Composable Computation in Discrete Chemical Reaction Networks (CRNs)

Eric E. Severson[1], David Haley[1], and David Doty[2]
eseverson@ucdavis.edu, drhaley@ucdavis.edu, doty@ucdavis.edu
1: Department of Applied Math, UC Davis 2: : Department of Computer Science, UC Davis

Full Paper on Arxiv:
https://arxiv.org/abs/1903.02637

We study CRN composition by classifying the **obliviously-computable** functions

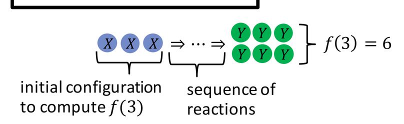
(Integer-valued functions computable by CRNs that don't use the output signal as a reactant)

Computing Functions

$f(x) = 2x$

$$\boxed{X \rightarrow 2Y}$$

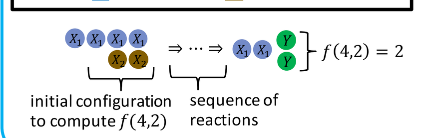
Input: X
Output: Y



$f(x_1, x_2) = \min(x_1, x_2)$

$$\boxed{X_1 + X_2 \rightarrow Y}$$

Input: X_1,X_2
Output: Y



$f(x_1, x_2) = \max(x_1, x_2)$

Input: X_1,X_2
Output: Y

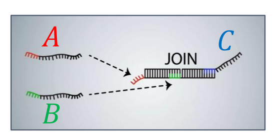
$$\begin{aligned} X_1 &\rightarrow Z_1 + Y \\ X_2 &\rightarrow Z_2 + Y \\ Z_1 + Z_2 &\rightarrow D \\ D + Y &\rightarrow \emptyset \end{aligned}$$

Works because $\max(x_1,x_2) = x_1 + x_2 - \min(x_1,x_2)$

We study Chemical Reaction Networks (CRNs) that compute integer-valued functions ($f: \mathbb{N}^d \rightarrow \mathbb{N}$).

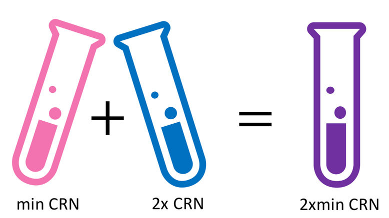
Motivation: Artificial CRNs have been implemented by technology called DNA Strand Displacement, so CRNs can be viewed as a programming language for molecular computation.

DNA Strand Displacement implementing $A + B \rightarrow C$



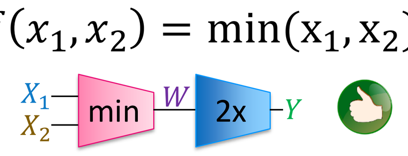
[Microsoft Research Cambridge]

Composing Functions



Goal: Modular CRN Design

$f(x_1, x_2) = \min(x_1, x_2)$

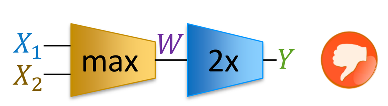


$$\boxed{\begin{aligned} X_1 + X_2 &\rightarrow W \\ W &\rightarrow 2Y \end{aligned}}$$

Input: X_1,X_2
Output: Y

output-oblivious min CRN can be composed

Fails with max CRN



max CRN is not **output-oblivious**

Lemma: This composition method (concatenating reactions & renaming species) works ⇔ upstream CRN is output-oblivious

$$\begin{aligned} X_1 &\rightarrow Z_1 + W \\ X_2 &\rightarrow Z_2 + W \\ Z_1 + Z_2 &\rightarrow D \\ D + W &\rightarrow \emptyset \\ W &\rightarrow 2Y \end{aligned}$$

reactions compete

Output-oblivious CRN: output Y is never a reactant (left side) in a reaction

Obliviously-computable function: computable by some output-oblivious CRN

Stable Computation

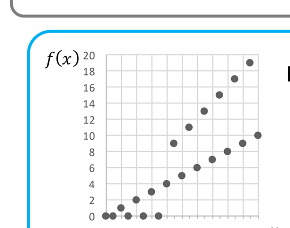
Rate-Independent Model

Model Definitions: CRNs defined by finite sets of **species** and **reactions**. To **stably compute** $f: \mathbb{N}^d \rightarrow \mathbb{N}$, define **input species** $X_1, \dots, X_d$ and **output species** Y. Encode input $\boldsymbol{x} \in \mathbb{N}^d$ with input species in the **initial configuration** (along with $1L$, the optional **leader species**). A correct **stable** configuration ($\#Y = f(\boldsymbol{x})$ can no longer change) must always be reachable by applying a sequence of reactions.

Intuitively, **stable computation** = probability 1 convergence to correct answer (no matter the rate / order of the reactions).

Theorem: $f: \mathbb{N}^d \rightarrow \mathbb{N}$ is stably computable ⇔ f is **semilinear**

[Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. Natural Computing, 2014.]



Example semilinear $f: \mathbb{N} \rightarrow \mathbb{N}$

$$f(x) = \begin{cases} x/2: x \equiv 0 \bmod 2 \\ x+1: x \equiv 1 \bmod 2, x \geq 9 \\ 0: x \equiv 1 \bmod 2, x < 9 \end{cases}$$

Piecewise Affine

Domains are **semilinear sets** (defined by thresholds / mods)

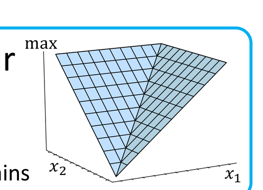
Semilinear function ($f: \mathbb{N}^d \rightarrow \mathbb{N}$) is a finite union of affine functions, whose domains are semilinear sets. A **semilinear set** ($S \in \mathbb{N}^d$) is a finite Boolean combination of **threshold** ($\boldsymbol{a} \cdot \boldsymbol{x} \geq b$ for $\boldsymbol{a} \in \mathbb{Z}^d, b \in \mathbb{Z}$) and **mod** ($\boldsymbol{a} \cdot \boldsymbol{x} \equiv b \bmod c$ for $\boldsymbol{a} \in \mathbb{Z}^d, b, c \in \mathbb{Z}_+$) sets.

max: $\mathbb{N}^2 \rightarrow \mathbb{N}$ is semilinear

$$\max(x_1, x_2) = \begin{cases} x_1 : x_1 \geq x_2 \\ x_2 : x_1 < x_2 \end{cases}$$

Piecewise Affine

semilinear domains

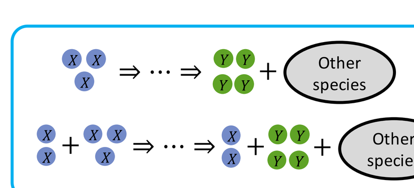


Which **semilinear** functions are also obliviously-computable?

Observation: f is obliviously-computable ⇒ f is **nondecreasing**

Proof idea: If f is decreasing, then any CRN computing f must be able to consume the output, because some sequence of reactions can overproduce output. Thus f is not obliviously-computable.

For example, if $f(3) = 4 > f(5) = 2$:



To stably compute $f(3) = 4$ there is a sequence of reactions from the $\{3X\}$ initial configuration producing $4Y$

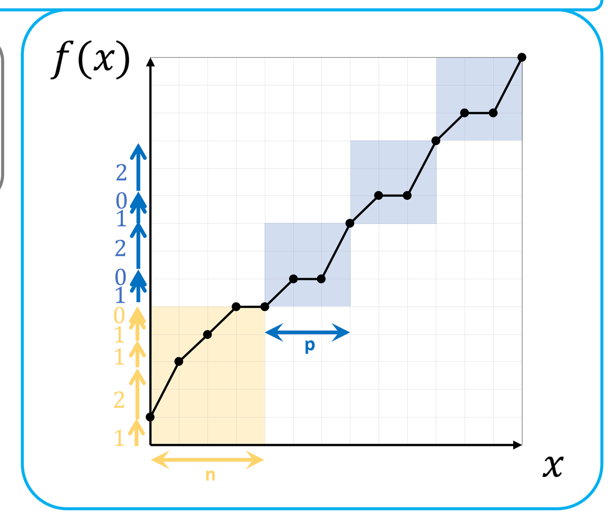
Following the same sequence of reactions from the $\{5X\}$ initial configuration will overproduce Y when computing $f(5) = 2$

Warmup: 1D Classification

Representative obliviously-computable $f: \mathbb{N} \rightarrow \mathbb{N}$

Theorem: $f: \mathbb{N} \rightarrow \mathbb{N}$ is obliviously-computable ⇔ f is semilinear and nondecreasing

Proof idea: Semilinear, nondecreasing $f: \mathbb{N} \rightarrow \mathbb{N}$ will have the structure pictured here, with a "periodic staircase" for sufficiently large inputs, because all affine partial functions on infinite domains must have the same slope for f to be nondecreasing. CRN construction: a leader and multiple "auxiliary leader species". Intuitively, the leader interacts with every input, produces the correct finite output, and "changes states" to track the number of inputs that have been processed (eventually wrapping around mod p).



CRN Construction

Input: X Output: Y
Leader: L (1L in initial config.)

$$\begin{aligned} L &\rightarrow 1Y + L_0 \\ L_0 + X &\rightarrow 2Y + L_1 \\ L_1 + X &\rightarrow 1Y + L_2 \\ L_2 + X &\rightarrow 1Y + L_3 \\ L_3 + X &\rightarrow 0Y + P_{\bar{1}} \end{aligned}$$

$$\begin{aligned} P_{\bar{1}} + X &\rightarrow 1Y + P_{\bar{2}} \\ P_{\bar{2}} + X &\rightarrow 0Y + P_{\bar{0}} \\ P_{\bar{0}} + X &\rightarrow 2Y + P_{\bar{1}} \end{aligned}$$

Impossibility Result

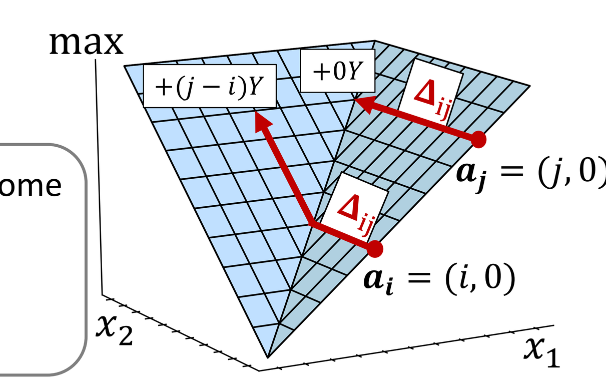
max: $\mathbb{N}^2 \rightarrow \mathbb{N}$ is NOT obliviously-computable

No output-oblivious CRN (even with a leader) can stably compute max. Since max is semilinear and nondecreasing, the 1D classification does not hold in higher dimensions. The following Lemma gives a generalized proof:

Lemma: Let $f: \mathbb{N}^d \rightarrow \mathbb{N}$, with sequence $(\boldsymbol{a}_1, \boldsymbol{a}_2, \dots) \in \mathbb{N}^d$ and some $\boldsymbol{\Delta}_{ij} \in \mathbb{N}^d$ for each $i < j$ such that

$$f(\boldsymbol{a}_i + \boldsymbol{\Delta}_{ij}) - f(\boldsymbol{a}_i) > f(\boldsymbol{a}_j + \boldsymbol{\Delta}_{ij}) - f(\boldsymbol{a}_j).$$

Then f is not obliviously-computable.



For $\max(x_1, x_2)$, take $\boldsymbol{a}_i = (i, 0)$ and $\boldsymbol{\Delta}_{ij} = (0, j)$, then $\max(i,j) - \max(i,0) = j - i > \max(j,j) - \max(j,0) = 0$

Proof idea: Consider a sequence $\boldsymbol{O}_i$ of correct output configurations to compute each $f(\boldsymbol{a}_i)$. Then $\boldsymbol{O}_i \leq \boldsymbol{O}_j$ for some $i < j$ (by Dickson's Lemma). Then adding $\boldsymbol{\Delta}_{ij}$ more input to $\boldsymbol{O}_i$ can produce $f(\boldsymbol{a}_i + \boldsymbol{\Delta}_{ij}) - f(\boldsymbol{a}_i)$ more output (to stably compute $f(\boldsymbol{a}_i + \boldsymbol{\Delta}_{ij})$). We then show a sequence of reactions can overproduce the output when computing $f(\boldsymbol{a}_j + \boldsymbol{\Delta}_{ij})$. Thus any CRN stably computing max must consume its output, so max is not obliviously-computable.

Key Proof Techniques

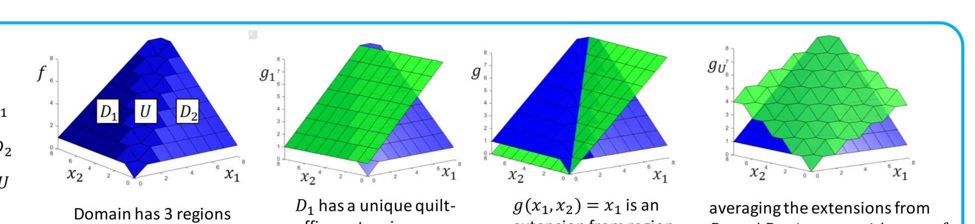
Key technical result
(showing the [eventually-min] condition ii. of the main theorem is necessary)

Lemma: Obliviously-computable f must be a min of quilt-affine functions (for sufficiently large input).

Motivating Example:

$$f(x_1, x_2) = \begin{cases} x_1 + 1 \text{ if } x_1 < x_2 \leftarrow \text{region } D_1 \\ x_2 + 1 \text{ if } x_1 > x_2 \leftarrow \text{region } D_2 \\ x_1 \text{ if } x_1 = x_2 \leftarrow \text{region } U \end{cases}$$

$f = \min(g_1, g_2, g_U)$ for quilt-affine functions $g_1 = x_1, g_2 = x_2, g_U = \left\lceil \frac{x_1 + x_2}{2} \right\rceil$



Domain has 3 regions

D_1 has a unique quilt-affine extension $g_1(x_1, x_2) = x_1 + 1$

$g(x_1, x_2) = x_1$ is an extension from region U, but $g < f$ on D_1

averaging the extensions from D_1 and D_2 gives g_U, with $g_U = f$ on U and $g_U \geq f$ everywhere

Proof ideas:

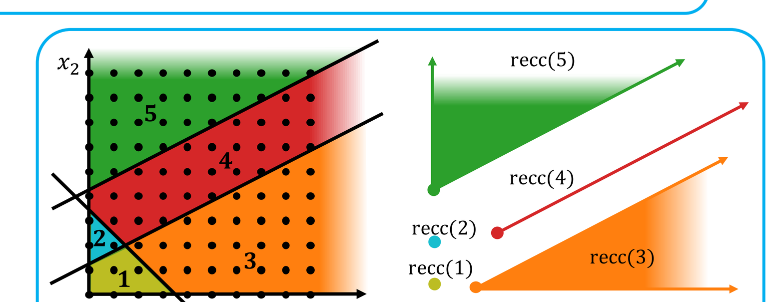
f is semilinear (piecewise affine on semilinear domains).

all hyperplanes given by threshold sets, partition domain into **regions** (convex polyhedra).

Goal: For each region R find a quilt-affine function g_R (the **extension**) such that $g_R = f$ on R and $g_R \geq f$ for large input

This is easier for "infinitely wide" **determined regions**:

Lemma: A determined region D has a unique quilt-affine extension g_D, and $g_D \geq f$ for large inputs.



Domain $\mathbb{N}^2$: three threshold hyperplanes creating five regions (convex polyhedra in $\mathbb{R}^2$). Regions 3,5 are determined with 2D recession cones. Region 4 is under-determined with 1D recession cone. Regions 1,2 are finite with trivial recession cone $\{\boldsymbol{0}\}$.

Key Definitions

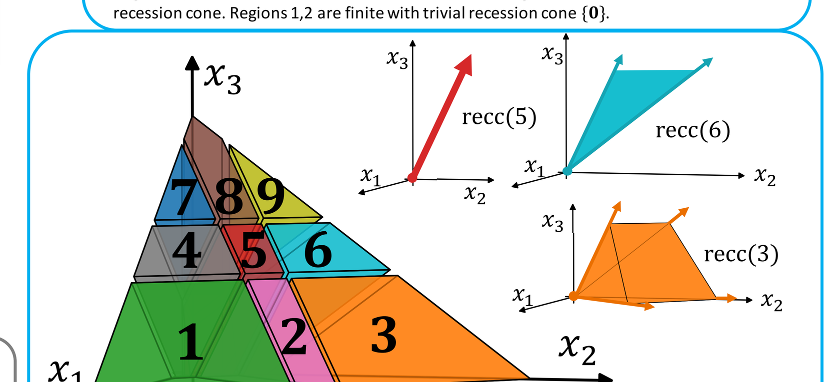
Recession cone: for a region R, the recession cone $\mathrm{recc}(R)$ is the set of all vectors along infinite direction in R (this gives a convex cone)

Determined region: a region D such that $\dim(\mathrm{recc}(D)) = d$

Under-determined region: a region U such that $\dim(\mathrm{recc}(U)) < d$

Neighbor: Region D is a **neighbor** of region U if $\mathrm{recc}(U) \subseteq \mathrm{recc}(D)$.

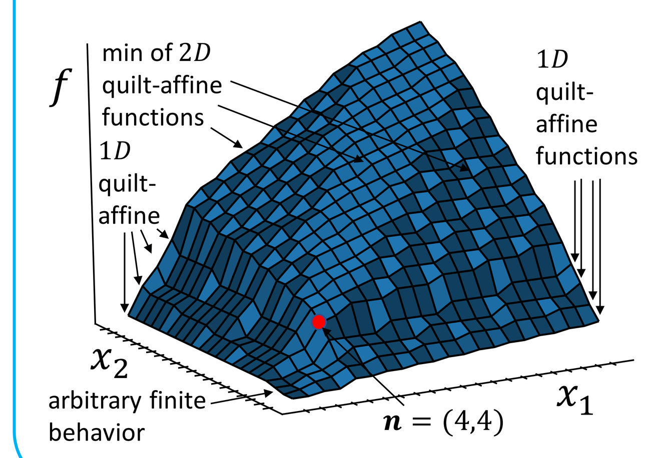
Lemma: For an under-determined region U, consider the quilt-affine extensions of all determined neighbors of U. Then by an averaging process we can construct an extension g_U such that $g_U = f$ on U and $g_U \geq f$ for large inputs.



Domain $\mathbb{N}^3$: four threshold hyperplanes (two parallel pairs) creating nine regions. Regions 1,3,7,9 are determined. Regions 2,4,6,8 are under-determined (2D recession cone) and region 5 is under-determined (1D recession cone). $\mathrm{recc}(5) \subseteq \mathrm{recc}(6) \subseteq \mathrm{recc}(3)$ so region 3 is a determined neighbor of region 5 and region 6.

Main Result: Full Classification

Representative example
(obliviously-computable $f: \mathbb{N}^2 \rightarrow \mathbb{N}$)

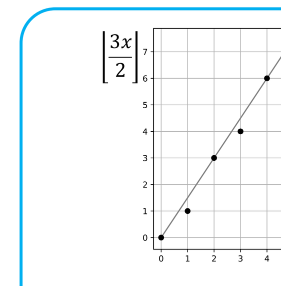


Quilt-affine functions

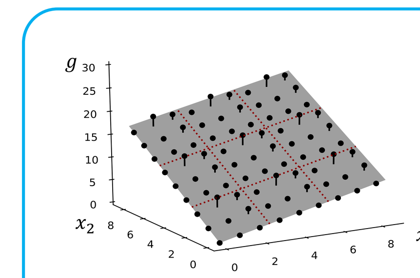
Affine function: linear with constant offset

Quilt-affine function: linear with periodic offset

Generalizes "periodic staircase" behavior from 1D to higher dimensions



$\left\lfloor \frac{3x}{2} \right\rfloor = \frac{3}{2} \cdot x + B(\bar{x} \bmod 2)$ is quilt-affine, where $B(\bar{0}) = 0$ and $B(\bar{1}) = -1/2$



$g(\boldsymbol{x}) = (1,2) \cdot \boldsymbol{x} + B(\bar{\boldsymbol{x}} \bmod 3)$ is quilt-affine, where $B = 0$ except $B(\overline{(1,2)}) = B(\overline{(2,2)}) = 1$ and $B(\overline{(2,1)}) = 2$

Theorem: $f: \mathbb{N}^d \rightarrow \mathbb{N}$ is obliviously-computable ⇔

i. [nondecreasing] f is nondecreasing.

ii. [eventually-min] there exist quilt-affine $g_1, \dots, g_m$ and $\boldsymbol{n} \in \mathbb{N}^d$ such that $f(\boldsymbol{x}) = \min_k(g_k(\boldsymbol{x}))$ for all $\boldsymbol{x} \geq \boldsymbol{n}$.

iii. [recursive] every *fixed-input restriction* $f_{[x_i \to j]}$ fixing some input to a constant value is obliviously-computable (so is also eventually-min of quilt affine functions).

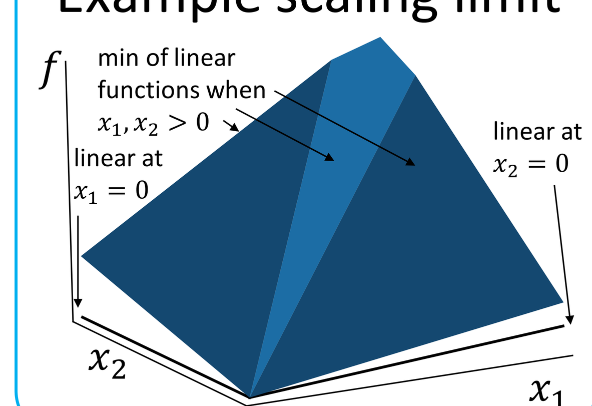
Proof outline:

⇒: If f is obliviously computable, then
Nondecreasing condition (i) is necessary by stated Observation
Eventually-min condition (ii) is necessary by stated Lemma
Recursive condition (iii) is necessary since fixed input restriction $f_{[x_i \to j]}$ is obliviously-computable by modifying the CRN computing f to "hardcode" input $x_i = j$.

⇐: Quilt-affine functions are obliviously-computable (via general CRN construction: auxiliary leader species track period, output correct finite differences).
If f satisfies (i), (ii), (iii), then a general CRN construction shows f is obliviously-computable.
Idea: compute "eventual region" as min of quilt-affine functions, compute all smaller values as fixed-input restrictions by recursive condition (iii). Combine the computations using a minimum and indicators.

Continuous Limit

Example scaling limit



Matches obliviously-computable classification from continuous model

[Cameron Chalk, Niels Kornerup, Wyatt Reeves, and David Soloveichik. Composable rate-independent computation in continuous chemical reaction networks. In Computational Methods in Systems Biology, 2018.]

gave a classification of output-oblivious *real-valued* functions stably computed by *continuous* CRNs (using concentration of species).

Our function class, in a scaling limit, corresponds to precisely their function class (superadditive, positive-continuous, piecewise rational linear).

Open Questions

Leaderless case

General constructions relied on unique leader species.

New necessary condition without a leader:

superadditivity ($f(\boldsymbol{x}) + f(\boldsymbol{y}) \leq f(\boldsymbol{x} + \boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y}$)

Conjecture: $f: \mathbb{N}^d \rightarrow \mathbb{N}$ is leaderlessly-obliviously-computable ⇔ f is obliviously-computable and also superadditive.

This conjecture holds in 1D. If f is also superadditive, we can modify our 1D CRN construction to remove the leader.

Acknowledgements

We thank Anne Condon, Cameron Chalk, Niels Kornerup, Wyatt Reeves, and David Soloveichik for discussing their related work with us and contributing early ideas.

Authors supported by NSF grants 1619343 and 1844976.