1. Introduction
   - GPAC
   - Computable Analysis
   - Analog Church Thesis
   - Complexity

2. Toward a Complexity Theory for the GPAC
   - What is the problem
   - Computational Complexity (Real Number)
   - Classical Computational Complexity

3. Conclusion
The case of discrete computations

Many models:
- Recursive functions
- Turing machines
- $\lambda$-calculus
- circuits
- ...
The case of discrete computations

Many models:
- Recursive functions
- Turing machines
- $\lambda$-calculus
- circuits
- ...

And

Church Thesis
All reasonable discrete models of computation are equivalent.
The case of analog computations

Several models:

- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...
The case of analog computations

Several models:
- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...

Questions:
- Church Thesis for analog computers?
The case of analog computations

Several models:
- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...

Questions:
- Church Thesis for analog computers? \( \Rightarrow \text{No (GPAC} \neq \text{BSS)} \)
The case of analog computations

Several models:
- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...

Questions:
- Church Thesis for analog computers? ⇒ No (GPAC ≠ BSS)
- Comparison with digital models of computation?
Introduction

The case of analog computations

Several models:
- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...

Questions:
- Church Thesis for analog computers? \(\Rightarrow\) No (GPAC \(\neq\) BSS)
- Comparison with digital models of computation? \(\Rightarrow\) How?
The case of analog computations

Several models:

- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...

Questions:

- Church Thesis for analog computers? \( \Rightarrow \) No (GPAC \( \neq \) BSS)
- Comparison with digital models of computation? \( \Rightarrow \) How?
- What is a “reasonable” model?
The case of analog computations

Several models:
- BSS model (Blum Shub Smale)
- Computable Analysis
- GPAC (General Purpose Analog Computer)
- ...

Questions:
- Church Thesis for analog computers? ⇒ No (GPAC ≠ BSS)
- Comparison with digital models of computation? ⇒ How?
- What is a “reasonable” model? ⇒ Unclear
General Purpose Analog Computer

by Claude Shannon (1941)
General Purpose Analog Computer

- by Claude Shannon (1941)
- idealization of an analog computer: Differential Analyzer
General Purpose Analog Computer

- by Claude Shanon (1941)
- idealization of an analog computer: Differential Analyzer
- circuit built from:

  - A constant unit: $k$
  - An adder unit: $u + v$
  - An multiplier unit: $uv$
  - An integrator unit: $\int u \, dv$
GPAC: beyond the circuit approach

**Theorem**

$y$ is generated by a GPAC iff it is a component of the solution $y = (y_1, \ldots, y_d)$ of the Polynomial Initial Value Problem (PIVP):

\[
\begin{cases}
  y' = p(y) \\
y(t_0) = y_0
\end{cases}
\]

where $p$ is a vector of polynomials.
Theorem

$y$ is generated by a GPAC iff it is a component of the solution $y = (y_1, \ldots, y_d)$ of the Polynomial Initial Value Problem (PIVP):

$$
\begin{aligned}
    y' &= p(y) \\
    y(t_0) &= y_0
\end{aligned}
$$

where $p$ is a vector of polynomials.

Remark

Other point of view: continuous dynamical system
Example (One variable, linear system)

\[ \int t \to e^t \]

\[ \begin{align*}
y' & = y \\
y(0) & = 1
\end{align*} \]
GPAC: examples

Example (One variable, linear system)

\[
\begin{aligned}
    y' &= y \\
    y(0) &= 1
\end{aligned}
\]

Example (One variable, nonlinear system)

\[
\begin{aligned}
    y' &= -2ty^2 \\
    y(0) &= 1
\end{aligned}
\]
Example (One variable, linear system)

\[ \begin{align*}
  y' &= y \\
  y(0) &= 1
\end{align*} \]

Example (Two variable, nonlinear system)

\[ \begin{align*}
  y' &= -2ty^2 \\
  y(0) &= 1 \\
  t' &= 1 \\
  t(0) &= 0
\end{align*} \]
Example (Two variables, linear system)

\[
\begin{align*}
    t & \rightarrow -1 \rightarrow t \\
    t & \rightarrow \times \rightarrow \int \rightarrow \int \\
    \sin(t) & \rightarrow \\
\end{align*}
\]

\[
\begin{align*}
    y' &= z \\
    z' &= -y \\
    y(0) &= 0 \\
    z(0) &= 1
\end{align*}
\]
GPAC: examples

Example (Two variables, linear system)

\[
\begin{align*}
-t & \quad \int & \quad \int \quad \sin(t) \\
\end{align*}
\]

\[
\begin{align*}
y' &= z \\
z' &= -y \\
y(0) &= 0 \\
z(0) &= 1
\end{align*}
\]

Example (Not so nice example)

\[
\begin{align*}
t & \quad \int & \quad \int & \quad \cdots & \quad \int \\
& & & & y_n(t)
\end{align*}
\]

\[
\begin{align*}
y_1' &= y_1 \\
y_2' &= y_2y_1' \\
& \vdots \\
y_n' &= y_ny_{n-1}'
\end{align*}
\]

\(-n\) integrators
GPAC: examples

Example (Two variables, linear system)

\[ \begin{align*}
  y' &= z \\
  z' &= -y \\
  y(0) &= 0 \\
  z(0) &= 1
\end{align*} \]

\[ t \quad -1 \quad \times \quad \int \quad \int \quad \sin(t) \]

Example (Not so nice example)

\[ y_1(t) = e^t \]
\[ y_2(t) = e^{e^t} \]
\[ \ldots \]
\[ y_n(t) = e^{e^{e^{\ldots^{e^t}}}} \]

\[ t \quad \int \quad \int \quad \int \quad \int \quad \int \quad \int \quad \int \quad y_n(t) \]

\[ n \text{ integrators} \]
Motivation

1. Study the computational power of such systems:
Motivation

1. Study the computational power of such systems:
   - (asymptotical) (properties of) solutions
Motivation

1. Study the computational power of such systems:
   - (asymptotical) (properties of) solutions
   - reachability properties
Motivation

1. Study the computational power of such systems:
   - (asymptotical) (properties of) solutions
   - reachability properties
   - attractors
Motivation

1. Study the computational power of such systems:
   - (asymptotical) (properties of) solutions
   - reachability properties
   - attractors

2. Use these systems as a model of computation
Motivation

1. Study the computational power of such systems:
   - (asymptotical) (properties of) solutions
   - reachability properties
   - attractors

2. Use these systems as a model of computation
   - on words
Motivation

1. Study the computational power of such systems:
   - (asymptotical) (properties of) solutions
   - reachability properties
   - attractors

2. Use these systems as a model of computation
   - on words
   - on real numbers
Computable real

Definition (Computable Real)

A real $r \in \mathbb{R}$ is computable if one can compute an arbitrary close approximation for a given precision:

Given $p \in \mathbb{N}$, compute $r_p$ such that $|r - r_p| \leq 2^{-p}$.

Example

Rational numbers, $\pi$, $e$, ... 

Example (Non-computable real)

$r = \sum_{n=0}^{\infty} d_n 2^{-n}$

where $d_n = 1$ iff the $n$th Turing Machine halts on input $n$.
Computable real

Definition (Computable Real)
A real $r \in \mathbb{R}$ is computable is one can compute an arbitrary close approximation for a given precision:
Computable real

Definition (Computable Real)
A real $r \in \mathbb{R}$ is computable is one can compute an arbitrary close approximation for a given precision:

Given $p \in \mathbb{N}$, compute $r_p$ s.t. $|r - r_p| \leq 2^{-p}$
Computable real

**Definition (Computable Real)**

A real \( r \in \mathbb{R} \) is computable is one can compute an arbitrary close approximation for a given precision:

\[
\text{Given } p \in \mathbb{N}, \text{ compute } r_p \text{ s.t. } |r - r_p| \leq 2^{-p}
\]

**Example**

Rational numbers, \( \pi \), e, \ldots
Computable real

**Definition (Computable Real)**
A real $r \in \mathbb{R}$ is computable is one can compute an arbitrary close approximation for a given precision:

Given $p \in \mathbb{N}$, compute $r_p$ s.t. $|r - r_p| \leq 2^{-p}$

**Example**
Rational numbers, $\pi$, $e$, ...

**Example (Non-computable real)**

$$r = \sum_{n=0}^{\infty} d_n 2^{-n}$$

where

$$d_n = 1 \iff \text{the } n^{th} \text{ Turing Machine halts on input } n$$
Computable function

**Definition (Computable Function)**

A function $f : \mathbb{R} \to \mathbb{R}$ is computable if there exist a Turing Machine $M$ s.t. for any $x \in \mathbb{R}$ and oracle $O$ computing $x$, $M^O$ computes $f(x)$. 
Computable function

**Definition (Computable Function)**
A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is computable if there exist a Turing Machine $M$ s.t. for any $x \in \mathbb{R}$ and oracle $O$ computing $x$, $M^O$ computes $f(x)$.

**Definition (Equivalent)**
A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is computable if $f$ is continuous and for a any rational $r$ one can compute $f(r)$.
Computable function

Definition (Computable Function)
A function \( f : \mathbb{R} \rightarrow \mathbb{R} \) is computable if there exist a Turing Machine \( M \) s.t. for any \( x \in \mathbb{R} \) and oracle \( O \) computing \( x \), \( M^O \) computes \( f(x) \).

Definition (Equivalent)
A function \( f : \mathbb{R} \rightarrow \mathbb{R} \) is computable if \( f \) is continuous and for any rational \( r \) one can compute \( f(r) \).

Example
Polynomials, trigonometric functions, \( e^x \), \( \sqrt{x} \), \ldots
Computable function

**Definition (Computable Function)**

A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is computable if there exist a Turing Machine $M$ s.t. for any $x \in \mathbb{R}$ and oracle $O$ computing $x$, $M^O$ computes $f(x)$.

**Definition (Equivalent)**

A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is computable if $f$ is continuous and for any rational $r$ one can compute $f(r)$.

**Example**

Polynomials, trigonometric functions, $e^x$, $\sqrt{x}$, ...  

**Example (Counter-Example)**

$$f(x) = \lceil x \rceil$$
Computable Analysis = GPAC ?
Computable Analysis = GPAC?

Seems not:
Computable Analysis $\neq$ GPAC?

Seems not:

- Solutions of a GPAC are analytic
Seems not:

- Solutions of a GPAC are analytic
- $x \rightarrow |x|$ is computable but not analytic

**Theorem (]**

Computable Analysis $\neq$ General Purpose Analog Computer
Computable Analysis = GPAC?

Seems not:
- Solutions of a GPAC are analytic
- $x \rightarrow |x|$ is computable but not analytic

Theorem ( ):

Computable Analysis $\neq$ General Purpose Analog Computer

Can we fix this?
GPAC: back to the basics

Definition

$y$ is **generated** by a GPAC iff it is a component of the solution $y = (y_1, \ldots, y_d)$ of the ordinary differential equation (ODE):

$$
\begin{align*}
    y' &= p(y) \\
    y(t_0) &= y_0
\end{align*}
$$

where $p$ is a vector of polynomials.
GPAC: back to the basics

Definition

*y* is **generated** by a GPAC iff it is a component of the solution *y* = \((y_1, \ldots, y_d)\) of the ordinary differential equation (ODE):

\[
\begin{align*}
  y' &= p(y) \\
y(t_0) &= y_0
\end{align*}
\]

where *p* is a vector of polynomials.

Definition

*f* is **computable** by a GPAC iff for all *x* ∈ \(\mathbb{R}\) the solution *y* = \((y_1, \ldots, y_d)\) of the ordinary differential equation (ODE):

\[
\begin{align*}
  y' &= p(y) \\
y(t_0) &= q(x)
\end{align*}
\]

where *p, q* is a vector of polynomials satisfies for all *f(x) = \lim_{t \to \infty} y_1(t)*.
GPAC: back to the basics

Definition

\( f \) is **computable** by a GPAC iff for all \( x \in \mathbb{R} \) the solution \( y = (y_1, \ldots, y_d) \) of the ordinary differential equation (ODE):

\[
\begin{align*}
y' &= p(y) \\
y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) is a vector of polynomials satisfies for all \( f(x) = \lim_{t \to \infty} y_1(t) \).

Example

\[
q(x) \quad y(t) \quad f(x)
\]
Computable Analysis = GPAC ? (again)

Theorem (Bournez, Campagnolo, Graça, Hainry)

The GPAC-computable functions are exactly the computable functions of the Computable Analysis.
Theorem (Bournez, Campagnolo, Graça, Hainry)

The GPAC-computable functions are exactly the computable functions of the Computable Analysis.

Proof.

- Any solution to a PIVP is computable + convergence
Theorem (Bournez, Campagnolo, Graça, Hainry)

The GPAC-computable functions are exactly the computable functions of the Computable Analysis.

Proof.

- Any solution to a PIVP is computable + convergence
- Simulate a Turing machine with a GPAC
What about complexity?
What about complexity?

- Computable Analysis: nice complexity theory (from Turing Machines)
What about complexity?

- Computable Analysis: nice complexity theory (from Turing Machines)
- General Purpose Analog Computer: nothing
What about complexity?

- Computable Analysis: nice complexity theory (from Turing Machines)
- General Purpose Analog Computer: nothing

**Conjecture**

Computable Analysis = General Purpose Analog Computer, *at the complexity level*
What about complexity?

- Computable Analysis: nice complexity theory (from Turing Machines)
- General Purpose Analog Computer: nothing

**Conjecture**

Computable Analysis = General Purpose Analog Computer, *at the complexity level*

First step: define a notion of complexity
# Time Scaling

<table>
<thead>
<tr>
<th>System</th>
<th>#1</th>
</tr>
</thead>
</table>
| ODE    | \[ \begin{align*}
    y'(t) &= p(y(t)) \\
    y(1) &= y_0
\end{align*} \] |

<table>
<thead>
<tr>
<th>#2</th>
</tr>
</thead>
</table>
| \[ \begin{align*}
    z'(t) &= u(t)p(z(t)) \\
    u'(t) &= u(t) \\
    z(t_0) &= y_0 \\
    u(1) &= 1
\end{align*} \] |
### Time Scaling

<table>
<thead>
<tr>
<th>System</th>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
</table>
| ODE    | \[
\begin{aligned}
    y'(t) &= p(y(t)) \\
    y(1) &= y_0
\end{aligned}
\] | \[
\begin{aligned}
    z'(t) &= u(t)p(z(t)) \\
    u'(t) &= u(t) \\
    z(t_0) &= y_0 \\
    u(1) &= 1
\end{aligned}
\] |

### Remark

Same curve, different speed: \( u(t) = e^t \) and \( z(t) = y(e^t) \)

### Example

The diagram illustrates the behavior of \( y(t) \) over different time scales, with \( y_0(x) \) and \( f(x) \) as reference points.
### Time Scaling

<table>
<thead>
<tr>
<th>System</th>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODE</td>
<td>( \begin{cases} y'(t) = p(y(t)) \ y(1) = y_0 \end{cases} )</td>
<td>( \begin{cases} z'(t) = u(t)p(z(t)) \ u'(t) = u(t) \ z(t_0) = y_0 \ u(1) = 1 \end{cases} )</td>
</tr>
</tbody>
</table>

**Computed Function**

\[ f(x) = \lim_{t \to \infty} y_1(t) = \lim_{t \to \infty} z_1(t) \]

**Remark**

Same curve, different speed: \( u(t) = e^t \) and \( z(t) = y(e^t) \)

**Example**

![Graph showing the function \( y(t) \) and \( f(x) \)](image)
# Time Scaling

<table>
<thead>
<tr>
<th>System</th>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
</table>
| ODE    | \[ \begin{align*} y'(t) &= p(y(t)) \\
|        | y(1) &= y_0 \end{align*} \] | \[ \begin{align*} z'(t) &= u(t)p(z(t)) \\
|        | u'(t) &= u(t) \\
|        | z(t_0) &= y_0 \\
|        | u(1) &= 1 \] |
| Computed Function | \[ f(x) = \lim_{t \to \infty} y_1(t) = \lim_{t \to \infty} z_1(t) \] |
| Convergence | Eventually | Exponentially faster |

## Example

![Graph](image.png)

- \( y_0(x) \)
- \( y(t) \)
- \( f(x) \)
Time Scaling

\[
\begin{align*}
\text{ODE} & \quad \begin{cases}
y'(t) = p(y(t)) \\
y(1) = y_0
\end{cases} & \quad \begin{cases}
z'(t) = u(t)p(z(t)) \\
u'(t) = u(t) \\
z(t_0) = y_0 \\
u(1) = 1
\end{cases}
\end{align*}
\]

\begin{tabular}{|c|c|c|}
\hline
Computed Function & \( f(x) = \lim_{t \to \infty} y_1(t) = \lim_{t \to \infty} z_1(t) \) & \\
\hline
Convergence & Eventually & Exponentially faster \\
\hline
Time for precision \( \mu \) & \( t_m(\mu) \) & \( t_m'(\mu) = \log(t_m(\mu)) \) \\
\hline
\end{tabular}

Example

\[ \|y_1(t_m(\mu)) - f(x)\| \leq \mu \]
# Time Scaling

### ODE

<table>
<thead>
<tr>
<th>ODE</th>
<th>( y' = p(y) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \left{ \begin{array}{l} z' = up(z) \ u' = u \end{array} \right. )</td>
<td></td>
</tr>
</tbody>
</table>

### Computed Function

\[
f(x) = \lim_{t \to \infty} y_1(t) = \lim_{t \to \infty} z_1(t)\]

### Time for precision \( \mu \)

| \( t \mu(\mu) \) | \( t \mu'(\mu) = \log(t \mu(\mu)) \) |

### Bounding box for ODE at time \( t \)

| \( sp(t) \) | \( sp'(t) = \max(sp(e^t), e^t) \) |

### Example

\[
sp'(t) = \sup_{\xi \in [1,t]} \| y(\xi) \|
\]

\[
sp(t) = \sup_{\xi \in [1,t]} \| z(\xi), u(\xi) \|
\]

\[
sp'(t) = \max(sp(e^t), e^t)
\]
## Time Scaling

<table>
<thead>
<tr>
<th>ODE</th>
<th>$y' = p(y)$</th>
<th>$\begin{cases} z' = u p(z) \ u' = u \end{cases}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computed Function</td>
<td>$f(x) = \lim_{t \to \infty} y_1(t) = \lim_{t \to \infty} z_1(t)$</td>
<td></td>
</tr>
<tr>
<td>Time for precision $\mu$</td>
<td>$tm(\mu)$</td>
<td>$tm'(\mu) = \log(tm(\mu))$</td>
</tr>
<tr>
<td>Bounding box for ODE at time $t$</td>
<td>$sp(t)$</td>
<td>$sp'(t) = \max(sp(e^t), e^t)$</td>
</tr>
<tr>
<td>Bounding box for ODE at precision $\mu$</td>
<td>$sp(tm(\mu))$</td>
<td>$\max(sp(tm(\mu)), tm(\mu))$</td>
</tr>
</tbody>
</table>

**Remark**
- $tm(\mu)$ and $sp(t)$ depend on the convergence rate
- $sp(tm(\mu))$ seems not
Proper Measures

Proper measures of “complexity”:
- time scaling invariant
- property of the curve
Proper Measures

Proper measures of “complexity”:
- time scaling invariant
- property of the curve

Possible choices:
- Bounding Box at precision $\mu \Rightarrow$ Ok but geometric interpretation ?
Proper Measures

Proper measures of “complexity”:
- time scaling invariant
- property of the curve

Possible choices:
- Bounding Box at precision $\mu \Rightarrow$ Ok but geometric interpretation?
- Length of the curve until precision $\mu \Rightarrow$ Much more intuitive
GPAC Computability (1)

Definition (GPAC-Computable Function)

\[ f \in GCOMP(\text{sp}, \text{tm}) \text{ iff } \exists p, q \text{ polynomials}, \text{ such that } \forall \alpha > 0, \forall \|x\| \leq \alpha, \exists y \text{ which satisfies:} \]

- \( \forall t \geq 0, y'(t) = p(y(t)) \) and \( y(0) = q(x) \)
- \( \forall \mu \geq 0, \forall t \geq \text{tm}(\alpha, \mu), |f(x) - y_1(t)| \leq e^{-\mu} \)
- \( \forall t \geq 0, \|y(t)\| \leq \text{sp}(\alpha, t) \)
**Definition (GPAC-Computable Function)**

Let \( f \in GCOMP(s_p, t_m) \) if there exist polynomials \( p, q \) such that for all \( \alpha > 0, \forall \|x\| \leq \alpha \), there exists a function \( y \) satisfying:

1. \( \forall t \geq 0, y'(t) = p(y(t)) \) and \( y(0) = q(x) \)
2. \( \forall \mu \geq 0, \forall t \geq t_m(\alpha, \mu), |f(x) - y_1(t)| \leq e^{-\mu} \)
3. \( \forall t \geq 0, \|y(t)\| \leq s_p(\alpha, t) \)

\( GP = GCOMP(\text{poly}, \text{poly}) \)
Definition (GPAC-Computable Function)

\( f \in GCOMP(sp, tm) \) iff \( \exists p, q \) polynomials, such that \( \forall \alpha > 0, \forall \|x\| \leq \alpha, \exists y \) which satisfies:

- \( \forall t \geq 0, y'(t) = p(y(t)) \) and \( y(0) = q(x) \)
- \( \forall \mu \geq 0, \forall t \geq tm(\alpha, \mu), |f(x) - y_1(t)| \leq e^{-\mu} \)
- \( \forall t \geq 0, \|y(t)\| \leq sp(\alpha, t) \)

\( GP = GCOMP(poly, poly) \)

Remark

- implies \( f(x) = \lim_{t \to \infty} y_1(t) \)
- can be extended to multi-dimensional functions
- can be defined over arbitrary input domain
Definition (Polytime GPAC-Computable Function (Alternative))

\( f \in GLEN(\text{len}) \iff \exists p, q \text{ polynomial such that } \forall \alpha > 0, \forall \|x\| \leq \alpha, \exists y \)

which satisfies:

- \( \forall t \geq 0, y'(t) = p(y(t)) \text{ and } y(0) = q(x) \)
- \( \forall \mu \geq 0, \forall t \geq \ell^{-1}(\text{len}(\alpha, \mu)), |f(x) - y_1(t)| \leq e^{-\mu} \)
- \( \ell(t) \) is the length of the curve \( y \) from 0 to \( t \).
**GPAC Computability (2)**

**Definition (Polytime GPAC-Computable Function (Alternative))**

\[ f \in GLEN(\text{len}) \text{ iff } \exists p, q \text{ polynomial such that } \forall \alpha > 0, \forall \|x\| \leq \alpha, \exists y \text{ which satisfies:} \]

- \( \forall t \geq 0, y'(t) = p(y(t)) \) and \( y(0) = q(x) \)
- \( \forall \mu \geq 0, \forall t \geq \ell^{-1}(\text{len}(\alpha, \mu)), |f(x) - y_1(t)| \leq e^{-\mu} \)
- \( \ell(t) \) is the length of the curve \( y \) from 0 to \( t \).

\[ GPLEN = GLEN(\text{poly}) \]
Definition (Polytime GPAC-Computable Function (Alternative))

\( f \in GLEN(len) \iff \exists p, q \text{ polynomial such that } \forall \alpha > 0, \forall \|x\| \leq \alpha, \exists y \) which satisfies:

- \( \forall t \geq 0, y'(t) = p(y(t)) \) and \( y(0) = q(x) \)
- \( \forall \mu \geq 0, \forall t \geq \ell^{-1}(len(\alpha, \mu)), |f(x) - y_1(t)| \leq e^{-\mu} \)
- \( \ell(t) \) is the length of the curve \( y \) from 0 to \( t \).

\[ GPLEN = GLEN(poly) \]

Remark

- implies \( f(x) = \lim_{t \to \infty} y_1(t) \)
- length of a curve: \( \ell(t) = \int_{t_0}^{t} \|p(y(u))\| \, du \)
- \( \ell^{-1}(l) = \) time to travel a length \( l \) on the curve \( y \)
Lemma (oversimplified)

\[ GP\text{LEN} = GP \]
Lemma (oversimplified)

\[ GPLEN = GP \]

Theorem

The polytime GPAC-computable functions (\( GP \)) are exactly the polytime computable functions of the Computable Analysis.
Computable Analysis = GPAC

Lemma (oversimplified)

\[ GPLEN = GP \]

Theorem

The polytime GPAC-computable functions (GP) are exactly the polytime computable functions of the Computable Analysis.

Proof.

- Any solution to a PIVP is polytime computable + exponential convergence
Computable Analysis = GPAC

Lemma (oversimplified)

\[ \text{GPLEN} = \text{GP} \]

Theorem

The polytime GPAC-computable functions (\text{GP}) are exactly the polytime computable functions of the Computable Analysis.

Proof.

- Any solution to a PIVP is polytime computable + exponential convergence
- Simulate a Turing machine with a GPAC
Proof sketch (1)

**Theorem**

If \( y(0) = y_0 \) and \( y' = p(y) \) Then \( y(t) \pm e^{-\mu} \) is computable is time

**Remark**

For \( L(t) = \text{poly}(t) \), it shows that \( y \) is polytime computable in the sense of Computable Analysis (nonuniformly).

**Proof.**

Numerical analysis, for another talk?
Proof sketch (1)

**Theorem**

If $y(0) = y_0$ and $y' = p(y)$ Then $y(t) \pm e^{-\mu}$ is computable is time

$$\text{poly}(\deg(p), L(t), \log \| y_0 \|, \log \Sigma p, \mu)^d$$

where

$$L(t) = \int_0^t \Sigma p \max(1, \| y(u) \|)^{\deg(p)} du \approx \text{length of } y \text{ over } [0, t]$$
Proof sketch (1)

Theorem

If \( y(0) = y_0 \) and \( y' = p(y) \) Then \( y(t) \pm e^{-\mu} \) is computable is time

\[
\text{poly}(\deg(p), L(t), \log\|y_0\|, \log\Sigma p, \mu)^d
\]

where

\[
L(t) = \int_0^t \Sigma p \max(1, \|y(u)\|)^{\deg(p)} du \approx \text{length of } y \text{ over } [0, t]
\]

Remark

For \( L(t) = \text{poly}(t) \), it shows that \( y \) is polytime computable in the sense of Computable Analysis (nonuniformly).
Proof sketch (1)

Theorem

If \( y(0) = y_0 \) and \( y' = p(y) \) Then \( y(t) \pm e^{-\mu} \) is computable is time

\[
\text{poly}(\deg(p), L(t), \log \|y_0\|, \log \Sigma p, \mu)^d
\]

where

\[
L(t) = \int_0^t \Sigma p \max(1, \|y(u)\|)^{\deg(p)} du \approx \text{length of } y \text{ over } [0, t]
\]

Remark

For \( L(t) = \text{poly}(t) \), it shows that \( y \) is polytime computable in the sense of Computable Analysis (nonuniformly).

Proof.

Numerical analysis, for another talk?
Proof sketch (2)

Simulating a Turing Machine with PIVP directly is tricky, we need more tools.
Proof sketch (2)

Simulating a Turing Machine with PIVP directly is tricky, we need more tools.

**Definition (Function Algebra)**

A *function algebra* $[\mathcal{F}; \text{OP}]$ is the smallest set of functions containing $\mathcal{F}$ and stable by all operators in $\text{OP}$. 
Simulating a Turing Machine with PIVP directly is tricky, we need more tools.

**Definition (Function Algebra)**

A *function algebra* $[\mathcal{F}; OP]$ is the smallest set of functions containing $\mathcal{F}$ and stable by all operators in $OP$.

**Example**

- $\mathbb{R}[X] = [0, -1, 1, X; +, \times]$
- primitive recursive $= [0, S, \pi_i; \circ, REC]$
- recursive $= [0, S, \pi_i; \circ, REC, MU]$
Proof sketch (3)

**Theorem**

\[ GP = [GP; LIM, \circ, IT] \]
Proof sketch (3)

**Theorem**

\[ GP = [GP; LIM, \circ, IT] \]

- \( LIM(f) = x \mapsto \lim_{\omega \to \infty} f(x, \omega) + \text{exponential convergence hypothesis} \)
Proof sketch (3)

\[ GP = [GP; LIM, \circ, IT] \]

- \( LIM(f) = x \mapsto \lim_{\omega \to \infty} f(x, \omega) \) + exponential convergence hypothesis
- \( IT(f) = (x, n) \mapsto f^n(x) \) + polynomial modulus of continuity hypothesis
Proof sketch (3)

Theorem

\[ GP = [GP; LIM, \circ, IT] \]

- \( LIM(f) = x \mapsto \lim_{\omega \to \infty} f(x, \omega) \) + exponential convergence hypothesis
- \( IT(f) = (x, n) \mapsto f^n(x) \) + polynomial modulus of continuity hypothesis

Proof.

Very technical
Proof sketch (4)

$f : \mathbb{R} \rightarrow \mathbb{R}$ polytime computable, $\mathcal{M}$ Turing Machine for $f$, $s_{\mathcal{M}}$ one step of $\mathcal{M}$

\[
f(x) = \lim_{\mu \rightarrow \infty} \mathcal{M}(x,\mu) = \lim_{\mu \rightarrow \infty} \lim_{n \rightarrow \infty} s_{\mathcal{M}}^{[n]}(x,\mu)
\]

and $s_{\mathcal{M}}$ can be built using composition and $GP$. 
GPAC as Language Recogniser

- GPAC as computable real function $\rightarrow$ Computable Analysis
GPAC as Language Recogniser

- GPAC as computable real function $\rightarrow$ Computable Analysis
- GPAC as language recogniser $\rightarrow$ classical computability ?
GPAC as Language Recogniser

- GPAC as computable real function $\rightarrow$ Computable Analysis
- GPAC as language recogniser $\rightarrow$ classical computability?

Remark

- words $\approx$ integers $\subseteq$ real numbers
GPAC as Language Recogniser

- GPAC as computable real function $\rightarrow$ Computable Analysis
- GPAC as language recogniser $\rightarrow$ classical computability?

Remark

- words $\approx$ integers $\subseteq$ real numbers
- decide $\approx$ \{Yes, No\} $\approx$ \{0, 1\} $\subseteq$ real numbers
GPAC as Language Recogniser

- GPAC as computable real function $\rightarrow$ Computable Analysis
- GPAC as language recogniser $\rightarrow$ classical computability ?

Remark

- words $\approx$ integers $\subseteq$ real numbers
- decide $\approx$ \{Yes, No\} $\approx$ \{0, 1\} $\subseteq$ real numbers
- language recogniser: special case of real function ?

\[ f : \mathbb{N} \subseteq \mathbb{R} \rightarrow \{0, 1\} \subseteq \mathbb{R} \]
GPAC as Language Recogniser

- GPAC as computable real function $\rightarrow$ Computable Analysis
- GPAC as language recogniser $\rightarrow$ classical computability?

**Remark**

- words $\approx$ integers $\subseteq$ real numbers
- decide $\approx \{\text{Yes, No}\} \approx \{0, 1\} \subseteq$ real numbers
- language recogniser: special case of real function?
  $$f : \mathbb{N} \subseteq \mathbb{R} \rightarrow \{0, 1\} \subseteq \mathbb{R}$$
- Yes but there is more!
Definition (GPAC-Recognisable Language)

\( L \subseteq \mathbb{N} \) GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{align*}
    y' &= p(y) \\
    y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in L \) then \( y_1(t) \geq 1 \) (accept)
- if \( x \notin L \) then \( y_1(t) \leq -1 \) (reject)
Definition (GPAC-Recognisable Language)

$L \subseteq \mathbb{N}$ is GPAC-recognisable if for any $x \in \mathbb{N}$, the solution $y$ to

$$
\begin{cases}
  y' = p(y) \\
y(t_0) = q(x)
\end{cases}
$$

where $p, q$ are vectors of polynomials

satisfies for $t \geq t_1(x)$:

- if $x \in L$ then $y_1(t) \geq 1$ (accept)
- if $x \notin L$ then $y_1(t) \leq -1$ (reject)

Theorem

The GPAC-recognisable languages are exactly the recursive languages.
**Definition (GPAC-Recognisable Language)**

$\mathcal{L} \subseteq \mathbb{N}$ is GPAC-recognisable if for any $x \in \mathbb{N}$, the solution $y$ to

$$
\begin{cases}
    y' = p(y) \\
y(t_0) = q(x)
\end{cases}
$$

where $p,q$ are vectors of polynomials satisfies for $t \geq t_1(x)$:

- if $x \in \mathcal{L}$ then $y_1(t) \geq 1$ (accept)
- if $x \notin \mathcal{L}$ then $y_1(t) \leq -1$ (reject)

**Theorem**

The GPAC-recognisable languages are exactly the recursive languages.

**Remark**

What about complexity?
Definition (Polytime GPAC-Recognisable Language)

\( \mathcal{L} \subseteq \mathbb{N} \) polytime GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{cases}
  y' = p(y) \\
y(t_0) = q(x)
\end{cases}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) (accept)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) (reject)
Definition (Polytime GPAC-Recognisable Language)

\( \mathcal{L} \subseteq \mathbb{N} \) polytime GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{aligned}
y' &= p(y) \\
y(t_0) &= q(x)
\end{aligned}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) (accept)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) (reject)

where \( t_1(x) = \ell^{-1}(\text{len}(\log(x))) \) where \( \ell(t) \) is the length of \( y \) from \( t_0 \) to \( t \) and \( \text{len} \) a polynomial.
Definition (Polytime GPAC-Recognisable Language)

\( \mathcal{L} \subseteq \mathbb{N} \) polytime GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{align*}
    y' &= p(y) \\
y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) (accept)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) (reject)

where \( t_1(x) = \ell^{-1}(\text{len} (\log(x))) \) where \( \ell(t) \) is the length of \( y \) from \( t_0 \) to \( t \) and \( \text{len} \) a polynomial.

Theorem

The class of polytime GPAC-recognisable languages is exactly \( P \).
Definition (Polytime GPAC-Recognisable Language)

\[ \mathcal{L} \subseteq \mathbb{N} \text{ polytime GPAC-recognisable if for any } x \in \mathbb{N}, \text{ the solution } y \text{ to } \]

\[
\begin{align*}
y' &= p(y) \\
y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) (accept)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) (reject)

where \( t_1(x) = \ell^{-1}(\text{len} (\log(x))) \) where \( \ell(t) \) is the length of \( y \) from \( t_0 \) to \( t \) and \( \text{len} \) a polynomial.

Theorem

The class of polytime GPAC-recognisable languages is exactly \( P \).

Remark (Why \( \log(x) \) ?)

Classical complexity measure: length of word \( \approx \log \) of value
Definition (Non-deterministic Polytime GPAC-Recognisable Language)

\( \mathcal{L} \subseteq \mathbb{N} \) non-deterministic polytime GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{align*}
    y' &= p(y, u) \\
    y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) for at least one digital controller \( u \)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) for all digital controller \( u \)

where \( t_1(x) = \ell^{-1}(\text{len}(\log(x))) \) and \( \text{len} \) a polynomial.
**Definition (Non-deterministic Polytime GPAC-Recognisable Language)**

\( \mathcal{L} \subseteq \mathbb{N} \) non-deterministic polytime GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{align*}
    y' &= p(y, u) \\
    y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) are vectors of polynomials

satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) for at least one digital controller \( u \)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) for all digital controller \( u \)

where \( t_1(x) = \ell^{-1}(\text{len}(\log(x))) \) and \( \text{len} \) a polynomial.

**Remark (Digital Controller)**

Digital Controller \( \approx u : \mathbb{R} \to \{0, 1\} \)
Definition (Non-deterministic Polytime GPAC-Recognisable Language)

\( \mathcal{L} \subseteq \mathbb{N} \) non-deterministic polytime GPAC-recognisable if for any \( x \in \mathbb{N} \), the solution \( y \) to

\[
\begin{align*}
    y' &= p(y, u) \\
    y(t_0) &= q(x)
\end{align*}
\]

where \( p, q \) are vectors of polynomials satisfies for \( t \geq t_1(x) \):

- if \( x \in \mathcal{L} \) then \( y_1(t) \geq 1 \) for at least one digital controller \( u \)
- if \( x \notin \mathcal{L} \) then \( y_1(t) \leq -1 \) for all digital controller \( u \)

where \( t_1(x) = \ell^{-1}(\text{len}(\log(x))) \) and \( \text{len} \) a polynomial.

Remark (Digital Controller)

Digital Controller \( \approx u : \mathbb{R} \rightarrow \{0, 1\} \)

Theorem

The class of non-deterministic polytime GPAC-recognisable languages is exactly \( NP \).
Conclusion

- Complexity theory for the GPAC
Conclusion

- Complexity theory for the GPAC
- Equivalence with Computable Analysis for polynomial time
Future Work

- Notion of reduction?
Future Work

- Notion of reduction?
- Space complexity?
Questions?

Do you have any questions?