Outline

• The Scientific Method
  - and its eventual automation

• Models
  - that know nothing about protocols
    Chemical Reaction Networks

• Lab Protocols
  - that know nothing about models
    Digital Microfluidics

• Integration
  Closed-loop modeling and protocol execution
  The Kaemika App
Discovery through Observation

The Scientific Method ~ 1638

1 Guy
Discovery through Collaboration

The Scientific Method ~ 2000’s

1 Lab

1 protein = 30 people / 30 years
Humans have >250,000 proteins 😊
Discovery through Automation

The Scientific Method ~ 2020’s

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

The Inner Loop

• A *model* is refined by testing a (fixed) *protocols* against a *systems*
• A *protocol* is refined by testing a (fixed) *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](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
- **Lifecycle**: Performance evaluation/optimization: of model+protocol+system combined
- **Management**: version control, equipment monitoring, data storage
The Inner Loop

- Tomorrow, automation
- Model: unambiguous (formal) 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
Why are chemical reactions interesting?

\[ X + Y \rightarrow^r Z + W \]

- 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*
100 years of chemical infinite loops

Chemical reaction networks are interesting independently of actual chemical substances:

2 ordinary differential equations (ODEs) with chemical reaction network (CRN) interpretation, but no actual chemicals.

1920 First theoretical proof of chemical oscillation [Lotka]
1921 First experimental (accidental) chemical oscillator [Bray]
1926 Predatory-prey interpretation (independent discovery) [Volterra]
1958 Bray ignored until the (accidental) BZ oscillator [Belousov–Zhabotinsky]
1963 Lorenz chaotic oscillator (3 ODEs, can be adapted to mass action)
1981 First intentionally-designed chemical oscillator [De Kepper]
2005 First biochemical protein/ATP oscillator (circadian clock) [Nakajima et al.]
2017 First DNA-only oscillator [Srinivas et al.] (a version of Lotka's)
Chemical algorithms

- Hungarian Lemma: ODE -> CRN

 Lotka-Volterra Population ODEs

\[ \frac{dx_1}{dt} = a_1 \cdot x_1 - b_1 \cdot x_1 \cdot x_2 \]
\[ \frac{dx_2}{dt} = a_2 \cdot x_1 \cdot x_2 - b_2 \cdot x_2 \]

Population algorithm (*)

\[ x_1 \rightarrow 2 \cdot x_1 \quad (a_1) \]
\[ x_1 + x_2 \rightarrow x_2 \quad (b_1) \]
\[ x_1 + x_2 \rightarrow x_1 + 2 \cdot x_2 \quad (a_2) \]
\[ x_2 \rightarrow \emptyset \quad (b_2) \]

preys increase predators decrease by \( b_1 \)
preys increase predators by \( a_2 \)
predators decrease (without preys)

- From populations to individuals (agents)

The interaction between prey population and predator population is indirect (2 separate reactions)
We can change it to a direct interaction between 1 prey agent and 1 predator agent, but we need to take \( a_2 = b_1 \). Consider the two \( x_1 + x_2 \) reactions:

\[ x_1 + x_2 \rightarrow x_2 \quad (b_1) \]
predators decrease preys (predators stay the same)

\[ x_1 + x_2 \rightarrow 2 \cdot x_2 \quad (b_1) \]
predator agent eats prey agent and reproduces

\[ x_1 + x_2 \rightarrow x_1 + 2 \cdot x_2 \quad (b_1) \]
preys increase predators (preys stay the same)

- Law of Mass Action: CRN -> ODE

Agent algorithm (*)

\[ x_1 \rightarrow 2 \cdot x_1 \quad (a_1) \]
\[ x_1 + x_2 \rightarrow 2 \cdot x_2 \quad (b_1) \]
\[ x_2 \rightarrow \emptyset \quad (b_2) \]

Agent ODEs

\[ \frac{dx_1}{dt} = a_1 \cdot x_1 - b_1 \cdot x_1 \cdot x_2 \]
\[ \frac{dx_2}{dt} = b_1 \cdot x_1 \cdot x_2 - b_2 \cdot x_2 \]

By turning each ODE monomial into one reaction
(Restricted to “Hungarian ODEs” such that all negative monomials have their l.h.s. differential variable as a factor. But by variable doubling this covers w.l.o.g. the solutions of all polynomial ODEs.)

\[ \frac{dx_1}{dt} = a_0 x_1 - b_3 x_1 x_2 \]
\[ \frac{dx_2}{dt} = a_0 x_1 x_2 - b_2 x_2 \]

\[ (A_0) \quad \frac{dx_1}{dt} = c(x_1 - \gamma x_2) \quad (A_0) \quad \frac{dx_2}{dt} = c(x_1 - \gamma x_2) \]

[Volterra 1926]

- Although Lotka 1920 intuitively describes just 3 reactions, \( b_1 \neq a_2 \) requires interpreting the ODEs as 4 reactions. Lotka’s motivation for \( b_1 \neq a_2 \), or rather for \( b_1 > a_2 \), is: “We may, however, make the more general supposition that along with \( x_2 \) any other substances are formed [further depleting \( x_1 \)].”

[Volterra 1926, eqn A1,A2] instead says that \( b_1 \) is the prey’s defensive ability, and \( a_2 \) is the predator’s offensive ability, so the intuition here is what happens when a population of prey meets a population of predators. This again is incompatible with a 3-reaction 1-prey vs. 1-predator CRN.

There can be multiple algorithms (CRNs) for the same behavior (ODEs). CRNs are programs.

ODEs support program equivalence.
Programming any dynamical system as a CRN

For example, take the canonical oscillator: sine/cosine

1. Polynomization: All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

2. Positivation: All polynomial ODEs can be exactly reduced to polynomial ODEs in the positive quadrant (as differences).

3. All positivized ODEs are Hungarian: I.e., all negative monomials have their l.h.s. differential variable as a factor.

4. Hungarization: All Hungarian ODEs can be exactly reduced to mass action CRNs.

5. Molecular Programming: All mass action CRNs, up to time rescaling, can be arbitrarily approximated by engineered DNA molecules.
Chemistry is also a formal language that we can use to implement *any* dynamical system with *real* (DNA) molecules.

- ANY collection of abstract chemical reactions can be implemented with specially designed DNA molecules, with accurate kinetics (up to time scaling).

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

DNA as a universal substrate for chemical kinetics

David Soloveichik, Georg Seelig, and Erik Winfree

Domains

- Subsequences on a DNA strand are called **domains**
  - *provided* they are “independent” of each other

- Differently named domains must not **hybridize**
  - With each other, with each other’s complement, with subsequences of each other, with concatenations of other domains (or their complements), etc.
Reaction \( x + y \rightarrow z + w \)

Reactants half

"join" structure

(2-input 2-output reactions are universal)
Reaction \[ x + y \rightarrow z + w \] products half

(2-input 2-output reactions are universal)
Reaction: $x + y \rightarrow z + w$

(2-input 2-output reactions are universal)
Join $x + y \rightarrow z$
DNA Implementation of the Approximate Majority Algorithm

\[ X + Y \rightarrow 2B \]
\[ B + X \rightarrow 2X \]
\[ B + Y \rightarrow 2Y \]
Experimental-Protocol Languages for Chemical Reaction Networks

- Overview and Reaction sublanguage
Automating “the whole thing”

- Protocols: sets of steps to direct lab machinery (or people)
  - Published in specialized journals. With varying accuracy.
- Models: sets of equations to predict the results of lab experiments
  - Published in Auxiliary Online Materials. With lots of typos.

- Protocols know nothing about models
  - What hypothesis is the protocol trying to test? It is not written in the protocol.
- Models know nothing about protocols
  - What lab conditions are being used to test the model? It is not written in the model.
- While presumably talking about the same system
  - Through the experiment.

- Reproducibility crisis
  - Experiments are hard to reproduce.
  - Even models are hard to reproduce!

- Similar to classical lifecycle problems in C.S.
  - Documentation (model) gets out of step from code (protocol) if their integration is not automated.
A Protocol

For DNA gate assembly and activation in vitro
A Model

A Chemical Reaction Network, provided explicitly or (in this case) generated from a higher-level description of the initial strands, according to the DNA strand displacement rules.
An Integrated Description

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

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

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

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

The deterministic case is a warm-up exercise, simpler to explain. Each program denotes a final state \( <\text{concentrations, volume, temperature}> \)

\( [P]^{\rho} \) 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} &= \\
&\quad \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 \frac{x_1^{1}V_1 + x_2^{2}V_2}{V_1 + V_2}, V_1 + V_2, \frac{T_1V_1 + T_2V_2}{V_1 + V_2} \)
\end{align*}
\]

\[
[\text{let } x = P_1 \text{ in } P_2]^{\rho} = \\
\quad \text{let } (x_0, V, T) = [P_1]^{\rho} \\
\quad \text{let } \rho_1 = \rho\{x \leftarrow (x_0, V \cdot p, T), y \leftarrow (x_0, V \cdot (1 - p), T)\} \\
\quad [P_2]^{\rho_1} \\
\text{[Equilibrate}(P, t)\text{]}^{\rho} = \\
\quad \text{let } (x_0, V, T) = [P]^{\rho} \\
\quad \text{[(A, R, x_0, V, T)](H)}(t) \\
\quad [\text{Dispose}(P)]^{\rho} = (0^{\vert\mathcal{A}\vert}, 0, 0),
\]

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

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

*Split* has a volume uncertainty.
*Equilibrate* has a time uncertainty.
Reactions have rate uncertainty and/or intrinsic molecular noise.

Each program now represents a Hybrid System with stochastic jumps between deterministic evolutions:

- **CRN deterministic evolution from** $x_0$ **to** $x_t$  
  stochastic jump at time $t$  
  its probability depends state $x_t$  
  sudden state change $x_t \rightarrow y_t$  
  e.g. due to *Mix* or *Split*  
- **CRN deterministic evolution from** $y_t$ **to** $y_u$  

Which in turn denotes a *Piecewise Deterministic Markov Process (PDMP)*
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?

1500 executions including protocol uncertainty due timing and pipetting errors (red).
1500 executions including only model uncertainty about rates of the CRN (yellow).
1500 executions including both sources of uncertainty (blue).

We may estimate by Statistic Model Checking, e.g. the probability that Output will fall in a certain range, given distributions over uncertain model and protocol parameters.
Kaemika

- A prototype language for chemical models & protocols
  
  - [http://lucacardelli.name/kaemika.html](http://lucacardelli.name/kaemika.html)
  
- Search "Kaemika" in the App stores

- CRN simulation
- Microfluidics simulation
- Reaction graphs
- ODE equations
- Stochastic noise (LNA)
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 x1₀ <- uniform(0,1)    // random x1₀
number x2₀ <- uniform(0,1)    // random x2₀

species x1 @ x1₀ M           // prey
species x2 @ x2₀ M           // predator

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

equililibrate for 40
Reaction scores (graphical representation of reaction networks)


- Reactants and products
- Repeated species
- Reactants but no products
- Products but no reactants
- Catalyst
- Catalyst but no reactants
- Catalyst but no products
- Autocatalyst
Writing Models Compositionally

- Models are generated by programs
  Freely containing 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”)

Ex: Predatorial

```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)
equilibrante for 50
```
Demo

- Lotka-Volterra
- Predatorial
Experimental-Protocol Languages for Chemical Reaction Networks

- Protocol sublanguage and Microfluidics
Describing a Protocol

- **Samples** (e.g. test tubes)
  - Are characterized by a volume and a temperature
  - Contain a specified set of species
  - Evolve according to reactions that operates on those species
  - Isolate species and reactions

- **Protocol Operations** (e.g. liquid handling)
  - Accept and produce samples
  - Accepted samples are *used up* (they can only be operated-on once)
Samples

- Samples contain concentrations of species, acted over by reactions.
- Each sample has a fixed volume and a fixed temperature through its evolution.
- Sample concentrations are in units of molarity $M = \text{mol/L}$.
- The default implicit sample is called the vessel $\{1 \text{ mL, 20 C}\}$.

An amount can also be given in grams (if molar mass is specified). The resulting concentration is then relative to sample volume.

species $\{c\}$ // a species for multiple samples

sample A $\{1\mu\text{L, 20C}\}$ // volume and temperature
species a @ 10mM in A // species local to A
amount c @ 1mM in A // amount of c in A
a + c -> a + a

sample B $\{1\mu\text{L, 20C}\}$
species b @ 10mM in B // species local to B
amount c @ 1mM in B // amount of c in B
b + c -> c + c

Reactions can be specified with Arrhenius parameters {collision frequency, activation energy}. The reaction kinetics is then relative to sample temperature $T$.

a + c -> $\{2, 5\}$ a + a
// rate is $2e^{(-5/(R*T))}$

species $\{\text{NaCl}\#58.44\}$
sample C $\{1 \text{ mL, 20C}\}$
amount NaCl @ 8g in C
Liquid Handling

Mix two samples into one
mix A = B, C

Split a sample into two
split B,C = A by 0.5

Let a sample evolve by its reactions
equilibrate A = B for 3

Throw away a sample
dispose C

Change sample temperature (heat or cool)
regulate A = B to 37°C

Change sample volume (concentrate or dilute)
concentrate A = B to 1mL
Demo: Sample Manipulation

species {c}
sample A
species a @ 1M in A
amount c @ 0.1M in A
a + c -> a + a
equilirate A1 = A for 1

ds | A1

sample B
species b @ 1M in B
amount c @ 0.1M in B
b + c -> c + c
equilirate B1 = B for 1

ds | B1

split C,D = A1 by 0.5
dispose C

mix E = D with B1
a + b -> b + b
equilirate F = E for 20
dispose F

Multiple equilibration (simulation) steps
Ex: Phosphate-buffered saline (PBS)

species {NaCl#58.44, KCl#74.5513, Na₂HPO₄#141.96, KH₂PO₄#136.086}
report NaCl, KCl, Na₂HPO₄, KH₂PO₄

function Autoclave(sample PBS, number t) {
    define
        // increase temperature, preserve volume:
        regulate hot = PBS to 121°C
        // bake
        equilibrate hot for t
        // decrease temperature, preserve volume:
        regulate PBS = hot to 20°C
    yield PBS
}

function MakePBS() {
    define
        sample PBS {800mL, 20°C}
        amount NaCl @ 8g in PBS
        amount KCl @ 0.2g in PBS
        amount Na₂HPO₄ @ 1.44g in PBS
        amount KH₂PO₄ @ 0.24g in PBS

        sample topup {200mL, 20°C}
        mix PBS = PBS,topup
    yield Autoclave(PBS, 20°60)
}

sample PBS = MakePBS()
Ex: Serial Dilution (recursive protocol)

```plaintext
network SerialDilution(number count, sample s, network f) {
    if count > 0 then
        sample solvent {9*observe(volume,s) L, observe(kelvin,s) K}
        mix s = s, solvent
        split s, dilution = s by 0.1, 0.9
        f(dilution)
        SerialDilution(count-1, s, f)
    end
}

//initial sample to be diluted:
sample init {1mL, 25C}
species A @ 1M in init
species B @ 1M in init
A + B ->{20} A
A -> Ø

//apply this network to each dilution;
//note that this invokes a simulation
//each time in each solution
network test(sample s) {
    equilibrate s for 10
    dispose s
}

//dilute 4 times
SerialDilution(4, init, test)
```

Prepare a series of increasingly diluted solutions and apply a network \( f \) to each \( (f \) can add species and reactions to the solutions)

RESULT:
- sample init {1mL, 298.2K} {A = 1M, B = 1M}
- sample s2 {1mL, 298.2K} {A = 100mM, B = 100mM}
- sample s4 {1mL, 298.2K} {A = 10mM, B = 10mM}
- sample s7 {1mL, 298.2K} {A = 1mM, B = 1mM}
- sample s10 {1mL, 298.2K} {A = 100uM, B = 100uM}
Executing the protocols

- We have seen that *reactions can be executed* by DNA

- But how can we *execute the protocols*, so that we can execute the whole thing together?

- -> Digital Microfluidics Compiler
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.
Digital Microfluidics

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

Speed test
https://www.youtube.com/watch?v=pSIs9L_h3Q0
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.
Digital 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
Demo

- Mix and Split
Extracting the Model and the Protocol

From the script

```
species {c}
sample A
  species a @ 1M in A
  amount c @ 0.1M in A
  a + c -> a + a
  equilibrate A1 = A for 1

sample B
  species b @ 1M in B
  amount c @ 0.1M in B
  b + c -> c + c
  equilibrate B1 = B for 1

split C, D = A1 by 0.5
dispose C
mix E = D with B1
  a + b -> b + b
equilibrate F = E for 20
dispose F
```

The protocol

The (final) model (sample E)
Extracting the Hybrid Transition System

From the script

species {c}

sample A
species a @ 1M in A
amount c @ 0.1M in A
a + c -> a + a
equilbrate A1 = A for 1

sample B
species b @ 1M in B
amount c @ 0.1M in B
b + c -> c + c
equilbrate B1 = B for 1

split C,D = A1 by 0.5
dispose C

mix E = D with B1
a + b -> b + b
equilbrate F = E for 20
dispose F

The full story (Hybrid system)
Conclusions

**Integrated modeling**
- Of chemical reaction networks and protocols
- How the Kaemika app supports it

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