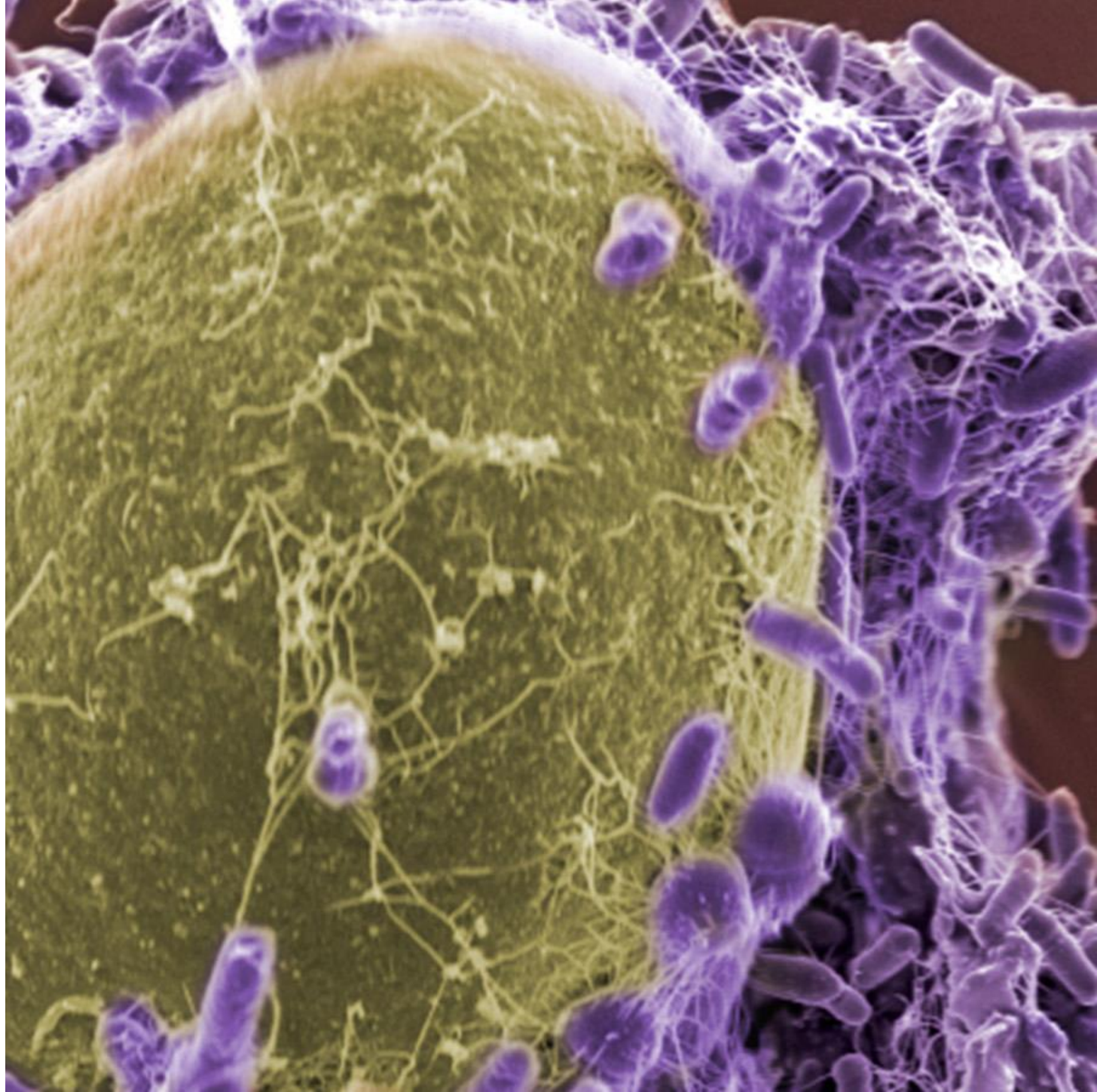
Pacific Northwest
NATIONAL LABORATORY

ChemComp: A Compilation Framework for Computing with Chemical Reaction Networks

Antonino Tumeo
Computer Scientist, PNNL

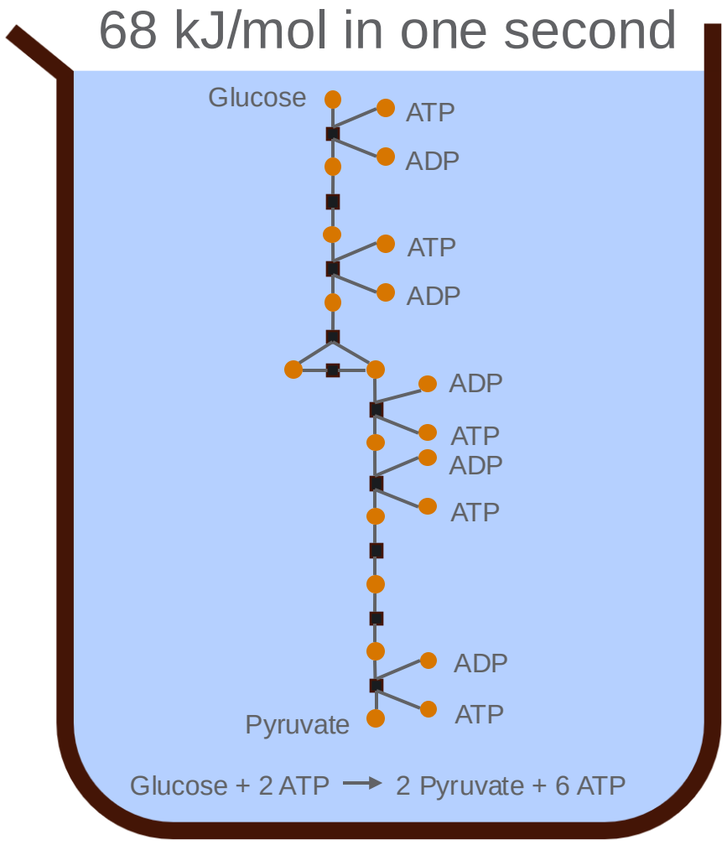
Authors: Nicolas Bohm Agostini
Connah G. M. Johnson
William Cannon
Antonino Tumeo



U.S. Department of ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

Chemistry conveys a “chemical advantage” for computation



Rules are ‘simple’ compared to biology

Potential of Biochemical Pathway
OPS: ~10^23
Energy/s: ~68 kW

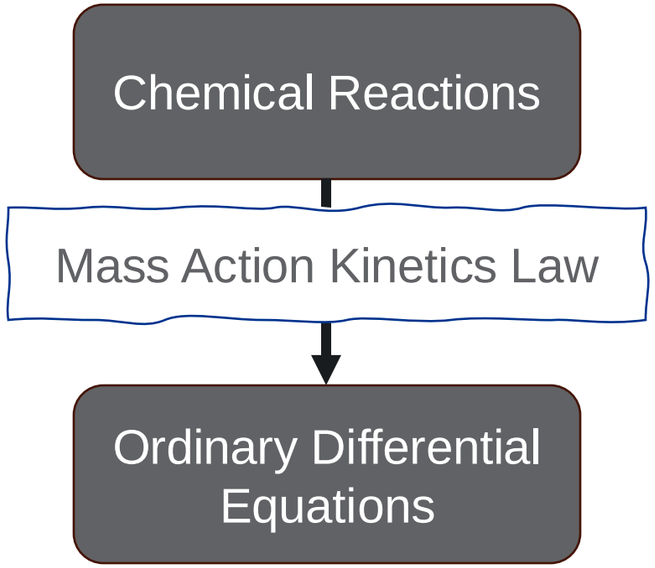
Frontier (ORNL, 2021)
FLOPS: ~10^18
Energy/s: ~29,000 kW

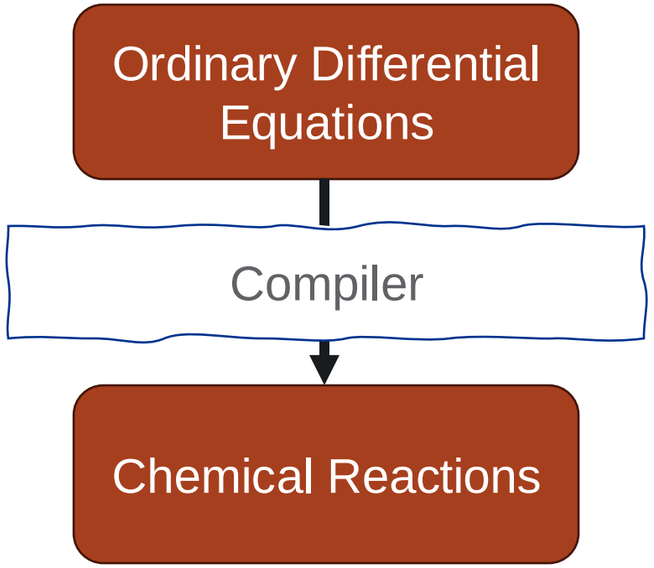
Potential Gain in OPS/W
~4x10^7 x

“Chemical advantage” meets the high-performance requirements for scientific workflows:
- Scalable
- Massively concurrent
- Continuous (time) representation
- “Low” energy

Beneficial to applications:
- Continuous systems such as ODEs as natural representation
- Neural networks/NLP as massively concurrent inference

Chemical reactions can be used for useful computation





D. Soloveichik, et.al. - Turing completeness [Natural Computing 2008]
M. Matejak, et.al. – Modelic Lib of Chem Processes [Modelica Conf 2015]
W. Poole, et.al. - BioCRNpyler [PLOS Computational Biology 2022]
R. Rendal, et.al. - Chemilang [Github 2023]

S. Shah, et.al. - CCompiler [arXiv 2008]
M. Vasic, et.al. - CRN++ [Natural Computing 2020]

RW does not consider:
- Concrete chemical reactions
- Tracking species over time

Chemical reaction can be mapped to ODEs

Using the Law of Mass Action kinetics:

$$\sum_i I_i^r X_i \xrightarrow{p_r(x)} \sum_i O_i^r X_i$$

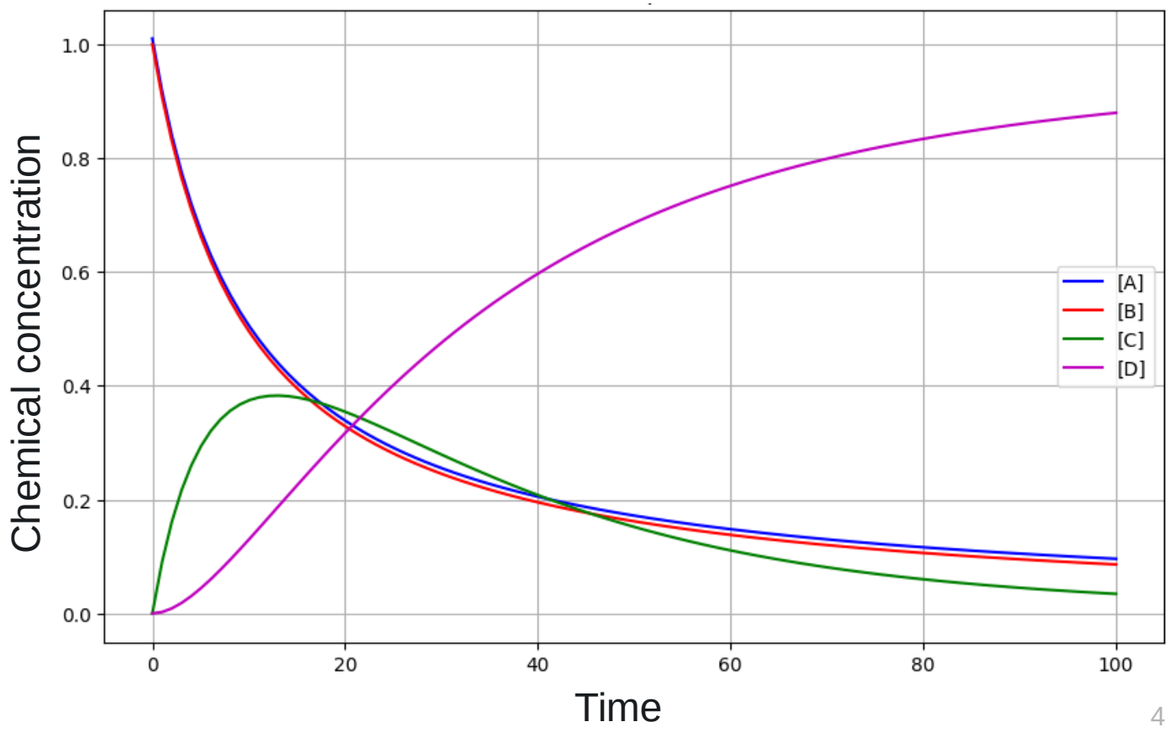
$$\frac{dX_i}{dt} = \sum_i (O_i^r - I_i^r) p_r(X)$$

I_i^r : Number of input species i in reaction r
O_i^r : Number of output species i in reaction r
$p_r(x)$: Propensity or Rate of the reaction r

r1: A + B $\xrightarrow{k1}$ C

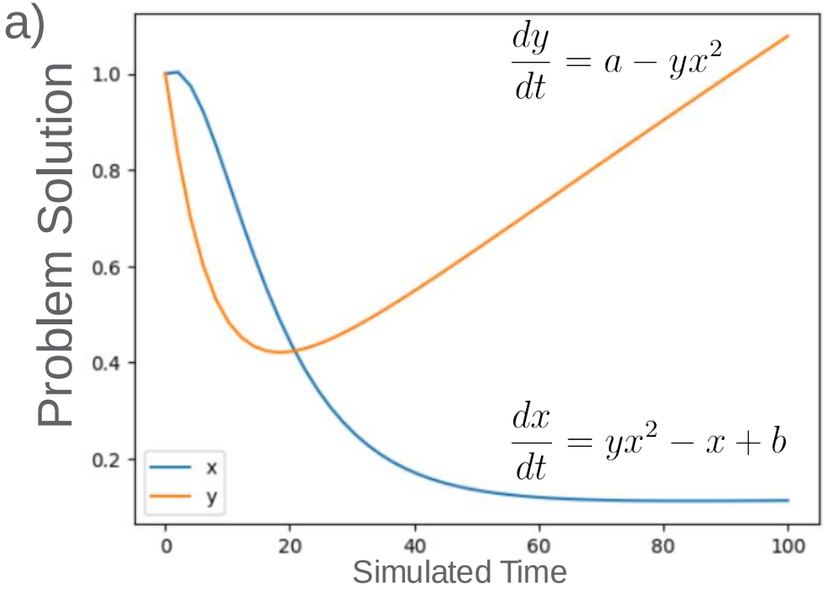
r2: C $\xrightarrow{k2}$ D

$$\frac{d[A]}{dt} = -k_1[A][B]$$
$$\frac{d[B]}{dt} = -k_1[A][B]$$
$$\frac{d[C]}{dt} = k_1[A][B] - k_2[C]$$
$$\frac{d[D]}{dt} = k_2[C]$$

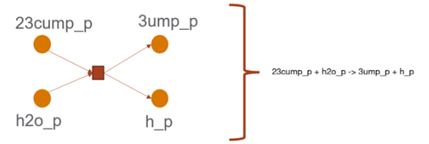


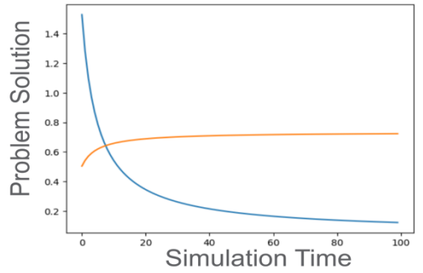
Fitting mathematical patterns into concrete CRs

a)

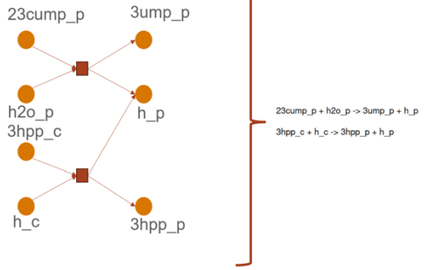


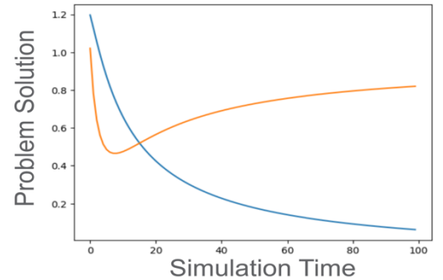
b) One reaction



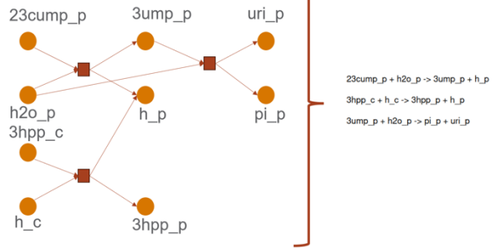


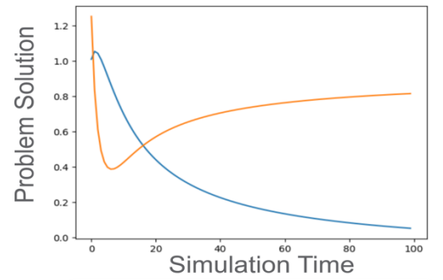
Two reactions



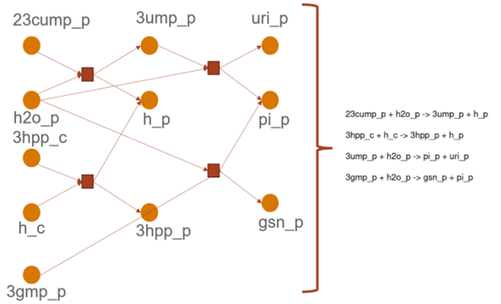


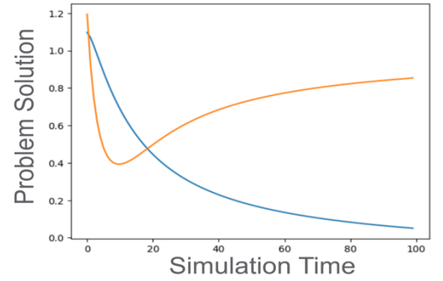
Three reactions



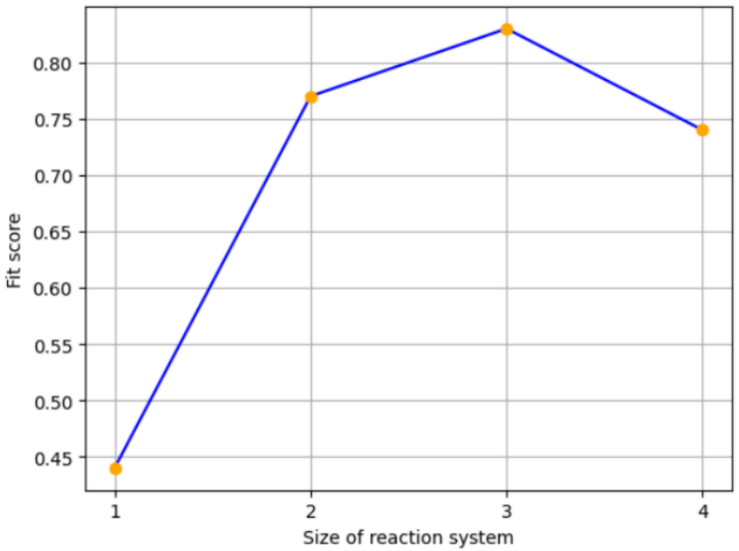


Four reactions

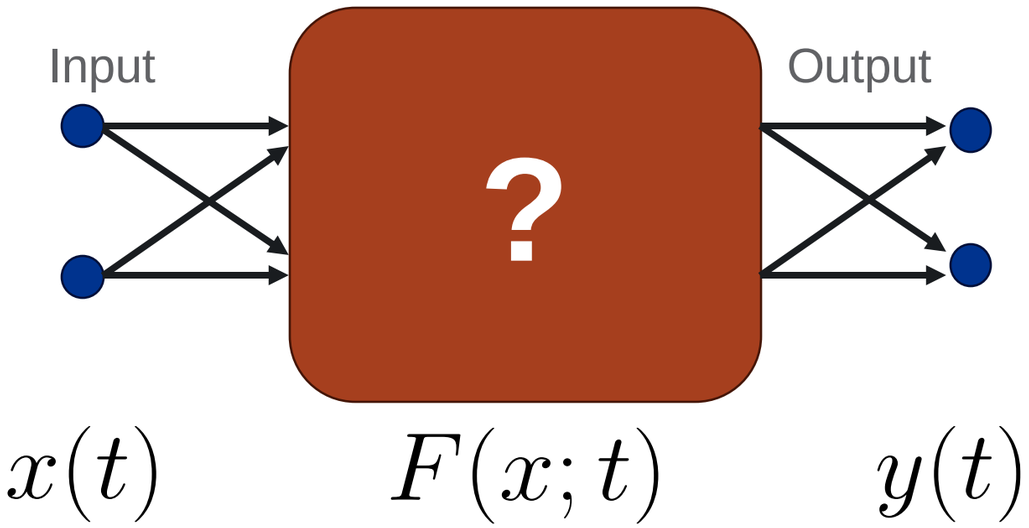




c)



Reservoir computing can harness real world systems for computation and forecasting



- Provide problem as input signal
- Map the signal onto a higher dimension physical system
- Dynamical system performs some process function
- Perform measurements of system states and form weighted combination of measurements
- Use linear least squares optimization of output weights

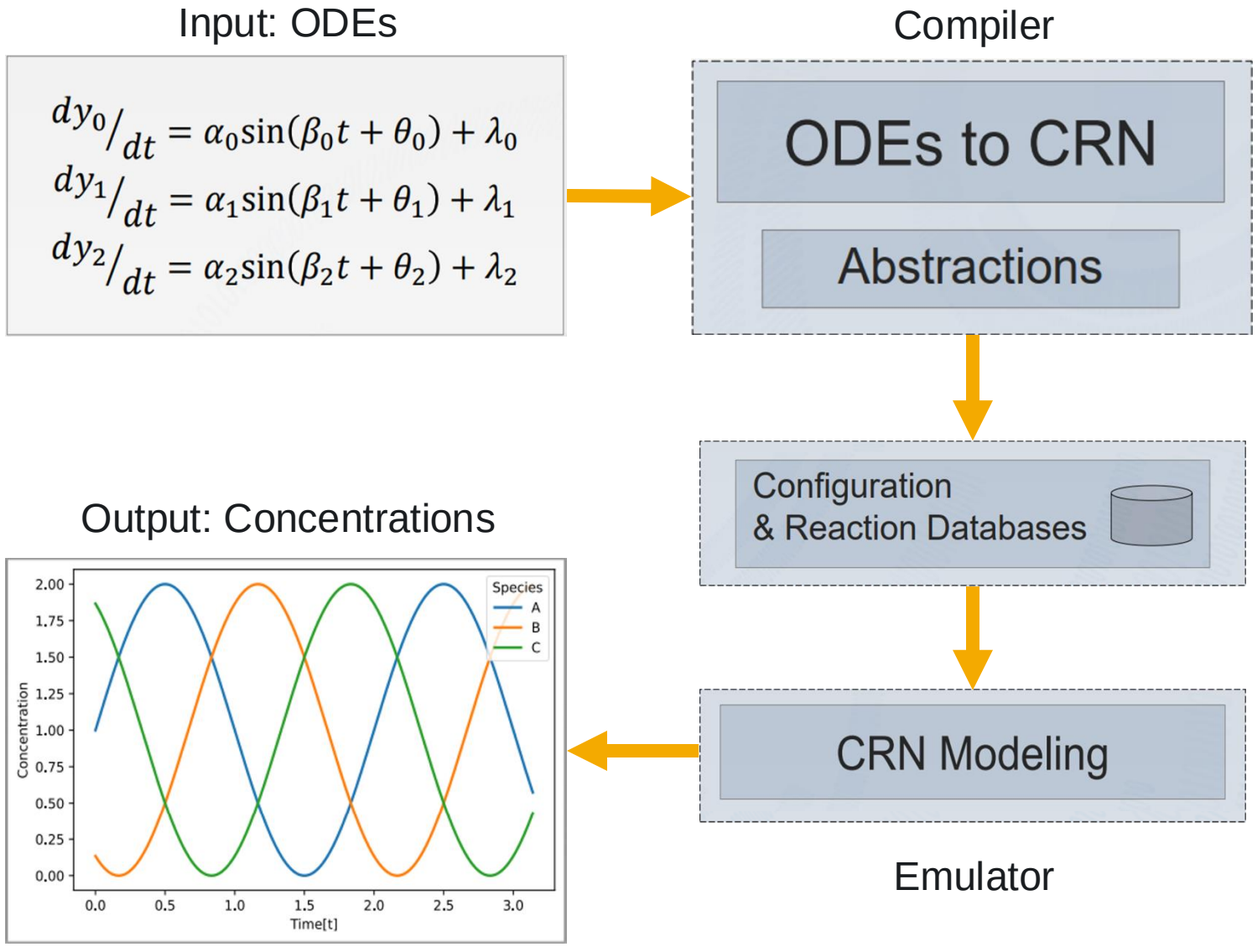
A range of dynamical systems have been considered:

- Optofluidic systems (Gao et al., 2022)
- Water waves (Maksymov and Pototsky, 2023)
- Ecosystems of Unicellular organisms (Ushio et al., 2023)
- Ferromagnetic materials (Everschor-Sitte et al., 2024)
- Formose reaction network (Baltussen et al., 2024)

ChemComp: A Compilation Framework for Computing with Chemical Reaction Networks

Approach

- Develop a compiler using MLIR [1] to capture ODE and realistic chemical and biological processes
- Create a compilation flow to convert ODEs into chemical operations from a curated database of known motifs
- Implement CRNCMs, leveraging the BiGG [2] database and respective mass action kinetics

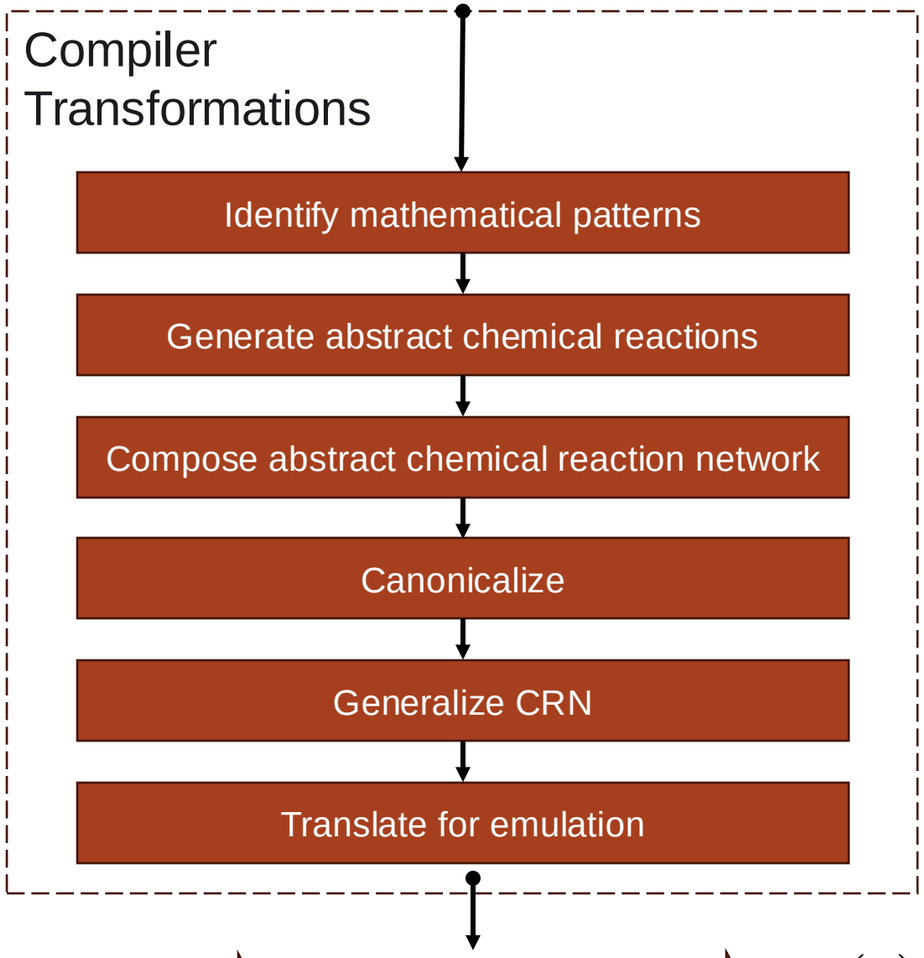


[1] C. Lattner, et al. "MLIR: Scaling compiler infrastructure for domain specific computation." 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO). IEEE, 2021

[2] Z. A. King, et al., "BiGG Models: A platform for integrating, standardizing and sharing genome-scale models, Nucleic Acids Research", Volume 44

Approach to compiler support

- Introduce a new dialect to represent chemical reactions
- Using MLIR, implement lowerings of mathematical patterns to abstract chemical reactions
- Incorporate concrete chemical reactions in the flow

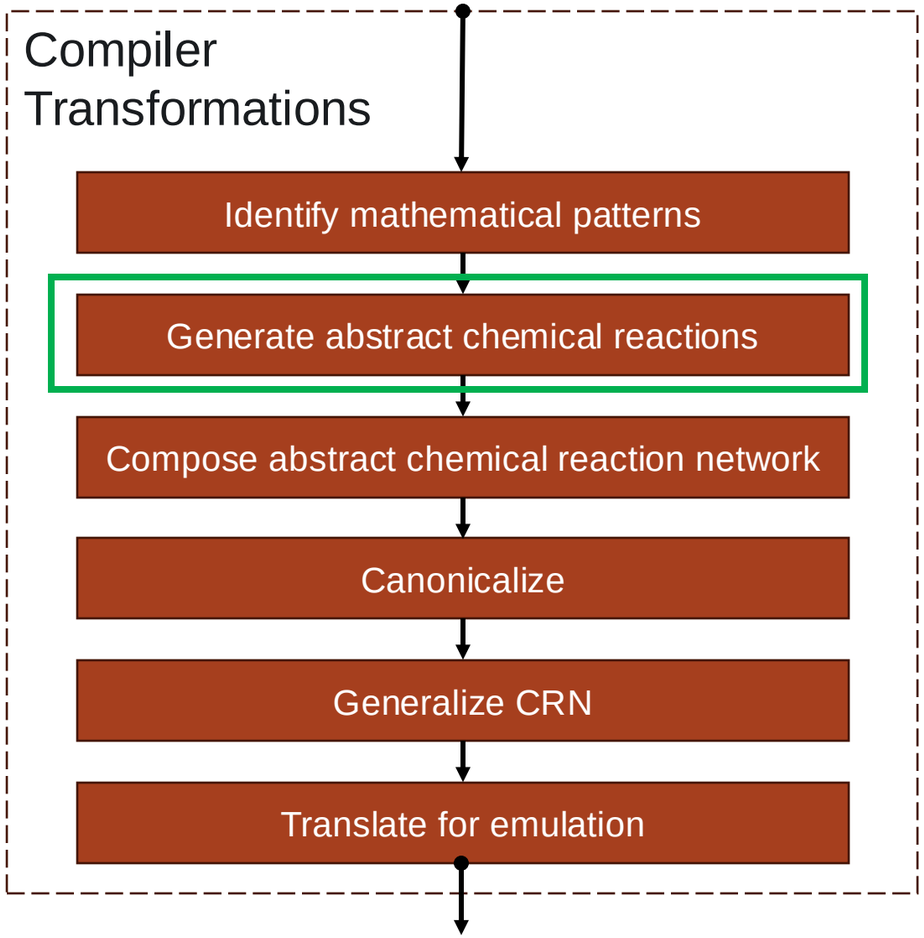


$\frac{dy}{dt} = f(t;\alpha)$ ➡ Mathematical patterns ➡ Abstract CRN ➡ Concrete CRN ➡ Device/Emulator ➡ $y(t)$

Approach to compiler support

```
// Define reaction 1: 2H2 + O2 -> 2H2O
// Reaction 1 adjusts concentrations of H2, O2, and H2O
acr.reaction @reaction_1(%h2: !acr.species,
                         %o2: !acr.species,
                         %h2o :!acr.species,
                         %rate_constant: f32)
  -> (!acr.species, !acr.species, !acr.species) {
 // Define the stoichiometry: 2H2 + O2 -> 2H2O
 %h2_stoich = acr.stoich_coeff -2 : f32
 %o2_stoich = acr.stoich_coeff -1 : f32
 %h2o_stoich = acr.stoich_coeff 2 : f32

 %h2_o, %o2_o, %h2o_o = acr.react(%h2, %o2, %h2o,
  %h2_stoich, %o2_stoich, %h2o_stoich,
  %rate_constant) : (!acr.species, !acr.species, !acr.species,
                     f32, f32, f32,
                     f32) -> (!acr.species, !acr.species, !acr.species)
 return %h2_o, %o2_o, %h2o_o
}
```

Define reaction $2H_2 + O_2 \rightarrow 2H_2O$



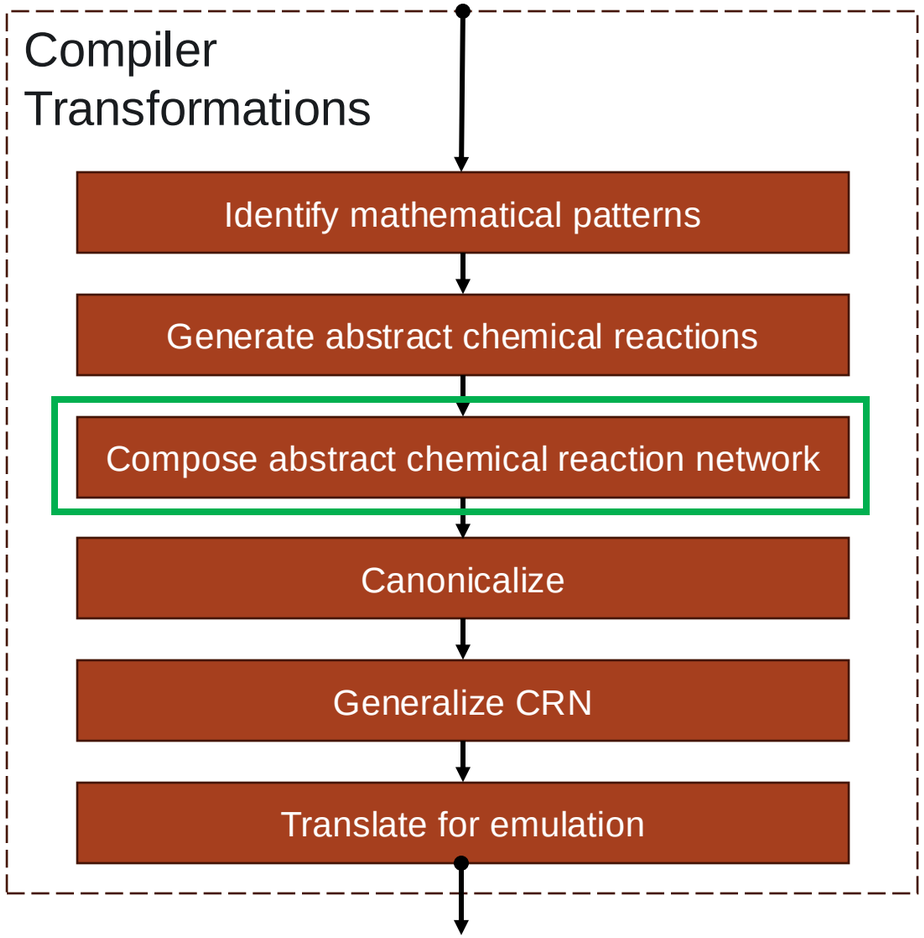
Approach to compiler support

```
// Declare reactions that were defined elsewhere
acr.reaction private @reaction_1( %h2: !acr.species, %o2: !acr.species,
   %h2o :!acr.species, %rate_constant: f32)
   -> (!acr.species, !acr.species, !acr.species)

acr.reaction private @reaction_2(%o2: !acr.species, %co: !acr.species,
                  %co2: !acr.species, %rate_2: f32)
   -> (!acr.species, !acr.species, !acr.species)

// Use reactions in a network
acr.crn @reaction_network(%h2: !acr.species, %o2: !acr.species,
                  %h2o : !acr.species, %co: !acr.species,
                  %co2: !acr.species,
                  %rate_1: f32, %rate_2: f32)
   -> (!acr.species, !acr.species, !acr.species, !acr.species, !acr.species) {
  // Instantiate reaction 1 within the network
  %h2_o, %o2_o, %h2o_o = acr.reaction_instance @reaction_1(%h2, %o2,
                                        %h2o, %rate_1)
   : (!acr.species, !acr.species, !acr.species, f32)
   -> (!acr.species, !acr.species, !acr.species)

  // Instantiate reaction 2 within the network
  %o2_o_final, %co_o, %co2_o = acr.reaction_instance @reaction_2(%o2_o, %co,
                                        %co2, %rate_2)
   : (!acr.species, !acr.species, !acr.species, f32)
   -> (!acr.species, !acr.species, !acr.species)

  // Return updated concentrations for all molecules: H2, O2, H2O, CO, and CO2
  return %h2_o, %o2_o_final, %h2o_o, %co_o, %co2_o
}
```

Instantiate CRN

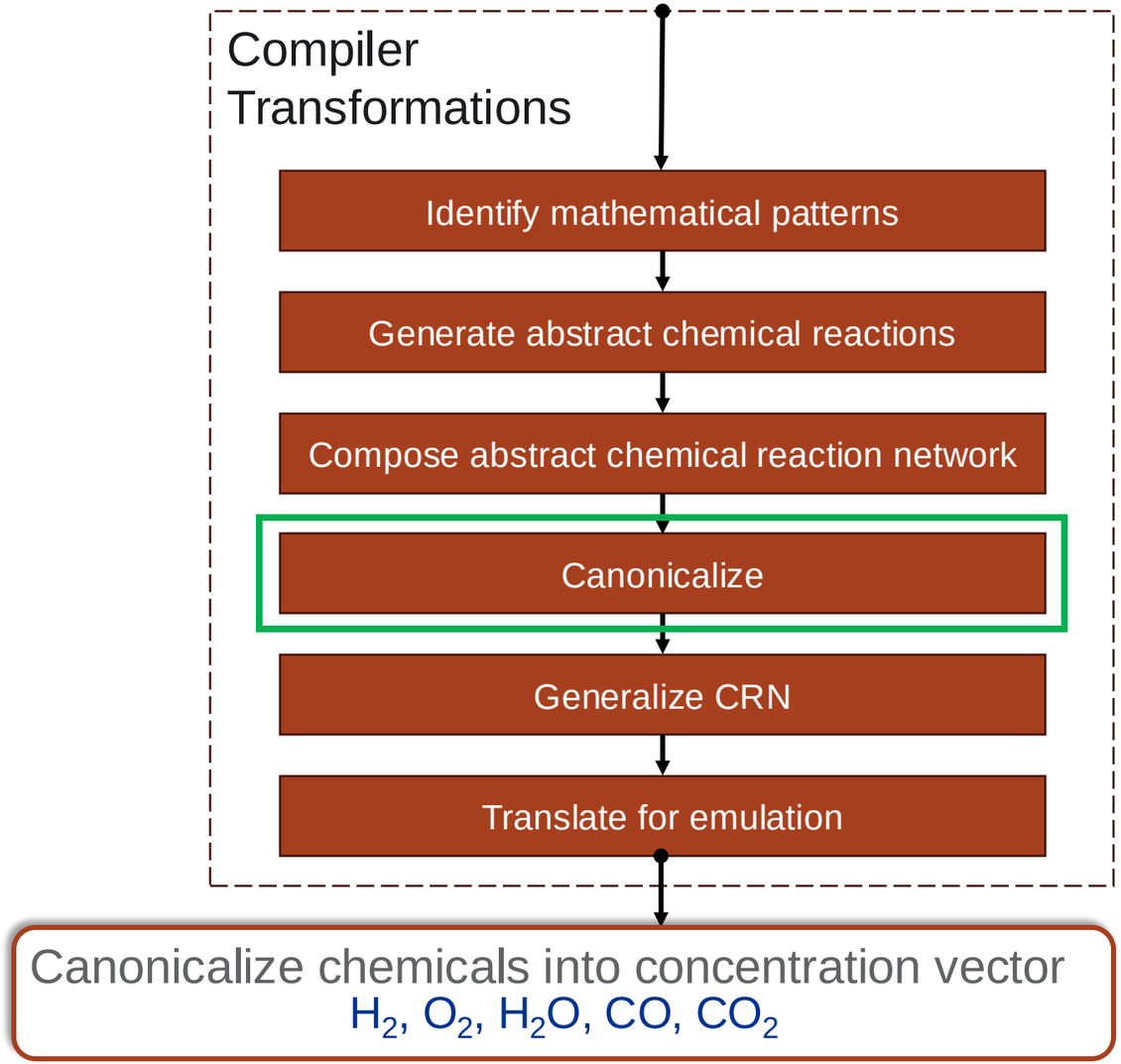
$2H_2 + O_2 \rightarrow 2H_2O$

$2CO + O_2 \rightarrow 2CO_2$



Approach to compiler support

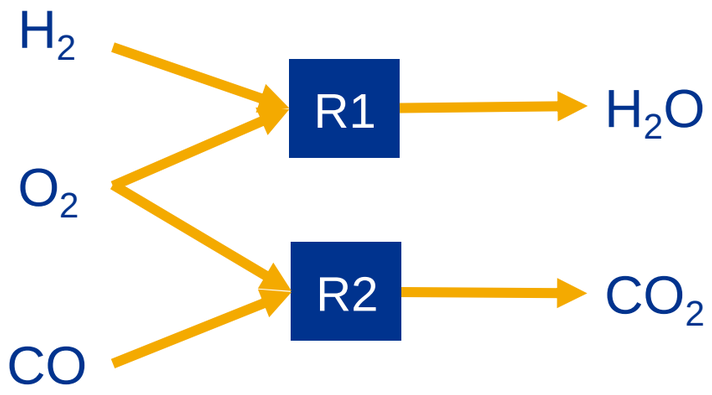
```
// Also declare @reaction_1 and @reaction_2 in canonical form
// Omitted here for conciseness...

// Define the CRN operation
acr.crn @reaction_network(%species_vec: vector<5x!acr.species>,
                          %rate_vec: vector<2xf32>)
  -> (vector<5x!acr.species>) {
 // For these 5 species: H2, O2, H2O, CO, CO2
 // Define the stoichiometry: 2H2 + O2 -> 2H2O
 //    and the stoichiometry: O2 + 2CO -> 2CO2

 // Select the subvector for reaction 1
 %subvector_r1 = vector.gather %species_vec, [0, 1, 2]
  : vector<5x!acr.species>, vector<3xindex>
 // Instantiate reaction 1 with the rate constant as an input
 %subvector_r1_out = acr.reaction_instance @reaction_1(%subvector_r1,
                                        %rate_vec[0])
  : (vector<3x!acr.species>, f32)
  -> vector<3x!acr.species>
 // Update the original species vector with the output subvector
 %species_vec_out1 = vector.scatter %species_vec, %subvector_r1_out, [0, 1, 2]
  : vector<5x!acr.species>, vector<3x!acr.species>, vector<3xindex>

 // Select the subvector for reaction 2
 %subvector_r2 = vector.gather %species_vec_out1, [1, 3, 4]
  : vector<5x!acr.species>, vector<3xindex>
 // Instantiate reaction 2 with the rate constant as an input
 %subvector_r2_out = acr.reaction_instance @reaction_2(%subvector_r2,
                                        %rate_vec[1])
  : (vector<3x!acr.species>, f32)
  -> vector<3x!acr.species>
 // Update the original species vector with the output subvector
 %species_vec_out = vector.scatter %species_vec_out1, %subvector_r2_out,
                          [1, 3, 4]
  : vector<5x!acr.species>, vector<3x!acr.species>, vector<3xindex>

 return %species_vec_out }
```

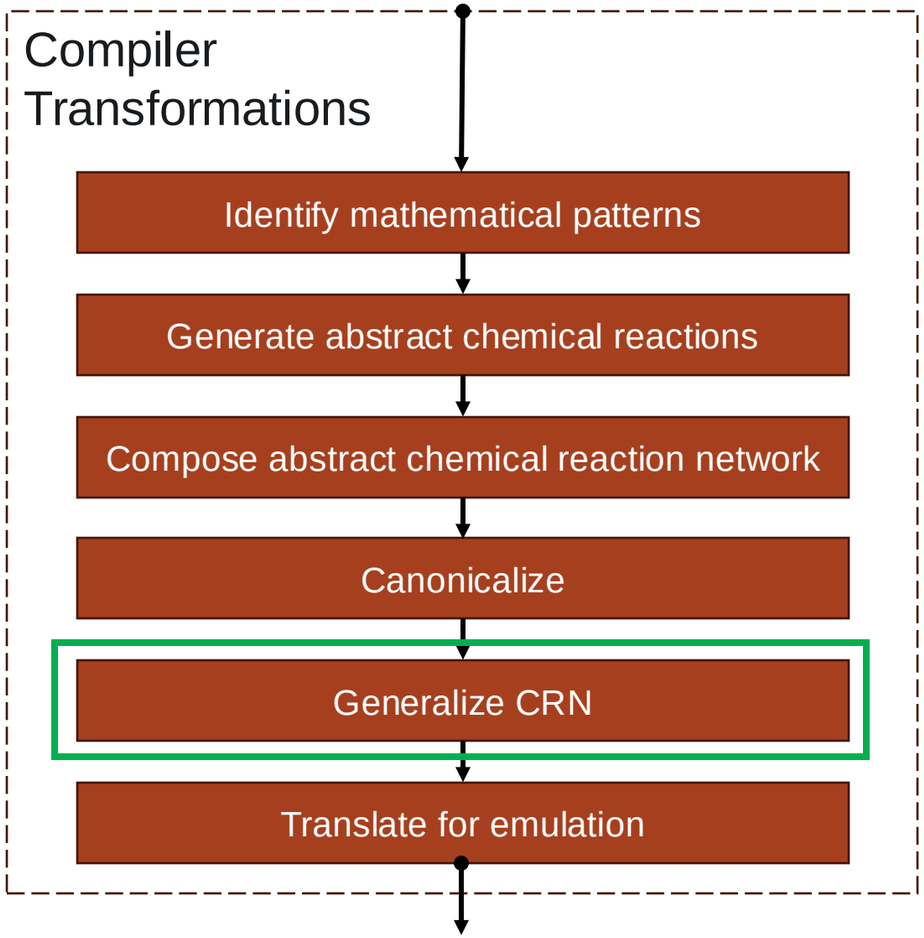


Approach to compiler support

```
// A generic operation that can represent any CRN
acr.crn_generic {stoich_matrix =
[
   // With the following order of species:
   // H2, O2, H2O, CO, CO2
   [-2,-1, 2, 0, 0], //2H2 + O2 -> 2H2O
   [ 0,-1, 0,-2, 2]  //2CO + O2 -> 2CO2
] : tensor<5x2x>
 } (%species_vec: vector<5x!acr.species>, %rate_vec: vector<2xf32>)
   -> (vector<5x!acr.species>)
```

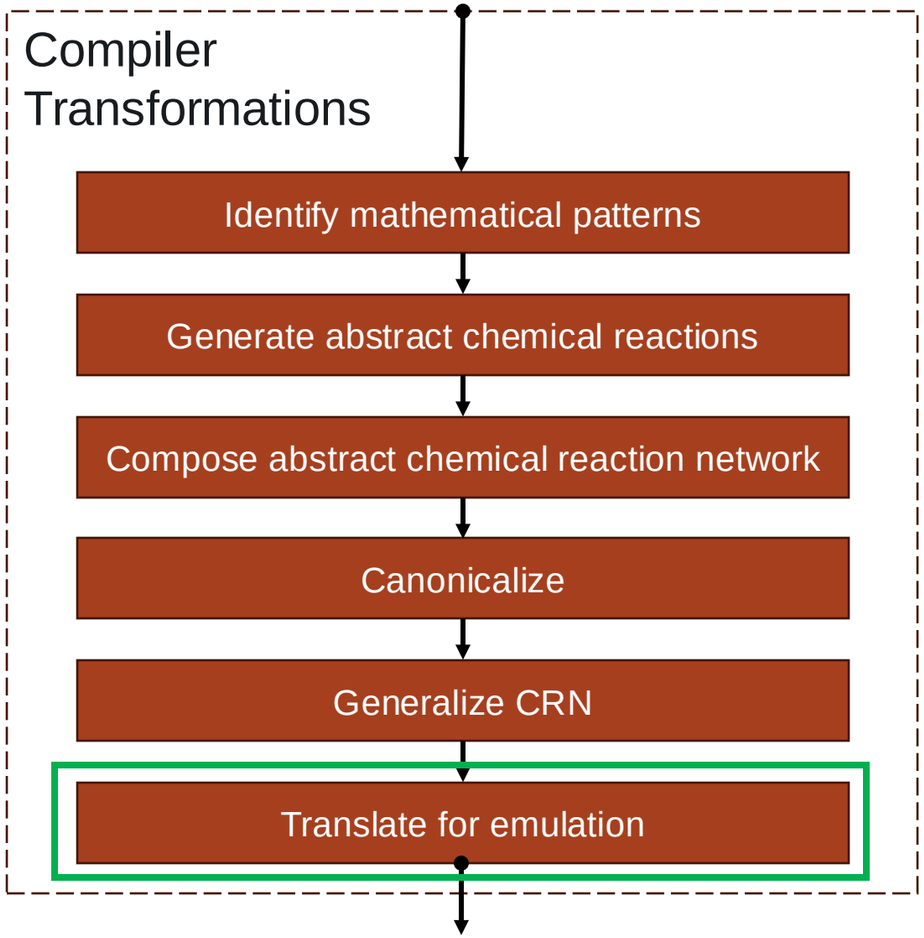
Represent with the generic CRN format





Approach to compiler support

```
{
  // Reactions are written in a list.
  // Species follow the order: H2, O2, H2O, CO, CO2
  "reactions": [
    {
      // 2H2 + O2 -> 2H2O
      // 2A + B -> 2C
      "species"   : ["A", "B", "C", "D", "E"],
      "reactants" : [2, 1, 0, 0, 0],
      "products"  : [0, 0, 2, 0, 0],
      "rate": 0.1,
      "name": "R1"
    },
    {
      // O2 + 2CO -> CO2
      // B + 2D -> 2E
      "species"   : ["A", "B", "C", "D", "E"],
      "reactants" : [0, 1, 0, 2, 0],
      "products"  : [0, 0, 0, 0, 2],
      "rate": 0.05,
      "name": "R2"
    }
  ],
  // Initial concentrations
  ”concentrations": {
     "A": 1, "B": 1, "C": 0, "D": 1, "E": 0
  },
  // List of species to track on the simulator,
  // this is the concentration that should implement our ODE
  "tracked_species": ["O2"]
}
```



Abstract reactions can be instantiated to concrete biochemical reactions

Problem → **Abstraction**

Mathematical pattern: x + y → Reaction(s)

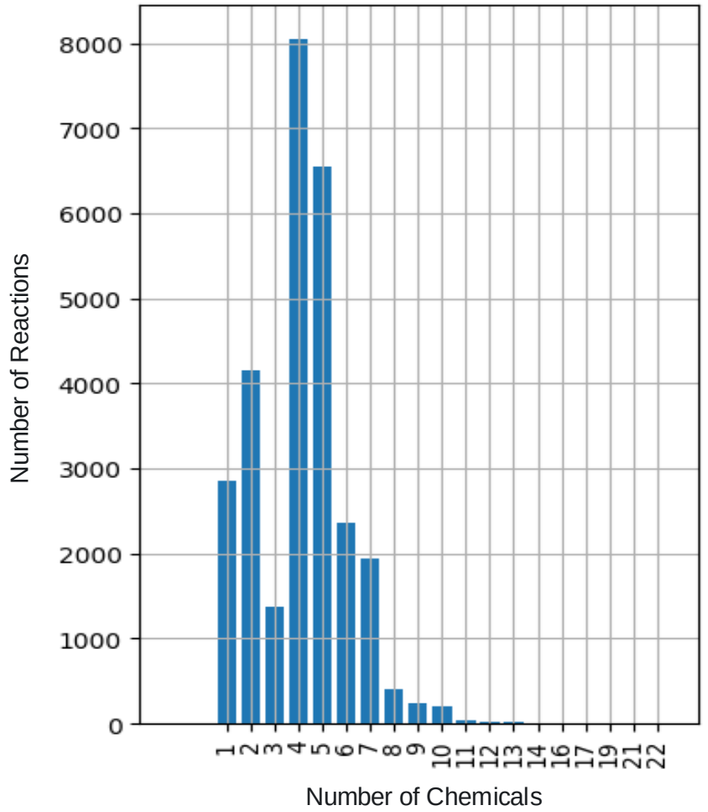
Scientific problem: f(x) → Reaction network

Instantiated ← **Transformed**

Instantiated	Transformed
	Generic form
$2H_2O \leftarrow 2H_2 + O_2$	$2C \leftarrow 2A + B$
$2CO_2 \leftarrow 2CO + O_2$	$2E \leftarrow 2D + B$

Species/Chemical concentrations

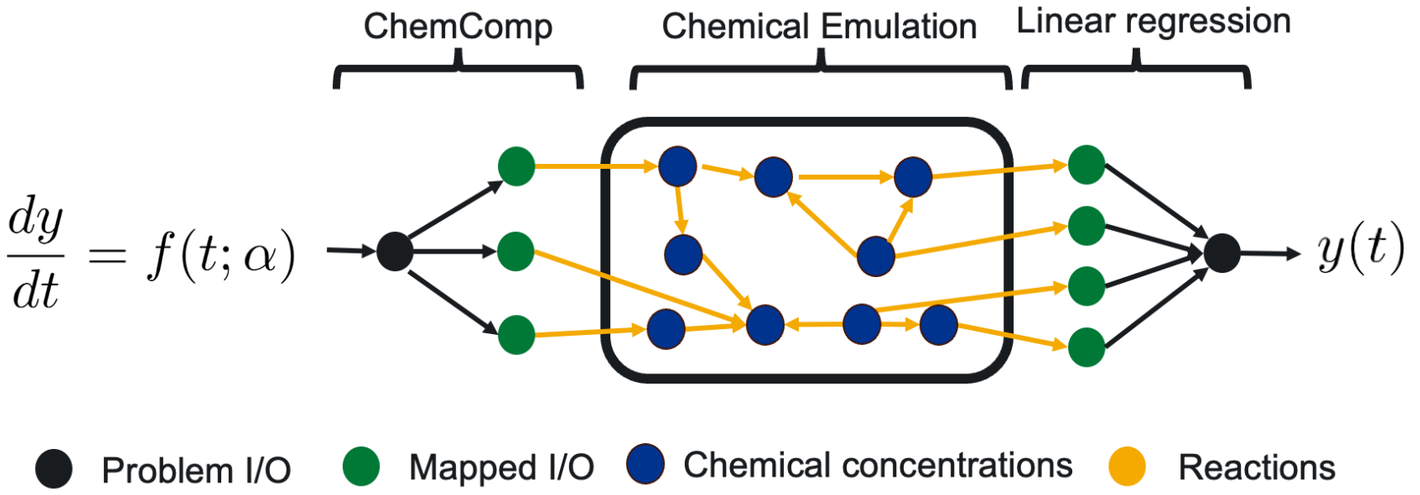
Reactions



Possible Abstract Reaction	Concrete Reaction (Motifs)
$A_2B + 2C_2 \rightleftharpoons 2A + BC_4$	$H_2S + 2O_2 \rightleftharpoons 2H + SO_4$
$2A_2B \rightarrow 2A_2 + B_2$	$2H_2O \rightarrow 2H_2\ O_2$
$AB_3 + B2C \rightarrow AB_4CB$	$NH_3 + H2O \rightarrow NH_4OH$

Emulator approach

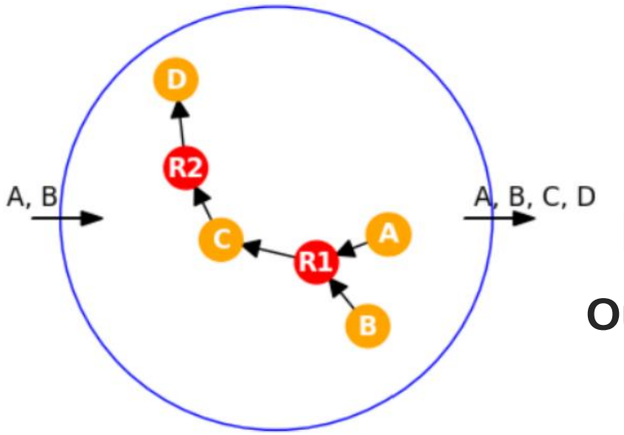
- Model device using Reservoir Computing
- Model reactions using continuum well-stirred system
- We can emulate both abstract and continuous reaction forms but are currently not considering noise



$A + B \rightarrow C$

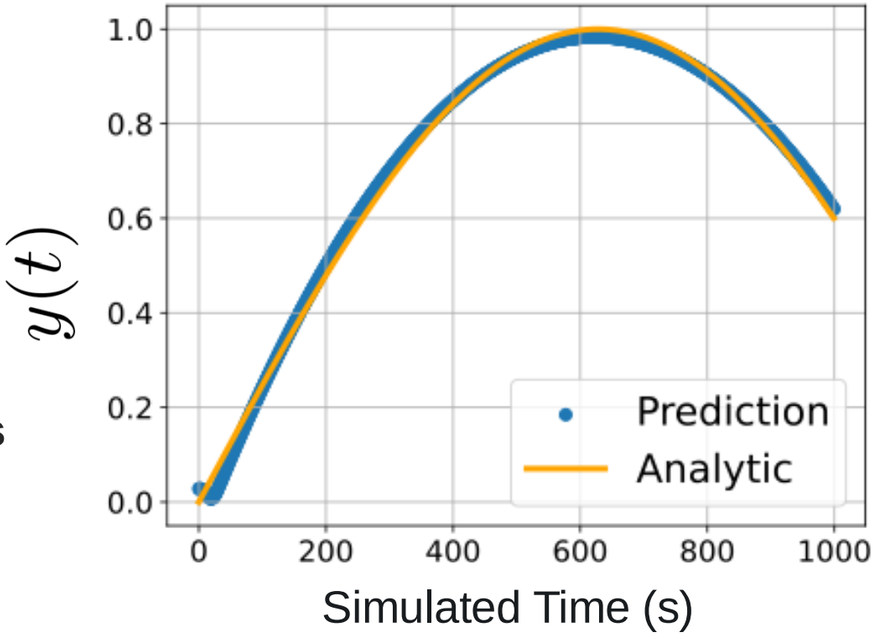
$C \rightarrow D$

Generic form



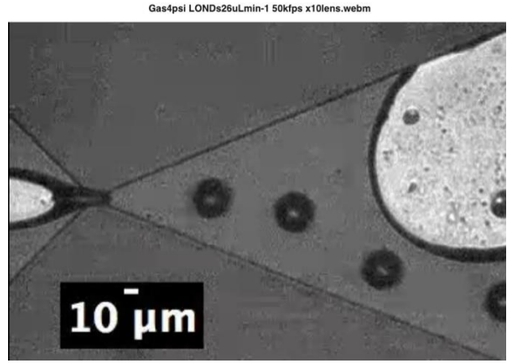
Modeled CRN

Outputs

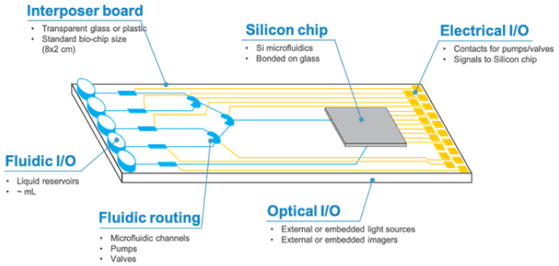


The future of bio-chemical computing devices: Scalability vs Feasibility

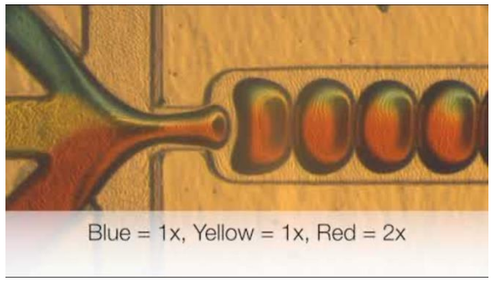
www.gaudi.ch/OpenDrop



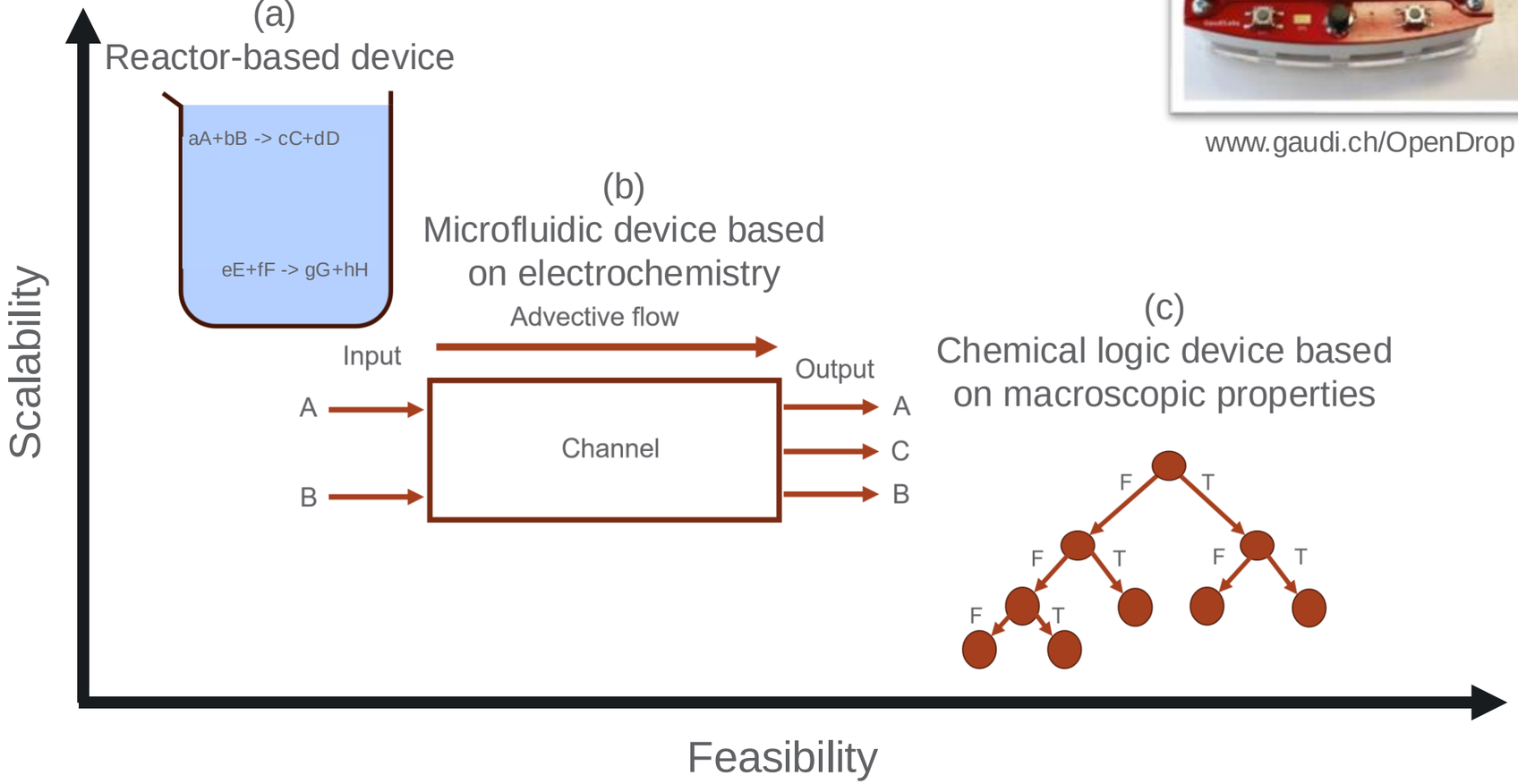
50kfps recording of droplets avoiding a bubble



Imec's lab on a chip



Content-, Size-, and rate- controlled microfluidic flow-focusing droplet generation



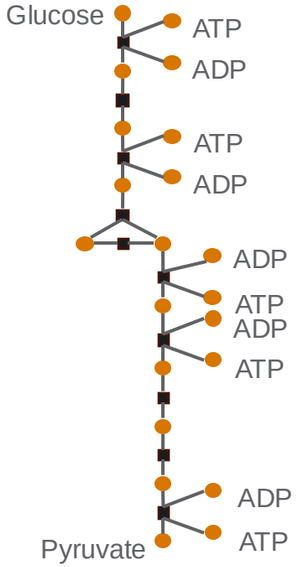
Closing the loop: Using compilers to study biochemical systems

Ordinary Differential Equations

Compiler

Chemical Reactions

Concrete CRN



Glycolysis pathway

CRN Dynamics

d(Glucose)dt =

.

.

.

.

.

d(Pyruvate)dt =

Complex ODE

Abstract CRN

$0 \rightleftharpoons U$

$0 \rightarrow V$

$2U + V \rightarrow 3U$

Sel'kov-Schnakenberg reaction system

ODE-representation

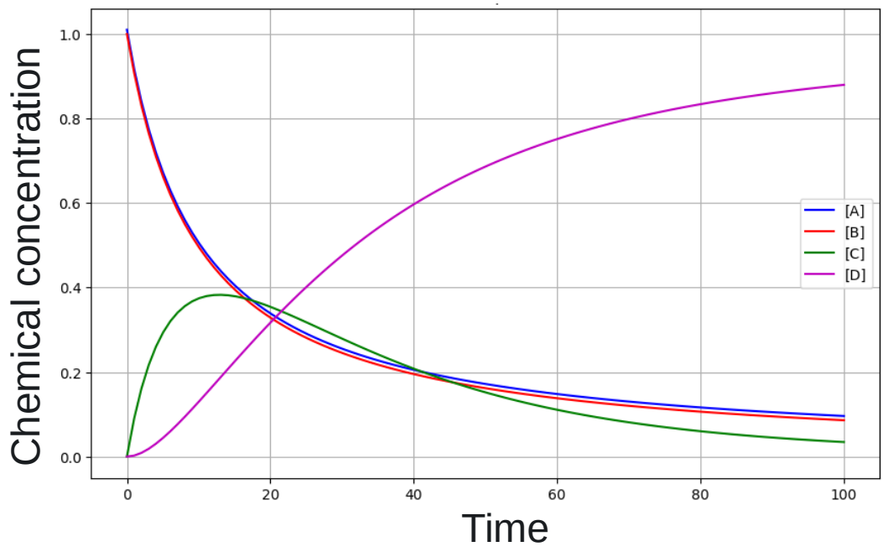
$$\frac{dy}{dt} = a - yx^2$$

$$\frac{dx}{dt} = yx^2 - x + b$$

Lower-dimension model

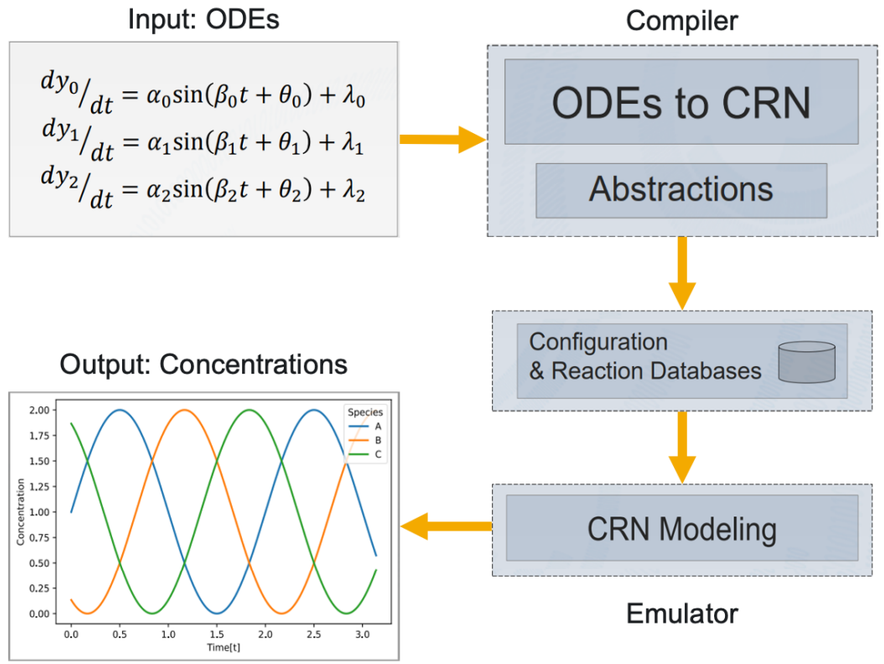
Take-home messages

- Chemistry can be utilized for ‘unconventional’ computation with a ‘chemical advantage’
- ChemComp is developing the necessary compilation and emulation frameworks
- Feasibility of a CRNCM needs to consider the physics and chemistry



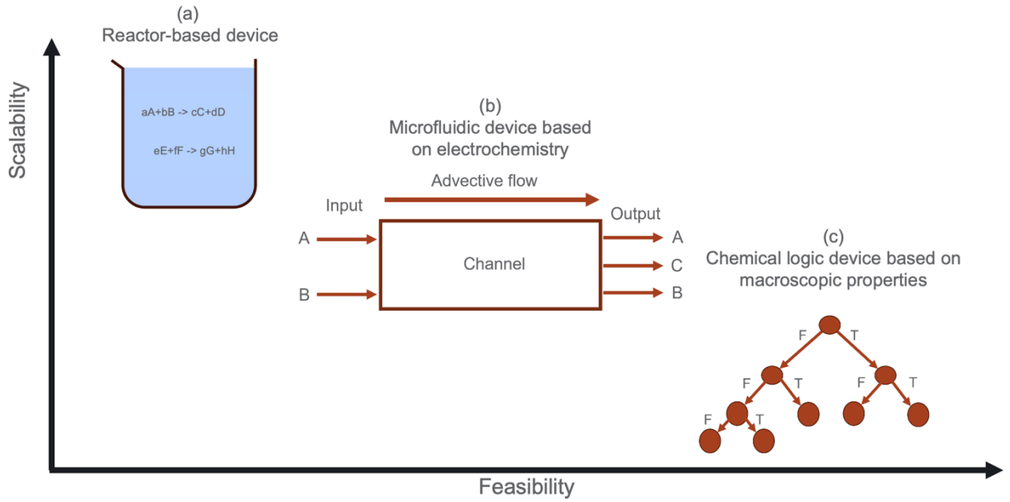
Take-home messages

- Chemistry can be utilized for ‘unconventional’ computation with a ‘chemical advantage’
- ChemComp is developing the necessary compilation and emulation frameworks
- Feasibility of a CRNCM needs to consider the physics and chemistry



Take-home messages

- Chemistry can be utilized for ‘unconventional’ computation with a ‘chemical advantage’
- ChemComp is developing the necessary compilation and emulation frameworks
- Feasibility of a CRNCM needs to consider the physics and chemistry



References

- Gao et al., 2022, “Thin liquid film as an optical nonlinear-nonlocal medium and memory element in integrated optofluidic reservoir computer”, 2022
- Ushio et al., 2023, “Computational capability of ecological dynamics”, Royal Society Open Science
- Maksymov and Pototsky, 2023, “Reservoir computing based on solitary-like waves dynamics of liquid film flows: A proof of concept”, Europhysics Letters
- Everschor-Sitte et al., 2024, “Topological magnetic and ferroelectric systems for reservoir computing”, Nature Reviews Physics
- Baltussen et al., 2024, “Chemical reservoir computation in a self-organizing reaction network“, Nature

Pacific Northwest
NATIONAL LABORATORY

Thank you

This research was supported by funding from the U.S. Department of Energy (DOE), Office of Advanced Scientific Computing Research (ASCR), as part of the ChemComp project under the Exploratory Research for Extreme-Scale Science (EXPRESS2023) program.

Email: antonino.tumeo@pnnl.gov