La Cellule
un Calculateur Analogique Chimique

François Fages

Project-Team Lifeware

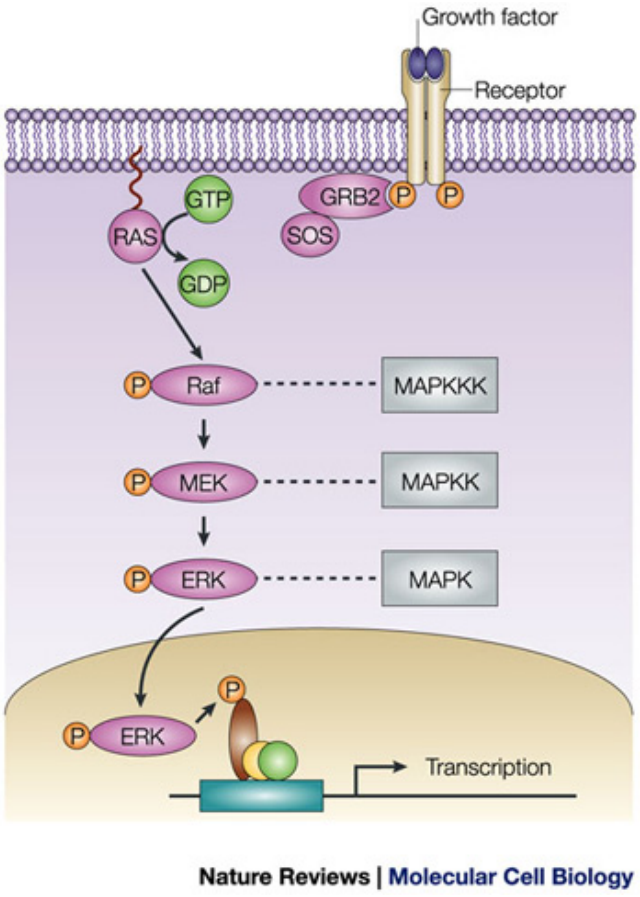
http://lifeware.inria.fr/

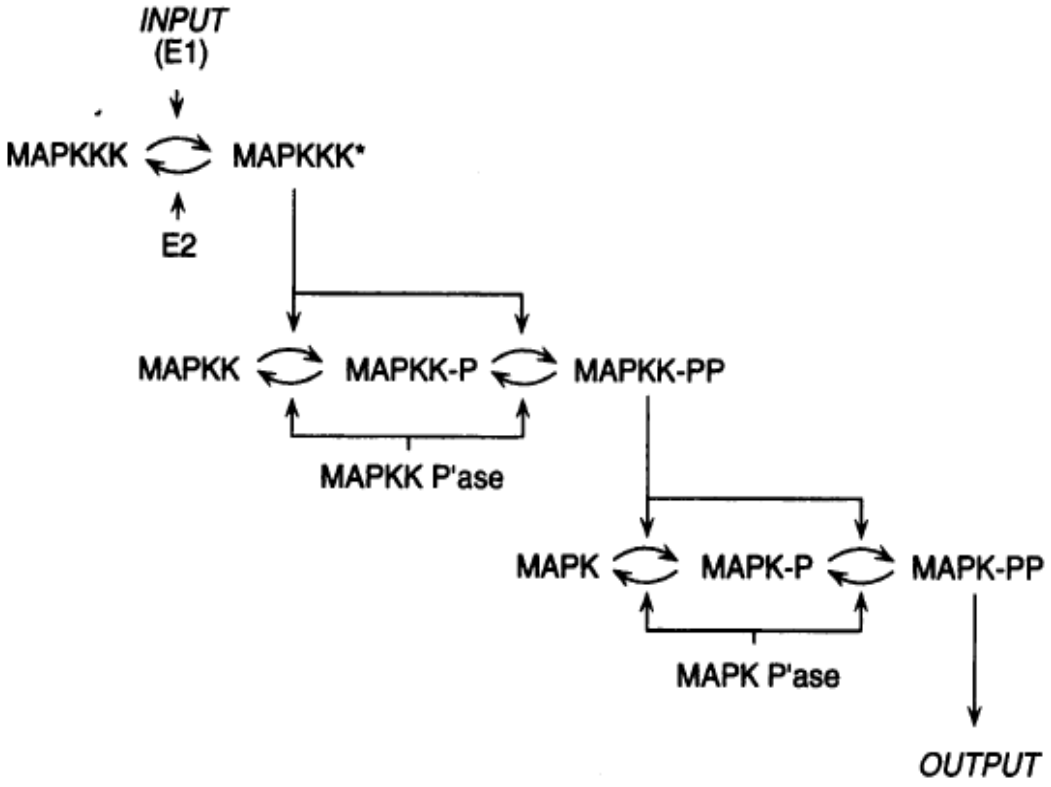
Institut National de Recherche en Informatique et Automatique

Inria Saclay – Ile de France

MAPK Signalling Cascade

MAPK Signaling Network: 30 reactions 18 species [Huang Ferrel PNAS 1996]

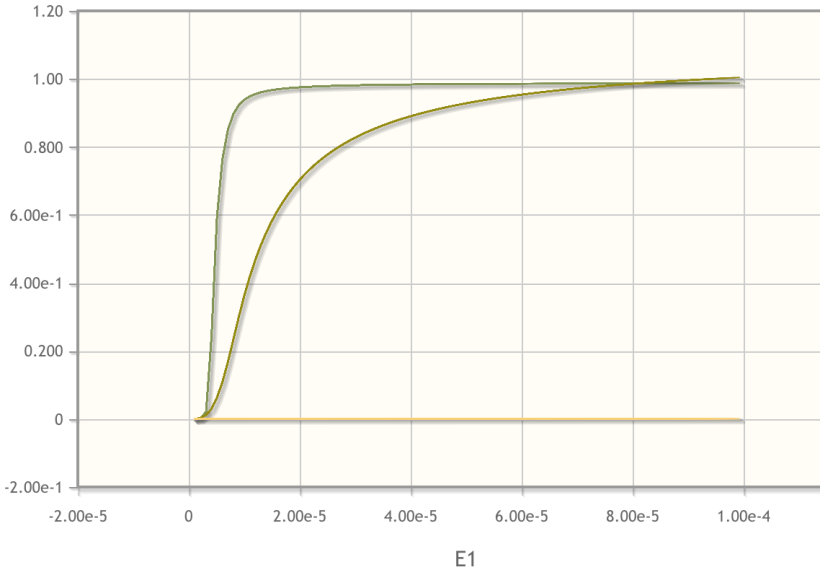




MAPK Input/Output Function

Dose-response diagrams alias Bifurcation diagrams

```
biocham: load(library:examples/mapk/mapk).
biocham: dose_response('E1',1.0e-6,1e-4,200).
```



MAPK responses as Hill function $\frac{x^n}{c + x^n}$

[Huang Ferrel 96 PNAS]

n ≈ 4.9 at 3rd level
n ≈ 1.7 at 2nd level
n = 1 at 1st level (Michaelis-Menten)

MAPK implements the function of an analog/digital converter in the cell.

How would one program $\frac{x^n}{c + x^n}$ with reactions ?

What does it mean to compute with real numbers ?

Computable Real Numbers and Functions

Classical definitions of computable analysis based on Turing machines

Definition. A real number *r* is computable if there exists a Turing machine with

Input: precision $p \in \mathbb{N}$

Output: rational number $q \in \mathbb{Q}$ with $|r-q| < 2^{-p}$

Examples. Rational numbers, limits of computable Cauchy sequences π, e, …

Definition. A real function $f:\mathbb{R} \to \mathbb{R}$ is computable if there exists a Turing machine that computes f(x) with an oracle for x.

Examples. Polynomials, trigonometric functions, …

Counter-examples. x=0, $\lceil x \rceil$ are not computable (undecidable on x=0.000…) discontinuous functions

Analog encoding e(w) of decision problems by f: accept w if $f(e(w)) \geq 1$ reject if ≤ -1

Analog Computer? Differential Analyzer [Bush 1931]

Underlying principles: Lord Kelvin, 1876

First ever built: Vannevar Bush, MIT, 1931

Applications: from gunfire control up to aircraft design

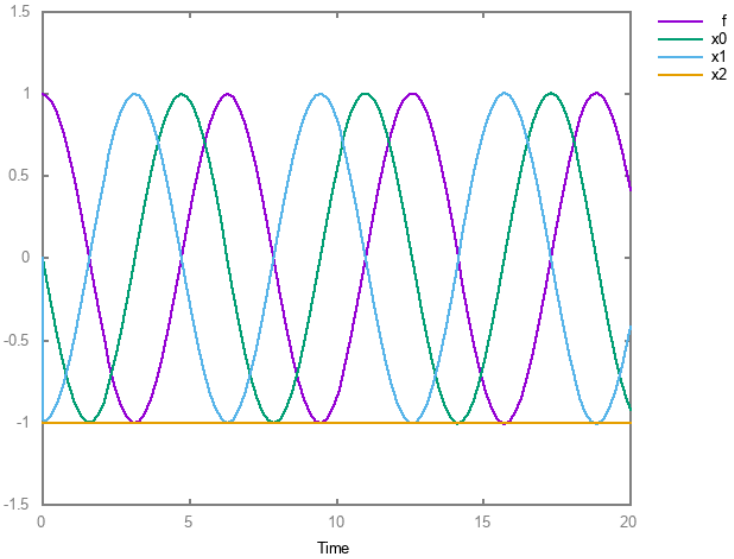
- Intensively used by the U.S. and Japanese armies during world war II
- Electronic versions from late 40s, used until 70s

General Purpose Analog Computer [Shannon 1941]

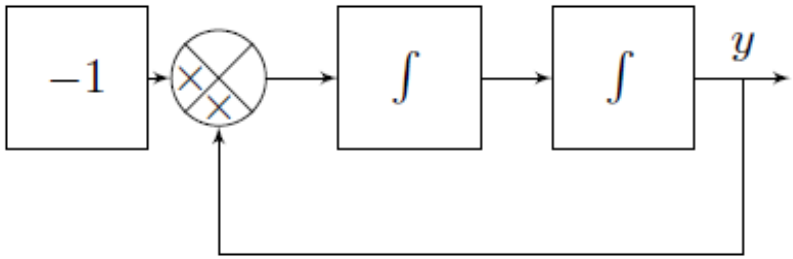
Shannon's formalization of the Differential Analyser by GPAC circuits

A time function if GPAC-generated if it is the output of some unit of a GPAC circuit built from:

1. Constant unit
2. Sum unit
3. Product unit
4. Integral $\int x\, dy$ unit



What does this GPAC circuit compute ?



$$y_1 = \frac{dy}{dt}$$

$$\frac{dy_1}{dt} = -y = y''$$

if $y(0) = 1, y_1(0) = 0$

$y(t) = cos(t) \quad y_1(t) = sin(t)$

CRN Implementation of GPAC Units

Mass action law kinetics reaction network with output concentration stabilizing on the result of the operation applied to the input concentrations

Positive constant units: molecular concentrations

Product unit $z = x.y$

$$x + y \xrightarrow{k.x.y} x + y + z$$
$$z \xrightarrow{k.z} \_$$
$$\frac{dz}{dt} = k(xy - z)$$
$$= 0 \text{ when } z = x.y$$

Sum unit $z = x + y$

$$x \xrightarrow{k.x} x + z$$
$$y \xrightarrow{k.y} y + z$$
$$z \xrightarrow{k.z} \_$$
$$\frac{dz}{dt} = k(x + y - z)$$
$$= 0 \text{ when } z = x + y$$

Time integral $z = \int x\, dt$ unit

$$x \xrightarrow{x} x + z$$
$$\frac{dz}{dt} = x$$
$$z = \int_0^T x\, dt$$

Polynomial ODE Initial Value Problems (PIVP)

Graça and Costa 2003's formalization of Shannon's GPAC

Definition. A real time function $f:R_+ \to R$ is GPAC-generable iff there exist a vector of polynomials $p \in R^n[R^n]$ and of initial values $y(0) \in R^n$
and a solution function $y:R_+ \to R^n$ such that $y'(t)=p(y(t))$ and $\mathbf{f(t)=y_1(t)}$

Closure properties:
f+g, f-g, f.g, 1/f, ,f ◦g, y s.t. y′ =f(y) are GPAC-generable if f, g are.

A GPAC-generated function must be analytic (locally convergent power series)
Famous analytic non-GPAC-generable functions [Shannon 41]

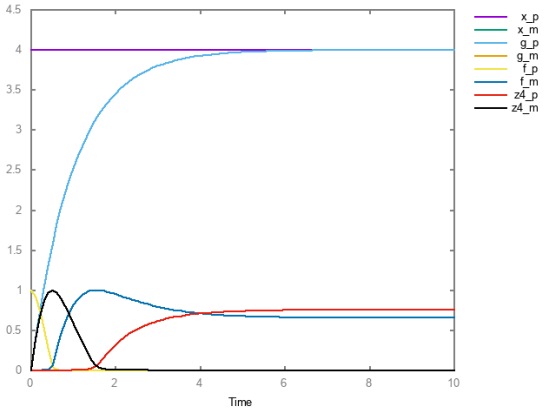
- Euler's Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ [Hölder1887]
- Riemann's Zeta function $\zeta(x) = \sum_{k=0}^{\infty} \frac{1}{k^x}$ [Hilbert]

But analytic functions are computable

PIVP-Computable Functions f(x)

Definition. [Graça Costa 03 J. Complexity] A real function f:R→R is PIVP-computable if there exists vectors of polynomials $p \in R^n[R^n]$ and $q \in R^n[R]$ and a function y: $R^n \to R^n$ such that $y'(t)=p(y(t))$, $\mathbf{y(0)=q(x)}$ and $\mathbf{|y_1(t)-f(x)|<y_2(t)}$ with $y_2(t) \geq 0$ decreasing for t>1 and $\lim_{t\to\infty} y_2(t) = 0$

Example. cos(4)



Reconciles Digital Computation and Analog Computation !

Theorem (analog characterization of Turing computability).

[Bournez Campagnolo Graça Hainry 07 J. Complex]]

A real function is computable (by Turing machine) iff it is PIVP-computable.

Analog characterization of Ptime

Time in ODE is a bad measure of complexity

- Exponential speedup by changing time variable $t' = e^t$
- But price to pay in the amplitude of t'

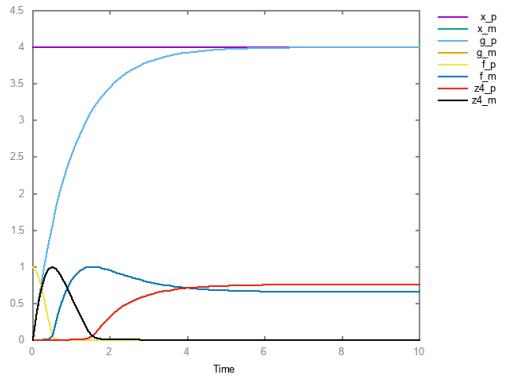
A computational complexity measure should combine time and space-amplitude

- length in the n dimensions of the trajectory to compute the result

Theorem [Pouly PhD thesis 2015, Bournez Graca Pouly 16 ICALP]

A real function is computable in P iff it is PIVP-computable with a trajectory of polynomial length (i.e. polynomial time and polynomial amplitude)

Reconciles Digital and Analog Ptime Complexity !



Turing Completeness of Continuous CRN?

- Mass action law kinetics
  - polynomial ODEs
  - PIVP computation by simulation
- Molecular concentration are positive real values
  - Restriction to positive dynamical systems ?
- Elementary reactions with at most two reactant
  - Restriction PIVP of degree at most 2 ?

Strong Turing Completeness of Continuous Chemical Reaction Networks and Compilation of Mixed Analog-Digital Programs, CMSB 2017

François Fages, Guillaume Le Guludec

Olivier Bournez [2], Amaury Pouly [2]

[2] *LIX, Ecole Polytechnique, Palaiseau, France*

Turing Completeness of Continuous CRNs 1/3

Lemma (positive systems) Any PIVP-computable function can be encoded by a PIVP of double dimension on R^+, preserving polynomial length complexity.

Proof. Encode $y_i \in R$ by y^-_i $y^+_i \in R^+$ such that $\mathbf{y_i = y^+_i - y^-_i}$ at each time

(encoding used in [Oishi Klavins 2011] for linear I/O systems)

Let $\underline{p}_i(y^+_1, y^-_1, \ldots, y^+_n, y^-_n) = p_i[y = y^+_i - y^-_i]$ and $\underline{p}_i = \underline{p}^+_i - \underline{p}^-_i$

$y^+_i{}' = \underline{q}^+_i - f_i\, y^+_i\, y^-_i$ $\qquad$ $y^+_i(0) = \max(0, y_i(0))$

$y^-_i{}' = \underline{q}^-_i - f_i\, y^+_i\, y^-_i$ $\qquad$ $y^-_i(0) = \max(0, -y_i(0))$

Where $f_i = q^+_i + q^-_i$ are positive coefficient polynomials $f_i \geqq \max(q^+_i, q^-_i)$

- Fast annihilation reactions: $y^+_i + y^-_i \xrightarrow{f_i} \_$
- n-ary catalytic synthesis reactions for each monomial $\underline{m}^+_{i,j}$ in $\underline{p}^+_i$, $\underline{m}^-_{i,j}$ in $\underline{p}^-_i$:

$$M_{i,j}{}^+ \xrightarrow{\underline{m}^+_{i,j}} y^+_i + M_{i,j}{}^+$$

$$M_{i,j}{}^- \xrightarrow{\underline{m}^-_{i,j}} y^+_i + Mi_{,j}{}^-$$

Turing Completeness of Continuous CRNs 2/3

Lemma (quadratic systems) [Carothers Parker Sochacki Warne 2005]
Any PIVP can be encoded by a PIVP of degree ≤ 2.

Proof. Introduce variable $v_{i1,\ldots,in}$ for each possible monomial $y_1^{i1}\ldots y_n^{in}$

We have $y_1 = v_{1,0\ldots,0}$, $y_2 = v_{0,1,0\ldots,0}$,…

y'_i is of degree one in $v_{i1,\ldots,in}$

$v'_{i1,\ldots,in} = \sum_{k=1}^{n} i_k \, v_{i_1,\ldots,i_{k-1},\ldots,i_n} \, y'_k$ is of degree at most 2.

i*.e. trade high dimension for low degrees.*

(yet algorithm of possibly exponential complexity)

Turing Completeness of Continuous CRNs 3/3

Theorem (Turing completeness of continuous CRNs) [F Le Guludec Bournez Pouly CMSB 2017]

Any computable function over the reals can be computed by a continuous CRN over a finite set of molecular species (no polymerization, no locations)

Proof: By previous lemmas, any PIVP-computable function can be encoded by a PIVP of degree at most 2 with positive variables. A positive PIVP of degree at most 2 can be represented by an elementary CRN with at most 2 reactants per reaction.

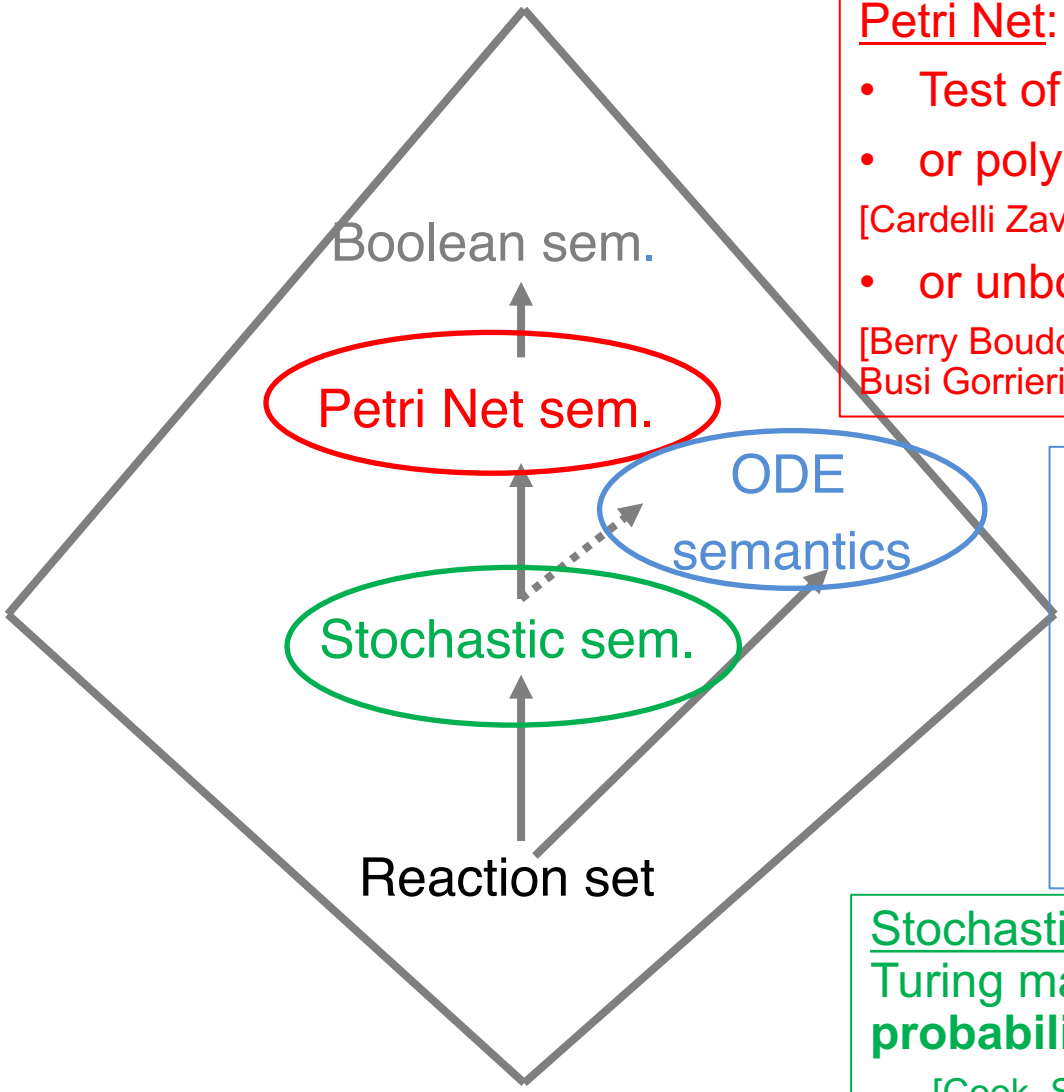
In this view, the (protein) concentrations are the information carriers.

The programs of a cell are implicitly defined by

- the set of all possible reactions with the proteins encoded in its genome
- and the chemicals of the environment.

Program change is determined by gene expression (= metaprogram).

Turing Completeness of CRNs



Petri Net: **Not Turing complete** without

- Test of absence (Petri net inhibitor arc)
- or polymerisation reactions

[Cardelli Zavatero MSCS 2010, Cook et al 2009]

- or unbounded nested membranes

[Berry Boudol CHAM 1994, Paun Rozenberg TCS 2002, Busi Gorrieri CMSB 2005]

DIfferential CRN: **Universality but for non uniform computability**: ∀function ∀input ∃circuit computing the result [Magnosco 1997 Phys Rev Helmfelt Weinberger PNAS 1991]

Strong Turing completeness

∀function ∃circuit computing ∀input [F. Le Guludec Bournez Pouly 2017 CMSB]

Stochastic CRN: Simulation of a Turing machine with a **small probability of error**

[Cook, Soloveichik, Winfree, Bruck 2009]

CRN, SBML, Biocham Compared with Kappa

CRN:

- interactions at the molecular species level
- no polymerization reaction
- reachability decidable with discrete semantics (Petri net)
- model-checking decidable with Boolean semantics
- Turing complete with continuous semantics

Kappa:

- interactions at the molecular binding sites level (e.g. protein domains)
- more expressive graph rewriting language (polymerization)
- reachability, model-checking undecidable (approximations by abstractions)
- Turing complete discrete semantics

Biocham (BIOCHemical Abstract Machine)

- CRN structure description language [compatible with SBML]
- CRN behaviour description language [based on temporal logic CTL, FO-LTL(Rlin)]
- CRN analysis and CRN synthesis tools

Abstract CRN Normal Form

Theorem

A real function is computable (respectively in polynomial time)

if and only if it is computable by a system of elementary reactions of the form

_ => z or x => x+z or x+y => x+y+z

plus annihilation reactions x+y => _ with mass action law kinetics

(respectively with trajectories of polynomial length as a function of both the unary precision and the argument values).

Proof Close analysis of the encoding used in the lemmas (positive monomials)

Intermediate CRN: Replace abstract reactions by realistic reactions

- activation (e.g. phosphorylation) instead of formal synthesis
- complexation instead of formal annihilation

Concrete CRN: Search in database of real enzymes (e.g. BRENDA,…)

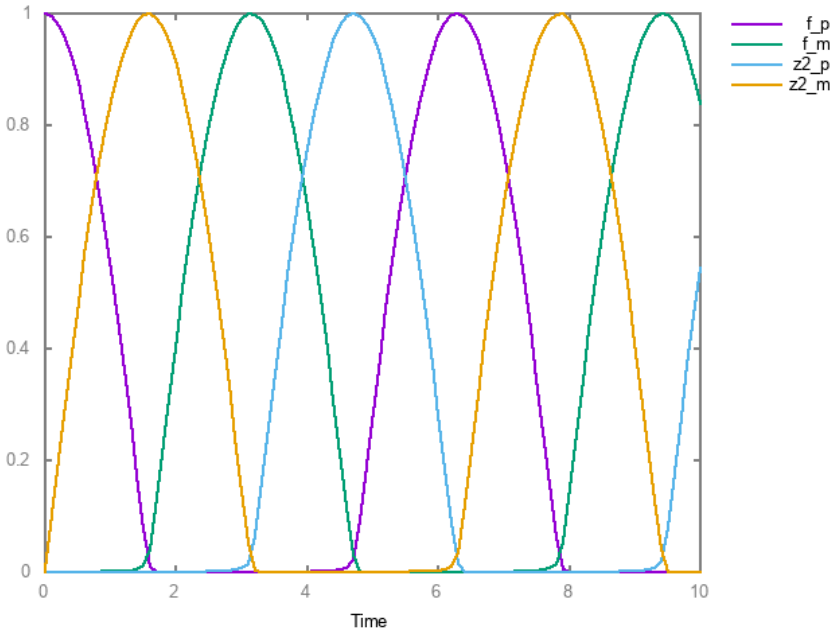
Biocham-4 Compiler of PIVP in CRN

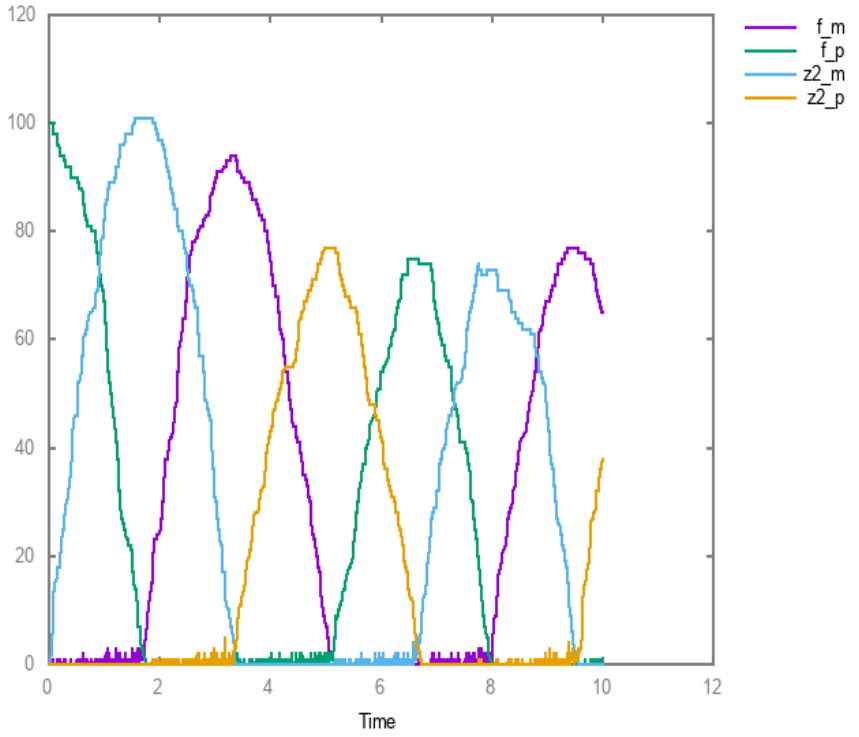
- Definition of mathematical functions and expressions by PIVPs
- No error control (no y2 component)
  - Annihilation reactions with “sufficiently high” kinetic rate constant *fast*
- Dual rail variables (*x_p, x_m)*
  - brute force algorithm (lemma 1)
  - recently added: lazy introduction of negative variables
- PIVP binomialization (rewriting with degree at most 2)
  - recently introduced option (lemma 2)
  - on-going NP-hardness proof and heuristics

Compilation of the Cosine(t) function

```
biocham: compile_from_expression(cos,time,f).
  _ =[z_p]=> f_p.    z_m+z_p => _.
  _ =[z_m]=> f_m.    f_m+f_p => _.
  _ =[f_m]=> z_p.
  _ =[f_p]=> z_m.
present(f_p,1).
```

$$d(f\_p)/dt = z\_p - k*f\_m*f\_p$$
$$d(f\_m)/dt = z\_m - k*f\_m*f\_p$$
$$d(z\_p)/dt = f\_m - k*z\_m*z\_p$$
$$d(z\_m)/dt = f\_p - k*z\_m*z\_p$$
$$f\_p(0)=1$$





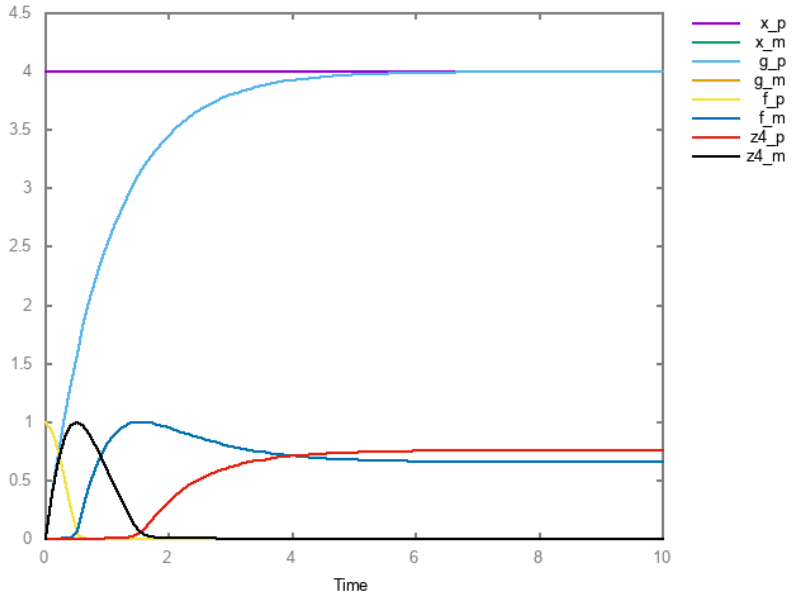
Compilation of the Cosine(x) Function

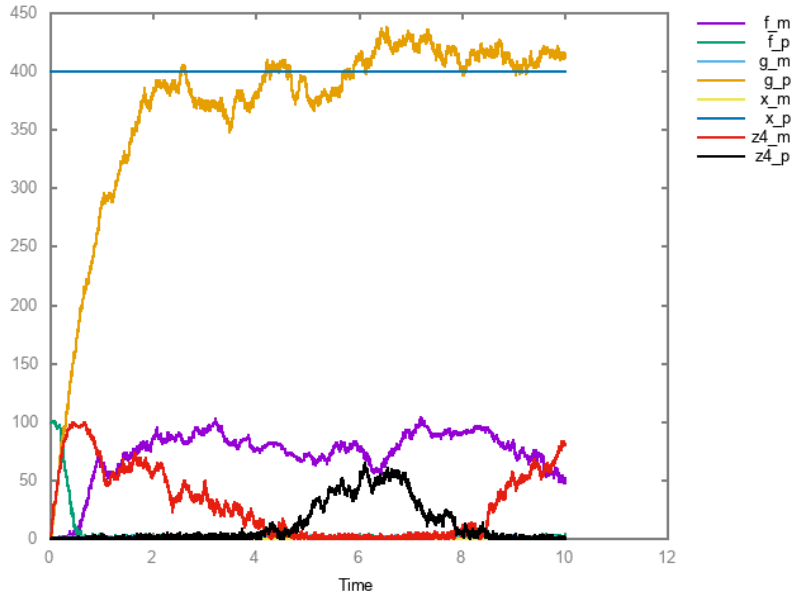
```
biocham: present(x_p, 4).
biocham: compile_from_expression(cos,x,f).
 present(f_p, 1).
_=[g_m]=>g_p.              _=[g_m+f_m]=>z_p.
_=[x_p]=>g_p.              _=[g_p+f_p]=>z_p.
_=[g_p]=>g_m.              _=[x_p+f_m]=>z_p.
_=[x_m]=>g_m.              _=[x_m+f_p]=>z_p.
_=[g_m+z_p]=>f_p.        _=[g_m+f_p]=>z_m.
_=[g_p+z_m]=>f_p.        _=[g_p+f_m]=>z_m.
_=[x_m+z_m]=>f_p.        _=[x_m+f_m]=>z_m.
_=[x_p+z_p]=>f_p.        _=[x_p+f_p]=>z_m.
_=[g_m+z_m]=>f_m.        _=[x_p+f_p]=>z_m.
_=[g_p+z_p]=>f_m.        _=[x_m+z_p]=>f_m.
```

PIVP that generates f(g(t))
with $\lim_{t-\infty} g(t) = x$

$g'(t) = x - g(t)$
$g(t) = x + (x0 - x)e^{-t}$

$(f\ o\ g)' = (f'\ o\ g).g'$





Sigmoid Functions

Hyperbolic tangent

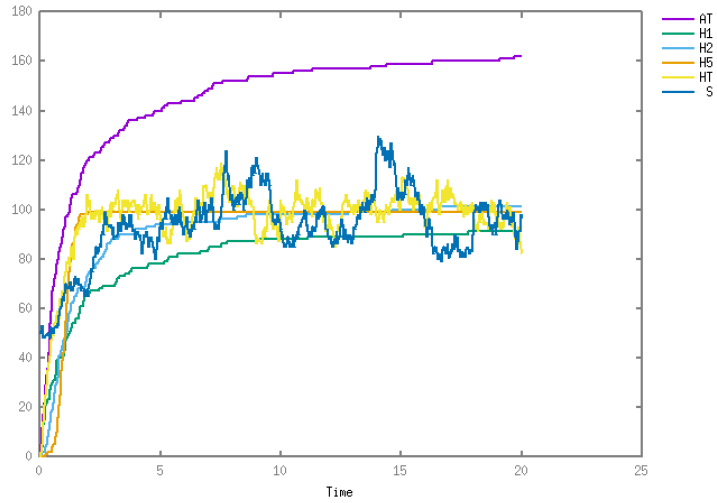
```
d(HT)/dt=1-HT^2
```

```
_=>HT.
HT=[HT]=>_.
```

Logistic

```
d(S)/dt=S-S^2
```

```
_=[S]=>S.
S=[S]=>_.
present(S,0.001).
```



Arc tangent

```
d(T)/dt=1
d(AT)/dt=1/ (1+T^2)
```

```
_=>T.
1/ (1+T^2) for _/T=>AT
```

Hill functions order 1,2,5

```
d(H1)/dt=NH1^2
d(NH1)/dt= -NH1^2
```

```
NH1=[NH1]=>_.
_=[2*NH1]=>H1.
present(NH1,1).
```

```
d(H2)/dt=2*T*NH2^2
d(NH2)/dt= - (2*T*NH2^2)
```

```
MA(2)for NH2=[T+NH2]=>_.
MA(2)for _=[T+2*NH2]=>H2.
present(NH2,1).
```

```
d(H5)/dt=5*T^4*NH5^2
d(NH5)/dt= - (5*T^4*NH5^2)
```

```
MA(5)for NH5=[4*T+NH5]=>_.
MA(5)for _=[4*T+2*NH5]=>H5.
present(NH5,1).
```

Logical Gates

$A,B \in \{0,1\}$

And $C = A \wedge B$

A+B => C	dC/dt = A.B	[C] = min([A],[B])
C(0)=0	dA/dt=dB/dt = -A.B	

Or $C = A \vee B$

A => C	dC/dt = A+B	[C] = [A]+[B]
B => C	dA/dt = -A	
C(0)=0	dB/dt = -B	

Not $C = \neg A$

C+A => _	dC/dt = -C.A	$[C] = \max([C_0]-[A], 0)$
C(0)=1	dA/dt = -C.A	

Sequentiality and Iteration

$Division(A,\ B)$
begin
01 **while** $A \geq B$
02 $A := A - B$
03 $Q := Q + 1$
04 $R := A$
end

1. Asynchronous (precondition) CRN programming

[Huang Jiang Huang Cheng 2012 ICCAD]
[Huang Huang Chiang Jiang Fages 2013 IWBDA]

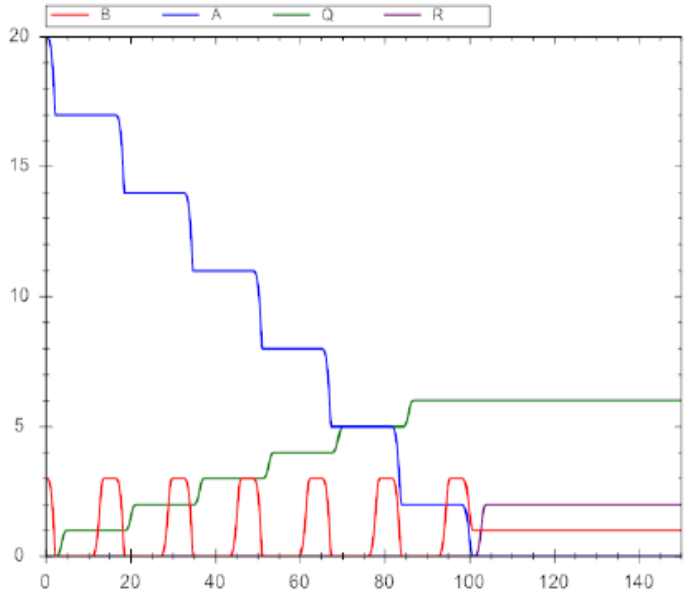
Pb many handshaking species and reactions

2. Synchronous (clock) CRN programming

[Vasic, Soloveichik, Khurshid 2018 CRN++]

Pb many reactions with the clock species

Main Reactions		Preconditions
01	**while** $[A] \geq [B]$	
02	$(A + B \rightarrow D)$	$\neg G_\theta$
03	$C \rightarrow Q + E$	$A_\theta \wedge \neg B_\theta$
04	$D \rightarrow F$	$\neg C_\theta$
05	$E \rightarrow G$	$\neg D_\theta$
06	$F \rightarrow B$	$\neg E_\theta$
07	$G \rightarrow C$	$\neg F_\theta$
08	$D \rightarrow R$	$\neg A_\theta$



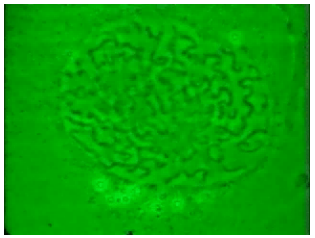
Cell Division Cycle Program

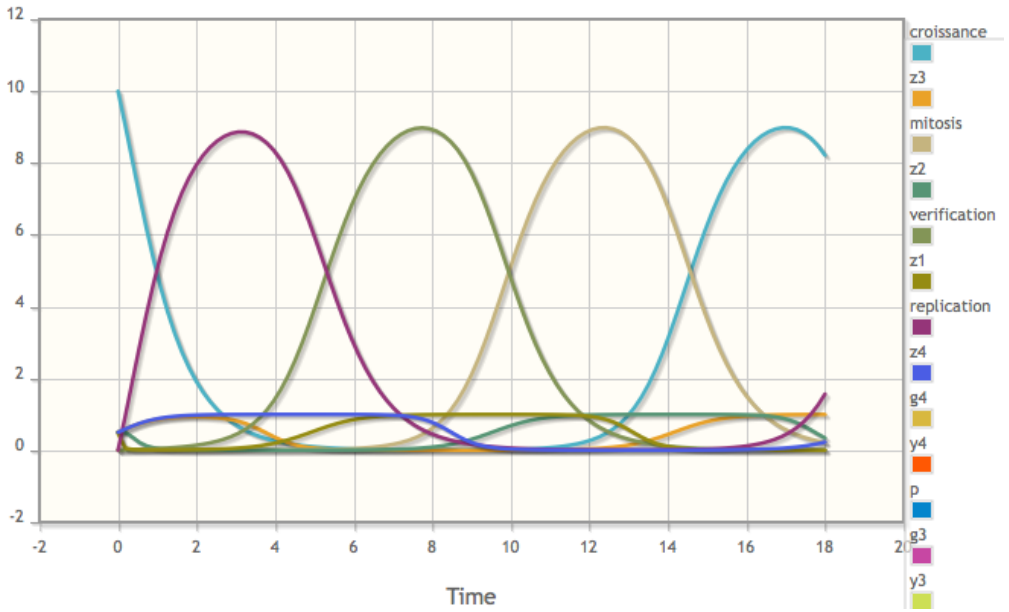
```
while true {growing; replication; verification; mitosis}
```

→ compilation of sequentiality and loops with program control variables

→ 50 reactions

→ 13 variables





Cyclins D, E, A, B as necessary markers for implementing sequentiality

From Abstract to Concrete CRN

Theorem

A real function is computable (respectively in polynomial time)

if and only if it is computable by a system of elementary reactions of the form

_ => z or x => x+z or x+y => x+y+z

plus annihilation reactions x+y => _ with mass action law kinetics

(respectively with trajectories of polynomial length as a function of both the unary precision and the argument values).

Proof Close analysis of the encoding used in the lemmas (positive monomials)

Intermediate CRN: Replace abstract reactions by realistic reactions

- activation (e.g. phosphorylation) instead of formal synthesis
- complexation instead of formal annihilation

Concrete CRN: Search in database of real enzymes (e.g. BRENDA,…)

Computer-Aided Biochemical Programming of Synthetic Micro-reactors as Diagnostic Devices

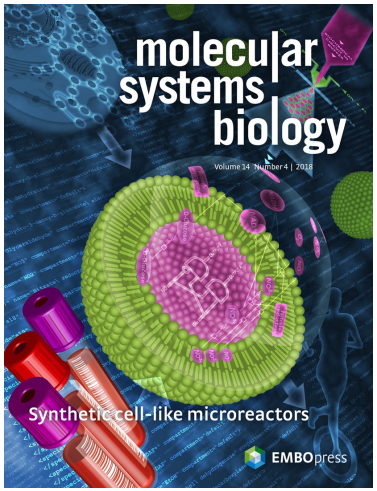
Alexis Courbet [1], Patrick Amar [2], François Fages [3],
Eric Renard [4], Franck Molina [1]

[1] *Sys2diag UMR9005 CNRS/ALCEDIAG, Montpellier*

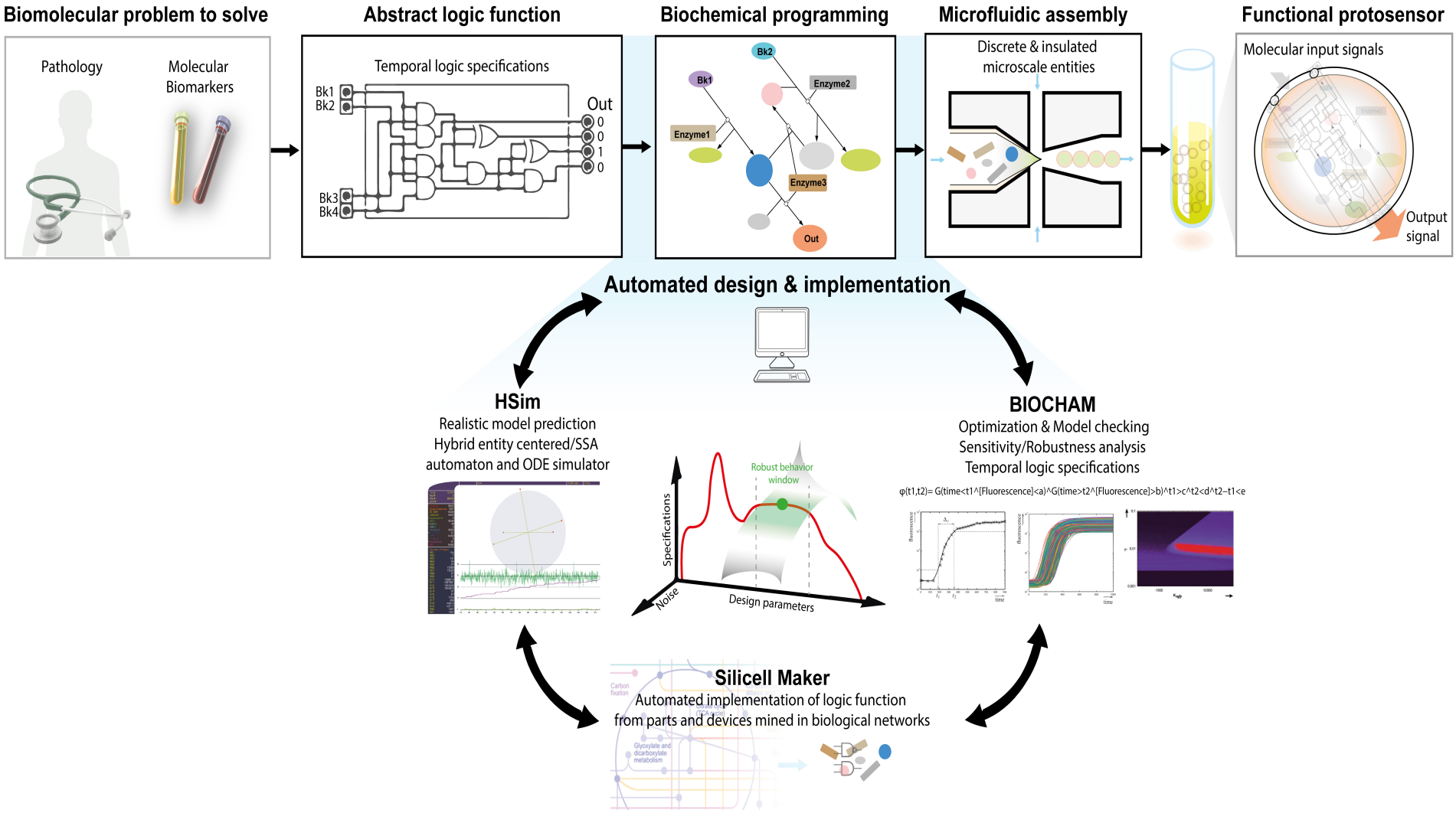
[2] *LRI, Université Paris Sud - UMR CNRS 8623, Orsay*

[3] *http://lifeware.inria.fr, Inria Saclay IdF, Palaiseau*

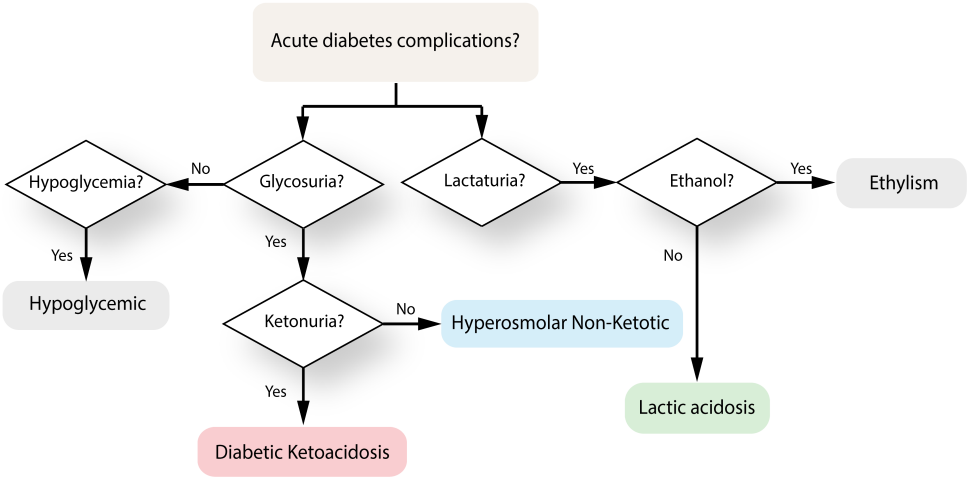
[4] *INSERM 1411, Montpellier University Hospital*

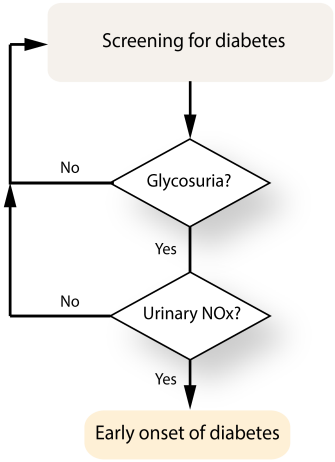


Protosensor CRN Design Workflow

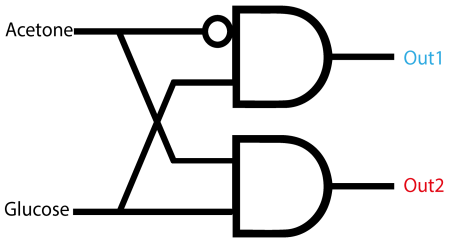


Diabetes Differential Diagnostic Algorithm



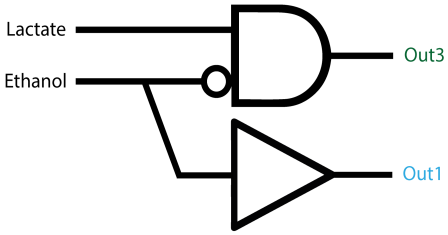


GluONe



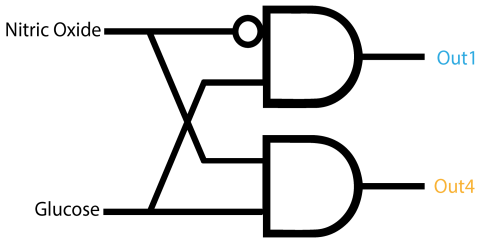
Glucose	Acetone	Out2	Out1
0	0	0	0
1	0	0	1
0	1	0	0
1	1	1	0

LacOH



Lactate	EtOH	Out3	Out1
0	0	0	0
1	0	1	0
0	1	0	1
1	1	0	1

GluNOx



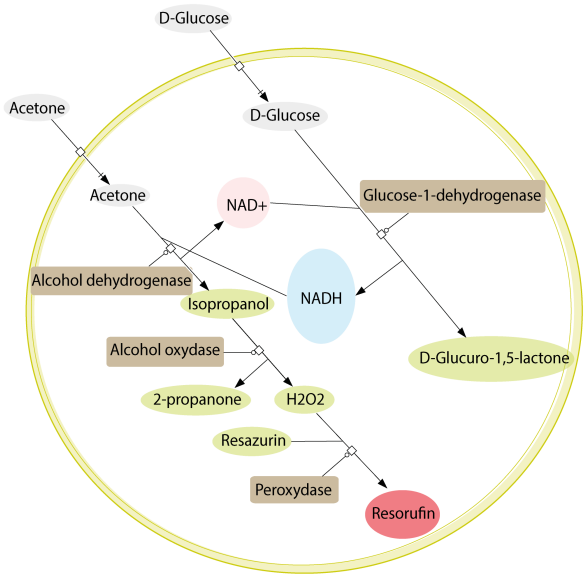
NOx	Glucose	Out4	Out1
0	0	0	0
1	0	0	0
0	1	0	1
1	1	1	0

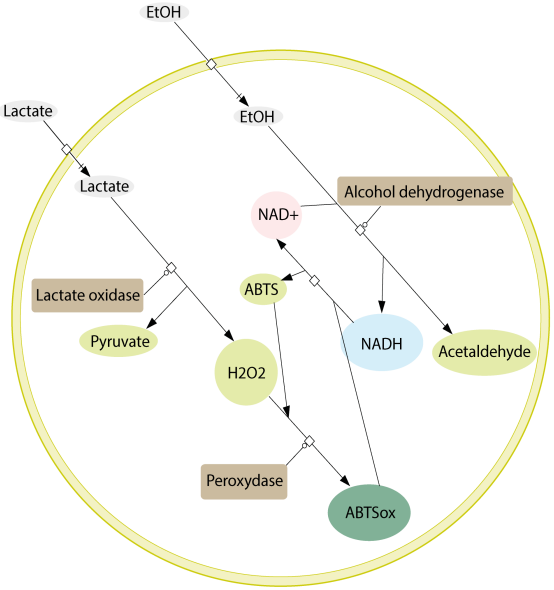
Reactions for Implementing Logical Gates

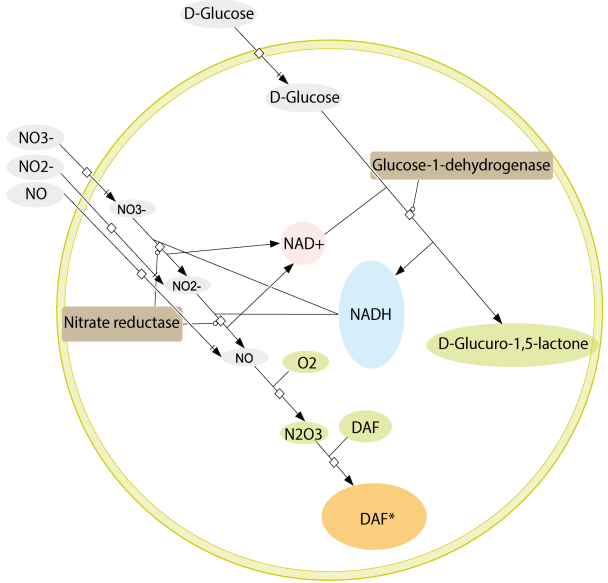
And C = A /\ B A+B => C [C] = min([A],[B])

Or C = A \/ B A => C
B => C [C] = [A]+[B]

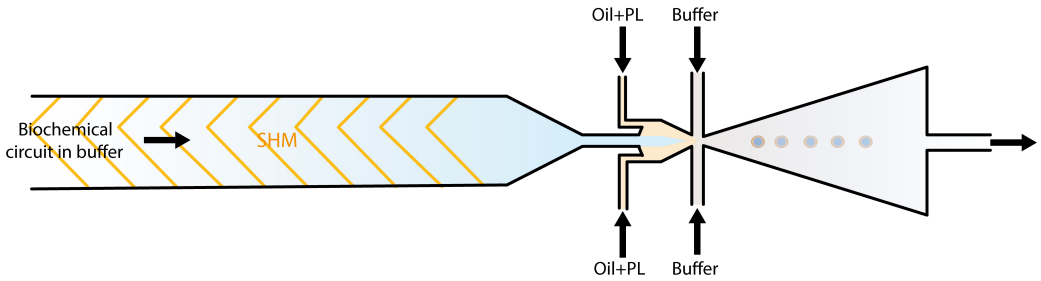
Not C = ¬ A C+A => _ [C] = max([C_0]-[A], 0)

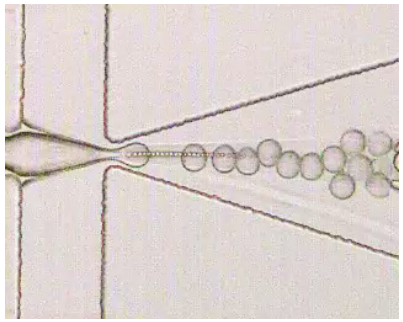


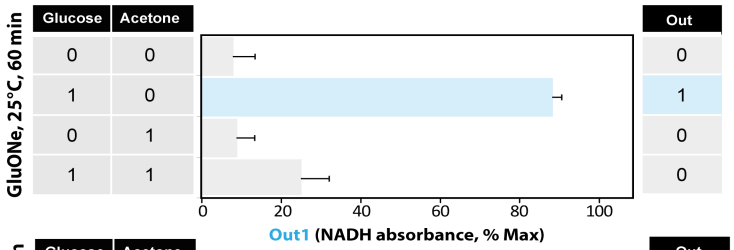


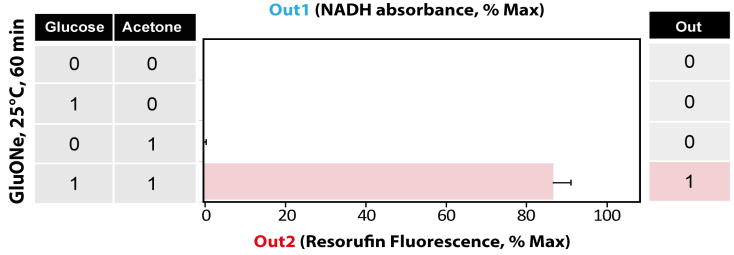


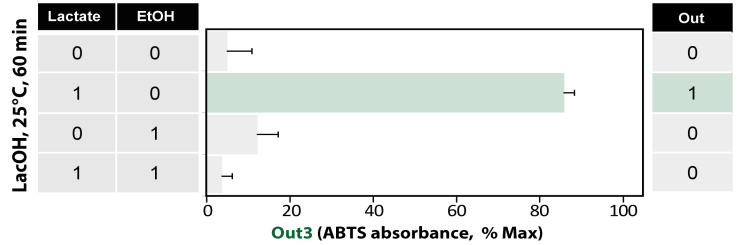
Microfluidic Assembly and Validation in Human Urine

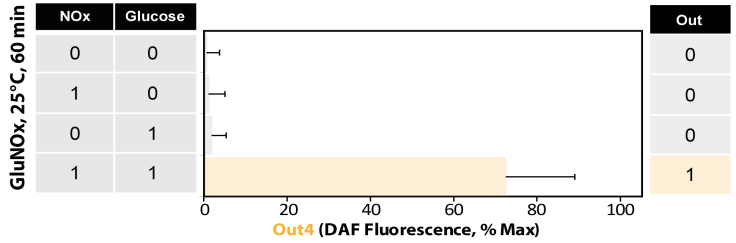


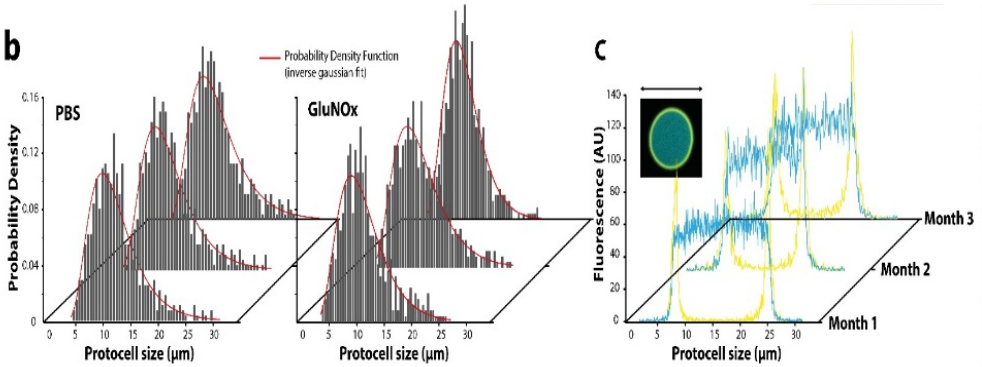












Doctor in the Cell

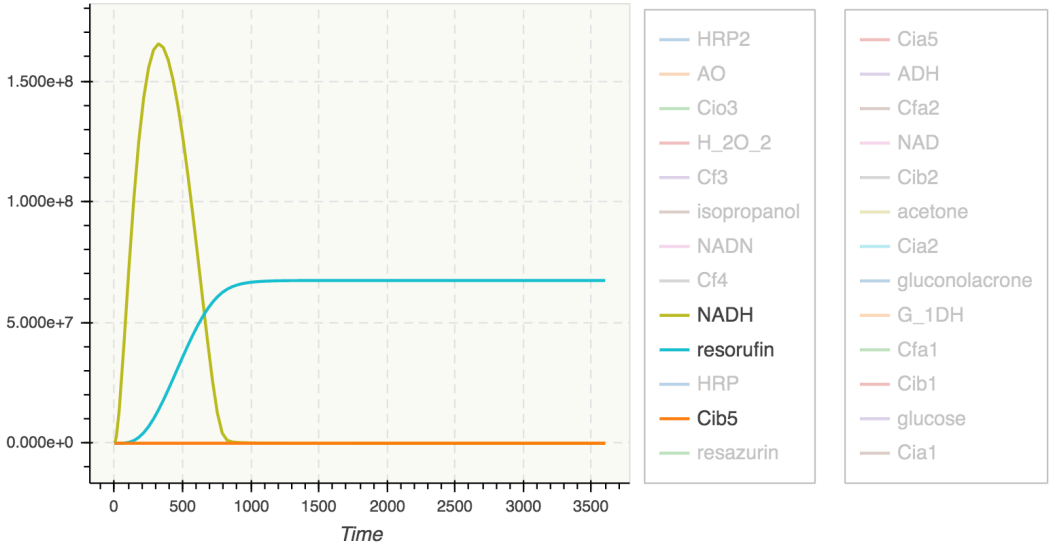
http://lifeware.inria.fr/biocham4/online/

```
In [9]: %slider a b c d e
```

a 9.20788e+07 b 175012

c 2.2326e+06 d 3e+08

e 3e+08




```
In [14]: seed(0). search_parameters(F(Time<T /\ G(resorufin>1e7)),
             [0 <=  a<= 1e9, 0 <= b <= 1e9, 0 <= c <= 1e9],
             [T -> 200]).
```

```
Out[14]:  Time: 2.116 s
Stopping reason: Fitness: function value -9.91e-01 <= stopFitness (1.00e-04)
Best satisfaction degree: 112.773262
[0] parameter(a=560308913.9562368)
[1] parameter(b=165571523.66216615)
[2] parameter(c=296073304.37417626)
```

Conclusion 1/2

- Binary reaction systems over a finite set of molecules (without polymerization) are Turing-complete under the differential semantics
  - PIVP definition of computable function
  - Notion of computational complexity as trajectory length of stabilizing PIVPs

- Analog compiler in CRN [Biocham v4]
  - Input: Function specification by PIVP, mixed digital-analog program
  - Output: system of binary reactions with mass action law kinetics
  - Exact characterization of the result for an ideal fluid implementation
  - Difficult to compare to natural circuits for similar functions

- Real implementation in artificial vesicles [Molina's lab CNRS-Alcediag]

- Alternative design by evolution/learning:
  - Artificial evolution of CRNs [Degrand Hemery F 2019]
  - Nature algorithms for learning [Valliant 2013]

CRN ↔ Function

↻

Mutations

Conclusion 2/2: CRN Design Methods in Biocham

- Quantitative Temporal Logic Workflow

  Input: 1. CRN structure (+ kinetic parameters)

  2. Behavior specification with FO-LTL($\mathbb{R}_{lin}$) formulae

  → Verification with continuous satisfaction degree in [0,1]

  → Parameter sensivity and model robustness wrt parameter perturbations

  → Parameter search by continuous optimization (CMA-ES), robustness optimization

- Polynomial ODE Initial Value Problem PIVP Workflow

  Input: Real valued function specification by PIVP

  → CRN structure with kinetic parameters: exact result (error control)

- Artificial Evolution/Learning Workflow

  Input: time series data (finite traces)

  → CRN structure with kinetic parameters: approximate result

  → genetic alg. + param. opt. [Elisabeth Degrand, Mathieu Hemery] (curve fitting)

  → statistical unsupervised learning of reactions [Julien Martinelli, Jeremy Grignard]