Continuous models of computation: computability, complexity, universality

Amaury Pouly

Joint work with Olivier Bournez and Daniel Graça

21 september 2018
What is a computer?
What is a computer?
What is a computer?

VS

VS
Church Thesis

All reasonable models of computation are equivalent.
Effective Church Thesis

All reasonable models of computation are equivalent for complexity.
Polynomial Differential Equations

General Purpose Analog Computer

Newton mechanics

Reaction networks:
- chemical
- enzymatic

Differential Analyzer

polynomial differential equations:
\[
\begin{cases}
  y(0) = y_0 \\
  y'(t) = p(y(t))
\end{cases}
\]

- Rich class
- Stable (+, ×, ⊙, /, ED)
- No closed-form solution
Example of dynamical system

\[ \ddot{\theta} + \frac{g}{\ell} \sin(\theta) = 0 \]
Example of dynamical system

\[ \ddot{\theta} + \frac{g}{\ell} \sin(\theta) = 0 \]

\[
\begin{cases}
y'_1 = y_2 \\
y'_2 = -\frac{g}{\ell} y_3 \\
y'_3 = y_2 y_4 \\
y'_4 = -y_2 y_3
\end{cases} \quad \Leftrightarrow \quad \begin{cases}
y_1 = \theta \\
y_2 = \dot{\theta} \\
y_3 = \sin(\theta) \\
y_4 = \cos(\theta)
\end{cases}
\]

Historical remark: the word "analog".
Example of dynamical system

\[ \ddot{\theta} + \frac{g}{\ell} \sin(\theta) = 0 \]

Historical remark: the word "analog"

The pendulum and the circuit have the same equation. One can study one using the other by analogy.
Example of dynamical system

\[ \ddot{\theta} + \frac{g}{\ell} \sin(\theta) = 0 \]

Historical remark: the word “analog”

The pendulum and the circuit have the same equation. One can study one using the other by analogy.
Computing with differential equations

Generable functions

\[
\begin{align*}
\left\{ 
\begin{array}{l}
y(0) = y_0 \\
y'(x) = p(y(x))
\end{array}
\right. \\
x \in \mathbb{R}
\end{align*}
\]

\[ f(x) = y_1(x) \]

Shannon’s notion

\[ f(x) = \lim_{t \to \infty} y_1(t) \]

\[ y_1(t) \]
Computing with differential equations

Generable functions

\[
\begin{align*}
y(0) &= y_0 \\
y'(x) &= p(y(x))
\end{align*}
\]

\[x \in \mathbb{R}\]

\[f(x) = y_1(x)\]

Shannon’s notion

\(\sin, \cos, \exp, \log, \ldots\)

Strictly weaker than Turing machines [Shannon, 1941]
Computing with differential equations

Generable functions
\[
\begin{aligned}
\begin{cases}
y(0) &= y_0 \\
y'(x) &= p(y(x))
\end{cases}
\quad x \in \mathbb{R}
\end{aligned}
\]
\[f(x) = y_1(x)\]

Shannon's notion
sin, cos, exp, log, ...
Strictly weaker than Turing machines [Shannon, 1941]

Computable
\[
\begin{aligned}
\begin{cases}
y(0) &= q(x) \\
y'(t) &= p(y(t))
\end{cases}
\quad x \in \mathbb{R}
\end{aligned}
\]
\[
\begin{aligned}
\begin{cases}
y'(t) &= p(y(t))
\end{cases}
\quad t \in \mathbb{R}_+
\end{aligned}
\]
\[f(x) = \lim_{t \to \infty} y_1(t)\]

Modern notion

Computing with differential equations

Generable functions

\[
\begin{cases}
y(0) = y_0 \\
y'(x) = p(y(x))
\end{cases} \quad x \in \mathbb{R}
\]

\[f(x) = y_1(x)\]

Shannon's notion

\[\sin, \cos, \exp, \log, \ldots\]

Strictly weaker than Turing machines [Shannon, 1941]

Computable

\[
\begin{cases}
y(0) = q(x) \\
y'(t) = p(y(t))
\end{cases} \quad x \in \mathbb{R}, \quad t \in \mathbb{R}_+
\]

\[f(x) = \lim_{t \to \infty} y_1(t)\]

Modern notion

\[\sin, \cos, \exp, \log, \Gamma, \zeta, \ldots\]

Turing powerful [Bournez et al., 2007]
**Equivalence with computable analysis**

**Definition (Bournez et al, 2007)**

A function $f$ is **computable by GPAC** if there exists a polynomial $p$ such that for all $x \in [a, b]$,

$$y(0) = (x, 0, \ldots, 0) \quad y'(t) = p(y(t))$$

satisfies $|f(x) - y_1(t)| \leq y_2(t)$ and $y_2(t) \to 0$ as $t \to \infty$.

**Theorem (Bournez et al, 2007)**

$f : [a, b] \to \mathbb{R}$ is computable if and only if $f$ is computable by GPAC.

In Computable Analysis, a standard model over reals built from Turing machines, $y_1(t) \to f(x)$ and $y_2(t) = \text{error bound}$.
Equivalence with computable analysis

**Definition (Bournez et al, 2007)**

$f$ **computable by GPAC** if \( \exists p \) polynomial such that \( \forall x \in [a, b] \)

\[
y(0) = (x, 0, \ldots, 0) \quad y'(t) = p(y(t))
\]
satisfies \( |f(x) - y_1(t)| \leq y_2(t) \) et \( y_2(t) \xrightarrow{t\to\infty} 0. \)

---

**Theorem (Bournez et al, 2007)**

\( f : [a, b] \to \mathbb{R} \) **computable** \( \iff \) \( f \) **computable by GPAC**
Equivalence with computable analysis

Definition (Bournez et al, 2007)

\( f \) computable by GPAC if \( \exists p \) polynomial such that \( \forall x \in [a, b] \)

\[
y(0) = (x, 0, \ldots, 0) \quad y'(t) = p(y(t))
\]

satisfies \( |f(x) - y_1(t)| \leq y_2(t) \) et \( y_2(t) \to 0 \) \( t \to \infty \).

Theorem (Bournez et al, 2007)

\( f : [a, b] \to \mathbb{R} \) computable\(^1 \) \iff \( f \) computable by GPAC

---

1. In Computable Analysis, a standard model over reals built from Turing machines.
Complexity of analog systems

- Turing machines: $T(x) = \text{number of steps to compute on } x$
Complexity of analog systems

- Turing machines: $T(x) = \text{number of steps to compute on } x$
- GPAC:

## Tentative definition

$T(x) = ??$

$y(0) = (x, 0, \ldots, 0)$ \hspace{1cm} $y' = p(y)$

$w(t) = y(\sqrt{e}t)$ \hspace{1cm} $x \to y \to y_1(t) \to f(x)$

Something is wrong... All functions have constant time complexity.
Complexity of analog systems

- Turing machines: $T(x) = \text{number of steps to compute on } x$
- GPAC:

**Tentative definition**

$$T(x, \mu) =$$

$$y(0) = (x, 0, \ldots, 0) \quad y' = p(y)$$

$y_1(t)$ and $f(x)$ are plotted on a diagram.
Complexity of analog systems

- Turing machines: \( T(x) = \) number of steps to compute on \( x \)
- GPAC:

**Tentative definition**

\[
T(x, \mu) = \text{first time } t \text{ so that } |y_1(t) - f(x)| \leq e^{-\mu}
\]

\[
y(0) = (x, 0, \ldots, 0) \quad y' = p(y)
\]
Complexity of analog systems

- Turing machines: \( T(x) \) = number of steps to compute on \( x \)
- GPAC:

Tentative definition

\[
T(x, \mu) = \text{first time } t \text{ so that } |y_1(t) - f(x)| \leq e^{-\mu}
\]

\[
y(0) = (x, 0, \ldots, 0) \quad y' = p(y)
\]

\[
z(t) = y(e^t)
\]

Something is wrong... All functions have constant time complexity.

\[
w(t) = y(e^{e^t})
\]

\[
x_0 \quad y_1(t) \quad f(x) \quad t
\]

\[
x_0 \quad z_1(t) \quad f(x) \quad t
\]
Complexity of analog systems

- Turing machines: \( T(x) \) = number of steps to compute on \( x \)
- GPAC:

Tentative definition

\[
T(x, \mu) = \text{first time } t \text{ so that } |y_1(t) - f(x)| \leq e^{-\mu}
\]

\[
y(0) = (x, 0, \ldots, 0) \quad y' = p(y)
\]

\[
\begin{align*}
y_1(t) & \sim f(x) \\
z(t) & = y(e^t) \\
w(t) & = y(e^{e^t})
\end{align*}
\]
Complexity of analog systems

- Turing machines: \( T(x) \) = number of steps to compute on \( x \)
- GPAC: time contraction problem → open problem

Tentative definition

\[ T(x, \mu) = \text{first time } t \text{ so that } |y_1(t) - f(x)| \leq e^{-\mu} \]

\[ y(0) = (x, 0, \ldots, 0) \quad y' = p(y) \]

Something is wrong...

All functions have constant time complexity.
Time-space correlation of the GPAC

\[ y(0) = q(x) \quad y' = p(y) \]

\[ z(t) = y(e^t) \]

Observation: Time scaling costs "space". Time complexity for the GPAC must involve time and space!
Time-space correlation of the GPAC

\[ y(0) = q(x) \quad y' = p(y) \]

\[ z(t) = y(e^t) \]

extra component: \( w(t) = e^t \)
Time-space correlation of the GPAC

Observation

Time scaling costs “space”.

Time complexity for the GPAC must involve time and space!
Complexity of solving polynomial ODEs

\[ y(0) = x \quad y'(t) = p(y(t)) \]

Theorem

If \( y(t) \) exists, one can compute \( p, q \) such that
\[ \left| p - y(t) \right| \leq 2^{-n} \text{ in time } \text{poly} \left( \text{size of } x \text{ and } p, n, \ell(t) \right) \]

where \( \ell(t) \approx \text{length of the curve (between } x \text{ and } y(t)) \)
Complexity of solving polynomial ODEs

\[ y(0) = x \quad y'(t) = p(y(t)) \]

**Theorem**

If \( y(t) \) exists, one can compute \( p, q \) such that

\[ \left| \frac{p}{q} - y(t) \right| \leq 2^{-n} \text{ in time } \text{poly (size of } x \text{ and } p, n, \ell(t)) \]

where \( \ell(t) \approx \text{length of the curve (between } x \text{ and } y(t)) \)

length of the curve = complexity = ressource
Characterization of real polynomial time

**Definition**: \( f : [a, b] \rightarrow \mathbb{R} \) in \( \text{ANALOG-P}_\mathbb{R} \) \( \iff \exists p \text{ polynomial}, \forall x \in [a, b] \)

\[
y(0) = (x, 0, \ldots, 0) \quad y' = p(y)
\]

![Diagram showing function and polynomial relationship](image-url)
Characterization of real polynomial time

**Definition:** \( f : [a, b] \rightarrow \mathbb{R} \) in \( \text{ANALOG-P}_{\mathbb{R}} \iff \exists p \) polynomial, \( \forall x \in [a, b] \\
\ y(0) = (x, 0, \ldots, 0) \quad y' = p(y) 
\)
satisfies:

1. \( |y_1(t) - f(x)| \leq 2^{-\ell(t)} \)
   «greater length \( \Rightarrow \) greater precision»

2. \( \ell(t) \geq t \)
   «length increases with time»

![Diagram showing the relationship between \( y_1(t) \), \( f(x) \), and \( \ell(t) \).]
Characterization of real polynomial time

**Definition**: \( f : [a, b] \rightarrow \mathbb{R} \) in \( \text{ANALOG-P}_\mathbb{R} \) \( \iff \exists p \) polynomial, \( \forall x \in [a, b] \)

\[
y(0) = (x, 0, \ldots, 0) \quad y' = p(y)
\]

satisfies:

1. \(|y_1(t) - f(x)| \leq 2^{-\ell(t)}\)
   «greater length ⇒ greater precision»

2. \(\ell(t) \geq t\)
   «length increases with time»

**Theorem**

\( f : [a, b] \rightarrow \mathbb{R} \) computable in polynomial time \( \iff f \in \text{ANALOG-P}_\mathbb{R} \).
Characterization of polynomial time

**Definition:** \( \mathcal{L} \in \text{ANALOG-PTIME} \iff \exists p \text{ polynomial}, \forall \text{ word } w \)

\[
y(0) = (\psi(w), |w|, 0, \ldots, 0) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}
\]

\( y_1(t) \) = length of \( y \)

\( \psi(w) \)
Characterization of polynomial time

**Definition**: \( \mathcal{L} \in \text{ANALOG-PTIME} \iff \exists p \text{ polynomial, } \forall \text{ word } w \)

\[
y(0) = (\psi(w), |w|, 0, \ldots, 0) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{\frac{|w|}{2}} w_i 2^{-i}
\]

\[
y_1(t) = \psi(w) \quad \ell(t) = \text{length of } y
\]

accept: \( w \in \mathcal{L} \)

computing

satisfies

1. if \( y_1(t) \geq 1 \) then \( w \in \mathcal{L} \)
**Characterization of polynomial time**

**Definition:** \( \mathcal{L} \in \text{ANALOG-PTIME} \iff \exists p \text{ polynomial, } \forall \text{ word } w \)

\[
y(0) = (\psi(w), |w|, 0, \ldots, 0) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}
\]

- **accept:** \( w \in \mathcal{L} \)
- **computing**
- **reject:** \( w \notin \mathcal{L} \)

satisfies

2. if \( y_1(t) \leq -1 \) then \( w \notin \mathcal{L} \)
Characterization of polynomial time

**Definition:** \( \mathcal{L} \in \text{ANALOG-PTIME} \iff \exists p \text{ polynomial}, \ \forall \text{ word } w \)

\[
y(0) = (\psi(w), |w|, 0, \ldots, 0) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{\text{|w|}} w_i 2^{-i}
\]

accept : \( w \in \mathcal{L} \)

computing

reject : \( w \notin \mathcal{L} \)

satisfies

3. if \( \ell(t) \geq \text{poly}(|w|) \) then \( |y_1(t)| \geq 1 \)
Characterization of polynomial time

**Definition**: \( \mathcal{L} \in \text{ANALOG-PTIME} \iff \exists \, \text{polynomial}, \, \forall \, \text{word } w \)

\[
y(0) = (\psi(w), |w|, 0, \ldots, 0) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}
\]

- **Accept**: \( w \in \mathcal{L} \)
- **Reject**: \( w \notin \mathcal{L} \)
- Computing: \( \ell(t) = \text{length of } y \)
- Forbidden: \( \psi(w) \)

**Theorem**: \( \text{PTIME} = \text{ANALOG-PTIME} \)
Theorem

- $\mathcal{L} \in \text{PTIME} \text{ of and only if } \mathcal{L} \in \text{ANALOG-PTIME}$
- $f : [a, b] \rightarrow \mathbb{R}$ computable in polynomial time $\iff f \in \text{ANALOG-P}_{\mathbb{R}}$

- Analog complexity theory based on length
- Time of Turing machine $\iff$ length of the GPAC
- Purely continuous characterization of PTIME
Theorem

- $\mathcal{L} \in \text{PTIME} \ if \ and \ only \ if \ \mathcal{L} \in \text{ANALOG-PTIME}$
- $f : [a, b] \rightarrow \mathbb{R} \text{ computable in polynomial time} \iff f \in \text{ANALOG-P}_\mathbb{R}$

- Analog complexity theory based on length
- Time of Turing machine $\iff$ length of the GPAC
- Purely continuous characterization of PTIME
- Only rational coefficients needed
In the remaining time...

Two applications of the “technology” we have developed:

- Universal differential equation
- Chemical Reaction Networks
Universal differential equations

Generable functions

subclass of analytic functions

Computable functions

any computable function
Universal differential equations

Generable functions

subclass of analytic functions

Computable functions

any computable function
Universal differential algebraic equation (DAE)

Theorem (Rubel, 1981)

For any continuous functions \( f \) and \( \varepsilon \), there exists \( y : \mathbb{R} \rightarrow \mathbb{R} \) solution to

\[
3y^4 y'' y''''^2 - 4y^4 y''''^2 y'''' + 6y^3 y''^2 y''' y'''' + 24y^2 y''^4 y''''
- 12y^3 y'' y'''^3 - 29y^2 y'''^3 y''''^2 + 12y''''^7 = 0
\]

such that \( \forall t \in \mathbb{R}, \)

\[
|y(t) - f(t)| \leq \varepsilon(t).
\]

Problem: this is «weak» result.

16 / 21
Theorem (Rubel, 1981)

There exists a fixed polynomial $p$ and $k \in \mathbb{N}$ such that for any continuous functions $f$ and $\varepsilon$, there exists a solution $y : \mathbb{R} \rightarrow \mathbb{R}$ to

$$p(y, y', \ldots, y^{(k)}) = 0$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leq \varepsilon(t).$$
Theorem (Rubel, 1981)

There exists a \textbf{fixed} polynomial \( p \) and \( k \in \mathbb{N} \) such that for any continuous functions \( f \) and \( \varepsilon \), there exists a solution \( y : \mathbb{R} \to \mathbb{R} \) to

\[
p(y, y', \ldots, y^{(k)}) = 0
\]

such that \( \forall t \in \mathbb{R} \),

\[
|y(t) - f(t)| \leq \varepsilon(t).
\]

Problem: this is «weak» result.
The problem with Rubel’s DAE

The solution $y$ is not unique, even with added initial conditions:

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

In fact, this is fundamental for Rubel’s proof to work!
The problem with Rubel’s DAE

The solution $y$ is not unique, **even with added initial conditions** :

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

In fact, this is fundamental for Rubel’s proof to work!

- Rubel’s statement: this DAE is universal
- More realistic interpretation: this DAE allows almost anything

Open Problem (Rubel, 1981)

Is there a universal ODE $y' = p(y)$?

**Note**: explicit polynomial ODE $\Rightarrow$ unique solution
Theorem

There exists a fixed (vector of) polynomial $p$ such that for any continuous functions $f$ and $\varepsilon$, there exists $\alpha \in \mathbb{R}^d$ such that

$$y(0) = \alpha, \quad y'(t) = p(y(t))$$

has a unique solution $y : \mathbb{R} \to \mathbb{R}^d$ and $\forall t \in \mathbb{R},$

$$|y_1(t) - f(t)| \leq \varepsilon(t).$$
There exists a fixed (vector of) polynomial \( p \) such that for any continuous functions \( f \) and \( \varepsilon \), there exists \( \alpha \in \mathbb{R}^d \) such that

\[
y(0) = \alpha, \quad y'(t) = p(y(t))
\]

has a unique solution \( y : \mathbb{R} \to \mathbb{R}^d \) and \( \forall t \in \mathbb{R} \),

\[
|y_1(t) - f(t)| \leq \varepsilon(t).
\]
Universal initial value problem (IVP)

Notes:
- system of ODEs,
- $y$ is analytic,
- we need $d \approx 300$.

Theorem

There exists a fixed (vector of) polynomial $p$ such that for any continuous functions $f$ and $\varepsilon$, there exists $\alpha \in \mathbb{R}^d$ such that

$$y(0) = \alpha, \quad y'(t) = p(y(t))$$

has a unique solution $y : \mathbb{R} \to \mathbb{R}^d$ and $\forall t \in \mathbb{R}$,

$$|y_1(t) - f(t)| \leq \varepsilon(t).$$

Remark: $\alpha$ is usually transcendental, but computable from $f$ and $\varepsilon$. 

\[18/21\]
Chemical Reaction Networks

**Definition**: a reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ ($a_i, b_i \in \mathbb{N}, f = \text{rate}$)

**Example**:

\[
2H_2 + O \rightarrow 2H_2O
\]
\[
C + O_2 \rightarrow CO_2
\]
**Definition**: a reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \quad (a_i, b_i \in \mathbb{N}, f = \text{rate})$

**Example**:

\[
\begin{align*}
2H_2 + O & \rightarrow 2H_2O \\
C + O_2 & \rightarrow CO_2
\end{align*}
\]

**Assumption**: law of mass action

\[
\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \sim f(y) = k \prod_i y_i^{a_i}
\]
Chemical Reaction Networks

**Definition**: a reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ \quad ($a_i, b_i \in \mathbb{N}, f = \text{rate}$)

**Example**:

$$2\text{H}_2 + \text{O} \rightarrow 2\text{H}_2\text{O}$$
$$\text{C} + \text{O}_2 \rightarrow \text{CO}_2$$

**Assumption**: law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \quad \sim \quad f(y) = k \prod_i y_i^{a_i}$$

**Semantics**:

- discrete
- differential
- stochastic
**Chemical Reaction Networks**

**Definition**: A reaction system is a finite set of
- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ $(a_i, b_i \in \mathbb{N}, f = \text{rate})$

**Example**:

$$2\text{H}_2 + \text{O} \rightarrow 2\text{H}_2\text{O}$$

$$\text{C} + \text{O}_2 \rightarrow \text{CO}_2$$

**Assumption**: law of mass action

$$\sum a_i y_i \xrightarrow{k} \sum b_i y_i \sim f(y) = k \prod_i y_i^{a_i}$$

**Semantics**:
- discrete
- differential →
- stochastic

$$y'_i = \sum_{\text{reaction } R} (b_i^R - a_i^R) f^R(y)$$
Chemical Reaction Networks

**Definition**: a reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ ($a_i, b_i \in \mathbb{N}$, $f = \text{rate}$)

**Example**:

$$2H_2 + O \rightarrow 2H_2O$$

$$CO + O_2 \rightarrow CO_2$$

**Assumption**: law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \sim f(y) = k \prod_i y_i^{a_i}$$

**Semantics**:

- discrete
- differential $\rightarrow$
- stochastic

$$y'_i = \sum_{\text{reaction } R} (b_i^R - a_i^R) k^R \prod_j y_j^{a_j}$$
Chemical Reaction Networks (CRNs)

- CRNs with differential semantics and mass action law = polynomial ODEs
- polynomial ODEs are Turing complete

Two “slight” problems:
- Concentrations cannot be negative (\(y_i < 0\))
- Arbitrary reactions are not realistic

Definition: a reaction is elementary if it has at most two reactants
⇒ can be implemented with DNA, RNA or proteins

Elementary reactions correspond to quadratic ODEs:
\[ ay + bz^k \rightarrow \cdots \]

\[ f(y, z) = ky^a z^b \]

Theorem (Folklore): Every polynomial ODE can be rewritten as a quadratic ODE.
Chemical Reaction Networks (CRNs)

- CRNs with differential semantics and mass action law = polynomial ODEs
- polynomial ODEs are Turing complete

CRNs are Turing complete?
CRNs are Turing complete? Two “slight” problems:

- Concentrations cannot be negative \( y_i < 0 \)
- Arbitrary reactions are not realistic
Chemical Reaction Networks (CRNs)

CRNs are Turing complete? Two “slight” problems:

- concentrations cannot be negative ($y_i < 0$)
- arbitrary reactions are not realistic

▶ easy to solve
▶ what is realistic?
Chemical Reaction Networks (CRNs)

CRNs are Turing complete? Two “slight” problems:

▶ concentrations cannot be negative \( y_i < 0 \)
▶ arbitrary reactions are not realistic

Definition: a reaction is **elementary** if it has at most two reactants

⇒ can be implemented with DNA, RNA or proteins
Chemical Reaction Networks (CRNs)

CRNs are Turing complete? Two “slight” problems:

▶ concentrations cannot be negative \((y_i < 0)\)  
▶ arbitrary reactions are not realistic

**Definition**: a reaction is **elementary** if it has at most two reactants  
⇒ can be implemented with DNA, RNA or proteins

Elementary reactions correspond to **quadratic** ODEs:

\[
ay + bz \xrightarrow{k} \cdots \quad \leadsto \quad f(y, z) = ky^az^b
\]
CRNs are Turing complete? Two “slight” problems:

▶ concentrations cannot be negative \( y_i < 0 \)
▶ arbitrary reactions are not realistic

Definition: a reaction is elementary if it has at most two reactants

⇒ can be implemented with DNA, RNA or proteins

Elementary reactions correspond to quadratic ODEs:

\[
ay + bz \xrightarrow{k} \cdots \quad \sim \quad f(y, z) = ky^a z^b
\]

Theorem (Folklore)

Every polynomial ODE can be rewritten as a quadratic ODE.
Chemical Reaction Networks (CRNs)

**Definition:** a reaction is **elementary** if it has at most two reactants

\[ \Rightarrow \text{can be implemented with DNA, RNA or proteins} \]

Elementary reactions correspond to **quadratic** ODEs:

\[
ay + bz \xrightarrow{k} \ldots \quad \sim \quad f(y, z) = ky^a z^b
\]

**Theorem (CMSB, joint work with François Fages, Guillaume Le Guludec)**

*Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.*

**Notes:**

- proof preserves polynomial length
- in fact the following elementary reactions suffice:

\[
\emptyset \xrightarrow{k} x \quad x \xrightarrow{k} x + z \quad x + y \xrightarrow{k} x + y + z \quad x + y \xrightarrow{k} \emptyset
\]
Future work

Reaction networks:
- chemical
- enzymatic

\[ y' = p(y) \]

\[ y' = p(y) + e(t) \]

- Finer time complexity (linear)
- Nondeterminism
- Robustness
- « Space» complexity
- Other models
- Stochastic
Rubel’s proof in one slide

➤ Take \( f(t) = e^{\frac{-1}{1-t^2}} \) for \(-1 < t < 1\) and \( f(t) = 0 \) otherwise.

It satisfies \((1 - t^2)^2 f''(t) + 2t f'(t) = 0\).
Rubel’s proof in one slide

- Take \( f(t) = e^{\frac{-1}{1-t^2}} \) for \(-1 < t < 1\) and \( f(t) = 0 \) otherwise. It satisfies \((1 - t^2)^2 f''(t) + 2tf'(t) = 0\).

- For any \( a, b, c \in \mathbb{R} \), \( y(t) = cf(at + b) \) satisfies

\[
3y'^4 y'' y'''^2 - 4y'^4 y''^2 y'''' + 6y'^3 y''^2 y''' y'''' + 24y'^2 y''^4 y'''' - 12y'^3 y'' y''''^2 - 29y'^2 y'''^2 y''' + 12y''''^7 = 0
\]
Rubel’s proof in one slide

- Take \( f(t) = e^{1-t^2} \) for \(-1 < t < 1\) and \( f(t) = 0 \) otherwise.
  
  It satisfies \((1 - t^2)^2 f''(t) + 2tf'(t) = 0\).

- For any \( a, b, c \in \mathbb{R} \), \( y(t) = cf(at + b) \) satisfies

  \[
  3y'^4y''y'''^2 - 4y'^4y''^2y'''+6y'^3y''^2y'''' + 24y'^2y''^4y'''' - 12y'^3y''y''''^2 - 29y'^2y''^3y'''^2 + 12y''^7 = 0
  \]

- Can glue together arbitrary many such pieces.
Rubel’s proof in one slide

- Take \( f(t) = e^{\frac{-1}{1-t^2}} \) for \(-1 < t < 1\) and \( f(t) = 0 \) otherwise.
  
  It satisfies \((1 - t^2)^2 f''(t) + 2tf'(t) = 0\).

- For any \( a, b, c \in \mathbb{R} \), \( y(t) = cf(at + b) \) satisfies

\[
3y''^4 y'''^2 - 4y''^4 y''^2 y''' + 6y'''^3 y''^2 y''' + 24y''^2 y''y''' - 12y'''^3 y''^3 - 29y''y'''^3 + 12y''^2 y'' + 12y'''^7 = 0
\]

- Can glue together arbitrary many such pieces
- Can arrange so that \( \int f \) is solution: \textbf{piecewise pseudo-linear}
Rubel’s proof in one slide

- Take \( f(t) = e^{\frac{-1}{1-t^2}} \) for \(-1 < t < 1\) and \( f(t) = 0 \) otherwise.
  
  It satisfies \((1 - t^2)^2 f''(t) + 2tf'(t) = 0.\)

- For any \( a, b, c \in \mathbb{R}, y(t) = cf(at + b) \) satisfies

\[
3y^4y''y''''^2 - 4y^4y''y''''^2 + 6y^3y''y''''y''' + 24y^2y''y''''^4 - 12y^3y''y''''^3 - 29y^2y''y''''^2 + 12y''^7 = 0
\]

- Can glue together arbitrary many such pieces
- Can arrange so that \( \int f \) is solution: piecewise pseudo-linear

Conclusion: Rubel’s equation allows any piecewise pseudo-linear functions, and those are dense in \( C^0 \)
Theorem

There exists a fixed polynomial \( p \) and \( k \in \mathbb{N} \) such that for any continuous functions \( f \) and \( \varepsilon \), there exists \( \alpha_0, \ldots, \alpha_k \in \mathbb{R} \) such that

\[
p(y, y', \ldots, y^{(k)}) = 0,
\]

\[
y(0) = \alpha_0, \ y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k
\]

has a unique analytic solution and this solution satisfies such that

\[
|y(t) - f(t)| \leq \varepsilon(t).
\]