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)

PTIME = 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

VS
Let's model!

<table>
<thead>
<tr>
<th>Physical Computer</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop, ...</td>
<td>Turing machines</td>
</tr>
<tr>
<td></td>
<td>$\lambda$-calculus</td>
</tr>
<tr>
<td></td>
<td>Recursive functions</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Discrete dynamical systems</td>
</tr>
<tr>
<td>Differential Analyzer, ...</td>
<td>GPAC</td>
</tr>
<tr>
<td></td>
<td>Continuous dynamical systems</td>
</tr>
</tbody>
</table>
Let’s model!

<table>
<thead>
<tr>
<th>Physical Computer</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop, ...</td>
<td>Turing machines</td>
</tr>
<tr>
<td></td>
<td>$\lambda$-calculus</td>
</tr>
<tr>
<td></td>
<td>Recursive functions</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Discrete dynamical systems</td>
</tr>
<tr>
<td>Differential Analyzer, ...</td>
<td>GPAC</td>
</tr>
<tr>
<td></td>
<td>Continuous dynamical systems</td>
</tr>
</tbody>
</table>

**Church Thesis**

All *reasonable* models of computation are equivalent.
Let’s model!

<table>
<thead>
<tr>
<th>Physical Computer</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop, ...</td>
<td>Turing machines</td>
</tr>
<tr>
<td></td>
<td>$\lambda$-calculus</td>
</tr>
<tr>
<td></td>
<td>Recursive functions</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Discrete dynamical systems</td>
</tr>
<tr>
<td>Differential Analyzer, ...</td>
<td>GPAC</td>
</tr>
<tr>
<td></td>
<td>Continuous dynamical systems</td>
</tr>
</tbody>
</table>

**Church Thesis**

All *reasonable* models of computation are equivalent.

**Implicit corollary**

Some models are *too general/unreasonable.*
Let’s model!

<table>
<thead>
<tr>
<th>Physical Computer</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop, ...</td>
<td>Turing machines</td>
</tr>
<tr>
<td></td>
<td>$\lambda$-calculus</td>
</tr>
<tr>
<td></td>
<td>Recursive functions</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Discrete dynamical systems</td>
</tr>
<tr>
<td>Differential Analyzer, ...</td>
<td>GPAC $\rightarrow$ reasonable?</td>
</tr>
<tr>
<td></td>
<td>Continuous dynamical systems</td>
</tr>
</tbody>
</table>

**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:
  - Constant $k$
  - Adder $u + v$
  - Multiplier $uv$
  - Integrator $\int u$
Examples of GPAC

Exponential:

\[ y(t) \sim y = \int y \sim y(t) = \exp(t) \]
Examples of GPAC

Exponential:

\[
\int y(t) \quad \rightsquigarrow \quad y' = y \quad \rightsquigarrow \quad y(t) = \exp(t)
\]
Examples of GPAC

Exponential:
\[
\int y(t) \quad \sim \quad y' = y \quad \sim \quad y(t) = \exp(t)
\]

(Co)sine:
\[
\begin{align*}
-1 & \quad \times \quad \int \quad \int \\
& \quad \downarrow \quad \downarrow \\
& \quad y_1(t) \quad y_2(t)
\end{align*}
\]

\[
\begin{align*}
y_1' &= y_2 \\
y_2' &= -y_1
\end{align*} \quad \sim \quad \begin{align*}
y_1(t) &= \sin(t) \\
y_2(t) &= \cos(t)
\end{align*}
\]
Examples of GPAC

Rational function:

\[ \int y^2(t) - 2 \times \int y_1(t) \]

Theorem (Graça and Costa)

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

\[ \begin{cases} y_1' = -2y_2y_1^2 \\ y_2' = 1 \end{cases} \sim \begin{cases} y_1(t) = \frac{1}{1+t^2} \\ y_2(t) = t \end{cases} \]
Examples of GPAC

Rational function:

\[ y_2(t) \]

\[ \int \frac{-2}{y_2(y_2^2)} \times \int \times \int y_1(t) \]

\[ y_1'(t) = -2y_2y_2^2 \]
\[ y_2'(t) = 1 \]

\[ \{ y_1(t) = \frac{1}{1+t^2} \]
\[ y_2(t) = t \]

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

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

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

Shannon’s notion
Computing with the GPAC

Generable functions

\[
\begin{aligned}
\left\{ 
& y(0) = y_0 \\
& y'(x) = p(y(x))
\right. \\
& f(x) = y_1(x)
\end{aligned}
\]

Shannon's notion

\sin, \cos, \exp, \log, \ldots

Strictly weaker than Turing machines [Shannon, 1941]
Computing with the GPAC

Generable functions
\[
\begin{align*}
  y(0) &= y_0 \\
y'(x) &= p(y(x)) & x \in \mathbb{R}
\end{align*}
\]

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

Shannon’s notion
sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

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

Modern notion
Computing with the GPAC

Generable functions

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

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

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

Shannon’s notion

sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

Computable

\[
\begin{align*}
  y(0) &= q(x) \\
y'(t) &= p(y(t))
\end{align*}
\]

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

\[t \in \mathbb{R}_+\]

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

Modern notion

sin, cos, exp, log, \Gamma, \zeta, ...

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”

---

**Quantum computers**

**Believed different**

**Equivalent**

**Boolean circuits**

**Turing machines**

**Recursive functions**

**Unknown**

**GPACs**
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”

Quantum computers
Believed different
Boolean circuits
Turing machines
Equivalent
Recursive functions
Unknown

Main Result of the paper
Turing machines and GPACs are equivalent for complexity.
Time complexity for continuous systems

- Turing machines: $T(x) = \text{number of steps to compute on } x$
Time complexity for continuous systems

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

**Intuitive definition**

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

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

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

Observation: This definition is broken: all functions have arbitrarily small complexity.
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 $|y_1(t) - f(x)| \leq e^{-\mu}$

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

$z(t) = y(e^t)$
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 $|y_1(t) - f(x)| \leq e^{-\mu}$

\[
y(0) = q(x) \quad y' = p(y) \quad z(t) = y(e^t)
\]

\[
y_1(t) \quad g(x) \quad \tilde{g}(x) \quad w(t) = y(e^{e^t})
\]

Observation: This definition is broken: all functions have arbitrarily small complexity.
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 $|y_1(t) - f(x)| \leq e^{-\mu}$

$y(0) = q(x)$ \hspace{1cm} $y' = p(y)$ \hspace{1cm} $z(t) = y(e^t)$

$g(x)$ \hspace{1cm} $y_1(t)$ \hspace{1cm} $f(x)$ \hspace{1cm} $\sim$

$\sim$

$\sim$

$\hat{g}(x)$ \hspace{1cm} $\tilde{g}(x)$ \hspace{1cm} $\hat{y}_1(t)$ \hspace{1cm} $\hat{y}(x)$ \hspace{1cm} $\sim$

$\sim$

$\hat{g}(x)$ \hspace{1cm} $\tilde{g}(x)$ \hspace{1cm} $\hat{y}_1(t)$ \hspace{1cm} $\hat{y}(x)$ \hspace{1cm} $\sim$

$\sim$

$\sim$

Observation

This definition is broken: all functions have arbitrarily small complexity.
Time-space correlation of the GPAC

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

Observation

Time scaling costs “space.”

Time complexity for the GPAC must involve time and space!
Time-space correlation of the GPAC

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

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

Observation
Time scaling costs “space”.

Time complexity for the GPAC must involve time and space!
Time-space correlation of the GPAC

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

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

Observation

Time scaling costs “space”.

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

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

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

**Observation**

Time scaling costs "space".

Time complexity for the GPAC must involve time and space!
Two equivalent notions of complexity

\[
\begin{align*}
&\begin{cases}
y(0) = q(x) \\
y'(t) = p(y(t))
\end{cases} \\
f(x) = \lim_{t \to \infty} y_1(t)
\end{align*}
\]
Two equivalent notions of complexity

\[
\begin{align*}
\begin{cases}
y(0) = q(x) \\
y'(t) = p(y(t))
\end{cases}
\end{align*}
\]

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

Length based complexity: L

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

\[
L(x, \mu) = \text{length } \ell(t) \text{ so that } \| y_1(t) - f(x) \| \leq e^{-\mu}
\]
Definition: $\mathcal{L} \subseteq \{0, 1\}^*$ is polytime-recognizable iff for all $w$: 

\[
\text{length of } y \text{ over } [0, t] = \int_0^t \|y'\| \ell(t) = q(\psi(w)) - 1 \sum_{i=1}^{|w|} w_i^2 - i
\]
Characterization of Turing polynomial time

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

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

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

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

satisfies:

$$= \int_0^t \|y'\|$$

$$\ell(t) = \text{length of } y \text{ over } [0, t]$$
Characterization of Turing polynomial time

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

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

satisfies:

- accept: \( w \in \mathcal{L} \)
  - if \( y_1(t) \geq 1 \) then \( w \in \mathcal{L} \)

\[
\ell(t) = \int_0^t \|y'\|
\]

\( \ell(t) \) is length of \( y \) over \([0, t]\)
Characterization of Turing polynomial time

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

\[
y(0) = q(\psi(w)) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{\lvert w \rvert} w_i 2^{-i}
\]

satisfies:

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

if \( y_1(t) \leq -1 \) then \( w \notin \mathcal{L} \)

\[
\ell(t) = \int_0^t \|y'\|
\]

length of \( y \) over \([0, t]\)
Characterization of Turing polynomial time

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

\[
\begin{align*}
    y(0) &= q(\psi(w)) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{\|w\|} w_i 2^{-i} \\
    \text{satisfies:} \\
    &\text{accept: } w \in \mathcal{L} \\
    &\text{computing} \\
    &\text{reject: } w \notin \mathcal{L} \\
    &\text{if } \ell(t) \geq \text{poly}(\|w\|) \text{ then } |y_1(t)| \geq 1
\end{align*}
\]

\[
\ell(t) = \int_0^t \|y'\| \\
= \text{length of } y \text{ over } [0, t]
\]
Characterization of Turing polynomial time

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

\[
y(0) = q(\psi(w)) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{\lfloor w \rfloor} w_i 2^{-i}
\]

satisfies:

- accept: $w \in \mathcal{L}$
- computing
- reject: $w \notin \mathcal{L}$

\[
\ell(t) = \int_0^t \|y'\| \quad \text{length of } y \text{ over } [0, t]
\]

\[
\ell(t) \geq t
\]
Characterization of Turing polynomial time

**Definition:** $L \subseteq \{0, 1\}^*$ is polytime-recognizable iff for all $w$: 
$$y(0) = q(\psi(w)) \quad y' = p(y) \quad \psi(w) = \sum_{i=1}^{\mid w \mid} w_i2^{-i}$$

satisfies:

- **accept:** $w \in L$
- **forbidden**
- **computing**
- **reject:** $w \notin L$

$$\ell(t) = \int_0^t \|y'\|$$

$\ell(t)$ is polytime when $y$ is $L$:

- $\mathcal{L} \in P$ if and only if $\mathcal{L}$ is polytime-recognizable.
Characterization of real polynomial time

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

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

satisfies:

1. $\forall n \in \mathbb{N},$ if $\ell(t) \geq \text{poly}(\|x\|, n)$ then $|y_1(t) - f(x)| \leq 2^{-n}$ where $\ell(t) = \int_0^t \|y'(u)\| \, du$ «If curve is long enough, precision is good enough»

2. $\forall t \in \mathbb{R}^+,$ $\|y'(t)\| \geq 1$ «Curve grows at least linearly with time»

**Main result**

$f : [a, b] \rightarrow \mathbb{R}$ is polytime computable iff $f$ is analog-polytime.
Characterization of real polynomial time

**Definition:** $f : [a, b] \rightarrow \mathbb{R}$ is analog-polytime iff for all $x$:
\[ y(0) = q(x) \quad y' = p(y) \]
satisfies:
\[ \forall n \in \mathbb{N}, \text{ if } \ell(t) \geq \text{poly}(\|x\|, n) \text{ then } |y_1(t) - f(x)| \leq 2^{-n} \]

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

«If curve is long enough, precision is good enough»
Characterization of real polynomial time

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

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

satisfies:

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

   where \( \ell(t) = \int_0^t \|y'(u)\| \, du \)

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

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

   «Curve grows at least linearly with time»
Characterization of real polynomial time

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

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

satisfies:

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

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

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

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

   «Curve grows at least linearly with time»

**Main result**

$f : [a, b] \rightarrow \mathbb{R}$ 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 $\pi$)
- Robustness of errors/perturbations
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.  