Chemical Reaction Networks for Computing Polynomials

Ahmad Salehi, Keshab Parhi, and Marc D. Riedel

University of Minnesota

Abstract. Chemical reaction networks (CRNs) are a fundamental model in the study of molecular reactions. Widely used as formalism for the *analysis* of chemical and biochemical systems, CRNs have received renewed attention as a model for molecular *computation*. Prior research has shown that, viewing the concentrations of specific molecular types as inputs and outputs, CRNs can only compute semilinear functions. This paper demonstrates that, with a different *encoding*, CRNs can, in fact, compute a much richer set of functions, namely all polynomial functions. The encoding is a fractional representation: inputs and outputs are represented as the fraction of concentrations of molecular types. The method is illustrated first for generic CRNs; then an example is mapped to DNA strand-displacement reactions.

1 Introduction

It has long been recognized that, viewed from a mathematical standpoint, a set of chemical reactions can exhibit rich dynamical behavior [1]. On the computational front, there has been a wealth of research into efficient methods for simulating chemical reactions, ranging from ordinary differential equations (ODEs) [2] to stochastic simulation [3]. On the mathematical front, entirely new branches of theory have been developed to characterize chemical dynamics [4].

The idea of computation directly *with* with chemical reactions – as opposed to writing computer programs to analyze chemical systems – dates back to the seminal work of Adleman [5]. In this context, a chemical reaction network (CRN) transforms *input* concentrations of molecular types into *output* concentrations and so implements computation.

The question of the computational power of chemical reactions has been considered by several authors. Magnasco demonstrated that chemical reactions can compute anything that digital circuits can compute [6]. Soloveichik *et al.* demonstrated that chemical reactions are *Turing Universal*, meaning that they can compute anything that a computer algorithm can compute [7]. This work was applicable to a discrete, stochastic model of chemical kinetics. The computation is probabilistic; the total probability of error of the computation can be made arbitrarily small (but not zero).

In a more narrow context, Chen *et. al* considered the question of what *functions*, of the form $f : \mathbb{R} \to \mathbb{R}$, CRNs can compute deterministically [9][10]. They proved that CRNs compute exactly the set of **semilinear functions**. That is to say, they proved that a function is computable by a deterministic CRN if and only if it is continuous piecewise linear.

In this prior work, the authors considered – either implicitly or explicitly – two types of *encodings* for the input and output variables of CRNs.

1. The value of each variable corresponds the concentration of a specific molecular type. Call this the **direct** representation.
2. The value of each variable is represented by the difference between the concentrations of a pair of molecular types. Call this the **dual-rail** representation.

In this paper we introduce a new representation, the **fractional** representation. A pair of molecular types is assigned to each variable, e.g., (X_0, X_1) for a variable x. The value of the variable is determined by the ratio of the assigned pair,

$$x = \frac{X_1}{X_0 + X_1}. \tag{1}$$

Evidently, the value is confined to the unit interval, $[0, 1]$. Based on this representation, we propose a CRN framework for computing univariate polynomials that map the unit interval [0,1] to itself. We demonstrate that a CRN exists that computes any such polynomial. This is a much richer set of functions that those in the construction by Chen et al. [9][10]. We first illustrate with a simple example.

Example 1. Consider the following CRN:

$$\begin{array}{ll} & S_0 + B_{0,0} \to Y_0 \\ X_{0,0} + X_{1,0} \to S_0 & S_0 + B_{0,1} \to Y_1 \\ X_{0,0} + X_{1,1} \to S_1 & S_1 + B_{1,0} \to Y_0 \\ X_{0,1} + X_{1,0} \to S_1 & S_1 + B_{1,1} \to Y_1 \\ X_{0,1} + X_{1,1} \to S_2 & S_2 + B_{2,0} \to Y_0 \\ & S_2 + B_{2,1} \to Y_1 \end{array}$$

(a) (b)

Set the initial concentrations as follows:

$$\left.\begin{array}{l} B_{0,0} = 25 \\ B_{0,1} = 75 \end{array}\right\} \quad \to \quad \frac{75}{25+75} = \frac{3}{4}$$

$$\left.\begin{array}{l} B_{1,0} = 75 \\ B_{1,1} = 25 \end{array}\right\} \quad \to \quad \frac{25}{75+25} = \frac{1}{4}$$

$$\left.\begin{array}{l} B_{2,0} = 50 \\ B_{2,1} = 50 \end{array}\right\} \quad \to \quad \frac{50}{50+50} = \frac{1}{2}$$

It may be shown that this CRN computes the function

$$y(x) = \frac{3}{4}x^2 - x + \frac{3}{4}, \tag{2}$$

where $0 \le x \le 1$.

The CRN is composed of two sets of reactions: the four reactions in group (a) that we call them *electing reactions* and the six reactions in group (b) that

we call the *transferring reactions*. We provide details regarding the synthesis method in Section 3. Here we simply note that, given a polynomial $y(x)$, the first step is to convert to its *Bernstein polynomial* equivalent, $g(x)$. For the polynomial $y(x)$ in Eq.2,

$$g(x) = \frac{3}{4}[(1-x)^2] + \frac{1}{4}[2x(1-x)] + \frac{1}{2}x^2. \tag{3}$$

(An explanation of what a Bernstein polynomial is is given in Section 2.) The coefficients of the Bernstein polynomial are the values used to initialize the types $B_{i,0}$ and $B_{i,1}$ for $i = 0, 1, 2$.

Suppose we want to evaluate $y(x)$ at x=0.5. We would initialize $X_{i,0} = X_{i,1}$=100, for i=0,1, such that

$$x = \frac{X_{i,1}}{X_{i0} + X_{i,1}} = 0.5. \tag{4}$$

We would set the initial concentration of the other types to zero. The output value, $y(x)$, is computed as the ratio of the final concentrations of Y_0 and Y_1, i.e.,

$$y(x) = \frac{Y_1}{Y_0 + Y_1}. \tag{5}$$

The simulation results for evaluating this example at x=0.5 using a continuous mass-action kinetics model are shown in Fig.1. As the time $t \to \infty$, the ratio

$$\frac{Y_1(t)}{Y_0(t) + Y_1(t)} \tag{6}$$

approaches the correct value of $y(0.5)$=0.4375.

2 Representation

In our method, the **Bernstein representation** of a polynomial is a key element. We briefly describe the relevant mathematics. The family of $n+1$ polynomials of the form

$$B_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i}, \qquad i = 0, \ldots, n \tag{7}$$

are called Bernstein basis polynomials of degree n. A linear combination of Bernstein basis polynomials of degree n,

$$g(x) = \sum_{i=0}^{n} b_{i,n} B_{i,n}(x), \tag{8}$$

is a Bernstein polynomial of degree n. The $b_{i,n}$'s are called Bernstein coefficients.

Polynomials are usually represented in power form,

$$y(x) = \sum_{i=0}^{n} a_{i,n} x^i, \tag{9}$$

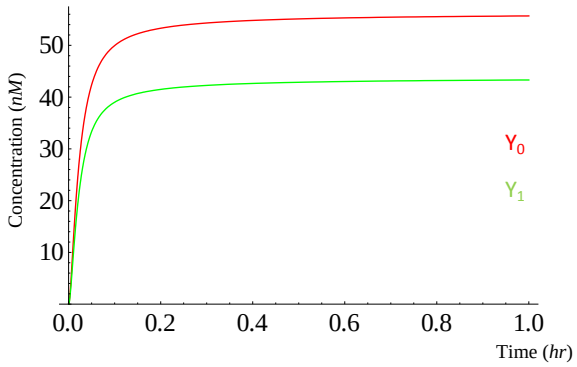

Fig. 1. Simulation results for the CRN implementing the polynomial $y(x) = \frac{3}{4}x^2 - x + \frac{3}{4}$ for $0 \leq x \leq 1$. These were obtained from an ODE simulation of the mass-action kinetics.

We can convert such a power-form polynomial of degree n into a Bernstein polynomial of degree n. The conversion from the power-form coefficients, $a_{i,n}$, to the Bernstein coefficients, $b_{i,n}$, is a closed-form expression:

$$b_{i,n} = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{n}{j}} a_{j,n}, \qquad 0 \leq i \leq n. \tag{10}$$

For a proof of this, the reader is referred to [11]. As an example consider the polynomial

$$y(x) = \frac{5}{4}x^3 - \frac{15}{8}x^2 + \frac{9}{8}x + \frac{1}{4}. \tag{11}$$

It can be converted into the following Bernstein polynomial of degree 3:

$$g(x) = \frac{2}{8}B_{0,3}(x) + \frac{5}{8}B_{1,3}(x) + \frac{3}{8}B_{2,3}(x) + \frac{6}{8}B_{3,3}(x). \tag{12}$$

Generally speaking, a power-form polynomial of degree n can be converted into an equivalent Bernstein polynomial of degree greater than or equal to n. The coefficients of a Bernstein polynomial of degree $m+1$ ($m \geq n$) can be derived from the Bernstein coefficients of an equivalent Bernstein polynomial of degree m

$$b_{i,m+1} = \begin{cases} b_{0,m} & i = 0 \\ (1 - \frac{i}{m+1})b_{i,m} + \frac{i}{m+1}b_{i-1,m} & 1 \leq i \leq m \\ b_{m,m} & i = m+1 \end{cases} \tag{13}$$

Again, for a proof the reader is referred to [11].

By encoding the values of variables as the ratio of the concentrations of two molecular types,

$$x = \frac{X_1}{X_0 + X_1}, \tag{14}$$

we can only represent numbers between 0 and 1. Accordingly, our method synthesizes functions that map the unit interval [0,1] onto itself. As was shown in Example 1, the coefficients of the polynomials that we compute are also represented in this fractional form. Fortunately, Qian *et al.* proved that any polynomial that maps the unit interval onto the unit interval can be converted into a Bernstein polynomial *with all coefficients in the unit interval* [12]. That paper shows that, by repeatedly elevating the degree of a Bernstein polynomial, one always obtains one with coefficients in the unit interval. Therefore, the necessary and sufficient condition for our method is that the target polynomial maps the unit interval to itself. The reader is referred to [12] for further details.

3 Synthesizing CRNs for computing polynomials

In this section we present a systematic methodology for synthesizing CRNs that can compute polynomials. As discussed in the previous section, we assume that the target polynomial is given in Bernstein form, with all coefficients in the unit interval. The method is composed of two parts, designing the CRN and initializing certain types to specific values, as discussed in the following.

3.1 Designing the CRN

The CRN reactions consists of two sets of reactions that we call the *electing reactions* and the *transferring reactions*.

First consider the electing reactions. If the degree of the Bernstein polynomial is m, we consider m pairs of molecular types $(X_{i,0}, X_{i,1})$ for $i = 0, 1, \ldots, m-1$, as reactants. One and only one type of each pair is selected to participate in each reaction. Therefore, each of the electing reactions has m reactants, one from each pair. In this manner, we generate 2^m reactions. Each reaction produces S_j, where j is the index of the reactant chosen from the second type in the pair, i.e., $X_{i,1}$. It is obvious that j can be an integer between 0 and m. For $m = 2$ the electing reactions are:

$$X_{0,0} + X_{1,0} \rightarrow S_0 \tag{15}$$
$$X_{0,0} + X_{1,1} \rightarrow S_1 \tag{16}$$
$$X_{0,1} + X_{1,0} \rightarrow S_1 \tag{17}$$
$$X_{0,1} + X_{1,1} \rightarrow S_2 \tag{18}$$
$$\tag{19}$$

For general m, they are:

$$X_{0,0} + X_{1,0} + X_{2,0} + \cdots + X_{m-1,0} \rightarrow S_0 \tag{20}$$
$$X_{0,1} + X_{1,0} + X_{2,0} + \cdots + X_{m-1,0} \rightarrow S_1 \tag{21}$$
$$X_{0,0} + X_{1,1} + X_{2,0} + \cdots + X_{m-1,0} \rightarrow S_1 \tag{22}$$
$$\vdots$$
$$X_{0,1} + X_{1,1} + X_{2,0} + \cdots + X_{m-1,0} \rightarrow S_2 \tag{23}$$
$$X_{0,0} + X_{1,1} + X_{2,1} + \cdots + X_{m-1,0} \rightarrow S_2 \tag{24}$$
$$\vdots$$
$$X_{0,1} + X_{1,1} + X_{2,1} + \cdots + X_{m-1,1} \rightarrow S_m \tag{25}$$

A degree m Bernstein polynomial has $m+1$ Bernstein coefficients. We consider $m+1$ pairs of types $(B_{j,0}, B_{j,1})$ for $j = 0, 1, \ldots, m$, to represent these coefficients. The transferring reactions produce the final output, Y_0 or Y_1, from the products of the electing reactions, the S_j's. They do so proportionally to the Bernstein coefficients. S_j goes to Y_0 if it combines with $B_{j,0}$ and goes to Y_1 if it combines with $B_{j,1}$. Accordingly, there are $2(m+1)$ transferring reactions reactions. For $m = 2$ the transferring reactions are:

$$S_0 + B_{0,0} \rightarrow Y_0 \tag{26}$$
$$S_0 + B_{0,1} \rightarrow Y_1 \tag{27}$$
$$S_1 + B_{1,0} \rightarrow Y_0 \tag{28}$$
$$S_1 + B_{1,1} \rightarrow Y_1 \tag{29}$$
$$S_2 + B_{2,0} \rightarrow Y_0 \tag{30}$$
$$S_2 + B_{2,1} \rightarrow Y_1 \tag{31}$$
$$\tag{32}$$

For general m, they are:

$$S_0 + B_{0,0} \rightarrow Y_0 \tag{33}$$
$$S_0 + B_{0,1} \rightarrow Y_1 \tag{34}$$
$$S_1 + B_{1,0} \rightarrow Y_0 \tag{35}$$
$$S_1 + B_{1,1} \rightarrow Y_1 \tag{36}$$
$$\vdots$$
$$S_m + B_{m,0} \rightarrow Y_0 \tag{37}$$
$$S_m + B_{m,1} \rightarrow Y_1 \tag{38}$$

3.2 Initialization

We initialize the pair $(B_{i,0}, B_{i,1})$ according to the Bernstein coefficients $b_{i,m}$, i.e., we have

$$b_{i,m} = \frac{B_{i,1}}{B_{i,0} + B_{i,1}}. \tag{39}$$

For simplicity we choose $B_{i,0}$ and $B_{i,1}$ such that the summation $B_{i,0}+B_{i,1}$ is the same for all i's. Call the sum $B_{i,0}+B_{i,1}=B$ for all i.

We initialize the corresponding item in each pair $(X_{i,0}, X_{i,1})$ to the same value for all i, based on the value x_{in} at which the polynomial is to be evaluated, i.e.,

$$x_{in} = \frac{X_{i,1}}{X_{i,0}+X_{i,1}}. \tag{40}$$

Call this common value X_0 for the $X_{i,0}$'s and X_1 for the $X_{i,1}$'s. The other types, namely the S_i's as well as Y_0 and Y_1, are initialized to zero.

4 Proof Based on the Mass-Action Kinetics

We use an ordinary differential model of the mass-action kinetics to prove the correctness of our proposed CRN design. The electing reactions (21)–(25) produce types S_j. The transferring reactions (34)–(38) consume them. Therefore the ODEs for the types S_j are:

$$\frac{dS_0}{dt} = X_{0,0}X_{1,0}\dots X_{m-1,0} - B_{0,0}S_0 - B_{0,1}S_0 = X_0^m - S_0(B_{0,0}+B_{0,1}) \tag{41}$$

$$\begin{aligned}\frac{dS_1}{dt} &= X_{0,1}X_{1,0}\dots X_{m-1,0} + X_{0,0}X_{1,1}\dots X_{m-1,0} + \dots + X_{0,0}X_{1,0}\dots X_{m-1,1} - B_{1,0}S_1 - B_{1,1}S_1 \\ &= \binom{m}{1}X_0^{m-1}X_1 - S_1(B_{1,0}+B_{1,1})\end{aligned} \tag{42}$$

$$\vdots$$

$$\begin{aligned}\frac{dS_k}{dt} &= X_{0,1}X_{1,1}\dots X_{m-1,0} + X_{0,0}X_{1,1}\dots X_{m-1,0} + \dots + X_{0,0}X_{1,0}\dots X_{m-1,1} - B_{k,0}S_k - B_{k,1}S_k \\ &= \binom{m}{k}X_0^{m-k}X_1^k - S_k(B_{k,0}+B_{k,1})\end{aligned} \tag{43}$$

$$\vdots$$

$$\frac{dS_m}{dt} = X_{0,1}X_{1,1}\dots X_{m-1,1} - B_{m,0}S_m - B_{m,1}S_m = X_1^m - S_m(B_{m,0}+B_{m,1}). \tag{44}$$

Recall that we assumed that $X_{i,0}=X_0$ and $X_{i,1}=X_1$ for all i's. At equilibrium $\frac{dS_i}{dt}$=0. Accordingly, we can compute the S_i's as:

$$S_i = \frac{\binom{m}{i}X_0^{m-i}X_1^i}{B_{i,0}+B_{i,1}} \quad 0 \le i \le m. \tag{45}$$

Now we write the ODEs for the output types Y_0 and Y_1. Based on the transferring reactions (34)–(38), we have:

$$\begin{aligned}\frac{dY_0}{dt} &= B_{0,0}S_0 + B_{1,0}S_1 + \dots + B_{m,0}S_m \\ \frac{dY_1}{dt} &= B_{0,1}S_0 + B_{1,1}S_1 + \dots + B_{m,1}S_m\end{aligned} \tag{46}$$

According to the fractional encoding, the output value, y, is calculated as follows.

$$\begin{aligned} y &= \frac{Y_1}{Y_0 + Y_1} = \frac{dY_1}{dY_0 + dY_1} \\ &= \frac{B_{0,1}S_0 + B_{1,1}S_1 + \ldots + B_{m,1}S_m}{(B_{0,0}S_0 + B_{1,0}S_1 + \ldots + B_{m,0}S_m) + (B_{0,1}S_0 + B_{1,1}S_1 + \cdots + B_{m,1}S_m)} \end{aligned} \tag{47}$$

By substituting S_i from Eq. (45) and with the assumption that $B_{i,0}+B_{i,1} = B$ for all i's, we have:

$$\begin{aligned} y &= \frac{B_{0,1}S_0 + B_{1,1}S_1 + \cdots + B_{m,1}S_m}{(B_{0,0} + B_{0,1})S_0 + (B_{1,0} + B_{1,1})S_1 + \cdots + (B_{m,0} + B_{m,1})S_m + (S_0 + B_{1,1}S_1 + \cdots + B_{m,1}S_m)} \\ &= \frac{B_{0,1}S_0 + B_{1,1}S_1 + \cdots + B_{m,1}S_m}{B(S_0 + S_1 + \cdots + S_m)} \\ &= \frac{\sum_{i=0}^{m} B_{i,1}S_i}{B(\sum_{i=0}^{m} S_i)} \\ &= \frac{\sum_{i=0}^{m} B_{i,1}\frac{\binom{m}{i}X_0^{m-i}X_1^i}{B}}{B(\sum_{i=0}^{m}\frac{\binom{m}{i}X_0^{m-i}X_1^i}{B})} \end{aligned} \tag{48}$$

We know that $\sum_{i=0}^{m}\binom{m}{k}X_0^{m-i}X_1^i = (X_0 + X_1)^m$, therefore, the denominator can be replaced by $(X_0 + X_1)^m$.

$$\begin{aligned} y &= \frac{\sum_{i=0}^{m} B_{i,1}\frac{\binom{m}{i}X_0^{m-i}X_1^i}{B}}{(X_0 + X_1)^m} \\ &= \sum_{i=0}^{m}\frac{B_{i,1}}{B}\binom{m}{i}\frac{X_0^{m-i}X_1^i}{(X_0 + X_1)^m} \\ &= \sum_{i=0}^{m} b_{i,m}\binom{m}{i}(1-x)^{m-i}x^i \end{aligned} \tag{49}$$

Eq. (49) is exactly the expression for a Bernstein polynomial of degree m. Thus, our CRN computes $y(x)$.

5 DNA Implementation

The proposed CRN for computing polynomials is general in the sense that it can be implemented by any chemical system with mass-action kinetics. As a practical medium, we choose DNA strand-displacement reactions. Indeed, Soleveichik et al. demonstrated that DNA strand-displacement reactions can emulate the kinetics of *any* CRN [16]. They presented a software tool that maps chemical CRNs to DNA reactions. Their tool can only map unimolecular and bimolecular reactions, i.e., reactions with one or two reactants. Our CRNs, however, include reactions with more than two reactants, for target polynomials of degree $n \geq 3$. Thus, in order to map these reactions to DNA strand-displacement reactions, we first break them down into bimolecular reactions.

Specifically, the reactions in the electing set (21)–(25) may have more than two reactants. Consider the case for $n = 3$:

$$\begin{array}{ll} X_{0,0} + X_{1,0} + X_{2,0} \to S_0 & X_{0,0} + X_{1,0} \to C_0 \\ X_{0,0} + X_{1,0} + X_{2,1} \to S_1 & C_0 + X_{2,0} \to S_0 \\ & C_0 + X_{2,1} \to S_1 \end{array} \tag{50}$$

$$\begin{array}{ll} X_{0,0} + X_{1,1} + X_{2,0} \to S_1 & X_{0,0} + X_{1,1} \to C_1 \\ X_{0,0} + X_{1,1} + X_{2,1} \to S_2 & C_1 + X_{2,0} \to S_1 \\ & C_1 + X_{2,1} \to S_2 \end{array} \tag{51}$$

$$\begin{array}{ll} X_{0,1} + X_{1,0} + X_{2,0} \to S_1 & X_{0,1} + X_{1,0} \to C_2 \\ X_{0,1} + X_{1,0} + X_{2,1} \to S_2 & C_2 + X_{2,0} \to S_1 \\ & C_2 + X_{2,1} \to S_2 \end{array} \tag{52}$$

$$\begin{array}{ll} X_{0,1} + X_{1,1} + X_{2,0} \to S_2 & X_{0,1} + X_{1,1} \to C_3 \\ X_{0,1} + X_{1,1} + X_{2,1} \to S_3 & C_3 + X_{2,0} \to S_2 \\ & C_3 + X_{2,1} \to S_3 \end{array} \tag{53}$$

The electing set consists of the trimolecular reactions on the left-hand side. Those on the right-hand side are the bimolecular equivalents. Every pair of trimolecular reactions can be represented by three bimolecular reactions.

Fig. 2 shows the simulation results for a DNA implementation of a CRN computing

$$y(x) = \frac{1}{4} + \frac{9}{8}x - \frac{15}{8}x^2 + \frac{5}{4}x^3 \tag{54}$$

at $x = 0.25$. The Bernstein polynomial for $y(x)$ is

$$g(x) = \frac{2}{8}[(1-x)^3] + \frac{5}{8}[3x(1-x)^2] + \frac{3}{8}[3x^2(1-x)] + \frac{6}{8}x^3. \tag{55}$$

From the Bernstein coefficients, we initialize the types $(B_{i,0}, B_{i,1})$ for $i = 0, 1, 2, 3$ as follows:

$$\left.\begin{array}{l} B_{0,0} = 60 \\ B_{0,1} = 20 \end{array}\right\} \quad \to \quad \frac{20}{60+20} = \frac{2}{8} \tag{56}$$

$$\left.\begin{array}{l} B_{1,0} = 30 \\ B_{1,1} = 50 \end{array}\right\} \quad \to \quad \frac{50}{30+50} = \frac{5}{8} \tag{57}$$

$$\left.\begin{array}{l} B_{2,0} = 50 \\ B_{2,1} = 30 \end{array}\right\} \quad \to \quad \frac{30}{50+30} = \frac{3}{8} \tag{58}$$

$$\left.\begin{array}{l} B_{3,0} = 20 \\ B_{3,1} = 60 \end{array}\right\} \quad \to \quad \frac{60}{20+60} = \frac{6}{8} \tag{59}$$

In order to evaluate $y(x)$ at $x = 0.25$ we initialize $X_{j,0} = 75$ and $X_{j,1} = 25$ for $j = 1, 2, 3$. The initial value for the other types is set to zero.

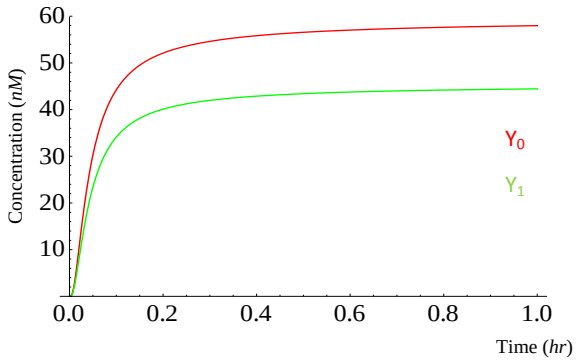

Fig. 2. Simulation results for a DNA implementation of a CRN computing $y(x) = \frac{1}{4} + \frac{9}{8}x - \frac{15}{8}x^2 + \frac{5}{4}x^3$ at $x = 0.25$.

Judging the concentrations of Y_0 and Y_1 to be close to their final values at at $t = 1$ hour, the output y evaluates to:

$$y = \frac{Y_1(1)}{Y_0(1) + Y_1(1)} = \frac{45.03}{57.357 + 45.03} = 0.4398 \tag{60}$$

Table 1 shows the computed values of $y(x)$ at several points and the corresponding errors.

x_{in}	Computed y(x)	Error (%)
0.1	0.3626	5
0.25	0.4398	1.4
0.5	0.5010	0.2
0.75	0.5590	1.3
0.9	0.6356	3

Table 1. Accuracy of a DNA Strand-Displacement Implementation of a CRN Computing $y(x) = \frac{1}{4} + \frac{9}{8}x - \frac{15}{8}x^2 + \frac{5}{4}x^3$ using the proposed method.

6 Conclusion

We introduced a new encoding for computation with CRNs: the value corresponding to each variable consist of the ratio of concentrations of two types. Based on this fractional representation, we proposed a construction for computing arbitrary polynomials that map the unit interval [0,1] on itself. This is a much richer class of functions than the semilinear functions considered in prior work.

We note that the fractional representation that we use here was inspired by our work on stochastic digital circuits [13] [14] [15]. Such circuits operate on random bit streams, with the variables represented as the fraction of 1's versus 0's in the bit stream. We have developed general scheme for synthesizing

circuits that compute arbitrary polynomials based on this encoding. In a sense, this paper is an application of those results to CRNs.

Although the proposed CRNs can compute polynomials, in real applications one might encounter non-polynomial functions, such as trigonometric functions. These cannot be computed exactly, but they can be approximated. In [14], we describe a efficient method for approximating non-polynomial functions through quadratic programming.

Clearly, the primary interest of this work is theoretical. CRNs are a fundamental model of computation, abstract yet conforming to the physical behavior of chemical systems. Delineating the range of behaviors of such systems has intellectual merit. These results may also have practical applications.

Practitioners in synthetic biology are striving to create "embedded controllers" – viruses and bacteria that are engineered to perform useful molecular computation in situ where it is needed, for instance for drug delivery and biochemical sensing. Such embedded controllers may be called upon to perform computation such as filtering or signal processing. Computing polynomial functions is at the core of many these computational tasks.

References

1. F. Horn and R. Jackson, "General Mass Action Kinetics," *Archive for Rational Mechanics and Analysis*, Vol. 47, pp. 81–116, 1972.
2. P. Érdi and J. Tóth, "Mathematical Models of Chemical Reactions: Theory and Applications of Deterministic and Stochastic Models," Manchester University Press, 1989.
3. , D. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *Journal of Physical Chemistry*, Vol. 81, No. 25, 2340–2361. 1977.
4. , S. Strogatz, "Nonlinear Dynamics and Chaos with Applications to Physics, Biology, Chemistry, and Engineering," Perseus Books, 1994.
5. L. Adelman, "Molecular Computation of Solutions to Combinatorial Problems," *Science*, Vol. 266, pp. 1021–1024, 1994.
6. M. O. Magnasco, "Chemical Kinetics is Turing Universal," *Physical Review Letters*, Vol. 78, No. 6, pp. 1190–1193, 1997.
7. D. Soloveichik, M. Cook, E. Winfree, and J. Bruck, "Computation with Winite Stochastic Chemical Reaction Networks," *Natural Computing*, Vol. 7, No. 4, pp. :615–633, 2008.
8. H. Jiang, S.A. Salehi, M.D. Riedel, and K.K. Parhi, "Discrete-Time Signal Processing with DNA," American Chemical Society (ACS) Synthetic Biology, Vol. 2, No. 5, pp. 245–254, 2013.
9. H. Chen, D. Doty, and D. Soloveichik, "Deterministic Function Computation with Chemical Reaction Networks," *DNA Computing and Molecular Programming*, Springer Lecture Notes in Computer Science, Vol. 7433, pp. 24–42, 2012.
10. H. Chen, D. Doty, and D. Soloveichik, "Rate-Independent Computation in Continuous Chemical Reaction Networks," Conference on Innovations in Theoretical Computer Science, pp. 313–326, 2014.
11. R. Farouki and V. Rajan, "On the Numerical Condition of Polynomials in Bernstein Form," *Computer-Aided Geometric Design*, Vol. 4, No. 3, pp. 191–216, 1987.
12. W. Qian, M. D. Riedel, and I. Rosenberg, "Uniform Approximation and Bernstein Polynomials with Coefficients in the Unit Interval," *European Journal of Combinatorics*, Vol. 32, No. 3, pp. 448–463, 2011.

13. W. Qian and M. D. Riedel, "The Synthesis of Robust Polynomial Arithmetic with Stochastic Logic," *Design Automation Conference*, pp. 648–653, 2008.
14. , W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," *IEEE Transactions on Computers*, Vol 60, No. 1, pp. 93–105, 2011.
15. , W. Qian, M. D. Riedel, H. Zhou and J. Bruck, "Transforming Probabilities with Combinational Logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 30, No. 9, 2011.
16. D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a Universal Substrate for Chemical Kinetics," *Proceedings of the National Academy of Sciences*, pp. 5393–5398, 2010.