Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length

Olivier Bournez, Daniel Graça and Amaury Pouly

July 13, 2015

Main result and consequences

Theorem (Informal)

$$\text{PTIME} = \text{PIVP of polynomial length}$$

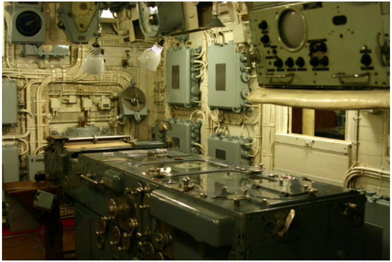
PIVP: Ordinary Differential Equations (ODE) with polynomial right-hand side.

- Implicit complexity: purely continuous (time and space) characterization of PTIME
- Continuous-time models of computations: Turing machines and the GPAC are equivalent at the complexity level

Digital vs analog computers

Digital vs analog computers

VS



Let's model!

Physical Computer	**Model**
Laptop, ...	Turing machines λ-calculus Recursive functions Circuits Discrete dynamical systems
Differential Analyzer, ...	GPAC Continuous dynamical systems

Let's model!

Physical Computer	**Model**
Laptop, ...	Turing machines λ-calculus Recursive functions Circuits Discrete dynamical systems
Differential Analyzer, ...	GPAC Continuous dynamical systems

Church Thesis

All reasonable models of computation are equivalent.

Let's model!

Physical Computer	**Model**
Laptop, ...	Turing machines λ-calculus Recursive functions Circuits Discrete dynamical systems
Differential Analyzer, ...	GPAC Continuous dynamical systems

Church Thesis

All reasonable models of computation are equivalent.

Implicit corollary

Some models are too general/unreasonable.

Let's model!

Physical Computer	**Model**
Laptop, ...	Turing machines λ-calculus Recursive functions Circuits Discrete dynamical systems
Differential Analyzer, ...	**GPAC**$\rightarrow$**reasonable ?** Continuous dynamical systems

Church Thesis

All reasonable models of computation are equivalent.

Implicit corollary

Some models are too general/unreasonable.

General Purpose Analog Computer (GPAC)

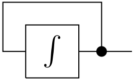
- invented by Shannon (1941)
- idealization of the Differential Analyzer:

- circuits made of:

$\boxed{k} - k$	Constant	$u, v \to \boxed{\times} - uv$	Multiplier
$u, v \to \boxed{+} - u + v$	Adder	$u \to \boxed{\int} - \int u$	Integrator

Examples of GPAC

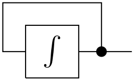
Exponential:



$y(t) \quad \rightsquigarrow \quad y = \int y \quad \rightsquigarrow \quad y(t) = \exp(t)$

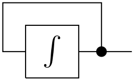
Examples of GPAC

Exponential:

 $y(t) \quad \rightsquigarrow \quad y' = y \quad \rightsquigarrow \quad y(t) = \exp(t)$

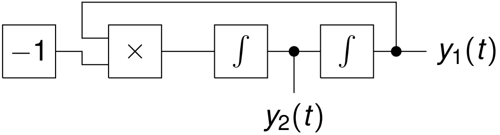
Examples of GPAC

Exponential:



$$y(t) \quad \rightsquigarrow \quad y' = y \quad \rightsquigarrow \quad y(t) = \exp(t)$$

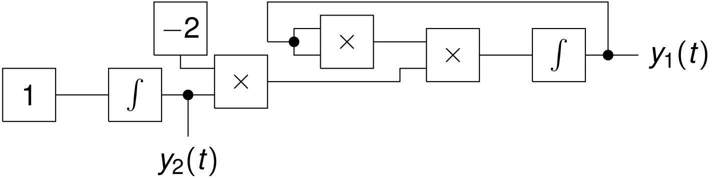
(Co)sine:



$$\begin{cases} y_1' = y_2 \\ y_2' = -y_1 \end{cases} \quad \rightsquigarrow \quad \begin{cases} y_1(t) = \sin(t) \\ y_2(t) = \cos(t) \end{cases}$$

Examples of GPAC

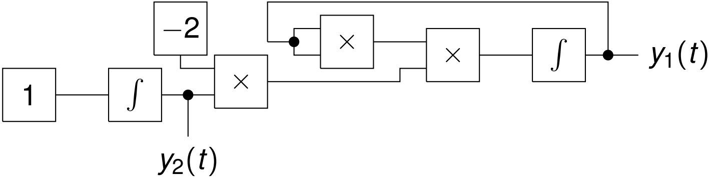
Rational function:



$$\begin{cases} y_1' = -2y_2y_1^2 \\ y_2' = 1 \end{cases} \quad \rightsquigarrow \quad \begin{cases} y_1(t) = \frac{1}{1+t^2} \\ y_2(t) = t \end{cases}$$

Examples of GPAC

Rational function:



$$\begin{cases} y_1' = -2y_2y_1^2 \\ y_2' = 1 \end{cases} \quad \rightsquigarrow \quad \begin{cases} y_1(t) = \frac{1}{1+t^2} \\ y_2(t) = t \end{cases}$$

Theorem (Graça and Costa)

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

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

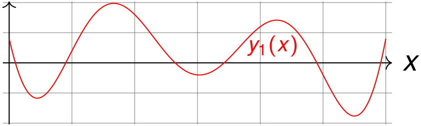
where p is a vector of polynomials.

Computing with the GPAC

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



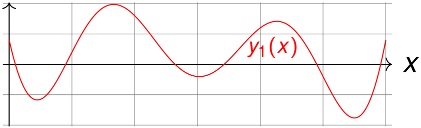
Shannon's notion

Computing with the GPAC

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



Shannon's notion

sin, cos, exp, log, ...

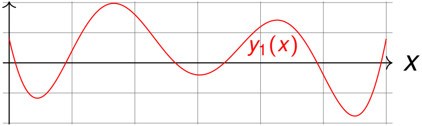
Strictly weaker than Turing machines [Shannon, 1941]

Computing with the GPAC

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



Shannon's notion

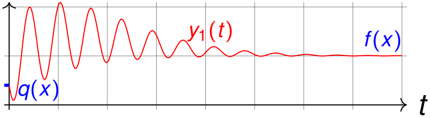
sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

Computable

$$\begin{cases} y(0) = q(x) & x \in \mathbb{R} \\ y'(t) = p(y(t)) & t \in \mathbb{R}_+ \end{cases}$$

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



Modern notion

Computing with the GPAC

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$

Shannon's notion

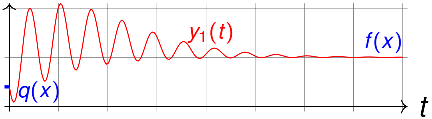
sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

Computable

$$\begin{cases} y(0) = q(x) & x \in \mathbb{R} \\ y'(t) = p(y(t)) & t \in \mathbb{R}_+ \end{cases}$$

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



Modern notion

sin, cos, exp, log, Γ, ζ, ...

Turing powerful [Bournez et al., 2007]

Different kinds of equivalence

Theorem (Bournez et al)

The GPAC is equivalent to Turing machines for computability.

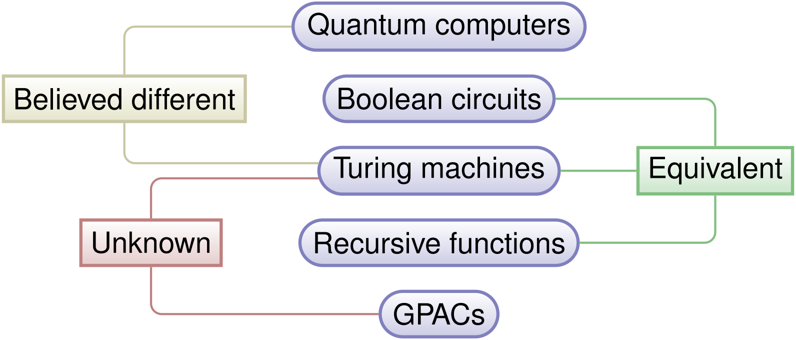
- Computability: compute the same functions

Different kinds of equivalence

Theorem (Bournez et al)

The GPAC is equivalent to Turing machines for computability.

- Computability: compute the same functions
- Complexity: same functions with same “complexity”

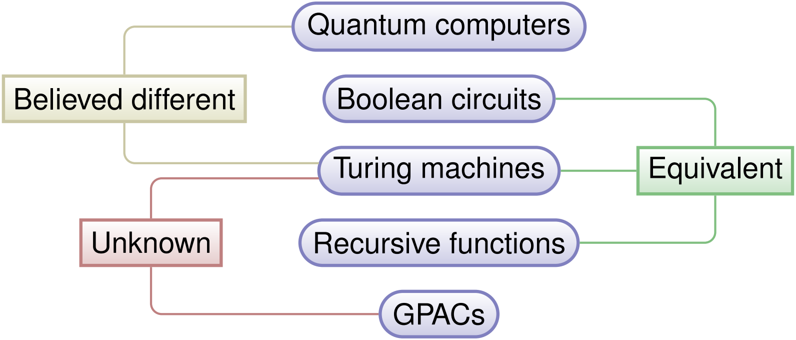


Different kinds of equivalence

Theorem (Bournez et al)

The GPAC is equivalent to Turing machines for computability.

- Computability: compute the same functions
- Complexity: same functions with same "complexity"



Main Result of the paper

Turing machines and GPACs are equivalent for complexity.

Time complexity for continuous systems

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

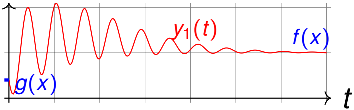
Time complexity for continuous systems

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

Intuitive definition

$T(x,\mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = q(x) \qquad y' = p(y)$



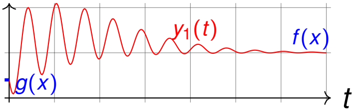
Time complexity for continuous systems

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

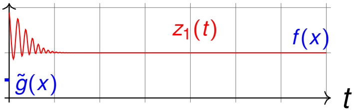
Intuitive definition

$T(x,\mu) =$ first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = q(x) \qquad y' = p(y)$ $\qquad\qquad\qquad z(t) = y(e^t)$



$\rightsquigarrow$



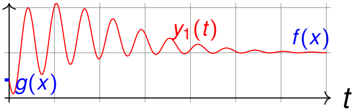
Time complexity for continuous systems

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

Intuitive definition

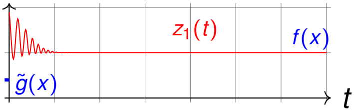
$T(x,\mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = q(x) \qquad y' = p(y)$



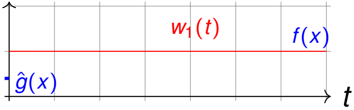
$\rightsquigarrow$

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



$w(t) = y(e^{e^t})$

$\rightsquigarrow$



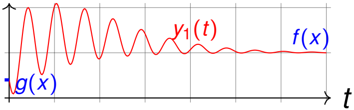
Time complexity for continuous systems

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

Intuitive definition

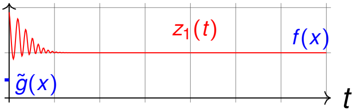
$T(x,\mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = q(x) \qquad y' = p(y)$



$\rightsquigarrow$

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

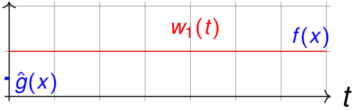


Observation

This definition is broken: all functions have arbitrarily small complexity.

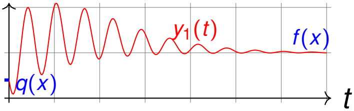
$w(t) = y(e^{e^t})$

$\rightsquigarrow$



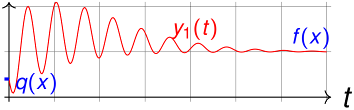
Time-space correlation of the GPAC

$$y(0) = q(x) \qquad y' = p(y)$$

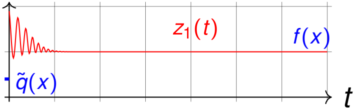


Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$ $\qquad\qquad\qquad\qquad$ $z(t) = y(e^t)$

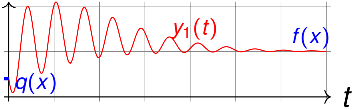


$\rightsquigarrow$



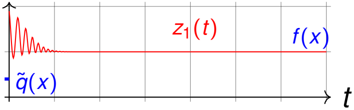
Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$

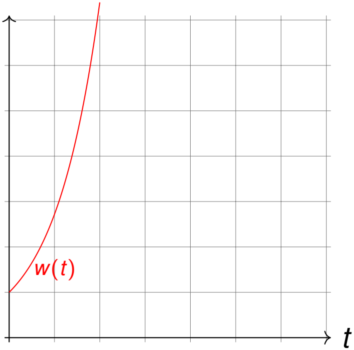


$\rightsquigarrow$

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



extra component: $w(t) = e^t$

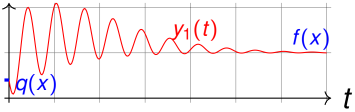


Observation

Time scaling costs “space”.

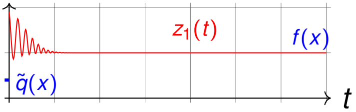
Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$

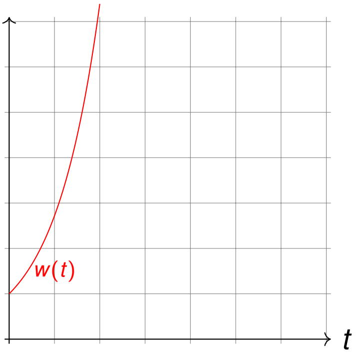


$\rightsquigarrow$

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



extra component: $w(t) = e^t$



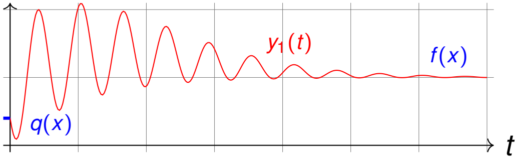
Observation

Time scaling costs "space".

$\rightsquigarrow$

Time complexity for the GPAC must involve time and space !

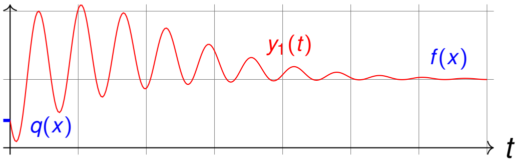
Two equivalent notions of complexity



$$\begin{cases} y(0) = q(x) \\ y'(t) = p(y(t)) \end{cases}$$

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

Two equivalent notions of complexity



$$\begin{cases} y(0) = q(x) \\ y'(t) = p(y(t)) \end{cases}$$

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

Length based complexity: L

$$\begin{aligned} \ell(t) &= \text{length of } y \text{ over } [0,t] \\ &= \int_0^t \|p(y(u))\|\,du \end{aligned}$$

$L(x,\mu) =$ length $\ell(t)$ so that

$$\|y_1(t) - f(x)\| \leqslant e^{-\mu}$$

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

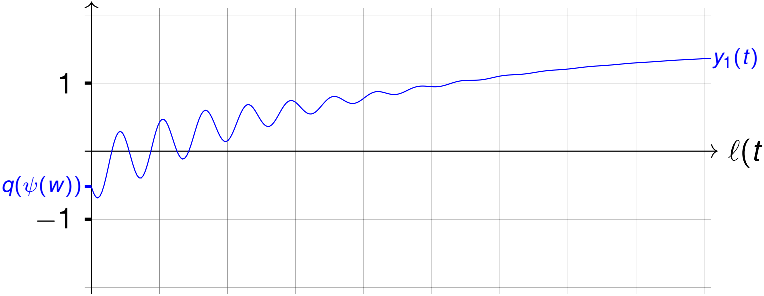
satisfies:

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

satisfies:



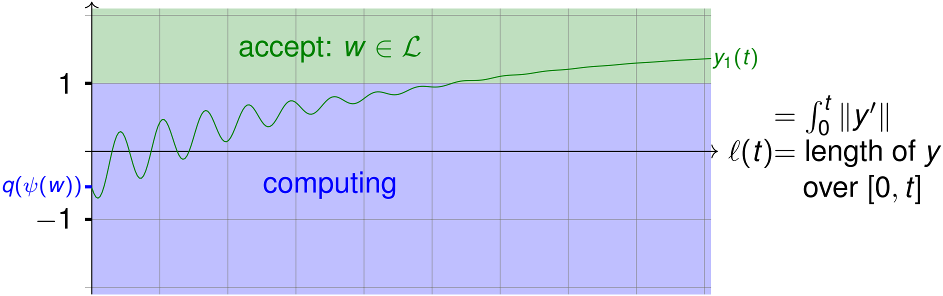
$\ell(t) =$ length of y over $[0,t]$ $= \int_0^t \|y'\|$

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

satisfies:



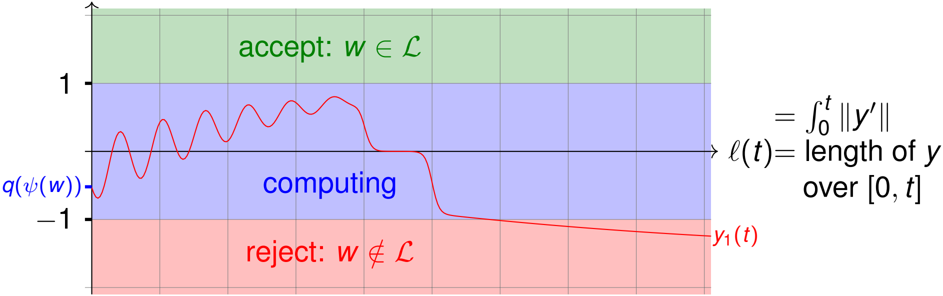
1. if $y_1(t) \geqslant 1$ then $w \in \mathcal{L}$

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

satisfies:



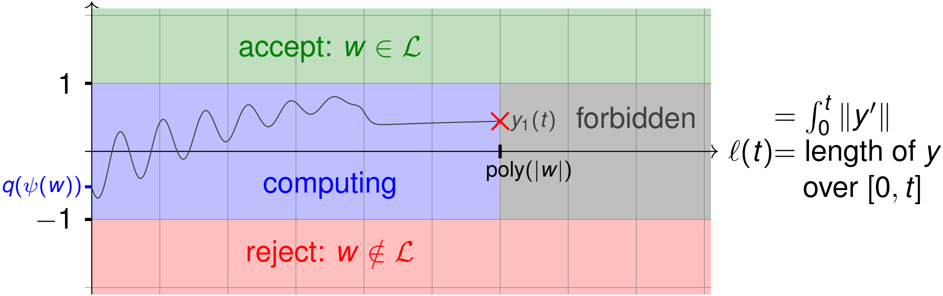
2. if $y_1(t) \leqslant -1$ then $w \notin \mathcal{L}$

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

satisfies:



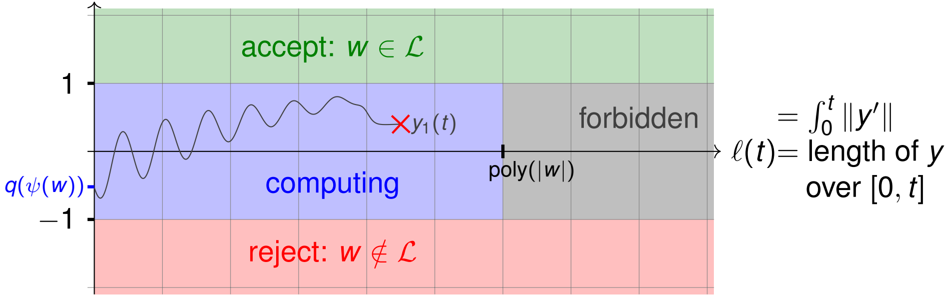
3. if $\ell(t) \geqslant \text{poly}(|w|)$ then $|y_1(t)| \geqslant 1$

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

satisfies:



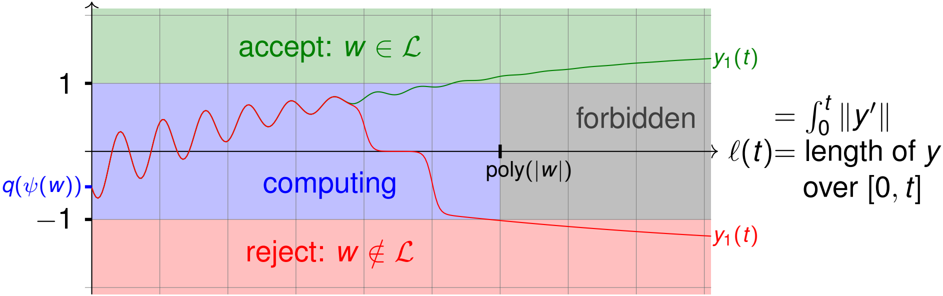
4. $\ell(t) \geqslant t$

Characterization of Turing polynomial time

Definition: $\mathcal{L} \subseteq \{0,1\}^*$ is polytime-recognizable iff for all w:

$$y(0) = q(\psi(w)) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

satisfies:



Theorem

$\mathcal{L} \in \mathrm{P}$ if and only if $\mathcal{L}$ is polytime-recognizable.

Characterization of real polynomial time

Definition: $f : [a, b] \to \mathbb{R}$ is analog-polytime iff for all x:

$$y(0) = q(x) \qquad y' = p(y)$$

satisfies:

Characterization of real polynomial time

Definition: $f : [a,b] \to \mathbb{R}$ is analog-polytime iff for all x:

$$y(0) = q(x) \qquad y' = p(y)$$

satisfies:

1. $\forall n \in \mathbb{N}$, if $\ell(t) \geqslant \text{poly}(\|x\|, n)$ then $|y_1(t) - f(x)| \leqslant 2^{-n}$

$$\text{where } \ell(t) = \int_0^t \|y'(u)\|\, du$$

«If curve is long enough, precision is good enough»

Characterization of real polynomial time

Definition: $f : [a, b] \to \mathbb{R}$ is analog-polytime iff for all x:

$$y(0) = q(x) \qquad y' = p(y)$$

satisfies:

1. $\forall n \in \mathbb{N}$, if $\ell(t) \geqslant \text{poly}(\|x\|, n)$ then $|y_1(t) - f(x)| \leqslant 2^{-n}$

   $$\text{where } \ell(t) = \int_0^t \|y'(u)\|\, du$$

   «If curve is long enough, precision is good enough»

2. $\forall t \in \mathbb{R}_+, \|y'(t)\| \geqslant 1$

   «Curve grows at least linearly with time»

Characterization of real polynomial time

Definition: $f : [a, b] \to \mathbb{R}$ is analog-polytime iff for all x:

$$y(0) = q(x) \qquad y' = p(y)$$

satisfies:

1. $\forall n \in \mathbb{N}$, if $\ell(t) \geqslant \text{poly}(\|x\|, n)$ then $|y_1(t) - f(x)| \leqslant 2^{-n}$

$$\text{where } \ell(t) = \int_0^t \|y'(u)\| \, du$$

«If curve is long enough, precision is good enough»

2. $\forall t \in \mathbb{R}_+, \|y'(t)\| \geqslant 1$

«Curve grows at least linearly with time»

Main result

$f : [a, b] \to \mathbb{R}$ is polytime computable iff f is analog-polytime.

Conclusion

- Time complexity for the GPAC: length or time+space
- Turing machines and GPACs are equivalent for time complexity
- Purely analog and machine-independent characterization of (discrete and real) polynomial time

Conclusion

- Time complexity for the GPAC: length or time+space
- Turing machines and GPACs are equivalent for time complexity
- Purely analog and machine-independent characterization of (discrete and real) polynomial time

Perspectives:

- Better understanding of time complexity
- Space complexity
- Nondeterminism
- Constants (a.k.a getting rid of π)
- Robustness of errors/perturbations

Bournez, O., Campagnolo, M. L., Graça, D. S., and Hainry, E. (2007).
Polynomial differential equations compute all real computable functions on computable compact intervals.
23(3):317–335.

Shannon, C. E. (1941).
Mathematical theory of the differential analyser.
Journal of Mathematics and Physics MIT, 20:337–354.