



Rate-Independent Continuous Inhibitory Chemical Reaction Networks Are Turing-Universal

Kim Calabrese^(✉) and David Doty^(✉)

University of California, Davis, USA
{ebcalabrese,doty}@ucdavis.edu

Abstract. We study the model of continuous chemical reaction networks (CRNs), consisting of reactions such as $A + B \rightarrow C + D$ that can transform some continuous, nonnegative real-valued quantity (called a *concentration*) of chemical species A and B into equal concentrations of C and D . Such a reaction can occur from any state in which both reactants A and B are present, i.e., have positive concentration. We modify the model to allow *inhibitors*, for instance, reaction $A + B \xrightarrow{I} C + D$ can occur only if the reactants A and B are present and the inhibitor I is absent.

The computational power of non-inhibitory CRNs has been studied. For instance, the reaction $X_1 + X_2 \rightarrow Y$ can be thought to compute the function $f(x_1, x_2) = \min(x_1, x_2)$. Under an “adversarial” model in which reaction rates can vary arbitrarily over time, it was found that exactly the continuous, piecewise linear functions can be computed, ruling out even simple functions such as $f(x) = x^2$. In contrast, in this paper we show that inhibitory CRNs can compute any computable function $f : \mathbb{N} \rightarrow \mathbb{N}$.

Keywords: Chemical Reaction Networks · Mass-Action · Analog Computation · Turing Universal

1 Introduction

The model of continuous chemical reaction networks (CRNs) consists of reactions such as $A + B \rightarrow C + D$ that can transform some continuous, nonnegative real-valued quantity (called a *concentration*) of chemical species A and B (the *reactants*) into equal concentrations of C and D (the *products*). This model has long held an important role in modeling naturally occurring chemical systems and predicting their evolution over time. Recently, the model has been investigated, not as a modeling language, but as a *programming* language for describing desired behavior of engineered chemicals. For example, the reaction $X_1 + X_2 \rightarrow Y$ can be thought to compute the function $f(x_1, x_2) = \min(x_1, x_2)$, in the sense that if we start in configuration $\{x_1 X_1, x_2 X_2\}$, i.e., concentration x_1 of species X_1 and concentration x_2 of species X_2 , as long as the reaction keeps happening, it will eventually produce concentration $\min(x_1, x_2)$ of species Y .

The computational power depends greatly on how reaction rates are defined. The most common rate model is *mass-action*, which says that the rate of a reaction like $A + B \xrightarrow{k} C + D$, with positive *rate constant* $k > 0$, proceeds at rate $k \cdot [A] \cdot [B]$, where $[S]$ represents the concentration of species S . The rates of all reactions affecting a species S determines its derivative $\frac{d[S]}{dt}$ (adding rates of reactions where S is a product, and subtracting rates where it is a reactant), so the concentrations evolve according to a system of polynomial ODEs. It was recently shown that mass-action CRNs are capable of Turing universal computation [5], a very complex construction resulting from a long and deep line of research that culminated in showing the surprising computational power of polynomial ODEs [2].

What if reaction rates are not so predictable over time? One could imagine a solution does not remain well-mixed, so that some reactions go faster in a certain part of the volume where some species are more concentrated. It is also the case that it is difficult experimentally to engineer precise rate constants [8]. To address these issues, Chen, Doty, Reeves, and Soloveichik [3] defined a model of *adversarial* reaction rates and asked what functions can be computed when the rates can vary arbitrarily over time. They found that this model, called *stable computation*, is much more computationally limited than with mass-action rates: exactly the continuous, piecewise linear functions $f : \mathbb{R}^d \rightarrow \mathbb{R}$ can be stably computed.¹ An open question from [3] concerns a natural modification of the CRN model, inspired by similar models of gene regulatory networks, in which the presence of a species can *inhibit* a reaction from occurring. For example, the reaction $A + B \xrightarrow{I} C + D$ can occur only if its reactants are present ($[A], [B] > 0$) and its inhibitor is absent ($[I] = 0$). We call such a network an *inhibitory chemical reaction network* (iCRN).²

The negative results of [3], showing computation is limited to continuous piecewise linear functions, heavily use the fact that the reachability relation \rightsquigarrow (defined in Sect. 2) on CRN configurations is *additive*: if $\mathbf{x} \rightsquigarrow \mathbf{y}$ for configurations \mathbf{x}, \mathbf{y} (nonnegative vectors representing concentrations of each species), then for all nonnegative \mathbf{c} , we have $\mathbf{x} + \mathbf{c} \rightsquigarrow \mathbf{y} + \mathbf{c}$; in other words the presence of extra molecules (represented by \mathbf{c}) cannot *prevent* reactions from occurring. However,

¹ Technically this is using the so-called *dual-rail* encoding, which represents a single real value x as the difference of *two* species concentrations $[X^+] - [X^-]$. If one encodes inputs and output directly as nonnegative concentrations, then some discontinuities can occur, but only when some input x_i goes from 0 to positive.

² Note that our notation $A + B \xrightarrow{I} C + D$ puts inhibitors above the reaction arrow where a rate constant would normally be written, but since we consider rate-independent computation, we will have no rate constants. We also note that in gene regulatory networks, typically a species (called *transcription factor* in that literature) inhibits another *species*, which is assumed to be produced at some otherwise constant rate by a single reaction, whereas our model is more general in allowing inhibitors of arbitrary reactions (so I could inhibit production of C via one reaction $A \xrightarrow{I} C$ but not via another reaction $B \rightarrow C$).

with inhibitors reachability is no longer additive (if \mathbf{c} contains inhibitors that are absent in \mathbf{x}), so it is natural to wonder if inhibitors increase the computational power of the model.

It is well-known “folklore” that in the *discrete* model of iCRNs, where the amount of a species is modeled as a nonnegative integer *count*, in which reactions discretely increment or decrement species counts, then inhibitors give the model Turing-universal power. It is worth seeing why this is true, to understand the novel contribution of this paper (and why it is not trivially solved in the continuous model by the discrete iCRN we describe next). It is well-known that register machines—finite-state machines equipped with a fixed number of nonnegative integer *registers*, each of which can be incremented, decremented, or tested for 0—are Turing universal [6]. An example register machine is:

```

1  dec r1,5
2  inc r2
3  inc r2
4  goto 1
5  halt

```

Line (a.k.a., *state*) 1 has the interpretation: decrement register r_1 and then go to line 2, unless r_1 is 0, in which case go to line 5. Increment instructions always increment the specified register and go to the next line. The `goto 1` statement on line 4 is syntactic sugar for `dec r3,1` for some register r_3 that is always 0. The above register machine, interpreted as taking an input x in register r_1 and halting with an output value in register r_2 , computes the function $f(x) = 2x$.

For a register machine consisting of such increment and decrement instructions, the following is a straightforward transformation of the instruction for line/state i to iCRN reactions:

<code>inc r-j</code>	$L_i \rightarrow L_{i+1} + R_j$
<code>dec r-j,k</code>	$L_i + R_j \rightarrow L_{i+1}$
	$L_i \xrightarrow[R_j]{\perp} L_k$

It is clear that at any time exactly one reaction is applicable, and it simulates the next instruction of the register machine. In particular, when on a decrement instruction, the power of inhibition is used to ensure that if R_j has positive count, then only the first of the two decrement reactions is applicable (and as in the non-inhibitory CRN model, when R_j is absent, only the second decrement reaction is applicable). Note that a `halt` instruction on line i is not explicitly implemented as any reaction; the simple lack of any reaction with L_i as a reactant means that the CRN will terminate when the register machine does.

Our main construction in Sect. 3 follows this basic strategy of simulating register machines, using inhibition to detect when a register is 0. However, our main novel contribution is a way to “discretize” the behavior of the continuous CRN, so that the discrete steps of the register machine can be simulated faithfully. This is primarily done by introducing a *stable oscillator*, shown in Sect. 3.1.

2 Preliminaries

These definitions largely follow those of [3], the only exception being the definition of *applicable reaction*, which is modified to account for inhibitors.

For any set A , let $\mathcal{P}(A)$ denote the power set of A (set of all subsets of A). Let \mathbb{N} denote the nonnegative integers and \mathbb{R} denote the real numbers. Given a finite set F and a set S , let S^F denote the set of functions $\mathbf{c} : F \rightarrow S$. In the case of $S = \mathbb{R}$ (resp. \mathbb{N}), we view \mathbf{c} equivalently as a real-valued (resp. integer-valued) vector indexed by elements of F . Given $x \in F$, we write $\mathbf{c}(x)$, to denote the real number indexed by x . The notation $\mathbb{R}_{\geq 0}^F$ is defined similarly for nonnegative real vectors. Throughout this paper, let Λ be a finite set of chemical *species*. Given $S \in \Lambda$ and $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda$, we refer to $\mathbf{c}(S)$ as the *concentration of S in \mathbf{c}* . When the configuration \mathbf{c} is understood from context, we write $[S]$ to denote $\mathbf{c}(S)$. For any $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda$, let $[\mathbf{c}] = \{S \in \Lambda \mid \mathbf{c}(S) > 0\}$, the set of species *present* in \mathbf{c} (a.k.a., the *support* of \mathbf{c}). We write $\mathbf{c} \leq \mathbf{c}'$ to denote that $\mathbf{c}(S) \leq \mathbf{c}'(S)$ for all $S \in \Lambda$. Given $\mathbf{c}, \mathbf{c}' \in \mathbb{R}_{\geq 0}^\Lambda$, we define the vector component-wise operations of addition $\mathbf{c} + \mathbf{c}'$, subtraction $\mathbf{c} - \mathbf{c}'$, and scalar multiplication $x\mathbf{c}$ for $x \in \mathbb{R}$.

A *reaction* over Λ is a triple $\alpha = (\mathbf{r}, \Delta, \mathbf{p}) \in \mathbb{N}^\Lambda \times \mathcal{P}(\Lambda) \times \mathbb{N}^\Lambda$, such that $\mathbf{r} \neq \mathbf{p}$, specifying the stoichiometry of the reactants, products, as well as the inhibitors of the reaction respectively.³ We say a reaction α is *inhibited* by species I if $I \in \Delta$. For instance, given $\Lambda = \{A, B, C, I\}$, the reaction $A + 2B \xrightarrow{I} A + 3C$ is the triple $((1, 2, 0), \{I\}, (1, 0, 3))$.

An *inhibitory chemical reaction network (iCRN)* is a pair $\mathcal{C} = (\Lambda, R)$, where Λ is a finite set of chemical *species*, and R is a finite set of reactions over Λ . A *configuration* of a iCRN $\mathcal{C} = (\Lambda, R)$ is a vector $\mathbf{c} \in \mathbb{R}_{\geq 0}^\Lambda$. Given a configuration \mathbf{c} and reaction $\alpha = (\mathbf{r}, \Delta, \mathbf{p})$, we say that α is *applicable* in \mathbf{c} if $[\mathbf{r}] \subseteq [\mathbf{c}]$ (i.e., \mathbf{c} contains positive concentration of all of the reactants) and $[\mathbf{c}] \cap \Delta = \emptyset$ (no inhibitor is present in \mathbf{c}). If no reaction is applicable in configuration \mathbf{c} , we say \mathbf{c} is *static*.

Fix an iCRN $\mathcal{C} = (\Lambda, R)$. We define the $|\Lambda| \times |R|$ *stoichiometry matrix* \mathbf{M} such that, for species $S \in \Lambda$ and reaction $\alpha = (\mathbf{r}, \Delta, \mathbf{p}) \in R$, $\mathbf{M}(S, \alpha) = \mathbf{p}(S) - \mathbf{r}(S)$ is the net amount of S produced by α (negative if S is consumed).⁴ For example,

³ It is customary to define, for each reaction, a *rate constant* $k \in \mathbb{R}_{>0}$ specifying a constant multiplier on the mass-action rate (i.e., the product of the reactant concentrations), but as we are studying CRNs whose output is independent of the reaction rates, we leave the rate constants out of the definition.

⁴ \mathbf{M} does not fully specify \mathcal{C} , since catalysts and inhibitors are not modeled: reactions $A + B \xrightarrow{C} A + D$ and $B \rightarrow D$ both correspond to the column vector $(0, -1, 0, 1)^\top$.

if we have the reactions $X \rightarrow Y$ and $X + A \rightarrow 2X + 3Y$, and if the three rows correspond to A , X , and Y , in that order, then

$$\mathbf{M} = \begin{pmatrix} 0 & -1 \\ -1 & 1 \\ 1 & 3 \end{pmatrix}$$

Definition 2.1. Configuration \mathbf{d} is straight-line reachable (aka 1-segment reachable) from configuration \mathbf{c} , written $\mathbf{c} \rightarrow^1 \mathbf{d}$, if $(\exists \mathbf{u} \in \mathbb{R}_{\geq 0}^R) \mathbf{c} + \mathbf{M}\mathbf{u} = \mathbf{d}$ and $\mathbf{u}(\alpha) > 0$ only if reaction α is applicable at \mathbf{c} . In this case write $\mathbf{c} \rightarrow_{\mathbf{u}}^1 \mathbf{d}$.

Intuitively, by a single segment we mean running the reactions applicable at \mathbf{c} at a constant (possibly 0) rate to get from \mathbf{c} to \mathbf{d} . In the definition, $\mathbf{u}(\alpha)$ represents the flux of reaction $\alpha \in R$.

Definition 2.2. Let $k \in \mathbb{N}$. Configuration \mathbf{d} is k -segment reachable from configuration \mathbf{c} , written $\mathbf{c} \rightsquigarrow^k \mathbf{d}$, if $(\exists \mathbf{b}_0, \dots, \mathbf{b}_k) \mathbf{c} = \mathbf{b}_0 \rightarrow^1 \mathbf{b}_1 \rightarrow^1 \mathbf{b}_2 \rightarrow^1 \dots \rightarrow^1 \mathbf{b}_k$, with $\mathbf{b}_k = \mathbf{d}$.

Definition 2.3. Configuration \mathbf{d} is segment-reachable (or simply reachable) from configuration \mathbf{c} , written $\mathbf{c} \rightsquigarrow \mathbf{d}$, if $(\exists k \in \mathbb{N}) \mathbf{c} \rightsquigarrow^k \mathbf{d}$.

Often Definition 2.3 is used implicitly, when we make statements such as, “Run reaction 1 until X is gone, then run reaction 2 until Y is gone”, which implicitly defines two straight lines in concentration space. Although we make no attempt to ascribe an “execution time” to any path followed by segments in Definition 2.3, it is sometimes useful to refer to such paths over time. In this case we suppose that each segment takes one unit of time, so that if $\mathbf{x} \rightsquigarrow^k \mathbf{y}$, we associate this to a trajectory $\rho : [0, k] \rightarrow \mathbb{R}_{\geq 0}^A$, where $\rho(t)$ represents the concentrations of species after t units of time have elapsed, i.e., following the first $\lfloor t \rfloor$ segments, then a fraction of the t ’th segment if $t \notin \mathbb{N}$ (so that for integer t , $\rho(t)$ is the configuration \mathbf{b}_t in Definition 2.2). In this case we write $\mathbf{x} \rightsquigarrow_{\rho} \mathbf{y}$.

Given configurations $\mathbf{x}, \mathbf{y}, \mathbf{z}$ such that $\mathbf{x} \rightsquigarrow_{\rho_1} \mathbf{y}$ and $\mathbf{y} \rightsquigarrow_{\rho_2} \mathbf{z}$, we denote the concatenation of trajectories ρ_1 and ρ_2 to be the trajectory $\rho_1 : \rho_2$ such that $\mathbf{x} \rightsquigarrow_{\rho_1 : \rho_2} \mathbf{z}$.

We now formalize what it means for an iCRN to “rate-independently” compute a function f . Since our main result is about simulating register machines that process natural numbers, we define stable computation for functions $f : \mathbb{N} \rightarrow \mathbb{N}$.⁵ An *inhibitory chemical reaction computer (iCRC)* is a tuple $\mathcal{C} = (\Lambda, R, \mathbf{s}, X, Y)$, where (Λ, R) is an iCRN, $\mathbf{s} \in \mathbb{N}^{\Lambda}$ is the *initial context* (species other than the input that are initially present with some constant concentration; in our case, $\mathbf{s}(A_1) = 1$ for a single species $A_1 \in \Lambda$ and 0 for all other

⁵ Since iCRNs operate on real-valued concentrations, a very similar definition for functions $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ makes sense (and was formally defined for non-inhibitory CRNs in [3]); Sect. 4 discusses this issue further. We could also extend the definition to take multiple inputs for a function $f : \mathbb{N}^d \rightarrow \mathbb{N}$, but since register machines are Turing universal, we could encode multiple input integers via a pairing function into a single integer, so it is no loss of generality to consider single-input functions.

species), $X \in \mathcal{A}$ is the *input species*, and $Y \in \mathcal{A}$ is the *output species*. We say a configuration $\mathbf{o} \in \mathbb{R}_{\geq 0}^{\mathcal{A}}$ is *stable* if, for all \mathbf{o}' such that $\mathbf{o} \rightsquigarrow \mathbf{o}'$, $\mathbf{o}(Y) = \mathbf{o}'(Y)$, i.e., the concentration of Y cannot change once \mathbf{o} has been reached. Let $f : \mathbb{N} \rightarrow \mathbb{N}$. We say \mathcal{C} *stably computes* f if, for all $n \in \mathbb{N}$, starting from initial configuration $\mathbf{i} = \mathbf{s} + \{nX\}$ (i.e., starting with initial context, plus the desired input amount of X), for all configurations \mathbf{c} such that $\mathbf{i} \rightsquigarrow \mathbf{c}$, there is \mathbf{o} such that $\mathbf{c} \rightsquigarrow \mathbf{o}$, such that \mathbf{o} is stable and $\mathbf{o}(Y) = f(n)$.

3 Main Results

Our goal is to design an iCRN that simulates the behavior of a register machine, similar to simulations by discrete CRNs [1, 7]. The inclusion of inhibitors to our model allows us to enforce deterministic state transitions in chemical reaction networks, but to emulate the sequential power of discrete computation, we need a mechanism to manage control flow. First, we describe a simpler “stably oscillating” iCRN that is, in a sense, the main conceptual contribution of this paper.

3.1 Stable Oscillation

The following definition captures the behavior of a system of chemical reactions that execute sequentially, and eventually repeat their execution. A similar definition for the discrete model of population protocols appears in [4].⁶ In particular, we have species A_1, \dots, A_k that all start at 0. A_1 monotonically goes up to 1, then monotonically down to 0, then A_2 goes up and down similarly, etc. After A_k does this, the whole thing repeats.

Definition 3.1. Let $\mathcal{A} = \{A_1, A_2, \dots, A_k\}$ be a set of species in an iCRN, and let ρ be a trajectory. We say $\rho([t_1, t_2])$ is a wave of A_i if for some $t_1 < t < t_2$

- $\rho(t_1)(A_i) = \rho(t_2)(A_i) = 0$,
- $\rho(t)(A_i) = 1$,
- $\rho([t_1, t])(A_i)$ is nondecreasing, and
- $\rho([t, t_2])(A_i)$ is nonincreasing.

$\rho([T_1, T_2])$ is a period of oscillation of \mathcal{A} if there exists $T_1 = t_1, t_2, \dots, t_k = T_2$ such that for all $0 \leq i < k$,

- $\rho([t_i, t_{i+1}])$ is a wave of A_i , and
- for all $j \neq i$ and all $t_i \leq t \leq t_{i+1}$, $\rho(t)(A_j) = 0$.

Definition 3.2. We say an iCRN \mathcal{C} stably oscillates on \mathcal{A} from configuration \mathbf{i} if for all \mathbf{c} such that $\mathbf{i} \rightsquigarrow_{\rho_1} \mathbf{c}$, we have $\mathbf{c} \rightsquigarrow_{\rho_2} \mathbf{i}$ such that letting $\rho = \rho_1 : \rho_2$, $\rho([0, t])$ is one or more periods of oscillation of \mathcal{A} .

⁶ However, Definition 3.2 is distinct from the that of [4], both by being defined in a continuous-state rather than a discrete-state model, and in that we do not require “self-stabilizing” behavior (which dictates that the behavior should occur from any possible initial state).

The next lemma demonstrates an iCRN that stably oscillates. We note that Lemma 3.3 is not used directly in the rest of the paper. Instead, the proof of Lemma 3.3 is intended to serve as a “warmup” to illustrate some of the key ideas used in the more complex iCRN defined in Sect. 3.2.

Lemma 3.3. *Let $n \geq 3$ and \mathcal{C} be the iCRN with species $\Lambda = \{X_0, X_1, \dots, X_{n-1}\}$ and for each $0 \leq i < n$, reaction $X_i \xrightarrow{X_{i-1}} X_{i+1}$, where $i-1$ and $i+1$ are both taken modulo n . If $\mathbf{i} = \{1X_0\}$ is the starting configuration, then \mathcal{C} stably oscillates on $\mathcal{O} = \{X_i \mid 0 \leq i \leq n, i \text{ is odd}\}$.*

Proof. For each $0 \leq i < n$, let α_i be the reaction $X_i \xrightarrow{X_{i-1}} X_{i+1}$. First, observe that for any configuration \mathbf{c} in which the species X_i and X_{i+1} are present, the only applicable reaction is α_i , since the reaction $X_{i+1} \xrightarrow{X_i} X_{i+2}$ is inhibited by X_i , and all other reactions have a reactant absent. Thus every sufficiently long path from \mathbf{c} just executes α_i until X_i is absent. Once X_i is absent, α_{i+1} becomes applicable. At this point, we have only X_{i+1} present, so by similar reasoning, only α_{i+1} is applicable and every sufficiently long path runs only α_{i+1} until X_{i+1} is absent.

Iterating this reasoning over all i , for each $0 \leq i < n$, let \mathbf{u}_i denote the flux vector with $\mathbf{u}_i(\alpha_i) = 1$ and $\mathbf{u}_i(\alpha_j) = 0$ for $j \neq i$ (i.e., execute only reaction α_i , for one unit of flux). Then starting from initial configuration $\mathbf{i} = \{1X_0\}$, we see that every path starting from \mathbf{i} is of the form

$$\begin{aligned} \{1X_0\} &\xrightarrow{\mathbf{u}_0} \{1X_1\} \xrightarrow{\mathbf{u}_1} \{1X_2\} \xrightarrow{\mathbf{u}_2} \dots \\ \{1X_{n-1}\} &\xrightarrow{\mathbf{u}_{n-1}} \{1X_0\} \xrightarrow{\mathbf{u}_0} \{1X_1\} \xrightarrow{\mathbf{u}_1} \dots \{aX_i, (1-a)X_{i+1}\}, \end{aligned}$$

for some $0 \leq a \leq 1$, or, assuming the path does not get to configuration $\{1X_0\}$ above, $\{1X_0\} \xrightarrow{\mathbf{u}_0} \{1X_1\} \xrightarrow{\mathbf{u}_1} \{1X_2\} \xrightarrow{\mathbf{u}_2} \dots \{aX_i, (1-a)X_{i+1}\}$.

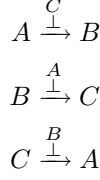
In either case, by continuing to apply α_i with flux a , then unit fluxes of $\alpha_{i+1}, \alpha_{i+2}$, etc. until we reach configuration $\{1X_0\}$, this does some positive integer number of periods of oscillation. Let $\mathbf{i} = \{1X_0\}$, $\mathbf{c} = \{aX_i, (1-a)X_{i+1}\}$, this satisfies the definition of oscillation for the species in \mathcal{O} . \square

3.2 Construction of iCRN Simulating a Register Machine

In this section we describe how to construct an iCRN \mathcal{C} to simulate an arbitrary register machine \mathcal{R} .

Let the set of states (or lines) of \mathcal{R} be $Q = \{1, 2, \dots, m\}$, supposing it starts in state 1 with initial input register value $n \in \mathbb{N}$. Suppose \mathcal{R} 's input register is $\mathbf{r_in}$ and its output register is $\mathbf{r_out}$. To simulate \mathcal{R} , \mathcal{C} has input species R_{in} and output species R_{out} , and starts with configuration $\{1A_1, nR_{\text{in}}\}$ (i.e., with initial context $\mathbf{s} = \{1A_1\}$).

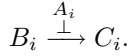
Consider these reactions, which implement the stable oscillator of Lemma 3.3 with 3 species (where $X_0 = A, X_1 = B, X_2 = C$).



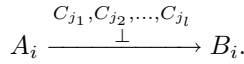
Although we do not use those exact reactions, it is helpful to see that iCRN as an introduction to how we implement the oscillator component of \mathcal{C} . \mathcal{C} has m variants of each of those species $\{A_1, B_1, C_1, \dots, A_m, B_m, C_m\}$, each subscript representing a state of \mathcal{R} . We will additionally have a species R_1, R_2, \dots to represent the various registers of \mathcal{R} as well as designated input and output species $R_{\text{in}}, R_{\text{out}}$. For ease of exposition, we use the convention that \mathcal{R} has exactly one input and output register, but this is easily extendable.

Intuitively, the variants of the last reaction $C \xrightarrow{B} A$ will perform all the logic of the register machine: incrementing, decrementing, and changing states. The other two (variants of) reactions $A \xrightarrow{C} B$ and $B \xrightarrow{A} C$ are simply to make the oscillator work while remembering the current state. However, since the stateful oscillator will change states in the last reaction, and the last reaction's reactant is an inhibitor for the first reaction, we need to be careful in selecting the correct inhibitors for the first reaction to acknowledge the states are different, and that *multiple* stateful variants of C could be inhibitors of a single variant of A .

Formally, for all $1 \leq i \leq m$, \mathcal{C} has the reaction

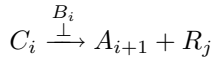


For all $1 \leq i \leq m$, let $\{j_1, j_2, \dots, j_l\}$ be the set of states that are potential predecessors of state i . This includes $j = i - 1$ if $i > 1$ and state j is not a **goto**, as well as all j such that a decrement test for 0 can cause a jump from j to i . For all $1 \leq i \leq m$, \mathcal{C} also has the reaction



Finally, for all $1 \leq i \leq m$, \mathcal{C} has the following reactions to simulate register machine instructions.

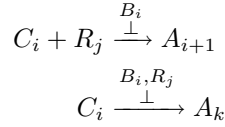
- if state i is **inc r_j** (increment register j and move from state i to $i + 1$):



Note the dual role of C_i : it helps the “clock” to oscillate, but its maximum concentration also defines one “unit” of concentration to help us use real-valued concentrations to represent discrete integer counts in registers of \mathcal{R} . In

other words, the initial amount of A_1 (which sets the maximum concentration achieved by any C_i) is also the amount by which $[R_j]$ increases (and the amount it decreases in a decrement instruction).

- if state i is **dec** r_j, k (decrement register j and move from state i to $i + 1$, unless it is 0, in which case go to state k):



As in the case for the discrete iCRN described in Sect. 1, no reactions are associated to C_i if state i is a **halt** instruction.

3.3 Proof of Correctness

In this section, we prove that the iCRN \mathcal{C} described in Sect. 3.2 correctly simulates the register machine \mathcal{R} .

In the definition of \rightsquigarrow , it is technically allowed for two consecutive segments to “point the same direction”, i.e., $\mathbf{x} \xrightarrow{\mathbf{u}_1} \mathbf{y} \xrightarrow{\mathbf{u}_2} \mathbf{z}$ such that \mathbf{u}_1 and \mathbf{u}_2 are multiples of each other. The next observation says that we can assume without loss of generality this does not happen, since any two such consecutive segments \mathbf{u}_1 and \mathbf{u}_2 can always be concatenated into a single segment $\mathbf{u}_1 + \mathbf{u}_2$.

Observation 3.4. *In any iCRN, if $\mathbf{x} \rightsquigarrow \mathbf{y}$, we may assume without loss of generality that each pair of consecutive segments are not multiples of each other. In particular, if exactly one reaction is applicable at any time, then any two consecutive segments use different reactions.*

We also note that there is a distinction between the function of species that “oscillate” (i.e. species $A_1, B_1, C_1, \dots, A_n, B_n, C_n$) and species that represent the value stored in a register ($R_1, R_2 \dots$). We call the former *oscillator species* and the latter *register species*. Since the control flow of our construction is driven primarily by the so-called oscillator species, it suffices to focus on their behavior when discussing the properties of the iCRN induced by our construction.

We develop machinery to talk about specific configurations of \mathcal{C} that contain oscillator species at concentration 1.

Definition 3.5. *Let $A \in \Lambda$ be an oscillator species. We say configuration $\mathbf{x} \in \mathbb{R}_{\geq 0}^A$ is a transition point of A if $\mathbf{x}(A) = 1$ and $\mathbf{x}(B) = 0$ for all other oscillator species B .*

Intuitively, a transition point marks the peak of a species’ oscillation, representing a configuration where a previously present oscillator species depletes, allowing a new reaction to become applicable. Definition 3.5 implicitly characterizes the configurations in \mathcal{C} : a configuration is either a transition point or lies

“between” two transition points. Furthermore, if a configuration is not a transition point, then the applicable reaction is exactly that applicable in the last reached transition point.

For example, the 3-species oscillator described at the start of Sect. 3.2 has A, B , and C as oscillator species. Consider the transition point $\{1A\}$. In this configuration, the only applicable reaction is the reaction $\alpha = A \xrightarrow{C} B$, since A is the only species present. Running α with flux $\frac{1}{2}$, we reach the configuration $\{\frac{1}{2}A, \frac{1}{2}B\}$. Notice that even though we have some amount of B present in this reaction, α is *still* the only applicable reaction, since A inhibits the reaction $\beta = B \xrightarrow{A} C$. β only becomes applicable once we reach the configuration $\{1B\}$, but this is a transition point. This behavior can be generalized as follows:

Observation 3.6. *Let \mathbf{x}, \mathbf{y} be configurations of the iCRN \mathcal{C} described in Sect. 3.2. If \mathbf{x} is a transition point of A , \mathbf{y} is not a transition point, and $\mathbf{x} \rightarrow^1 \mathbf{y}$, then the reactions applicable in \mathbf{y} are exactly the reactions applicable in \mathbf{x} .*

This observation indicates that the applicable reactions of \mathcal{C} changes only upon reaching a new transition point. Therefore, instead of reasoning about arbitrary configurations in concentration space, we can just consider the reachability of transition points. Additionally, observation 3.4 implies that we can assume transition points are reached in a single flux 1 line segment, enabling discrete arguments about the behavior of our construction.

Theorem 1. *Suppose that \mathcal{R} computes a function $f : \mathbb{N} \rightarrow \mathbb{N}$ in the sense that, starting with input register having value n , it halts with output register having value $f(n)$. Then the iCRN \mathcal{C} described above stably computes f from the initial configuration $\mathbf{i} = \{1A_1, nR_{\text{in}}\}$.*

Proof. A complete example of this construction is given in Sect. 3.4.

Let R_{in} be the input species and R_{out} be the output species. For \mathcal{C} to stably compute f , we need that for any valid initial configuration $\mathbf{i} = \{1A_1, nR_{\text{in}}\}$, and any configuration \mathbf{c} such that $\mathbf{i} \rightsquigarrow \mathbf{c}$, there exists a configuration \mathbf{o} such that $\mathbf{c} \rightsquigarrow \mathbf{o}$, $\mathbf{o}(R_{\text{out}}) = f(n)$ and for all \mathbf{o}' such that $\mathbf{o} \rightsquigarrow \mathbf{o}'$, $\mathbf{o}'(R_{\text{out}}) = \mathbf{o}(R_{\text{out}})$.

It suffices to show that for any integer initial concentration of R_{in} , there exists exactly one trajectory, ending in a static configuration \mathbf{h} such that $\mathbf{h}(R_{\text{out}}) = f(n)$.

We first prove that the following invariants hold at every reachable transition point \mathbf{x} .

- (a) For every register species R_j , $\mathbf{x}(R_j) \in \mathbb{N}$.
- (b) Exactly one reaction is applicable in \mathbf{x} (unless $\mathbf{x}(A_i) = 1$ for a halting state i , in which case no reactions are applicable).

We proceed by induction on the number of flux one-line segments connecting transition points \mathbf{i} and \mathbf{c} . (By Observation 3.4 we may assume each segment is not a multiple of the previous.)

For the base case, we show these invariants hold at **i**. Invariant (b) is established for all transition points below, including **i**. By construction, the only register species present in **i** is R_{in} , with concentration $n \in \mathbb{N}$, so invariant (a) is satisfied.

Now, we show the inductive case that if the invariants hold at a transition point **x**, then we can execute the one applicable reaction (guaranteed to exist by invariant (b) unless we have halted) with flux 1, and that this will reach the next transition point **y**, such that the invariants still hold.

First, we claim that at any transition point, **x** with oscillator species O_i having $\mathbf{x}(O_i) = 1$, at most one reaction is possible, exactly 1 if i is a non-halting state, and 0 if i is a halting state and $O_i = A_i$. If O_i is B_i or C_i , this is evident by the fact that each of those is a reactant in exactly one reaction in the network, and at transition points all other oscillator species are absent. In the case $O_i = A_i$, this is again evident if i represents an increment instruction,

since the $C_i \xrightarrow{B_i} A_{i+1} + R_j$ reaction is the only one with C_i as a reactant. If i is a decrement, then C_i is a reactant in two reactions $C_i + R_j \xrightarrow{B_i} A_{i+1}$ and

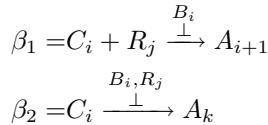
$C_i \xrightarrow{B_i, R_j} A_k$, but one has R_j as a reactant, and the other has R_j as an inhibitor, so exactly one of those two reactions is applicable. This establishes that invariant (b) holds at the next transition point reached, when the applicable reaction is executed for one unit of flux. By Observation 3.4 we assume a single segment applies this reaction until it is inapplicable, reaching the next transition point.

It remains to argue that invariant (a) also holds at the next transition point. Let $O_i \in \{A_1, B_1, C_1, \dots, A_m, B_m, C_m\}$ be an oscillator species and let $\mathbf{x} = \{1O_i, m_1R_1, m_2R_2, \dots, m_nR_n\}$, be a transition point that is reached from **i**. Assume the induction hypothesis that invariants (a) and (b) hold at **x**. Then each $m_i \in \mathbb{N}$ by invariant (a). By (b) **x** has exactly one applicable reaction. If **x** is a transition point of C_i , and state i of the register machine \mathcal{R} is **inc** **r_j**, then

the applicable reaction in **x** is $\alpha = C_i \xrightarrow{B_i} A_{i+1} + R_j$. By construction, there is no

other reaction with C_i as a reactant, and the reaction $A_{i+1} \xrightarrow{C_{j_1}, \dots, C_{j_l}, \dots, C_i, \dots, C_{j_l}} B_{i+1}$ (where each C_{j_x} is a potential predecessor state of $i+1$) is inhibited by C_i , so every sufficiently long path from **x** just executes α until we reach the transition point **y** = $\{1A_{i+1}, m_1R_1, \dots, (m_j+1)R_j, \dots, m_nR_n\}$. So (a) holds.

If line i of \mathcal{R} is instead **dec** **r_j**, **k** then there are reactions



When R_j is present in **x**, the only applicable reaction is β_1 . By a similar argu-

ment to the previous case, the reaction $A_{i+1} \xrightarrow{C_i} B_{i+1}$ is inhibited, so every sufficiently long path from **i** executes β_1 until we reach $\{1B_{i+1}, m_1R_1, \dots,$

$(m_j - 1)R_j, \dots, m_n R_n\}$. If R_j is not present then the only applicable reaction is now β_2 . Then every sufficiently long path from **i** reaches the transition point $\{1B_k, m_1 R_1, \dots, m_n R_n\}$. In either case, invariant (a) holds. This establishes the claim that invariants (a) and (b) hold at each reachable transition point.

We now show that the sequence of states for oscillator species aligns with the execution order of lines in \mathcal{R} and results in a correct simulation of \mathcal{R} . By construction of \mathcal{C} , for each line i of \mathcal{R} of the form **inc** r_j , \mathcal{C} has corresponding

reaction $C_i \xrightarrow{B_i} A_{i+1} + R_j$. By invariant (b), this reaction will be applicable when transition point **c** has a C_i species present, so the next transition point will contain species A_{i+1} . Thus when \mathcal{R} goes from line i to $i + 1$, the present oscillator species in \mathcal{C} simulates this transition in the sense that the subscript i is updated to $i + 1$, and the concentration of R_j increases by 1. Similarly, for each

line i of \mathcal{R} of the form **dec** r_j, k , there are reactions $C_i + R_j \xrightarrow{B_i} A_{i+1}$ and $C_i \xrightarrow{B_i, R_j} A_k$. When R_j is present, species A_{i+1} is 1, and R_j decreases by 1 at the next transition point, and when R_j is not present, A_k is 1. Thus decrement reactions are also properly simulated by \mathcal{C} .

Since \mathcal{R} halts with its output register having value $f(n)$, and \mathcal{C} simulates \mathcal{R} , by (b) any sufficiently long sequence of reactions will eventually reach some static configuration **h** representing \mathcal{R} 's halting configuration. Furthermore, by (a) the values of the register species at the halting point are equal to the values of the registers in \mathcal{R} when it halts. Thus the configuration contains the correct concentration of R_{out} . Since this is a static (thus stable) configuration, this shows that \mathcal{C} stably computes f . \square

3.4 Example of Construction of iCRN from Register Machine

We demonstrate an example of our construction by translating a register machine \mathcal{R} that computes the function $f(n) = 2n$ to an iCRN \mathcal{C} . The machine \mathcal{R} that computes f requires only input and output registers $r_{\text{in}}, r_{\text{out}}$.

Figure 1 shows a plot of this iCRN's trajectory, under the mass-action rate model for reactants, and where each inhibitor I contributes a term $1/(1+10^5 \cdot [I])$ to the rate of the reaction, as an approximation of "absolute" inhibition.⁷

⁷ The long wave seen in the middle is because the reaction $C_1 + R_{\text{in}} \xrightarrow{B_1} A_2$, when R_{in} starts at 1, has a much slower rate of convergence (linear, compared to exponential convergence when R_{in} starts 2 or higher). Consequently, C_1 from time ≈ 300 to time ≈ 800 , despite being "close" to 0, is decaying to 0 much more slowly than in previous oscillations. Thus C_1 much more strongly inhibits the reaction $A_2 \xrightarrow{C_1} B_2$ than in previous oscillations. A_2 and B_2 are the two species "swapping" very slowly between time 300 and 900.

Instructions	Reactions
1: dec r_in,5	$C_4 \xrightarrow{\perp} A_1$
	$A_1 \xrightarrow{\perp} B_1$
	$B_1 \xrightarrow{\perp} C_1$
	$C_1 + R_{in} \xrightarrow{B_1} A_2$
	$C_1 \xrightarrow{B_1, R_{in}} A_5$
2: inc r_out	$A_2 \xrightarrow{C_1} B_2$
	$B_2 \xrightarrow{A_2} C_2$
	$C_2 \xrightarrow{B_2} A_3 + R_{out}$
3: inc r_out	$A_3 \xrightarrow{C_2} B_3$
	$B_3 \xrightarrow{A_3} C_3$
	$C_3 \xrightarrow{B_3} A_4 + R_{out}$
4: goto 1	$A_4 \xrightarrow{C_3} B_4$
	$B_4 \xrightarrow{A_4} C_4$
	$C_4 \xrightarrow{B_4} A_1$
5: halt	no reactions

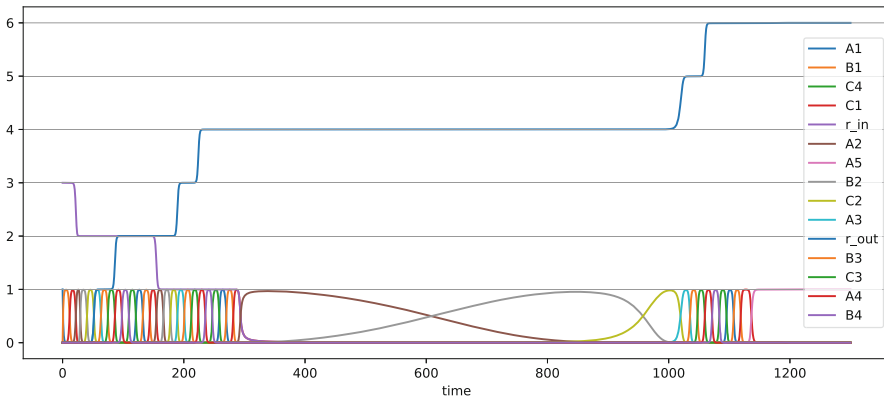


Fig. 1. Plot of iCRN simulating “multiply-by-2” register machine, with input register `r_in` having initial value 3. Note the species `r_in` decrements from 3 down to 0, and the species `r_out` increments from 0 up to 6, while other species oscillate.

4 Conclusion

There are some interesting questions for future research.

Relaxing Absolute Inhibition. The most glaring shortcoming of the inhibitory CRN model is the notion of “absolute” inhibition: any positive concentration of an inhibitor completely disables the reaction. This is clearly unrealistic when taken to extremes: with an enormous amount of reactant R , a tiny amount of I cannot be expected to stop all R from reacting via $R \xrightarrow{I} \dots$. A more realistic model might say that the rate of a reaction is an increasing function of the concentration of its reactants and a decreasing function of the concentration of its inhibitors, for example using a Hill function such as $\frac{[R]}{1+[I]}$ for the rate of the reaction. However, any way of doing this seems to talk about rates, and it is not clear how to meaningfully ask what tasks can be done in a rate-independent way in such a model. One possible way to study this question meaningfully is similar to an approach suggested in the Conclusions of [3] (for studying rate-independence in mass-action CRNs): define a mass-action-like rate law in which a reaction’s rate is a decreasing function of its inhibitors’ concentrations, and allow the adversary to set constant parameters in the rate law, but not to change the rate law itself.

Characterizing Real-Valued Functions. We have demonstrated that the iCRN model is Turing universal in the sense that it can compute any computable function $f : \mathbb{N} \rightarrow \mathbb{N}$. However, the natural data type for continuous iCRNs to process is real numbers. It remains to characterize what functions $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ (or $f : \mathbb{R}_{\geq 0}^d \rightarrow \mathbb{R}_{\geq 0}$) can be stably computed by continuous iCRNs. Using a *dual-rail encoding* to encode a value x as the difference of two concentrations $[X^+] - [X^-]$, one can also meaningfully investigate computation of functions $f : \mathbb{R}^d \rightarrow \mathbb{R}$ with negative inputs and outputs, similar to the characterization of continuous piecewise linear functions stably computable by continuous (non-inhibitory) CRNs using dual-rail encoding [3].

Acknowledgements. DD and KC were supported by NSF awards 2211793, 1900931, 1844976, and DoE EXPRESS award SC0024467.

References

1. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. *Distrib. Comput.* **21**(3), 183–199 (2008). Preliminary version appeared in DISC 2006
2. Bournez, O., Graça, D.S., Pouly, A.: Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. *J. ACM* **64**(6) (2017). <https://doi.org/10.1145/3127496>
3. Chen, H.L., Doty, D., Reeves, W., Soloveichik, D.: Rate-independent computation in continuous chemical reaction networks. *J. ACM* **70**(3) (2023). <https://doi.org/10.1145/3590776>

4. Cooper, C., Lamani, A., Viglietta, G., Yamashita, M., Yamauchi, Y.: Constructing self-stabilizing oscillators in population protocols. *Inf. Comput.* **255**, 336–351 (2017)
5. Fages, F., Le Guludec, G., Bournez, O., Pouly, A.: Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In: *International Conference on Computational Methods in Systems Biology*, pp. 108–127. Springer (2017)
6. Minsky, M.L.: *Computation*. Prentice-Hall Englewood Cliffs (1967)
7. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. *Nat. Comput.* **7**(4), 615–633 (2008). <http://dx.doi.org/10.1007/s11047-008-9067-y>
8. Srinivas, N., Parkin, J., Seelig, G., Winfree, E., Soloveichik, D.: Enzyme-free nucleic acid dynamical systems. *Science* **358**(6369) (2017)