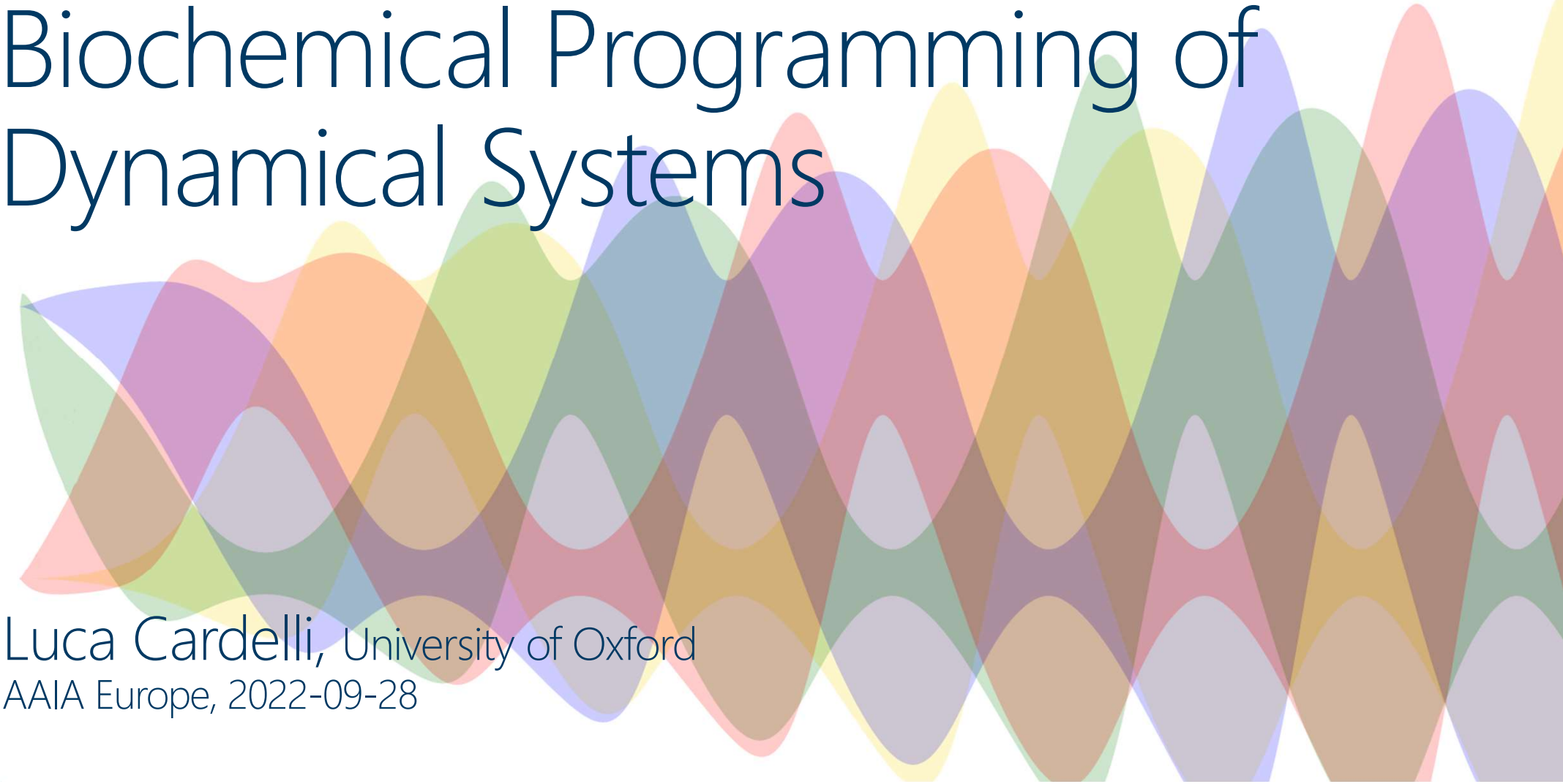
Biochemical Programming of Dynamical Systems



Luca Cardelli, University of Oxford

AAIA Europe, 2022-09-28

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 := $\lfloor X/2 \rfloor$	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

Finally, Some *Bad Bad* Programs

X -> Y

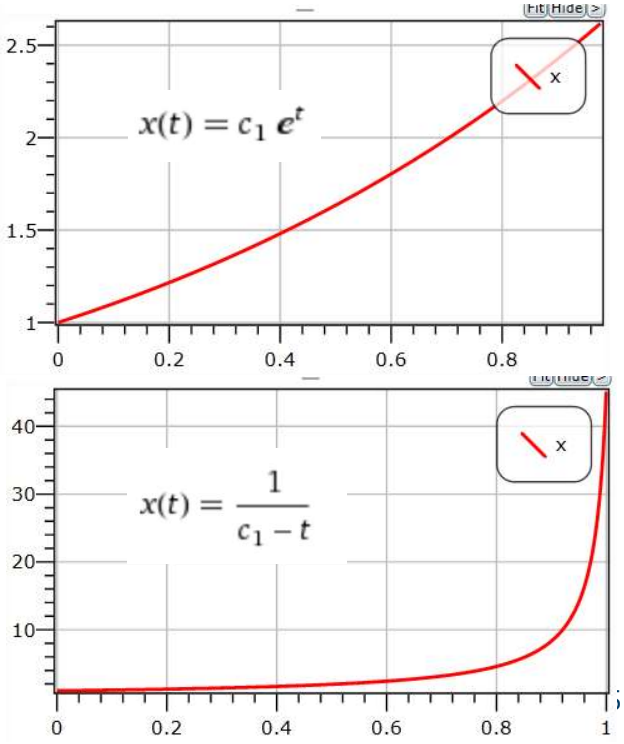
Violates thermodynamics.
(No biggie, assume there is a tiny reverse reaction.)

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.)



Programming "approximately" *any* algorithm as a FSCRN

A FSCRN is a *finite* set of reactions over a *finite* set of species
with a *stochastic* reaction activation rule base on the reaction rates

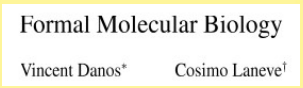
FSCRNs are not Turing complete
Like Petri nets: reachability is decidable

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

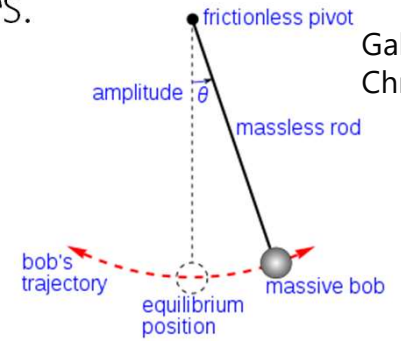

Formal Molecular Biology

Vincent Danos* Cosimo Laneve†

"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

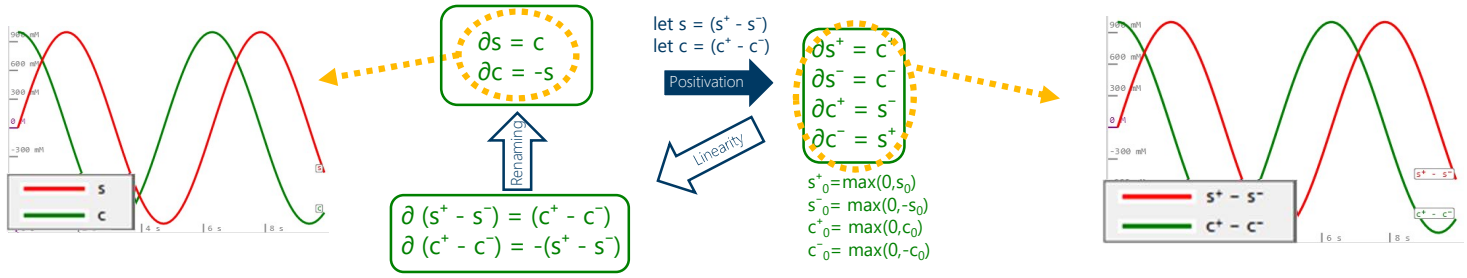
By Claude E. Shannon

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.

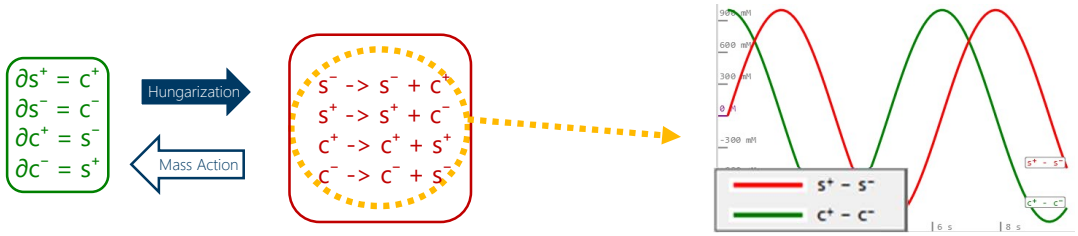
But variables go negative: we can't have that in a CRN (no negative concentrations).

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

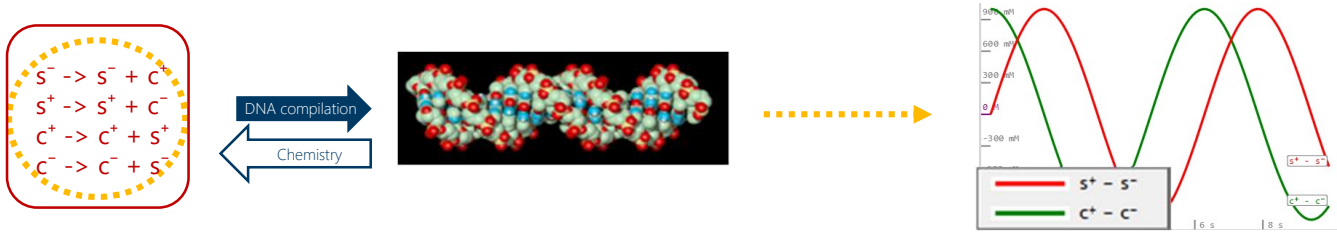
V. HÁRS - J. TÓTH

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

E.g. the Lorenz 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

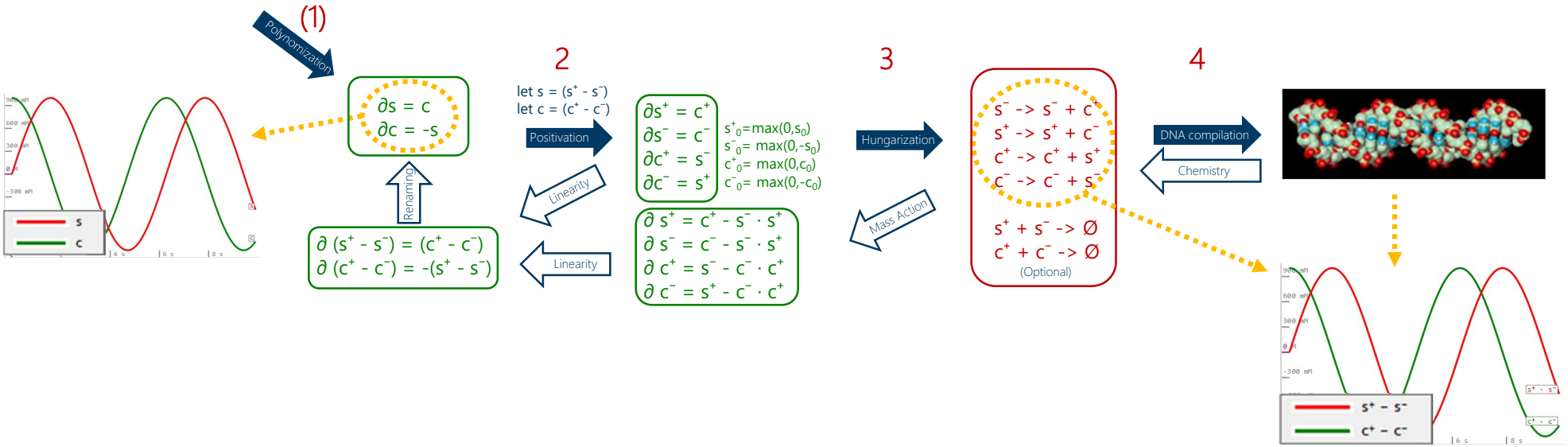
David Soloveichik, Georg Seelig, and Erik Winfree

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

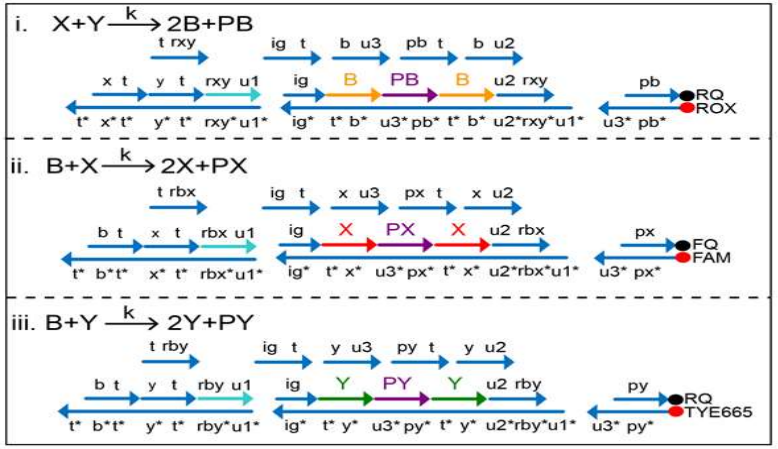


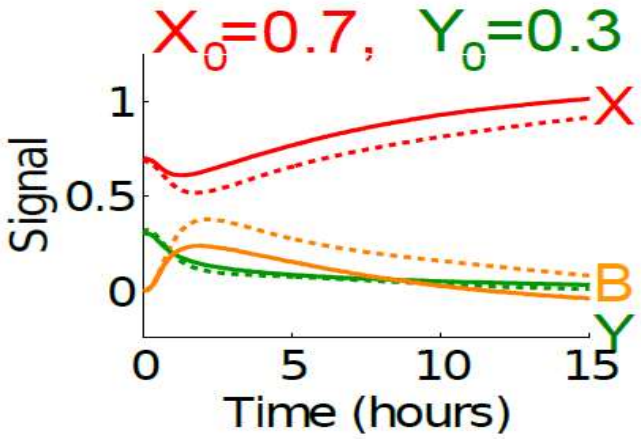
DNA Implementation of the Approximate Majority algorithm

nature nanotechnology

Programmable chemical controllers made from DNA

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



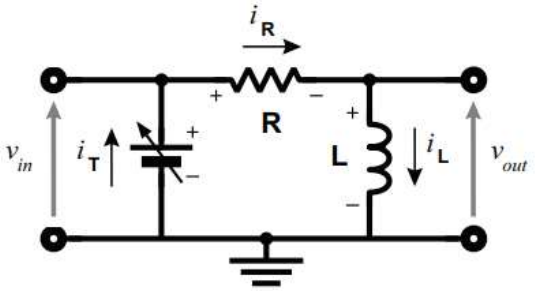


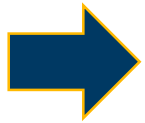
Extended to Linear DAE (differential-algebraic equations)

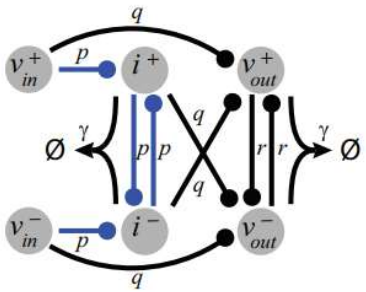
- An electric (RLC) circuit requires a mixture of
  - differential equations (Faraday's law of induction)
  - algebraic equations (Ohm's and Kirchhoff's laws)

- In general, of the DAE form $E\partial_t x = Ax + Bu$
  where *E* produces a linear combination of derivatives on the l.h.s.
  (not always reduceable to semi-explicit form)

- A reduction (involving an approximation) exists:







From Electric Circuits to Chemical Networks

Luca Cardelli[1], Mirco Tribastone[2], and Max Tschaikowski[3]

Conclusions

- 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.

Powered by Sothink

Join x+y → z