Outline

• The Scientific Method
  Its eventual automation

• Lab Protocols (that know nothing about models)
  Digital Microfluidics

• Models (that know nothing about protocols)
  Chemical Reaction Networks

• Integration
  Closed-loop modeling and protocol execution
The Scientific Method

Ḥasan Ibn al-Haytham (1027) Book of Optics

Galileo Galilei (1638) Two New Sciences
Discovery through Observation

The Scientific Method ~ 1638

Discovery through Collaboration

The Scientific Method ~ 2000’s

1 protein = 30 people / 30 years
Humans have >250,000 proteins ☺

Discovery through Automation

The Scientific Method ~ 2020's

while (true) {
    predict();
    falsify();
}

Robot scientist becomes first machine to discover new scientific knowledge

The Inner Loop

• A model is refined by testing a protocol against a systems
• A protocol is refined by testing a model against a systems

• Today: publication does not accurately reflect execution

  • Model: poorly-maintained matlab script
  • Protocol: poorly-described manual steps in the lab
  • System: poorly-characterized and hardly “resettable”

  • ⇒ Crisis in biology: experiments are done once and are hard to reproduce
    [http://www.nature.com/news/reproducibility-1.17552]
The Inner Loop

- Tomorrow, automation
  - Model: unambiguous (mathematical) description (CompBio)
  - Protocol: standardized (engineered) parts and procedures (SynthBio)
  - System: characterized (biological) organism and foundries (SysBio)

- Verification: simulation / analysis / model checking / theorem proving
- Observation: lab automation
- Falsification: statistical inference / model reduction

- Performance evaluation/optimization: of model+protocol+system combined
- Management: version control, equipment monitoring, data storage
The Inner Loop

• A specific domain

• Aiming for closed-loop automated modelling and experimentation

• Via Molecular Programming
Protocols

(lab procedures that know nothing about models)
A Protocol
For DNA gate assembly and activation in vitro

Protocol steps (liquid handling)
Digital Microfluidics
Manipulating droplets by electrical fields

OpenDrop
https://www.youtube.com/watch?v=ncfZWqPm7-4

OpenDrop speed test
https://www.youtube.com/watch?v=pSIs9L_h3Q0

Purple Drop (UW)

```
# Sample program in Python using Puddle.

a = input(substance_A)
b = input(substance_B)
ab = mix(a, b)

while get_pH(ab) > 7:
    heat(ab)
    acidify(ab)
```
Digital Microfluidics

- A general, *programmable*, platform to execute the main liquid-handling operations

- To close the cycle, it can support many automated observation techniques on-board or off-board via peripheral pumps (sequencing, mass spec, ...) although these are all very hardware-dependent.
A Protocol Language

**Samples**: containers with volume, temperature, concentrations

\[ P = \]

\[ x \quad (\text{a sample variable}) \]

\[ (x_0, V, T) \quad (\text{initial condition}) \]

\[ \text{let } x = P_1 \text{ in } P_2 \quad (\text{define local variable}) \]

\[ \text{Mix}(P_1, P_2) \quad (\text{mix samples}) \]

\[ \text{let } x, y = \text{Split}(P_1, p) \text{ in } P_2 \quad (\text{split samples}) \]

\[ \text{Equilibrate}(P, t) \quad (\text{equilibrate sample for } t \text{ seconds}) \]

\[ \text{Dispose}(P) \quad (\text{discard sample}) \]
Protocol Semantics (deterministic)

Each program denotes a final state \(<\text{concentrations, volume, temperature}>\)

\([P]^{\rho}\) is the final state produced by a protocol \(P\) where \(\rho\) binds its free variables:

\[
\begin{align*}
[x]^{\rho} &= \rho(x) \\
[x_0, V, T]^{\rho} &= (x_0, V, T) \\
[Mix(P_1, P_2)]^{\rho} &= \\
&= \text{let } (x_1^1, V_1, T_1) = [P_1]^{\rho} \\
&\quad \text{let } (x_2^2, V_2, T_2) = [P_2]^{\rho} \\
&\quad \left( \frac{x_0^1 V_1 + x_0^2 V_2}{V_1 + V_2}, V_1 + V_2, \frac{T_1 V_1 + T_2 V_2}{V_1 + V_2} \right) \\
[\text{let } x = P_1 \text{ in } P_2]^{\rho} &= \\
&= \text{let } (x_0, V, T) = [P_1]^{\rho} \\
&\quad \text{let } \rho_1 = \rho\{x \leftarrow (x_0, V \cdot p, T), y \leftarrow (x_0, V \cdot (1 - p), T)\} \\
&\quad [P_2]^{\rho_1}
\end{align*}
\]

(Equilibrate semantics)

\([\text{Dispose}(P)]^{\rho} = (0^{|A|}, 0, 0)\)

(CRN semantics)
Summarizing

- Our protocols are (liquid handling) programs
- We can compute their behavior (their final state)
- We can (virtually) run them (by simulation)
- We can (physically) run them (e.g., by digital microfluidics)
Models

(equations that know nothing about protocols)

We could choose Differential Equations as our modeling language, as in most of science.

Instead, we choose Chemical Reaction Networks (this is roughly equivalent).

Anyway, in order to “implement differential equations” we need to “implement chemical reactions” (or some other physical realization).
Chemical Reaction Networks (CRN)

$$X + Y \rightarrow^r Z + W$$

- A *phenomenological model* of kinetics in the natural sciences
  By (only) observing naturally occurring reactions
- A *programming language*, *finitely* encoded in the genome
  By which living things manage the *unbounded* processing of matter and information
- A *mathematical structure*, rediscovered in many forms
  Vector Addition Systems, Petri Nets, Bounded Context-Free Languages, Population Protocols, ...

- A description of *mechanism* (“instructions” / “interactions”) rather than *behavior* (“equations” / “approximations”)
  Although the two are related in precise ways
  Enabling, e.g., the study of the evolution of *mechanism* through unchanging *behavior*
E.g., a CRN model of DNA interactions

- Strand displacement reaction between DNA strands

• (It says nothing about the protocols we just saw)
A dynamical systems is anything characterized by a system of differential equations (ODEs).

Elementary dynamical systems are those that include on the r.h.s. only polynomials, trigonometry, exponentials, fractions, and their inverses. (All of biochemistry, all of electronics, most of physics, deterministic chaos, etc.)

**STEP 1, Polynomization:** Elementary ODEs can be exactly reduced to just polynomial ODEs.

Equation of motion of the simple pendulum

\[ \ddot{\theta} = -\frac{g}{l} \sin \theta \]

https://en.wikipedia.org/wiki/Pendulum
Programming any dynamical system as a CRN

Consider the canonical polynomial oscillator: sine/cosine

\[ \begin{align*}
\partial s &= c \\
\partial c &= -s
\end{align*} \]

let \( s = (s^+ - s^-) \)

let \( c = (c^+ - c^-) \)

\[ \begin{align*}
\partial (s^+ - s^-) &= (c^+ - c^-) \\
\partial (c^+ - c^-) &= -(s^+ - s^-)
\end{align*} \]

A very simple elementary ODE system.

But variables go negative: we can’t have that in a CRN (no negative concentrations).

**STEP 2, Positivation:** Split potentially negative variables of polynomial ODEs into the difference of two positive variables. Obtain the same trajectories as differences.
Programming any dynamical system as a CRN

Translate positive ODEs to chemical reactions

The Law of Mass Action tells us how to produce polynomial ODEs from CRNs. The inverse process is called Hungarization, it works for Hungarian ODEs (polynomial ODEs where each negative monomial has the l.h.s. differentiated variable as a factor).

**STEP 3, Hungarization:** Translate polynomial ODEs to chemical reaction networks: each monomial on the r.h.s. produces one reaction.

Subject to the ODEs being Hungarian, but that is always satisfied after positivation!

E.g. the Lorenz chaotic attractor is already polynomial but not Hungarian, it cannot be translated to mass action reactions without first doing positivation.
Programming any dynamical system as a CRN

Translate those CNRs to (real, DNA) molecules

Chemistry tells us (sometimes) what reactions molecules obey. The inverse process is possible for DNA molecules, because we can “program” them.

**STEP 4, Molecular programming:** Translate any mass action chemical reaction network into a set of DNA molecules that obey those reactions.

Works up to an arbitrarily good approximation of Mass Action kinetics, and up to time rescaling.
Programming any dynamical system as a CRN

Thus, CNRs are “Shannon complete”, and can be physically realized.
Programming any dynamical system as a CRN

- Chemistry is (also) a formal language that we can use to implement ~any dynamical system with real (DNA) molecules.

- Approaching a situation where we can "systematically compile" (synthesize) a model to DNA molecules, run an (automated) protocol, and observe (sequence) the results in a closed loop.

- N.B.: DNA can be used to manipulate and organize programmatically other forms of matter, so this is not really restricted to DNA experiments.
Model Semantics (deterministic)

- ODE semantics of CRNs

**Definition** (CRN Flux) Let \((\mathcal{A}, \mathcal{R})\) be a CRN. Let \(F(V, T) \in \mathbb{R}^{\left|\mathcal{A}\right|} \rightarrow \mathbb{R}^{\left|\mathcal{A}\right|}\) be the flux of the CRN at volume \(V \in \mathbb{R}_{\geq 0}\) and temperature \(T \in \mathbb{R}_{\geq 0}\). For a concentration vector \(\mu \in \mathbb{R}_{\geq 0}^{\left|\mathcal{A}\right|}\) we assume \(F(V, T)(\mu) = \sum_{r \in \mathcal{R}} v_r \alpha_r(V, T, \mu)\) with stoichiometric vector \(v_r\) and rate function \(\alpha_r\).

**Law of Mass Action** \(F(V, T)\) makes up the r.h.s. of an ODE system \(\partial \mathcal{A} = F(V, T)\).

**State produced by a CRN** \(\mathcal{C} = (\mathcal{A}, \mathcal{R})\) (species \(\mathcal{A}\), reactions \(\mathcal{R}\)) with flux \(F\) (r.h.s. of its mass action ODEs) at time \(t\), from initial state \((x_0, V, T)\) (initial concentrations \(x_0\), volume \(V\), temperature \(T\)):

\[
[(\mathcal{A}, \mathcal{R}, x_0), V, T)](H)(t) = (G(t), V, T)
\]

let \(G : [0...H] \rightarrow \mathbb{R}^{\left|\mathcal{A}\right|}\) be the solution of \(G(t') = x_0 + \int_0^{t'} F(V, T)(G(s)) ds\).
Summarizing

- Our models are (chemical) programs
- We can compute their behavior (their final state)
- We can (virtually) run them by integration of the ODEs
- We can (physically) run them by DNA nanotech

- Recall: we are aiming for models that can be placed into a closed-loop automated model+protocol cycle.
Models together with Protocols
An Integrated Description

**Samples**: containers with volume, temperature, concentrations

\[ P = \begin{cases} 
  x & \text{ (a sample variable) } \\
  (x_0, V, T) & \text{ (initial condition) } \\
  \text{let } x = P_1 \text{ in } P_2 & \text{ (define local variable) } \\
  \text{Mix}(P_1, P_2) & \text{ (mix samples) } \\
  \text{let } x, y = \text{Split}(P_1, p) \text{ in } P_2 & \text{ (split samples) } \\
  \text{Equilibrate}(P; t) & \text{ (equilibrate sample for } t \text{ seconds) } \\
  \text{Dispose}(P) & \text{ (discard sample) } 
\end{cases} \]

Each sample evolves (via Equilibrate) according to a given overall CRN:

\[ C = (\mathcal{A}, \mathcal{R}) \quad \text{(species, reactions)} \]
Program Semantics (deterministic)

Each program denotes a final state <concentrations, volume, temperature>

\[[P]^\rho\] is the final state produced by a protocol \(P\) for a fixed CRN \(\mathcal{C} = (\mathcal{A}, \mathcal{R})\):

\[
\begin{align*}
[x]^\rho &= \rho(x) \\
[x_0, V, T]^\rho &= (x_0, V, T) \\
[Mix(P_1, P_2)]^\rho &= \\
&\text{let } (x_0^1, V_1, T_1) = [P_1]^\rho \\
&\text{let } (x_0^2, V_2, T_2) = [P_2]^\rho \\
&\quad \left(\frac{x_0^1 V_1 + x_0^2 V_2}{V_1 + V_2}, V_1 + V_2, \frac{T_1 V_1 + T_2 V_2}{V_1 + V_2}\right) \\
[let \: x = P_1 \: in \: P_2]^\rho &= \\
&\text{let } (x_0, V, T) = [P_1]^\rho \\
&\text{let } \rho_1 = \rho\{x \leftarrow (x_0, V, T)\} \\
&[P_2]^\rho_1
\end{align*}
\]

\[
\begin{align*}
[let \: x, y = Split(P_1, p) \: in \: P_2]^\rho &= \\
&\text{let } (x_0, V, T) = [P_1]^\rho \\
&\text{let } \rho_1 = \rho\{x \leftarrow (x_0, V \cdot p, T), y \leftarrow (x_0, V \cdot (1 - p), T)\} \\
&[P_2]^\rho_1
\end{align*}
\]

\[
\begin{align*}
[Equilibrate(P, t)]^\rho &= \\
&\text{let } (x_0, V, T) = [P]^\rho \\
&\text{let } \rho_1 = \rho\{(A, R, x_0), V, T\} \cdot (H)(t) \\
&[\text{Dispose}(P)]^\rho = (0^{\vert\mathcal{A}\vert}, 0, 0),
\end{align*}
\]

State produced by CRN \(\mathcal{C} = (\mathcal{A}, \mathcal{R})\) with flux \(F\) at time \(t\):

\[
[\text{((A, R, x_0), V, T))} \cdot (H)(t) = \\
\text{let } G : [0...H) \rightarrow \mathbb{R}^{\vert\mathcal{A}\vert} \text{ be the solution of } G(t') = x_0 + \int_0^{t'} F(V, T)(G(s))ds \\
(G(t), V, T)
\]
A Joint Semantics

This semantics gives us a joint simulation algorithm, connecting chemical simulation with protocol simulation.

In this presentation everything is deterministic. The state of the protocol is passed to the chemical simulator, which computes a new state that it passes to the protocol simulator, and so on.

We can also define a joint stochastic simulation, passing mean and variance information back and forth between chemical and protocol simulation. This requires an extension of the above semantics using the Linear Noise Approximation of chemical kinetics, which computes mean and variance of concentrations (both by ODEs, not e.g. by Gillespie algorithm), and a similar extension of the protocol operations.
A Language for Modeling and Optimizing Experimental Biological Protocols

Luca Cardelli *, Marta Kwiatkowska and Luca Laurenti †

Definition 3. (CRN Flux) Let \((A, R)\) be a CRN. Let \(F(V, T) \in \mathbb{R}_{\geq 0}^{A \rightarrow A}\) be the flux of the CRN at volume \(V \in \mathbb{R}_{>0}\) and temperature \(T \in \mathbb{R}_{>0}\). For a concentration vector \(\mu \in \mathbb{R}_{\geq 0}^{A}\) we assume \(F(V, T)(\mu) = \sum_{r \in R} v_r \alpha_r(V, T, \mu)\), with stoichiometric vector \(v_r\) and rate function \(\alpha_r\). We call \(F\) the Jacobian of \(F(V, T)\), and \(J^T\) its transpose. Further, define \(W(V, T)(\mu) = \sum_{r \in R} v_r V_t \alpha_r(V, T, \mu)\) to be the diffusion term.

Definition 4. (CRN Time Evolution) Given a CRN \((A, R, (\mu, \Sigma, V, T))\), its evolution at time \(t < H\) (where \(H \in \mathbb{R}_{\geq 0} \cup \{\infty\}\) is a time horizon) is the state \((\mu_t(t), \Sigma_{\mu, \Sigma}(t), V, T)\) obtained by integrating its flux up to time \(t\), where:

\[
\mu_t(t) = \mu + \int_0^t F(V, T)(\mu(s))ds
\]

\[
\Sigma_{\mu, \Sigma}(t) = \Sigma + \int_0^t J F(\mu(s))\Sigma_{\mu, \Sigma}(s) + \Sigma_{\mu, \Sigma}(s) J^T F(\mu(s)) + W(V, T)(\mu(s))ds,
\]

with \(\mu_t(0) = \mu\) and \(\Sigma_{\mu, \Sigma}(0) = \Sigma\). If, for such an \(H\), \(\mu\) or \(\Sigma\) are not unique, then we say that the evolution is ill-posed. Otherwise, \(\mu_t(t)\) and \(\Sigma_{\mu, \Sigma}(t)\) define a Gaussian process with that mean and covariance matrix for \(t < H\).

\[
[x]^p = \rho(x)
\]

\[
[(P_1 \ldots P_{|A|}, r_V, r_T)]^p = ([P_1]^p \ldots [P_{|A|}]^p, 0^{|A| \times |A|}, r_V, r_T)
\]

\[
\text{let } x = P_i \text{ in } P_j]^p = [P_j]^p
\]

\[
\text{where } \rho = \rho(x \leftarrow [P_i]^p)
\]

\[
[(\mu_1, \Sigma_1, V_1, T_1)]^p = [P_1]^p \text{ and } (\mu_2, \Sigma_2, V_2, T_2)]^p = [P_2]^p
\]

\[
\text{where } (\mu_1, \Sigma_1, V_1, T_1) = [P_1]^p \text{ and } (\mu_2, \Sigma_2, V_2, T_2)]^p = [P_2]^p
\]

\[
\text{let } x = \text{Split}(P_1, p) \text{ in } P_2]^p = [P_2]^p
\]

\[
\text{where } r = [p]^p, 0 < r < 1 \text{ and } (\mu, \Sigma, V, T) = [P_1]^p
\]

\[
\text{and } \rho = \rho(x \leftarrow (\mu, \Sigma, V, T), y \leftarrow (\mu, \Sigma, (1 - r)V, T))
\]

\[
[(\mu_1, \Sigma_{\mu, \Sigma}(t), V, T)]^p = (\mu_1(t), \Sigma_{\mu, \Sigma}(t), V, T)
\]

\[
\text{where } t = [p]^p \text{ and } (\mu, \Sigma, V, T) = [P]^p
\]

\[
[\text{Dispose}(P)]^p = (0^{|A|}, 0^{|A| \times |A|}, 0, 0)
\]

\[
\text{together with } [p]^p \text{ defined as:}
\]

\[
[z]^p = \rho(z)
\]

\[
[r]^p = r
\]
Stochastic Analysis

- We can ask: what is the probability of a certain outcome given uncertainties in *both the protocol and the model*?
- Conversely: which parameters of *both the protocol and the model* best fit the observed result?
- E.g., we can use Statistical Model checking to estimate the probability that the output will fall in a certain range, given the distributions over uncertain model and protocol parameters.
Automated discovery loop:

- **Model + Protocol**
- **Realization of the model** (e.g., DNA Synthesis)
- **Protocol execution** (e.g., digital microfluidics)
- **Readout** (e.g., DNA sequencing)
- **Data analysis**
Simulating Reaction Networks together with Digital Protocols
An integrated language for chemical models & experimental protocols

Deterministic (ODE) and stochastic (LNA) simulation

Chemical reaction networks (CRNs) and liquid-handling protocols

Reaction scores

Functional scripting

GUI

Search "Kaemika" in the app stores
http://lucacardelli.name/kaemika.html

CMSB’2020 Best Tool Paper Award
Main features

• *Species and reactions*
  • Characterized by initial values and rates

• “Samples” (*compartments*) and *Protocols*
  • Isolate species and reactions in a compartment, and mix compartments

• *Kinetics (simulation)*
  • Deterministic (ODE) or stochastic (LNA) for chemical models
  • Digital microfluidics for chemical protocols

• *Programming abstractions*
  • Assemble models and protocols as compositions of modules
Species and Reactions

//======================================
// Lotka 1920, Volterra 1926
// (simplified with all rates = 1)
//======================================

number x1_0 <- uniform(0, 1) // random x1_0
number x2_0 <- uniform(0, 1) // random x2_0

species x1 @ x1_0 M // prey
species x2 @ x2_0 M // predator

x1 -> x1 + x1 {1} // prey reproduces
x1 + x2 -> x2 + x2 {1} // predator eats prey
x2 -> Ø {1} // predator dies

equilibrate for 40

<= Demo: LotkaVolterra
Ex: Predatorial (recursive model)

```plaintext
function Predatorial(number n) {
    if n = 0 then
        define species prey @ 1 M
        prey -> 2 prey // prey reproduces
        report prey
        yield prey
    else
        define species predator @ 1/n M
        species prey = Predatorial(n-1)
        prey + predator ->{n} 2 predator // predator eats
        predator -> Ø // predator dies
        report predator
        yield predator
    end
}

species apexPredator = Predatorial(5)
equilibrate for 50

//======================================
// Creates a stack of predator-prey
// relationships in Lotka-Volterra style,
// and returns the apex predator.
//======================================
```

<= Demo: Predatorial
Extracting the Model and the Protocol

From the script

- **species** \{c\}
- **sample** A
  - **species** a @ 1M in A
  - **amount** c @ 0.1M in A
  - **reaction**: \(a + c \rightarrow a + a\)
  - **equilibrate** A1 = A for 1
- **sample** B
  - **species** b @ 1M in B
  - **amount** c @ 0.1M in B
  - **reaction**: \(b + c \rightarrow c + c\)
  - **equilibrate** B1 = B for 1
- **split** C,D = A1 by 0.5
- **dispose** C
- **mix** E = D with B1
- **reaction**: \(a + b \rightarrow b + b\)
- **equilibrate** F = E for 20
- **dispose** F

The protocol

The (final) model (sample E)
Writing Models Compositionally

• Embedded chemical notation
  Programs freely contain both chemical reactions and control flow
  Can generate unbounded-size reaction networks

• Rich data types
  numbers, species, functions, networks, lists, flows (time-courses)
  flows are composable functions of time used in rates, plotting, and observation

• Modern abstractions
  Functional: programs take data as parameters and produce data as results
  Monadic: programs also produce effects (species, reactions, liquid handling)
  Nominal: lexically scoped chemical species (species are not “strings”)
Mass Action Compiler

- Lorenz chaotic attractor

\[ \begin{align*}
\frac{dx}{dt} &= s \cdot y - s \cdot x \\
\frac{dy}{dt} &= r \cdot x - x \cdot z - y \\
\frac{dz}{dt} &= x \cdot y - b \cdot z
\end{align*} \]

\[ s = 10 \]
\[ b = 8/3 \]
\[ r = 28 \]
\[ x_0 = 1 \]
\[ y_0 = 0 \]
\[ z_0 = 28 \]

\text{not mass action}

\begin{align*}
x' + x' &\rightarrow \emptyset \\
y' &\rightarrow y' + x' \ (10) \\
x' &\rightarrow x' + y' \ (10) \\
x' &\rightarrow x' + x' \\
y' + y' &\rightarrow \emptyset \\
z' + x' &\rightarrow z' + x' + y' \\
z' + x' &\rightarrow z' + x' + y' \\
x' &\rightarrow x' + y' \ (28) \\
y' &\rightarrow y' \ \\
z' + x' &\rightarrow z' + y' + y' \\
x' &\rightarrow x' + y' \ (28) \\
y' &\rightarrow y' + y' \\
z' + z' &\rightarrow \emptyset \\
y' + x' &\rightarrow y' + x' + z' \\
y' + x' &\rightarrow y' + x' + z' \\
z' &\rightarrow z' + z' \ (2.667) \\
y' + x' &\rightarrow y' + x' + z' \\
y' + x' &\rightarrow y' + x' + z' \\
z' &\rightarrow z' + z' \ (2.667)
\end{align*}

Initial:
\[ x' = 1 \]
\[ x' = 0 \]
\[ y' = 0 \]
\[ y' = 0 \]
\[ z' = 28 \]
\[ z' = 0 \]

\text{<= Demo: LorenzAttractor}
Kaemika Microfluidics Compiler

- Mix, split, equilibrate, dispose
- Automatic routing – no geometrical information
- Hot/cold zones

sample A \{3\mu L, 20^\circ C\}

split B,C,D,E = A

mix F = E,C,B,D

dispose F
Conclusions

Integrated modeling
Of chemical reaction networks and protocols
How the Kaemika app supports it
Why it needs a new language for smooth integration

Closed-loop modeling, experimentation and analysis
For complete lab automation
To “scale up” the scientific method