Robust Real-time Computing with Chemical Reaction Networks [⋆]

Willem Fletcher[1], Titus H. Klinge[2][0000−0002−2297−6712], James I. Lathrop[3], Dawn A. Nye[3][0000−0003−1192−2740], and Matthew Rayman[3]

[1] Carleton College, Northfield, MN, USA, fletcherw@carleton.edu
[2] Drake University, Des Moines, IA, USA, titus.klinge@drake.edu
[3] Iowa State University, Ames, IA, USA, jil,omacron,marayman@iastate.edu

Abstract. Recent research into analog computing has introduced new notions of computing real numbers. Huang, Klinge, Lathrop, Li, and Lutz defined a notion of computing real numbers in real-time with chemical reaction networks (CRNs), introducing the classes $\mathbb{R}_{\mathrm{LCRN}}$ (the class of all Lyapunov CRN-computable real numbers) and $\mathbb{R}_{\mathrm{RTCRN}}$ (the class of all real-time CRN-computable numbers). In their paper, they show the inclusion of the real algebraic numbers $ALG \subseteq \mathbb{R}_{\mathrm{LCRN}} \subseteq \mathbb{R}_{\mathrm{RTCRN}}$ and that $ALG \subsetneq \mathbb{R}_{\mathrm{RTCRN}}$ but leave open where the inclusion is proper. In this paper, we resolve this open problem and show $ALG = \mathbb{R}_{\mathrm{LCRN}} \subsetneq \mathbb{R}_{\mathrm{RTCRN}}$. However, their definition of real-time computation is fragile in the sense that it is sensitive to perturbations in initial conditions. To resolve this flaw, we further require a CRN to withstand these perturbations. In doing so, we arrive at a discrete model of memory. This approach has several benefits. First, a bounded CRN may compute values approximately in finite time. Second, a CRN can tolerate small perturbations of its species' concentrations. Third, taking a measurement of a CRN's state only requires precision proportional to the exactness of these approximations. Lastly, if a CRN requires only finite memory, this model and Turing machines are equivalent under real-time simulations.

Keywords: Real Time, Chemical Reaction Networks, Robustness, Analog Computing

1 Introduction

Over the last few decades, many theories of molecular computing have emerged. These theories help inform experimental research and help explore the boundaries of nanoscale computation. Some models of molecular programming are structural, such as algorithmic self-assembly [8, 9]; some models are amorphous, such as chemical reaction networks [4, 15]; and some models combine these to characterize more complex interactions [5, 14]. Since molecular programming is a relatively new field, many open problems exist concerning the computational limits of these models.

[⋆] This research supported in part by NSF grants 1900716 and 1545028.

Investigating the complexity of computing real numbers in computational models has historically significant roots. In Turing's famous 1936 paper [16], he defined a real number to be computable if its "expression as a decimal is calculable with finite means." Real numbers can also be classified according to how efficiently they can be computed by a Turing machine. For example, rational numbers are efficiently computable because their recurring decimal pattern can be produced in real time—even by a finite automaton. More formally, a number $\alpha \in \mathbb{R}$ is *real-time computable by a Turing machine* if n bits of its fractional component can be produced in $O(n)$ time. Many transcendental numbers are known to be real-time computable, but surprisingly, no irrational algebraic number is known to be real-time computable. In fact, in 1965, Hartmanis and Stearns conjectured that if $\alpha \in \mathbb{R}$ is real-time computable by a Turing machine, then it is either rational or transcendental [10].

Recent research into analog computing introduced new notions of computing real numbers. Bournez et al. introduced the notion of computing a real number *in the limit* with a general purpose analog computer (GPAC) [1]. To compute $\alpha \in \mathbb{R}$ "in the limit," a designated variable $x(t)$ must satisfy $\lim_{t\to\infty} x(t) = \alpha$. Computing real numbers in this way has also been investigated in population protocols [2] and chemical reaction networks (CRNs) [11]. Huang et al. defined a number $\alpha \in \mathbb{R}$ to be *real-time computable by chemical reaction networks*, written $\alpha \in \mathbb{R}_{\mathrm{RTCRN}}$, if there exists a CRN with integral rate constants and a designated species X such that, if all species concentrations are initialized to zero, then $x(t)$ converges to α exponentially quickly [12]. This means that after n seconds, the concentration of X is within 2^{-n} of α, so the CRN gains one bit of accuracy every second. Huang et al. also required that all species concentrations be bounded to avoid the so-called *Zeno paradox* of performing an infinite amount of computation in finite time using a fast-growing catalyst species [3]. When this restriction is lifted, the measure of time is no longer linear but rather a function of arc length. In this sense, no power is lost via imposing a boundedness requirement. Further, it eliminates the undesirable Zeno paradox from the model.

A key aspect of Huang et al.'s definition of $\mathbb{R}_{\mathrm{RTCRN}}$ is the requirement that the CRN be initialized to *all zeros*, prohibiting any encoding of α in the initial condition of the CRN. The authors showed that $e, \pi \in \mathbb{R}_{\mathrm{RTCRN}}$, leveraging the fact that the initial condition is *exact*. However, these constructions fail if their initial conditions are perturbed by any $\epsilon > 0$. Huang et al. also defined a subfield of $\mathbb{R}_{\mathrm{RTCRN}}$ they called *Lyapunov CRN-computable real numbers*, written $\mathbb{R}_{\mathrm{LCRN}}$. The definition of $\mathbb{R}_{\mathrm{LCRN}}$ is similar to $\mathbb{R}_{\mathrm{RTCRN}}$ except with the additional constraint that the terminating state of the CRN must be an *exponentially stable equilibrium point*. Since an exponentially stable equilibrium point is *attracting*, any initial condition within its basin of attraction will converge exponentially quickly to it. As a result, any $\alpha \in \mathbb{R}_{\mathrm{LCRN}}$ can be computed even in the presence of bounded perturbations to initial conditions. Huang et al. also proved that $ALG \subseteq \mathbb{R}_{\mathrm{LCRN}} \subseteq \mathbb{R}_{\mathrm{RTCRN}}$ where ALG is the set of algebraic real numbers. The authors left as an open problem which of these inclusions is strict.

An additional consequence of computing a real number α "in the limit" with CRNs is that recovering the bits of α is difficult. Even if we produce α *exactly* in the concentration of a species X, we cannot read its individual bits without an infinitely precise measurement device. Alternatively, if a CRN produced the bits of α as a sequence of measurable memory states, then the bits can be read even with imperfect measurements.

Another limitation of this method of computation is in implementation. The concentration of a species in a solution containing a CRN is ultimately determined by the discrete, integral count of the species. This places a countable limit on the number of "exact" values a concentration can achieve even when a CRN is otherwise perfectly initialized and executed. In the mass action model, we often wave away this issue precisely because we do not have an infinitely precise measurement device. This does, however, somewhat obviate the point of being able to calculate values precisely. In fact, previous results concerning CRNs frequently abuse this hand waving to reach theorems that are true of the mass action kinematics but not of the reality it models. Instead, a more reasonable question to ask is what values can we calculate robustly, quickly, and approximately.

In this paper, we show $ALG = \mathbb{R}_{\text{LCRN}} \subsetneq \mathbb{R}_{\text{RTCRN}}$ to resolve the open problem stated above. This fully characterizes what values we may compute robustly and quickly; however, this definition of computation yet suffers from the inherent flaws described above. To resolve this weakness of the model, we acknowledge these limitations and loosen the definition of computation to accept approximate results. To do so, we only require that a CRN produces approximations of numbers in the sense that an open interval around a concentration α is in an equivalence class with α itself. This approach has three major benefits. First, a bounded CRN may compute not only a single value in finite time but also a sequence of values. Second, a CRN can tolerate small perturbations of its species' concentrations (and potentially other parameters). Third, taking a measurement of a CRN's state only requires precision proportional to the smallest of these intervals.

If we then fix a collection of these intervals into collection of memory maps for a CRN's species and allow it to compute their corresponding memory states in sequence, we obtain a discrete model characterizing a robust chemical computer. Indeed, given, in this sense, a robust CRN and a memory map which fully describes it, a Turing machine may simulate the CRN by maintaining a tape for each of its species indicating what memory state that species is in. We show that this simulation can be done in real-time for CRNs which use only a finite amount of memory. Although we conjecture that CRNs which use an unbounded amount of memory can also be simulated in real-time (which, if true, would unify the analog and discrete Hartmanis-Stearns Conjectures), finite memory suffices for many real world applications.

The rest of the paper is organized as follows, with many proofs omitted for brevity. Section 2 reviews some necessary preliminaries used in the remainder of the paper. Section 3 resolves the open problem $ALG = \mathbb{R}_{\text{LCRN}} \subsetneq \mathbb{R}_{\text{RTCRN}}$.

Section 4 characterizes CRNs in terms of a robust memory map. Lastly, Section 5 discusses the consequences of the proceeding sections.

2 Preliminaries

A *Chemical Reaction Network* (CRN), N, is a tuple $N = (S, R)$, where S is a finite number of species and R is a finite set of reactions on those species. In this paper we investigate *deterministic* CRNs, i.e., CRNs under deterministic mass action semantics that are modeled with systems of differential equations [6]. Given a deterministic CRN, let $x_i(t)$ denote the real-valued concentration of X_i at time t for each species $X_i \in S$. Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote the state of N, where $n = |S|$. We write the rate of change of each x_i as $\frac{dx_i}{dt} = f_i(x_1, \ldots, x_n)$ and the rate of change of the entire system as $\frac{d\mathbf{x}}{dt} = \mathbf{f}_N = (f_1, \ldots, f_n)$. Each f_i is a polynomial determined by N [6]. In this paper, rate constants for each reaction in R are integral, and thus each $f_i \in \mathbb{Z}[x_1, \ldots, x_n]$. Furthermore, the initial concentrations of the species, given by an initial state $\mathbf{x}(0) = \mathbf{x}_0$, along with $\mathbf{f}_N$ determine the unique behavior of N. Lastly, when $f_N(\mathbf{z}) = 0$, we call $\mathbf{z}$ a fixed point.

The definition of real-time computable by a CRN used in this paper is given by [11, 12]. We repeat the definition here for convenience.

Definition 1. *A real number α is* real-time computable by CRNs *if there exists a CRN $N = (S, R)$ and a species $X \in S$ with the following properties:*

1. *(*Integrality.*) All rate constants of R are positive integers.*
2. *(*Boundedness.*) The concentration $x_i(t)$ for each species in S is bounded by a constant β for all time $t \in [0, \infty)$ when $\boldsymbol{x}_0 = 0$.*
3. *(*Real-Time Convergence.*) If N is initialized with $\boldsymbol{x}_0 = 0$, then for all times $t \geqslant 1$, $|x(t) - |\alpha|| < 2^{-t}$.*

We denote the set of all real-time CRN-computable real numbers as $\mathbb{R}_{RTCRN}$.

Excluding the species that converges to α, the above definition places no restrictions on any species beyond that they be bounded. In many cases, this may be undesirable. The next definition formalizes the notion of converging to a single state, at which point the CRN can be considered finished.

Definition 2. *An* exponentially stable point *of a CRN is a state $\boldsymbol{z} \in \mathbb{R}^n_{\geqslant 0}$ for which there exists $\alpha, \delta, C > 0$ such that, if the CRN is initialized to a state $\boldsymbol{x}_0$ satisfying $|\boldsymbol{z} - \boldsymbol{x}_0| < \delta$, then for all times $t \geqslant 0$, $|\boldsymbol{z} - \boldsymbol{x}(t)| \leqslant Ce^{-\alpha t}|\boldsymbol{z} - \boldsymbol{x}_0|$.*

Definition 3. *A real number α is* Lyapunov-CRN computable *if there exists a CRN $N = (S, R)$, a species $X_i \in S$, and a state $\boldsymbol{z}$ with $\boldsymbol{z}(X_i) = |\alpha|$ that satisfies the following properties:*

1. *(*Integrality.*) All rate constants of R are positive integers.*
2. *(*Boundedness.*) The concentration $x_i(t)$ for each species in S is bounded by a constant β for all time $t \in [0, \infty)$ when $\boldsymbol{x}_0 = 0$.*

3. *(*Exponential Stability.*) $\boldsymbol{z}$ is an exponentially stable point.*
4. *(*Convergence.*) If N is initialized with $\boldsymbol{x}_0 = 0$, then $\lim\limits_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{z}$.*

We denote the set of all Lyapunov-CRN computable real numbers as $\mathbb{R}_{LCRN}$.

Observation 4 *If $\boldsymbol{z}$ is an exponentially stable point of a CRN, then it is a fixed point of that CRN.*

Note that the converse of Observation 4 is not true.

We use ALG to denote the set of real algebraic numbers of the rationals. This is the set of real numbers which are the root of some polynomial $f \in \mathbb{Q}[x]$, with rational coefficients.

3 Lyapunov Reals are Algebraic

To investigate robustness issues in real-time computing, we first look at the relationship between $\mathbb{R}_{\text{LCRN}}$ and ALG and show that $\mathbb{R}_{\text{LCRN}} = ALG$. As a consequence, a bounded CRN may only compute the algebraic numbers reliably in the sense that they exist inside of a potential well. Since Huang et al. proved that $ALG \subsetneq \mathbb{R}_{\text{RTCRN}}$ and $ALG \subseteq \mathbb{R}_{\text{LCRN}} \subseteq \mathbb{R}_{\text{RTCRN}}$ [12], it suffices to show that $ALG = \mathbb{R}_{\text{LCRN}}$ to resolve that $\mathbb{R}_{\text{LCRN}} \subsetneq \mathbb{R}_{\text{RTCRN}}$. We prove this result in two parts. First, we show that every exponentially stable fixed point is isolated. Second, we show that isolated fixed points necessarily have algebraic components.

Let E_N denote the set of exponentially stable points of a CRN, N, and let F_N denote the set of fixed points of N. Recall that fixed points are not necessarily isolated (consider a CRN which does nothing once initialized), however, the set of exponentially stable fixed points, E_N, *are* isolated in F_N (not just E_N).

Below are two supporting lemmas, as described above. The proofs are omitted for brevity.

Lemma 5. *If $\boldsymbol{z}$ is an exponentially stable point of a CRN, N, then $\boldsymbol{z}$ is isolated in F_N.*

Lemma 6. *If $\boldsymbol{z}$ is a fixed point of a CRN, N, that is isolated in F_N, then the components of $\boldsymbol{z}$ are in ALG.*

Using these lemmas, it is now straightforward to prove the theorem.

Theorem 7. *$ALG = \mathbb{R}_{LCRN}$*

Proof. Let $\alpha \in \mathbb{R}_{\text{LCRN}}$, and let N, X_i, and $\boldsymbol{z}$ be the CRN, designated species, and exponentially stable point that testify to this. By definition, $\boldsymbol{z}$ is exponentially stable; by Lemma 5, $\boldsymbol{z}$ is isolated in F_N; by Lemma 6, every component of $\boldsymbol{z}$ is algebraic. Thus, $\boldsymbol{z}(X_i) = |\alpha|$ is algebraic, and therefore $\alpha \in ALG$. □

4 A Robust Notion of Memory in CRNs

In the previous section, we concerned ourselves with CRNs which are permitted infinite precision to compute real values robustly in the limit. This excuses several impossibilities for the elegance of its model at the expense of realism. In practice, these CRNs would compute their intended values robustly in approximation and would require only finite time.

In this section we explore the consequences of requiring a CRN to be robust in this sense, that is that they compute values approximately in finite time. In particular, we characterize the behavior of these robust CRNs in terms of these approximations to arrive at a somewhat paradoxical discrete model of analog computing.

Recall that boundedness is one of the three criteria for a real-time CRN. For this section, we use the following definitions of boundedness.

Definition 8. *A CRN $N = (S, R)$ is* β-bounded *at $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$ if, when initialized to $\boldsymbol{x_0}$, there exists some $\beta > 0$ such that $x < \beta$ for each $X \in S$. Moreover, N is* uniformly β-bounded *on $O \subseteq \mathbb{R}^S_{\geqslant 0}$ if there is some $\beta > 0$ for which N is bounded on each $\boldsymbol{x_0} \in O$ by β.*

Unless otherwise specified, a bounded CRN is initialized to the point at which it is bounded. Similarly, a uniformly bounded CRN is initialized to a point at which it is bounded (and is implicitly bounded at any initial point).

There are two natural ways by which a CRN may compute a number α. It may either do so exactly when a species' concentration becomes α or in the limit as per Lyapunov-CRN computability, real-time computability, or some slower manner. Both approaches, however, are imperfect. In the latter case, the concentration of the species computing α either must always maintain a non-zero distance from α after any finite time or, at best, suffers from the same limitation of computing α exactly: the inability to remain at α. The following theorem and corollary formalize this notion.

Theorem 9. *Let $N = (S, R)$ be a bounded CRN. For each species $X \in S$, x is either constant or the set of times for which $\frac{dx}{dt} = 0$ is countable.*

Corollary 10. *Let $N = (S, R)$ be a bounded CRN. Pick $c \in \mathbb{R}_{\geqslant 0}$. Then for any non-constant species $X \in S$, the set of times $t \in \mathbb{R}_{\geqslant 0}$ where $x(t) = c$ is countable.*

It is clear from Corollary 10 that computing an exact value with a CRN is, if not impossible, then a less meaningful concept than one would prefer. This is not inherently problematic as a model of computation. A CRN is capable of computing any computable function in the limit [7].

In each of these models, however, there is the implicit assumption that a CRN may be precisely constructed by which we mean each rate constant and the initial concentration of each species is exactly as prescribed. In practice, this is impractical, which leads us to a notion of robustness. A CRN, informally speaking, is "robust" if it can tolerate a small perturbation of its concentrations

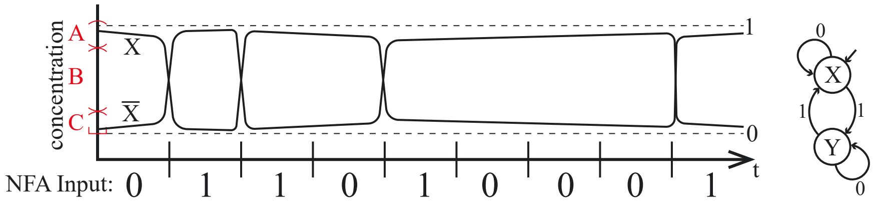

Fig. 1. In [13], Klinge, Lathrop, and Lutz provide a general CRN construction for nondeterministic finite automata (NFAs). These NFAs utilize a dual rail system for each state Z with $z(t) \approx 1$ indicating that the NFA is in state Z at time t and $z(t) \approx 0$ indicating the NFA is not in state Z while the complementary species $\overline{Z}$ has the opposite meaning. Above, we graph the concentration of X and $\overline{X}$ as input changes and show the approximation regions.

(or rate constants) at any time without affecting its function. This is intuitively a difficult task since changing any such condition clearly alters the solution to the system of ODEs describing the CRN.

Exponentially stable points are a good example of robustness in the following sense. If a CRN manages to get within an ϵ-ball of such a point $\boldsymbol{z}$, it proceeds to $\boldsymbol{z}$ in the limit without exception. Ideally, a robust CRN would transition from exponentially stable point to exponentially stable point during its computation with some outside force periodically driving it away from each stable equilibrium.

Exponential stability is a far stricter requirement than is necessary to compute a number α, but it does illustrate an important point. If a CRN computes α either in the limit or for longer than a countable set of times, there is always a buffer zone around it which must necessarily be considered in an equivalence class with α. In Figure 1, this corresponds to the intervals labeled A, B, and C which could be considered equivalence classes for 1, $\frac{1}{2}$, and 0 respectively. We formalize this notion in the following theorems and definitions.

Theorem 11. *Let $N = (S, R)$ be a bounded CRN, and let $X \in S$ be a non-constant species. For any time $t_0 \in \mathbb{R}_{\geqslant 0}$, there exists a $\delta > 0$ such that for all $t \in (t_0, t_0 + \delta)$, $x(t) \neq x(t_0)$. Moreover, there exists an $\epsilon > 0$ and a $t \in (t_0, t_0 + \delta)$ such that $|x(t) - x(t_0)| > \epsilon$.*

In light of Theorem 11, we state a notion of computation useful (but alone insufficient) for CRNs.

Definition 12. *A CRN $N = (S, R)$* (ϵ, d)-computes *a real number $\alpha \in \mathbb{R}_{\geqslant 0}$ if there is an $X \in S$ and a time $t_0 \in \mathbb{R}_{\geqslant 0}$ such that $|x(t) - \alpha| < \epsilon$ for all $t \in (t_0, t_0 + d)$.*

Less formally, a CRN (ϵ, d)-*computes* a real number α if it gets close enough to it for a long enough time. To continue our earlier example, the correct choice of ϵ and d make $x(t)$ correctly compute 0 and 1 but never the garbage state $\frac{1}{2}$ in

Figure 1. This underscores that the particular choice of these two parameters is critical for the CRN's intended purpose. Indeed, a species X of a *bounded* CRN so computes every element of the closure of its image for some single choice of d for every ϵ and vice versa! The latter is obvious (pick ϵ to be larger than the CRN's bound), and we formally state the former.

Theorem 13. *Let $N = (S, R)$ be a bounded CRN, and let $\epsilon > 0$. Then there exists a $d > 0$ such that each $X \in S$ (ϵ, d)-computes every element of $cl(x(\mathbb{R}_{\geqslant 0}))$.*

The following definition resolves this ϵ, d conundrum described above by eliminating any overlap of (ϵ, d)-computed real numbers.

Definition 14. *A CRN $N = (S, R)$* unambiguously computes *a set $A \subseteq \mathbb{R}_{\geqslant 0}$ if for each $\alpha \in A$ there exists a species $X \in S$ which $(\epsilon_\alpha, d_\alpha)$-computes α for some $\epsilon_\alpha, d_\alpha > 0$ and for each distinct $\alpha_1, \alpha_2 \in A$ which X $(\epsilon_{\alpha_1}, d_{\alpha_1})$-computes and $(\epsilon_{\alpha_2}, d_{\alpha_2})$-computes respectively, the intervals $(\alpha_1 - \epsilon_{\alpha_1}, \alpha_1 + \epsilon_{\alpha_1})$ and $(\alpha_2 - \epsilon_{\alpha_2}, \alpha_2 + \epsilon_{\alpha_2})$ are disjoint.*

This notion of unambiguous computation leads directly to a robust notion of CRN memory, but we first state a motivating theorem behind its construction.

Theorem 15. *No CRN can unambiguously compute a somewhere dense subset D of $\mathbb{R}_{\geqslant 0}$ for any choice of $\epsilon_\alpha, d_\alpha > 0$ for each $\alpha \in D$.*

Theorem 15 shows that many natural encodings of countably infinite sets to bounded intervals cannot be unambiguously computed by a CRN. An example of such is given below where we encode $1 \to .1$, $2 \to .01$, $3 \to .11$, and so on.

Corollary 16. *Let $f : \mathbb{N} \to [0, 1]$ be the map*

$$f(n) = \sum_{i=0}^{\infty} \left(\left\lfloor \frac{n}{2^i} \right\rfloor \mod 2 \right) 2^{-i-1}.$$

No CRN can unambiguously compute the set $f(\mathbb{N})$.

To avoid this problem, any encoding requires an open interval around each value α the CRN must compute wherein the entire interval is considered to be α. Moreover, a CRN can only have countably many such disjoint sets. In our running example, Figure 1 demonstrates three such intervals for each state species. This leads to the following definition wherein we encode a collection of disjoint open intervals to map to identifying natural numbers.

Definition 17. *Let $c \in \mathbb{R}^+$. A* memory map *is a map $f : \overline{\mathbb{N}} \to \mathscr{P}([0, c])$ satisfying the following conditions:*

- *$f(0) = [0, b)$, where $0 < b \leqslant c$.*
- *$\forall n \in \mathbb{Z}^+$, $f(n) = (a, b)$, where $a, b \in \mathbb{Q}$ and $0 < a \leqslant b \leqslant c$.*
- *$\forall m, n \in \mathbb{N}$, if $m \neq n$, then $f(n)$ and $f(m)$ are disjoint.*
- *$f(\infty) = [0, c] \setminus \bigcup_{n \in \mathbb{N}} f(n)$ and is countable.*

Definition 18. *The set of all memory maps on* $[0, c]$ *is* $\mathcal{M}_c$*. The* order *of* $f \in \mathcal{M}_c$*, written* $ord(f)$*, is the cardinality of the support of* f *over* $\mathbb{N}$*.*

Definition 19. *The* inverse memory map *of* $f \in \mathcal{M}_c$ *is a map* $f^{\leftarrow} : [0, c] \to \overline{\mathbb{N}}$ *such that for all* $r \in [0, c]$*,* $r \in f(f^{\leftarrow}(r))$*.*

In principle, a CRN cannot reasonably be initialized to any state more precise than to an interval of a memory map. Indeed, the consequence of Corollary 10 is the well known fact that if a species ever has a non-zero concentration, it will at almost every time $t > 0$, so no power is gained from being able to initialize a species to 0.

Before proceeding, the definition of a memory map, it should be noted, is *descriptive* of a CRN, not *prescriptive.* Any memory map can model any CRN, but not all memory maps model any particular CRN *well.* For example, any β-bounded CRN can be modeled by the uninteresting memory map that maps every concentration less than β to 0. Similarly, a memory map with randomly chosen intervals is both equally valid and equally ill-suited. We do not yet, however, have all of the definitions necessary to describe what makes for a *good* choice of memory map and so return to this topic later in this section.

Now equipped with a notion of memory, we must define the trajectory of a species X through that memory (and a CRN's trajectory in terms of its species'). This is not inherently clear because a species X must pass over all intermediate memory locations when transitioning between two non-adjacent states. Even if there are only finitely many such intermediary states, including them in the trajectory provides no additional information. That X passes through them during the transition is a direct consequence of x being continuous. In Figure 1, for example, we never want to include the B interval in our trajectory.

But since each memory state consists of an open interval, X must spend a non-zero length of time inside of it. This brings us back to the definition of (ϵ, d)-computability. If we require X to (ϵ, d)-compute the midpoint of the interval of a memory state with ϵ being half of the interval's width and d being an adjustable parameter, we can arrive at a useful definition of trajectory. To fully formalize this, however, we first have to develop a bit more notation.

Definition 20. *Let* $N = (S, R)$ *be a* β*-bounded CRN. For each* $X \in S$*, let* $f_X \in \mathcal{M}_\beta$ *be a memory map. A species* $X \in S$ *is in the* memory state $m \in \overline{\mathbb{N}}$ *at time* t *if* $x(t) \in f_X(m)$*. Similarly,* N *is in the memory state* $\boldsymbol{m} \in \overline{\mathbb{N}}^S$ *at time* t *if for each* $X \in S$*,* X *is in the memory state* $\boldsymbol{m}(X)$*.*

Definition 21. *Let* N *be a* β*-bounded CRN. For* $X \in S$*, let* $f_X \in \mathcal{M}_\beta$ *be a memory map.* X enters *a memory state* $n \in \overline{\mathbb{N}}$ *at time* t_0 *if there exists an* $\epsilon > 0$ *such that for all* $0 < \epsilon' < \epsilon$*,* $x(t_0 - \epsilon') \notin f_X(n)$ *and* $x(t_0 + \epsilon') \in f_X(n)$*. Similarly,* X leaves n *at time* t_1 *if there exists an* $\epsilon > 0$ *such that for all* $0 < \epsilon' < \epsilon$*,* $x(t_1 - \epsilon') \in f_X(n)$ *and* $x(t_1 + \epsilon') \notin f_X(n)$*. The* state time *of* X *in* n *(with no intermediate states) is the difference* $t_1 - t_0$*.*

There are a few consequences to the above definitions worth mentioning. When a species is initialized to a memory state n, it leaves n without first having entered n. A species may also transition from $n \in \mathbb{N}$ to ∞ (or in other words, it may touch the boundary of n) and then return to n, in which case it does not leave or enter n. This is a desirable property as ∞ is not a useful memory state except, perhaps, in the limit as $t \to \infty$. Further, no species may enter or leave the memory state ∞ by definition.

There remain a few edge cases in the above definitions. If a species never entered a memory state n before it leaves n (i.e. it was initialized to n), then we say it entered at time $t = 0$. In a similar vein, if a species never leaves a memory state, we say it leaves at $t = \infty$ purely as a matter of notational convenience (even if in the limit it transitions to the memory state ∞).

Lastly, we remark that sojourn time (arc length) is generally a better measure of runtime for CRNs [3]. In the case of bounded CRNs, however, state time suffices as it is always within a constant factor of sojourn time. This is because each species concentration of a bounded CRN necessarily has a bounded rate of change.

We now have the tools necessary to define the trajectory of a CRN. We first give an informal description here with example and then rigorously define it (see Definition 22). The trajectory of a β-bounded CRN N initialized to $\boldsymbol{x_0}$ is the ordered sequence of memory states obtained as follows. Start from the initial memory state $\boldsymbol{n_0}$. Each time one or more species enters a new memory state for which its state time is at least d, append the new state of N to the sequence. Continue indefinitely or until there are no further memory state changes.

This construction avoids the undesirability of recording in-between memory states of other species as they transition to their next memory state. It also has the added benefit of bringing into the trajectory a notion of a species staying in a memory state for a long time. In a species trajectory, we merely record where the species's concentration goes to but not for how long it stays there. In the memory trajectory of a full CRN, however, if a species's memory state only rarely changes, we can see that behavior in how infrequently it changes in comparison to other species. For example, if one wishes to record a species's memory state at regular intervals, the simple solution is to set up a clock with an appropriate period which has no interaction with the rest of the CRN except to place itself into the memory trajectory as a timestamp.

To complete our running example, Figure 1 has the following trajectory (for the species X, $\overline{X}$): $(A, C)(C, A)(A, C)(C, A)(A, C)$. In this case, because the construction of the CRN requires that $x + \overline{x} = 1$, a symmetric choice of intervals A and C across $\frac{1}{2}$ causes X and $\overline{X}$ to always be in opposite states at all times.

Using this intuition, we now formally construct the definition of trajectory as follows.

Definition 22. *Let $N = (S, R)$ be a β-bounded CRN at $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$. For $X \in S$, let $f_X \in \mathcal{M}_\beta$ be a memory map. The* memory trajectory *of N when N is initialized to $\boldsymbol{x_0}$ with delay $d \in \mathbb{R}^+$, written $\boldsymbol{traj}(\boldsymbol{x_0}, d)$, is a sequence of $\mathbb{N}^S$ defined by $\boldsymbol{traj}(\boldsymbol{x_0}, d)(n)(X) = m_{last}(X, \boldsymbol{x_0}, \boldsymbol{m}^n_{next}(\boldsymbol{x_0}, 0, d), d)$ where m_{last} and $\boldsymbol{m}^n_{next}$ are helper functions defined below.*

To define the helper functions in the above definition, let $N = (S, R)$ be a β-bounded CRN at $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$. For $X \in S$, let $f_X \in \mathcal{M}_\beta$ be a memory map. Let $T(X, \boldsymbol{x_0}, d)$ be the set of times when X enters a memory state for which its state time is at least $d \in \mathbb{R}^+$ when N is initialized to $\boldsymbol{x_0}$, and define $\boldsymbol{T}(\boldsymbol{x_0}, d) = \bigcup\limits_{X \in S} T(X, \boldsymbol{x_0}, d)$.

Next define $\boldsymbol{m}_{\text{next}} : \mathbb{R}^S_{\geqslant 0} \times \mathbb{R}_{\geqslant 0} \times \mathbb{R}^+ \to \mathbb{R}_{\geqslant 0}$ to be the function which, given an initial state $\boldsymbol{x_0}$ of N, a time $t \in \mathbb{R}_{\geqslant 0}$, and a delay $d > 0$, selects the least $t_0 \in \boldsymbol{T}(\boldsymbol{x_0}, d)$ for which $t_0 > t$.

First, by Corollary 10, $f_X^{\leftarrow} = \infty$ only when X is instantaneously between memory states or if X is constant and initialized to such a value. Both are undesirable system behavior easily avoided. $\boldsymbol{m}_{\text{next}}$ specifically excludes the former from trajectories while the latter is a mere matter of initialization. Unless otherwise specified, we *never* initialize a CRN to such a state even if it is a state for which the CRN is bounded. Second, there is always a least element of each $\boldsymbol{T}(\boldsymbol{x_0}, d)$ for $\boldsymbol{m}_{\text{next}}$ to select since a species X can be in at most two memory states (leaving one for the other) per every d interval of time, which we formally state below.

Lemma 23. *Let $N = (S, R)$ be a β-bounded CRN initialized to $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$. Fix a delay $d \in \mathbb{R}^+$. Then for any d interval of time, N's memory trajectory contains at most $2|S|$ memory states.*

From this lemma, $\boldsymbol{m}_{\text{next}}$ is well defined. We extend its definition to a recursive form as follows.

$$\boldsymbol{m}^n_{\text{next}}(\boldsymbol{x_0}, t, d) = \begin{cases} \boldsymbol{m}^{n-1}_{\text{next}}(\boldsymbol{x_0}, \boldsymbol{m}_{\text{next}}(\boldsymbol{x_0}, t, d), d) & \Big| n > 0 \\ t & \Big| n = 0 \end{cases}$$

To finish formalizing the definition of memory trajectory, $m_{\text{last}} : S \times \mathbb{R}^S_{\geqslant 0} \times \mathbb{R}_{\geqslant 0} \times \mathbb{R}^+ \to \mathbb{N}$ is the function which, given a species $X \in S$, an initial state $\boldsymbol{x_0}$ of N, a time $t \in \mathbb{R}_{\geqslant 0}$, and a delay $d > 0$, returns the last memory state $n \in \overline{\mathbb{N}}$ for which species X enters n at a time $t_0 \leqslant t$ when N is initialized to $\boldsymbol{x_0}$. More intuitively, m_{last} remembers the current memory state of a species while it is transitioning to another memory state.

We can at last now state what makes for a good memory map. The guiding principle behind the choice of memory map is that a CRN in a memory state should behave identically going forward regardless of what particular concentration each species has inside of it. In the spirit of Theorem 15, this then leads to the following natural definition.

Definition 24. *Let $N = (S, R)$ be a uniformly β-bounded CRN and, for each $X \in S$, a memory map $f_X \in \mathcal{M}_\beta$. Fix a delay $d \in \mathbb{R}^+$. N is* memory deterministic *(with respect to $\{f_X\}_{X \in S}$ and delay d) if there is a function $\delta : \mathbb{N}^S \to \mathbb{N}^S$ such that if $\boldsymbol{m} \in \mathbb{N}^S$ is a memory state in N's memory trajectory, then the next memory state in N's memory trajectory (if one exists) is $\delta(\boldsymbol{m})$. When no such memory state exists, $\delta(\boldsymbol{m}) = \boldsymbol{m}$.*

It is important to note that the transition function δ described in the above definition is relative to which memory state(s) the CRN it describes may be initialized. The behavior of unreachable memory states are outside the scope of the definition. With respect to any such memory state, δ's behavior is unrestricted. In general, δ itself need not necessarily even be computable (although it typically should be). For a well behaved CRN (one which admits many possible initializations), however, δ must satisfy the definition for every valid initialization simultaneously.

Before moving on, observe that *all* bounded CRNs are memory deterministic for *a* choice of memory map. Recall that the memory map which maps all concentrations to 0 is valid for every CRN. Similarly, each CRN modeled with this memory map is memory deterministic with $\delta(\boldsymbol{m}) = \boldsymbol{m}$. Otherwise put, a good choice of memory map for a CRN requires not just that it be memory deterministic but that it is also sufficiently refined to produce a useful model.

Now, unsurprisingly, the notion of a memory map bears a strong resemblance to a Turing machine tape. We know that CRNs and Turing machines are equivalent models [7]. The question remains, however, if one model can outperform the other in some significant way. We can address this question in one direction by providing a means for a Turing machine to simulate a CRN. In general, this is a difficult task. With memory maps, this becomes easier. Since neither model is allowed to speed up indefinitely, we may treat a single step of a Turing machine as a constant length of time. Then we may define that a Turing machine simulates a CRN N if it follows N's memory trajectory on its tape(s). Formally, we have the following definitions.

Definition 25. *Fix $k \in \mathbb{Z}^+$. Let $\Lambda = \{(\boldsymbol{\omega_n}, t_n)\}_{n\in\mathbb{N}}$ be a sequence of tuples in $(\mathbb{Z}_2^*)^k \times \mathbb{R}_{\geqslant 0}$. A Turing machine with at least k tapes initialized to $\boldsymbol{\omega_0}$* follows *Λ if there is a strictly increasing computable sequence $\{s_n\}_{n\in\mathbb{N}}$ of $\mathbb{N}$ such that for each $i \in \mathbb{Z}_k$, the contents of tape T_i at step s_n is $\boldsymbol{\omega_n}(i)$. Similarly, M* real-time follows *Λ if there is a constant $c > 0$ such that each $s_n \leqslant ct_n$*

Definition 26. *Let $N = (S, R)$ be a (uniformly) β-bounded CRN and, for each $X \in S$, let $f_X \in \mathcal{M}_\beta$. Fix a delay $d \in \mathbb{R}^+$. A Turing machine M* follows N according to $\{f_X\}_{X\in S}$ with delay d *if for each $X \in S$ there exists a computable injective map $itoa_X : \mathbb{N} \to \mathbb{Z}_2^*$ such that M follows*

$$\Lambda = \left\{\left(\boldsymbol{itoa}(\boldsymbol{traj}(\boldsymbol{x_0}, d)(n)), \boldsymbol{m}^n_{next}(\boldsymbol{x_0}, 0, d)\right)\right\}_{n\in\mathbb{N}}$$

when initialized to $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$ (for every initialization $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$ for which N is bounded), where $\boldsymbol{itoa}(\boldsymbol{traj}(\boldsymbol{x_0}, d)(n))(X) = itoa_X(\boldsymbol{traj}(\boldsymbol{x_0}, d)(n)(X))$ for $X \in S$ and $n \in \mathbb{N}$. Similarly, M real-time follows N according to $\{f_X\}_{X\in S}$ with delay d *if M real-time follows Λ when initialized to $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$ (for every initialization $\boldsymbol{x_0} \in \mathbb{R}^S_{\geqslant 0}$ for which N is bounded).*

We can extend our running example to these definitions as follows. For the itoa functions, interval A maps to 1, B maps to 10, and C maps to 0. Moreover, since the CRN was constructed directly from a finite automaton, it only takes two steps to compute each subsequent memory state and write it to the appropriate tape. If follows trivially, then, that there is a Turing machine which real-time follows the CRN.

More generally, since Turing machines and CRNs are equivalent models [7], there is always a Turing machine that follows any CRN N. The more interesting (and far more difficult) question is if there always exists a Turing machine M and some choice of itoa functions for which M real-time follows N. Intuitively, analog computing *should* be more efficient than discrete computing in some respect. Indeed, were a CRN either unbounded or if it were allowed an unbounded number of species, this is easy to show. To see why this is less certain for robust, bounded CRNs, we need a few lemmas.

The natural first question to ask is how can a Turing machine can keep up in real-time with a CRN from the definition of real-time following. A CRN, after all, is allowed to change all of its species concentrations simultaneously while a Turing machine, following the CRN's memory trajectory and not directly simulating the CRN, must keep all but one (except in the unlikely case where two or more species change memory state at the exact same time) of its species-tracking tapes effectively constant between memory trajectory transitions.

This is not a limitation since a species must linger in a memory state for a minimum length of time. A CRN with n species and delay d can only experience at most n memory states in every open interval of length d (see Lemma 23). This is what makes a Turing machine M real-time following a CRN occur in real-time. It follows that M can compute each of these state changes sequentially while only requiring a constant factor of $|S|$ more time in the worst case.

The remaining difficulty is to show that a bounded CRN cannot 'cheat' in the sense that a Turing machine would require an infinite alphabet or an infinite number of states or tapes to real-time follow it. We show this is the case when each memory map has only finite order and the transitions between memory states is memory deterministic.

Theorem 27. *Let $N = (S, R)$ be a uniformly β-bounded CRN, let $f_X \in \mathcal{M}_\beta$ for each $X \in S$ with $ord(f_X) < \infty$, and let $d \in \mathbb{R}^+$. If N is memory deterministic, then there is a Turing machine which real-time follows N according to $\{f_X\}_{X \in S}$ with delay d.*

Corollary 28. *Let $N = (S, R)$ be a uniformly β-bounded CRN, and let $f_X \in \mathcal{M}_\beta$ for each $X \in S$. Fix a delay $d \in \mathbb{R}^+$. If N is memory deterministic and N's memory trajectory is either finite or there exists a memory state which appears at least twice in it, then there is a Turing machine which real-time follows N according to $\{f_X\}_{X \in S}$ with delay d.*

5 Discussion

In this paper, we have shown that *only* the algebraic real numbers are computable by CRNs using exponentially stable equilibria. Intuitively, this means that every transcendental real number cannot be computed robustly by a CRN in the sense of Definition 3. This led us to explore in Section 4 what it means for a CRN to compute robustly. We started from two notions of computation. First, a CRN can compute a value exactly, which a non-constant CRN can achieve only for a measure zero length of time. Second, a CRN may compute a value in the limit, which has two problems of its own. A CRN never precisely achieves a value computed in the limit. Moreover, we showed earlier in Theorem 7 that only the algebraic numbers can be so computed reliably. Any non-algebraic number, if the CRN is improperly initialized with any epsilon error, cannot be computed in the limit.

These limitations led us to ask what happens when we require a CRN to behave identically for a range of inputs rather than a single set of concentrations. The result was the notion of a memory map, a strangely discrete model of an analog implementation of computation. Arguably, under this model, a CRN's reactions correspond to transitions between states of a Turing machine while species concentrations correspond to tape states.

This ultimately led to Theorem 27. In the more familiar terminology of the discrete world, it tells us that a robust CRN is no more capable of executing a NFA than a Turing machine is. This is perhaps unsurprising. The main advantage a CRN has to leverage over a Turing machine is in its ability to rewrite its entire tape with a new word of any length. With finite memory, this advantage is lost.

Notice, however, that Theorem 27 says nothing about the *existence* of a robust CRN capable of simulating a NFA. For that, we turn to [13] for a CRN with a more restrictive notion of robustness which nonetheless satisfies the definitions we derived here and Theorem 27. We briefly summarize this below as an illustrative example.

Given a NFA M, a CRN N is constructed with two species, X_q and $\overline{X}_q$, for each state q of M. These species alternatively take concentrations close to 1 or 0 to represent M being in state q or not in state q respectively for X_q and vice versa for $\overline{X}_q$. The appropriate memory map for each of these species would be to map 0 to an interval around 0, 1 to an interval around 1, and 2 to everything in-between. The correct delay to choose for this CRN is the length of the clock cycle (which also admits an identical memory map). For the input signal (which, again, admits an identical memory map), we may assume that there is an external CRN generating it which makes the N memory deterministic.

First, note that N is uniformly bounded on all of its valid inputs. Moreover, if we apply Theorem 27 to the CRN described above, we obtain a Turing machine which not only behaves identically but can be transformed back into the same CRN [13]. As such, these are truly inverse statements. Moreover, the theorem can be applied to more general cases as well.

Given a Turing machine M, Fages et al. construct a GPAC-generable function (easily translated into the CRN world) that simulates M within bounded time and tape space [7]. The input parameters for each bound can be adjusted, of course, but once fixed, the resulting simulation permits a single memory map model for all of its input configurations to which Corollary 28 applies. In a sense, these, too, are inverse statements.

Now the question becomes where to go from here. It is known that, given an NFA, there is a robust CRN which simulates it in real-time [13]. Similarly, we have provided a proof that, given a robust CRN with memory maps of only finite order, there is a Turing machine which real-time follows it. In short, for the regular languages, robust CRNs and Turing machines are fully equivalent models with neither having an advantage over the other. We conjecture that the same is true of an arbitrary robust CRN, that is given a robust CRN with a memory deterministic collection of memory maps, there is a Turing machine which real-time follows it. This, if true, has several important implications.

First, it's known that CRNs and Turing machines can simulate each other with a polynomial-time slowdown [3]. If this conjecture is true, even in a more restricted form, it would eliminate the slowdown from a Turing machine simulating a CRN.

Of perhaps more interest is the Hartmanis-Stearns Conjecture (HSC) [10]. Both Turing machines and robust CRNs are clearly capable of outputting the digits of a rational number in real-time. For Turing machines, this means writing to some output tape. For CRNs, this *does not* mean outputting a concentration but rather raising a concentration high or low in a memory trajectory in the appropriate sequence. In this manner, assuming our stated conjecture, then if one could construct a robust CRN to output the digits of a nonrational algebraic number, it would also resolve the HSC for Turing machines.

Acknowledgments. The authors thank anonymous reviewers for useful feedback and suggestions. This research supported in part by NSF grants 1900716 and 1545028.

References

1. Bournez, O., Campagnolo, M.L., Graça, D.S., Hainry, E.: The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation. In: Cai, J.Y., Cooper, S.B., Li, A. (eds.) Theory and Applications of Models of Computation. pp. 631–643. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
2. Bournez, O., Fraigniaud, P., Koegler, X.: Computing with large populations using interactions. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) Mathematical Foundations of Computer Science 2012. pp. 234–246. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
3. Bournez, O., Graça, D.S., Pouly, A.: Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. J. ACM **64**(6) (Oct 2017). https://doi.org/10.1145/3127496, https://doi.org/10.1145/3127496
4. Cappelletti, D., Ortiz-Muñoz, A., Anderson, D.F., Winfree, E.: Stochastic chemical reaction networks for robustly approximating arbitrary probability distributions. Theoretical Computer Science **801**, 64–95 (2020). https://doi.org/https://doi.org/10.1016/j.tcs.2019.08.013
5. Clamons, S., Qian, L., Winfree, E.: Programming and simulating chemical reaction networks on a surface. Journal of The Royal Society Interface **17**(166), 20190790 (2020). https://doi.org/10.1098/rsif.2019.0790

6. Epstein, I.R., Pojman, J.A.: An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos. Oxford University Press (1998)
7. 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: Feret, J., Koeppl, H. (eds.) Computational Methods in Systems Biology. pp. 108–127. Springer International Publishing, Cham (2017)
8. Furcy, D., Summers, S.M., Wendlandt, C.: Self-assembly of and optimal encoding within thin rectangles at temperature-1 in 3D. Theoretical Computer Science (2021). https://doi.org/https://doi.org/10.1016/j.tcs.2021.02.001
9. Hader, D., Patitz, M.J.: Geometric tiles and powers and limitations of geometric hindrance in self-assembly. Natural Computing (Mar 2021). https://doi.org/10.1007/s11047-021-09846-2
10. Hartmanis, J., Stearns, R.E.: On the computational complexity of algorithms. Transactions of the American Mathematical Society **117**, 285–306 (1965), http://www.jstor.org/stable/1994208
11. Huang, X., Klinge, T.H., Lathrop, J.I.: Real-time equivalence of chemical reaction networks and analog computers. In: Thachuk, C., Liu, Y. (eds.) DNA Computing and Molecular Programming. pp. 37–53. Springer International Publishing, Cham (2019)
12. Huang, X., Klinge, T.H., Lathrop, J.I., Li, X., Lutz, J.H.: Real-time computability of real numbers by chemical reaction networks. Natural Computing **18**(1), 63–73 (Mar 2019). https://doi.org/10.1007/s11047-018-9706-x
13. Klinge, T.H., Lathrop, J.I., Lutz, J.H.: Robust biomolecular finite automata. Theoretical Computer Science **816**, 114–143 (2020). https://doi.org/https://doi.org/10.1016/j.tcs.2020.01.008
14. Klinge, T.H., Lathrop, J.I., Moreno, S., Potter, H.D., Raman, N.K., Riley, M.R.: ALCH: An Imperative Language for Chemical Reaction Network-Controlled Tile Assembly. In: Geary, C., Patitz, M.J. (eds.) 26th International Conference on DNA Computing and Molecular Programming (DNA 26). Leibniz International Proceedings in Informatics (LIPIcs), vol. 174, pp. 6:1–6:22. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.DNA.2020.6
15. Severson, E.E., Haley, D., Doty, D.: Composable computation in discrete chemical reaction networks. Distributed Computing (May 2020). https://doi.org/10.1007/s00446-020-00378-z
16. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society **s2-42**(1), 230–265 (1937). https://doi.org/10.1112/plms/s2-42.1.230