝口試委員們: 于天立、劉智弘、呂學一教授們。謝謝三位老師在過年前抽空來當我的口試委員，並且在口試中對於論文的內容、報告方式提出了非常多的實用建議。

2023.2.12 吳昱融

doi:10.6342/NTU202300354

doi:10.6342/NTU202300354

摘要

隨機化學反應網路 (Chemical Reaction Network) 被科學家廣泛的使用來描述分子之間的反應。化學反應網路可以視為一種用來描述分子反應的語言。在化學反應網路中所能夠做到的計算，在現實中的分子也能夠做到。我們假設每個反應的反應率都是 1，因為在穩定計算下反應率不會影響到最後的結果。在此篇中我們專注在”對輸出視而不見的化學反應網路”上，此種網路可以直接和其他化學反應網路串接且不會有錯誤。[3] 第一個定義了輸出不會當成某些反應的反應物的化學反應網路為”對輸出視而不見的化學反應網路”，並且證明了這類網路能夠計算什麼函數，但裡面有一些很慢的反應。我們對這些慢的反應進行改良，並將計算”對輸出視而不見的化學反應網路”的時間從 $O(N^r)$ 降到 $O(N \log N)$.

關鍵字：分子計算、化學反應網路

doi:10.6342/NTU202300354

doi:10.6342/NTU202300354

Abstract

Stochastic Chemical Reaction Networks(CRNs) are widely used to describe how molecules interact with each other by scientists. CRNs can be viewed as a language for describing the reaction between molecules. If the CRNs can compute the function f, so do the molecules in nature. We assume that the rate of each reaction is one because it doesn't affect the output under the stable computation. In this thesis, we only consider one special type of CRNs called output oblivious CRN. These CRNs can concatenate to other CRNs directly. Chugg et al. [3] first define the CRN whose output species never being a reactant of reaction in CRN as output oblivious CRN. They have characterized all functions $f : N^2 \to N$ which can be stably computed by output oblivious CRNs with a leader. However, the CRNs they designed may require an extremely long time to converge to the correct output. We modify the original reactions and reduce the time needed for output oblivious CRN from $O(N^r)$ to $O(N \log N)$.

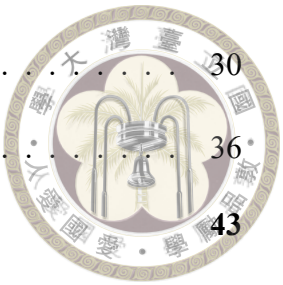
Keywords: Molecular Computation, Chemical Reaction Network, output oblivious

doi:10.6342/NTU202300354

doi:10.6342/NTU202300354

Contents

doi:10.6342/NTU202300354



doi:10.6342/NTU202300354

Chapter 1 Introdution

Stochastic Chemical Reaction Networks(CRNs) are a system of reactions involving chemical species. CRNs are described as finite sets of chemical reactions such as $A + B + C \to D$. Stochastic Chemical Reaction Networks are widely used to describe how molecules interact with each other by scientists. Molecules interact with a stochastic process in reality, so we study the computational power of the stochastic CRNs.

Soloveichik et al. [6] designed an implementation that finds a set of DNA molecules corresponding to the species of any arbitrary set of reactions. That is, CRNs can be viewed as a language for describing the reaction between molecules. If the CRNs can compute the function f, so do the molecules in nature. The model is described as a continuous time, discrete state, Markov process [2, 7, 10]. A configuration represents the integer count of the species. A reaction can occur only when the current configuration has enough count of each species.

CRNs can be used to perform various computations, but we only focus on function computation in this thesis. We only consider the type of computation that the input of a function is represented by the number of specific species, and the output of the function is also represented by the number of specific species [3, 8, 11]. For instance, the CRN, $X \to 2Y$, stably compute the function $f_1(x) = 2x$. X is the input species and Y is

doi:10.6342/NTU202300354

the output species. Population Protocols [1, 4] are closely related to this computation, but the number of reactants and products are two in Population Protocols. These models can simulate Turing machines with a small probability of error. In order to have a stable computation (i.e., without any error), the ability of computation has been affected. Angluin et al. [5] show that the class of semilinear predicates can be stably computed in population protocols. Chen et al. [9] show that the function can be computed by CRN if and only if it is a semilinear function.

In this thesis, we assume that the rate of each reaction is 1, because it is difficult for the implementation from [6] to precisely control the rate of the reactions. The second point is that we focus on functions that can be stably computed by CRNs. The rate of reactions in the stable computation only affects the execution sequence, not the output. Thus, we don't care about the rate of reactions.

We can sometimes concatenate two CRNs directly without any error, but sometimes not. We will show two examples for the two cases. We have two CRNs, $X \to 2Y$ computes $f_1(x) = 2x$, and $Y \to 3Z$ compute $f_2(x) = 3x$. If we want to compute the function $f = f_2(f_1(x))$, we can directly concatenate the two CRNs without any error. We give another two systems which can't concatenate directly. The first system begins with n copies of X and a leader L. The first CRN has two reactions, $X \to Y$ and $Y + L \to \emptyset$. The second CRN has one reaction, $Y \to 2Z$. The first CRN produces $n-1$ output species Y. If the second CRN consumes the output of the first CRN before it is eliminated, then we will get a wrong answer like $2n$ instead of $2n-2$. We don't know what time the first CRN eliminates the Y. Thus, composability is an important issue for CRNs.

In this thesis, we only consider one special type of CRNs called output oblivious

doi:10.6342/NTU202300354

CRN. These CRNs can concatenate to other CRNs directly. Chugg et al. [3] first define the CRN whose output species never being a reactant of reaction in CRN as output oblivious CRN. Output oblivious CRN can be the upstream CRN of other CRNs or concatenate to other output oblivious CRN directly without any error. Meanwhile, they specify all functions $f : N^2 \to N$, which can be stably computed by output oblivious CRN with a leader. They also left an open problem about the time complexity of output oblivious CRNs. This motivates us to study the work in this thesis. Severson et al. [11] specify all functions $f : N^d \to N$ can be stably computed by output oblivious CRN with a leader. Hashemi et al. [8] specify all functions $f : N^d \to N$ can be stably computed by output oblivious CRN without a leader. The future work for us is to deal with these two conditions.

Our work : we construct new reactions to replace the original reactions in [3], which are not single-molecular or bi-molecular reactions and are very slow. Meanwhile, we make sure that the racing conditions between different reactions do not affect the speed of computation. We have a big improvement compared to the speed of original reactions in [3]. We reduce the time from $O(N^r)$ to $O(N \log N)$. Chen et al. [9] imply that it is always possible to reach a state (speed fault), which needs $\Omega(N)$ time to finish, during the process of the computation. Hence, maybe the optimal speed of output oblivious CRN could be $\theta(N)$.

In chapter 2, we provide some definitions and background knowledge. In chapter 3, we list all the constrictions that come from [3] and explain every single chemical reaction network. This helps us to design new reactions that have the same function as the original reactions. In chapter 4, we modify the reaction with many reactants to single-molecular or bi-molecular reactions because it could be the speed bottleneck.

doi:10.6342/NTU202300354

doi:10.6342/NTU202300354

Chapter 2 Preliminaries

In this chapter, we provide some definitions and background knowledge. We use the same notation and definition defined in [3]. The definitions are stated again in this chapter. A *chemical reaction network* (CRN) is defined as a pair $\mathcal{C} = (S, R)$. $S = \{S_1, S_2, ..., S_m\}$ is the finite set of species and R is the finite set of reactions. A *reaction* is defined as a pair $(\mathbf{R}, \mathbf{P}) \in N^m \times N^m$. **R** specifies the counts of consumed species that the reaction need. **P** specifies the counts of product species that the reaction produces. For instance, let $S = \{X, Y\}$ and the pair $= ((1, 1), (2, 2))$. The reaction consumes one copy of X and one copy of Y, then produces two copies of X and two copies of Y. We also describe this as $X + Y \rightarrow 2X + 2Y$.

A *configuration* **c** specifies the count of all species $s \in S$. When any reaction $r \in R$ occurs, **c** changes. We defined a reaction is *applicable* if the current configuration $\mathbf{c} > \mathbf{R}$ which means the number of species S_i of **c** is greater than or equal to the number of species S_i of **R**, $1 \leq i \leq m$. The new configuration $\mathbf{c}' = \mathbf{c} - \mathbf{R} + \mathbf{P}$ when a reaction occurs. We say $\mathbf{c}'$ is *reachable* from **c**. Moreover, we say a configuration **d** is reachable from **c** if there exists a execution sequence $\mathbf{c} \rightarrow \mathbf{c_2} \rightarrow ... \rightarrow \mathbf{d}$.

Let the $f : N^l \rightarrow N$ be a function and the input $\mathbf{n} = (n_1, n_2, ..., n_l)$. The computation starts at an initial configuration $\mathbf{c_0}$ which only has n_i copies of X_i, $1 \leq i \leq l$,

doi:10.6342/NTU202300354

and a leader L. To compute the function, the *CRN computer* with a leader is defined as $\mathcal{C} = (\mathcal{S}, \mathcal{R}, \mathcal{I}, \mathcal{O}, L)$. $\mathcal{S}$ is a finite set of species. $\mathcal{R}$ is a finite set of reactions. $\mathcal{I} = \{X_1, X_2, ..., X_l\}$ and $\mathcal{O} = \{Y\}$ are the ordered set of input species and output species, respectively. L is the leader species. A *leader* is a molecule that is present in the system initially. A *computation* is an execution of $\mathcal{C}$ from an initial configuration $\mathbf{c_0}$ to a stable configuration. A configuration $\mathbf{c}$ is *stable* if $\mathbf{c}(Y) = \mathbf{d}(Y)$ where $\mathbf{d}$ is all configurations reachable from $\mathbf{c}$. $\mathbf{c}(Y)$ and $\mathbf{d}(Y)$ represent the number of output species Y in configuration $\mathbf{c}$ and $\mathbf{d}$ respectively.

For all configurations $\mathbf{a}$ reachable from $\mathbf{c_0}$, if there exists a stable configuration $\mathbf{a}'$ reachable from $\mathbf{a}$ and $\mathbf{a}'(\mathbf{Y}) = f(\mathbf{n})$, we call this is a *stable computation*.

definition 1. *If a CRN never consumes its output species, then we call it output oblivious CRN.*

Here we show two CRNs for the function $f(x_1, X_2) = \max(x_1, x_2)$ and the function $f(x) = 3x$. The CRNs for the function $f(\mathbf{n}) = f(x_1, x_2) = \max(x_1, x_2)$ given from [11] is showing below. Input $\mathbf{n} = (x_1, x_2)$ represents we have n_1 copies of X_1 and n_2 copies of X_2 initially. X_1 and X_2 are the input species. Y is the output species. As we can see, Y is the reactant of reaction 2.4. Thus, the CRN is not an output-oblivious CRN. The function is computed as $x_1 + x_2 - \min(x_1, x_2)$. The reaction 2.1 and 2.2 transfers all X_1 and X_2 into $Y + Z_1$ and $Y + Z_2$ respectively. The reaction 2.3 transfers one copy of Z_1 and one copy of Z_2 into one copy of K. We use K to eliminate the over-produced output Y by

doi:10.6342/NTU202300354

reaction 2.4.

$$X_1 \to Y + Z_1, \tag{2.1}$$

$$X_2 \to Y + Z_2, \tag{2.2}$$

$$Z_1 + Z_2 \to K, \tag{2.3}$$

$$K + Y \to \emptyset. \tag{2.4}$$

The CRN for the function $f(\mathbf{n}) = f(x) = 3x$ is shown below. Input $\mathbf{n} = x$ represents we have x copies of Y species. In this CRN, Y is the input species, and Z is the output species. The output species doesn't be the reactant of any reaction, so the CRN is output oblivious CRN. The CRN transfers every single Y into $3Z$.

$$Y \to 3Z. \tag{2.5}$$

Here we start to talk about the *composability* of CRN. The two CRNs described in definition 1 stably compute the two functions separately. If we concatenate these two CRNs together to compute the function $f(x_1, x_2) = \frac{\max(x_1,x_2)}{3}$, it may lead to a wrong answer and not be stable.

For example, we want to compute $f(5, 4) = \frac{\max(5,4)}{3} = 1$. The up-stream CRN for the function $f(x_1, x_2) = \max(x_1, x_2)$ may produce nine copies of Y, five copies of Z_1 and four copies of Z_2. The downstream CRN for the function $f(x) = 3x$ may consume the output species Y of the upstream CRN before the over-produced Ys are eliminated. At this condition, we can obtain the answer 3, which is wrong.

The output-oblivious CRN handles the condition. The downstream CRN can directly

doi:10.6342/NTU202300354

use the output of the upstream CRNs without any error, even if the output of the upstream CRN has not been stable. As the output of the upstream CRN wouldn't be consumed, it can be used directly as the input of the downstream CRN. [3] proves that a semilinear function $f : N^2 \to N$ is stably computable by an output-oblivious CRN with a leader if and only if it is both increasing and either *grid-affine* or the minimum of a finite set of *fissure functions*.

We define what *grid* is before defining the *grid-affine function*.

definition 2. *A grid is* $A = \{(p, 0)a_1 + (0, q)a_2 : a_i \in N\} + \mathbf{o}$ *where* p *and* $q \in N$ *and* $\mathbf{o} \in N^2$.

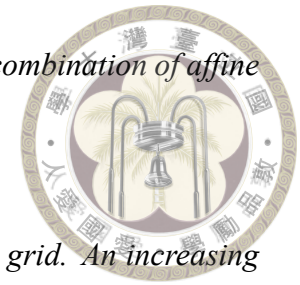
p and q are the periods of the grid. $\boldsymbol{o}$ is the offset of grid. $(p, 0)$ and $(0, q)$ are the base vectors of the grid. Grids could be one-dimensional, two-dimensional, or zero-dimensional. A one-dimensional grid means either $p = 0$ or $q = 0$ on the grid. Precisely, we can say the grid lacks of one base vector. If $p > 0$ and $q = 0$, the grid is a line along the x-axis with period p. If $p = 0$ and $q > 0$, the grid is a line along the y-axis with period q. Similarly, zero-dimensional gird means the grid lacks of the two base vectors, so we can view a zero-dimensional grid as a point which is the offset $\mathbf{o}$. A two-dimensional grid means the grid has two base vectors.

Next, we define *increasing* because all functions in this thesis are increasing.

definition 3. *Let* $\mathbf{n_1} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_l}\}$ *and* $\mathbf{n_2} = \{\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_l}\}$, *A function* $f : N^l \to N$ *is increasing if for all* $n_1 \geq n_2$, *then* $f(\mathbf{n_1}) \geq f(\mathbf{n_2})$.

definition 4. *An affine function* $f : N^m \to N$ *is defined as below. Let input* $\mathbf{n} = (x_1, x_2, ..., x_m) \in N^m$, $f(\mathbf{n}) = \sum_{i=1}^{m} a_i x_i$ *where* $a_i \in Q$, $1 \leq i \leq m$.

doi:10.6342/NTU202300354



definition 5. *A function is a grid-affine function if and only if f is a combination of affine functions on grids of period p, for some $p \in N^+$.*

definition 6. *Let* **G** *be a two-dimensional grid.* **o** *is the offset of the grid. An increasing semilinear function $f : \mathbf{G} \to N$ is a partial fissure function if it can be represented as the following form for all* $\mathbf{n} \in \mathbf{G} \geq \mathbf{o}$.

$$f(n) = \begin{cases} \varphi_A(\mathbf{n}) & , if\ \varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n}) \leq -k, \\ \varphi_{-i}(\mathbf{n}) = \varphi_A(\mathbf{n}) - d_{-i} & , if\ \varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n}) = -i, 1 \leq i < k, \\ \varphi_i(\mathbf{n}) = \varphi_A(\mathbf{n}) - d_i & , if\ \varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n}) = i, 0 \leq i < k, \\ \varphi_B(\mathbf{n}) & , if\ \varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n}) \geq k. \end{cases}$$

where $\varphi_A(\mathbf{n}) = A_0 + A_1 n_1 + A_2 n_2$ and $\varphi_B(\mathbf{n}) = B_0 + B_1 n_1 + B_2 n_2$. A_0, B_0, A_1, B_1, A_2 and B_2 are nonnegative. $k \in N^+$. We call the line $\varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n}) = i$ is fissure line if $-k < i < k$. d_i is a nonnegative integer if i is on the fissure line, otherwise $d_i = 0$.

A (complete) fissure function $f : N^2 \to N$ is a combination of partial fissure functions on the grid.

2.1 Kinetic model

The kinetic model is described the same as [9]. The kinetic model defines the time of a reaction by probability. In this thesis, all reactions have the same rate constants. This model allows us to study the computational complexity of CRN computation in section 4. Given a volume v and current configuration **c**. The volume is proportional to the maximum number of molecules during the process of computation in our thesis [7]. That is, the density is bounded. The *propensity* of a reaction $a : X \to ...$ in configuration **c** is $\rho(\mathbf{c}, a) =$

doi:10.6342/NTU202300354

$\#_c^X$. $\#_c^X$ is the number of X. The propensity of a reaction $a : X + Y \to ...$, where $X \neq Y$, is $\rho(\mathbf{c}, a) = \frac{(\#_c^X)(\#_c^Y)}{v}$. $\#_c^Y$ is the number of Y. The propensity of a reaction $a : X + X \to ...$, where $X \neq Y$, is $\rho(\mathbf{c}, a) = \frac{(\#_c^X)(\#_c^X)}{v}$. The time for the reaction occurs is inversely proportional to its propensity in configuration **c**.

doi:10.6342/NTU202300354

Chapter 3 Construction analysis

In this chapter, we will list all the constrictions below come from [3] and explain every single chemical reaction network, which will be reconstructed in Section 4. We have to analyze the following reactions because they may have a race condition, making it hard to estimate the speed of computation of reactions.

The result from [3] proves that a semilinear function $f : N^2 \to N$ is stably computable by an output-oblivious CRN with a leader if and only if it is both increasing and either grid-affine (intuitively, its domains are congruence classes), or the minimum of a finite set of fissure functions (intuitively, functions behaving like the min function). Hence, three parts remain to be analyzed in this construction: partial grid-affine function, partial fissure functions, and combining either partial grid-affine function or partial fissure functions to a complete function whose domains are completely two-dimensional. It is just a roughly classified with more detailed chemical reaction networks.

In section 3.1, we will describe CRN which computes the partial grid affine function. In section 3.2, we will handle the part about the partial fissure function. In section 3.3, we will describe CRN which combines partial functions into a single function.

doi:10.6342/NTU202300354

3.1 Partial affine function

In this section, we restate reactions of partial grid affine CRNs more derailed. [3] proves Lemma 1 by giving a construction and showing it can stably compute any increasing affine function. Because they give the construction, we show the lemma 1 here.

Lemma 1. ***([3]lemma 8)*** *Let $\boldsymbol{G}$ be a grid. Any increasing affine function $f : \boldsymbol{G} \to N$ is output-oblivious.*

All affine functions have the form $f(\mathbf{n}) = a_1 n_1 + a_2 n_2 + a_o$ as an example of a partial increasing affine function where $a_1, a_2 \in Q^+$ and $a_0 \in Q$, then we explain how to construct an output oblivious CRNs that computes it. Since all the functions in this paper are increasing, a_1, a_2 must be greater than or equal to 0. Let $\boldsymbol{G} = \{(p,0)b_1 + (0,q)b_2 : b_i \in N\} + \mathbf{o}$ is a grid $\in N^2$.$(p,0)$ and $(0,q)$ are base vectors of $\boldsymbol{G}$. $\mathbf{o}$ is offset of $\boldsymbol{G}$. We also can view $\mathbf{o}$ as the start point of the grid. we will show how the constructions given from [3] below compute the function:

$$L + o_1 X_1 + o_2 X_2 \to L^{'} + (a_1 o_1 + a_2 o_2 + a_0) Y$$

$$L' + pX_1 \to L^{'} + a_1 pY$$

$$L' + qX_2 \to L^{'} + a_2 qY$$

On input $\mathbf{n} = (n_1, n_2) \in \boldsymbol{G}$, i.e., we have n_1 copies of X_1 and n_2 copies of X_2, the reactions above can correctly produce $f(\mathbf{n})$ copies of Y. There are only three reactions here to compute the increasing affine function. The first reaction produces the base case of output which means the input $\mathbf{n} = (0,0)$. The second reaction produces Y when p number of X_1 gets together. Since the base vector of $\boldsymbol{G}$ in the x-axis is $(p,0)$, it says that

doi:10.6342/NTU202300354

every p X_1 can execute the second reaction once, then transfer every single X_1 into a_1 Y. Similarly, the third reaction transfers every X_2 into a_2 Y when every q number of X_1 gets together. These three reactions will not have any race condition because we have a leader L in the first reaction, so the n_1 copies of X_1 and n_2 copies of X_2 aren't available for the second and third reaction until the first reaction occurs. Moreover, the second and third reactions deal with the X_1 and X_2 respectively. The reactions don't have any racing conditions, so if we want to estimate the speed of the reactions, we can handle the reactions one by one directly.

3.2 Partial fissure functions

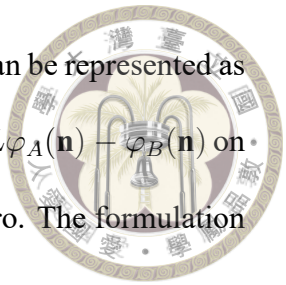
In this section, we concentrate on the construction given from lemma 2 in this section. In section 3.2.1, we handles the case the input $\mathbf{n} = (0, 0)$. In section 3.2.2, we describe the Z-producing reactions which consume the input species, X_1 and X_2, and produce the Z_1 and Z_2. In section 3.2.3, we describe the Z-consuming reactions which consume the Z_1 and Z_2, then change the value of state component d we will introduce later. In section 3.2.4, we describe the Y-producing reactions which transfer the d into output species Y.

Lemma 2. ***([3] lemma 10)*** *Any partial fissure function $f : \boldsymbol{G} \to N$ is output-oblivious.*

For convenience, we assume that $\boldsymbol{G}$ is N^2, i.e., the base vectors of the grid are $(1, 0)$ and $(0, 1)$. simultaneously, the offset $\mathbf{o}$ is $(0, 0)$. It can easily generalize to larger grid periods which means that the base vectors can be $(p, 0)$ $(0, q)$ and $p, q \in N$. There are a few parts in the constructions and we can roughly classify the constructions into three parts, Z-producing reactions, Z-consuming reactions, and Y-producing reactions.

Partial fissure functions described by the definition 6 seem a bit complicated, so we

doi:10.6342/NTU202300354



view it in another way mentioned in [1]. The partial fissure function can be represented as $f(\mathbf{n}) = \min\{\varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n})\} - d_i$, where d_i is determined by the line $L\varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n})$ on which the input $\mathbf{n}$ resides. If d_i is not on the fissure line, then d_i is zero. The formulation above seems different from the definition 6, but they are equivalent.

We know that the partial fissure function is a function on a grid that is two-way-infinite. For all points $\mathbf{p}$ on the grid, there exists a line L_i which contains the point. In other words, L_i is the line where $\varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n}) = i$, where $\mathbf{n} \in N^2$ and $i \in N$, resides on. All of the lines have the same slope. At the same time, the partial fissure function includes $2k-1$ "fissure lines" L_i, $-k < i < k$. Furthermore, there are two other lines called the *lower boundary line* and the *upper boundary line* in this construction. The *lower boundary line* is the line L_i for i in range $[k, ..., K-1]$,where $K = k + d_{max}$. Similarly, the *upper boundary line* is the line L_i for i in range $[-K+1, ..., -k]$.

In section 3.2.1, we describe which reactions handle the case that input $\mathbf{n} = (0, 0)$ and explain some notions about the system here. In section 3.2.2, we explain the Z-producing reactions which consume X_1 and X_2, and produce Z_1 and Z_2. In section 3.2.3, we explain the Z-consuming reactions which consume Z_1 and Z_2, then change the value of state component d we will introduce later. In section 3.2.4, we explain the Y-producing reactions which transfer the d into output species Y.

3.2.1 Base case

In this section, we describe the case that input $\mathbf{n} = (0, 0)$. First, we introduce some notations. For $\mathbf{n} \in \boldsymbol{G}$,$M(\mathbf{n}) = (\varphi_A(\mathbf{n}), \varphi_B(\mathbf{n}))$. $M^{-1}(\varphi_A(\mathbf{n}), \varphi_B(\mathbf{n})) = \mathbf{n}$. We have a state component $[l_x, l_z, d]$ with three state component values. $[l_x, l_z, d]$ is a species.

doi:10.6342/NTU202300354

1. l_x represents the line L_i where the current $\mathbf{x} = (X_1, X_2)$ resides, $\mathbf{x}$ would change arbitrarily before all the input has been consumed out and l_z is in the range $-K+1 \leq l_z \leq K-1$.

2. l_z represents the line L_i where $M^{-1}(\mathbf{z})$ resides, and l_z is in range $-K+1 \leq l_z \leq K-1$.

3. $d = f(M^{-1}(\mathbf{z}) - y$ is the deficit in the number of y' s produced. $f(M^{-1}(\mathbf{z})$ is the "true" output and the y is the number of outputs that have been produced so far, so we still have to produce the d copies of species Y. In brief, we can consider d as the number of output that we have to produce and it would be consumed in the process of the reactions that occur and transfer to the output species Y. d is in the finite range $-d_{max} \leq d \leq 2d_{max} + 1$, where $d_{max} = max(d_i| - k \leq i \leq k)$.

Lets consider input $\mathbf{n} = (0, 0)$, we have the construction below to handle $(\varphi_A(0,0), \varphi_B(0,0))$ = (A_0, B_0).

$$L \to L^{'} + A_0 Z_1 + B_0 Z_2$$

The constructions of the partial fissure function compute the output backward from the quantities $\varphi_A(\mathbf{x})$ and $\varphi_B(\mathbf{x})$ instead of computing the output directly because the partial fissure function is a min-like function which can be described as $f(\mathbf{n}) = min\{\varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n})\} - d_i$ and the min is easy to compute. Thus, we have to transfer all X_1 and X_2 into Z_1 and Z_2 respectively.

Definition 6 tell us that $\varphi_A(\mathbf{n}) = A_0 + A_1 n_1 + A_2 n_2$ and $\varphi_B(\mathbf{n}) = B_0 + B_1 n_1 + B_2 n_2$, so when $n_1 = 0$ and $n_2 = 0$, we only have to produce A_0 copies of Z_1, B_0 copies of Z_2 and $L^{'}$. This is the first Z-producing reaction, but the reaction only occurs once. We can

doi:10.6342/NTU202300354

view L' as a trigger which will tell us that the system is ready for a Z-producing reaction. When the L' is present, Z-producing reactions can be executed, but other reactions are not necessarily disabled. Although it is ready for a Z-producing reaction, it doesn't mean that the other reactions will be unable. That is, the Z-producing reactions, Z-consuming reactions, and Y-producing reactions have a race condition. We will explain how their interaction could be and how it could guarantee that after all the reactions finish, the d is not a negative number, which means that we didn't overproduce the output species Y, in the rest of the subsections.

3.2.2 Z-producing reactions

Z-producing reactions consume the input species, X_1 and X_2, and produce the Z_1 and Z_2. We have two Z-producing reactions here shown below, where * represents that the value of the state component doesn't change when the following response executes.

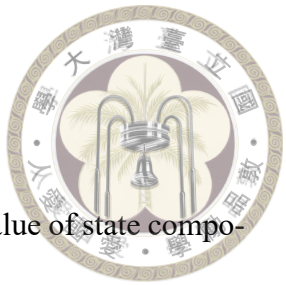
$$[l_x, *, *] + X_1 \rightarrow [l_x + (A_1 - B_1) \mod 2K - 1, *, *] + A_1 Z_1 + B_1 Z_2$$

$$[l_x, *, *] + X_2 \rightarrow [l_x + (A_2 - B_2) \mod 2K - 1, *, *] + A_2 Z_1 + B_2 Z_2$$

When we have a X_1 and the first reaction occurs, we consume one copy of X_1, then produce $A_1 Z_1$ and $B_1 Z_2$ for the n_1 coefficient part of $\varphi_A(\mathbf{x})$ and $\varphi_B(\mathbf{x})$ respectively. Meanwhile, we change the state value l_x to the correct value. That is to say, the component state l_x moves between $lower\ boundary\ line$, $upper\ boundary\ line$, and the fissure line. Similarly, the second reaction does the same thing for X_2. The state species is not initially present. It will be produced in the first reaction in the Z-consuming reactions section, and we only have a state specie. The state component is shared with all the reactions in this state specie.

doi:10.6342/NTU202300354

3.2.3 Z-consuming reactions



Z-consuming reactions consume the Z_1 and Z_2 and change the value of state component d. The first Z-consuming reaction below is the starting point of the output oblivious CRNs. It produces the state species, the reactant of almost all reactions here.

$$L' + A_0Z_1 + B_0Z_2 \to [A_0 - B_0, A_0 - B_0, min\{A_0, B_0\} - d_{A_0-B_0}]$$

In fact, we can view Z_1 as the value of $\varphi_A(\mathbf{x}')$,where $\mathbf{x}'$ is the number of X_1 and X_2 that have been consumed. Similarly, we also can view Z_2 as the quantities of $\varphi_B(\mathbf{x}')$. Thus, it initially consumes the A_0Z_1 and B_0Z_2 to produce the state specie by updating the state component values. Once we obtain the state specie, we can start to consume the Z_1 and Z_2 in other reactions involved.

Now, we introduce the remaining two Z-consuming reactions.

$$[l_x, l_z, d] + Z_1 \to [l_x, l_z + 1, d + d^+_{l_z}], -K < l_z < k, d \le d_{max}$$

$$[l_x, l_z, d] + Z_2 \to [l_x, l_z - 1, d + d^-_{l_z}], -k < l_z < K, d \le d_{max},$$

where $d^+_{l_z} = \begin{cases} d_{l_z} - d_{l_z+1}, & l_z \ge 0 \\ d_{l_z} - d_{l_z+1} + 1, & l_z < 0 \end{cases}$ and $d^-_{l_z} = \begin{cases} d_{l_z} - d_{l_z-1}, & l_z > 0 \\ d_{l_z} - d_{l_z+1} + 1, & l_z \le 0 \end{cases}$

When any one of the two reactions above occurs, we update the remaining two state components, l_z and d, to keep track of which fissure line or boundary line contains the $M^{-1}(\mathbf{z})$, where $\mathbf{z} = (z_1, z_2)$ is the number of Z_1 and Z_2 that have been consumed so far. As described above, Z_1 can be viewed as the quantities of φ_A. Intuitively, when one Z_1 is consumed, l_z plus 1. It means the amount of $\varphi_A - \varphi_B$ plus 1, so the line l_z contains

doi:10.6342/NTU202300354

the $M^{-1}(\mathbf{z})$ also plus 1. The second reaction handles the work like the first reaction. The only thing different is Z_2 represents the quantities of φ_B.

The last component to be explained is $d_{l_z}^+$, which appears only when Z_1 is consumed in advance. "$d_{l_z} - d_{l_z+1}$" means that when the line l_z contains $M^{-1}(\mathbf{z})$ move to $l_z + 1$, we should plus d_{l_z} and minus d_{l_z+1}, because the d_i also changes to different value when the l_z changes. We focus on the two functions φ_A and φ_B in the partial fissure function. For the constant part "+1" in $d_{l_z}^+$, $l_z < 0$ means $\varphi_A < \varphi_B$. And we are computing the function $f(\mathbf{n}) = min\{\varphi_A(\mathbf{n}) - \varphi_B(\mathbf{n})\} - d_i$, so when we increase the φ_A and $\varphi_A < \varphi_B$, then the number of d should increase by 1. Similarly, the $d_{l_z}^-$ almost do the same thing but for Z_2. Precisely, the value of d, which will be transferred into output specie Y, should increase by 1 when we increase the small one between φ_A and φ_B.

3.2.4 Y-producing reactions

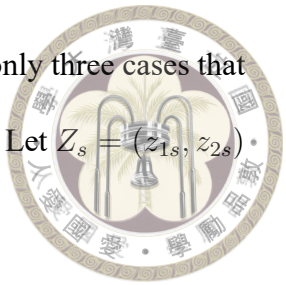
Y-producing reactions transfer the d into output species Y. The first Y-producing reaction handles the condition that d is strictly greater than d_{max}. If the condition occurs, we can just consume the part beyond the d_{max}. That is to say, we have d_{max} specie of d which is reversed for handling the line l_x moves between the fissure lines. Otherwise, we may overproduce the output Y.

$$[*, *, d] \to [*, *, d_{max}] + (d - d_{max})Y, \text{if } d > d_{max}$$

Before describing the last three Y-producing reactions, we describe what Z-stalls means. [3] says that Z-consumption stalls if none of the Z-consuming reactions are ever applicable again. The first Y-producing reaction ensures that d is too large and can never be

doi:10.6342/NTU202300354

a condition so that the Z-consuming reaction stalls. Hence, there are only three cases that result in stalling the Z-consuming reaction. We will show this below. Let $Z_s = (z_{1s}, z_{2s})$ be the count of (Z_1,Z_2) consumed when Z-consuming reactions stall.



1. The first condition is straightforward. All copies of Z_1 and Z_2 have been consumed directly. At the same time, all X_1 and X_2 have been consumed, so no more (Z_1, Z_2) will be produced.Thus, $z_s = (\varphi_A(\mathbf{n}), \varphi_B(\mathbf{n}))$.

2. All copies of Z_2 have been consumed and no more Z_2 will be produced. In this case, if the Z-consuming reactions stall, $M^{-1}(z_s)$ must be on the lower boundary lines. Meanwhile, $z_{2s} = \varphi_B(\mathbf{n})$ but $z_{1s} < \varphi_A(\mathbf{n})$. This information tells us that $f(M^{-1}(z_s)) = f(\mathbf{n})$ because l_z is on the lower boundary lines which means that $z_{1s} > z_{2s}$. So no matter how many Z_1 will be consumed in the future, it didn't influence $f(\mathbf{n})$ at all.

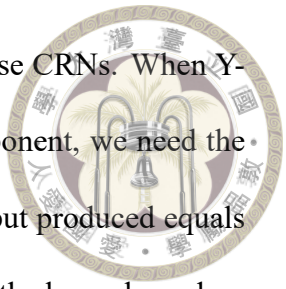
3. All copies of Z_1 have been consumed and no more Z_1 will be produced. In this case, if the Z-consuming reactions stall, $M^{-1}(z_s)$ must be on the upper boundary lines.So $z_{1s} = \varphi_A(\mathbf{n})$ but $z_{2s} < \varphi_B(\mathbf{n})$.Meanwhile $z_{2s} > z_{1s}$. So it is like case 2. No matter how many Z_2 will be consumed, it didn't influence the output $f(\mathbf{n})$.

According to the three cases above, we can know $f(M^{-1}(z_s)) = f(\mathbf{n})$.

Now we can describe the last three Y-producing reactions. The next reaction consumes the positive d and produces the output when the lines l_x and l_z lie on the same fissure line. It means the function minus the correct d_i, so we can just consume the remaining positive d.

$$[l_x, l_z, d] \to [l_x, l_z, 0] + dY, \text{if } -k < l_x = l_z < k$$

doi:10.6342/NTU202300354



The last two Y-producing reactions play an important role in these CRNs. When Y-producing reactions stalls and we have a positive d in the state component, we need the two reactions to help us clear all the positive d to ensure that the output produced equals $f(\mathbf{n})$. The next Y-producing reactions handles the cases $M^{-1}(\mathbf{z})$ is on the lower boundary line and $m^{-1}(\mathbf{z}+(r,0))$, $r>0$,lies on the line L_l where $l = l_x \bmod 2K-1, d>0$. Let r_1 be the smallest such integer.

$$[l_x, l_z, d] + r_1 Z_1 \to [l_x, l_z, 0] + r_1 Z_1 + dY, l_z >= k, l_z + r_1 = l_x \mod 2K-1, d>0.$$

The last reaction does the same thing as the previous one. r_2 also has a similar definition to r_1. It handles the case that $M^{-1}(\mathbf{z})$ lies on the upper boundary line.

$$[l_x, l_z, d] + r_2 Z_2 \to [l_x, l_z, 0] + r_2 Z_2 + dY, l_z >= k, l_z + r_2 = l_x \mod 2K-1, d>0.$$

The state component d may be negative at certain moments because of the randomness of CRNs. All the reactions above handle the case when d is positive. Hence, it seems to have a problem in the case that d is negative, then the system may be stopped. All the function in this paper is increasing. Thus, even if there is a certain moment that $d<0$, d will be positive during the execution process of other CRNs because there must have some Z_1 or Z_2 left in this case. The Z-consuming reactions will increase the d. At the end, d must be greater than or equal to 0 and be consumed.

3.3 Stitching

Stitching is to combine either the partial grid affine function or the partial fissure function into a single function whose domain is N^2. In this section, let the output oblivious

doi:10.6342/NTU202300354

function be composed of either finite partial grid affine functions or finite partial fissure functions, says $f_1, f_2, ..., f_m$, whose domains are grids. All the Dom_j of f_j are grids that are one of the one-dimensional, two-dimensional, or finite points, for $1 \leq j \leq m$. Let the offset of the j-th grid is $\mathbf{o_j} = (o_{j1}, o_{j2})$. On the input $\mathbf{n}$, $\mathbf{n_j} = \mathbf{n}$ if $\mathbf{n} \in Dom_j$, otherwise $\mathbf{n_j} \geq \mathbf{n}$. If we can produce the point $\mathbf{n_j}$ which is the smallest one that is greater than or equal to $\mathbf{n}$, then the function can be merged easily. If $\mathbf{n} \notin \mathbf{n_j}$, then $\mathbf{n_j} \geq \mathbf{n}$. If $\mathbf{n} \in \mathbf{n_j}$, then $\mathbf{n_j} = \mathbf{n}$. Let $y_j = f_j(\mathbf{n_j})$. Our construction produces m "output" for every input $\mathbf{n_j}$, for $1 \leq j \leq m$. All the function in this paper are increasing, $y_j = f_j(\mathbf{n_j}) = f_j(\mathbf{n}) = f(\mathbf{n})$ if $\mathbf{n} \in Dom_j$, otherwise $f(\mathbf{n_j}) \geq f(\mathbf{n})$ because $\mathbf{n_j} \geq \mathbf{n}$. We only need to compute $\min\{y_1, y_2, ..., y_m\} = y = f(\mathbf{n})$.

Let the $(a_1^{'}, 0)$ and $(0, a_2^{'})$ be the base case of dom_j, and let $\mathbf{o}^{'} = (o_1^{'}, o_2^{'})$ be its offset. The following nine reaction produce the $\mathbf{n_j} = \mathbf{n}^{'} = (n_1^{'}, n_2^{'})$ copies of $(X_1^{'}, X_2^{'})$ where $\mathbf{n}^{'}$ is the smallest one of Dom_j such that $\mathbf{n} \leq \mathbf{n}^{'}$ from a leader.

$$L^{'} \rightarrow L_{00}^{'} + o_1^{'}X_1^{'} + o_2^{'}X_2^{'}$$

$$L_{0b}^{'} + (o_1^{'} + 1)X_1 \rightarrow L_{1b}^{'} + a_1^{'}X_1^{'}, \text{ for b = 0 and b = 1}$$

$$L_{b0}^{'} + (o_2^{'} + 1)X_2 \rightarrow L_{b1}^{'} + a_2^{'}X_2^{'}, \text{ for b = 0 and b = 1}$$

$$L_{1b}^{'} + a_1^{'}X_1 \rightarrow L_{1b}^{'} + a_1^{'}X_1^{'}, \text{ for b = 0 and b = 1}$$

$$L_{b1}^{'} + a_2^{'}X_2 \rightarrow L_{b1}^{'} + a_2^{'}X_2^{'}, \text{ for b = 0 and b = 1}$$

Because we have m functions, we first copy m "input" for every partial grid affine function in order to produce all inputs $\mathbf{n^1}, \mathbf{n^2}, ..., \mathbf{n^m}$. Three additional reactions is needed

doi:10.6342/NTU202300354

here.

$$L \to L_1' + L_2' + ... + L_m'$$

$$X_1 \to X_1^1 + X_1^2 + ... + X_1^m$$

$$X_2 \to X_2^1 + X_2^2 + ... + X_2^m$$

Everything is fine if all the grids are two-dimensional. The partial fissure function satisfies the condition so that we can merge all the partial fissure functions by the reaction below.

$$Y_1 + Y_2 + ... + Y_m \to Y$$

However, we have the partial grid affine function whose domain may be a point, line or wedge on the grid. Input **n** may be out of the domain of a partial grid function f_j.This is possible to lead to a wrong output. So we have to detect if the input **n** is in the domain of f_j or not. Meanwhile, modifying part of the construction.

We add additional reactions to the partial grid affine function and generate the new specie S' when input **n** is not in Dom_j. Dom_j is a line or point which means it contains no one or two of base vectors of $(a_1', 0)$ and $(0, a_2')$.

If Dom_j contains no $(a_1', 0)$.We add

$$L_{1b}' \to S', \text{ for b = 0 and b = 1}$$

and remove the reaction involving in a_1'.

doi:10.6342/NTU202300354

If Dom_j contains no $(0, a_2^{'})$.We add

$$L_{b1}^{'} \to S^{'}, \text{ for b = 0 and b = 1}$$

and remove the reaction involving in $a_2^{'}$. Moreover, S' is produced $\iff$ Dom_j has no point $\mathbf{n}^{'}$ such that $\mathbf{n}^{'} \geq \mathbf{n}$.

We can make sure that y_j is min of Dom_j if Dom_j contains n^j. Otherwise, it will produce the specie $S^{'}$. To merge all the partial grid affine functions, we replace $Y_1 + Y_2 + ... + Y_m \to Y$ to 2^m new reactions. One for each subset I of $\{1, 2, ..., m\}$, which means there is some i missed. The new reaction corresponding to I is described below.

$$Y_1^{'} + Y_2^{'} + ... + Y_m^{'} \to Y + Y_1^{''} + Y_2^{''} + ... + Y_m^{''}$$

where $Y_j^{'} = Y_j$ if $j \in I$ and $Y_j^{'} = S_j$ otherwise.$Y^{''}$ is inactive specie if $j \in I$.Otherwise $Y_j^{''} = S_j$.

In fact, we can view S_j as an infinite element. If Dom_j didn't contains the $\mathbf{n_j}$ such that $\mathbf{n_j} \geq \mathbf{n}$. The partial grid affine CRNs may have an unpredictable output $y^{''}$ which may lead to a wrong output of the complete function. Let $y^{'}$ be the output of f_j whose domain contains input $\mathbf{n}$. If $y^{''}$ is less than $y^{'}$, then the "true" output is changed to the unpredictable output $y^{''}$. When this condition occurs, the specie S_j can be the alternative output specie for the unpredictable output $y^{''}$ that should be consumed, but didn't consume.

doi:10.6342/NTU202300354

doi:10.6342/NTU202300354

Chapter 4 Output oblivious CRNs which compute efficiently

This chapter is the main result of this thesis. In this chapter, we will list the construction from [3] in section 3 again, then show how to design new reactions which compute the same functions of the original reactions but converges to the correct output much faster. In section 4.1 we construct CRNs that compute partial affine functions. In section 4.2 and section 4.3, we do the same on partial fissure functions and stitching parts respectively.

The slow part of previous constructions is the reactions with many reactants. For instance, if we have a tri-molecular reaction $A + A + A \to Y$. When we only have the constant number of molecule A left, the propensity is $\frac{\text{number of} A}{N^2}$, where N is the volume. Hence, the last execution needs $O(N^2)$ time.

We convert all previous reactions to single-molecular and bi-molecular reactions and make sure that the racing conditions between different reactions do not affect the speed of computation.

doi:10.6342/NTU202300354

4.1 Partial affine function

First, we will show the construction in section 3.1, then modify the CRNs if they need to be modified to make sure that the speed wouldn't too slow. Precisely, the reaction should be modified when it is not a bi-molecular or single-molecular reaction.

Lemma 3. *Partial affine function can be computed by output oblivious CRNs in O(N log N) time.*

Proof. The constructions from section 3.1 are shown below.

$$L + o_1X_1 + o_2X_2 \to L^{'} + (a_1o_1 + a_2o_2 + a_0)Y \tag{4.1}$$

$$L' + pX_1 \to L^{'} + a_1pY \tag{4.2}$$

$$L' + qX_2 \to L^{'} + a_2qY \tag{4.3}$$

For the reaction 4.1, we have to deal with the o_1X_1 and o_2X_2 respectively. Because we want to make the construction not too slow, we can't have any reaction which is not a bi-molecular reaction. Hence, we have to divide the part "o_1X_1" of reaction 4.1 into a few reactions shown below. X_{1_i} plays the same role as i copies of X_1, but it is a single molecule.

Before we handle the "o_1X_1" part of reaction 4.1, We prove lemma 4 and 5 which are frequently used afterward first.

Lemma 4. *If we have a reaction with a leader L and N copies of the molecule X to be the reactant and produce the Z and the leader, which is not identical to X, then it can finish in $O(N \log N)$ time, where N is the volume. The reaction would not be stopped by other*

doi:10.6342/NTU202300354

reactions, and other reactions would not produce X.

Proof. When the reaction occurs once, the number of the molecular X decrease by 1. Hence, the time it needs can be calculated by the following equation $\sum_{i=1}^{N} \frac{N}{i} = O(N \log N)$ because the propensity is $\geq \frac{i}{N}$ and the time of reaction occurs is $\leq \frac{N}{i}$.

□

Lemma 5. *If we have the form of reactions shown below, where $c \in N$, then the time to finish these reactions is $O(N \log N)$, where N is the volume. The reactions would not be stopped by other reactions, and other reactions would not produce X_i. i and j is less than c.*

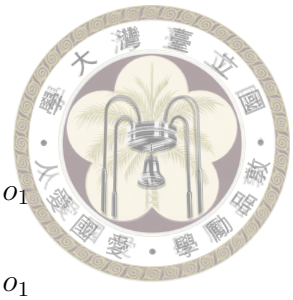
$$X_{1\_i} + X_{1\_j} \rightarrow \begin{cases} X_{1\_c} + X_{1\_(i+j-c)}, & \text{if } i+j > c \\ X_{1\_(i+j)}, & \text{if } i+j < c \\ X_{1\_c}, & \text{if } i+j = c \end{cases} \tag{4.4}$$

Proof. The proof is the same as lemma 4. When the reaction occurs once, the number of the molecular X decrease by 1. Hence, the time it needs can be calculated by the following equation $\sum_{i=1}^{N} \frac{N}{i} = O(N \log N)$ because the propensity is $\geq \frac{i}{N}$ and the time of reaction occurs is $\leq \frac{N}{i}$.

□

doi:10.6342/NTU202300354

For all i and j less than o_1, we have reactions below.



$$X_{1\_i} + X_{1\_j} \rightarrow \begin{cases} X_{1\_o_1} + X_{1\_(i+j-o_1)}, & \text{if } i+j > o_1 \\ X_{1\_(i+j)}, & \text{if } i+j < o_1 \\ X_{1\_o_1}, & \text{if } i+j = o_1 \end{cases}$$

We merge every single X_1 to a large X_{1_i} in order to get $X_{1\_o_1}$ which means we have $o_1 X_1$. No matter which one of the reactions above occurs, the quantities of X_1 must be decreased by 1. So we will get $X_{1\_o_1}$ in expected time $O(N \log N)$ by lemma 5. For the part "$o_2 X_2$" of reaction 4.1, we also do the same thing on $o_2 X_2$. X_{2_i} has a similar definition like X_{1_i}. We separate the part "$o_2 X_2$" of reaction 4.1 into a few reactions. For all i and j less than o_2, we have reactions below.

$$X_{2\_i} + X_{2\_j} \rightarrow \begin{cases} X_{2\_o_2} + X_{2\_(i+j-o_2)}, & \text{if } i+j > o_2 \\ X_{2\_(i+j)}, & \text{if } i+j < o_2 \\ X_{2\_o_2}, & \text{if } i+j = o_2 \end{cases}$$

The reactions above have the same time complexity as the part "$o_1 X_1$". All the reactions finish in the expected time $O(N \log N)$. After $X_{1\_o_1}$ and $X_{2\_o_2}$ are produced, we add the last reaction to make the CRNs above equivalent to the reaction 4.1.

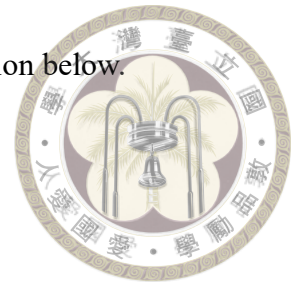
$$L + X_{1\_o_1} \rightarrow L'',$$

$$L'' + X_{2\_o_2} \rightarrow L_3' + (a_1 o_1 + a_2 o_2 + a_0) Y.$$

L_3' is used to transfer X_{1_i} and X_{2_j} into X_{1_i}' and X_{2_j}' respectively, $1 \leq i \leq o_1$ and $1 \leq j \leq o_2$, in order to enables the reaction 4.2 and reaction 4.3. X_{1_i}' and X_{2_j}' will be

doi:10.6342/NTU202300354

consumed in the reaction 4.2 and 4.3 respectively. We have the reaction below.



$$X_{1\_i} + L_3^{'} \to L_3^{'} + X_{1\_i}^{'},\ 1 \le i \le o_1$$

$$X_{2\_j} + L_3^{'} \to L_3^{'} + X_{2\_j}^{'},\ 1 \le j \le o_2$$

$$X_{1\_i}^{'} \to iX_1^{'},\ 1 \le i \le o_1$$

$$X_{2\_j}^{'} \to jX_2^{'},\ 1 \le j \le o_2.$$

The time complexity of the two reactions above is $O(N \log N)$ by lemma 4.

Now, we reconstruct the reaction 4.2 into the following reactions. We have to produce an output species Y when we get $pX_1^{'}$ together.

$$X_{1\_i}^{'} + X_{1\_j}^{'} \to \begin{cases} a_1 pY + X_{1\_(i+j-p)}^{'}, & \text{if } i+j > p \\ X_{1\_(i+j)}^{'}, & \text{if } i+j < p \\ a_1 pY, & \text{if } i+j = p \end{cases}$$

And for reaction 4.3, we have similar reactions like reaction 4.2. The only thing different is that p turns into q.

$$X_{2\_i}^{'} + X_{2\_j}^{'} \to \begin{cases} a_2 qY + X_{2\_(i+j-q)}^{'}, & \text{if } i+j > q \\ X_{2\_(i+j)}^{'}, & \text{if } i+j < q \\ a_2 qY, & \text{if } i+j = q \end{cases}$$

Reaction 4.2 and reaction 4.3 have the same time complexity as the "$o_1 X_1$" part of reaction4.1.

doi:10.6342/NTU202300354

We have reconstructed all the reactions in section 3.1 and make sure all the time complexity of the reactions we reconstructed is $O(N \log N)$, so we have proven that the speed of computation of partial affine function is $O(N \log N)$. □

The time complexity of original construction from [3] is $O(N^{r-1})$ where r is a constant value which could be p or q. p and q are the periods of the base vectors. We get a significant improvement from reconstructing the construction.

4.2 Partial fissure function

In this section, we focus on constructing CRNs to compute the partial fissure function and analyze the time to compute the target functions. We don't divide the constructions into three parts, which are Z-producing reactions, Z-consuming reactions, and Y-producing reactions, like section 3. Z-producing reactions transfer the input molecules X_1 and X_2 into Z_1 and Z_2 respectively. Z-consuming reactions transfer the molecules Z_1 and Z_2 into the state component d. Y-producing reactions transfer the state component d into the output species Y.

Lemma 6. *A partial fissure function can be computed by output oblivious CRNs in $O(N \log N)$ time.*

Proof. We handle the reactions described in section 3 one by one. First, our system only has a leader L initially. We don't need to modify the Reaction 4.5 because it is a single molecular reaction. The propensity of reaction 4.5 is 1 and no race condition, so we only

doi:10.6342/NTU202300354

need constant time to finish it.

$$L \rightarrow L^{'} + A_0 Z_1 + B_0 Z_2 \tag{4.5}$$

$$L^{'} + A_0 Z_1 + B_0 Z_2 \rightarrow [A_0 - B_0, A_0 - B_0, min\{A_0, B_0\} - d_{A_0 - B_0}] \tag{4.6}$$

The reaction 4.6 is a high-order reaction. We have to modify it to be bi-molecular reactions which are shown below.

For the "$A_0 Z_1$" part, we want to get the species Z_{A_0} which plays the same role as A_0 copies of Z_1. For all i and j less than A_0, we have reactions below.

$$Z_{1\_i} + Z_{1\_j} \rightarrow \begin{cases} Z_{1\_A_0} + Z_{1\_(i+j-A_0)}, & \text{if } i+j > A_0 \\ Z_{1\_(i+j)}, & \text{if } i+j < A_0 \\ Z_{1\_A_0}, & \text{if } i+j = A_0 \end{cases}$$

For the "$B_0 Z_2$" part, we do the same thing as the "$A_0 Z_1$" part. For all i and j less than B_0, we have reactions below.

$$Z_{2\_i} + Z_{2\_j} \rightarrow \begin{cases} Z_{2\_B_0} + Z_{2\_(i+j-A_0)}, & \text{if } i+j > B_0 \\ Z_{2\_(i+j)}, & \text{if } i+j < B_0 \\ Z_{2\_A_0}, & \text{if } i+j = B_0 \end{cases}$$

By the lemma 5, the "$A_0 Z_1$" part and "$B_0 Z_2$" part can be finished in $O(N \log N)$, the N is the volume of current state. In order to make the reactions equivalent to reaction 4.6, we have reactions below to make them have the same function. The first two reactions can finish in $O(N)$ because their propensities both are $\frac{1}{N}$ and they only occur once. The

doi:10.6342/NTU202300354

last two reactions can finish in $O(N \log N)$ time by lemma 4.

$$L^{'} + Z_{1\_A_0} \rightarrow L^{''},$$

$$L^{''} + Z_{2\_B_0} \rightarrow [A_0 - B_0, A_0 - B_0, \min\{A_0, B_0\} - d_{A_0 - B_0}] + L^{''},$$

$$L^{''} + Z_{1\_i} \rightarrow L^{''} + Z^{'}_{1\_i}, \text{ for } 1 \leq i \leq A_0,$$

$$L^{''} + Z_{2\_i} \rightarrow L^{''} + Z^{'}_{2\_i}, \text{ for } 1 \leq i \leq B_0,$$

$$Z^{'}_{1\_i} \rightarrow iZ^{'}_{1}, \text{ for } 1 \leq i \leq A_0,$$

$$Z^{'}_{2\_j} \rightarrow jZ^{'}_{2}, \text{ for } 1 \leq j \leq B_0,$$

We merge all the molecules of $Z^{'}_{1}$ and $Z^{'}_{2}$ into $Z^{'}_{1\_2K-1}$ and $Z^{'}_{2\_2K-1}$ by the following reactions respectively. The molecules comprised will be the reactant of the latter reaction. K is described in the section 3.2. For all i and j less than $2K - 1$, we have reactions below.

$$Z^{'}_{1\_i} + Z^{'}_{1\_j} \rightarrow \begin{cases} Z^{'}_{1\_2K-1} + Z^{'}_{1\_(i+j-2K-1)}, & \text{if } i + j > 2K - 1 \\ Z^{'}_{1\_(i+j)}, & \text{if } i + j < 2K - 1 \\ Z^{'}_{1\_2K-1}, & \text{if } i + j = 2K - 1 \end{cases}$$

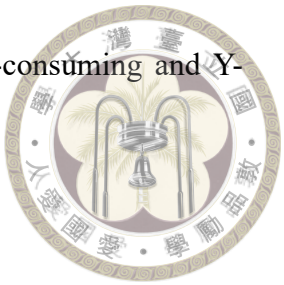
and

$$Z^{'}_{2\_i} + Z^{'}_{2\_j} \rightarrow \begin{cases} Z^{'}_{2\_2K-1} + Z^{'}_{2\_(i+j-2K-1)}, & \text{if } i + j > 2K - 1 \\ Z^{'}_{2\_(i+j)}, & \text{if } i + j < 2K - 1 \\ Z^{'}_{2\_2K-1}, & \text{if } i + j = 2K - 1 \end{cases}$$

By the lemma 5, these reaction can finish in $O(N \log N)$.

doi:10.6342/NTU202300354

Next, we will analyze the time complexity of Z-producing, Z-consuming and Y-producing reactions from [3] at the same time. We list them all first.



$$[l_x, *, *] + X_1 \to [l_x + (A_1 - B_1) \mod 2K - 1, *, *] + A_1 Z_1 + B_1 Z_2 \tag{4.7}$$

$$[l_x, *, *] + X_2 \to [l_x + (A_2 - B_2) \mod 2K - 1, *, *] + A_2 Z_1 + B_2 Z_2 \tag{4.8}$$

$$[l_x, l_z, d] + Z_1 \to [l_x, l_z + 1, d + d_{l_z}^+], -K < l_z < k, d \leq d_{max} \tag{4.9}$$

$$[l_x, l_z, d] + Z_2 \to [l_x, l_z - 1, d + d_{l_z}^-], -k < l_z < K, d \leq d_{max} \tag{4.10}$$

$$[l_x, l_z, d] + r_1 Z_1 \to [l_x, l_z, 0] + r_1 Z_1 + dY, l_z \geq k, l_z + r_1 = l_x \mod 2K - 1, d > 0 \tag{4.11}$$

$$[l_x, l_z, d] + r_2 Z_2 \to [l_x, l_z, 0] + r_2 Z_2 + dY, l_z \geq k, l_z + r_2 = l_x \mod 2K - 1, d > 0 \tag{4.12}$$

$$[l_x, l_z, d] \to [l_x, l_z, 0] + dY, \text{if } -k < l_x = l_z < k \tag{4.13}$$

$$[*, *, d] \to [*, *, d_{max}] + (d - d_{max})Y, \text{if } d > d_{max} \tag{4.14}$$

For the Z-producing reactions 4.7, 4.8, it finish in $O(N \log N)$ originally by the lemma 4. For the reaction 4.8, it is completely the same as the reaction 4.7. That is, the Z-producing reactions can finish in $O(N \log N)$.

Here we construct new reaction which is equivalent to the reaction 4.9, 4.10, 4.11 and 4.12. Because they are equivalent, so the execution sequence must be the same. For

doi:10.6342/NTU202300354

all $i \leq 2K-1$, we have reactions below which is equivalent to the reaction 4.9.

$$[l_x, l_z, d] + Z'_{1\_i} \rightarrow \begin{cases} [l_x, l_z+1, d+d_{l_z}^+] + Z'_{1\_i-1}, & \text{if } i > 1 \\ [l_x, l_z+1, d+d_{l_z}^+], & \text{if } i = 1 \end{cases}$$

For all $i \leq 2K-1$, we have reactions below which is equivalent to the reaction 4.10.

$$[l_x, l_z, d] + Z'_{2\_i} \rightarrow \begin{cases} [l_x, l_z+1, d+d_{l_z}^-] + Z'_{2\_i-1}, & \text{if } i > 1 \\ [l_x, l_z+1, d+d_{l_z}^-], & \text{if } i = 1 \end{cases}$$

For all $i \leq 2K-1$, we have reactions below which are equivalent to the reaction 4.11. The reactions subject to the condition that $l_z \geq k, l_z + r_1 = l_x \mod 2K-1, d > 0$.

$$[l_x, l_z, d] + Z'_{1\_i} \rightarrow \begin{cases} [l_x, l_z, 0] + Z'_{1\_i-r_1}, & \text{if } i > r_1 \\ [l_x, l_z, 0], & \text{if } i = r_1 \end{cases}$$

For all $i \leq 2K-1$, we have reactions below which are equivalent to the reaction 4.12. The reactions subject to the condition that $l_z \geq k, l_z + r_2 = l_x \mod 2K-1, d > 0$.

$$[l_x, l_z, d] + Z'_{2\_i} \rightarrow \begin{cases} [l_x, l_z, 0] + Z'_{2\_i-r_2}, & \text{if } i > r_2 \\ [l_x, l_z, 0], & \text{if } i = r_2 \end{cases}$$

we start to analyze the time complexity. In this part, we assume that the reaction 4.7 and 4.8 have finished and the time is $O(N \log N)$ by the lemma 4. This would not make the speed of the system slow because the reaction 4.7 and 4.8 consume the molecular X_1 and X_2 only and no other reactions produce X_1 and X_2. No matter how many reactions

doi:10.6342/NTU202300354

occur in the process of the execution of the reaction 4.7 and 4.8, it only speeds up the execution of the system. By the assumption, we only have species of Z_{1_i} and Z_{2_i} left in the system. That is, we only consider the reaction 4.9, 4.10, 4.11 and 4.12 now.

The reaction 4.11 and 4.12 is used to avoid the condition that the Z-consuming reaction stalls. Hence, the reaction 4.11 and 4.12 either occurs once or didn't happen under the assumption. For the reaction 4.11, at the worst case, we have r_1 copies of Z_{1_1} and the time to merge them into $Z_{1\_r_1}$ is $O(r_1 N)$.The reaction 4.12 has similar analysis and the same time.

We only have the reaction 4.9, 4.10 left now. Let N be the current number of molecules in the system. Without loss of generality, we assume that $n_1 < n_2$ where n_1 and n_2 are numbers of Z_1 and Z_2 respectively. When the reaction 4.9 occurs, the number of molecular Z_1 decreases by 1. Hence the time it needs can be computed as the equation $\sum_{i=1}^{N}(2K-1)\frac{N}{i} = O(N \log N)$ if the reaction 4.9 would not be stopped by the constraint. If the reaction 4.9 is stopped by the constraint which means it stalls on the lower boundary line, then it is enabled after the reaction 4.10 occurs once. The number of Z_2 is larger than the number of Z_1, so the time for the reaction 4.10 occurs once, must be faster or equal to the reaction 4.9 at any moment. Once the $n_1 = 0$, then the reaction 4.9 and 4.10 finish. That is, the time of the reaction 4.9 at most two times the speed of the $O(N \log N)$. In the process of the reaction 4.9, 4.10 execution, d may be larger than d_{max}. This condition may stall the system. The reaction 4.14 handles the condition and only needs constant time because it is a single molecular reaction. The system at most stalls N times, so the total cost of the reaction 4.14 is $O(N)$. The reaction 4.13 is also a single molecular reaction and occurs at most N time, so the total time of the reaction 4.13 is also $O(N)$. We have proved all the reactions of the partial fissure function can be computed in $O(N \log N)$ time so

doi:10.6342/NTU202300354

far. □

4.3 Stitching

In this section, we construct the CRNs to combine the output of either the partial grid affine function or the partial fissure function into a proper output. We focus on the construction in the section. The correctness of the reactions is guaranteed by [3], and the details are shown in section3. The new reactions we create are equivalent to the original reaction in [3], so the two CRNs have the same execution sequence.

Lemma 7. *Stitching part of output oblivious CRNs can finish in $O(NlogN)$ time.*

Proof. First, the following reactions from [3] are listed in section 3.3, we will reconstruct the reactions one by one. The detection part is used to produce the $S^{'}$ species when the partial fissure function lacks one of the base vectors of the x-axis or the y-axis.

$$L^{'} \rightarrow L^{'}_{00} + o^{'}_{1}X^{'}_{1} + o^{'}_{2}X^{'}_{2} \tag{4.15}$$

$$L^{'}_{0b} + (o^{'}_{1} + 1)X_1 \rightarrow L^{'}_{1b} + a^{'}_{1}X^{'}_{1}, \text{ for b = 0 and b = 1} \tag{4.16}$$

$$L^{'}_{b0} + (o^{'}_{2} + 1)X_2 \rightarrow L^{'}_{b1} + a^{'}_{2}X^{'}_{2}, \text{ for b = 0 and b = 1} \tag{4.17}$$

$$L^{'}_{1b} + a^{'}_{1}X_1 \rightarrow L^{'}_{1b} + a^{'}_{1}X^{'}_{1}, \text{ for b = 0 and b = 1} \tag{4.18}$$

$$L^{'}_{b1} + a^{'}_{2}X_2 \rightarrow L^{'}_{b1} + a^{'}_{2}X^{'}_{2}, \text{ for b = 0 and b = 1} \tag{4.19}$$

Reaction 4.16 and reaction 4.17 occur once at most. And reaction 4.18 will start to work after the reaction 4.16 occurs and produces the species $L^{'}_{1b}$. Also reaction 4.19 will start to work after the reaction 4.17 occurs and produces the species $L^{'}_{b1}$. Thus, we handles reaction 4.16 and reaction 4.17 first.

doi:10.6342/NTU202300354

Let $\mathbf{n}' = (n_1', n_2')$ is the smallest point greater than input $\mathbf{n}$ in Dom_j if it exists. For the reaction 4.16, let $w_1 = o_1' + 1$, then we have the reaction below to produce n_1', $0 \leq i < w_1$, $0 \leq j < w_1$.

$$X_{1\_i} + X_{1\_j} \rightarrow \begin{cases} X_{1\_w_1} + X_{1\_(i+j-w_1)}, & \text{if } i+j > w_1 \\ X_{1\_(i+j)}, & \text{if } i+j < w_1 \\ X_{1\_w_1}, & \text{if } i+j = w_1 \end{cases}$$

We can get the species $X_{1\_w_1}$ after expected time $O(N \log N)$ by lemma 5. Once we get the species $X_{1\_w_1}$, we have another reaction below to make these reactions equal to reaction 4.16 and enable reaction 4.18 with L_{1b}'. The reaction also finishes in expected time $O(N \log N)$.

$$L_{0b}' + X_{1\_w_1} \rightarrow L_{1b}' + a_1' X_1' + L_1'', \text{ for } b = 0 \text{ and } b = 1$$

$$L_1'' + X_{1\_i} \rightarrow X_{1\_i}'', \text{ for } 1 \leq i \leq w_1,$$

$$X_{1\_i}'' \rightarrow i X_1''. \text{ for } 1 \leq i \leq w_1,$$

For the reaction 4.18, the reconstruct constructions is showing below, $0 \leq i < a_1'$ and $0 \leq j < a_1'$.

$$X_{1\_i}'' + X_{1\_j}'' \rightarrow \begin{cases} X_{1\_a_1'}'' + X_{1\_(i+j-a_1')}'', & \text{if } i+j > a_1' \\ X_{1\_(i+j)}'', & \text{if } i+j < a_1' \\ X_{1\_a_1'}'', & \text{if } i+j = a_1' \end{cases}$$

doi:10.6342/NTU202300354

and

$$L^{'}_{1b} + X^{''}_{1\_a^{'}_1} \rightarrow L^{'}_{1b} + a^{'}_1 X^{'}_1, \text{ for } b = 0 \text{ and } b = 1,$$

The total time complexity of the reconstructed reactions above for reaction 4.16 and 4.18 is $O(N \log N)$ by lemma 5 and 4.

Now, we have the reaction 4.17 and reaction 4.19 left. they are almost the same as reaction 4.16 and reaction 4.18. Hence,we show the construction below. The reconstructed reaction for reaction 4.17 is showing below and produce $n^{'}_2$. Let $w_2 = o^{'}_2 + 1, 0 \leq i < w_2$ and $0 \leq j < w_2$.

$$X_{2\_i} + X_{2\_j} \rightarrow \begin{cases} X_{2\_w_2} + X_{2\_(i+j-w_2)}, & \text{if } i + j > w_2 \\ X_{2\_(i+j)}, & \text{if } i + j < w_2 \\ X_{2\_w_2}, & \text{if } i + j = w_2 \end{cases}$$

and

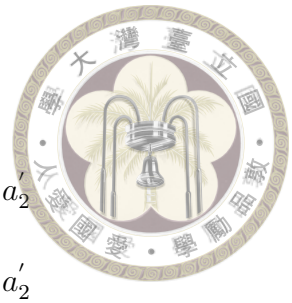
$$L^{'}_{b0} + X_{2\_w_2} \rightarrow L^{'}_{b1} + a^{'}_2 X^{'}_2 + L^{''}_2, \text{ for } b = 0 \text{ and } b = 1,$$

$$L^{''}_2 + X_{2\_i} \rightarrow X^{''}_{2\_i}, \text{ for } 1 \leq i \leq w_2,$$

$$X^{''}_{2\_i} \rightarrow i X^{''}_2. \text{ for } 1 \leq i \leq w_2,$$

doi:10.6342/NTU202300354

The reconstructed reaction for reaction 4.19 is showing below.



$$X''_{2\_i} + X''_{2\_j} \rightarrow \begin{cases} X''_{2\_a'_2} + X''_{2\_(i+j-a'_2)}, & \text{if } i+j > a'_2 \\ X''_{2\_(i+j)}, & \text{if } i+j < a'_2 \\ X''_{2\_a'_2}, & \text{if } i+j = a'_2 \end{cases}$$

and

$$L'_{b1} + X''_{2\_a'_2} \rightarrow L'_{b1} + a'_2 X'_2, \text{ for } b = 0 \text{ and } b = 1$$

The time complexity of reaction 4.17 and reaction 4.19 are equal to reaction 4.16 and reaction 4.18.

If the function f_j lacks one of the base vectors, we replace the reaction 4.17 and 4.19 with the reaction shown below respectively.

$$L'_{1b} \rightarrow S', \text{ for b = 0 and b = 1}$$

$$L'_{b1} \rightarrow S', \text{ for b = 0 and b = 1}$$

The two reactions are single molecular reactions. So they only need constant time to finish. By this, we show that the detection part from section 3.3 is $O(N \log N)$.

By the result from section 4.1 and section 4.2, we can stably compute the partial grid affine function or partial fissure function in expected time $O(N \log N)$. The remaining work is to merge the solution Y_j which is the output of function f_j.

For the partial fissure function, we need to modify the construction below in sec-

doi:10.6342/NTU202300354

tion 3.1 to make sure they are bi-molecular.

$$Y_1 + Y_2 + ... + Y_m \to Y$$

The reconstructed construction is shown below. The species $Y_{123...j}$ is the "true" output species. It means that we consume one output species from each function f_1,f_2,...,f_m and get a "true" output species.

$$Y_1 + Y_2 \to Y_{12}$$

$$Y_{12} + Y_3 \to Y_{123}$$

$$...$$

$$Y_{123...m-1} + Y_m \to Y$$

Without loss of generality, we assume that the c, number of Y_1, is smaller than others output species. Here the number of Y_2 is greater than 1, so the time complexity of the first reaction is $O(N \log N)$ by lemma 4. The time complexity of these reactions is $O(mN \log N)$, where m is the number of functions because we have m functions.

For the partial grid affine function, we transfer all the Y_j to S_j if S_j exists. And the reaction can finish in the expected time $O(N \log N)$ by lemma 4.

$$Y_j + S_j \to S_j, \text{ for } 1 \leq i \leq m$$

As described in section 3.3, we have 2^m new function, one for each subset I of $\{1, 2, ..., m\}$. The reaction corresponding with I is described below.

$$Y_1^{'} + Y_2^{'} + ... + Y_m^{'} \to Y + Y_1^{''} + Y_2^{''} + ... + Y_m^{''}$$

doi:10.6342/NTU202300354

where $Y_j' = Y_j$ if $j \in I$ and $Y_j' = S_j$ otherwise.Y'' is inactive species if $j \in I$. Otherwise $Y_j'' = S_j$.

The reconstructed reaction is shown below and it can finish in the expected time $O(mN \log N)$, m is the number of functions.

$$\begin{cases} Y_1 + Y_2 & \to Y_{12}, \\ Y_1 + S_2 & \to Y_{12} + S_2, \\ S_1 + Y_2 & \to Y_{12} + S_1, \\ S_1 + S_2 & \to S_{12}, \end{cases}$$

$$\begin{cases} Y_{12} + Y_3 & \to Y_{123}, \\ Y_{12} + S_3 & \to Y_{123} + S_3, \\ S_{12} + Y_3 & \to Y_{123} + S_{12}, \\ S_{12} + S_3 & \to S_{123}, \end{cases}$$

...

doi:10.6342/NTU202300354

$$\begin{cases} Y_{123\ldots m-1} + Y_m & \to Y, \\ Y_{123\ldots m-1} + S_m & \to Y + S_m, \\ S_{123\ldots m-1} + Y_m & \to Y + S_{123\ldots m-1}, \\ S_{123\ldots m-1} + S_m & \to S_{123\ldots m}, \end{cases}$$

We have $m-1$ similar case above, so we take the first case which has four reactions as an example to analyze the time complexity. The reaction $Y_1 + Y_2 \to Y_{12}$ can finish in $O(N \log N)$ time by lemma 4. Because the number of Y_1 or Y_2 is greater than a constant value, the propensity is greater than the condition in lemma 4. Hence the speed is faster than lemma 4.

The reaction $Y_1 + S_2 \to Y_{12} + S_2$ and $S_1 + Y_2 \to Y_{12} + S_1$ finish also in $O(N \log N)$ time can finish by lemma 4.

The reaction $S_1 + S_2 \to S_{12}$ needs $O(N)$ time to occur, but the reactions of all the cases only occur once. The total time of this reaction of all cases is $O(mN)$.

Every reaction except for the fourth reaction in cases above can finish in $(N \log N)$ and the total time for all the fourth reactions in cases above is $O(mN)$, so the time complexity of the CRN is $O(mN \log N) = O(N \log N)$. We finished the proof for the stitching part that it can finish in the expected time $O(N \log N)$. By the lemma 3, 6 and 7, we show that the output oblivious CRNs can finish in expected time $O(N \log N)$. □

doi:10.6342/NTU202300354

Chapter 5 Conclusion

We have proved that all functions $f : N^2 \to N$ can be stably computed by output oblivious chemical reaction networks(CRNs) with a leader in $O(N \log N)$ time. Compared to the original construction in [3], it is a significant improvement that reduces the time from $O(N^r)$ to $O(N \log N)$.

The ability of computation of output oblivious CRNs has been completely proven. Severson et al. [11] prove all functions $f : N^d \to N$ can be computed by output oblivious CRNs with a leader. Hashemi et al. [8] prove all functions $f : N^d \to N$ can be computed by output oblivious CRNs without a leader.

The future work for us is to prove how fast the function $f : N^d \to N$ can be computed by an output oblivious CRNs in these two cases.

$$X_{1\_i} + X_{1\_j} \to \begin{cases} X_{1\_p} + X_{1\_(i+j-p)}, & \text{if } i+j > p \\ X_{1\_(i+j)}, & \text{if } i+j < p \\ X_{1\_p}, & \text{if } i+j = p \end{cases} \tag{5.1}$$

doi:10.6342/NTU202300354

doi:10.6342/NTU202300354

References

[1] Alistarh, Dan, Gelashvili, Rati, and M. Vojnović. Fast and exact majority in population protocols. In Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, pages 47–56, 2018.

[2] A. Case, J. H. Lutz, and D. M. Stull. Reachability problems for continuous chemical reaction networks. In Natural Computing, volume 17, pages 223–230, 2018.

[3] B. Chugg, H. Hashemi, and A. Condon. Output-oblivious stochastic chemical reaction networks. In J. Cao, F. Ellen, L. Rodrigues, and B. Ferreira, editors, 22nd International Conference on Principles of Distributed Systems (OPODIS 2018), volume 125 of Leibniz International Proceedings in Informatics (LIPIcs), pages 21:1–21:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[4] A. Dana, A. James, and E. David. Stably computable predicates are semilinear. In Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, pages 292–299, 2006.

[5] A. Dana, A. James, Eisenstat, David, and R. Eric. The computational power of population protocols. In Distributed Computing, volume 20, pages 279–304, 2007.

[6] S. David, S. Georg, and W. Erik. Dna as a universal substrate for chemical kinetics.

doi:10.6342/NTU202300354

In Proceedings of the National Academy of Sciences, volume 107, pages 5393–5398, 2010.

[7] Gillespie and D. T. Exact stochastic simulation of coupled chemical reactions. In The journal of physical chemistry, volume 81, pages 2340–2361, 1997.

[8] H. Hashemi, B. Chugg, and A. Condon. Composable computation in leaderless, discrete chemical reaction networks. In C. Geary and M. J. Patitz, editors, 26th International Conference on DNA Computing and Molecular Programming (DNA 26), volume 174 of Leibniz International Proceedings in Informatics (LIPIcs), pages 3:1–3:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[9] C. Ho-Lin, D. Doty, and D. Soloveichik. Deterministic function computation with chemical reaction networks. In Natural computing, volume 13, pages 517–534. Springer, 2014.

[10] C. Ho-Lin, D. Doty, and D. Soloveichik. Rate-independent computation in continuous chemical reaction networks. In Proceedings of the 5th conference on Innovations in theoretical computer science, pages 313–326, 2014.

[11] E. E. Severson, D. Haley, and D. Doty. Composable computation in discrete chemical reaction networks. In P. Robinson and F. Ellen, editors, Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, page 14–23, 2019.

doi:10.6342/NTU202300354