Homework 2 – ECS 289, Winter 2018

Required problems

1. **Composition.** Recall that the following CRC with input species X_1, X_2 and output species Y stably computes $y = \max(x_1, x_2)$:

$$\begin{aligned} X_1 &\to Y + W_1 \\ X_2 &\to Y + W_2 \\ W_1 + W_2 &\to K \\ K + Y &\to \varnothing \end{aligned}$$

   And the following CRC with input species Y, X_3 and output species Z stably computes $z = \min(y, x_3)$

$$Y + X_3 \to Z$$

   (a) Show that the CRC obtained by simply combining the above 5 reactions, with input species X_1, X_2, X_3 and output species Z, does *not* stably compute $z = \min(\max(x_1, x_2), x_3)$.

   (b) Design a CRC that stably computes $z = \min(\max(x_1, x_2), x_3)$.

2. **Leader election.** Consider the following CRC, which stably computes $f(x) = 1$

$$\begin{aligned} X &\to Y \\ 2Y &\to Y \end{aligned}$$

   (a) Design a leaderless CRC that stably computes $f(x) = 2$, in which every reaction has at most two reactants and two products.

   (b) Describe how to generalize the previous CRC to produce, for each $k \in \mathbb{N}$, a leaderless CRC that stably computes the function $f(x) = k$, in which every reaction has at most two reactants and two products.

3. **Fast stable computation.** Design a CRC that stably computes $f(n_1, n_2) = n_2$ if $n_1 > 0$ and $f(n_1, n_2) = 0$ otherwise. It should reach a stable configuration in expected time $O(\log n)$, where $n = n_1 + n_2$, and the volume is n.

4. **Chemical caucusing.** Imagine we have two candidates, B and H, being voted on by $n \in \mathbb{Z}^+$ caucus-goers. If B and H supporters encounter each other, one becomes undecided.

Undecided voters are swayed by decided voters:

$$\begin{aligned} B + H &\to B + U \\ B + H &\to H + U \\ B + U &\to 2B \\ H + U &\to 2H \end{aligned}$$

I lived in Iowa for 12 years, and this is a mostly accurate summary of how the caucus works.

(a) Show that for any initial configuration $\mathbf{i}$ with $\|\mathbf{i}\| = n$ and $\mathbf{i}(B) + \mathbf{i}(H) > 0$, for all $\mathbf{c}$ such that $\mathbf{i} \Longrightarrow \mathbf{c}$, there is a terminal configuration $\mathbf{o}$ such that $\mathbf{c} \Longrightarrow \mathbf{o}$ and either $\mathbf{o}(B) = n$ or $\mathbf{o}(H) = n$. That is, the CRN is guaranteed eventually to reach a stable consensus.

(b) Deriving how the expected time scales as a function of n is remarkably difficult. (Or, there is a short, elegant proof that has not yet been discovered.)

Run simulations for different initial configurations and compute the time to stabilize to a consensus.[1] How does the time scale with different initial population sizes, and how is it affected by different initial distributions of B, H, and U?

Optional problems

You don't have to do these, but I think they are interesting to think about.

1. **Combining function and predicate computation.** Design a CRC that stably computes the function

$$f(x_1, x_2) = \begin{cases} x_1, & \text{if } x_1 \geq x_2; \\ 0, & \text{otherwise.} \end{cases}$$

2. **Impossibility of stable multiplication.** I claimed in class that no non-semilinear set or function can be stably decided/computed by a CRN. In this problem you will use one of these facts to prove the other.

   You may assume that no CRD stably decides the set $\left\{ (n_1, n_2) \in \mathbb{N}^2 \;\middle|\; n_1 = (n_2)^2 \right\}$. Use this fact to show that no CRC stably computes the function $f(n_1, n_2) = n_1 \cdot n_2$.

3. **Reachability versus fair executions.** We observed that the definition of stable computation did not say that a CRC *will* stabilize to the correct answer, merely that it always *could* (using the reachability relation $\Longrightarrow$ to define what could happen). We then proved that a CRC stably computing a function under the reachability definition, when simulated under the kinetic model, actually *will* stabilize to the correct output with probability 1. Now we discuss an alternate nonprobabilistic way to formalize that the CRC *will* stabilize to the correct output. It defines "fair" executions and requires that every fair execution do the correct thing.

[1] Since the CRN has nothing but 2-reactant/2-product/unit-rate-constant reactions, you don't have to worry about simulating the full chemical kinetic model with all its exponential random variables. Just pick a pair of molecules at random to "interact". The interaction counts whether the molecules react or not, e.g., if you pick two copies of B, this counts as an interaction even though there is no reaction $B + B \to \ldots$. Count how many interactions are needed to stabilize, divide this by n, and call this the "time" (since we expect about n interactions per unit time in a solution with volume n and n molecules).

A configuration $\mathbf{c}$ is *terminal* if no reaction is applicable to it. A finite execution $\mathcal{E} = (\mathbf{c}_0, \ldots, \mathbf{c}_k)$ is *fair* if $\mathbf{c}_k$ is terminal, and an infinite execution $\mathcal{E} = (\mathbf{c}_0, \mathbf{c}_1, \ldots)$ is *fair* if, for all $\mathbf{c}$, if there are infinitely many $i \in \mathbb{N}$ such that $\mathbf{c}_i \Longrightarrow \mathbf{c}$, then there are infinitely many $j \in \mathbb{N}$ such that $\mathbf{c}_j = \mathbf{c}$. In other words, every configuration that is infinitely often reach*able* in $\mathcal{E}$ is infinitely often reach*ed*.[2]

Show that a CRC $\mathcal{C} = (\Lambda, R, \Sigma, Y, \mathbf{s})$ stably computes a function f under the definition given in lecture if and only if, for every valid initial configuration $\mathbf{i} \in \mathbb{N}^\Lambda$, every fair execution $\mathcal{E} = (\mathbf{i}, \ldots)$ contains an output stable configuration $\mathbf{o} \in \mathbb{N}^\Lambda$ such that $\mathbf{o}(Y) = f(\mathbf{i} \upharpoonright \Sigma)$.

4. **Majority wins.** In problem 4, show that if the initial configuration is $\mathbf{i} = \{0.51n\ B, 0.49n\ H\}$, then for sufficiently large n, with probability at least 99%, the CRN reaches the configuration $\{nB\}$. That is, the initial majority probably wins.

5. **Consuming output.** Some CRCs, such as $X \to 2Y$ and $2X \to Y$, only produce the output species. Others, such as $X_1 \to Y$, $X_2 + Y \to \varnothing$ to compute subtraction, do consume the output species.

   (a) Show that if $f : \mathbb{N}^k \to \mathbb{N}$ is not a monotone function, then any CRC stably computing f must consume the output species Y, i.e., must have a reaction where Y's stoichiometric coefficient as a reactant exceeds its stoichiometric coefficient as a product (e.g., $X+Y \to A$ or $3Y \to 2Y$).

   (b) Show that any leaderless CRC stably computing $f(x_1, x_2) = \max(x_1, x_2)$ must consume the output species Y.

[2]This concept is borrowed from distributed computing. A distributed algorithm runs over a network in which the order in which various nodes will receive messages from each other is unknown. It would be too restrictive to imagine that the algorithm should work under *any* schedule of message delivery. For example, there there are three nodes a, b, c, if the scheduler always delivers messages to a and b before it delivers them to c, and if every message received by a triggers an immediate response message to b and vice versa, then c will be "starved" under this unfair schedule.

There are many definitions of fair schedules in CRNs. This is the one most commonly used. Others look different but are equivalent; for instance, requiring that if there are infinitely many $i \in \mathbb{N}$ such that $\mathbf{c}_i \Longrightarrow^1 \mathbf{c}$ (i.e., $\mathbf{c}_i$ can reach to $\mathbf{c}$ by a single reaction), then there are infinitely many $j \in \mathbb{N}$ such that $\mathbf{c}_j = \mathbf{c}$. Others that look reasonable actually define a weaker notion; for instance, requiring that every reaction that is infinitely often applicable is infinitely often applied. By "weaker", we mean that more executions are fair under the latter definition, and some of the new executions fail to get the answer correct.