1 Computing with Deterministic CRNs

—Me

1.1 Overview

In biology and chemistry, we observe one or more species or reactants over time to understand the patterns of their changes. Essentially, they are time signals. These signals remain positive over time, and their changes are governed by a network of reactions, hence the term chemical reaction network. As molecular programmers, our goal is to manipulate the quantities of these species, whether continuous or stochastic, through interactions with other species. There are various interpretations of how reactions influence the quantities of the species of interest. In this section, we focus on mass-action kinetics, which yields a system of ordinary differential equations (ODEs) that describe the behavior of the dynamics. We refer to chemical reaction networks that utilize this kinetics as *deterministic chemical reaction networks*.

It is beneficial to place chemical reaction networks within the broader context of analog computing. Analog computing involves processing actual physical, chemical, or mechanical signals. For instance, you can use a voltage to represent a number and connect a few resistors to perform addition. In a similar manner, the concentration of our chemical species can be used to encode our computing processes. In this context, a chemical reaction network is an analog computational model. It is important to discuss analog computing because many of the signals we aim to design might not be directly implementable by chemical reactions. Often, we need to first design these signals in the form of polynomial ordinary differential equations, also known as general-purpose analog computers. After that, we look into methods to convert these designs into CRNs.

Mass-Action Kinetics: An Informal Discussion

Before we discuss the mass-action kinetics for the *network* of chemical reactions, let's first consider a much simpler target: a single reaction with one reactant and one product that looks like:

$$X \xrightarrow{k} Y$$

This chemical reaction represents X transforming into Y. We want to describe how fast which this occurs. We can imagine that in a *unit* of space, the more X is consumed, the more Y is produced. Therefore, the rate of producing Y is proportional to the concentration of X in that space. We call the constant ratio k of the the proportionality relation the *rate constant* of the reaction. To simplify the problem, deterministic chemical reaction networks (CRNs) assume that the solution is well-mixed, so everywhere in a volume the concentration is the same.

When a reaction involves two more reactants, such as $X + Y \to Z$, we consider the reactants on the left-hand side collectively; this group is called a *complex*.

Now consider the hypothetical $(X + Y)$ complex as a single species in our system. What would be a reasonable "concentration" for the $(X + Y)$ complex? It would be the product of the concentrations of X and Y, denoted as xy. Informally, we can viewed the concentration of x can be interpreted as the probability of x appearing in a location. The reaction implies that when x collides with y, they will transform into z. The concentration of the complex can be interpreted as the probability that x and y appear at the same location. Hence, the multiplication.

For reactions with multiple reactants and products, such as:

$$x + y \xrightarrow{k} z + w$$

we consider the products as being produced independently. The reaction indicates that the concentrations of the reactants decrease and those of the products increase at the reaction rate. Therefore, x and y will see a rate of change of $-kxy$, while z and w will see an increase of $+kxy$.

When there are multiple reactions, since they can all occur simultaneously, their effects are cumulative. We sum the individual rates to obtain the total rate of change. This cumulative process is exactly what mass-action kinetics describes.

Note that in the above informal discussion, we only view the left-hand side of a reaction as complex, but not the right-hand side. In the study of the so-called detailed balanced chemical reaction networks, we view both sides of a reaction as complexes and we study how the complexes impact each other. We then use the stoichiometry of the complexes to get the rates of change for the variables. A detailed introduction to this notion and its connection to the so-called Global Attractor Conjecture can be found in [14].

We use the following example to demonstrate mass-action kinetics and translation between CRNs and ODEs.

Box 1.1: OED/CRN Example

Let $F(t) = \frac{1}{2}e^{\frac{1}{2}(e^{-t}-1)}$, $E(t) = \frac{1}{2}e^{-t}$, and $G(t)$ be a function such that its derivative "cancels" with E and F's derivative and $F(0) = E(0) = \frac{1}{2}$, $G(0) = 0$. We have the ODE system and the corresponding chemical reaction network by the mass-action kinetics.

$$\text{ODE:} \quad \begin{cases} F' = -F \cdot E \\ E' = -E \\ G' = FE + E \end{cases} \qquad \text{CRN/PP:} \quad \begin{cases} F + E \to E + G \\ E \to G. \end{cases}$$

Do the following exercises:

1. Take the derivative of E, F. Introduce G and let $G' = -(E' + F')$. Try to

Box 1.1: OED/CRN Example (cont.)

use polynomial of E, F and G to represent the derivatives.

2. Turn the ODE to a CRN and Turn the CRN into a ODE.

Some useful basic translations: Let x' be the derivative of x. We elaborate how to turn the terms in x into reactions term by term.

1. Constant term. For example, if we have an ODE $x' = 1 - x$. To turn the constant term "1" in x' into CRN. We need to use a null species, denoted as "$\varnothing$" and write
$$\varnothing \to X.$$

2. Negative terms. If the the coefficient is -1, then just write out all the reactants (X must be one of them) on the left-hand side and drop X on the right-hand side. For example, for the $-FE$ term in the ODE $F' = -FE$, we produce the reaction
$$E + F \to E.$$

3. Positive terms. If the the coefficient is positive 1, then we write out all the reactants on the left-hand side and add an extra x on the right-hand side. For example, for the term FE in G' in Box 1.1, we produce
$$E + F \to E + F + G$$

4. The use of rate constant. From discussion of mass-action Kinetics, we see that the coefficient of a term in the resulting ODE comes from two places: the rate constant and stoichiometry, we can call the later the delta part. We can split the coefficient either in the rate constant or in the delta part.

   For example, to write a term $-2x^2y$ in x', one can either do
$$X + X + Y \to Y,$$
   where we put the coefficient all on the delta part; or
$$X + X + Y \xrightarrow{2} X + Y$$
   where we put the coefficient all on the rate constant.

Note that the above term-by-term translation, when we are translating a term in x, we try to leave all other variables untouched. That is, they serve as *catalysts* in the

reactions we produce.

This does not always produce the optimal number of reactions. For example, in Box 1.1, in stead of translating terms one by one, we can pair up the positive and negative EF term in F' and G'. The two terms means that when E and F bump into each other, we want to decrease F and increase G. We then observe the following reactions

$$E + F \to E + G$$

can fulfill both targets. One stone, two birds.

Polynomial ODE characterization

As we can see, the ODE we obtain from a CRN is a polynomial ODE. Is the the converse also true? The answer is negative. The polynomial ODE corresponds to another computational model, the general-purpose analog computer (GPAC) [13].

Definition 1.1: GPAC

$$\begin{cases} \mathbf{x}' = \mathbf{p}(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \qquad t \in \mathbb{R}$$

where $\mathbf{p}$ is a vector of multivariate polynomials, $\mathbf{x}$ is a vector of variables, and $\mathbf{x}_0$ is its initial value.

We view Chemical Reaction Networks (CRNs) as a restricted form of GPACs.

Theorem 1.2: CRN-implementable Functions [12] Theorem 3.1 and 3.2

A function is CRN-implementable if and only if it is GPAC-implementable with further restriction such that the derivative of each component x_i in its variable vector $\mathbf{x} = (x_1, \cdots, x_n)$ is of the form

$$x_i' = p - qx_i, \quad \text{for all } i \tag{1.1}$$

where p and q are polynomials of $(x_1, \cdots, x_n)$ with positive coefficients.

The theorem align with modeling with CRNs. In CRNs, the change of the variables are governed by reactions. The nature of reaction determine that if we want to decrease a species, it must participate in some reaction and the reaction consumes it. Otherwise there is no way we can decrease a variable. So, in the ODE that describes the change of x, the negative term must contain x as a factor. We say an ODE is CRN-implementable or simply, implementable, if it is in the form specified in Theorem 1.1.

Definition 1.3: Population Protocol

In this section, we call the CRNs with all their reactions having the two-in-two-out form Population Protocols.

One can refer to [1] for the detailed polynomial characterization and proof. The additional constraints are:

1. The system must preserve population, making it conservative. This means the sum of the derivatives of all variables equals zero.
2. Positive square terms x^2 do not appear in the expression for x'. This constraint stems from the nature of the reactions we are restricted to use. In a two-in-two-out reaction, the concentration of x cannot increase if two X molecules are already present as reactants.

Dual Railing

For systems that cannot be directly implemented by a Chemical Reaction Network (CRN), we can often employ a technique called dual railing. This method involves encoding a signal in a General Purpose Analog Computer (GPAC) as the difference between two signals in a CRN. To better illustrate this technique, let's consider an example.

Box 1.2: $\sin(t)$ and $\cos(t)$ Example

We cannot implement $\sin(t)$ and $\cos(t)$ directly in a CRN, as they obviously take negative values. However, we can first implement them in a GPAC and then dual rail them. Let $u = \sin(t)$ and $v = \cos(t)$.

$$u' = \cos(t) = v$$
$$v' = -\sin(t) = -u$$

We find that $\sin(t)$ and $\cos(t)$ cannot be implemented independently even in a GPAC. They come in pairs, each helping to implement the other. In more complex CRNs, this behavior is pervasive, typically involving large dependency loops with multiple species.
We dual rail u by u_1 and u_2, and v by v_1 and v_2. We obtain the following ODE:

$$(u_1 - u_2)' = v_1 - v_2$$
$$(v_1 - v_2)' = u_2 - u_1$$

In the first ODE, we gather all the positive terms (that is v_1) and all the negative terms (that is v_2, if we ignore the negative sign), and attempt to let $u_1' = v_1$

Box 1.2: $\sin(t)$ and $\cos(t)$ Example (cont.)

and $u_2' = v_2$, which is a natural choice for u_1' and u_2'. However, the resulting signals u_1 and u_2 would shoot to infinity, which is not a desirable design in many cases. So we subtract a term $(v_1 + v_2)u_1u_2$ in both u_1' and u_2' to suppress the signal. Additionally, note that the u_1u_2 factor ensures the implementability of the system in CRN. Likewise, we design the ODE for v_1 and v_2 similarly, resulting in the following ODE system.

$$\begin{cases} u_1' = v_1 - (v_1 + v_2)u_1u_2 \\ u_2' = v_2 - (v_1 + v_2)u_1u_2 \\ v_1' = u_2 - (u_1 + u_2)v_1v_2 \\ v_2' = u_1 - (u_1 + u_2)v_1v_2 \end{cases}$$

with initial values set to:

$$u_1(0) = 0,\ u_2(0) = 0,\ v_1(0) = 1,\ \text{and } v_2(0) = 0$$

We have "implemented" $\sin(t)$ and $\cos(t)$ by CRN, with the caveat that we need to use two variables to encode each function. See Figure 1.1 for a plot of the results of the above ODE/CRN.

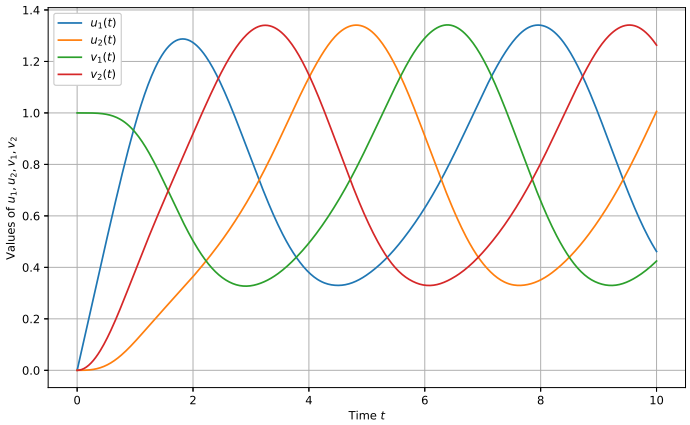

Figure 1.1: $\sin(t)$ and $\cos(t)$ via dual-railing

[*Credit: Xiang Huang*]

Basic functions that are implementable by CRN

We collect a list of useful basic functions here. Typically, you can identify whether a function is CRN-implementable by taking its derivative and rewriting it into the form as in Section 1.1. The resulting CRNs are also provided.

We emphasize the design idea; formal analytic proofs are not provided in most cases. Unless specified otherwise, assume an initial value of zero.

1. $E = e^t$

$$\begin{aligned} &\text{ODE:} \quad E' = E \\ &\text{CRN:} \quad E \to 2E \end{aligned}$$

2. $E = e^{-t}$

$$\begin{aligned} &\text{ODE:} \quad E' = -E \\ &\text{CRN:} \quad E \to \varnothing \end{aligned}$$

3. $U = \frac{t}{1+t}$

$$\begin{aligned} &\text{ODE:} \quad U' = (1-U)^2 \\ &\text{CRN:} \quad \begin{cases} \varnothing \to U \\ U \xrightarrow{2} \varnothing \\ U + U \to 3U \end{cases} \end{aligned}$$

4. $L = \log(1+t)$ Let $V = \frac{1}{1+t}$. Then:

$$\begin{aligned} &\text{ODE:} \quad \begin{aligned} L' &= V \\ V' &= -V^2 \end{aligned} \\ &\text{CRN:} \quad \begin{cases} V \to L + V \\ V + V \to V \end{cases} \end{aligned}$$

Initial conditions: $L(0) = 0$, $V(0) = 1$

The trick is to introduce auxiliary variables as needed and identify combinations of basic functions. These functions and their CRNs serve as building blocks for designing larger chemical reaction networks. We leave the implementation of more complex functions to CRNs as exercises for the reader.

Arithmetic Computations

We aim to construct more complex CRN circuits, analogous to performing arithmetic manipulations with numbers. In this section, we consider CRN signals that have a stabilized limit. Let x and y be two such signals. We'll examine the CRN implementability of functions $x + y$, $x - y$, $\frac{1}{x}$, and xy.

First, we observe that the sum and difference of two CRN-implementable functions are not necessarily implementable. Consider the following examples:

$$x' = 1 - x$$
$$y' = (1 - y)^2$$

Let $z = x + y$. Then $z' = x' + y'$, and we cannot rewrite the negative terms to include a z factor. A similar issue arises for $x - y$.

However, the situation is different for multiplication. It turns out that CRN-implementability is closed under multiplication.

Theorem 1.4: Multiplication

If x, y are CRN-implementable, then so is xy.

Proof:

We write $x' = p_1 - q_1 x$ and $y' = p_2 - q_2 y$, where p_i, q_i's are polynomials with positive coefficients. Let $z = xy$. We have

$$\begin{aligned} z' = x'y + xy' &= (p_1 - q_1 x)y + x(p_2 - q_2 y) \\ &= (p_1 y + p_2 x) - (q_1 + q_2)xy \\ &= (p_1 y + p_2 x) - (q_1 + q_2)xy \\ &= (p_1 y + p_2 x) - (q_1 + q_2)z \end{aligned}$$

So z is CRN implementable.

Note that in the above implementation, z equals to xy at *every point of time*. In comparison, in general we can not have that for addition and subtraction. So, we instead pursuit them *in the limit*.

Box 1.3: Addition

Let x, y be two CRN-implementable function, then the function z with

$$z' = x + y - z$$

Box 1.3: Addition (cont.)

approaches $x + y$ in the limit.
The corresponding CRN for the construction is

$$\begin{aligned} \mathrm{X} &\to \mathrm{X} + \mathrm{Z} \\ \mathrm{Y} &\to \mathrm{Y} + \mathrm{Z} \\ \mathrm{Z} &\to \varnothing \end{aligned}$$

Box 1.4: Subtraction ([2], see also in [3])

Let x, y be two CRN-implementable function, then the function z in the ODE

$$\begin{cases} z' = x - zh - z \\ h' = y - zh \end{cases}$$

approaches $x - y$ in the limit.

To understand how we construct this circuit, we start with the straightforward idea to introduce z such that $z' = (x-y)-z$. That is, we hope z goes to $(x-y)$ in its equilibrium. While this idea works as an ODE, z is not CRN-implementable, since z' contains a negative term of y, without z being a factor. How to fix this? We apply yet another straightforward idea to "add" a z factor to y. We introduce a helper variable h and we hope $y = zh$ or they become very close asymptotically. So we let $h' = y - zh$ and then we replace y with zh in $(x - y) - z$. Hence we finish the construction.
The corresponding CRN for the construction is

$$\begin{aligned} \mathrm{X} &\to \mathrm{X} + \mathrm{Z} \\ \mathrm{Y} &\to \mathrm{Y} + \mathrm{H} \\ \mathrm{Z} + \mathrm{H} &\to \varnothing \\ \mathrm{Z} &\to \varnothing \end{aligned}$$

We leave the implementation of reciprocal ($\frac{1}{x}$), division, and an alternative method for subtraction as exercises for the reader. For more details on other arithmetic operations in CRN, one can refer to [15] and [16].

A combination of the basic functions and the arithmetic operations can greatly facilitate more complex circuit design.

Another useful constructions is function composition. Since what we compose is the time variable, it can be viewed as manipulation of how time flows in a system.

Composition and Integration

When we take the derivative composition of functions, as what we learn in Calculus, we need to use chain rule. We use the following example to demonstrate chain rule and construction of CRN that implements a composite function.

Box 1.5: Composition example

Implement $\widehat{E}(t) = e^{1-e^{-t}}$ in CRN.

We denote $E(t) = e^t$ and $u(t) = 1 - e^t$. Then $\widehat{E}(t) = e^u$ can be viewed this as a the composition of E and u. By chain rule we have

$$\begin{aligned}\widehat{E}' &= e^u \cdot u' \\ &= \widehat{E} \cdot (1 - u).\end{aligned}$$

compare it to the derivative of E, $E' = E$. We found that the ODE for $\widehat{E}$ can be constructed by

1. Change the variable from E to $\widehat{E}$.
2. Multiply it by the extra factor $u' = (1 - u)$ in the ODE, which is resulted from the chain rule.

We leave it as an exercise to write out the CRN for $\widehat{E}$ and u.
Further, if we introduce u's derivative as a new variable. That is, we let $v = u'$. Then $\widehat{E}$'s ODE becomes

$$\widehat{E}' = \widehat{E}'v.$$

The new variable v is CRN-implementable. We compare the ODE an CRN for E and $\widehat{E}$ again.

The ODE for E:	The ODE for $\widehat{E}$:
$E' = E.$	$\widehat{E}' = \widehat{E}v.$
The CRN for E:	The CRN for $\widehat{E}$:
$\varnothing \to E.$	$V \to \widehat{E} + V.$

We find that E and $\widehat{E}$ has the same CRN except that in the reaction(s) for $\widehat{E}$, we need to add V as a catalyst. It is important to know that although v is corresponding to the catalyst, the effect of it, or the resulting time dilation is u, the integral of v.

Box 1.5: Composition example (cont.)

Also note that since E and $\widehat{E}$ basically has the same CRN structures, sometime we abuse the notation to write E for $\widehat{E}$ when the context is clear. But one should keep the distinction of the two in mind.

A general procedure for composing a CRN implementable function v to another CRN implementable function f can be demonstrated in Figure 1.2. A formal discussion can be found in [10] (page 15, Theorem 3.7).

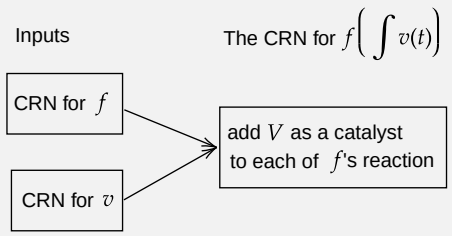

Figure 1.2: Compose two CRN-implementable functions
[*Credit: Xiang Huang*]

Another useful basic operations for CRN circuit construction is integration.

Box 1.6: Integral

To take the the indefinite integral of a species $x(t)$. One just need to introduce a new variable y and make

$$y' = x$$

in terms of CRN, that is

$$X \to X + Y.$$

Then y represents the integral of x.

Normalization and Majority Algorithm

Equipped with the basic operations and implementations we've discussed, we can now explore an important circuits in molecular programming design: the approximate majority algorithm.

To compare two signals x and y, we first normalize them first to the interval [0,1].

Box 1.7: Normalization [3]

Given two CRN signals x and y, normalize them into two values u and v such that

$$u = \frac{x}{x+y}, \quad v = \frac{y}{x+y}$$

by a **population protocol**.

Construction: Consider the constraint

$$\begin{cases} u + v = 1 \\ \frac{x}{y} = \frac{u}{v} \end{cases}$$

We then have $xv = yu$. We create an ODE to ensure this constraint is satisfied in the equilibrium state

$$\begin{cases} u' = xv - uy \\ v' = uy - xv \end{cases}$$

Note that the above ODE also satisfies the algebraic constraint for polynomial systems of population protocols. To enforce $u + v = 1$, we can initiate the system at $(u, v) = (0.5, 0.5)$. Written in reaction form, the above system looks like

$$\begin{cases} X + V \to X + U, \\ Y + U \to Y + V. \end{cases}$$

We then demonstrate the design for the approximate majority algorithm, which can be used to compare two signals. In fact, in the continuous world, we can drop the term "approximate" since it deterministically "decides" which one of x or y is larger.

Box 1.8: Majority [5]

Design a **population protocol** so that the greater of the initial values of two variables x and y takes all the mass. That is: if $x(0) > y(0)$, let $x(t)$ go to $x(0) + y(0)$. Otherwise, let $y(t)$ go to $x(0) + y(0)$.

Construction:
Attempt 1: We try the straightforward idea of setting $x' = x - y$, hence if $x - y > 0$, then x increases. Similarly, $y' = y - x$, so $x' + y' = 0$. We get a system somewhat

Box 1.8: Majority [5] (cont.)

close to a PP:

$$\begin{cases} x' = x - y \\ y' = y - x \end{cases}$$

But this system is obviously not CRN-implementable.
Attempt 2: We multiply x to the above expression for x' and similarly y for y'. The resulting system is a CRN. But it fails to be conservative, resulting in x shooting to infinity if $x > y$.
Attempt 3: We multiply xy instead to both x' and y' to guarantee conservation. That results in the termolecular system in [6], though not exactly what we want.
Attempt 4: We introduce a new variable, b, the "blank state", such that we hope b stabilizes between x and y, that is $x < b < y$ or $y < b < x$. Note that in our first and second attempts, what matters is the *sign* of $x - y$, to guarantee correctly deciding which variable to increase or decrease. We observe that if b is between x and y, then $x - y$ and $b - y$ have the same sign, likewise $y - x$ and $y - b$ have the same sign. We substitute $b - y$ for $x - y$ and $y - b$ for $y - x$ in the result of attempt 2, resulting the following ODE

$$\begin{cases} x' = (b - y)x \\ y' = (b - x)y \\ b' = -(x' + y') = 2xy - b(x + y). \end{cases}$$

in which in order to make it a population protocol, we set $b' = -(x' + y')$ to enforce conservation. The variable b tends to $\frac{2xy}{x+y}$, or $\frac{2}{\frac{1}{x}+\frac{1}{y}}$, which is the *harmonic mean* and falls between x and y asymptotically, testifying to the designed property. Another good property of the harmonic mean is that, as one of x or y goes to zero, b also goes to zero. So b as a helper variable can be set to zero initially and it will die out eventually.
We result in the system presented in [5], detailed below

$$\begin{cases} X + Y \to X + B \\ Y + X \to Y + B \\ X + B \to X + X \\ Y + B \to Y + Y. \end{cases}$$

The above discussion is summarized in Figure 1.3.

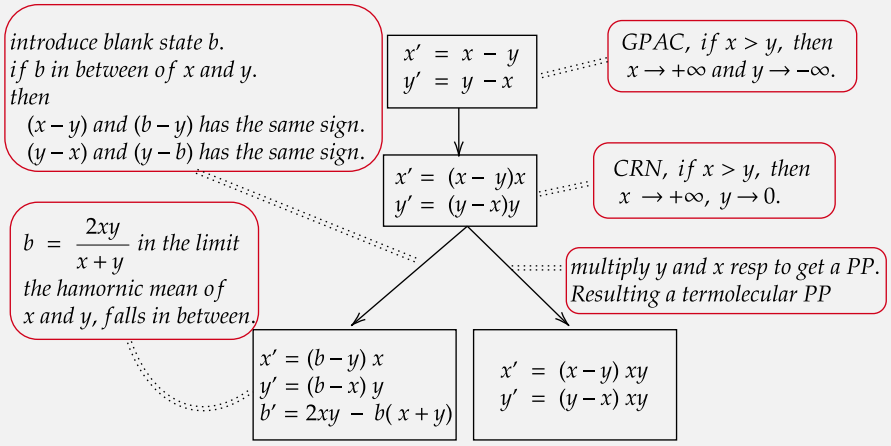

Figure 1.3: Designing a majority algorithm

[*Credit: Xiang Huang*]

We plot the system with initial value $(x, y, b) = (0.6, 0.4, 0)$ in Figure 1.4. We can see how b traces the harmonic mean of x and y.

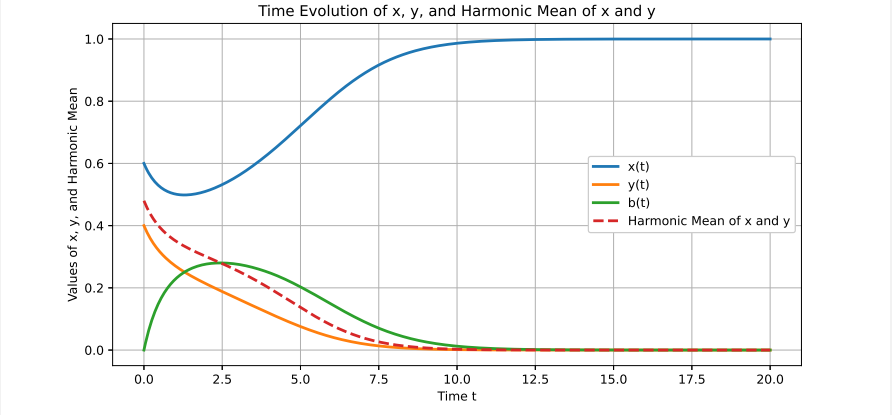

Figure 1.4: Majority Algorithm with (x=0.6, y=0.4)

[*Credit: Xiang Huang*]

1.1[quantitative] Are the variables in the following ODE CRN-implementable?

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x)\\ \frac{dy}{dt} &= x(\rho-z)-y\\ \frac{dz}{dt} &= xy-\beta z\end{aligned}$$

where x, y, and z are the state variables, and t represents time. The parameters σ, ρ, and β are positive constants.

1.2[quantitative] Are the variables in the following ODE CRN-implementable?

$$\begin{cases} x' = \mu x - x^3\\ y' = -y\end{cases}$$

where μ is some parameter.

1.3[advanced] Is the following system implementable by a Population Protocol?

$$\begin{cases} x_0' = x_0^2 + 7x_0x_1 - x_1^2,\\ x_1' = -x_0^2 + 7x_0x_1 + x_1^2.\end{cases}$$

1.4[quantitative] Implement the function $\tan(t)$ in CRN.

1.5[advanced] Implement the function $e^{-t}\log(1+t)$ in CRN.

1.6[advanced] Implement the function te^{-t} in CRN.

1.7[quantitative] Given a CRN signal x, construct a circuit to compute $1/x$.

1.8[quantitative] Given two CRN signals x and y, construct a circuit to compute y/x. (Let's assume $x(0) > 0$. So $x(t)$ never hits 0.)

1.9[quantitative] Let x and y be two CRN signals that converge. What does z in the following circuit

$$z' = 1 - (x-y)z$$

approach to? Base on this, can you design a CRN circuit for subtraction? What other assumption do we need to make about x and y's limit? See [8] for a solution and analysis. [Hint: Use the reciprocal algorithm.]

1.10[quantitative] Write our the CRN for the ODE in Box 1.5.

$$\begin{cases} \widehat{E}' = \widehat{E} \cdot (1-u), \\ u' = 1-u. \end{cases}$$

1.11[quantitative] Recall that we listed $f(t) = \frac{t}{1+t}$ as a basic function. Now we want to implement the sigmoid function

$$\sigma(t) = \frac{e^t}{1+e^t}$$

by CRN via composing $f(t)$ with e^t. Write out the ODE and the CRN for the purpose and compare their forms with the ones for $f(t)$.

1.12[quantitative] Design a GPAC for a variable x such that $x' = \cos(x) - x$. Note that the current ODE is not a GPAC since the term $cos(x)$ is not a polynomial of x. The variable x goes to a number known as the Dottie number in the limit. It is the global attractor of the ODE. So you can pick any initial value you want when you plot the system.
[Hints: You can refer to the implementation of $\sin(x)$ and $\cos(x)$ in Section Section 1.1. You can check [7] for a solution].

1.13[advanced] Implement the GPAC in Problem 1.12 in CRN using dual-railing.

1.2 Bibliography

[1] (huang-huls; doi:10.4230/LIPIcs.DNA.28.7) Huang, Xiang, and Huls, Rachel N. "Computing Real Numbers with Large-Population Protocols Having a Continuum of Equilibria." In 28th International Conference on DNA Computing and Molecular Programming (DNA 28). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[2] (buisman-algebraic; doi:10.1162/artl.2009.15.1.15101) Buisman, H. J., ten Eikelder, H. M., Hilbers, P. A., and Liekens, A. M. (2009). Computing algebraic functions with biochemical reaction networks. Artificial life, 15(1), 5-19.

[3] (Vasić-crn++; doi:10.1007/s11047-019-09775-1) Vasić, M., Soloveichik, D., and Khurshid, S. (2020). CRN++: Molecular programming language. Natural Computing, 19(2), 391-407.

[4] (Huang-GPAC; doi:10.1145/3637543.3654758) Huang, Xiang, and Andrei Migunov. "A General Purpose Analog Computer to Population Protocol Compiler." In Proceedings of the 21st ACM International Conference on Computing Frontiers Workshops and Special Sessions, pp. 1-4. 2024.

[5] (Angluin-majority; doi:10.1007/s00446-008-0059-z) Angluin, Dana, James Aspnes, and David Eisenstat. "A simple population protocol for fast robust approximate majority." Distributed Computing 21, no. 2 (2008): 87-102.

[6] (Klinge-majority; doi:10.1145/2967446.2967465) Klinge, Titus H. "Robust signal restoration in chemical reaction networks." In Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication, pp. 1-6. 2016.

[7] (Huang-thesis; doi:10.31274/etd-20210114-63) Huang, Xiang. "Chemical reaction networks: Computability, complexity, and randomness." PhD diss., Iowa State University, 2020.

[8] (Huang-CRN-GPAC; doi:10.1007/978-3-030-26807-7_3) Huang, Xiang, Titus H. Klinge, and James I. Lathrop. "Real-time equivalence of chemical reaction networks and analog computers." In International Conference on DNA Computing and Molecular Programming, pp. 37-53. Cham: Springer International Publishing, 2019.

[9] (Huang-comp; doi:10.1145/3637543.3654758) Huang, Xiang, and Andrei Migunov. "A General Purpose Analog Computer to Population Protocol Compiler." In Proceedings of the 21st ACM International Conference on Computing Frontiers Workshops and Special Sessions, pp. 1-4. 2024.

[10] (Klinge-composition; doi:10.31274/etd-180810-5369) Titus H. Klinge. Modular and Robust Computation with Deterministic Chemical Reaction Networks. PhD thesis, Iowa State University, 2016.

[11] (Huang-bounded; doi:10.1007/s11047-018-9706-x) Huang, Xiang, Titus H. Klinge, James I. Lathrop, Xiaoyuan Li, and Jack H. Lutz. "Real-time computability of real numbers by chemical reaction networks." Natural Computing 18, no. 1 (2019): 63-73.

[12] (Hars-Toth; doi:) Vera Hárs and János Tóth. On the inverse problem of reaction kinetics. In Colloquia Mathematica Societatis János Bolyai, 30: Qualitative Theory of Differential Equations, pages 363–379, 1981.

[13] (Silva-GPAC; doi:10.1002/malq.200310113) Silva Graça, Daniel. "Some recent developments on Shannon's general purpose analog computer." Mathematical Logic Quarterly: Mathematical Logic Quarterly 50, no. 4-5 (2004): 473-485.

[14] (Johnston-IntroGAC; doi:[No DOI available]) Johnston, Matthew D. (2016). Introduction to the Global Attractor Conjecture. Retrieved August 30, 2024, from https://johnstonmd.wordpress.com/wp-content/uploads/2016/02/introgac.pdf

[15] (alge-funcs; doi:10.1162/artl.2009.15.1.15101) Buisman H. J., Huub MM ten Eikelder, Peter AJ Hilbers, and Anthony ML Liekens. "Computing algebraic functions with biochemical reaction networks." Artificial life 15, no. 1 (2009): 5-19.

[16] (arith-comp; doi:10.48550/arXiv.2404.04396) Anderson, David F., and Badal Joshi. "Chemical mass-action systems as analog computers: implementing arithmetic computations at specified speed." arXiv preprint arXiv:2404.04396 (2024).

2 My Amazing Textbook Section

—Anne Author

2.1 A section

A subsection

A **keyword**. *Emphasis*. Inline math: $3 + 4 = 7$.

Pop Quiz 2.1: Check your understanding!

What is $3 + 4$?

Solution on page 22

- Don't adjust spacing, figure placement, or other typographic tweaks. These will change significantly in the final version.
- Use "quotes" 'properly'. I.e. ``quotes'' `properly'.
- 'Escape' spaces after abbreviations such as e.g. and i.e.: `e.g.\ and i.e.\` . This stops LaTeX thinking the sentence has finished.
- Keep your LaTeX simple, preferring the macros used in this tutorial.
- If you want to use another macro or environment, please contact your editor in the first instance who will coordinate with the infrastructure lead[1].

[1] Hannah Earley

- Use `\textrm` or `\mathrm` to put text/words in math. For example, k_{cat} is typeset as `$k_{\textrm{cat}}$`.
- Prefer 'semantic' macros, e.g. `\Pr`: $\Pr(\text{raining}|\text{UK}) = 1$. This is so we can easily change the way these are typeset later on.

1. A
    a) numbered
2. list

Molecular programming

Molecular programming arguably started when Author et al. [1] solved the Hamiltonian path problem. Later, approaches such as the **Tile Assembly Model** [2] were developed. See Definition 2.1 to find out what AoMP is.

Use `\Cref{label}` to reference something (it will automatically put in what object it is, e.g. Definition 1.1 above). You can separate multiple labels in with commas, e.g. Pop Quizzes 2.1 and 2.2.

Definition 2.1: The Art of Molecular Programming

The principles of molecular programming are currently scattered across thousands of papers, which presents a barrier for new researchers entering the field. The Art of Molecular Programming is a grassroots community initiative to collect these principles in one location, providing tutorial lessons to guide students' learning, and presenting a collective vision on where the field is heading.

Your labels should be 'namespaced': for the tutorial, the namespace is `tut`, so we put `:tut:` in the label somewhere. For example, Definition 2.1 has label `dfn:tut:aomp`. You should also prefix your labels appropriately. `eqn` for equations, `dfn` for definitions, `fig` for figures, `tbl` for tables, `pop` for pop quizzes, `prob` for problems, etc. For your section, you should be assigned a namespace. The point of this is to avoid conflicts between labels in different sections.

2.2 My second section

Pop Quiz 2.2: Check your understanding!

Is the Riemann hypothesis true?

Prefer the `align` and `align*` environments for large math blocks. Use the starred version unless you are going to label and reference an equation. The quadratic formula

is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Euler's identity is

$$1 + e^{i\pi} = 0. \tag{2.1}$$

Recall Euler's identity, Equation (2.1). Consider the functional equation

$$W = 1 + pxtW + q\bar{x}t(W - W|_{x=0}); \tag{$*$}$$

Equation ($*$) represents the generating function for a biased random 1D walk.

2.3 Boxes

Theorem 2.2: Fermat's Last Theorem

There are no positive integers satisfying $a^n + b^n = c^n$ for $n > 2$.

Proof:

The proof is too large to fit in this margin.

Box 2.1: Euclid's Algorithm

```
a, b ← the inputs
while a ≠ b do
    if a > b then
        a ← a − b                    if g|a and g|b, then g|a − b
    else
        b ← b − a
    end if
end while
return a
```

This is Euclid's original algorithm to compute the **Greatest Common Divisor** of two numbers, a and b. It requires only subtraction. GCD can be computed more efficiently using integer division with remainder.

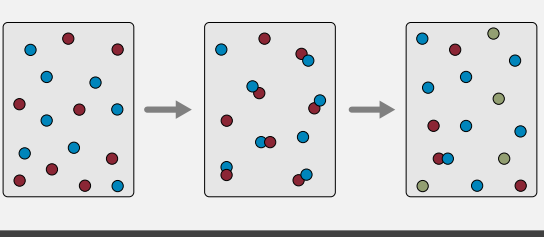

Figure 2.1: An example CRN
[*Credit: Anastasia Ershova*]

p	q	$\overline{pq}$
0	0	1
0	1	1
1	0	1
1	1	0

Table 2.1: Truth table for `NAND`

2.4 Packages

You may find the documentation for these packages helpful:

`physics` for formatting differentials and derivatives.

Examples:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = 3x + 4,$$
$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} = 3,$$
$$y = \int^x (3x' + 4)\,\mathrm{d}x',$$
$$\frac{\partial^3 f}{\partial x^3} = g(x, y, z).$$

`siunitx` for formatting numbers and quantities with units.

Examples: $\hbar = 1.054\,572 \times 10^{-34}\,\mathrm{J\,s^{-1}}$, water has triple point 273.16 K and 611.657 Pa. Avogadro's constant is 6.022×10^{23}. At sea level, water is liquid from 0 °C to 100 °C.

`mhchem` for formatting chemical species and equations.

Examples: H_2O, Al_2O_3, X, X_i, $A + B \to C$, $C + D \underset{k_2}{\overset{k_1}{\rightleftharpoons}} E + F \leftarrow G$.

$$\sum_i \alpha_i X_i \to \sum_i \beta_i Y_i$$

$$\text{Water} + \text{Carbon Dioxide} \rightleftharpoons \text{Glucose} + \text{Oxygen}$$

`algorithmicx` for formatting algorithmic pseudocode. See Box 2.1 for an example.

2.5 Problems

2.1[quantitative] Plot $\sin x$.

2.2[advanced] Write a program to compute a billion digits of π.

2.6 Bibliography

[1] (adleman-hampath; doi:10.1126/science.7973651) Adleman, Leonard M. "Molecular computation of solutions to combinatorial problems." Science 266.5187 (1994): 1021-1024.

[2] (winfree-tam; doi:10.1038/28998) Winfree, Erik, et al. "Design and self-assembly of two-dimensional DNA crystals." Nature 394.6693 (1998): 539-544.

2.7 Further Reading

[3] *(adleman2+; doi:10.1126/science.7973651) Adleman, Leonard M. "Molecular computation of solutions to combinatorial problems." Science 266.5187 (1994): 1021-1024.*

Read this to find out more about the origins of dna computing.

[4] *(winfree2+; doi:10.1038/28998) Winfree, Erik, et al. "Design and self-assembly of two-dimensional DNA crystals." Nature 394.6693 (1998): 539-544.*

Read this to find out more about tile assembly.

2.8 Solutions

Pop Quiz 2.1: Solution(s)

7.

Pop Quiz 2.1 on page 18