Output-Oblivious Chemical Reaction Networks

Ben Chugg, Anne Condon, and Hooman Hashemi, The University of British Columbia

Introduction

We are interested in composability of Chemical reaction networks (CRNs), a stochastic, distributed computing model [1, 2].

Example 1: A CRN to compute $f(n) = 2n - 1$

Input: n copies of X
one copy of a "leader" species L

Reactions

$$X \to 2Y$$
$$L + Y \to \emptyset$$

Output: $2n - 1$ copies of Y

It can be desirable for a CRN to be:

- *Stable*: Always correct.
- *Output-oblivious*: Outputs are never reactants, and so the CRN is easy to compose:

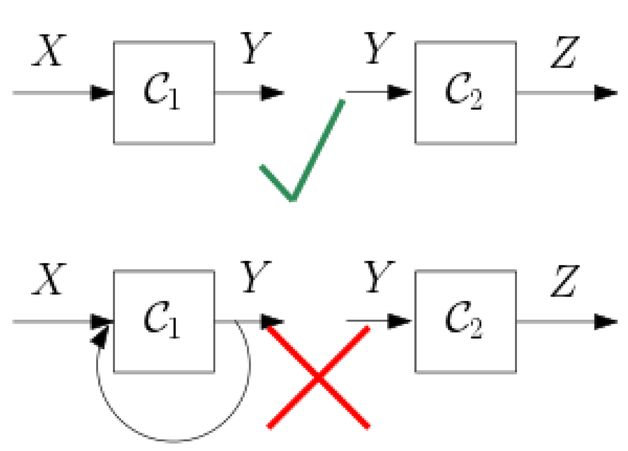


Figure: Top: $\mathcal{C}_1$ does not consume its output, so $\mathcal{C}_2$ can consume Y's without affecting $\mathcal{C}_1$. Bottom: $\mathcal{C}_1$ consumes some Y's, so $\mathcal{C}_2$ might affect the correctness of $\mathcal{C}_1$ by prematurely consuming Y's.

However, Example 1 is not output-oblivious :(.

All semilinear functions (essentially piecewise affine functions) have stable CRNs [2, 3].

Research Question

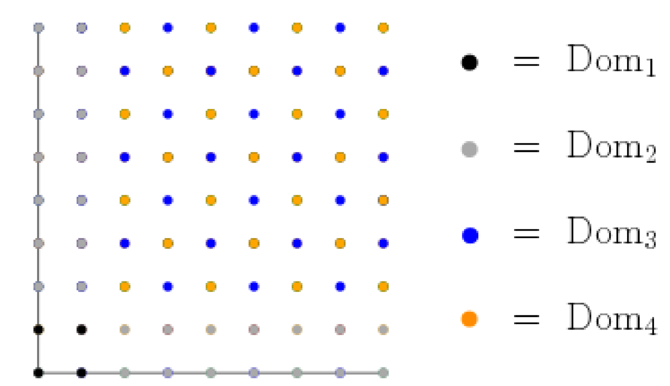
Are all semilinear functions stably computable by output-oblivious CRNs? If not, what subclass of functions are?

Grid-affine functions

These are piecewise affine functions defined on different "grids" (which can be zero-, one-, or two-dimensional).

Example 2:

$$f(\mathbf{n}) = \begin{cases} 2, & \mathbf{n} \in \mathrm{Dom}_1 \cup \mathrm{Dom}_2 \\ 2n_1 + 3n_2 + 1, & \mathbf{n} \in \mathrm{Dom}_3 \\ 2n_1 + 3n_2, & \mathbf{n} \in \mathrm{Dom}_4 \,. \end{cases}$$

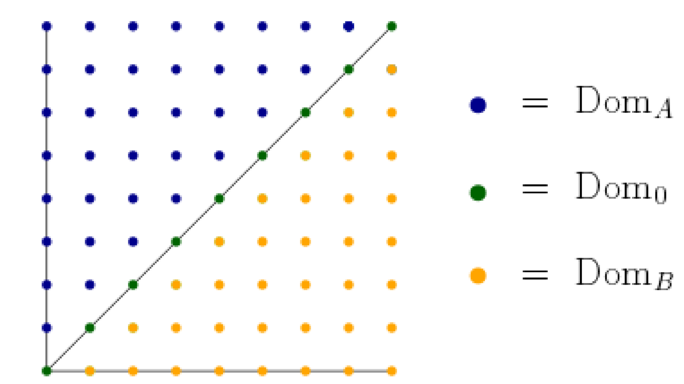


Fissure functions

These are "min-like". The fissure function of Example 3 agrees with $\min\{2n_1+3n_2+2, 3n_1+2n_2+1\}$, except on the "fissure line" Dom_0.

Example 3:

$$g(\mathbf{n}) = \begin{cases} 3n_1 + 2n_2 + 2, & \mathbf{n} \in \mathrm{Dom}_A, \\ 5n_1, & \mathbf{n} \in \mathrm{Dom}_0, \\ 2n_1 + 3n_2 + 2, & \mathbf{n} \in \mathrm{Dom}_B \,. \end{cases}$$



Theorem

An increasing, semilinear function $f : \mathbb{N}^2 \to \mathbb{N}$ is stably computable by an output-oblivious CRN iff it is grid-affine or the minimum of finitely many fissure functions.

Proof of sufficiency

- If f is grid-affine, we compute f separately on each grid, and "stitch" the results together.
- A "line tracking" CRN can stably compute simple fissure functions, such as the function $f(\mathbf{n}) = n_2 + 1$ if $n_1 > n_2$, $f(\mathbf{n}) = n_1 + 1$ if $n_2 > n_1$, and $f(\mathbf{n}) = n_1$ if $n_1 = n_2$:

Input: n_i copies of X_i, one copy of L_0
Reactions:

$$L_0 + X_1 \to L_A + Y$$
$$L_0 + X_2 \to L_B + Y$$
$$L_A + X_2 \to L_0$$
$$L_B + X_1 \to L_0$$

Output: $g(n_1, n_2)$ copies of Y

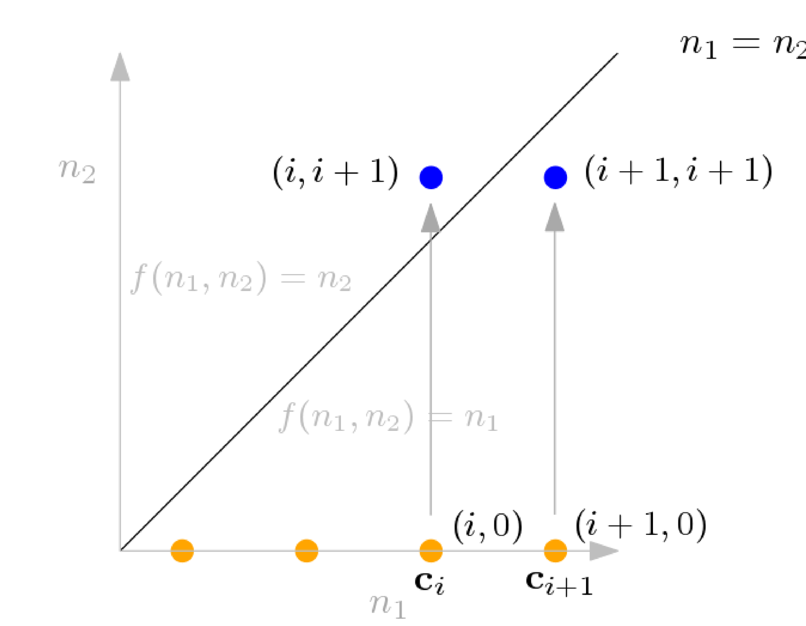
The three possible states L_0, L_A and L_B of the leader track whether the input lies on, above or below the fissure line Dom_0.

- A more sophisticated construction is necessary for more complex fissure functions such as the function g in Example 3 above.

Proof of necessity

As an illustration, we show that if max is stably computed by CRN C, then C cannot be output-oblivious.

1. Take sequence $\{(i,0)\}_{i\in\mathbb{N}}$. On input $(i,0)$, C produces i copies of Y and eventually reaches a stable configuration described by vector $\mathbf{c}_i$, recording counts of species.
2. Apply Dickson's Lemma and relabel to obtain infinite subsequence $\mathbf{c}_1 \leq \mathbf{c}_2 \leq \mathbf{c}_3 \leq \ldots$.
3. On input $(i, i+1)$ C can first produce i Y's (following the execution sequence taken to reach $\mathbf{c}_i$) and then produce one additional Y.



4. On input $(i+1, i+1)$, C can first produce $i+1$ Y's (following the execution sequence taken to reach $\mathbf{c}_{i+1}$) Since $\mathbf{c}_{i+1} \geq \mathbf{c}_i$, C can produce an extra copy of Y. There are now $i+2$ copies of Y present, so C must re-consume output.

References

[1] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7, 2008.

[2] D. Angluin, J. Aspnes, and D. Eisentat. Stably computable predicates are semi-linear. *PODC*, pages 292–299, 2006.

[3] H. Chen, D. Doty, and D. Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, Dec 2014.