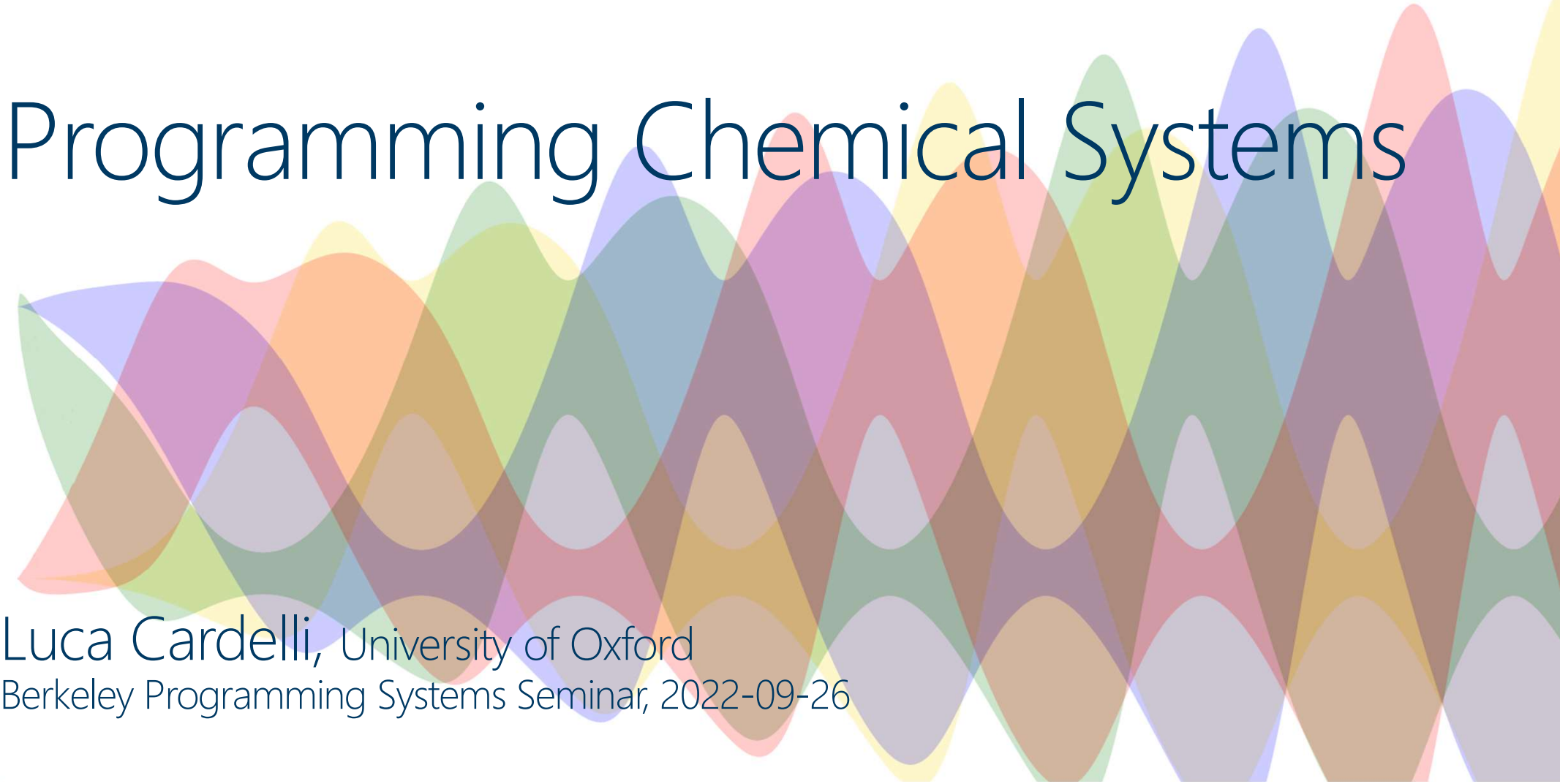
Programming Chemical Systems



Luca Cardelli, University of Oxford

Berkeley Programming Systems Seminar, 2022-09-26

Outline

1. From “any” Digital or Analog system
   to a Chemical Reaction Network

2. From (made-up) Chemical Reaction Networks
   to (real) Molecules that implement them
   (skipped)

3. Languages for Chemical Reaction Networks

Part 1

From (almost) any algorithm and (almost) any dynamical system to a Chemical Reaction Network

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*

Part 1a

"Digital" computation = algorithms

Programming Examples

spec	*program*
$Y := 2X$	$X \rightarrow Y + Y$
$Y := \lfloor X/2 \rfloor$	$X + X \rightarrow Y$
$Y := X1 + X2$	$X1 \rightarrow Y$ $X2 \rightarrow Y$
$Y := \min(X1, X2)$	$X1 + X2 \rightarrow Y$

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

CRN Semantics (discrete state space)*

- No-time (concurrent) semantics
  - Ignore rates. The multisets of molecules are rewritten according to the reactions, which may fire concurrently when not in resource conflict. This results in a Petri Net.
- Discrete time semantics
  - Reaction rates determine the *probability* with which reactions fire at discrete time intervals, then they behave as multiset rewrites at each discrete time interval.
    This results in a Discrete Time Markov Chain (DTMC).
- Continuous time semantics of CRNs
  - Reaction rates determine the *propensity* with which reactions fire (both the probability of firing and the inter-firing intervals), then they behave as multiset rewrites.
    This results in a Continuous Time Markov Chain (CTMC).
  - These CRNs are called FSCRN (finite stochastic CRN).

*Discrete state space means that each chemical species has a number of molecules (a nonnegative integer); then time can be modeled as one of the above.

Programming "approximately" *any* algorithm as a FSCRN

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

FSCRNs are not Turing complete

Like Petri nets: reachability is decidable

But unlike Petri nets, FSCRNs 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 the model makes it fully Turing complete

but the syntax becomes considerably more complex

Formal Molecular Biology

Vincent Danos* Cosimo Laneve†

Register Machines (almost…)

i: INC R_1; JMP j	$PC_i \rightarrow R_1 + PC_j$
i: DEC R_1; JMP j	$PC_i + R_1 \rightarrow PC_j$
i: IF R_2>0 {INC R_1; JMP j}	$PC_i + R_2 \rightarrow R_2 + R_1 + PC_j$
i: IF R_2=0 …	??? Whatever trick we use will have some error

- Turing-complete up to an arbitrarily small error
  - The error bound is set in advance uniformly for any computation of arbitrary length (because we cannot know how long the computation will last), and the machine will progressively "slow down" to always stay below that bound.

- David Soloveichik, Matt Cook, Erik Winfree, Shuki Bruck, "Computation with Finite Stochastic Chemical Reaction Networks".
  [Natural Computing, (online Feb 2008), or Technical Report: CaltechPARADISE:2007.ETR085: .pdf]

CRN Semantics (continuous state space/deterministic)*

- ODE semantics of CRNs
  - The chemical *Law of Mass Action* says that the *flux* of a reaction is determined by the product of the concentrations of the reagents, times the reaction rate.

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*):

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

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

*Continuous state space means each chemical species has a concentration (a real number); concentrations are approximations of the number of molecules via the Avogadro constant.

CRN Semantics (continuous state space/stochastic)*

- CME semantics of CRNs (Chemical Master Equation)
  - *Kolmogorov forward equation* of the Markov Chain produced by the CRN.
  - Unfeasible to solve or even simulate (to compute the distribution of outcomes)
  - The Gillespie algorithm produces individual samples (traces) of the CME distribution
- LNA semantics of CRNs (Linear Noise Approximation)

Gaussian state (mean & variance) 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*, $with\ \boldsymbol{\mu}_\mu(0) = \mu\ and\ \boldsymbol{\Sigma}_{\mu,\Sigma}(0) = \Sigma.$

$$[\![((\mathcal{A},\mathcal{R}, x_0), V, T)]\!](H)(t) = (\boldsymbol{\mu}_\mu(t), \boldsymbol{\Sigma}_{\mu,\Sigma}(t), V, T)$$

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

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

$F(V,T)(\mu) = \sum_{\tau\in\mathcal{R}} \upsilon_\tau \alpha_\tau(V,T,\mu)$, *with stoichiometric vector* υ_τ *and rate function* α_τ. *We call* J_F *the Jacobian of* $F(V,T)$, *and* $J_F^\top$ *its transpose. Further, define* $W(V,T)(\mu) = \sum_{\tau\in\mathcal{R}} \upsilon_\tau \upsilon_\tau^\top \alpha_\tau(V,T,\mu)$

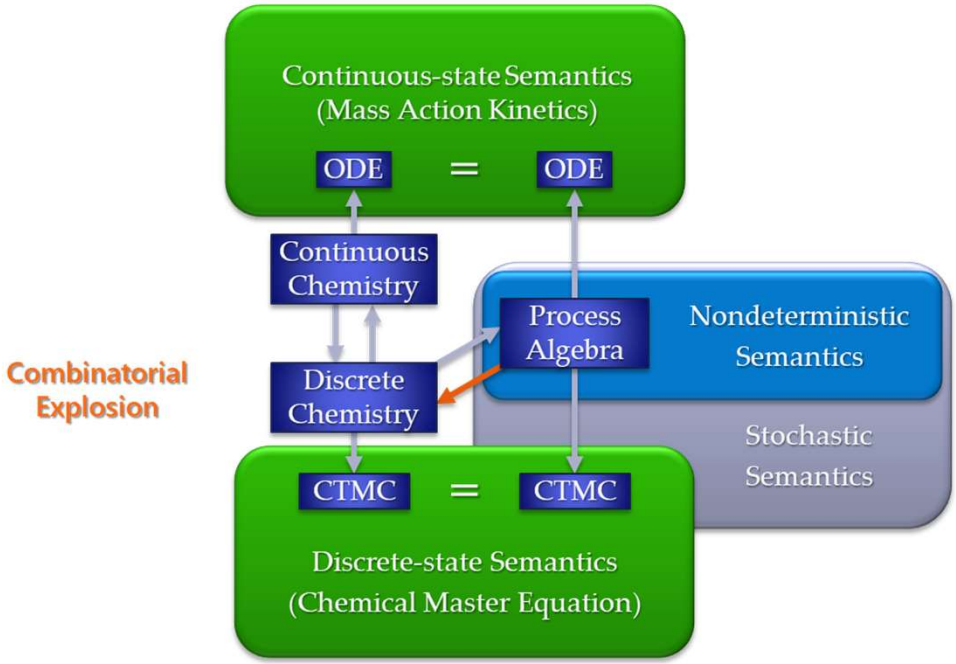
A Language for Modeling And Optimizing Experimental Biological Protocols

Luca Cardelli [1], Marta Kwiatkowska [1] and Luca Laurenti [1,†]

*Continuous state space means each species has a concentration (a real number); concentrations are an approximation of the number of molecules via the Avogadro constant.

Chemistry as a Concurrent Language

- A connection with the theory of concurrency
  - Via Process Algebra and Petri Nets



Finally, Some *Bad Bad* Programs

X -> Y

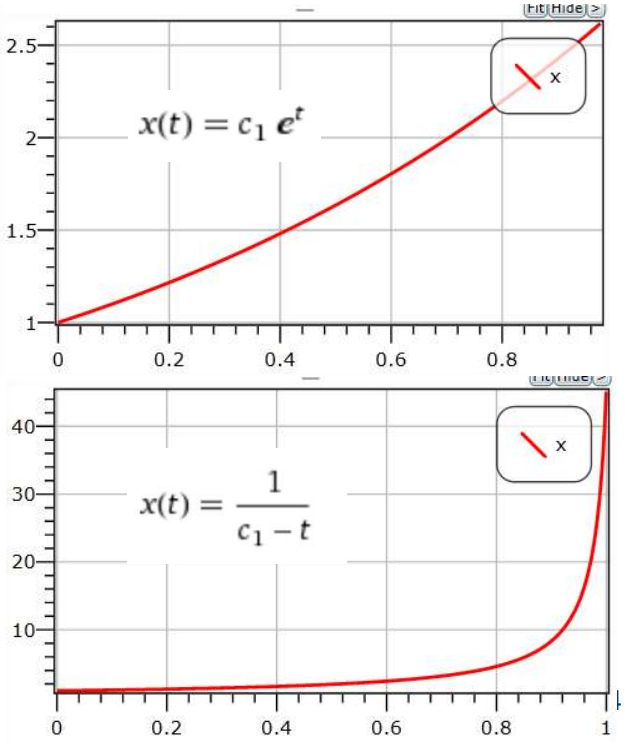
Violates thermodynamics.
(No biggie, assume there is a tiny reverse reaction.)

X -> X + X

Violates conservation of mass.
(No biggie, assume there is inflow/outflow.)

X + X -> X + X + X

Violates finite density.
(This is *really* bad.)



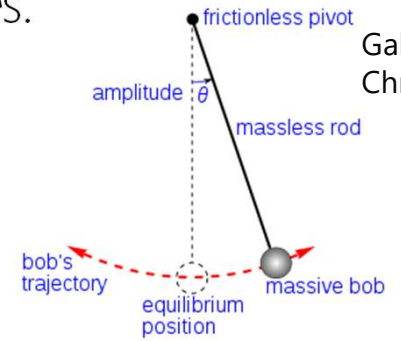
Part 1b

“Analog” computation = dynamical system

"Elementary" (NOT!) dynamical systems

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.



Galileo Galilei 1602
Christiaan Huygens 1673

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

E.g., physics: the equation of the simple pendulum has trigonometry on the r.h.s.:

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

E.g., biology: the enzyme kinetics equation has fractions on the r.h.s.:

$$\partial[P] = V_{max}[S] / (K_M + [S])$$

E.g., metereology: the chaotic Lorenz attractor has just 3 polynomial equations:

$$\partial x = ay - ax \qquad \partial y = cx - xz - y \qquad \partial z = xy - bz$$

E.g., chemistry: the law of mass action for CRNs implies that their ODEs are (a restricted "Hungarian" class) of polynomials

STEP 1, Polynomization: All elementary ODEs can be exactly reduced to polynomial ODEs.

MATHEMATICAL THEORY OF THE DIFFERENTIAL ANALYZER

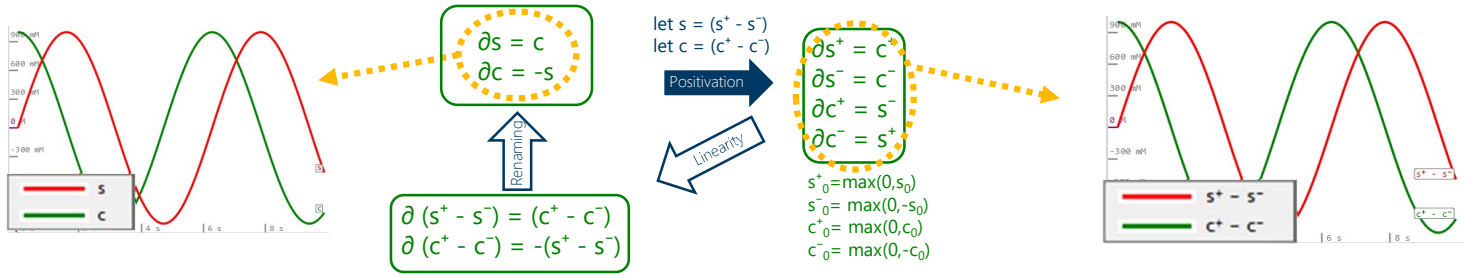
By Claude E. Shannon

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.

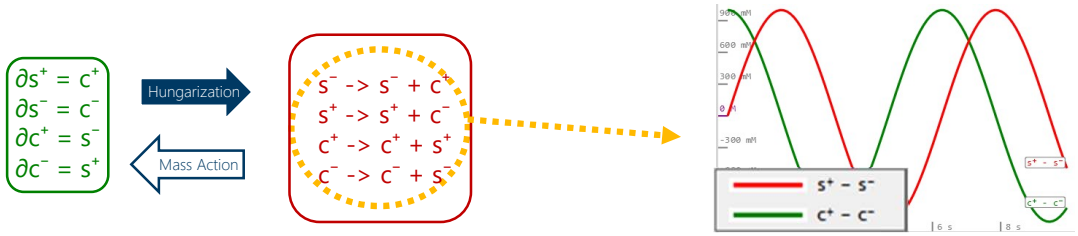
But variables go negative: we can't have that in a CRN (no negative concentrations).

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

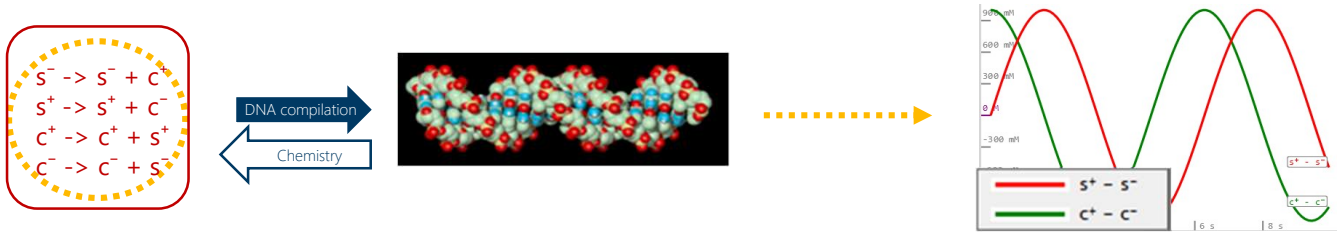
V. HÁRS - J. TÓTH

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

E.g. the Lorenz chaotic attractor is already polynomial but not Hungarian,
it cannot be translated to mass action reactions without first doing positivation.

Programming *any* "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

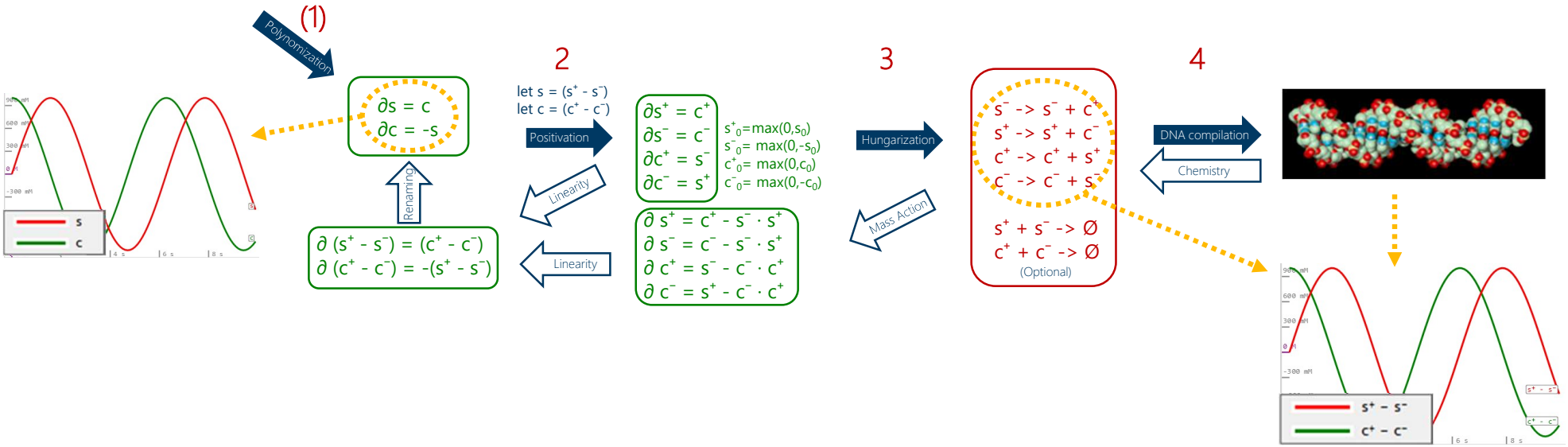
David Soloveichik, Georg Seelig, and Erik Winfree

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



Summarizing

- Chemistry is (also) a formal language that we can use to implement *any* algorithm and *any* dynamical system with *real* (DNA) molecules
- Turing complete and "Shannon complete"
- ANY collection of abstract chemical reactions can be implemented with specially designed DNA molecules, with accurate kinetics (up to time scaling).
- Approaching a situation where we can "systematically compile" (synthesize) a model to DNA molecules, run an (automated) protocol, and observe (sequence) the results in a closed loop.

Part 3

Languages for CRNs

Obviously…

Yes of course, there are CRN packages in Python, Matlab, Mathematica, etc. etc.

Yes of course, there are scripting languages, and even operating systems, for all kinds of lab equipment and for Digital Microfluidics, like PurpleDrop [Stephenson et al. 2020]

Yes of course, there are domain specific languages like CRN++ [Vasic et al. 2018]

I wanted to investigate "closing the loop" between mathematical modeling and lab protocols, based on a language for CRNs.

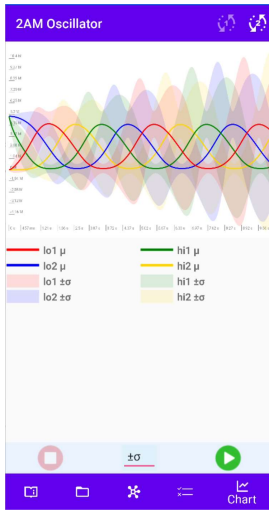
CMSB'2020 Best Tool Paper Award

Kaemika* app
Integrating protocols and chemical simulation

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

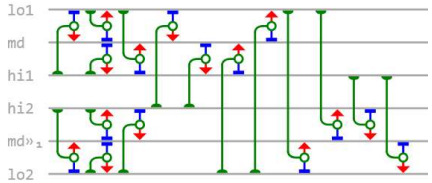
Luca Cardelli *, Marta Kwiatkowska and Luca Laurenti †

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

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

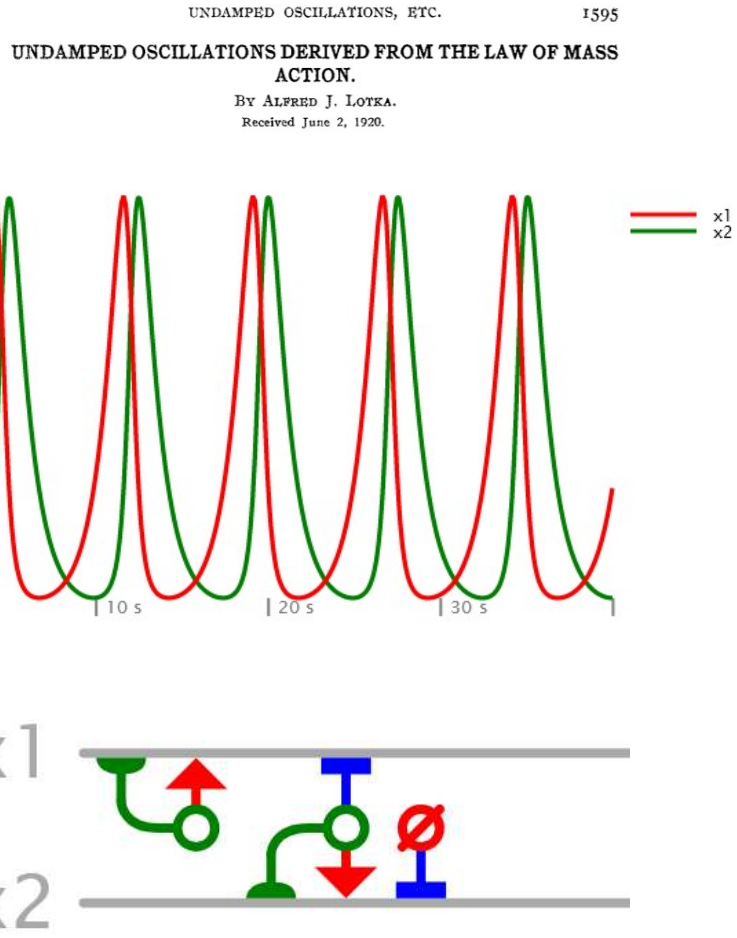
```
//=======================================
// Lotka 1920, Volterra 1926
// (simplified with all rates = 1)
//=======================================

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
```

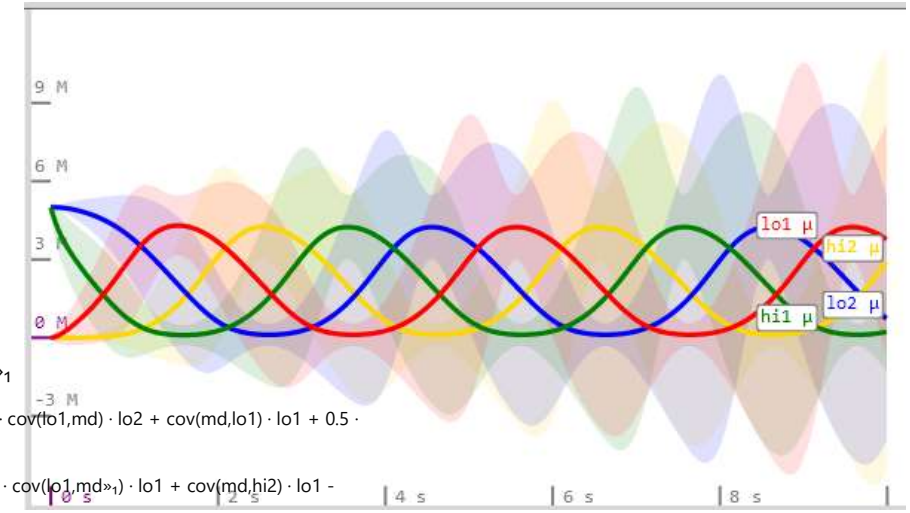


<= Demo: LotkaVolterra

Stochastic (LNA) simulation

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

2AM Oscillator



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

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

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

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

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

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

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

...

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

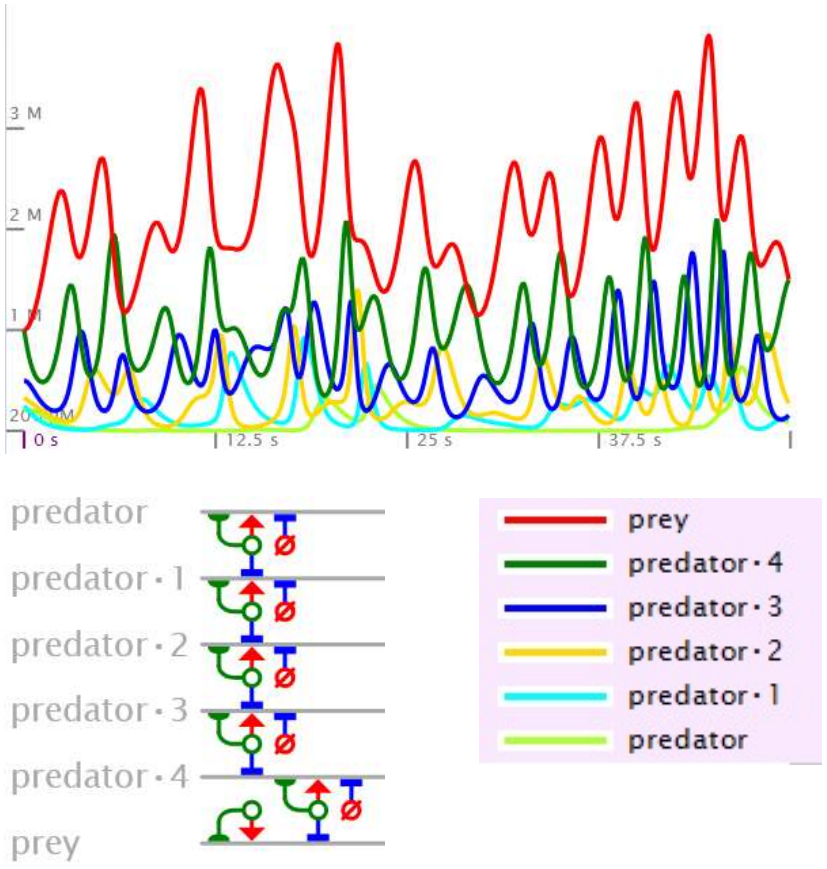
  Functional: programs take *data* as parameters and produce *data* as results
  Monadic: programs also produce *effects* (*species*, *reactions*, *liquid handling*)
  Nominal: *lexically scoped* chemical species (species are not "strings")

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 -> ∅ // predator dies
    report predator
    yield predator
  end
}

species apexPredator = Predatorial(5)
equilibrate for 50
```

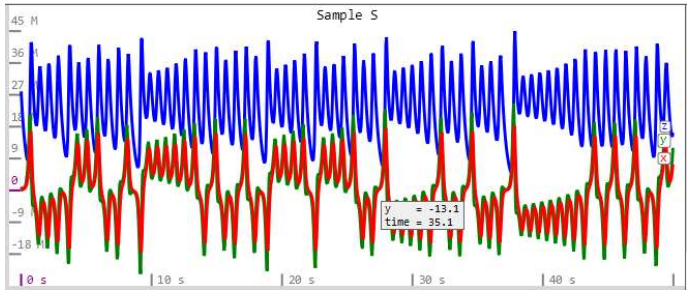
```
//=====================================
// Creates a stack of predator-prey
// relationships in Lotka-Volterra style,
// and returns the apex predator.
//=====================================
```



<= Demo: Predatorial

Mass Action Compiler

- Lorenz chaotic attractor

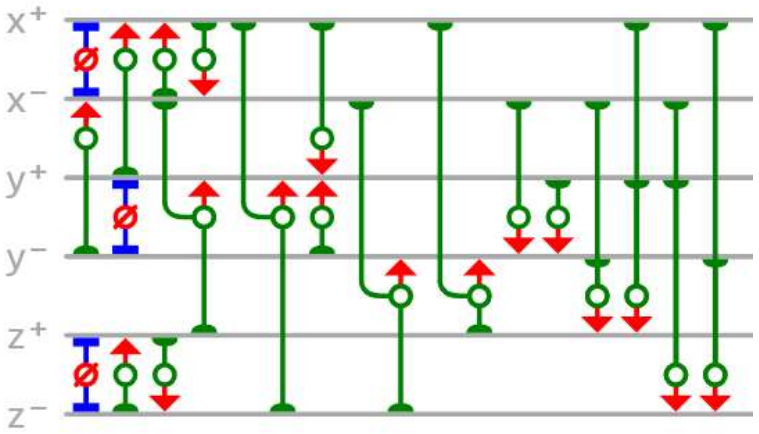


not mass action

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

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

Hungarian criterion violation



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

References

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 (general obfuscator)
- App store certificates
- Dark mode support

Conclusions

Chemical reaction networks

- A fun language to program with
- Compilable to real molecules
- Executable “in your kitchen”

- Still relative primitive, we need to build more programming abstractions