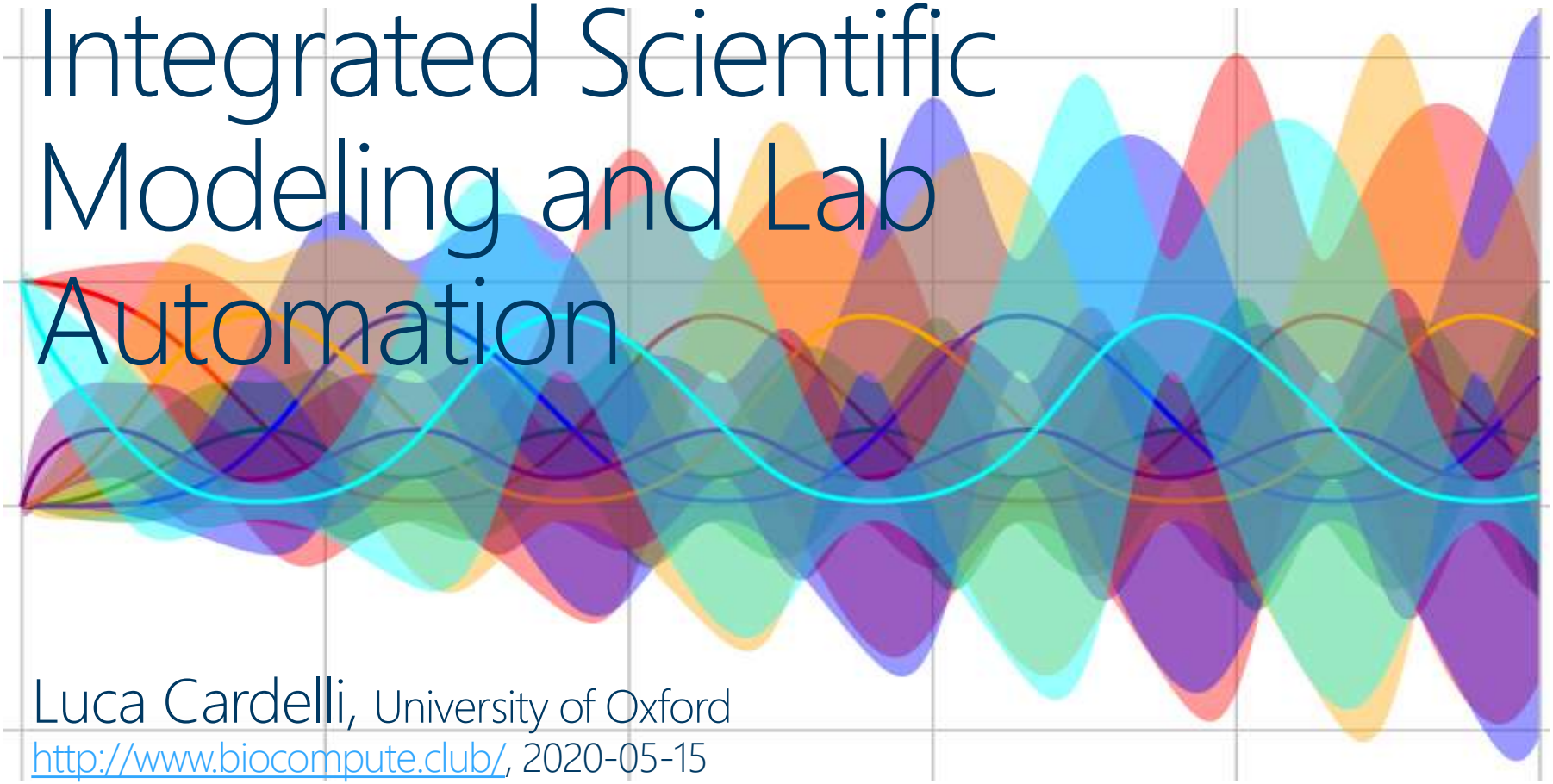
Integrated Scientific Modeling and Lab Automation



Luca Cardelli, University of Oxford

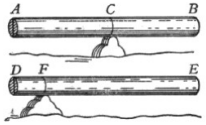
http://www.biocompute.club/, 2020-05-15

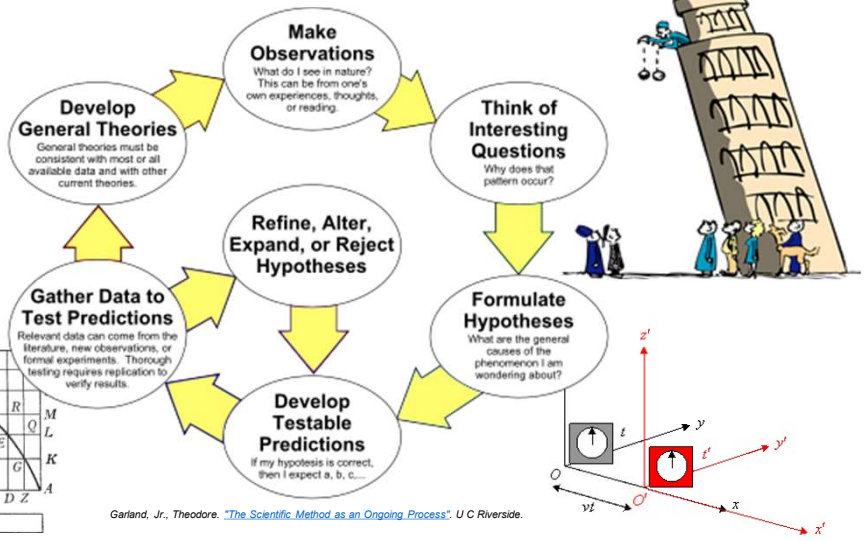
Discovery through Observation

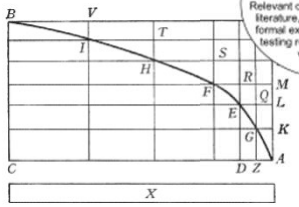
The Scientific Method ~ 1638

1 Guy


DISCORSI
E
DIMOSTRAZIONI
MATEMATICHE,
Mecanica & i Movimenti Locali,
GALILEO GALILEI LINCEO,
IN LEIDA.







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

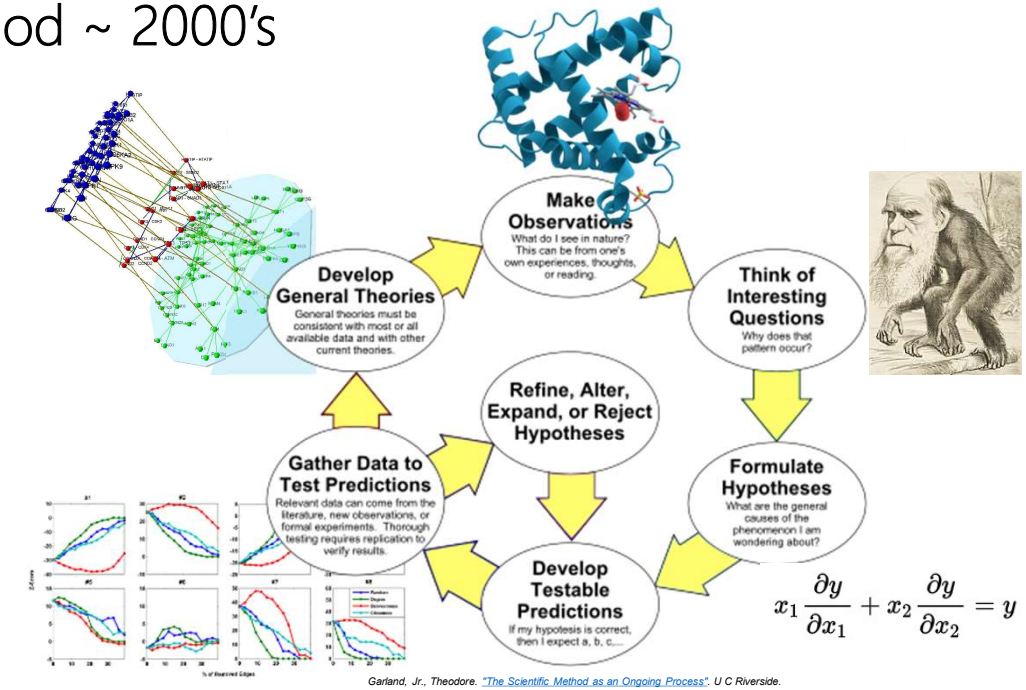
Discovery through Collaboration

The Scientific Method ~ 2000's

1 Lab

1 protein = 30 people / 30 years

Humans have >250,000 proteins ☹

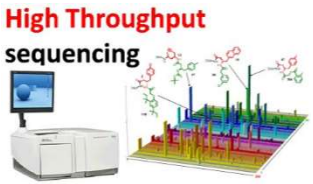


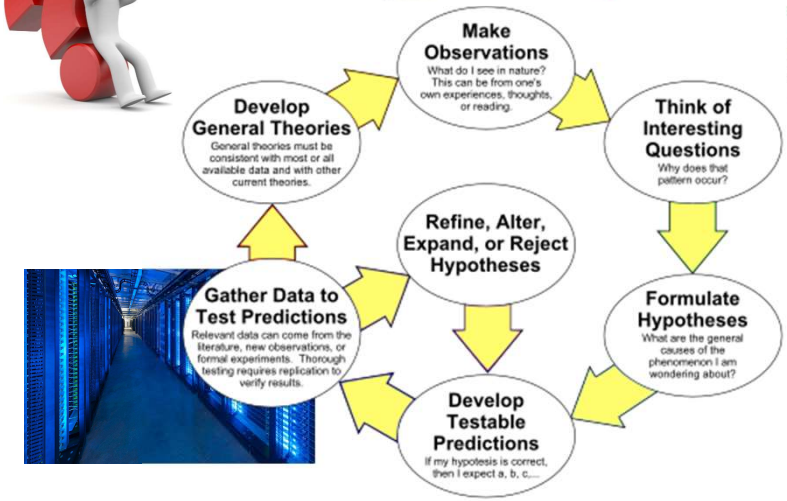
Garland, Jr., Theodore. "The Scientific Method as an Ongoing Process". U C Riverside.

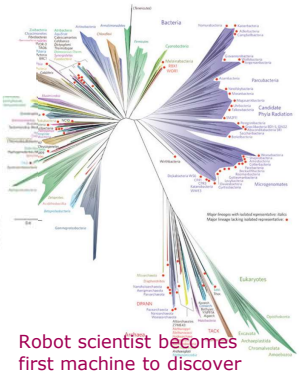
Discovery through Automation

The Scientific Method ~ 2020's


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







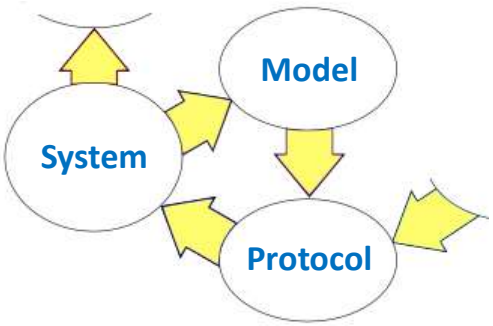
Robot scientist becomes first machine to discover new scientific knowledge

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

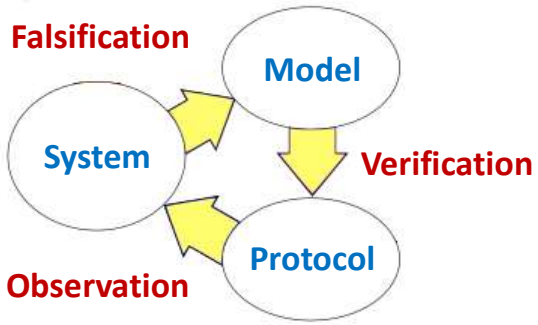
- Model: unambiguous (mathematical) description (CompBio)
- Protocol: standardized (engineered) parts and procedures (SynthBio)
- System: characterized (biological) organism and foundries (SysBio)

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



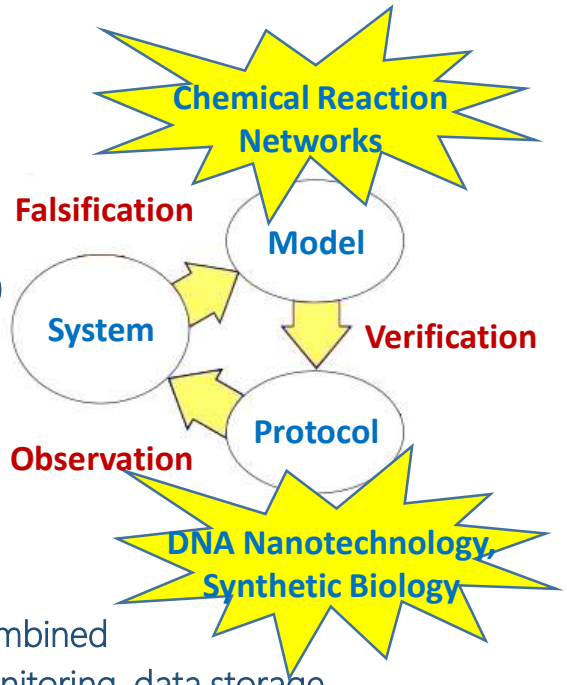
The Inner Loop

- 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



Why are *abstract* chemical reactions interesting?

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

- A fundamental model of kinetics in the natural sciences
- A fundamental mathematical structure, rediscovered in many forms
  - Vector Addition Systems, Petri Nets, Bounded Context-Free Languages, Population Protocols, …
- A description of mechanism rather than just behavior
  - A way of describing and comparing biochemical algorithms
  - Enabling addition analysis techniques, e.g. evolution of mechanism through unchanging behavior
- A programming language (coded up in the genome) by which living things manage the processing of matter and information

Also, a formal language we can implement 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).

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

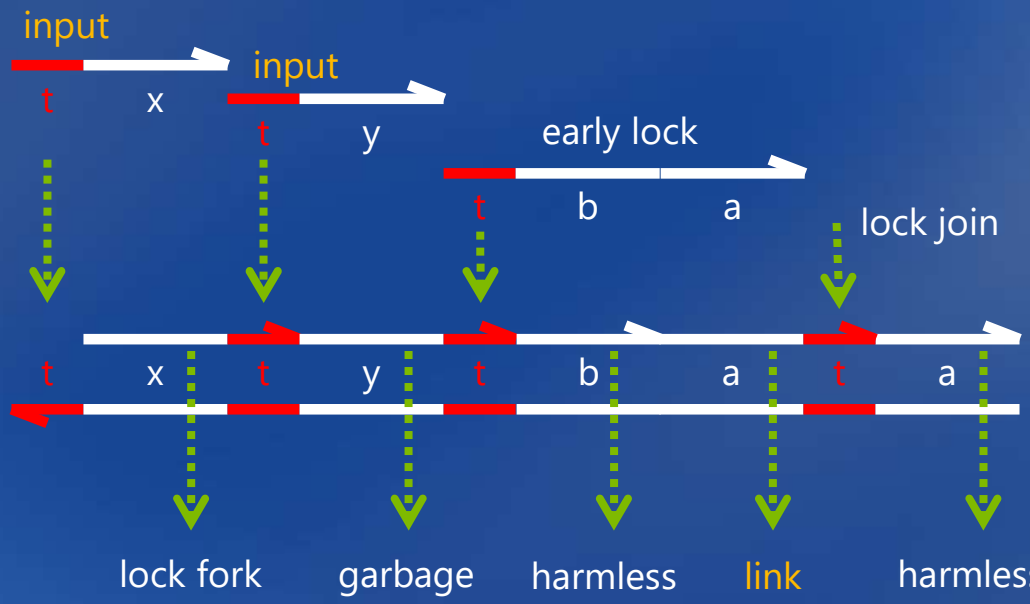
DNA as a universal substrate for chemical kinetics

David Soloveichik, Georg Seelig, and Erik Winfree

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

Reaction $x + y \rightarrow z + w$

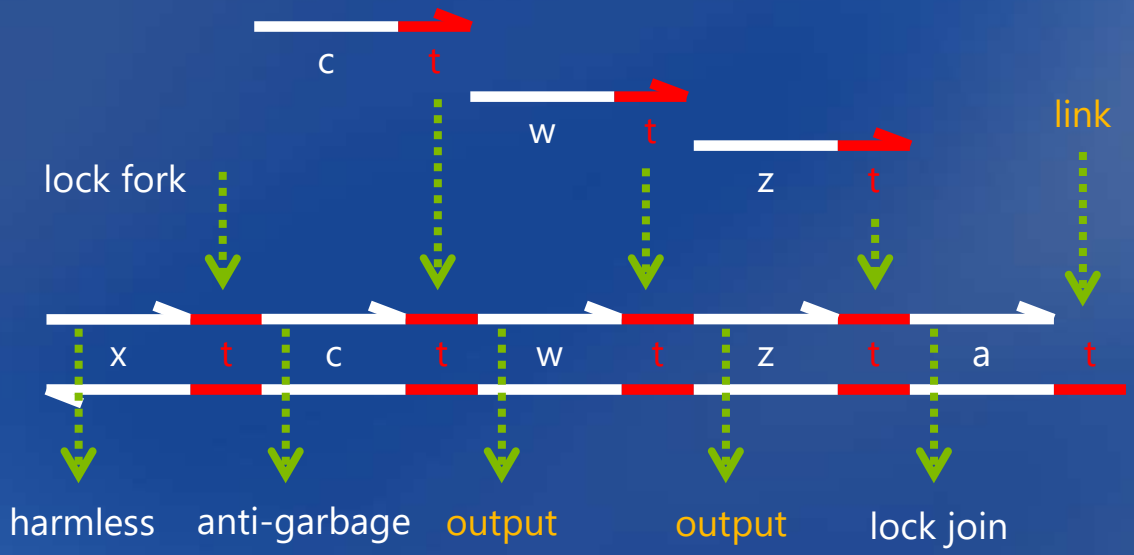
reactants
half



"join" structure

Reaction $x + y \rightarrow z + w$

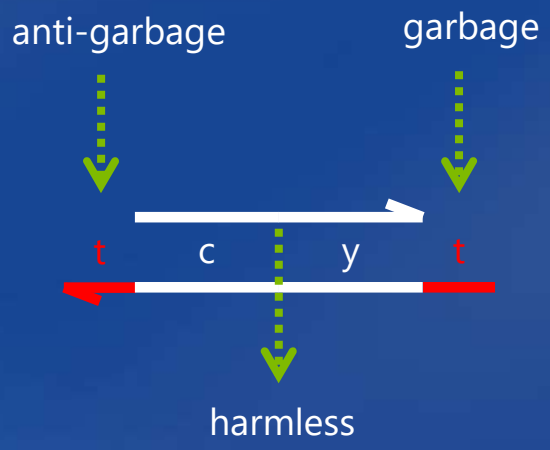
products
half



"fork" structure

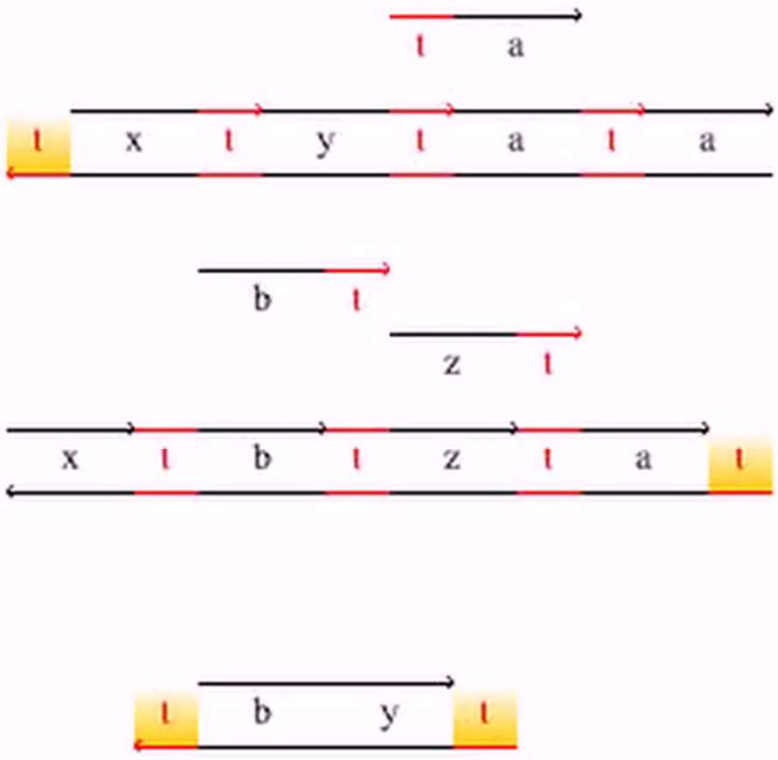
Reaction $x + y \rightarrow z + w$

garbage collection



Powered by Sothink

Join x+y → z



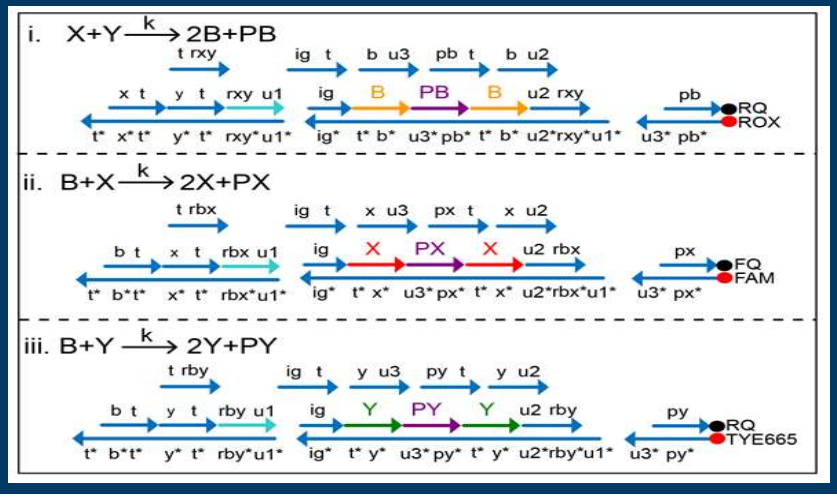
DNA Implementation of the Approximate Majority Algoithm

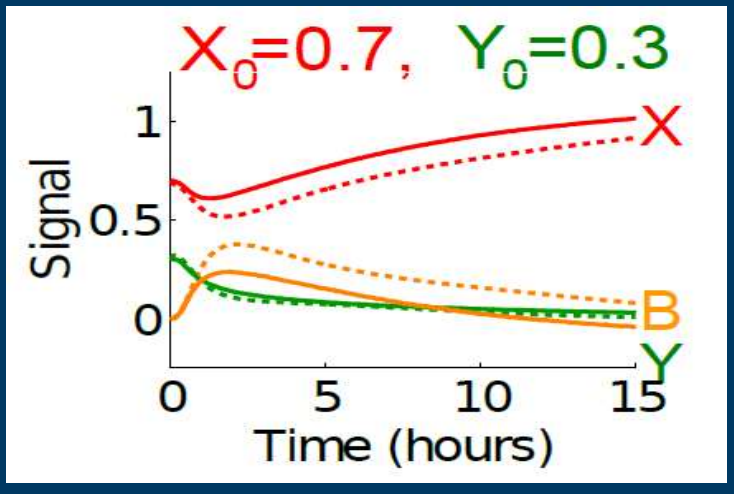
$$X + Y \rightarrow 2B$$
$$B + X \rightarrow 2X$$
$$B + Y \rightarrow 2Y$$

nature nanotechnology

Programmable chemical controllers made from DNA

Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik & Georg Seelig





Experimental-Protocol Languages for Chemical Reaction Networks

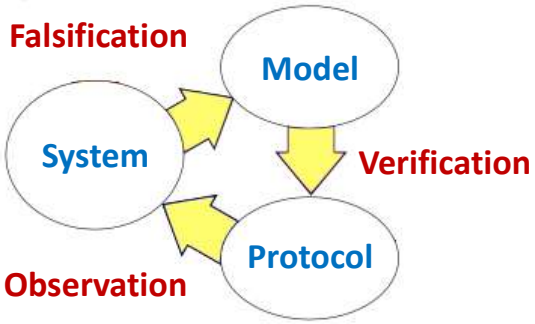
Automating “the whole thing”

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

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

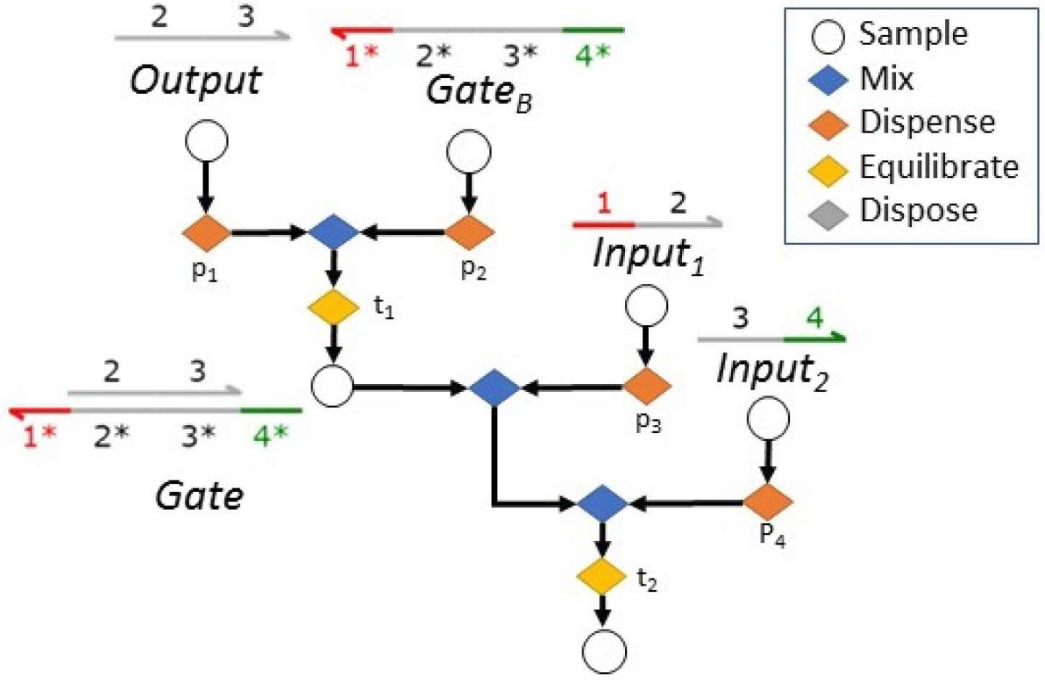
- Reproducibility crisis
  - Experiments are hard to reproduce.
  - Even models are hard to reproduce!

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



A Protocol

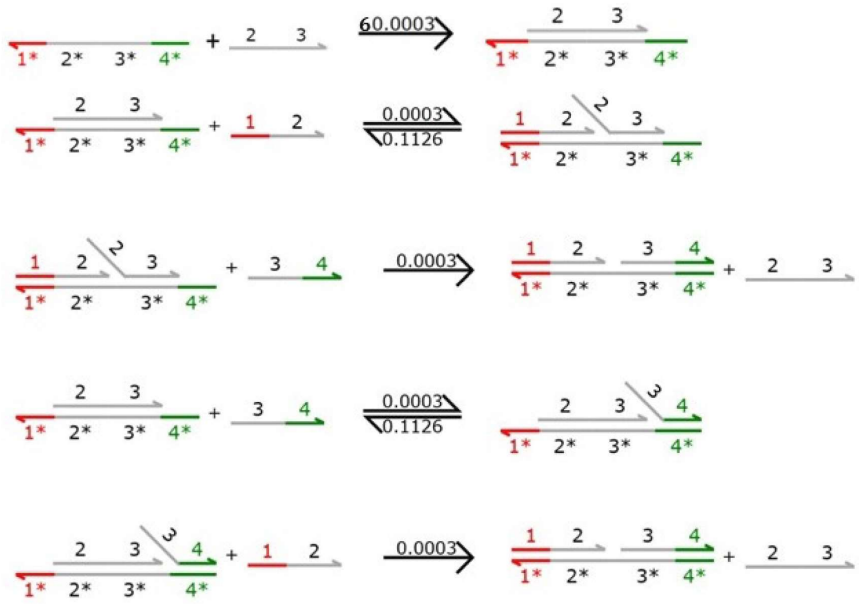
For DNA gate assembly and activation in vitro



Protocol steps (liquid handing)

A Model

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



An Integrated Description

This requires a language

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

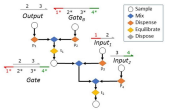
$$
\begin{aligned}
P = \quad & x && (\textit{sample variable}) \\
& (x_0, V, T) && (\textit{initial condition}) \\
& Mix(P_1, P_2) && (\textit{mix samples}) \\
& let\, x = P_1\, in\, P_2 && (\textit{define variable}) \\
& let\, x, y = Dispense(P_1, p)\, in\, P_2 && (\textit{dispense samples}) \\
& Equilibrate(P, t) && (\textit{let time pass}) \\
& Dispose(P) && (\textit{discard } P)
\end{aligned}
$$

Experimental Biological Protocols with Formal Semantics

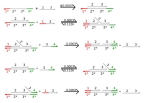
Alessandro Abate[2], Luca Cardelli[1,2], Marta Kwiatkowska[2], Luca Laurenti[2], and Boyan Yordanov[1]

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

The CRN can be computed from *{Input$_1$, Input$_2$, Output, Gate}*, and its initial conditions and evolution are determined by the protocol steps.



+



=

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

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

Language Semantics (deterministic)

The deterministic case is a warm-up exercise, but simple to explain

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

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

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

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

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

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

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

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

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

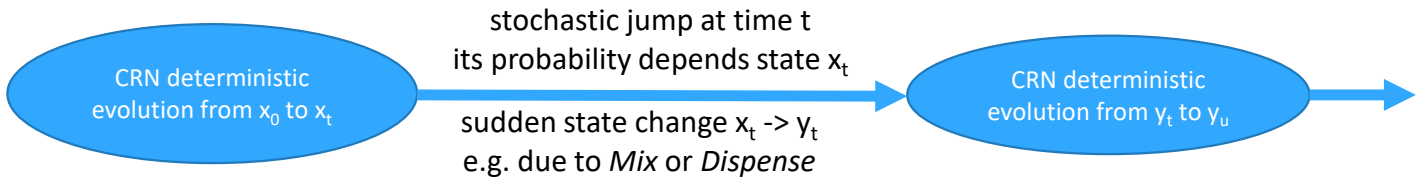
State produced by CRN $\mathcal{C} = (\mathcal{A}, \mathcal{R})$ at time t:

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

Language Semantics (stochastic)

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

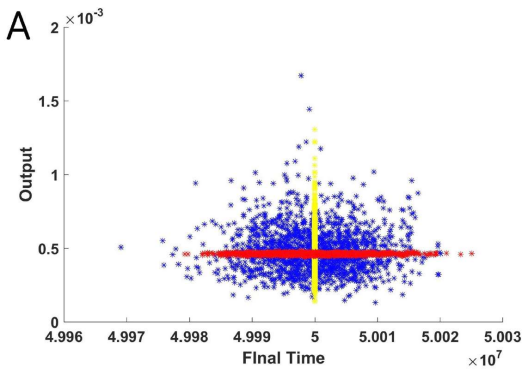
Each program now represents a Hybrid System with stochastic jumps between deterministic evolutions:



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

Stochastic Analysis

- We can ask: what is the probability of a certain outcome given uncertainties in *both the protocol and the model*?
- Conversely: which parameters of *both the protocol and the model* best fit the observed result?



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

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

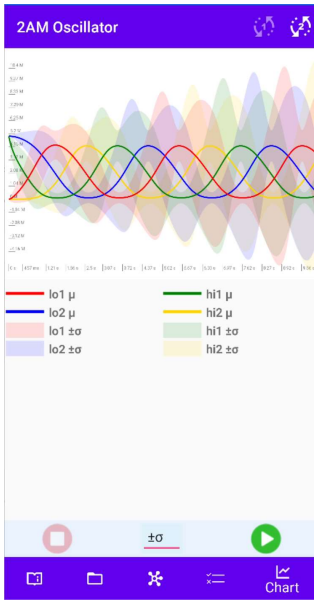
Kaemika

- A prototype language for chemical models & protocols

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

- Search "Kaemika" in the App stores

macOS
iOS
Android
Windows
GitHub



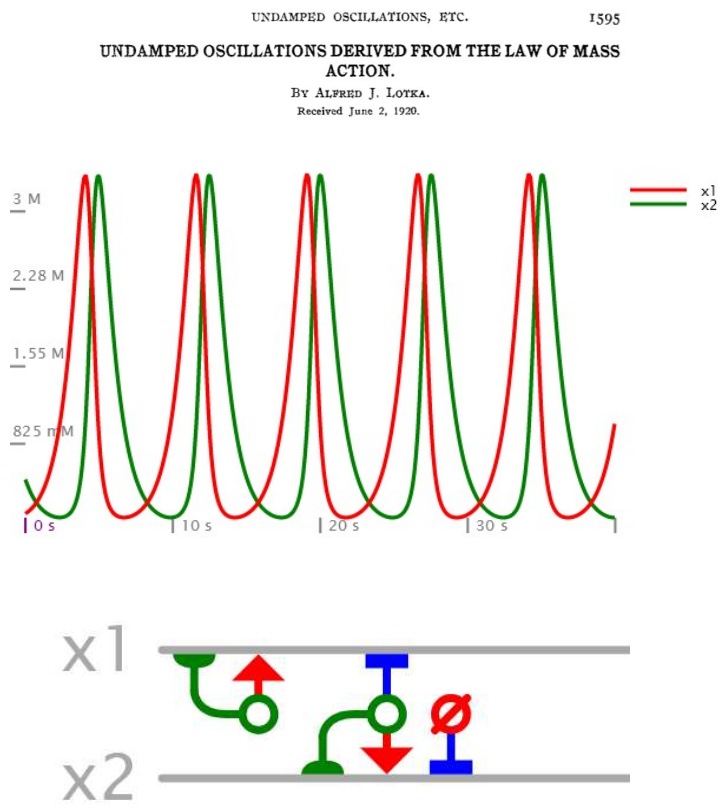
- CRN simulation
- Microfluidics simulation
- Reaction graphs
- ODE equations
- Stochastic noise (LNA)

Describing a Model

- *Species* and *reactions*
  - Characterized by initial values and rates
- *Kinetics*
  - Deterministic (ODE) or stochastic (LNA)
- *"Samples" (compartments) and Protocols*
  - Isolate species and reactions in a compartment, and mix compartments
- *Programming abstractions*
  - Assemble models as compositions of modules

Species and Reactions

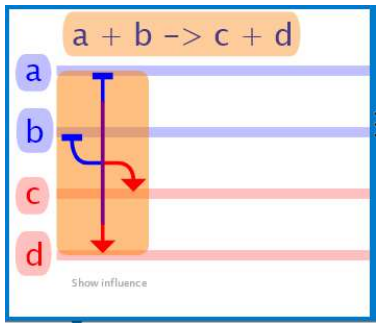
```
//=======================================
// Lotka 1920, Volterra 1926
// (simplified with all rates = 1)
//=======================================

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
```



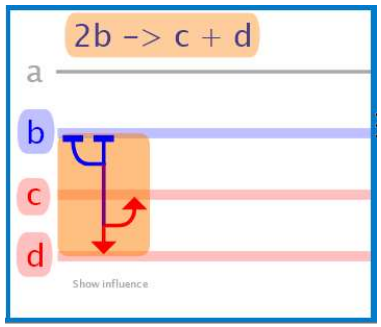
Reaction scores (graphical representation of reaction networks)

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

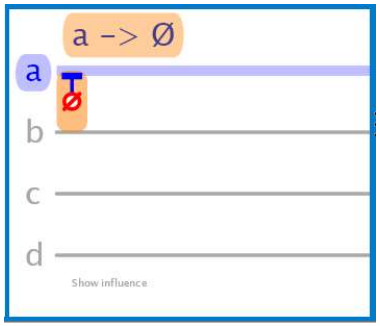
Reactants and products



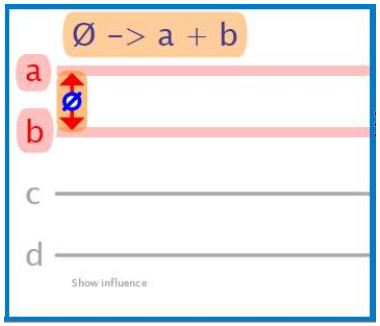
Repeated species



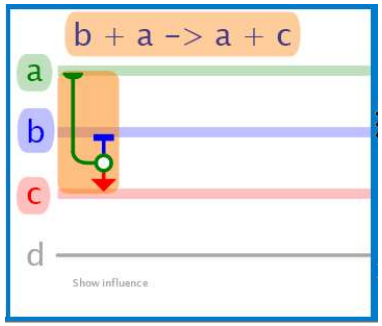
Reactants but no products



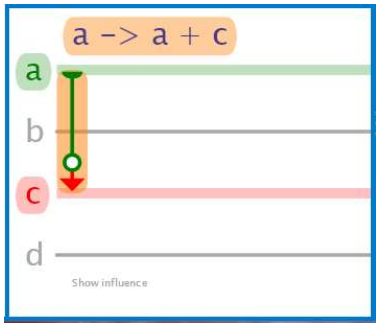
Products but no reactants



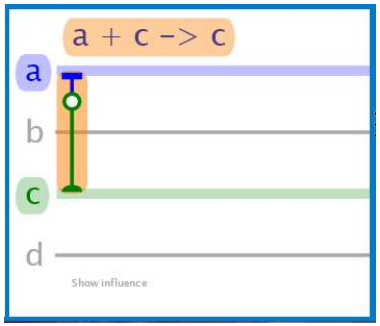
Catalyst



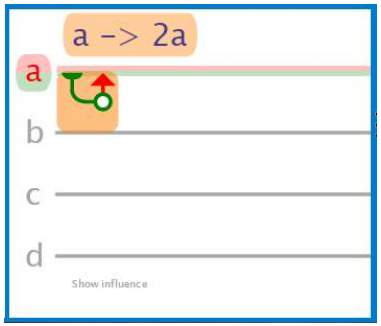
Catalyst but no reactants



Catalyst but no products



Autocatalyst



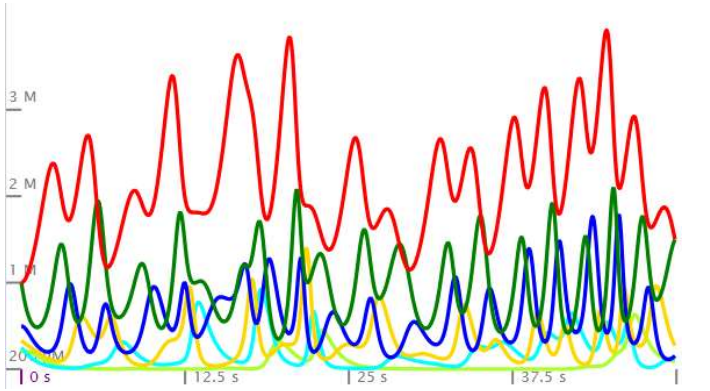
Writing Models Compositionally

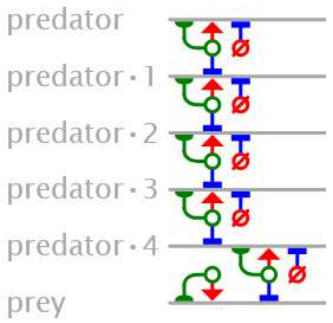
- Functional-monadic approach
  - *Functions* take *data* as parameters and produce *data* as results
  - *Networks* take *data* as parameters and produce *effects* as results
  - *Data* is *numbers*, *species*, *functions*, *networks*, *flows*, etc.
  - *Effects* are *species creation*, *reaction definitions*, *and sample handling*
  - A program execution produces both a final *result* and a sequence of *effects*
- (Temporal) Flows
  - Flows are functions of time (mostly real-valued)
  - Can be assembled programmatically (as a data structure)
  - Can be used as ***rates*** (leading to programmable kinetics)
  - Can be ***observed*** at specific times (leading to protocol observations)
  - Can be ***plotted*** over time (leading to chart series and legends)

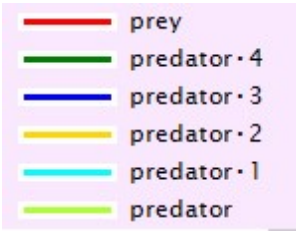
Ex: Predatorial

```
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 -> ∅ // 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.
//=====================================
```







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

- *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))
```

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
```

Experimental Biological Protocols with Formal Semantics

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

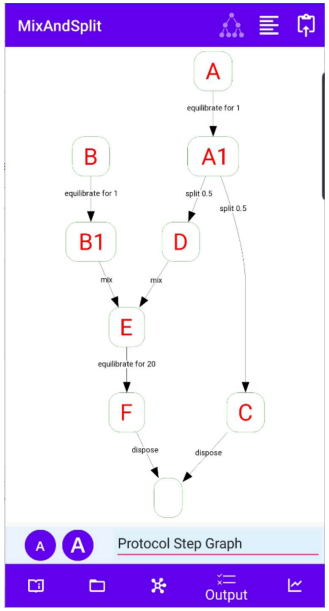
[1] Microsoft Research Cambridge
[2] Department of Computer Science, University of Oxford

Ex: Sample Manipulation

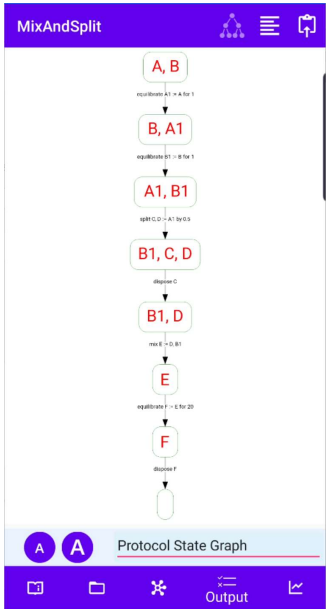
Multiple equilibration steps

```
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
```

"Protocol step graph"



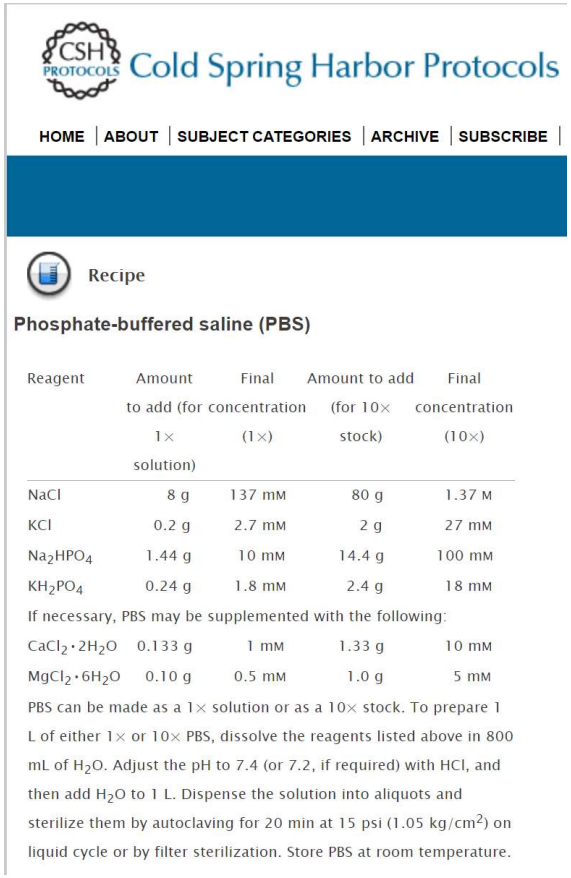
"Protocol state graph"



PDMP ("System Equations")

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 PROTOCOLS Cold Spring Harbor Protocols

HOME | ABOUT | SUBJECT CATEGORIES | ARCHIVE | SUBSCRIBE |

Recipe

Phosphate-buffered saline (PBS)

Reagent	Amount to add (for 1× solution)	Final concentration (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.

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

Ex: Serial Dilution

```
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 -> ∅

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)

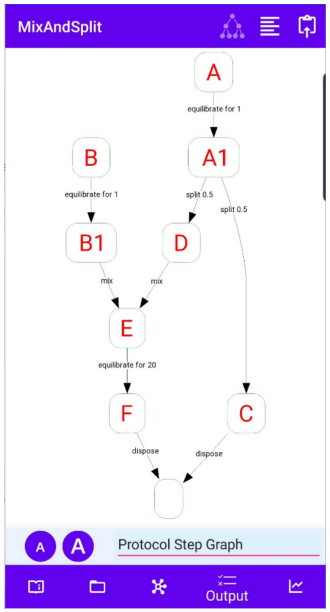
```
RESULT:
sample init {1mL, 298.2K} {A = 1M, B = 1M}
sample s2 {1mL, 298.2K} {A = 100mM, B = 100mM}
sample s4 {1mL, 298.2K} {A = 10mM, B = 10mM}
sample s7 {1mL, 298.2K} {A = 1mM, B = 1mM}
sample s10 {1mL, 298.2K} {A = 100uM, B = 100uM}
```

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)

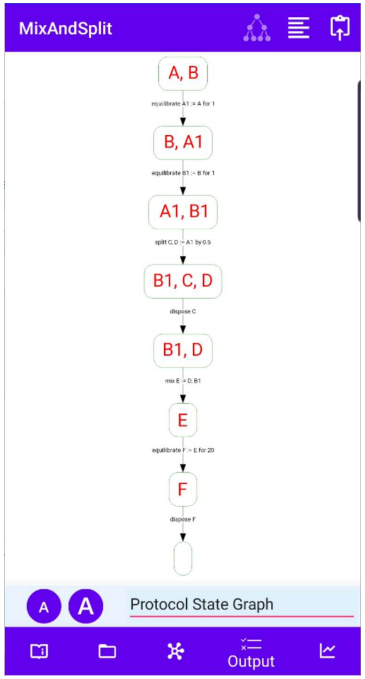
```
STATE_5
sample E {1.5mL, 293.2K} {
   a = 354.5mM
   c = 178mM
   b = 0.5674M
   consumed
   a + c -> a + a
   b + c -> c + c
   a + b -> b + b
}

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
```

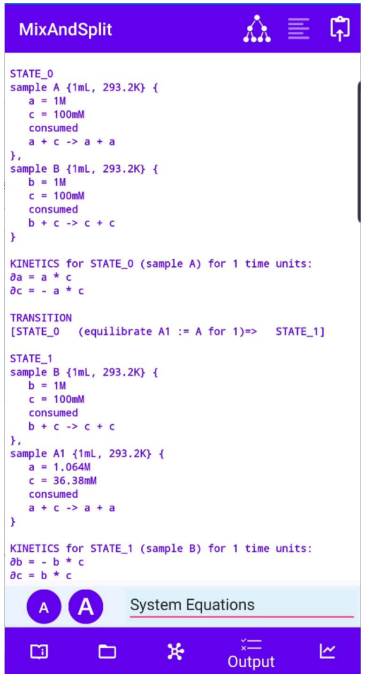
Extracting the Hybrid Transition System

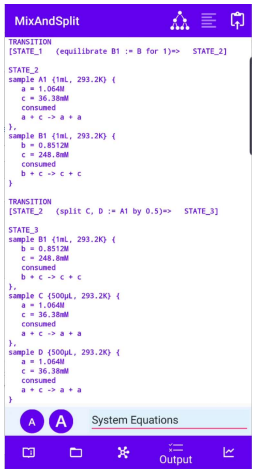
From the script

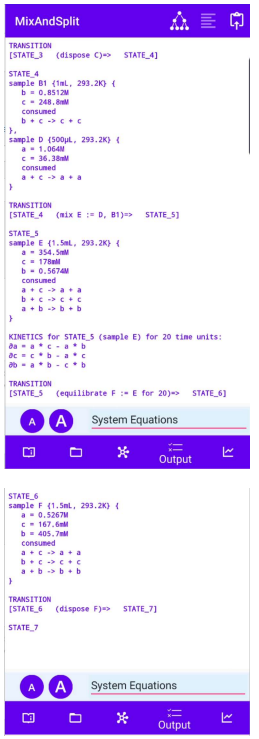
```
species {c}

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)









Executing the protocols

- We have seen that *reactions can be executed* by DNA
- But how can we *execute the protocols*, so that we can execute the whole thing together?
- -> Digital Microfluidics Compiler

Digital Microfluidics

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

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

Digital Microfluidics

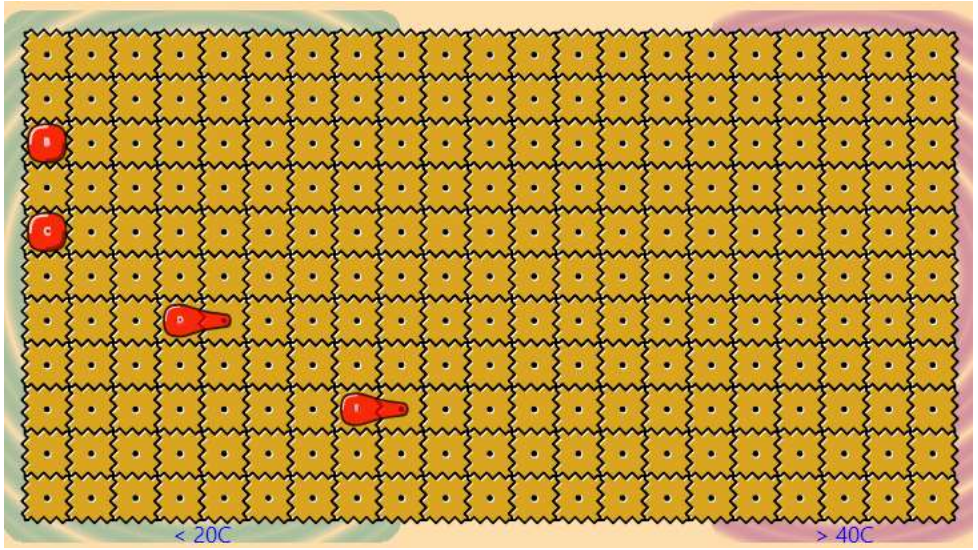
- A general, *programmable*, platform to execute the main liquid-handling operations

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

Digital Microfluidics Compiler

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

```
sample A {3µL, 20C}

split B,C,D,E = A

mix F = E,C,B,D

dispose F
```



Other features

- Timeflows
  - General kinetic rates (fractions, rational powers, exponentials, trigonometry) work with both deterministic and stochastic simulation and equation-extraction
  - 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 the Omega probabilistic language, with rejection sampling.

Conclusions

Bridging culture gaps

We can have more sophisticated modeling languages than chemical reactions
And we can have more sophisticated protocols than liquid handling
But it is good to find an intersection where we can get them into an automated loop

Chemical reaction networks

An interface between engineering (algorithms, programming, verification)
and science (dynamical and stochastic systems in nature, laboratory protocols)

Closed-loop models and protocols

Unified description of the scientific cycle

Automation (programmability)

Generating networks of parametric size and complexity
Scripting protocols