

# Kaemika app

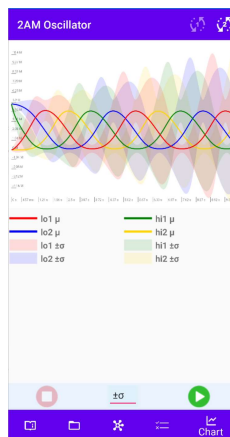
Integrating protocols and  
chemical simulation

Luca Cardelli, University of Oxford  
CMSB 2020-09-25



# Kaemika

/'kimika/



Search "Kaemika" in the app stores  
<http://lucacardelli.name/kaemika.html>

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



# *Chemical sublanguage and Simulation*



# CRN Models

The screenshot displays the Kaemika software interface. On the left, a code editor contains the following text:

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

parameter x1o <- uniform(0,1)
parameter x2o <- uniform(0,1)

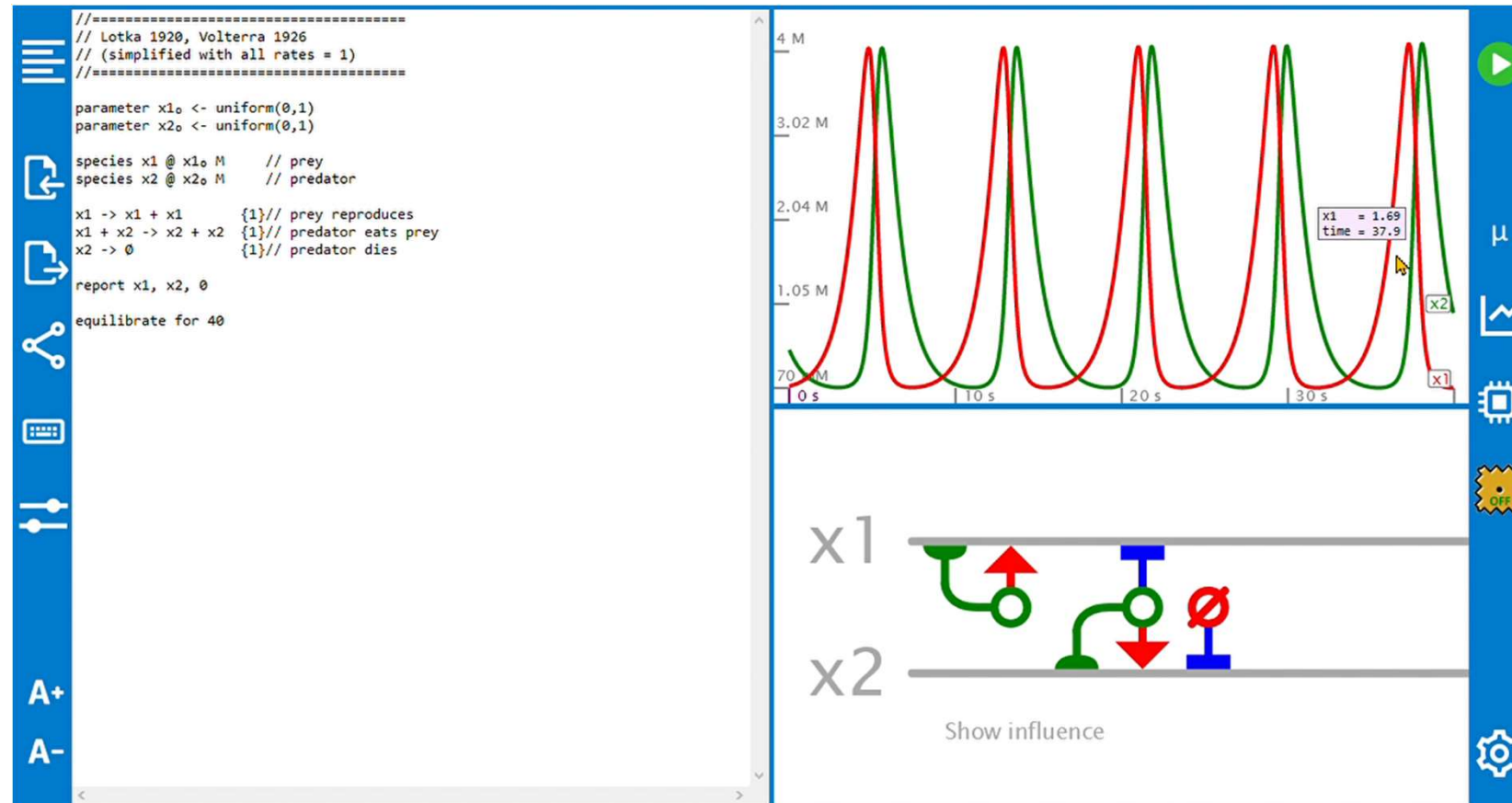
species x1 @ x1o M // prey
species x2 @ x2o M // predator

x1 -> x1 + x1 {1} // prey reproduces
x1 + x2 -> x2 + x2 {1} // predator eats prey
x2 -> 0 {1} // predator dies

report x1, x2, 0
equilibrate for 40
```

The main window on the right features the word "Kaemika" in a large, black, stylized font, with the phonetic transcription */'kimika/* below it in a blue, sans-serif font. The interface includes a blue sidebar on the left with icons for a menu, file operations, and zoom controls (A+, A-). The right sidebar contains a play button, a Greek letter mu (μ), a gear icon, a yellow "OFF" button, and another gear icon. The number "4" is visible in the bottom right corner of the slide.

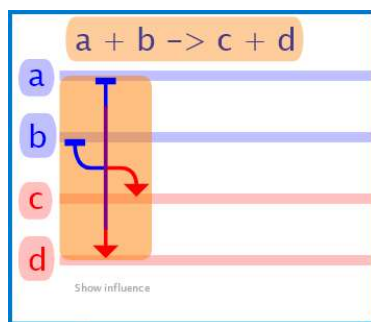
# CRN Models



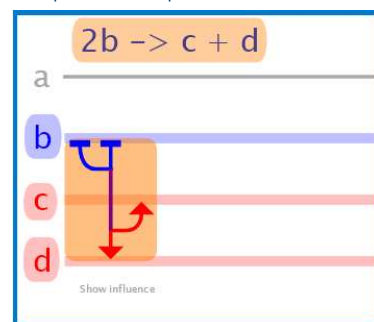
# Reaction scores (graphical representation of reaction networks)

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

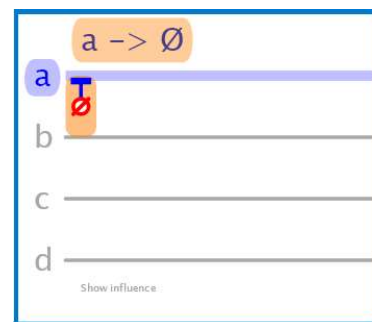
Reactants and products



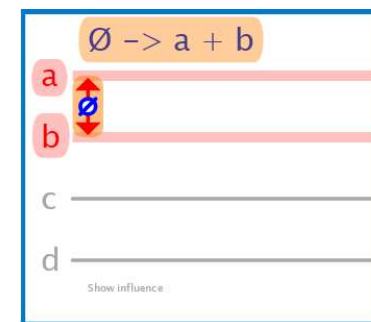
Repeated species



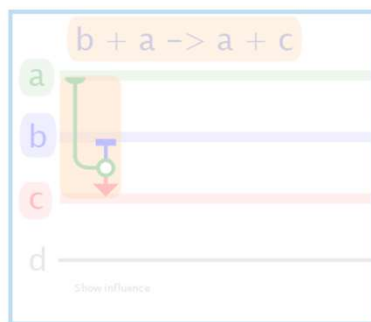
Reactants but no products



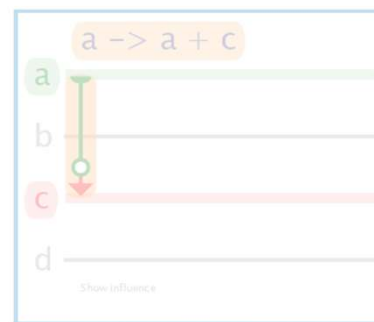
Products but no reactants



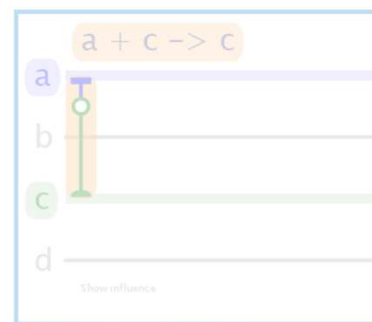
Catalyst



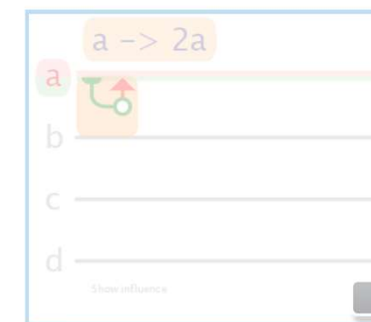
Catalyst but no reactants



Catalyst but no products



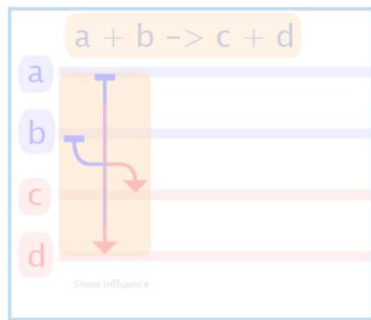
Autocatalyst



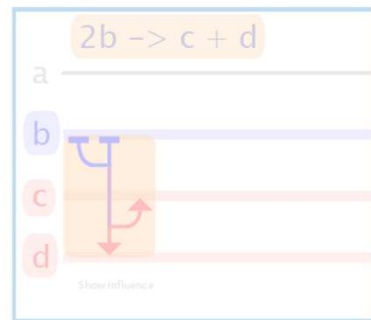
# Reaction scores (graphical representation of reaction networks)

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

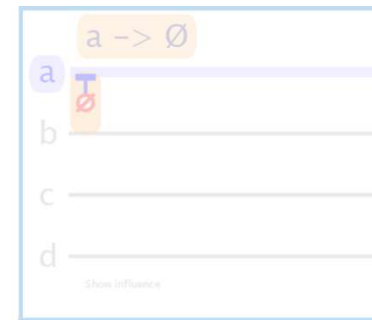
Reactants and products



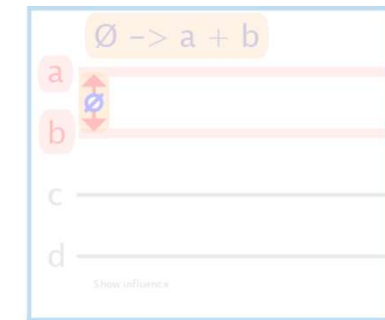
Repeated species



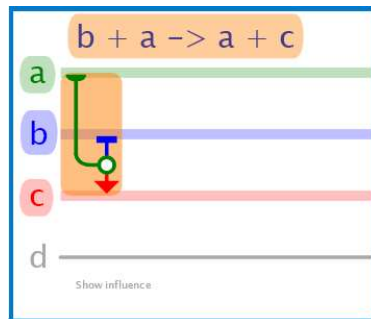
Reactants but no products



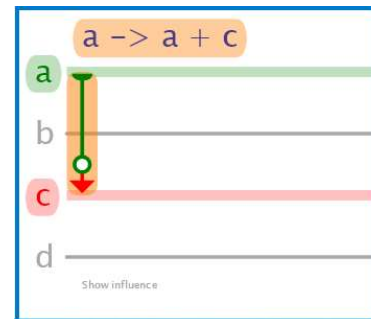
Products but no reactants



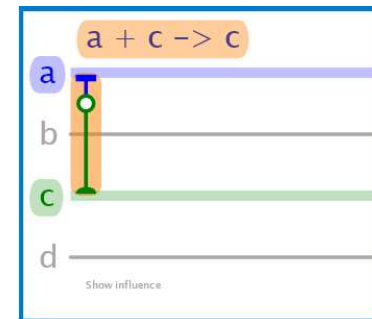
Catalyst



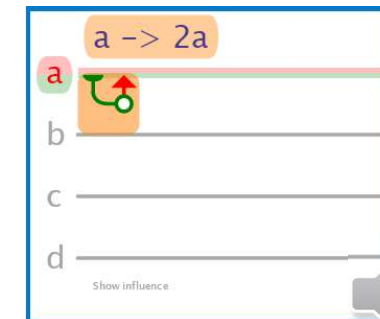
Catalyst but no reactants



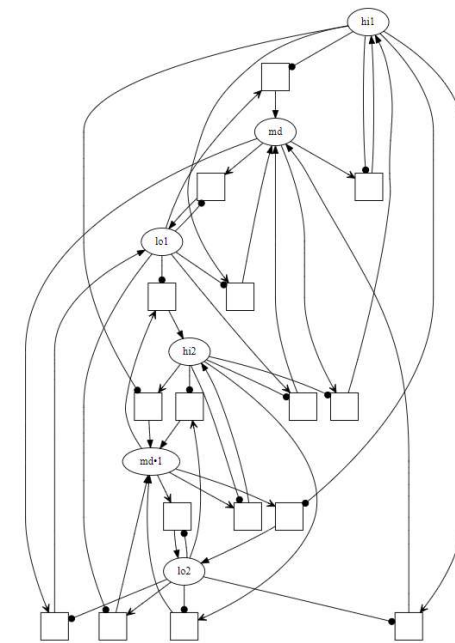
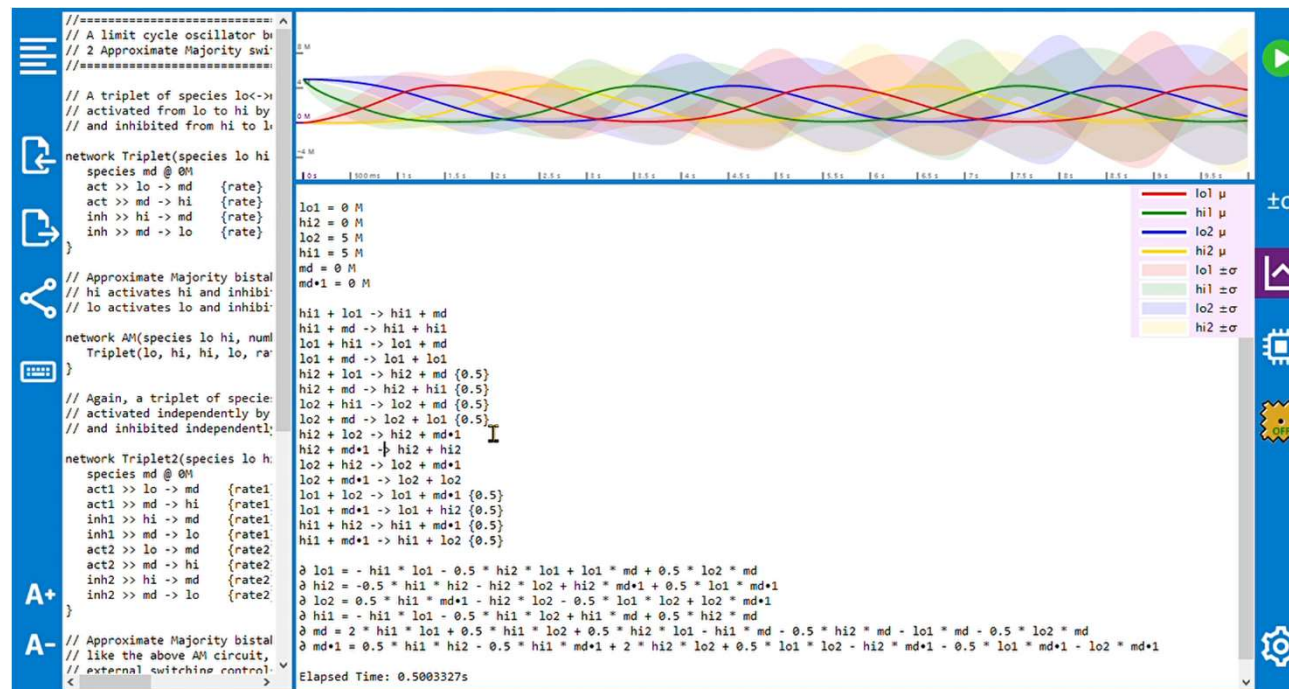
Catalyst but no products



Autocatalyst



# A larger reaction score



GraphViz



# The Modeling Language

- Models are generated by programs

Freely containing 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

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

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 -> ∅
    report predator
    yield predator
  end
}

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



**Kaemika**  
/'kimika/

The image shows a software interface with a code editor on the left and a presentation slide on the right. The code editor contains a NetLogo function named 'Predatorial' that simulates a predator-prey model. The function takes a number 'n' and returns the apex predator. The simulation starts with 5 levels of predation. The presentation slide displays the word 'Kaemika' in a stylized font, with its pronunciation given as /'kimika/.

# *Protocol sublanguage and Microfluidics*



# 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
  - Isolate species and reactions
- *Protocol Operations* (e.g. liquid handling)
  - Accept and produce samples
  - Accepted samples are *used up* (they can only be operated-on once)



# Mix and Split

```
//=====
// Example of Sample Manipulation
//=====

species {c}

sample A {1μL, 20C}
species a @ 10mM in A
amount c @ 1mM in A
a + c -> a + a
equilibrate A1 = A for 100


sample B {1μL, 20C}
species b @ 10mM in B
amount c @ 1mM in B
b + c -> c + c
equilibrate B1 = B for 100

split C,D = A1
dispose C

mix E = D, B1
a + b -> b + b

A+
equilibrate F = E for 1000
A-
dispose F
```

**Kaemika**  
/'kimika/

The right side of the interface features a large white area with the word "Kaemika" in a stylized black font and its phonetic transcription "/'kimika/" in blue below it. To the right of this area is a vertical blue bar containing several icons: a green play button, a Greek letter mu (μ), a white microchip icon, a yellow gear icon, and a white gear icon at the bottom.

# 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()
```



Cold Spring Harbor Protocols

[HOME](#) | [ABOUT](#) | [SUBJECT CATEGORIES](#) | [ARCHIVE](#) | [SUBSCRIBE](#)



Recipe

## Phosphate-buffered saline (PBS)

Reagent	Amount to add (for concentration 1× solution)	Final (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 <sub>2</sub> HPO <sub>4</sub>	1.44 g	10 mM	14.4 g	100 mM
KH <sub>2</sub> PO <sub>4</sub>	0.24 g	1.8 mM	2.4 g	18 mM

If necessary, PBS may be supplemented with the following:

CaCl <sub>2</sub> ·2H <sub>2</sub> O	0.133 g	1 mM	1.33 g	10 mM
MgCl <sub>2</sub> ·6H <sub>2</sub> O	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<sub>2</sub>O. Adjust the pH to 7.4 (or 7.2, if required) with HCl, and then add H<sub>2</sub>O to 1 L. Dispense the solution into aliquots and sterilize them by autoclaving for 20 min at 15 psi (1.05 kg/cm<sup>2</sup>) on liquid cycle or by filter sterilization. Store PBS at room temperature.

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



14

# 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

## OpenDrop

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



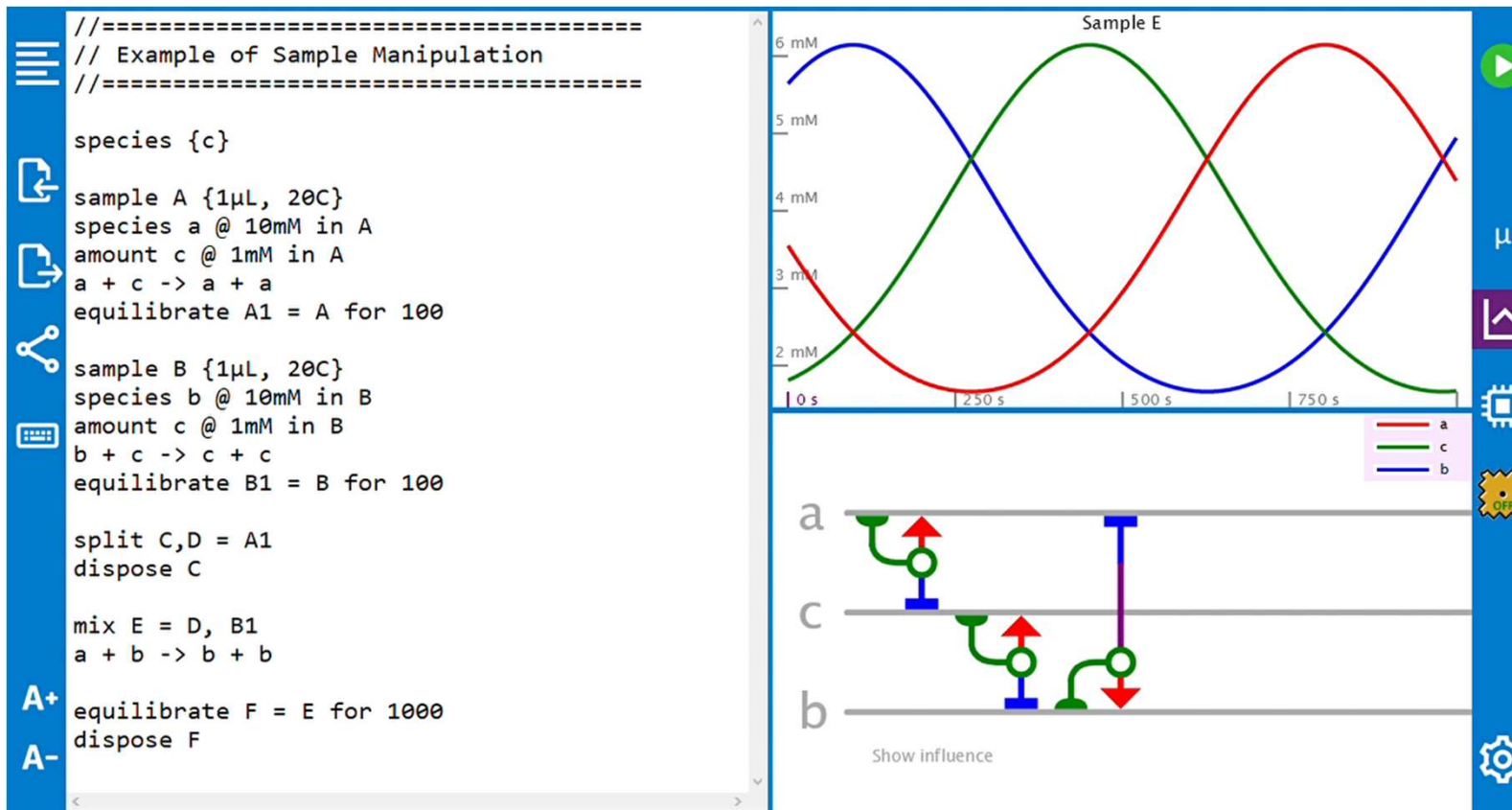
## Speed test

[https://www.youtube.com/watch?v=pSIs9L\\_h3Q0](https://www.youtube.com/watch?v=pSIs9L_h3Q0)





# Digital Microfluidics Compiler



# Digital Microfluidics Compiler

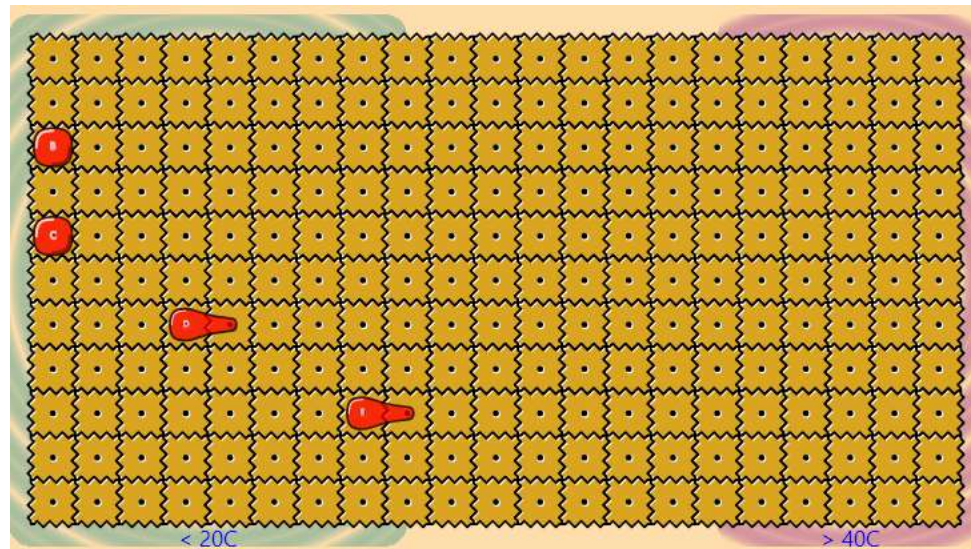
- Mix, split, equilibrate, dispose
- Automatic routing – no geometrical information
- Hot/cold zones

sample A {3 $\mu$ L, 20C}

split B,C,D,E = A

mix F = E,C,B,D

dispose F



# *The model AND the protocol*



# Extracting the Model and the Protocol

```
//=====
// Example of Sample Manipulation
//=====

species {c}

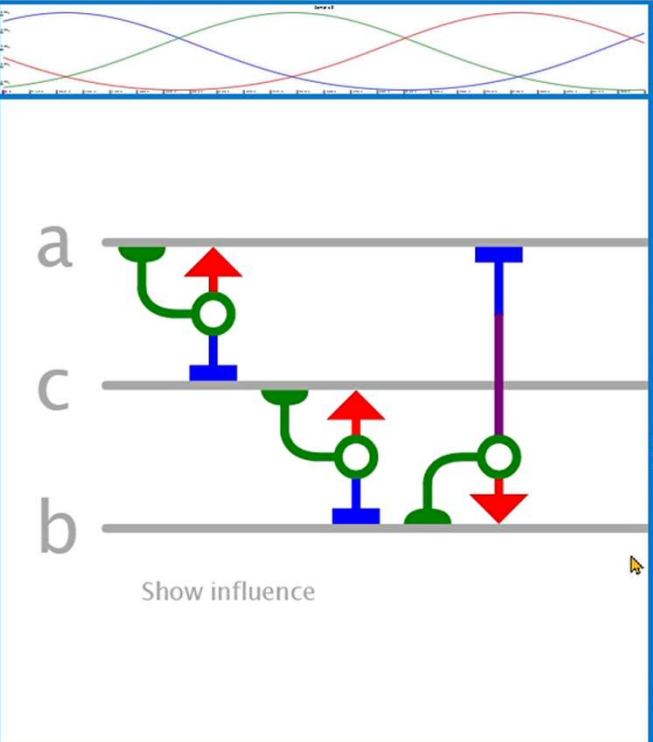
sample A {1μL, 20C}
species a @ 10mM in A
amount c @ 1mM in A
a + c -> a + a
equilibrate A1 = A for 100

sample B {1μL, 20C}
species b @ 10mM in B
amount c @ 1mM in B
b + c -> c + c
equilibrate B1 = B for 100

split C,D = A1
dispose C

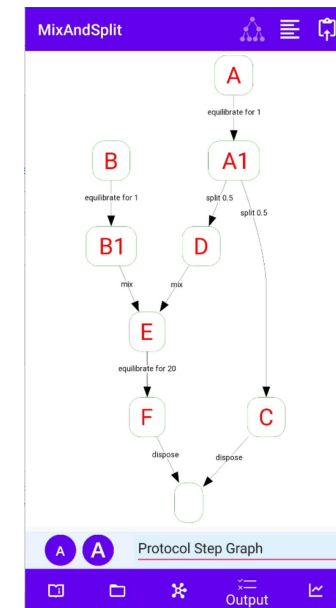
mix E = D, B1
a + b -> b + b

A+ equilibrate F = E for 1000
A- dispose F
```



Graph showing concentration profiles over time for species a, b, and c. The x-axis represents time and the y-axis represents concentration. Species a (red) starts high and decreases. Species b (green) starts low and increases. Species c (blue) starts low and increases, then decreases.

The protocol



# Conclusions

## **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.

### Integrated modeling

Of chemical reaction networks and protocols  
How the Kaemika app supports it

### Closed-loop modeling, experimentation and analysis

For complete lab automation  
To “scale up” the scientific method

### Thanks to:

Gold (parser)  
OSLO (simulator)  
C#/Xamarin (IDE)  
App reviewers

### No thanks to:

XAML (bug generator)

