Computational Complexity of the GPAC

Amaury Pouly
Joint work with Olivier Bournez and Daniel Graça

April 28, 2014

Outline

The case of discrete computations

Many models:

- Recursive functions
- Turing machines
- λ-calculus
- circuits
- ...

The case of discrete computations

Many models:

- Recursive functions
- Turing machines
- λ-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 ? $\Rightarrow$ No (GPAC $\neq$ 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 ? $\Rightarrow$ No (GPAC $\neq$ BSS)
- Comparison with digital models of computation ?

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 ? $\Rightarrow$ No (GPAC $\neq$ BSS)
- Comparison with digital models of computation ? $\Rightarrow$ How ?
- What is a “reasonable” model ? $\Rightarrow$ Unclear

GPAC

General Purpose Analog Computer

- by Claude Shanon (1941)

GPAC

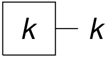
General Purpose Analog Computer

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

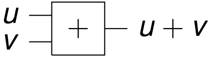
GPAC

General Purpose Analog Computer

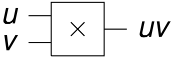
- by Claude Shanon (1941)
- idealization of an analog computer: Differential Analyzer
- circuit built from:



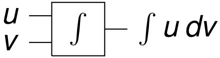
A constant unit



An adder unit



An multiplier unit



An integrator unit

GPAC: beyond the circuit approach

Theorem

y is generated by a GPAC iff it is a component of the solution $y = (y_1, \dots, y_d)$ 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.

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):

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

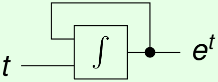
where p is a vector of polynomials.

Remark

Other point of view: continuous dynamical system

GPAC: examples

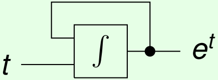
Example (One variable, linear system)



$$\begin{cases} y' = y \\ y(0) = 1 \end{cases}$$

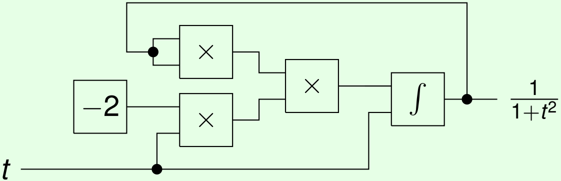
GPAC: examples

Example (One variable, linear system)



$$\begin{cases} y' = y \\ y(0) = 1 \end{cases}$$

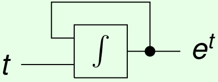
Example (One variable, nonlinear system)



$$\begin{cases} y' = -2ty^2 \\ y(0) = 1 \end{cases}$$

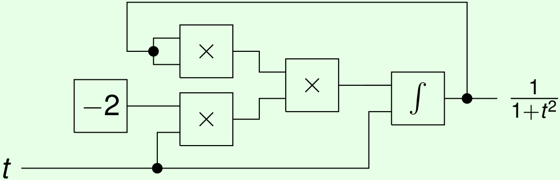
GPAC: examples

Example (One variable, linear system)



$$\begin{cases} y' &= y \\ y(0) &= 1 \end{cases}$$

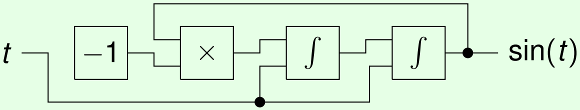
Example (Two variable, nonlinear system)



$$\begin{cases} y' &= -2ty^2 \\ y(0) &= 1 \\ t' &= 1 \\ t(0) &= 0 \end{cases}$$

GPAC: examples

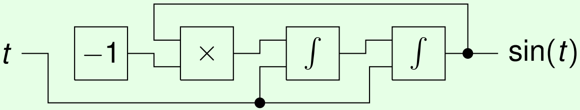
Example (Two variables, linear system)



$$\begin{cases} y' &= z \\ z' &= -y \\ y(0) &= 0 \\ z(0) &= 1 \end{cases}$$

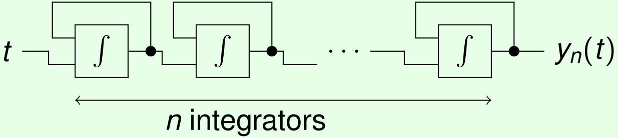
GPAC: examples

Example (Two variables, linear system)



$$\begin{cases} y' &= z \\ z' &= -y \\ y(0) &= 0 \\ z(0) &= 1 \end{cases}$$

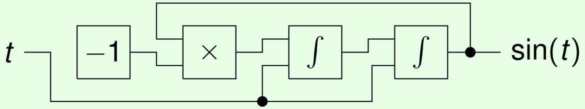
Example (Not so nice example)



$$\begin{cases} y_1' = y_1 \\ y_2' = y_2 y_1' \\ \vdots \\ y_n' = y_n y_{n-1}' \end{cases}$$

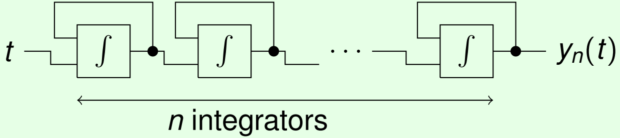
GPAC: examples

Example (Two variables, linear system)



$$\begin{cases} y' &= z \\ z' &= -y \\ y(0) &= 0 \\ z(0) &= 1 \end{cases}$$

Example (Not so nice example)



$$\begin{cases} y_1(t) = e^t \\ y_2(t) = e^{e^t} \\ \quad \dots \\ y_n(t) = e^{e^{\cdot^{\cdot^{\cdot^{t}}}}} \end{cases}$$

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

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:

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

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:

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

Example

Rational numbers, π, 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:

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

Example

Rational numbers, π, e, ...

Example (Non-computable real)

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

where

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

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 $\mathcal{O}$ computing x, $M^{\mathcal{O}}$ computes $f(x)$.

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 $\mathcal{O}$ computing x, $M^{\mathcal{O}}$ computes $f(x)$.

Definition (Equivalent)

A function $f : \mathbb{R} \to \mathbb{R}$ 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 $\mathcal{O}$ computing x, $M^{\mathcal{O}}$ 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 $f(r)$.

Example

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

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 $\mathcal{O}$ computing x, $M^{\mathcal{O}}$ computes $f(x)$.

Definition (Equivalent)

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

Example

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

Example (Counter-Example)

$$f(x) = \lceil x \rceil$$

Computable Analysis = GPAC ?

Computable Analysis = GPAC ?

Seems not:

Computable Analysis = GPAC ?

Seems not:

- Solutions of a GPAC are analytic

Computable Analysis = GPAC ?

Seems not:

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

Theorem (☹)

Computable Analysis $\neq$ General Purpose Analog Computer

Computable Analysis = GPAC ?

Seems not:

- Solutions of a GPAC are analytic
- $x \to |x|$ 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):

$$\begin{cases} y' = p(y) \\ y(t_0) = y_0 \end{cases}$$ 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):

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

where p is a vector of polynomials

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):

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

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 $y = (y_1, \dots, y_d)$ of the ordinary differential equation (ODE):

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

where p,q is a vector of polynomials

satisfies for all $f(x) = \lim_{t \to \infty} y_1(t)$.

Example

Computable Analysis = GPAC ? (again)

Theorem (Bournez, Campagnolo, Graça, Hainry)

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

Computable Analysis = GPAC ? (again)

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

Computable Analysis = GPAC ? (again)

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

$\square$

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

System	#1	#2
ODE	$\begin{cases} y'(t) = p(y(t)) \\ y(1) = y_0 \end{cases}$	$\begin{cases} z'(t) = u(t)p(z(t)) \\ u'(t) = u(t) \\ z(t_0) = y_0 \\ u(1) = 1 \end{cases}$

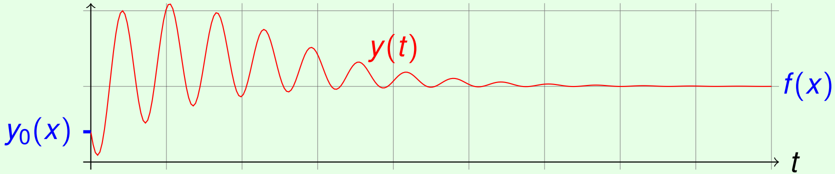
Time Scaling

System	#1	#2
ODE	$\begin{cases} y'(t) = p(y(t)) \\ y(1) = y_0 \end{cases}$	$\begin{cases} z'(t) = u(t)p(z(t)) \\ u'(t) = u(t) \\ z(t_0) = y_0 \\ u(1) = 1 \end{cases}$

Remark

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

Example



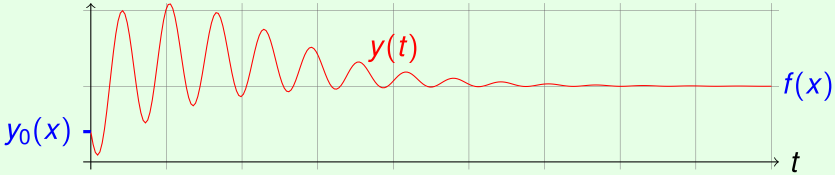
Time Scaling

System	#1	#2
ODE	$\begin{cases} y'(t) = p(y(t)) \\ y(1) = y_0 \end{cases}$	$\begin{cases} z'(t) = u(t)p(z(t)) \\ u'(t) = u(t) \\ z(t_0) = y_0 \\ u(1) = 1 \end{cases}$
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 $z(t) = y(e^t)$

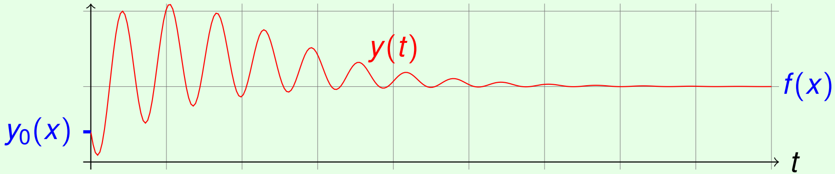
Example



Time Scaling

System	#1	#2
ODE	$\begin{cases} y'(t) = p(y(t)) \\ y(1) = y_0 \end{cases}$	$\begin{cases} z'(t) = u(t)p(z(t)) \\ u'(t) = u(t) \\ z(t_0) = y_0 \\ u(1) = 1 \end{cases}$
Computed Function	$f(x) = \lim_{t\to\infty} y_1(t) = \lim_{t\to\infty} z_1(t)$	
Convergence	Eventually	Exponentially faster

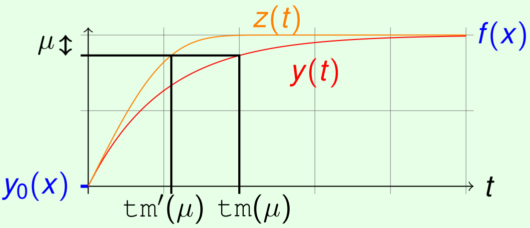
Example



Time Scaling

ODE	$\begin{cases} y'(t) = p(y(t)) \\ y(1) = y_0 \end{cases}$	$\begin{cases} z'(t) = u(t)p(z(t)) \\ u'(t) = u(t) \\ z(t_0) = y_0 \\ u(1) = 1 \end{cases}$
Computed Function	$f(x) = \lim_{t\to\infty} y_1(t) = \lim_{t\to\infty} z_1(t)$	
Convergence	Eventually	Exponentially faster
Time for precision μ	$\mathtt{tm}(\mu)$	$\mathtt{tm}'(\mu) = \log(\mathtt{tm}(\mu))$

Example

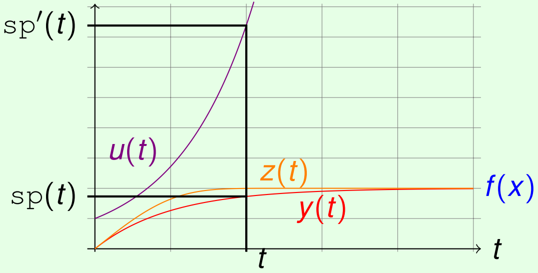


$$\|y_1(\mathtt{tm}(\mu)) - f(x)\| \leqslant \mu$$

Time Scaling

ODE	$y' = p(y)$	$\begin{cases} z' = up(z) \\ u' = u \end{cases}$
Computed Function	$f(x) = \lim_{t\to\infty} y_1(t) = \lim_{t\to\infty} z_1(t)$	
Time for precision μ	$\mathtt{tm}(\mu)$	$\mathtt{tm}'(\mu) = \log(\mathtt{tm}(\mu))$
Bounding box for ODE at time t	$\mathtt{sp}(t)$	$\mathtt{sp}'(t) = \max(\mathtt{sp}(e^t), e^t)$

Example



$$\mathtt{sp}(t) = \sup_{\xi\in[1,t]} \|y(\xi)\|$$

$$\mathtt{sp}'(t) = \sup_{\xi\in[1,t]} \|z(\xi), u(\xi)\|$$

Time Scaling

ODE	$y' = p(y)$	$\begin{cases} z' = up(z) \\ u' = u \end{cases}$
Computed Function	$f(x) = \lim_{t\to\infty} y_1(t) = \lim_{t\to\infty} z_1(t)$	
Time for precision μ	$\mathtt{tm}(\mu)$	$\mathtt{tm}'(\mu) = \log(\mathtt{tm}(\mu))$
Bounding box for ODE at time t	$\mathtt{sp}(t)$	$\mathtt{sp}'(t) = \max(\mathtt{sp}(e^t), e^t)$
Bounding box for ODE at precision μ	$\mathtt{sp}(\mathtt{tm}(\mu))$	$\max(\mathtt{sp}(\mathtt{tm}(\mu)), \mathtt{tm}(\mu))$

Remark

- $\mathtt{tm}(\mu)$ and $\mathtt{sp}(t)$ depend on the convergence rate
- $\mathtt{sp}(\mathtt{tm}(\mu))$ 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)

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

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

GPAC Computability (1)

Definition (GPAC-Computable Function)

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

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

$$GP = GCOMP(\text{poly}, \text{poly})$$

GPAC Computability (1)

Definition (GPAC-Computable Function)

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

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

$$GP = GCOMP(\text{poly}, \text{poly})$$

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

GPAC Computability (2)

Definition (Polytime GPAC-Computable Function (Alternative))

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

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

GPAC Computability (2)

Definition (Polytime GPAC-Computable Function (Alternative))

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

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

$$GPLEN = GLEN(\text{poly})$$

GPAC Computability (2)

Definition (Polytime GPAC-Computable Function (Alternative))

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

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

$$GPLEN = GLEN(\text{poly})$$

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

Computable Analysis = GPAC

Lemma (oversimplified)

$$GPLEN = GP$$

Computable Analysis = GPAC

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)

$$GPLEN = GP$$

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
- 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

Proof sketch (1)

Theorem

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

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

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

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

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 $L(t)=\operatorname{poly}(t)$, 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* $[\mathcal{F}; OP]$ is the smallest set of functions containg $\mathcal{F}$ and stable by all operators in OP.

Proof sketch (2)

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 containg $\mathcal{F}$ 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

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

- $LIM(f) = x \mapsto \lim_{\omega \to \infty} f(x, \omega)$ + exponential convergence 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 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 $\square$

Proof sketch (4)

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

$$\begin{aligned} f(x) &= \lim_{\mu\to\infty} \mathcal{M}(x,\mu) \\ &= \lim_{\mu\to\infty} \lim_{n\to\infty} s_{\mathcal{M}}^{[n]}(x,\mu) \end{aligned}$$

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 $\to$ Computable Analysis
- GPAC as language recogniser $\to$ 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 ?
  $f : \mathbb{N} \subseteq \mathbb{R} \to \{0,1\} \subseteq \mathbb{R}$
- Yes but there is more !

Definition (GPAC-Recognisable Language)

$\mathcal{L} \subseteq \mathbb{N}$ 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 $t \geqslant t_1(x)$:

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

Definition (GPAC-Recognisable Language)

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

$$\begin{cases} y' &= p(y) \\ y(t_0) &= q(x) \end{cases} \qquad \text{where } p,q \text{ are vectors of polynomials}$$

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

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

Theorem

The GPAC-recognisable languages are exactly the recursive languages.

Definition (GPAC-Recognisable Language)

$\mathcal{L} \subseteq \mathbb{N}$ 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 $t \geqslant t_1(x)$:

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

Theorem

The GPAC-recognisable languages are exactly the recursive languages.

Remark

What about complexity ?

Definition (Polytime GPAC-Recognisable Language)

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

$$\begin{cases} y' &= p(y) \\ y(t_0) &= q(x) \end{cases} \qquad \text{where } p,q \text{ are vectors of polynomials}$$

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

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

Definition (Polytime GPAC-Recognisable Language)

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

$$\begin{cases} y' &= p(y) \\ y(t_0) &= q(x) \end{cases} \qquad \text{where } p,q \text{ are vectors of polynomials}$$

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

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

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

Definition (Polytime GPAC-Recognisable Language)

$\mathcal{L} \subseteq \mathbb{N}$ poyltime 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 $t \geqslant t_1(x)$:

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

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

Theorem

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

Definition (Polytime GPAC-Recognisable Language)

$\mathcal{L} \subseteq \mathbb{N}$ poyltime 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 $t \geqslant t_1(x)$:

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

where $t_1(x) = \ell^{-1}(\texttt{len}(\log(x))$ where $\ell(t)$ is the length of y from t_0 to t and `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 poyltime GPAC-recognisable if for any $x \in \mathbb{N}$, the solution y to

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

where p,q are vectors of polynomials

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

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

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

Definition (Non-deterministic Polytime GPAC-Recognisable Language)

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

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

where p,q are vectors of polynomials

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

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

where $t_1(x) = \ell^{-1}(\mathtt{len}(\log(x))$ and $\mathtt{len}$ 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 poyltime GPAC-recognisable if for any $x \in \mathbb{N}$, the solution y to

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

where p,q are vectors of polynomials

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

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

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

Remark (Digital Controller)

Digital Controller $\approx u : \mathbb{R} \to \{0, 1\}$

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 ?