Computational Complexity of the GPAC

Amaury Pouly
Joint work with Olivier Bournez and Daniel Graça

April 10, 2014

Outline

GPAC

General Purpose Analog Computer

- by Claude Shanon (1941)

GPAC

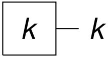
General Purpose Analog Computer

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

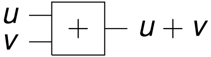
GPAC

General Purpose Analog Computer

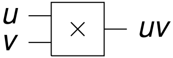
- by Claude Shanon (1941)
- idealization of an analog computer: Differential Analyzer
- circuit built from:



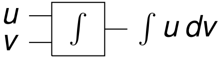
A constant unit



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 $y = (y_1, \ldots, y_d)$ 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.

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.

Remark

- continuous dynamical system

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.

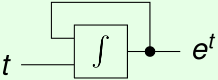
Remark

- continuous dynamical system
- the GPAC is just one reason to look at them[a]

[a] Ask question

GPAC: examples

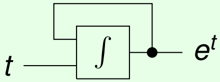
Example (One variable, linear system)



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

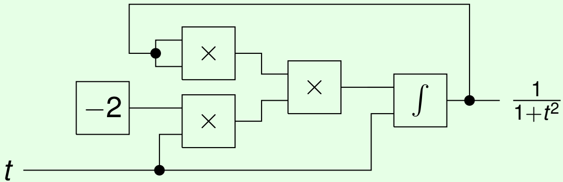
GPAC: examples

Example (One variable, linear system)



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

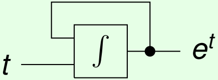
Example (One variable, nonlinear system)



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

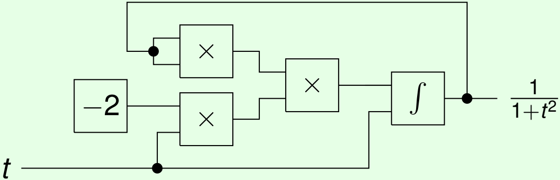
GPAC: examples

Example (One variable, linear system)



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

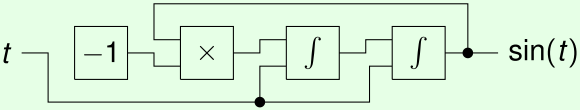
Example (Two variable, nonlinear system)



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

GPAC: examples

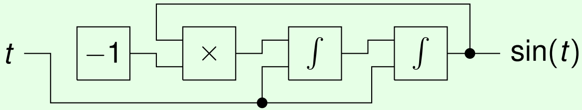
Example (Two variables, linear system)



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

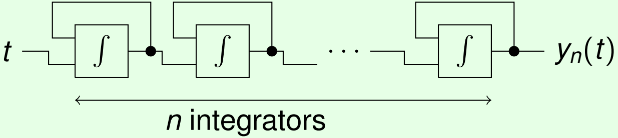
GPAC: examples

Example (Two variables, linear system)



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

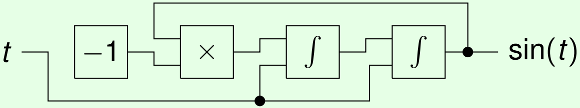
Example (Not so nice example)



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

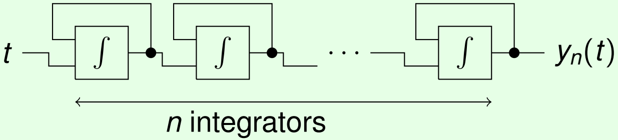
GPAC: examples

Example (Two variables, linear system)



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

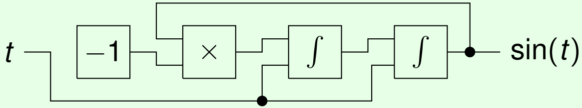
Example (Not so nice example)



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

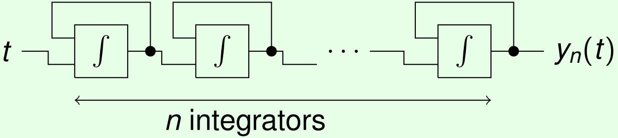
GPAC: examples

Example (Two variables, linear system)



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

Example (Not so nice example)



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

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, ...

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 (Counter-Example)

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

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 $f : \mathbb{R} \to \mathbb{R}$ 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, $e^{\cdot}$, $\sqrt{\cdot}$, ...

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
- $x \rightarrow |x|$ is computable but not analytic

Theorem (☹)

Computable Analysis $\neq$ General Purpose Analog Computer

Computable Analysis = GPAC ?

Seems not:

- Solutions of a GPAC are analytic
- $x \to |x|$ 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 $y = (y_1, \dots, y_d)$ 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

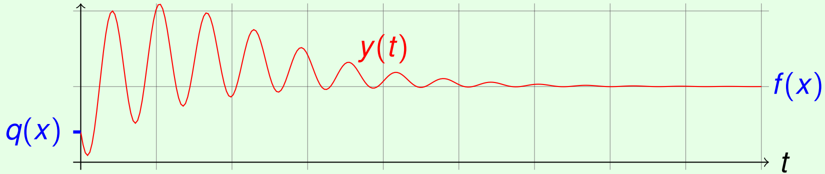
f is **computable** by a GPAC iff for all $x \in \mathbb{R}$ the solution $y = (y_1, \dots, y_d)$ of the ordinary differential equation (ODE):

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

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 (☺)

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

Computable Analysis = GPAC ? (again)

Theorem (☺)

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 (☺)

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[a]

□

[a]Details on blackboard

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

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*

Time Scaling

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

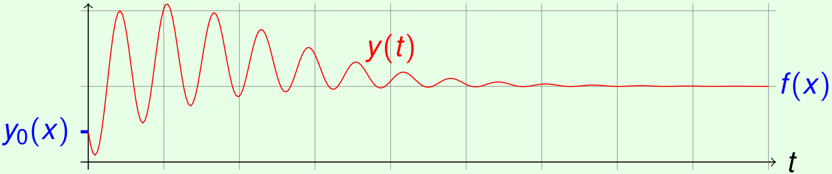
Time Scaling

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



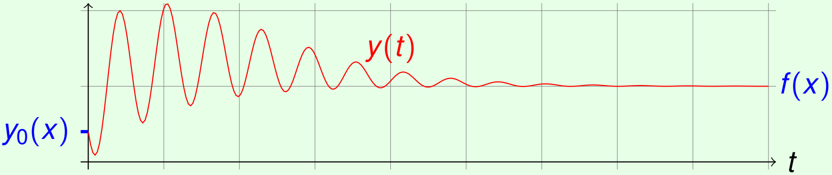
Time Scaling

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 $z(t) = y(e^t)$

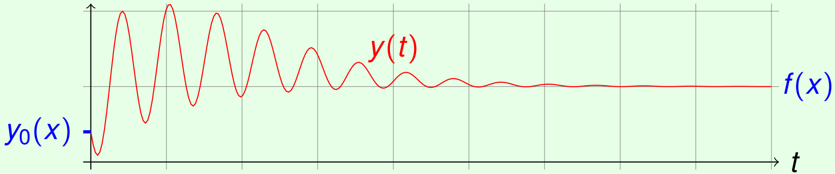
Example



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

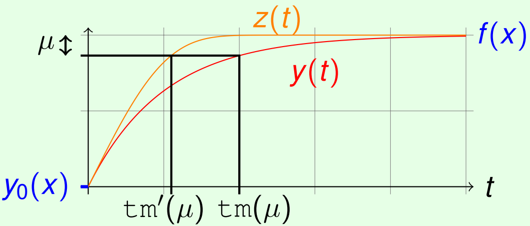
Example



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

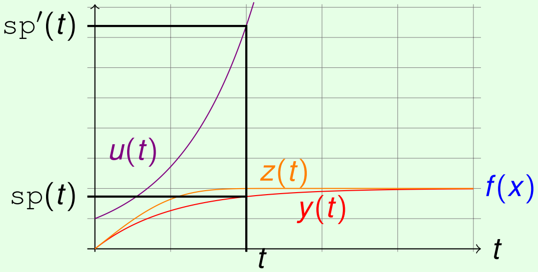


$$\|y_1(\mathtt{tm}(\mu)) - f(x)\| \leqslant \mu$$

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

Definition (Polytime GPAC-Computable Function)

f is **polytime** 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 are vectors of polynomials

satisfies $\left\| f(x) - y_1(\ell^{-1}(\mathtt{len}(x,\mu)) \right\| \leqslant e^{-\mu}$ where

- $\mathtt{len}$ is a polynomial [polytime]
- $\ell(t)$ is the length of the curve y from t_0 to t.
- $\ell^{-1}(l)$ is the time to reach a length l on the curve y

Definition (Polytime GPAC-Computable Function)

f is **polytime** 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 are vectors of polynomials

satisfies $\left\|f(x) - y_1(\ell^{-1}(\mathtt{len}(x,\mu))\right\| \leqslant e^{-\mu}$ where

- $\mathtt{len}$ is a polynomial [polytime]
- $\ell(t)$ is the length of the curve y from t_0 to t.
- $\ell^{-1}(l)$ is the time to reach a length l on the curve y

Remark

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

Computable Analysis = GPAC ?

Theorem (Almost ☺)

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

Computable Analysis = GPAC ?

Theorem (Almost ☺)

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

Remark (Polytime computable in CA)

f polytime computable:

- polynomial modulus of continuity mc: $\|x-y\| \leqslant 2^{-\mathrm{mc}(\mu)} \Rightarrow \|f(x)-f(y)\| \leqslant 2^{-\mu}$
- polynomial time computable over $\mathbb{Q}$

Conclusion

- Complexity theory for the GPAC

Conclusion

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

Conclusion

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

Not mentioned in this talk:

- The GPAC as a language recogniser

Conclusion

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

Not mentioned in this talk:

- The GPAC as a language recogniser
- Equivalence with *P* and *NP*

Future Work

- Notion of reduction ?

Future Work

- Notion of reduction ?
- Space complexity ?

Questions ?

- Do you have any questions ?

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

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

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 (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.

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

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

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

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

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.

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

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

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