Simulating Reaction Networks together with Digital Protocols
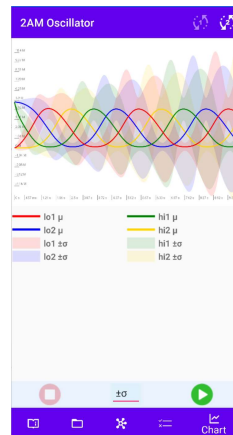
Luca Cardelli, University of Oxford
2021-06-17 IMT Lucca

Outline

- The Scientific Method

  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
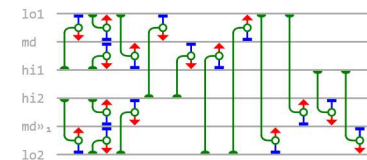 **The Kaemika App**

An integrated language for chemical models & experimental protocols

Deterministic (ODE) and stochastic (LNA) simulation

Chemical reaction networks (CRNs) and liquid-handling protocols
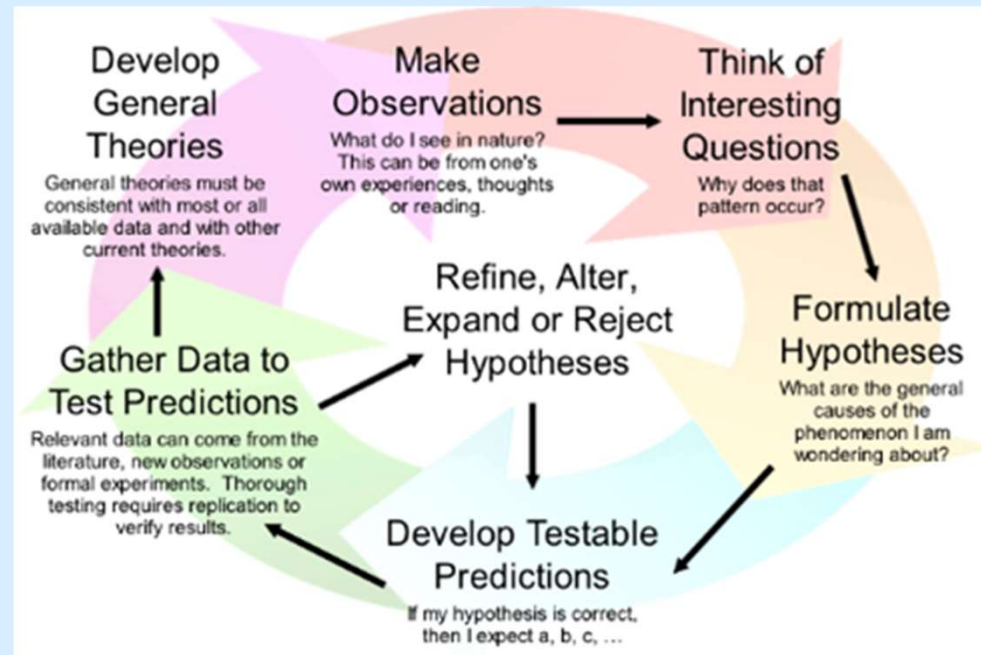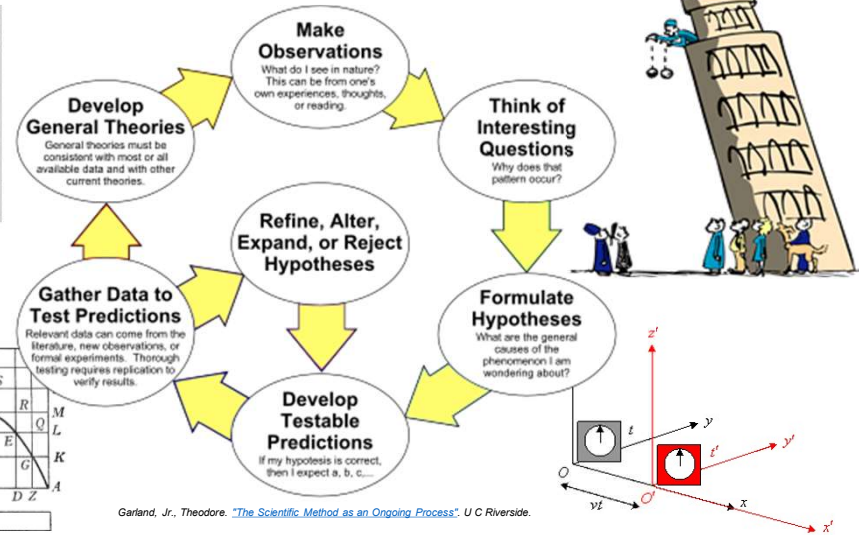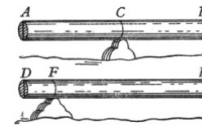
Reaction scores

Functional scripting

GUI

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

3

The Scientific Method

And its automation

Discovery through Observation

The Scientific Method ~ 1638

1 Guy

Make Observations
What do I see in nature? This can be from one's own experiences, thoughts, or reading.

Develop General Theories
General theories must be consistent with most or all available data and with other current theories.

Think of Interesting Questions
Why does that pattern occur?

Refine, Alter, Expand, or Reject Hypotheses

Gather Data to Test Predictions
Relevant data can come from the literature, new observations, or formal experiments. Thorough testing requires replication to verify results.

Formulate Hypotheses
What are the general causes of the phenomenon I am wondering about?

Develop Testable Predictions
If my hypotesis is correct, then I expect a, b, c,...

Garland, Jr., Theodore. *"The Scientific Method as an Ongoing Process"*. U C Riverside.

Discovery through Collaboration

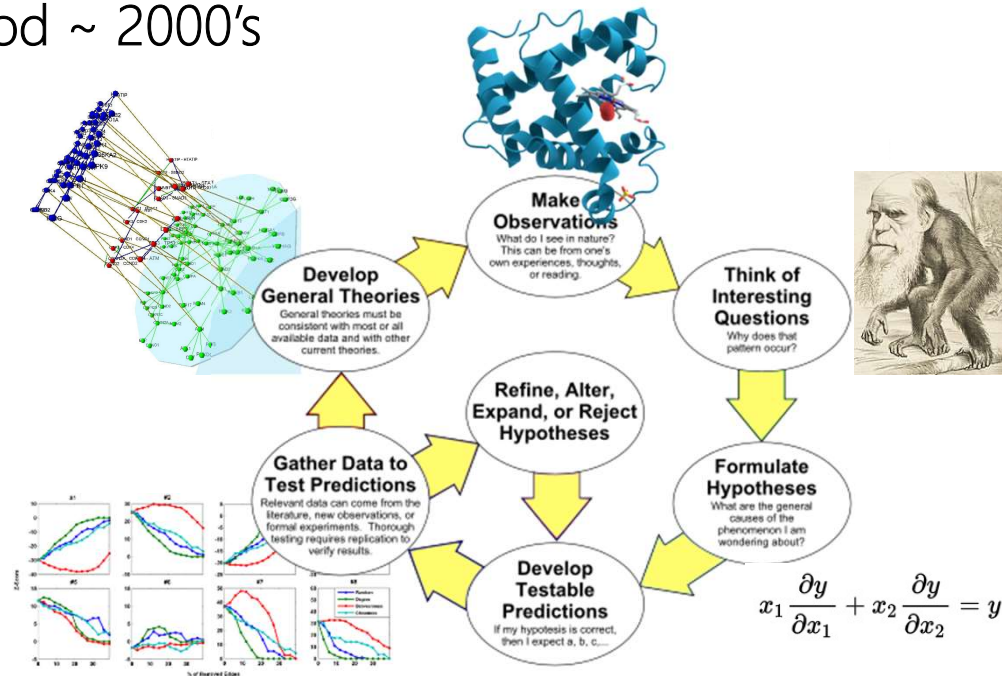## The Scientific Method ~ 2000's

1 Lab

1 protein = 30 people / 30 years

Humans have >250,000 proteins ☹

Make Observations
What do I see in nature?
This can be from one's own experiences, thoughts, or reading.

Think of Interesting Questions
Why does that pattern occur?

Develop General Theories
General theories must be consistent with most or all available data and with other current theories.

Refine, Alter, Expand, or Reject Hypotheses

Formulate Hypotheses
What are the general causes of the phenomenon I am wondering about?

Gather Data to Test Predictions
Relevant data can come from the literature, new observations, or formal experiments. Thorough testing requires replication to verify results.

Develop Testable Predictions
If my hypotesis is correct, then I expect a, b, c....

$$x_1 \frac{\partial y}{\partial x_1} + x_2 \frac{\partial y}{\partial x_2} = y$$

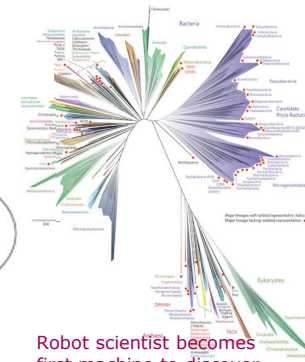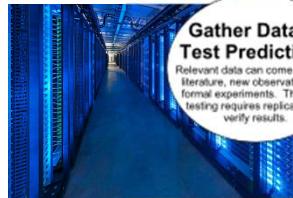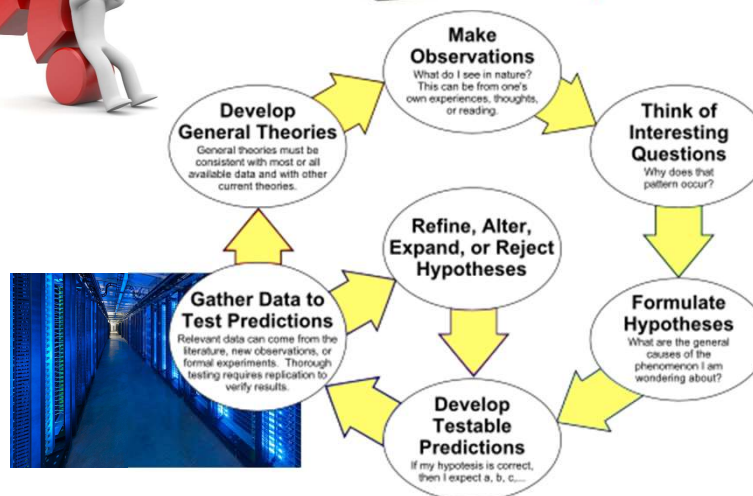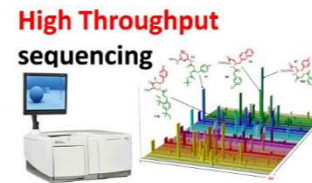Garland, Jr., Theodore. *"The Scientific Method as an Ongoing Process".* U C Riverside.

Discovery through Automation

The Scientific Method ~ 2020's

1 Program

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

High Throughput sequencing

Make Observations
What do I see in nature? This can be from one's own experiences, thoughts, or reading.

Think of Interesting Questions
Why does that pattern occur?

Develop General Theories
General theories must be consistent with most or all available data and with other current theories.

Refine, Alter, Expand, or Reject Hypotheses

Formulate Hypotheses
What are the general causes of the phenomenon I am wondering about?

Gather Data to Test Predictions
Relevant data can come from the literature, new observations, or formal experiments. Thorough testing requires replication to verify results.

Develop Testable Predictions
If my hypotesis is correct, then I expect a, b, c....

Robot scientist becomes first machine to discover new scientific knowledge
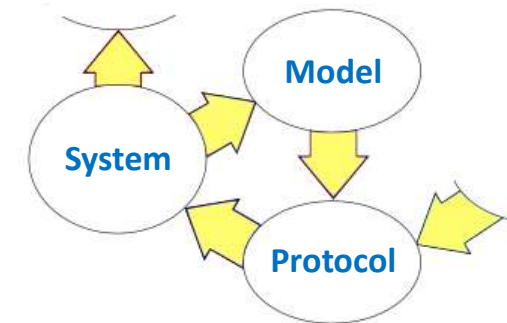
Ross King

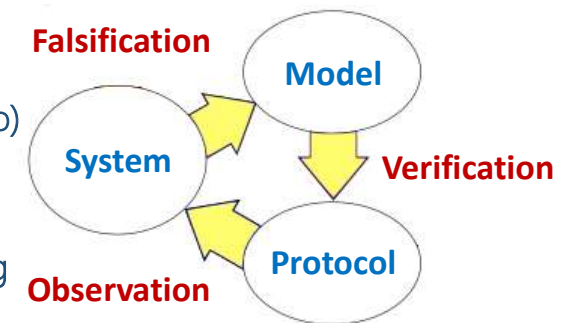Garland, Jr., Theodore. *"The Scientific Method as an Ongoing Process"*. U C Riverside.

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

The Inner Loop

- Tomorrow, automation

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

Arcs
- 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
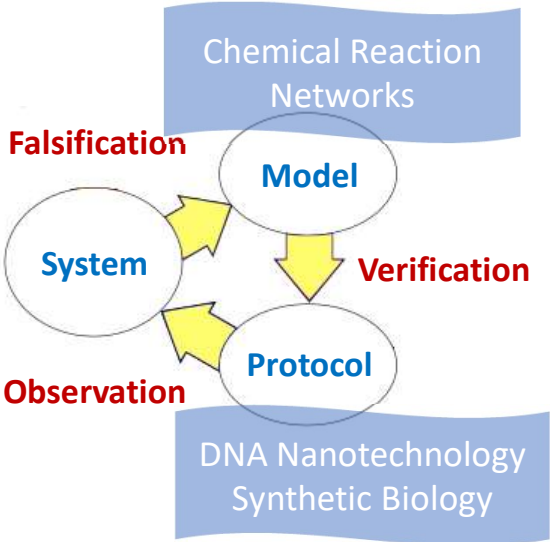
The Inner Loop

In this talk

via Molecular Programming

- Tomorrow, automation

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

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

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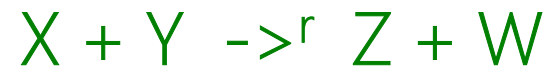- Management: version control, equipment monitoring, data storage

Chemical Reaction Networks

Falsification

Model

System

Verification

Protocol

Observation

DNA Nanotechnology Synthetic Biology

Models

(that know nothing about protocols)

Chemical reactions. Why are they interesting?

$$X + Y \, ->^r \, Z + W$$

- 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*

12

100 years of chemical infinite loops

It is, therefore, somewhat contrary to his first expectations that the writer now finds the conditions for undamped oscillations may occur in the absence of any geometrical causes in a homogeneous system.
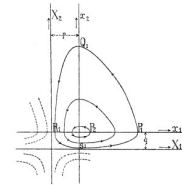
[Lotka, 1920]

Chemical reaction networks are interesting independently of actual chemical substances:

UNDAMPED OSCILLATIONS DERIVED FROM THE LAW OF MASS ACTION.

By ALFRED J. LOTKA.
Received June 2, 1920.

$$\frac{dX_1}{dt} = a_1 X_1 - b_1 X_1 X_2$$
$$\frac{dX_2}{dt} = a_2 X_1 X_2 - b_2 X_2.$$

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]

The first single, homogeneous oscillating chemical reaction was discovered accidentally by Bray [6] in 1921.

OSCILLATIONS AND CHAOS IN CHEMICAL SYSTEMS
Irving R. EPSTEIN
Department of Chemistry, Brandeis University, Waltham, MA 02254, USA

1958 Bray ignored until the (accidental) BZ oscillator [Belousov–Zhabotinsky]
1963 Lorenz chaotic oscillator (3 ODEs, can be adapted to mass action)
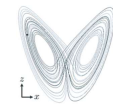1981 First intentionally-designed chemical oscillator [De Kepper]
2005 First biochemical protein/ATP oscillator (circadian clock) [Nakajima et al.]
2017 First DNA-only oscillator [Srinivas et al.] (a version of Lotka's)

13

Chemical *algorithms*

$$\frac{dX_1}{dt} = a_1 X_1 - b_1 X_1 X_2$$
$$\frac{dX_2}{dt} = a_2 X_1 X_2 - b_2 X_2.$$

prey population X_1

predator population X_2

$(A_1) \quad \frac{dN_1}{dt} = (\varepsilon_1 - \gamma_1 N_2) N_1 \qquad (A_2) \quad \frac{dN_2}{dt} = (-\varepsilon_2 + \gamma_2 N_1) N_2$ [Volterra 1926]

- Hungarian Lemma: ODE -> CRN

Lotka-Volterra
Population ODEs

$\partial\, x1 = a1 \cdot x1 - b1 \cdot x1 \cdot x2$
$\partial\, x2 = a2 \cdot x1 \cdot x2 - b2 \cdot x2$

Hungarization

Mass Action

Population algorithm [*]

x1 -> 2 · x1	{a1}	prey increase
x1 + x2 -> x2	{b1}	predators decrease prey by b1
x1 + x2 -> x1 + 2 · x2	{a2}	prey increase predators by a2
x2 -> Ø	{b2}	predators decrease (without prey)

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.)

[*] Although [Lotka 1920] intuitively describes just 3 reactions, b1≠a2 requires interpreting the ODEs as 4 reactions. Lotka's motivation for b1≠a2, or rather for b1>a2, is: "We may, however, make the more general supposition that along with x2 any other substances are formed [further depleting x1]".

[Volterra 1926, eqn A1,A2] instead says that b1 is the prey's defensive ability, and a2 is the predator's offensive ability, so the intuition here is what happens when a population of prey meets a population predators. This again is incompatible with a 3-reaction 1-prey vs. 1-predator CRN.
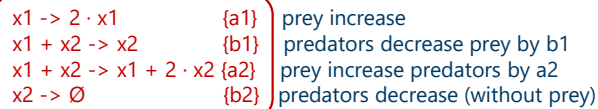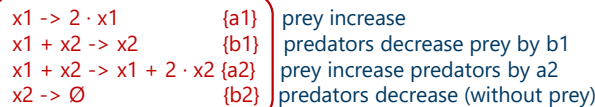
Chemical *algorithms*

$$\frac{dX_1}{dt} = a_1 X_1 - b_1 X_1 X_2$$
$$\frac{dX_2}{dt} = a_2 X_1 X_2 - b_2 X_2.$$

prey population X_1

predator population X_2

$(A_1) \quad \frac{dN_1}{dt} = (\varepsilon_1 - \gamma_1 N_2) N_1 \qquad (A_2) \quad \frac{dN_2}{dt} = (-\varepsilon_2 + \gamma_2 N_1) N_2$ [Volterra 1926]

- ## Hungarian Lemma: ODE -> CRN

Lotka-Volterra
Population ODEs

Population algorithm (*)

$\partial\, x1 = a1 \cdot x1 - b1 \cdot x1 \cdot x2$
$\partial\, x2 = a2 \cdot x1 \cdot x2 - b2 \cdot x2$

Hungarization

Mass Action

x1 -> 2 · x1 {a1}	prey increase
x1 + x2 -> x2 {b1}	predators decrease prey by b1
x1 + x2 -> x1 + 2 · x2 {a2}	prey increase predators by a2
x2 -> Ø {b2}	predators decrease (without prey)

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.)

(*) Although [Lotka 1920] intuitively describes just 3 reactions, b1≠a2 requires interpreting the ODEs as 4 reactions. Lotka's motivation for b1≠a2, or rather for b1>a2, is: "We may, however, make the more general supposition that along with x2 any other substances are formed [further depleting x1]".

[Volterra 1926, eqn A1,A2] instead says that b1 is the prey's defensive ability, and a2 is the predator's offensive ability, so the intuition here is what happens when a population of prey meets a population predators. This again is incompatible with a 3-reaction 1-prey vs. 1-predator CRN.

- ## 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 a2 = b1. Consider the two x1 + x2 reactions:

x1 + x2 -> x2 {b1}	predators *decrease* prey (predators stay the same)
x1 + x2 -> x1 + 2 · x2 {a2=b1}	prey *increase* predators (prey stay the same)

???

x1 + x2 -> 2 · x2 {b1} predator agent *eats* prey agent and *reproduces*

Chemical *algorithms*

$$\frac{dX_1}{dt} = a_1 X_1 - b_1 X_1 X_2$$
$$\frac{dX_2}{dt} = a_2 X_1 X_2 - b_2 X_2.$$

prey population X_1

predator population X_2

(A₁) $\frac{dN_1}{dt} = (\varepsilon_1 - \gamma_1 N_2) N_1$ (A₂) $\frac{dN_2}{dt} = (-\varepsilon_2 + \gamma_2 N_1) N_2$ [Volterra 1926]

- ## Hungarian Lemma: ODE -> CRN

Lotka-Volterra
Population ODEs

$\partial\, x1 = a1 \cdot x1 - b1 \cdot x1 \cdot x2$
$\partial\, x2 = a2 \cdot x1 \cdot x2 - b2 \cdot x2$

Hungarization →
← Mass Action

Population algorithm (*)

x1 -> 2 · x1	{a1}	prey increase
x1 + x2 -> x2	{b1}	predators decrease prey by b1
x1 + x2 -> x1 + 2 · x2	{a2}	prey increase predators by a2
x2 -> Ø	{b2}	predators decrease (without prey)

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.)

(*) Although [Lotka 1920] intuitively describes just 3 reactions, b1≠a2 requires interpreting the ODEs as 4 reactions. Lotka's motivation for b1≠a2, or rather for b1>a2, is: "We may, however, make the more general supposition that along with x2 any other substances are formed [further depleting x1]".

[Volterra 1926, eqn A1,A2] instead says that b1 is the prey's defensive ability, and a2 is the predator's offensive ability, so the intuition here is what happens when a population of prey meets a population predators. This again is incompatible with a 3-reaction 1-prey vs. 1-predator CRN.

- ## From populations to individuals (agents)

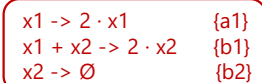The interaction between prey population and predator population is indirect (2 separate reactions)
We can change it to a direct interaction between 1 prey agent and 1 predator agent, but we need to take a2 = b1. Consider the two x1 + x2 reactions:

| x1 + x2 -> x2 | {b1} | predators *decrease* prey (predators stay the same) |
| x1 + x2 -> x1 + 2 · x2 | {a2=b1} | prey *increase* predators (prey stay the same) |

→ ??? →

| x1 + x2 -> 2 · x2 | {b1} |

predator agent *eats* prey agent and *reproduces*

- ## Law of Mass Action: CRN -> ODE

Agent algorithm (*)

x1 -> 2 · x1	{a1}
x1 + x2 -> 2 · x2	{b1}
x2 -> Ø	{b2}

Mass Action →

Agent ODEs

$\partial\, x1 = a1 \cdot x1 - b1 \cdot x1 \cdot x2$
$\partial\, x2 = b1 \cdot x1 \cdot x2 - b2 \cdot x2$

PolulationAlg = AgentAlg, when a2=b1

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

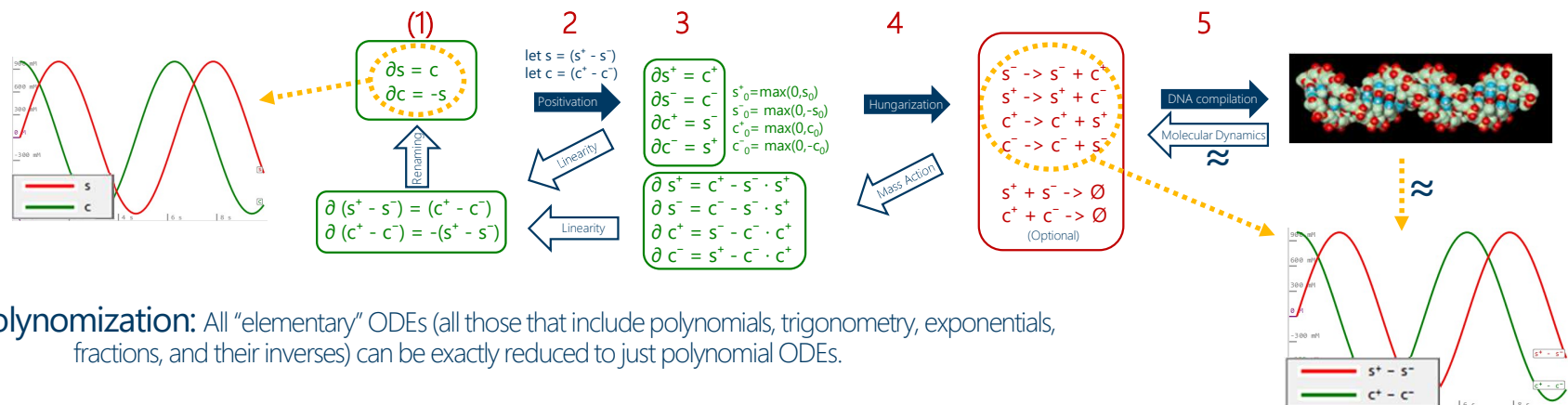# Programming *any* "*elementary*" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine

(1)
$$\partial s = c$$
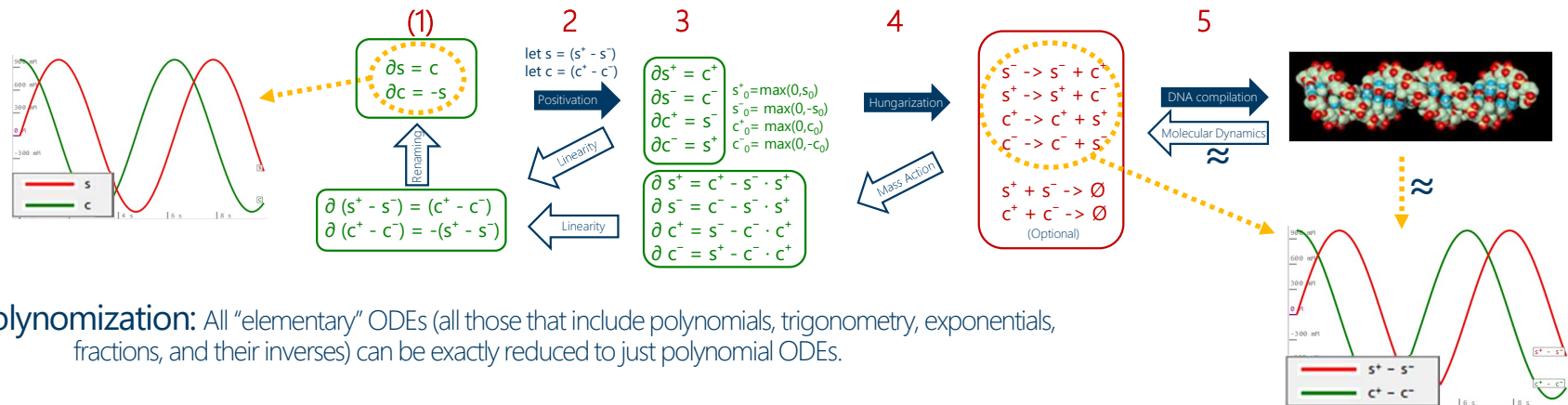$$\partial c = -s$$

Renaming

$$\partial (s^+ - s^-) = (c^+ - c^-)$$
$$\partial (c^+ - c^-) = -(s^+ - s^-)$$

Linearity

2

let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

Positivation

Linearity

3

$$\partial s^+ = c^+$$
$$\partial s^- = c^-$$
$$\partial c^+ = s^-$$
$$\partial c^- = s^+$$

$s^+_0 = max(0, s_0)$
$s^-_0 = max(0, -s_0)$
$c^+_0 = max(0, c_0)$
$c^-_0 = max(0, -c_0)$

$$\partial s^+ = c^+ - s^- \cdot s^+$$
$$\partial s^- = c^- - s^- \cdot s^+$$
$$\partial c^+ = s^- - c^- \cdot c^+$$
$$\partial c^- = s^+ - c^- \cdot c^+$$

Hungarization

Mass Action

4

$$s^- \rightarrow s^- + c^-$$
$$s^+ \rightarrow s^+ + c^-$$
$$c^+ \rightarrow c^+ + s^+$$
$$c^- \rightarrow c^- + s^-$$

$$s^+ + s^- \rightarrow \varnothing$$
$$c^+ + c^- \rightarrow \varnothing$$
(Optional)

DNA compilation

Molecular Dynamics ≈

5

≈

17

Programming *any* "elementary" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine

(1)
$$\partial s = c$$
$$\partial c = -s$$

Renaming

$$\partial (s^+ - s^-) = (c^+ - c^-)$$
$$\partial (c^+ - c^-) = -(s^+ - s^-)$$

Linearity

2
let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

Positivation

3
$$\partial s^+ = c^+$$
$$\partial s^- = c^-$$
$$\partial c^+ = s^-$$
$$\partial c^- = s^+$$

$s^+_0 = max(0, s_0)$
$s^-_0 = max(0, -s_0)$
$c^+_0 = max(0, c_0)$
$c^-_0 = max(0, -c_0)$

Linearity

$$\partial s^+ = c^+ - s^- \cdot s^+$$
$$\partial s^- = c^- - s^- \cdot s^+$$
$$\partial c^+ = s^- - c^- \cdot c^+$$
$$\partial c^- = s^+ - c^- \cdot c^+$$

Mass Action

4

Hungarization

$$s^- \rightarrow s^- + c^+$$
$$s^+ \rightarrow s^+ + c^-$$
$$c^+ \rightarrow c^+ + s^+$$
$$c^- \rightarrow c^- + s^-$$

$$s^+ + s^- \rightarrow \emptyset$$
$$c^+ + c^- \rightarrow \emptyset$$
(Optional)

5
DNA compilation
Molecular Dynamics
≈

≈

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

Programming *any* *"elementary"* dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine

(1)

$$\partial s = c$$
$$\partial c = -s$$

2

let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

Positivation

Renaming

$$\partial (s^+ - s^-) = (c^+ - c^-)$$
$$\partial (c^+ - c^-) = -(s^+ - s^-)$$

Linearity

3

$$\partial s^+ = c^+$$
$$\partial s^- = c^-$$
$$\partial c^+ = s^-$$
$$\partial c^- = s^+$$

$s^+_0 = max(0, s_0)$
$s^-_0 = max(0, -s_0)$
$c^+_0 = max(0, c_0)$
$c^-_0 = max(0, -c_0)$

Linearity

$$\partial s^+ = c^+ - s^- \cdot s^+$$
$$\partial s^- = c^- - s^- \cdot s^+$$
$$\partial c^+ = s^- - c^- \cdot c^+$$
$$\partial c^- = s^+ - c^- \cdot c^+$$

Hungarization

Mass Action

4

$s^- \to s^- + c^+$
$s^+ \to s^+ + c^-$
$c^+ \to c^+ + s^+$
$c^- \to c^- + s^-$

$s^+ + s^- \to \emptyset$
$c^+ + c^- \to \emptyset$
(Optional)

5

DNA compilation

Molecular Dynamics

≈

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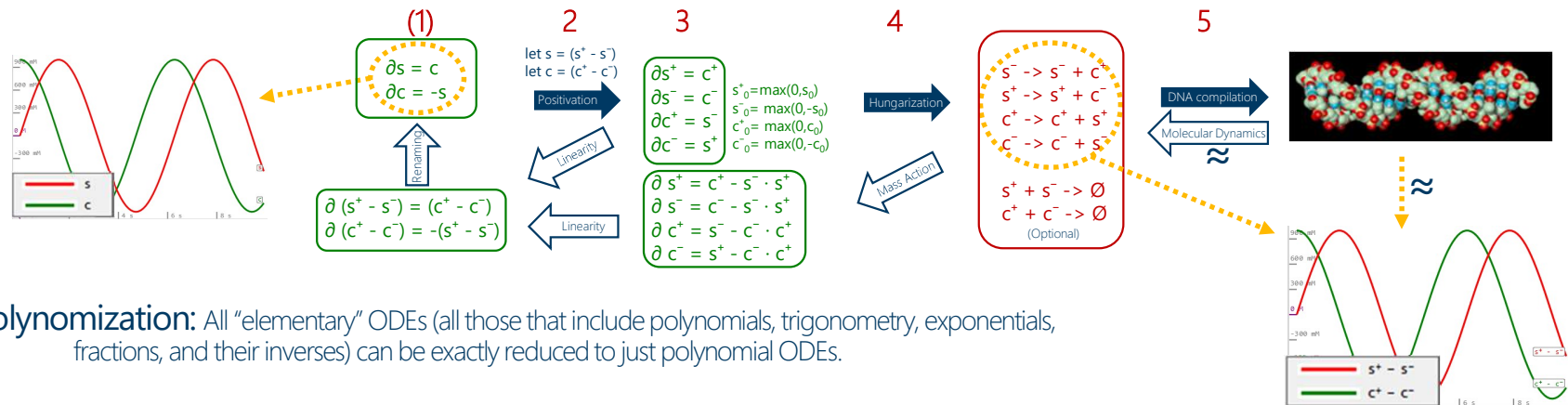
1. Polynomization: All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

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

Programming *any* "elementary" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine

(1)
$$\partial s = c$$
$$\partial c = -s$$

2
let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$
Positivation

3
$$\partial s^+ = c^+$$
$$\partial s^- = c^-$$
$$\partial c^+ = s^-$$
$$\partial c^- = s^+$$

$s^+_0 = \max(0, s_0)$
$s^-_0 = \max(0, -s_0)$
$c^+_0 = \max(0, c_0)$
$c^-_0 = \max(0, -c_0)$

4
Hungarization

$$s^- \rightarrow s^- + c^+$$
$$s^+ \rightarrow s^+ + c^-$$
$$c^+ \rightarrow c^+ + s^+$$
$$c^- \rightarrow c^- + s^-$$
$$s^+ + s^- \rightarrow \emptyset$$
$$c^+ + c^- \rightarrow \emptyset$$
(Optional)

5
DNA compilation
Molecular Dynamics
≈

Renaming

$$\partial (s^+ - s^-) = (c^+ - c^-)$$
$$\partial (c^+ - c^-) = -(s^+ - s^-)$$

Linearity

$$\partial s^+ = c^+ - s^- \cdot s^+$$
$$\partial s^- = c^- - s^- \cdot s^+$$
$$\partial c^+ = s^- - c^- \cdot c^+$$
$$\partial c^- = s^+ - c^- \cdot c^+$$

Mass Action

≈

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.

Programming *any* "*elementary*" 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.

Programming *any* "elementary" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine

(1)

$\partial s = c$
$\partial c = -s$

2

let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

Positivation

3

$\partial s^+ = c^+$
$\partial s^- = c^-$
$\partial c^+ = s^-$
$\partial c^- = s^+$

$s^+_0 = max(0, s_0)$
$s^-_0 = max(0, -s_0)$
$c^+_0 = max(0, c_0)$
$c^-_0 = max(0, -c_0)$

4

Hungarization

$s^- \rightarrow s^- + c^+$
$s^+ \rightarrow s^+ + c^-$
$c^+ \rightarrow c^+ + s^+$
$c^- \rightarrow c^- + s^-$

$s^+ + s^- \rightarrow \varnothing$
$c^+ + c^- \rightarrow \varnothing$
(Optional)

5

DNA compilation

Molecular Dynamics $\approx$

Renaming / Linearity / Linearity

$\partial (s^+ - s^-) = (c^+ - c^-)$
$\partial (c^+ - c^-) = -(s^+ - s^-)$

Mass Action

$\partial s^+ = c^+ - s^- \cdot s^+$
$\partial s^- = c^- - s^- \cdot s^+$
$\partial c^+ = s^- - c^- \cdot c^+$
$\partial c^- = s^+ - c^- \cdot c^+$

$\approx$

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 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).

DNA as a universal substrate for chemical kinetics

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

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
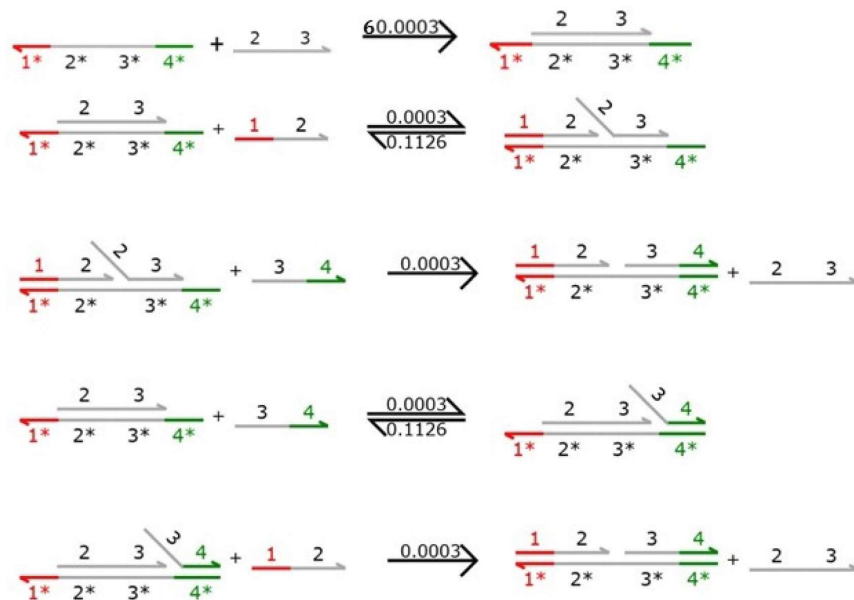
Model Semantics (deterministic)

- ODE semantics of CRNs

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):

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

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

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

Protocols

(that know nothing about models)

A Protocol

For DNA gate assembly and activation in vitro

Protocol steps (liquid handing)
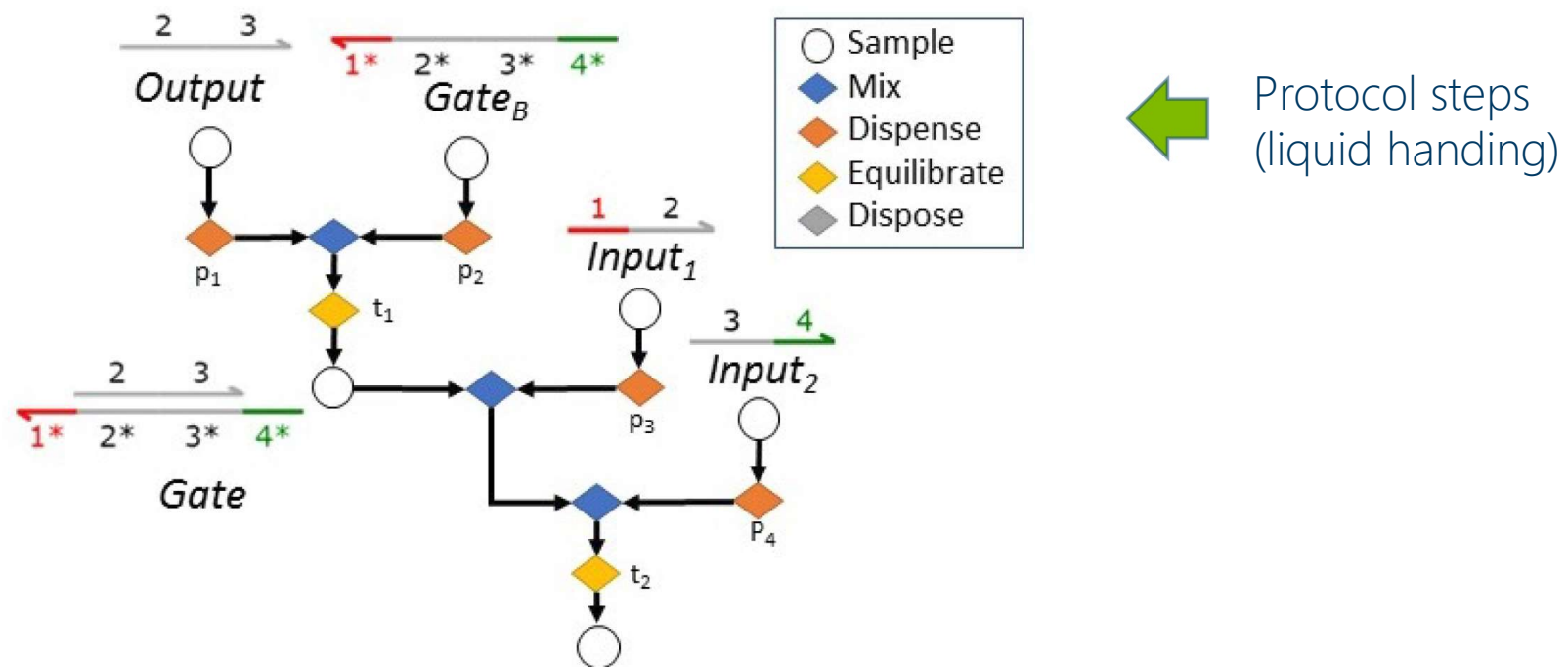
Digital Microfluidics

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

OpenDrop speed test
https://www.youtube.com/watch?v=pSls9L_h3Q0

Purple Drop (UW)
https://misl.cs.washington.edu/projects/fluidics.html
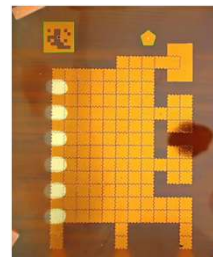
Digital Microfluidics

- 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.

30

A Protocol Language

Samples: containers with volume, temperature, concentrations

$$P =$$

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

Alessandro Abate[2], Luca Cardelli[1,2], Marta Kwiatkowska[2], Luca Laurenti[2], and Boyan Yordanov[1]

[1] Microsoft Research Cambridge
[2] Department of Computer Science, University of Oxford

Protocol Semantics (deterministic)

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

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

$$[\![x]\!]^{\rho} = \rho(x)$$
$$[\![x_0, V, T]\!]^{\rho} = (x_0, V, T)$$
$$[\![Mix(P_1, P_2)]\!]^{\rho} =$$
$$\quad let\ (x_0^1, V_1, T_1) = [\![P_1]\!]^{\rho}$$
$$\quad let\ (x_0^2, V_2, T_2) = [\![P_2]\!]^{\rho}$$
$$\quad (\frac{x_0^1 V_1 + x_0^2 V_2}{V_1 + V_2}, V_1 + V_2, \frac{T_1 V_1 + T_2 V_2}{V_1 + V_2})$$
$$[\![let\ x = P_1\ in\ P_2]\!]^{\rho} =$$
$$\quad let\ (x_0, V, T) = [\![P_1]\!]^{\rho}$$
$$\quad let\ \rho_1 = \rho\{x \leftarrow (x_0, V, T)\}$$
$$\quad [\![P_2]\!]^{\rho_1}$$

$$[\![let\ x, y = Split(P_1, p)\ in\ P_2]\!]^{\rho} =$$
$$\quad let\ (x_0, V, T) = [\![P_1]\!]^{\rho}$$
$$\quad let\ \rho_1 = \rho\{x \leftarrow (x_0, V \cdot p, T), y \leftarrow (x_0, V \cdot (1-p), T)\}$$
$$\quad [\![P_2]\!]^{\rho_1}$$

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

Kaemika Microfluidics Compiler

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

```
sample A {3µL, 20C}

split B,C,D,E = A

mix F = E,C,B,D

dispose F
```


< 20C   > 40C
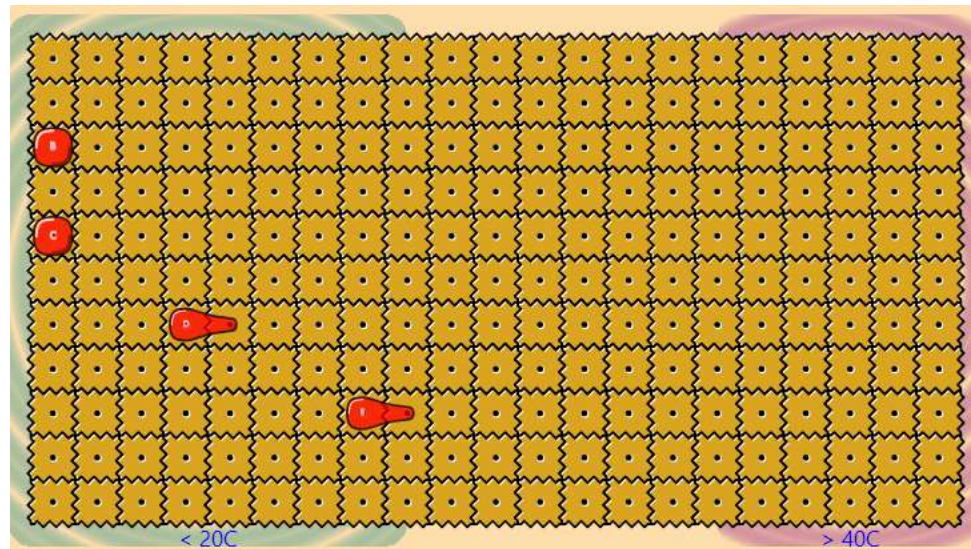
Summarizing

- 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 (by digital microfluidics)

Models *together with* Protocols

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. (materials, conditions, shortcuts)
  - Even models are hard to reproduce! (typos in equations, sketchy diagrams, unexplained graphs, mysterious scripts)

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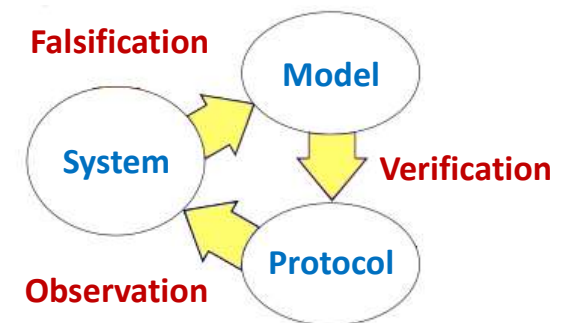
An Integrated Description

Samples: containers with volume, temperature, concentrations

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

Protocol   Model

+

=

Joint script

$Input_1 = <\ 1^*\ 2\ > \quad Output = <\ 2\ 3\ >$
$Input_2 = <\ 3\ 4^*\ > \quad Gate = \{1^*\}[2\ 3]\{4^*\}$

$P_1 = let\ In1 = ((Input1, 100.0nM), 0.1mL, 25.0^oC)\ in$
$\quad let\ In2 = ((Input2, 100.0nM), 0.1mL, 25.0^oC)\ in$
$\quad let\ GA = ((Output, 100.0nM), 0.1mL, 25.0^oC)\ in$
$\quad let\ GB = ((Gate_B, 100.0nM), 0.1mL, 25.0^oC)\ in$
$\quad let\ sGA_{,=}\ Dispense(GA, p_1)\ in$
$\quad let\ sGB_{,=}\ Dispense(GB, p_2)\ in$
$\quad let\ sIn1_{,=}\ Dispense(In1, p_3)\ in$
$\quad let\ sIn2_{,=}\ Dispense(In1, p_4)\ in$
$\quad Observe(Equilibrate(Mix(Mix(Equilibrate($
$\quad\quad Mix(sGA, sGB), t_1), sIn1), sIn2), t_2), idn).$

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

$\mathcal{C} = (\mathcal{A}, \mathcal{R})$ (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 $\mathcal{C} = (\mathcal{A}, \mathcal{R})$:

$[\![x]\!]^\rho = \rho(x)$

$[\![x_0, V, T]\!]^\rho = (x_0, V, T)$

$[\![Mix(P_1, P_2)]\!]^\rho =$
    $let\ (x_0^1, V_1, T_1) = [\![P_1]\!]^\rho$
    $let\ (x_0^2, V_2, T_2) = [\![P_2]\!]^\rho$
    $(\dfrac{x_0^1 V_1 + x_0^2 V_2}{V_1 + V_2}, V_1 + V_2, \dfrac{T_1 V_1 + T_2 V_2}{V_1 + V_2})$

$[\![let\ x = P_1\ in\ P_2]\!]^\rho =$
    $let\ (x_0, V, T) = [\![P_1]\!]^\rho$
    $let\ \rho_1 = \rho\{x \leftarrow (x_0, V, T)\}$
    $[\![P_2]\!]^{\rho_1}$

$[\![let\ x, y = Split(P_1, p)\ in\ P_2]\!]^\rho =$
    $let\ (x_0, V, T) = [\![P_1]\!]^\rho$
    $let\ \rho_1 = \rho\{x \leftarrow (x_0, V \cdot p, T), y \leftarrow (x_0, V \cdot (1 - p), T)\}$
    $[\![P_2]\!]^{\rho_1}$

$[\![Equilibrate(P, t)]\!]^\rho =$
    $let\ (x_0, V, T) = [\![P]\!]^\rho$
    $[\![(\mathcal{A}, \mathcal{R}, x_0), V, T]\!](H)(t)$

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

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

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

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.

Kaemika uses such a joint simulation algorithm for *stochastic* simulation, passing also 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.

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?

- Also, we can use Statistical Modelchecking:
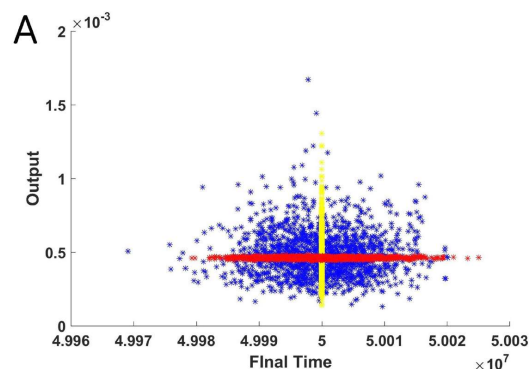
1500 executions including protocol uncertainty due timing and pipetting errors (red).
1500 executions including only model uncertainty about rates of the CRN (yellow).
1500 executions including both sources of uncertainty (blue).

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.

Simulating Reaction Networks together with Digital Protocols

Kaemika

- A prototype language for chemical models & protocols

- http://lucacardelli.name/kaemika.html

- Search "Kaemika" in the App stores
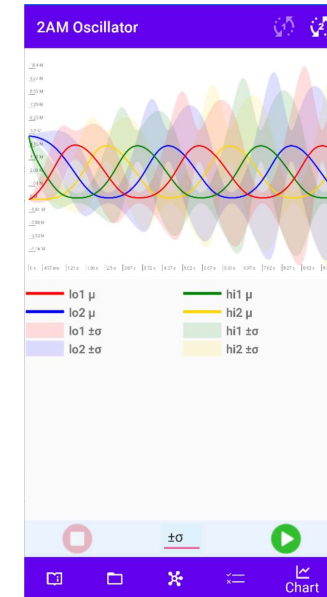
- CRN simulation
- Microfluidics simulation
- Reaction graphs
- ODE equations
- Stochastic noise (LNA)

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

UNDAMPED OSCILLATIONS, ETC.　1595

UNDAMPED OSCILLATIONS DERIVED FROM THE LAW OF MASS ACTION.

By Alfred J. Lotka.

Received June 2, 1920.

```
//=======================================
// 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


equilibrate for 40
```
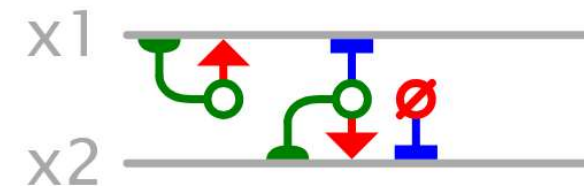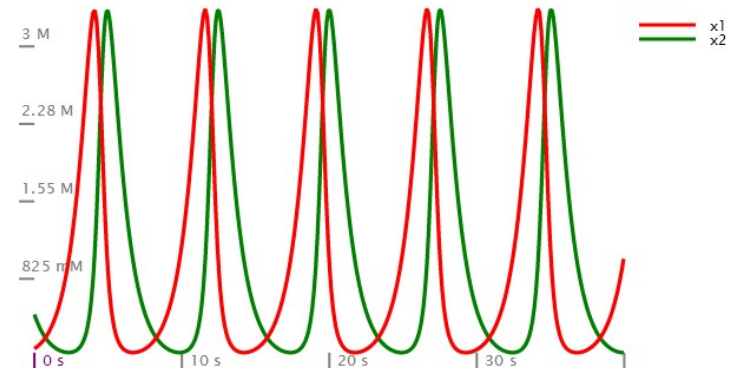
Reaction scores (graphical representation of reaction networks)

Horizonal lines: *species*. Vertical stripes: *reactions*.   Blue: reagents. Red: products. Green: catalysts.

Reactants and products

a + b -> c + d

Repeated species

2b -> c + d
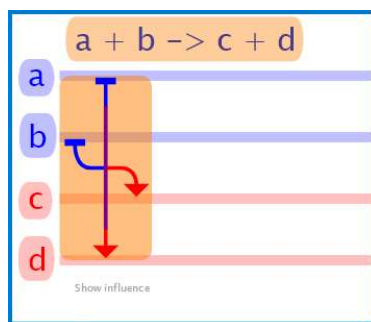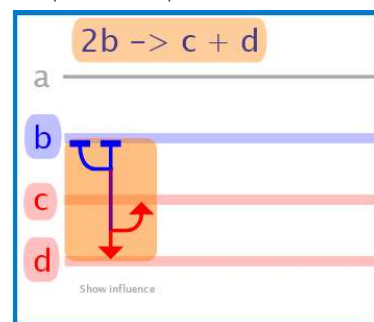
Reactants but no products

a -> Ø
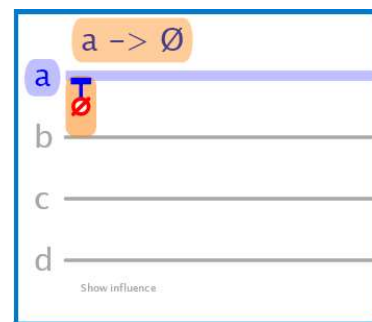
Products but no reactants

Ø -> a + b

Catalyst

b + a -> a + c
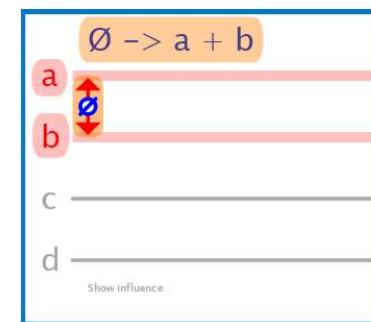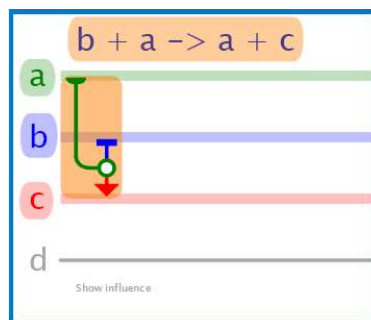
Catalyst but no reactants

a -> a + c
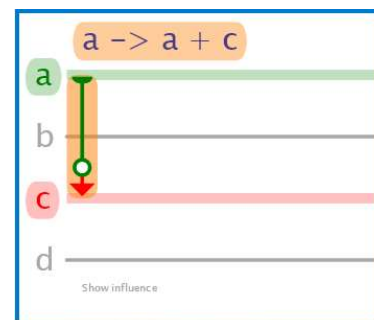
Catalyst but no products

a + c -> c

Autocatalyst

a -> 2a
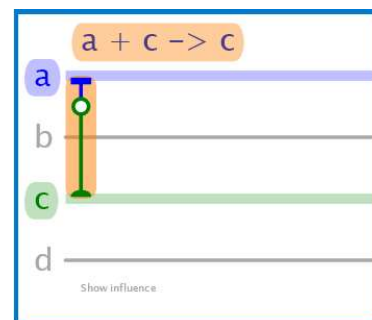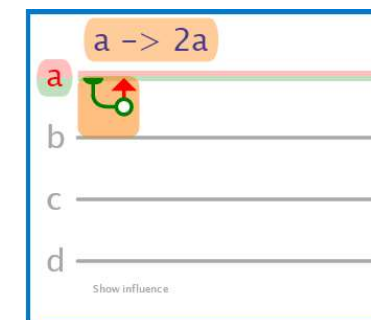
Reaction Scores vs. Reaction Graphs

- 2AM Oscillator

```
hi1 + md -> 2hi1
lo1 + hi1 -> lo1 + md
lo1 + md -> 2lo1
hi2 + lo1 -> hi2 + md {0.5}
hi2 + md -> hi2 + hi1 {0.5}
lo2 + hi1 -> lo2 + md {0.5}
lo2 + md -> lo2 + lo1 {0.5}
hi2 + lo2 -> hi2 + md»1
hi2 + md»1 -> 2hi2
lo2 + hi2 -> lo2 + md»1
lo2 + md»1 -> 2lo2
lo1 + lo2 -> lo1 + md»1 {0.5}
lo1 + md»1 -> lo1 + hi2 {0.5}
hi1 + hi2 -> hi1 + md»1 {0.5}
hi1 + md»1 -> hi1 + lo2 {0.5}
```


GraphViz

46
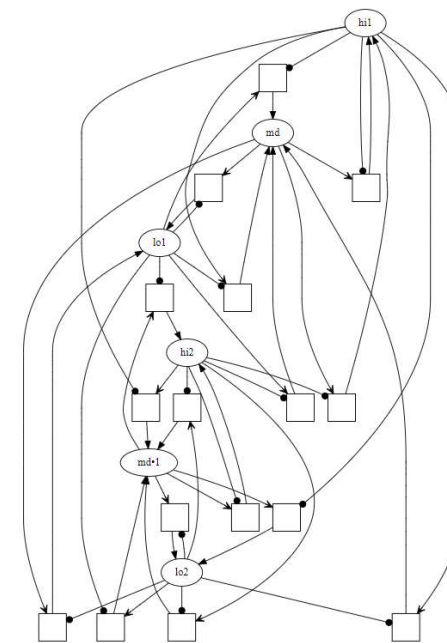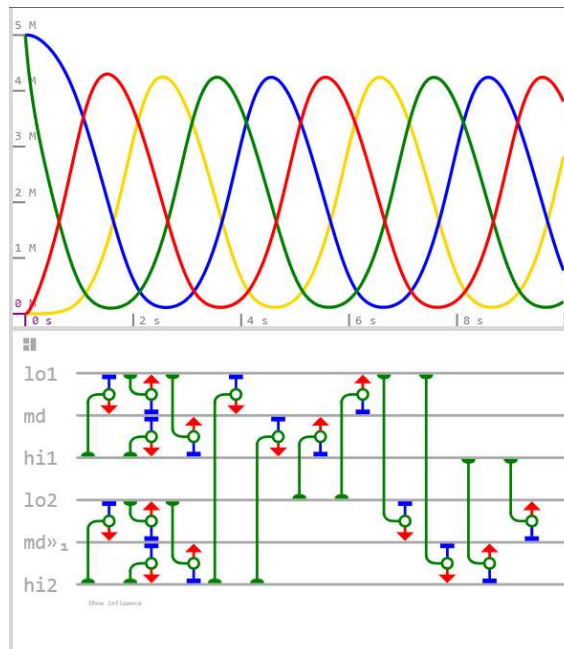
Stochastic (LNA) simulation

- For *all* programs (any CRN, any Protocol)

2AM Oscillator

$\partial lo1$ = - hi1 · lo1 - 0.5 · hi2 · lo1 + lo1 · md + 0.5 · lo2 · md
$\partial hi2$ = -0.5 · hi1 · hi2 - hi2 · lo2 + hi2 · md»$_1$ + 0.5 · lo1 · md»$_1$
$\partial lo2$ = 0.5 · hi1 · md»$_1$ - hi2 · lo2 - 0.5 · lo1 · lo2 + lo2 · md»$_1$
$\partial hi1$ = - hi1 · lo1 - 0.5 · hi1 · lo2 + hi1 · md + 0.5 · hi2 · md
∂md = 2 · hi1 · lo1 + 0.5 · hi1 · lo2 + 0.5 · hi2 · lo1 - hi1 · md - 0.5 · hi2 · md - lo1 · md - 0.5 · lo2 · md
$\partial md»_1$ = 0.5 · hi1 · hi2 - 0.5 · hi1 · md»$_1$ + 2 · hi2 · lo2 + 0.5 · lo1 · lo2 - hi2 · md»$_1$ - 0.5 · lo1 · md»$_1$ - lo2 · md»$_1$

$\partial var(lo1)$ = - cov(hi1,lo1) · lo1 - 0.5 · cov(hi2,lo1) · lo1 - cov(lo1,hi1) · lo1 - 0.5 · cov(lo1,hi2) · lo1 + cov(lo1,md) · lo1 + hi1 · lo1 + 0.5 · hi2 · lo1 + 0.5 · cov(lo1,md) · lo2 + cov(md,lo1) · lo1 + 0.5 · cov(md,lo1) · lo2 + 0.5 · cov(lo1,lo2) · md + 0.5 · cov(lo2,lo1) · md + lo1 · md + 0.5 · lo2 · md - 2 · hi1 · var(lo1) - hi2 · var(lo1) + 2 · md · var(lo1)

$\partial cov(lo1,hi2)$ = cov(lo1,md»$_1$) · hi2 - 0.5 · cov(lo1,hi1) · hi2 - cov(hi1,hi2) · lo1 - 1.5 · cov(lo1,hi2) · hi1 - 0.5 · cov(lo1,hi2) · hi2 - cov(lo1,lo2) · hi2 + 0.5 · cov(lo1,md»$_1$) · lo1 + cov(md,hi2) · lo1 - cov(lo1,hi2) · lo2 + 0.5 · cov(md,hi2) · lo2 + cov(lo1,hi2) · md + 0.5 · cov(lo2,hi2) · md + cov(lo1,hi2) · md»$_1$ - 0.5 · lo1 · var(hi2) + 0.5 · md»$_1$ · var(lo1)

$\partial cov(lo1,lo2)$ = 0.5 · cov(lo1,md»$_1$) · hi1 - cov(hi1,lo2) · lo1 - 0.5 · cov(hi2,lo2) · lo1 + cov(lo1,md»$_1$) · lo2 + cov(md,lo2) · lo1 + 0.5 · cov(md,lo2) · lo2 + 0.5 · cov(lo1,hi1) · md»$_1$ - 0.5 · cov(lo1,lo2) · lo1 - cov(lo1,hi2) · lo2 - cov(lo1,lo2) · hi1 - 1.5 · cov(lo1,lo2) · hi2 + cov(lo1,lo2) · md + cov(lo1,lo2) · md»$_1$ - 0.5 · lo2 · var(lo1) + 0.5 · md · var(lo2)

$\partial cov(lo1,hi1)$ = cov(lo1,md) · hi1 + 0.5 · cov(lo1,md) · hi2 - cov(lo1,hi1) · lo1 + cov(md,hi1) · lo1 - 0.5 · cov(lo1,hi1) · lo2 - 0.5 · cov(lo1,lo2) · hi1 - 0.5 · cov(hi2,hi1) · lo1 - cov(lo1,hi1) · hi1 - 0.5 · cov(lo1,hi1) · hi2 + 0.5 · cov(md,hi1) · lo2 + 2 · cov(lo1,hi1) · md + 0.5 · cov(lo1,hi2) · md + 0.5 · cov(lo2,hi1) · md - lo1 · var(hi1) - hi1 · var(lo1)

$\partial cov(lo1,md)$ = 2 · cov(lo1,hi1) · lo1 - cov(hi1,md) · lo1 - cov(lo1,md) · lo1 - hi1 · lo1 - 0.5 · hi2 · lo1 + 0.5 · cov(lo1,hi1) · lo2 - 0.5 · cov(lo1,md) · lo2 - cov(lo1,hi1) · md + 0.5 · cov(lo1,lo2) · hi1 - 0.5 · cov(hi2,md) · lo1 + 0.5 · cov(lo1,hi2) · md - 0.5 · cov(lo1,lo2) · md - 2 · cov(lo1,md) · hi1 - cov(lo1,md) · hi2 + cov(lo1,md) · md + 0.5 · cov(lo2,md) · md - lo1 · md - 0.5 · lo2 · md + 2 · hi1 · var(lo1) + 0.5 · hi2 · var(lo1) + lo1 · var(md) + 0.5 · lo2 · var(md) - md · var(lo1)

$\partial cov(lo1,md»_1)$ = 0.5 · cov(lo1,hi1) · hi2 - cov(hi1,md»$_1$) · lo1 - 0.5 · cov(lo1,md»$_1$) · lo1 - cov(lo1,md»$_1$) · lo2 + cov(md,md»$_1$) · lo1 + 0.5 · cov(md,md»$_1$) · lo2 - 0.5 · cov(lo1,hi1) · md»$_1$ + 0.5 · cov(lo1,hi2) · hi1 + 2 · cov(lo1,lo2) · hi2 - 0.5 · cov(hi2,md»$_1$) · lo1 + 0.5 · cov(lo1,lo2) · lo1 + 2 · cov(lo1,hi2) · lo2 - cov(lo1,lo2) · md»$_1$ + 0.5 · cov(lo2,md»$_1$) · md - cov(lo1,hi2) · md»$_1$ - 1.5 · cov(lo1,md»$_1$) · hi1 - 1.5 · cov(lo1,md»$_1$) · hi2 + cov(lo1,md»$_1$) · md + 0.5 · lo2 · var(lo1) - 0.5 · md»$_1$ · var(lo1)

...
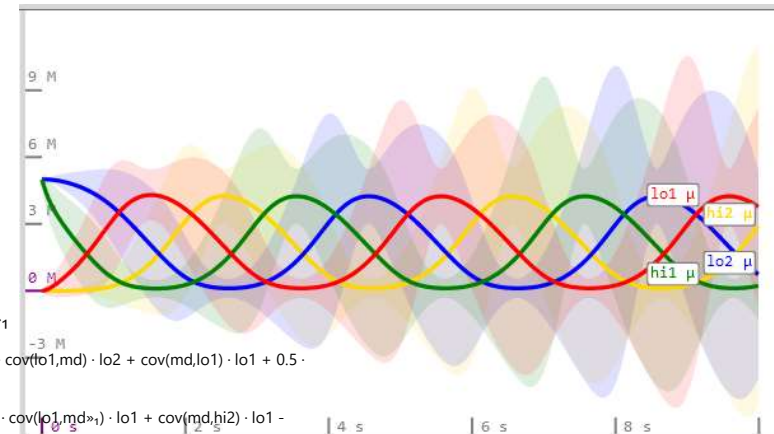
Writing Models Compositionally

- 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")

Ex: Predatorial

```
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
```


<= Demo: Predatorial

49
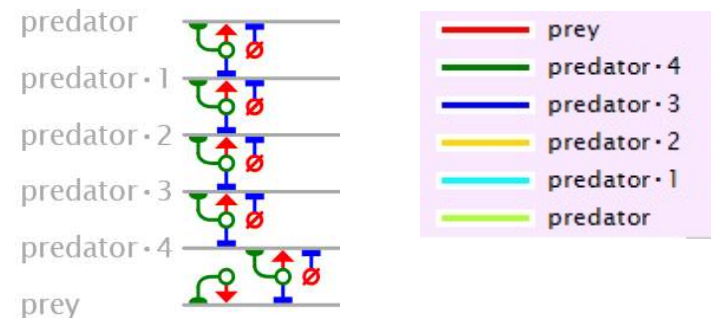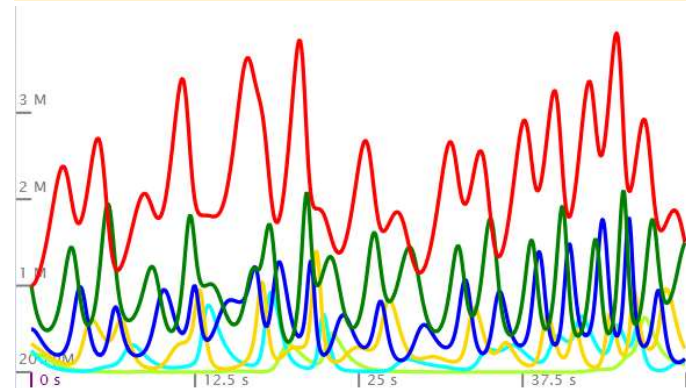
Describing a Protocol

- *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 = mol/L.
- The default implicit sample is called the **vessel** {1 mL, 20 C}

```
species {c} // a species for multiple samples

sample A {1µL, 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 {1µL, 20C}
species b @ 10mM in B    // species local to B
amount c @ 1mM in B      // amount of c in B
b + c -> c + c
```

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

species {NaCl#58.44}

sample C {1mL, 20C}
amount NaCl @ 8g in 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 2*e^(-5/(R*T))
```

<= Demo: MixAndSplit

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 37C
```

Change sample volume (concentrate or dilute)

```
concentrate A = B to 1mL
```

Alessandro Abate[2], Luca Cardelli[1,2], Marta Kwiatkowska[2], Luca Laurenti[2], and Boyan Yordanov[1]

[1] Microsoft Research Cambridge
[2] Department of Computer Science, University of Oxford

Demo: Sample Manipulation

```
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
```
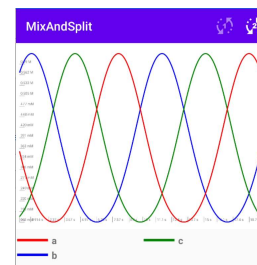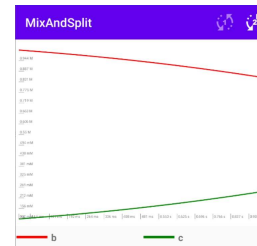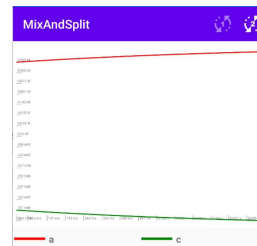

Multiple equilibration (simulation) steps

53

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 121C
        // bake
        equilibrate hot for t
        // decrease temperature, preserve volume:
        regulate PBS = hot to 20C
    yield PBS
}

function MakePBS() {
    define
        sample PBS {800mL, 20C}
        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, 20C}
        mix PBS = PBS,topup
    yield Autoclave(PBS, 20*60)
}

sample PBS = MakePBS()
```
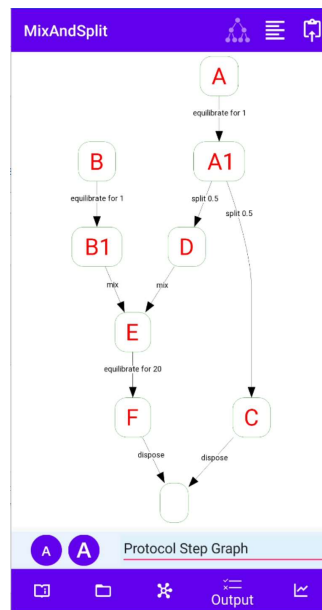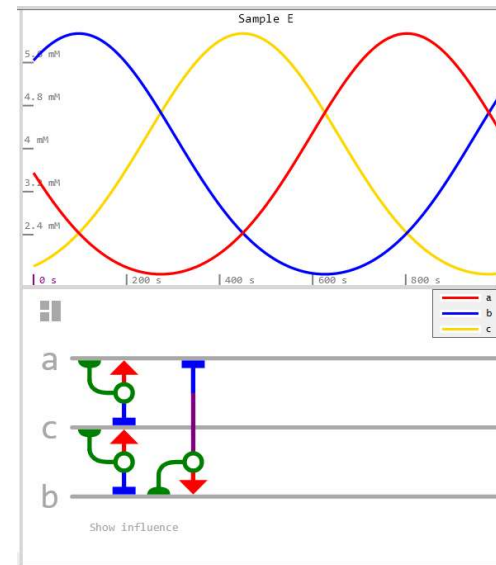
CSH Cold Spring Harbor Protocols

Recipe

Phosphate-buffered saline (PBS)

Reagent	Amount to add (for concentration 1× solution)	Final (1×)	Amount to add (for 10× stock)	Final concentration (10×)
NaCl	8 g	137 mM	80 g	1.37 M
KCl	0.2 g	2.7 mM	2 g	27 mM
Na_2HPO_4	1.44 g	10 mM	14.4 g	100 mM
KH_2PO_4	0.24 g	1.8 mM	2.4 g	18 mM
If necessary, PBS may be supplemented with the following:				
$CaCl_2 \cdot 2H_2O$	0.133 g	1 mM	1.33 g	10 mM
$MgCl_2 \cdot 6H_2O$	0.10 g	0.5 mM	1.0 g	5 mM

PBS can be made as a 1× solution or as a 10× stock. To prepare 1 L of either 1× or 10× PBS, dissolve the reagents listed above in 800 mL of H_2O. Adjust the pH to 7.4 (or 7.2, if required) with HCl, and then add H_2O to 1 L. Dispense the solution into aliquots and sterilize them by autoclaving for 20 min at 15 psi (1.05 kg/cm^2) on liquid cycle or by filter sterilization. Store PBS at room temperature.

54

http://cshprotocols.cshlp.org/content/2006/1/pdb.rec8247

Ex: Serial Dilution (recursive protocol)

```
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}

Digital Microfluidics Compiler

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

```
sample A {3µL, 20C}

split B,C,D,E = A

mix F = E,C,B,D

dispose F
```


< 20C   > 40C
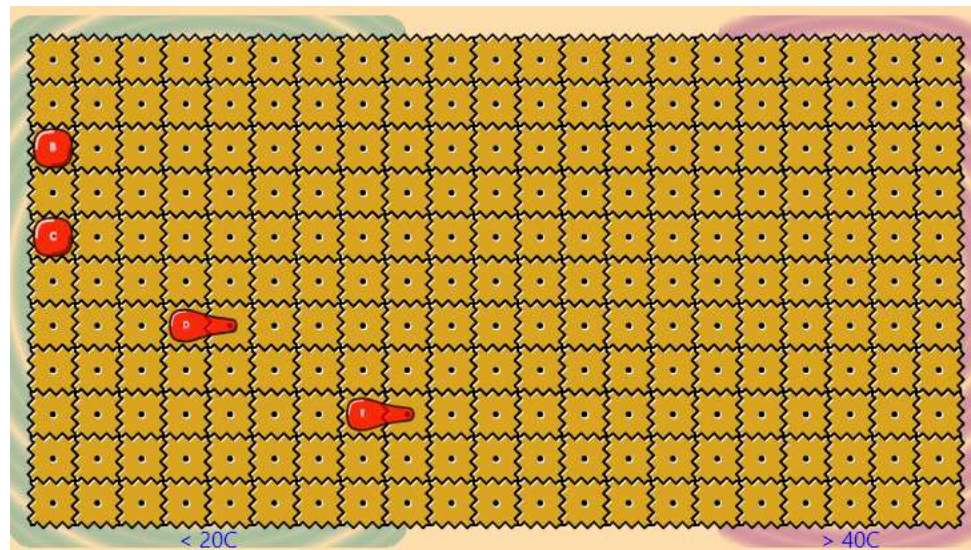
<= Demo: MixAndSplit
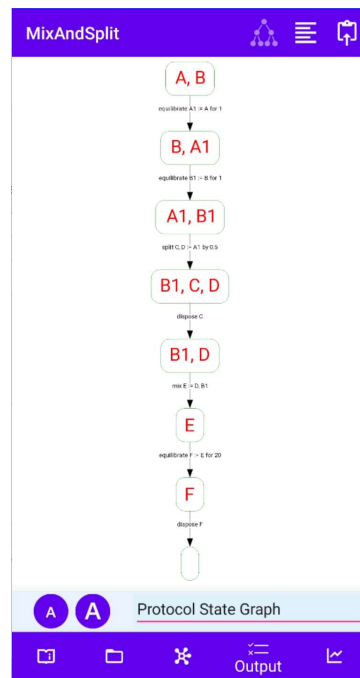
Extracting the Model and the Protocol

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

MixAndSplit

A

equilibrate for 1

B → A1

equilibrate for 1   split 0.5

split 0.5

B1   D

mix   mix

E

equilibrate for 20

F   C

dispose   dispose

A A   Protocol Step Graph

Output

The (final) model (sample E)

Sample E

```
STATE_5
sample E {1.5mL, 293.2K} {
    a = 354.5mM
    c = 178mM
    b = 0.5674M
    consumed
    a + c -> a + a
    b + c -> c + c
    a + b -> b + b
}

KINETICS for STATE_5 (sample E) for 20 time units:
∂a = a * c - a * b
∂c = c * b - a * c
∂b = a * b - c * b
```

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
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 full story (Hybrid system)

# Kaemika: Extra features

Extra features

- General kinetic rates
  - Fractions, rational powers, exponentials, trigonometry. E.g., x -> y {{ 1/x }}
  - Work with both deterministic and stochastic simulation and equation-extraction
  - Even triggers (discontinuous waveforms)

- Direct ODE notation
  - Instead of a reaction, just write an ODE like $\partial x = s \cdot y - s \cdot x$
  - This is translated to the reaction $\varnothing$ -> x {{s · y – s · x}} using general kinetic rates

- Timeflows (trajectories as first-class values)
  - Programmable plot reports (e.g., var(2 · a - 3 · b))
  - Capture timeflow outputs to combine (e.g., avg) and re-plot/export them later

- Mass action compiler
  - Turn *any* elementary ODE system (with fractions, rational powers, exponentials, trigonometry) into an equivalent system of pure mass action reactions.

- Programmable random numbers and distributions
  - As in MIT's Omega probabilistic language, with rejection sampling.

- Export
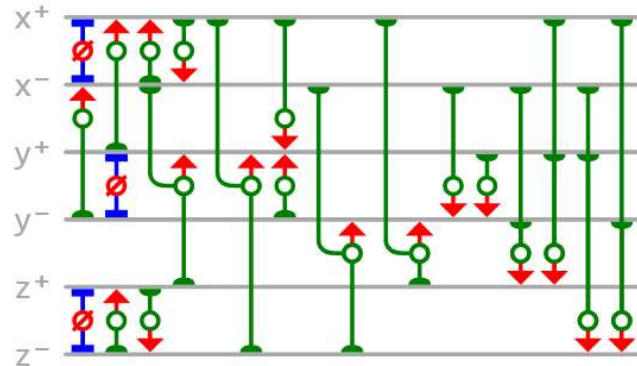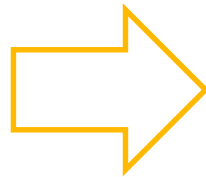 - SBML, ODE, Bitmap, SVG, GraphViz

60

Mass Action Compiler

- Lorenz chaotic attractor
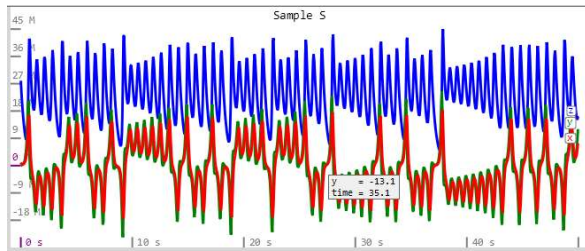
not mass action

$$\partial x = s \cdot y - s \cdot x$$
$$\partial y = r \cdot x - x \cdot z - y$$
$$\partial z = x \cdot y - b \cdot z$$

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

$x^+ + x^- \to \emptyset$
$y^+ \to y^+ + x^+$ {10}
$x^- \to x^- + x^+$ {10}
$y^- \to y^- + x^-$ {10}
$x^+ \to x^+ + x^-$ {10}
$y^+ + y^- \to \emptyset$
$z^+ + x^- \to z^+ + x^- + y^+$
$z^- + x^+ \to z^- + x^+ + y^+$
$x^+ \to x^+ + y^+$ {28}
$y^- \to y^- + y^+$
$z^- + x^- \to z^- + x^- + y^-$
$z^+ + x^+ \to z^+ + x^+ + y^-$
$x^- \to x^- + y^-$ {28}
$y^+ \to y^+ + y^-$
$z^+ + z^- \to \emptyset$
$y^- + x^- \to y^- + x^- + z^+$
$y^+ + x^+ \to y^+ + x^+ + z^+$
$z^- \to z^- + z^+$ {2.667}
$y^+ + x^- \to y^+ + x^- + z^-$
$y^- + x^+ \to y^- + x^+ + z^-$
$z^+ \to z^+ + z^-$ {2.667}

Initial:
x^+ =1
x^- = 0
y^+ = 0
y^- = 0
z^+ = 28
z^- = 0

<= Demo: LorenzAttractor

Global Sensitivity Analysis (of a Lotka-Volterra system)
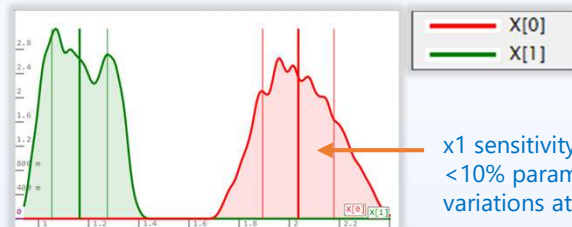
```
function f(number r1 r2 r3) {
  define
    sample S
    species x1 @ 0.66 M in S
    species x2 @ 0.44 M in S
    x1 -> x1 + x1          {r1}
    x1 + x2 -> x2 + x2     {r2}
    x2 -> Ø                {r3}
    equilibrate S for 2.5
  yield [observe(x1,S), observe(x2,S)]
}


random X(omega w) {
    f(1+(w(0)-0.5)/10, 1+(w(1)-0.5)/10, 1+(w(2)-0.5)/10)
}


draw 2000 from X
```

<- A function f to run one simulation (ri are the input parameters to be perturbed)
<- define D yield E returns the value of E after executing the statements D
<- Make a new sample S to contain species and reactions for simulation
<- Lotka-Volterra prey species x1 (initial conditions could be a parameter as well)
<- Lotka-Volterra predator species x2
<- Prey reproduces, with perturbed rate r1
<- Predator eats prey, with perturbed rate r2
<- Predator dies, with perturbed rate r3
<- Simulate the system up to time 2.5 (first peak of the oscillation)
<- Return the output concentrations of x1,x2 from S at time 2.5 as pairs

<- Create a bivariate random variable X over uniform[0..1) sample spaces w(i)
<- producing random instances f(1+e_1, 1+e_2, 1+e_3) = [x1,x2]$_{e_1,e_2,e_3,t=2.5}$
   with e_1, e_2, e_3 being 10% independent perturbations of the parameters

<- Produce a density plot of 2000 instances drawn from X
   i.e. a plot of the distributions of X[0]=x1 and X[1]=x2 at time 2.5
   vertical bars are mean and standard deviation

x1 sensitivity to random
<10% parameter
variations at time 2.5

N.B., consider also exporting your Kaemika model to SBML and
use the Sobol' method of global sensitivity analysis in e.g. Copasi.

Conclusions

Experimental biological protocols with formal semantics
Alessandro Abate, Luca Cardelli, Marta Kwiatkowska,
Luca Laurenti, Boyan Yordanov. CMSB 2018.

Kaemika app - Integrating protocols and chemical simulation
Luca Cardelli. CMSB 2020.

Kaemika User Manual
http://lucacardelli.name/Papers/Kaemika%20User%20Manual.pdf

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