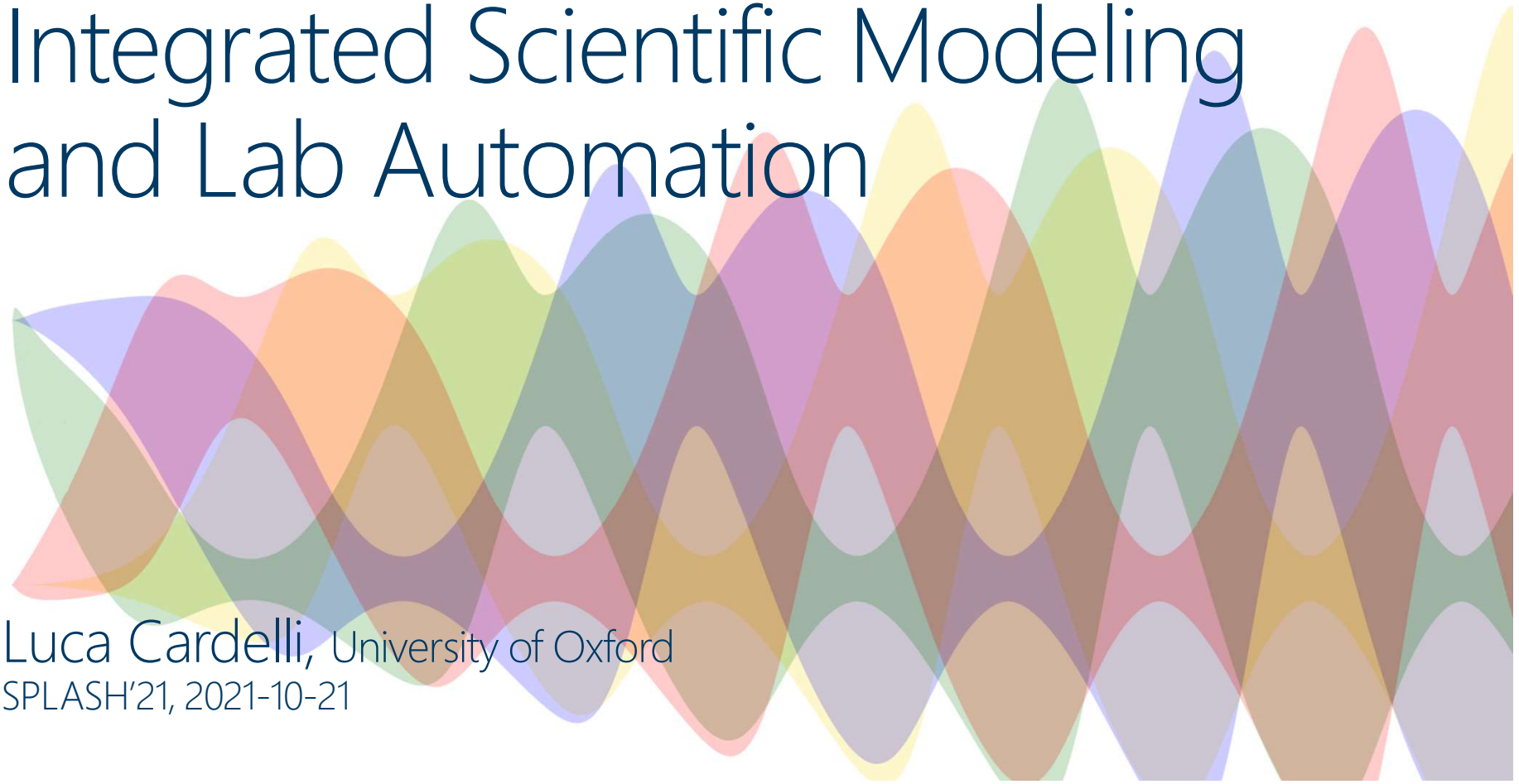
Integrated Scientific Modeling and Lab Automation



Luca Cardelli, University of Oxford

SPLASH'21, 2021-10-21

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

CMSB'2020 Best Tool Paper Award

Kaemika

/'kimika/

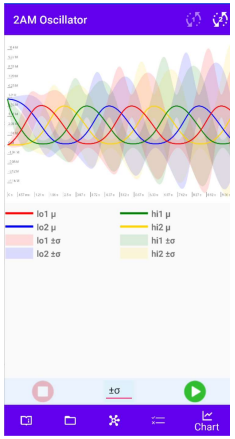
macOS

iOS

Android

Windows

GitHub



Search "Kaemika" in the app stores

http://lucacardelli.name/kaemika.html

computation

MDPI

Article

A Language for Modeling and Optimizing Experimental Biological Protocols

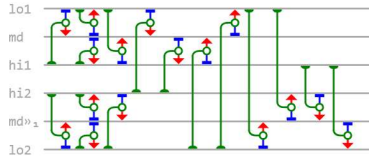
Luca Cardelli *, Marta Kwiatkowska and Luca Laurenti †

An integrated language for chemical models & experimental protocols

Deterministic (ODE) and stochastic (LNA) simulation

Chemical reaction networks (CRNs) and liquid-handling protocols

Reaction scores



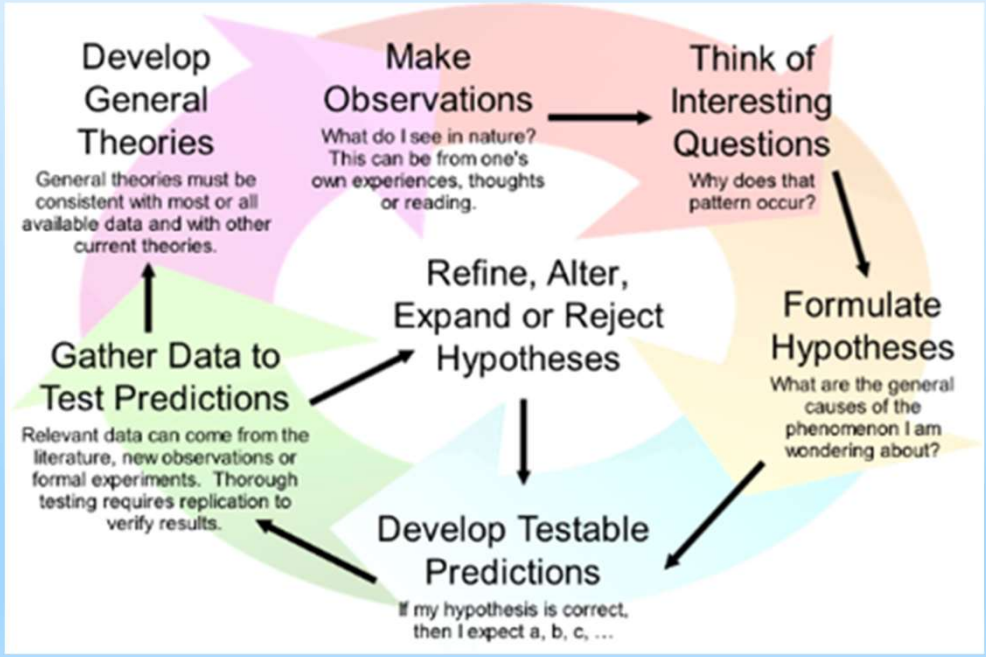
Functional scripting

GUI

The Scientific Method

Ḥasan Ibn al-Haytham (1027) Book of Optics

Galileo Galilei (1638) Two New Sciences

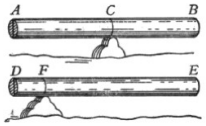


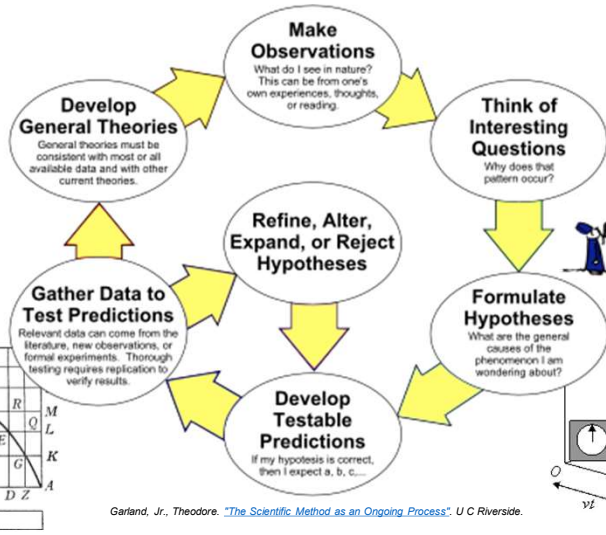
Discovery through Observation

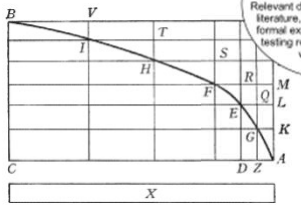
The Scientific Method ~ 1638

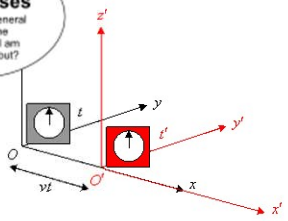
1 Person


DISCORSI
E
DIMOSTRAZIONI
MATEMATICHE,
intorno à due nuoue scienze
Attenenti alla
MECANICA & i MOVIMENTI LOCALI,
del Signor
GALILEO GALILEI LINCEO,
Filosofo e Matematico primario del Serenissimo
Gran Duca di Toscana.
IN LEIDA,
Appresso gli Elsevirii. a. d. c. xxxviii.









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

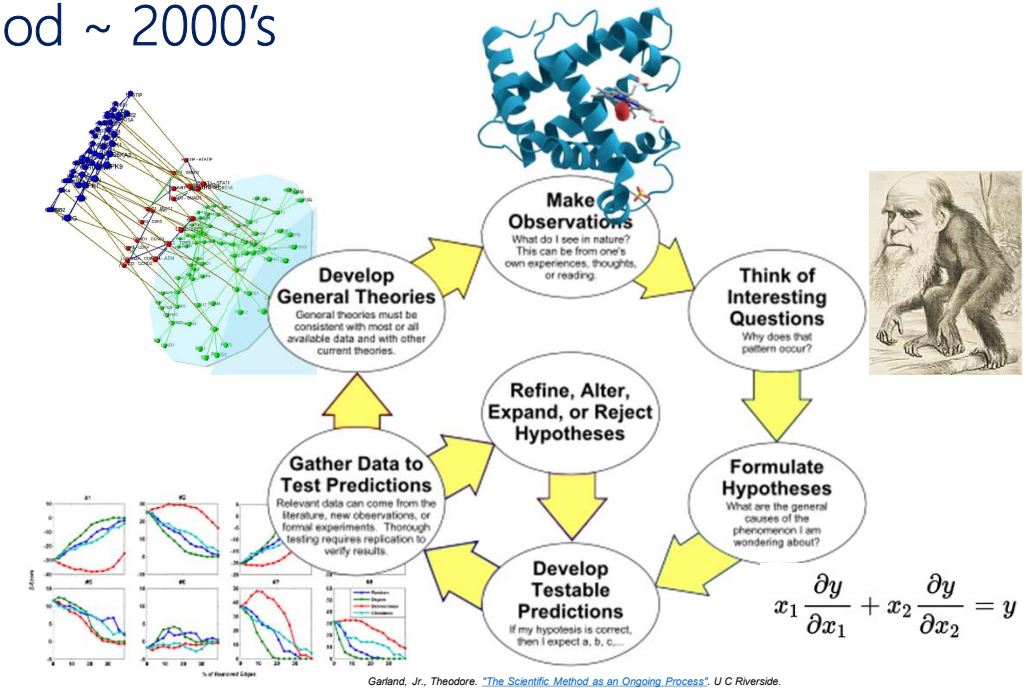
Discovery through Collaboration

The Scientific Method ~ 2000's

1 Lab

1 protein = 30 people / 30 years

Humans have >250,000 proteins ☹



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

Discovery through Automation

The Scientific Method ~ 2020's


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

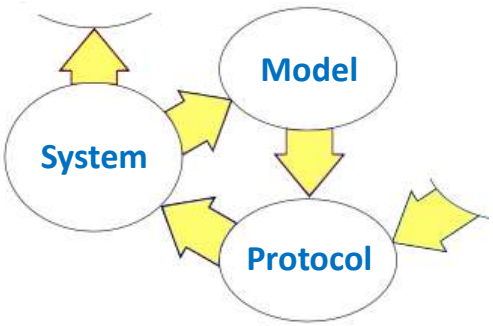
Garland, Jr., Theodore. "The Scientific Method as an Ongoing Process". U C Riverside.

Robot scientist becomes first machine to discover new scientific knowledge

Ross King

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

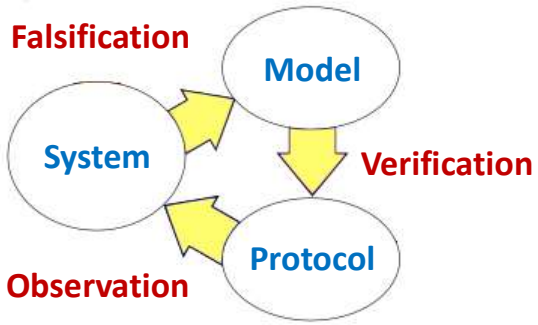
    http://www.nature.com/news/reproducibility-1.17552

The Inner Loop

- Tomorrow, automation

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

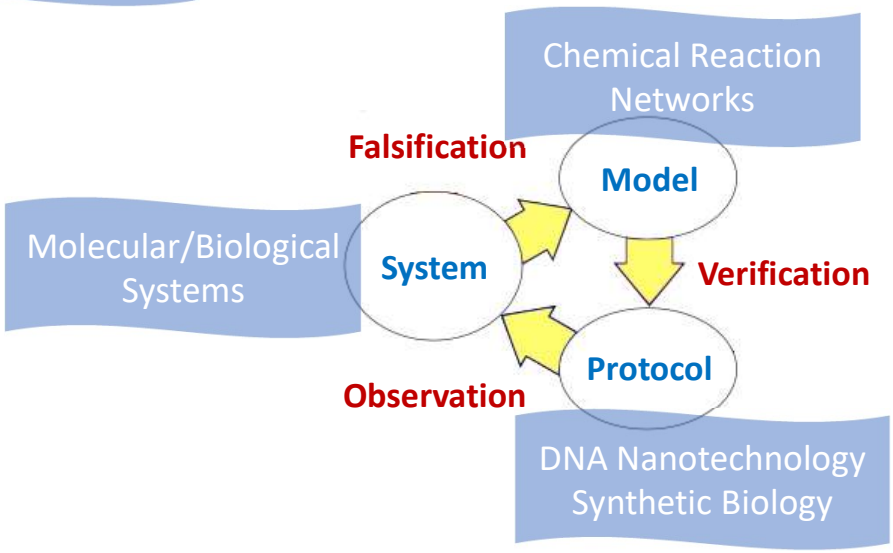
Lifecycle

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

The Inner Loop

In this talk

- A specific domain
- Aiming for closed-loop automated modelling and experimentation
- Via Molecular Programming



Models

(those things 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).

It turns out that in order to "*implement differential equations*" we need to "*implement chemical reactions*" anyway (or some other physical realization).

Chemical Reaction Networks (CRN)

$$X + Y \xrightarrow{r} Z + W$$

- 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*

Programming Examples

spec	*program*
Y := 2X	X -> Y + Y
Y := ⌊X/2⌋	X + X -> Y
Y := X1 + X2	X1 -> Y X2 -> Y
Y := min(X1, X2)	X1 + X2 -> Y

Advanced Programming Examples

spec

Y := max(X1, X2)

program

X1 -> L1 + Y
X2 -> L2 + Y
L1 + L2 -> K
Y + K -> 0

max(X1,X2)=
(X1+X2)-min(X1,X2)

(but is not computed "sequentially")

Approximate Majority

(X,Y) :=
if X≥Y then (X+Y, 0)
if Y≥X then (0, X+Y)

X + Y -> Y + B
Y + X -> X + B
B + X -> X + X
B + Y -> Y + Y

Programming *any* algorithm as a CRN

"approximately"

A CRN is a *finite* set of reactions over a *finite* set of species

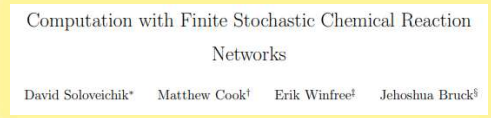
CRNs are not Turing complete

Like Petri nets: reachability is decidable

But unlike Petri nets, CRNs are *approximately* Turing complete

Because reactions have also *rates*

This make it possible to approximate Turing completeness by approximating test-for-zero in a register machine.
The probability of error (in test-for-zero) can be made arbitrarily small over the entire (undecidably long) computation.


Computation with Finite Stochastic Chemical Reaction Networks

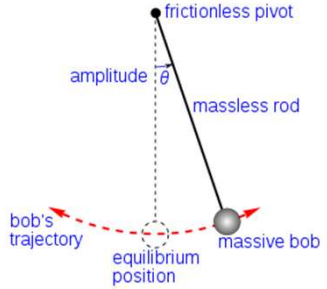
David Soloveichik* Matthew Cook† Erik Winfree‡ Jehoshua Bruck§

Adding polymerization to CRNs makes them fully Turing complete


Formal Molecular Biology

Vincent Danos* Cosimo Laneve†

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



https://en.wikipedia.org/wiki/Pendulum

Galileo Galilei 1602
Christiaan Huygens 1673

$$\partial^2\theta = -g/l \sin\theta$$

Equation of motion of the simple pendulum

A *dynamical systems* is anything characterized by a system of differential equations (ODEs).

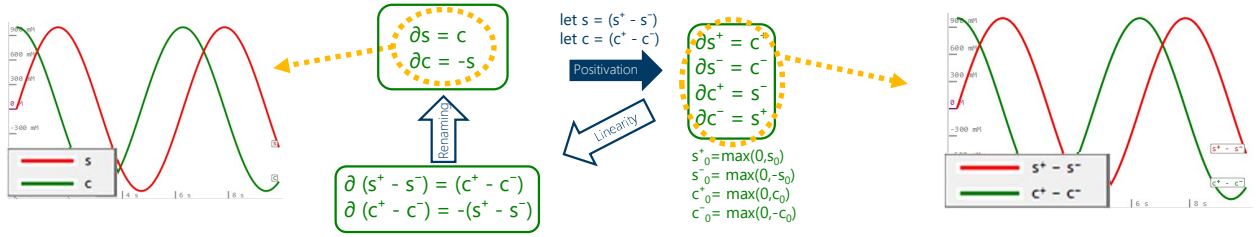
Elementary dynamical systems are those that include on the r.h.s. only polynomials, trigonometry, exponentials, fractions, and their inverses.
(All of biochemistry, all of electronics, most of physics.)

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

Abstraction of Elementary Hybrid Systems by Variable Transformation
Jiang Liu[1], Naijun Zhan[2], Hengjun Zhao[1], and Liang Zou[2]

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

Consider *the* canonical polynomial oscillator: sine/cosine



A very simple elementary ODE system.

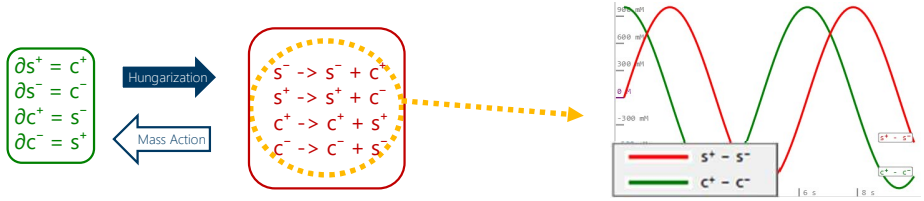
But variables go negative: we can't have that in a CRN (no negative concentrations).

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

Biomolecular implementation of linear I/O systems
K. Oishi E. Klavins

Programming *any* "elementary" 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.

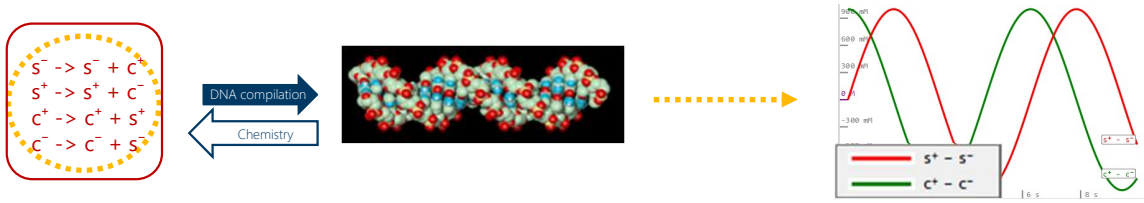
ON THE INVERSE PROBLEM OF REACTION KINETICS

V. HÁRS - J. TÓTH

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

Programming *any* "elementary" 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.

DNA as a universal substrate for chemical kinetics

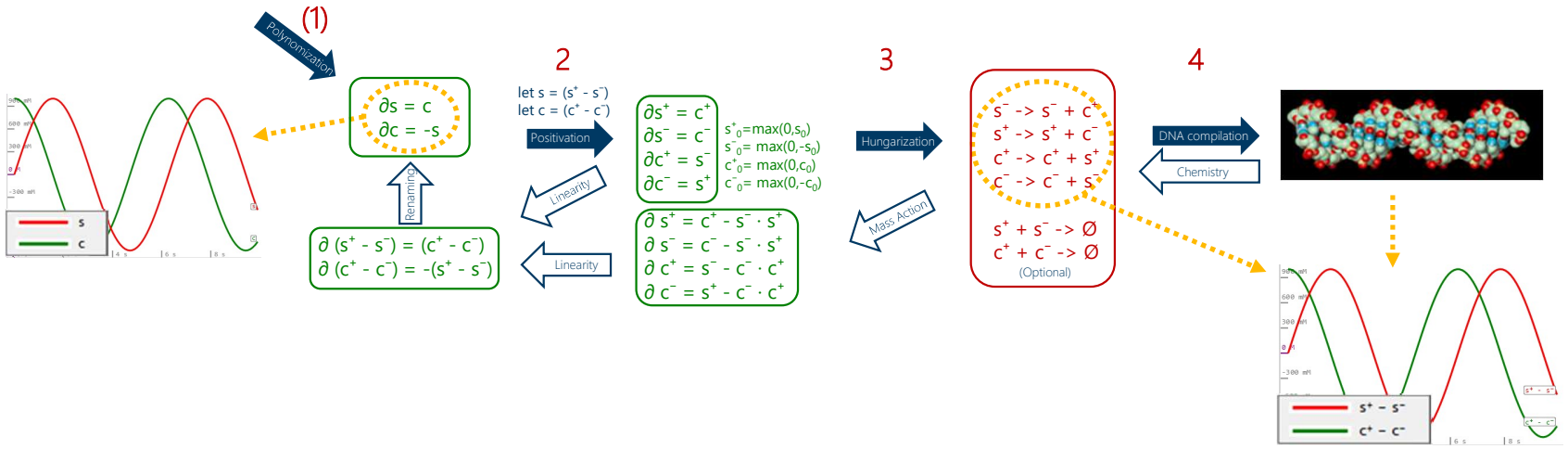
David Soloveichik, Georg Seelig, and Erik Winfree

PNAS March 23, 2010 107 (12) 5393-5398; https://doi.org/10.1073/pnas.0909380107

Works up to an arbitrarily good approximation of Mass Action kinetics, and up to time rescaling.

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

Thus, CNRs are "Shannon complete", and can by physically realized



MATHEMATICAL THEORY OF THE DIFFERENTIAL ANALYZER

By Claude E. Shannon

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* $F(V,T) \in \mathbb{R}_{\geq 0}^{|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{A}|}$ *be the flux of the CRN at volume* $V \in \mathbb{R}_{>0}$ *and temperature* $T \in \mathbb{R}_{>0}$*. For a concentration vector* $\mu \in \mathbb{R}_{\geq 0}^{|\mathcal{A}|}$ *we assume* $F(V,T)(\mu) = \sum_{\tau \in \mathcal{R}} v_\tau \alpha_\tau(V,T,\mu)$*, with stoichiometric vector* v_τ *and rate function* α_τ*.*

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

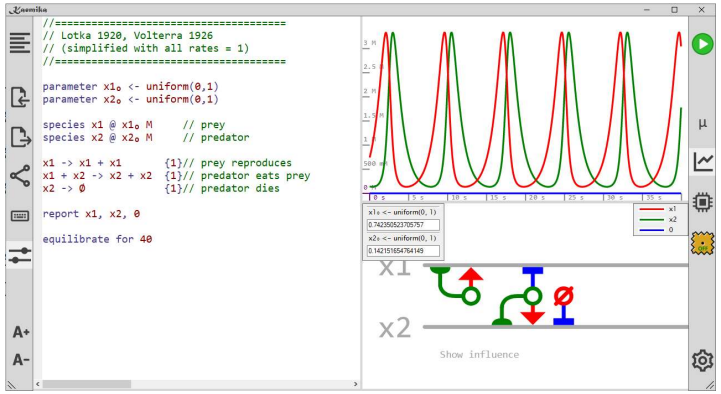
State produced by a CRN $\mathcal{C} = (\mathcal{A}, \mathcal{R})$ (species $\mathcal{A}$, reactions $\mathcal{R}$)
with flux F (r.h.s. of its mass action ODEs) at time *t*,
from initial state (x_0, V, T) (initial concentrations x_0, volume *V*, temperature *T*):

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

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

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



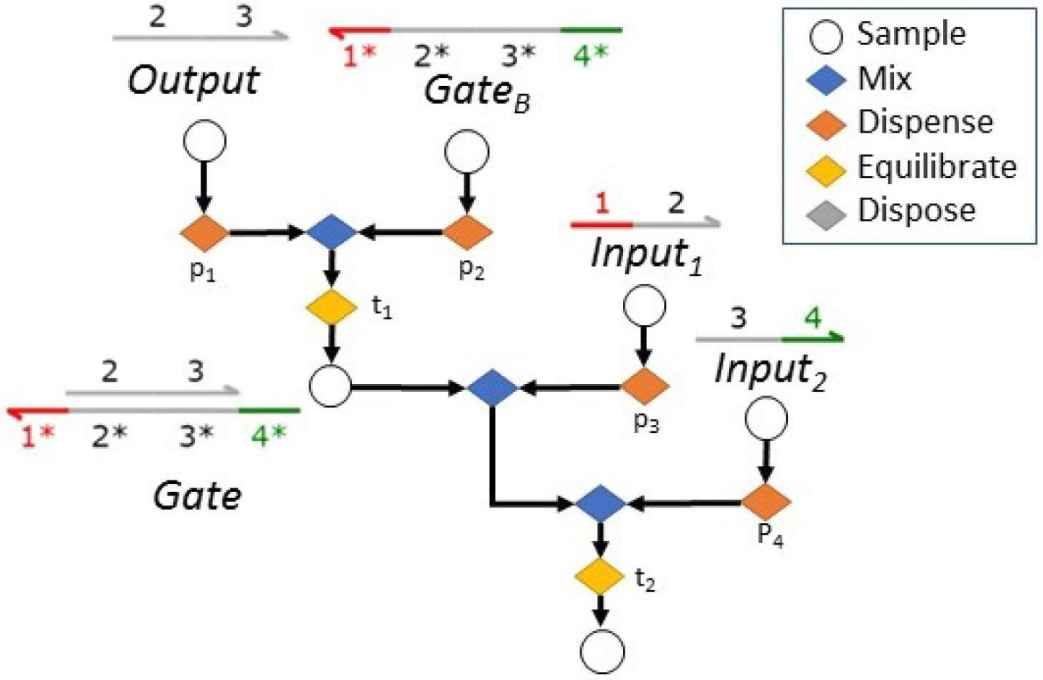
- Recall: we are aiming for models that can be placed into a closed-loop automated model+protocol cycle.

Protocols

(those things that know nothing about models)

A Protocol

For DNA gate assembly and activation in vitro



Protocol steps (liquid handing)

Digital Microfluidics

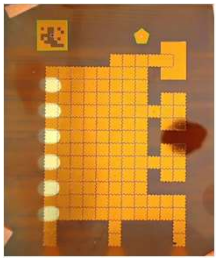
Manipulating droplets by electrical fields

OpenDrop

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

OpenDrop speed test

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



Purple Drop (UW)

https://misl.cs.washington.edu/projects/fluidics.html

Digital Microfluidics

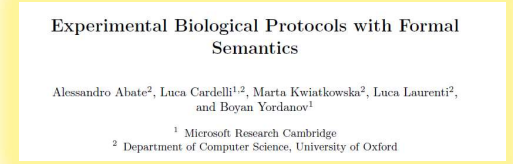
- A general, *programmable*, platform to execute the main liquid-handling operations

- 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.

A Protocol Language

Samples: containers with volume, temperature, concentrations

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



Experimental Biological Protocols with Formal Semantics

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:

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

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

(*Equilibrate semantics*)

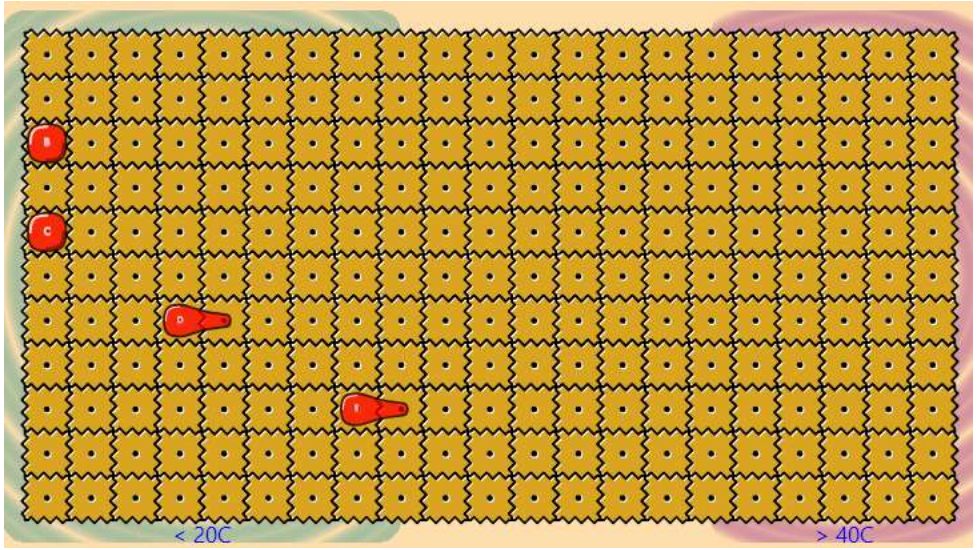
$$[\![Dispose(P)]\!]^\rho = (0^{|\Lambda|}, 0, 0),$$

(*CRN semantics*)

Kaemika Microfluidics Compiler

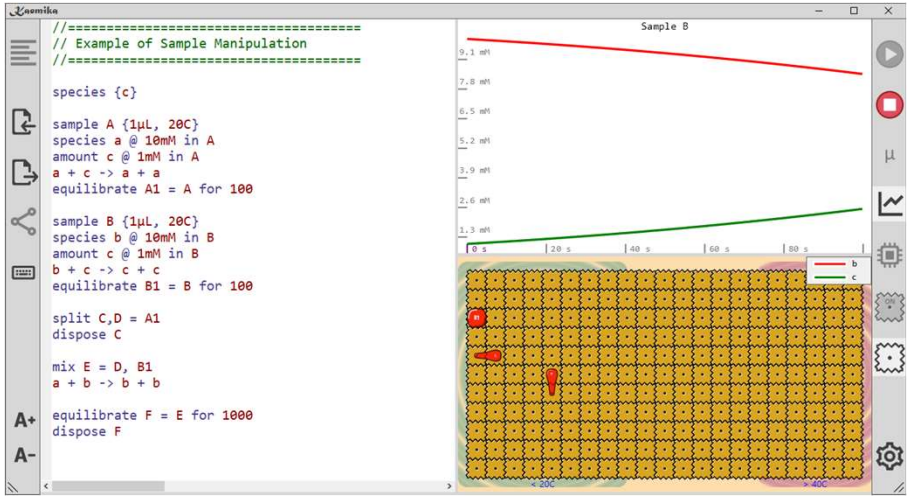
- Mix, split, equilibrate, dispose
- Automatic routing – no geometrical information
- Hot/cold zones

```
sample A {3µL, 20C}

split B,C,D,E = A

mix F = E,C,B,D

dispose F
```



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

An Integrated Description

Samples: containers with volume, temperature, concentrations

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

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

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

Protocol + Model =

Joint script

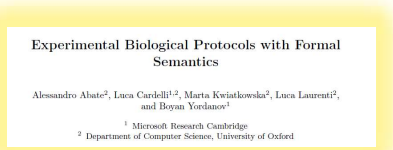
$Input_1 =< 1^*\ 2 >\quad Output =< 2\ 3 >$

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

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

Program Semantics (deterministic)

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


Experimental Biological Protocols with Formal Semantics

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

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

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

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

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

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

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

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

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.

Program Semantics (stochastic)

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


computation

MDPI

Article

A Language for Modeling and Optimizing Experimental Biological Protocols

Luca Cardelli *, Marta Kwiatkowska and Luca Laurenti †

A Language for Modeling and Optimizing Experimental Biological Protocols

Luca Cardelli *, Marta Kwiatkowska and Luca Laurenti †

Definition 3. *(CRN Flux) Let $(\mathcal{A}, \mathcal{R})$ be a CRN. Let $F(V,T) \in \mathbb{R}_{\geq 0}^{|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{A}|}$ 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 $\mu \in \mathbb{R}_{\geq 0}^{|\mathcal{A}|}$ we assume $F(V,T)(\mu) = \sum_{\tau \in \mathcal{R}} v_\tau \alpha_\tau(V,T,\mu)$, with stoichiometric vector v_τ and rate function α_τ. We call J_F the Jacobian of $F(V,T)$, and $J_F^\top$ its transpose. Further, define $W(V,T)(\mu) = \sum_{\tau \in \mathcal{R}} v_\tau v_\tau^\top \alpha_\tau(V,T,\mu)$ to be the diffusion term.*

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

$$\boldsymbol{\mu}_\mu(t) = \mu + \int_0^t F(V,T)(\boldsymbol{\mu}_\mu(s))ds \quad (1)$$

$$\boldsymbol{\Sigma}_{\mu,\Sigma}(t) = \Sigma + \int_0^t J_F(\boldsymbol{\mu}_\mu(s))\boldsymbol{\Sigma}_{\mu,\Sigma}(s) + \boldsymbol{\Sigma}_{\mu,\Sigma}(s)J_F^\top(\boldsymbol{\mu}_\mu(s)) + W(V,T)(\boldsymbol{\mu}_\mu(s))ds, \quad (2)$$

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

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

$$[\![(p_1 \ldots p_{|\mathcal{A}|}, r_V, r_T)]\!]^\rho = ([\![p_1]\!]^\rho \ldots [\![p_{|\mathcal{A}|}]\!]^\rho, 0^{|\mathcal{A}| \times |\mathcal{A}|}, r_V, r_T)$$

$$[\![let\ x = P_1\ in\ P_2]\!]^\rho = [\![P_2]\!]^{\rho_1}$$
$$where \quad \rho_1 = \rho\{x \leftarrow [\![P_1]\!]^\rho\}$$

$$[\![Mix(P_1,P_2)]\!]^\rho = \left(\frac{V_1\mu_1 + V_2\mu_2}{V_1+V_2}, \frac{V_1^2\Sigma_1 + V_2^2\Sigma_2}{(V_1+V_2)^2}, V_1+V_2, \frac{V_1T_1 + V_2T_2}{V_1+V_2}\right)$$
$$where \quad (\mu_1,\Sigma_1,V_1,T_1) = [\![P_1]\!]^\rho \quad and \quad (\mu_2,\Sigma_2,V_2,T_2) = [\![P_2]\!]^\rho$$

$$[\![let\ x,y = Split(P_1,p)\ in\ P_2]\!]^\rho = [\![P_2]\!]^{\rho_1}$$
$$where \quad r = [\![p]\!]^\rho, \quad 0 < r < 1 \quad and \quad (\mu,\Sigma,V,T) = [\![P_1]\!]^\rho$$
$$and \quad \rho_1 = \rho\{x \leftarrow (\mu,\Sigma,rV,T), y \leftarrow (\mu,\Sigma,(1-r)V,T)\}$$

$$[\![Equilibrate(P,p)]\!]^\rho = (\boldsymbol{\mu}_\mu(t), \boldsymbol{\Sigma}_{\mu,\Sigma}(t), V, T)$$
$$where \quad t = [\![p]\!]^\rho \quad and \quad (\mu,\Sigma,V,T) = [\![P]\!]^\rho$$

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

together with $[\![p]\!]^\rho$ defined as:

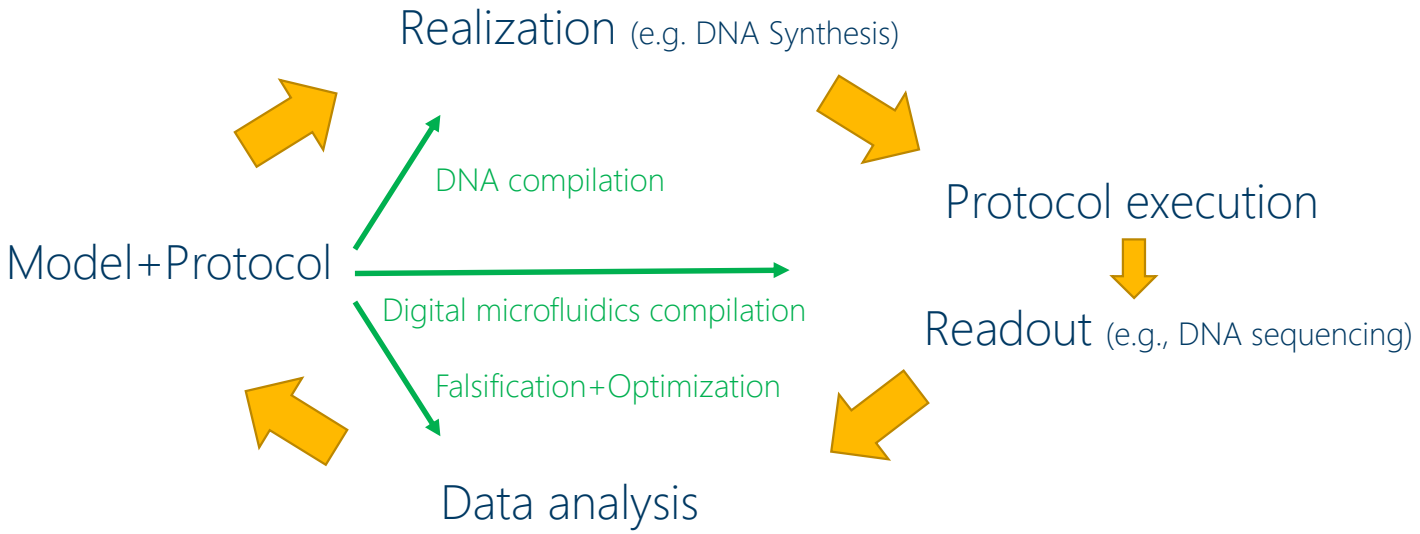
$$[\![z]\!]^\rho = \rho(z)$$
$$[\![r]\!]^\rho = r$$

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 Modelchecking to estimate the probability that the output will fall in a certain range, given the distributions over uncertain model and protocol parameters.

Summarizing

Automated discovery loop:



Simulating Reaction Networks together with Digital Protocols

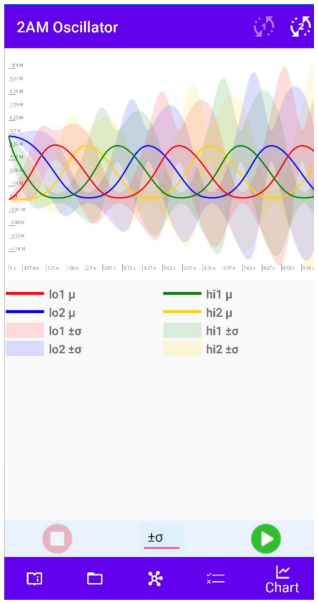
Kaemika

- A prototype language for chemical models & protocols

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

- Search "Kaemika" in the App stores

macOS
iOS
Android
Windows
GitHub



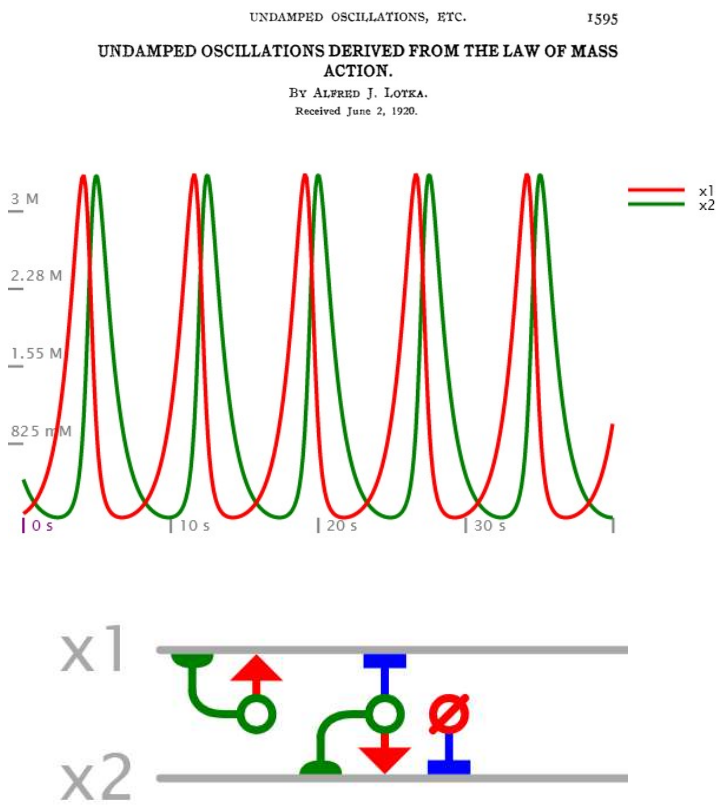
- CRN simulation
- Microfluidics simulation
- Reaction graphs
- ODE equations
- Stochastic noise (LNA)

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

equilibrate for 40
```



<= Demo: LotkaVolterra

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***

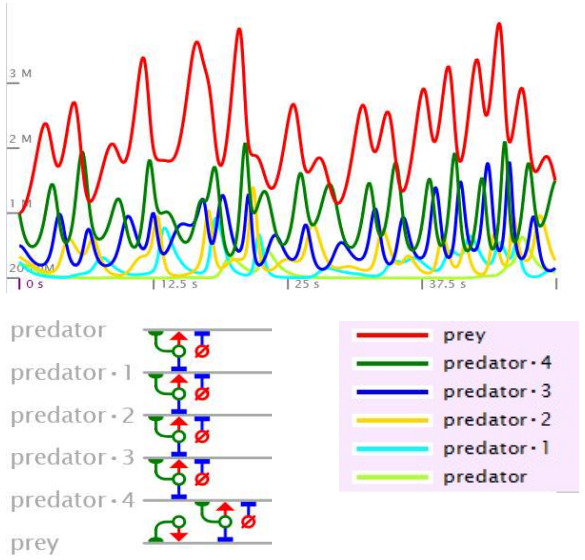
- Modern abstractions

  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 (recursive model)

```
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 -> ∅ // 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

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)

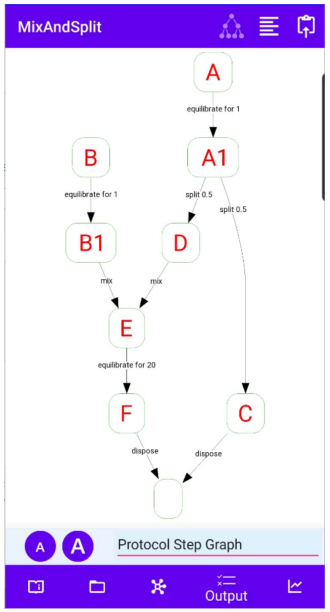
```
RESULT:
sample init {1mL, 298.2K} {A = 1M, B = 1M}
sample s2 {1mL, 298.2K} {A = 100mM, B = 100mM}
sample s4 {1mL, 298.2K} {A = 10mM, B = 10mM}
sample s7 {1mL, 298.2K} {A = 1mM, B = 1mM}
sample s10 {1mL, 298.2K} {A = 100uM, B = 100uM}
```

Extracting the Model and the Protocol

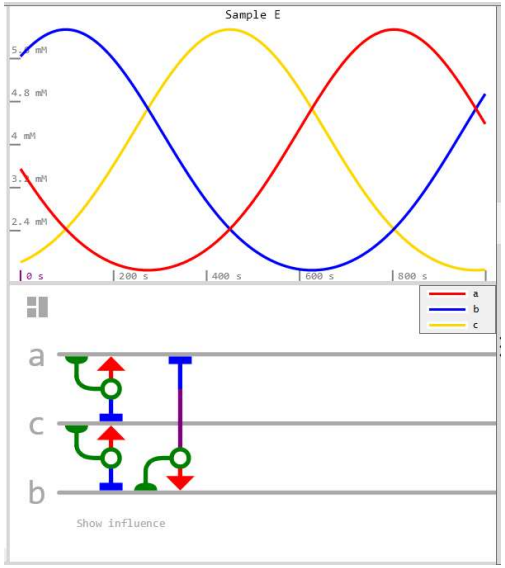
From the script

```
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)



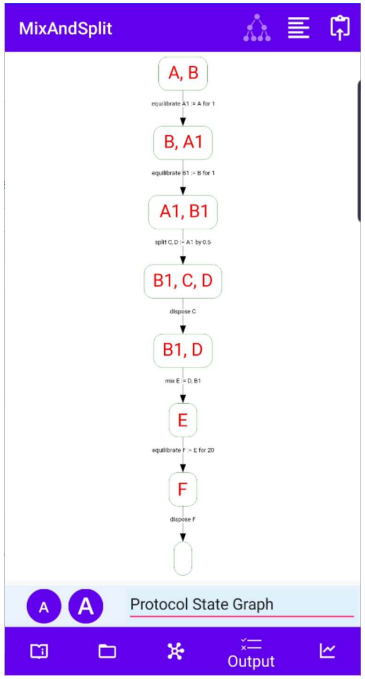

```
STATE_5
sample E {1.5mL, 293.2K} {
   a = 354.5mM
   c = 178mM
   b = 0.5674M
   consumed
   a + c -> a + a
   b + c -> c + c
   a + b -> b + b
}

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
```

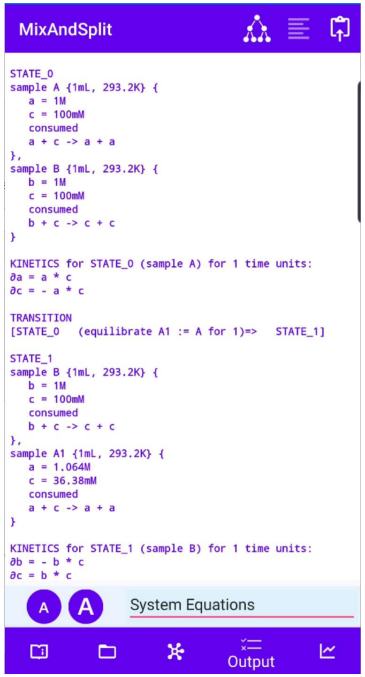
Extracting the Hybrid Transition System

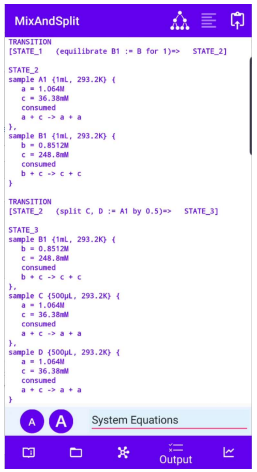
From the script

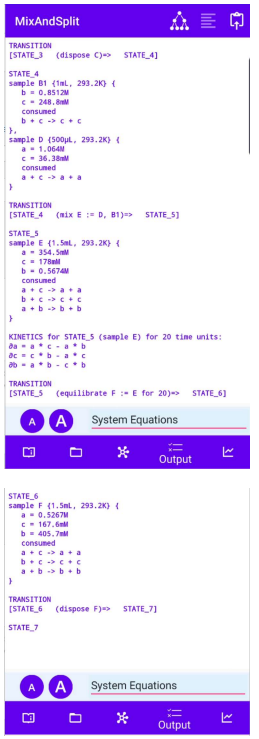
```
species {c}

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)





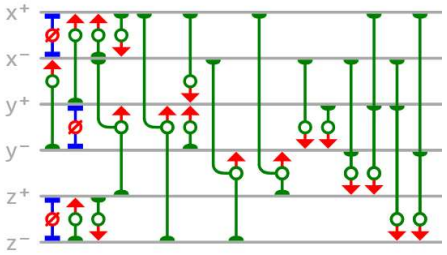




Extra features

- General kinetic rates
  - Fractions, rational powers, exponentials, trigonometry. E.g., x -> y {{ 1/x }}
  - Work with both deterministic and stochastic simulation and equation-extraction
  - Event triggers (discontinuous waveforms)
- Direct ODE notation
  - Instead of a reaction, just write an ODE like ∂x = s · y - s · x
  - This is translated to the reaction ∅ -> 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

- Reaction scores
  - A new representation of directed hypergraphs with catalysis



Conclusions

A Language for Modeling and Optimizing Experimental Biological Protocols
Luca Cardelli, Marta Kwiatkowska, Luca Laurenti. MDPI Computation 2021.

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

Thanks to:
- Gold (parser generator)
- OSLO (ODE simulator)
- C#/Xamarin (IDE)
- App store reviewers

NO thanks to:
- XAML (uber obfuscator)
- App store certificates
- Dark mode support