Complexity-theoretic barriers to validated solution of initial value problems

Aki KAWAMURA
www.cs.toronto.edu/~kawamura/

Department of Computer Science
University of Toronto

May 23, 2008

The problem

The initial value problem:
given $g : [0,1] \times \mathbf{R} \to \mathbf{R}$, find $h : [0,1] \to \mathbf{R}$ such that

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big).$$

Question: How computationally complex is h?

To discuss computational complexity of real functions, we use the framework of Computable Analysis:

- computability (Grzegorczyk 1955);
- polynomial-time computability (Ko–Friedman 1981).
  (henceforth PTIME)

1. Complexity of real functions: How do we define (PTIME) computability of real functions?
2. Complexity of IVPs: How complex can h be when g is PTIME computable?
    - Lipschitz IVP is PSPACE complete
    - Analytic IVP is PTIME computable
3. Discussion

➤ 1. Complexity of real functions: How do we define (PTIME) computability of real functions?

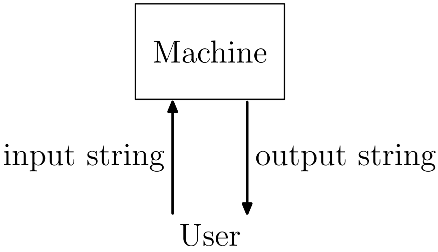
2. Complexity of IVPs: How complex can h be when g is PTIME computable?
   - Lipschitz IVP is PSPACE complete
   - Analytic IVP is PTIME computable

3. Discussion

The “digital” computer

Computers (= Turing machines) manipulate bits.

Objects (integers, graphs, . . .) must be encoded by bit strings.



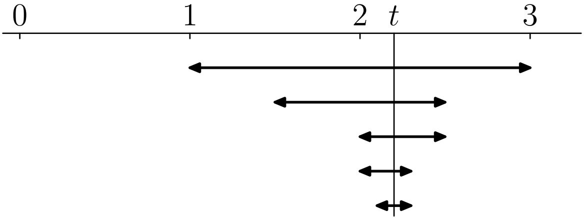
But real numbers cannot be named by bit strings, because there are uncountably many.

Representing real numbers

Represent real numbers by string-to-string functions.

Definition

Function $\varphi : \{0,1\}^* \to \{0,1\}^*$ represents $t \in \mathbf{R}$ if for each $n \in \mathbf{N}$, the value $\varphi(0^n)$ is (the binary notation of) either $\lfloor t \cdot 2^n \rfloor$ or $\lceil t \cdot 2^n \rceil$. We also say φ is a name of t.



$$\varphi(\varepsilon) = 10$$
$$\varphi(0) = 100$$
$$\varphi(00) = 101$$
$$\varphi(000) = 1001$$
$$\varphi(0000) = 10011$$
$$\vdots$$

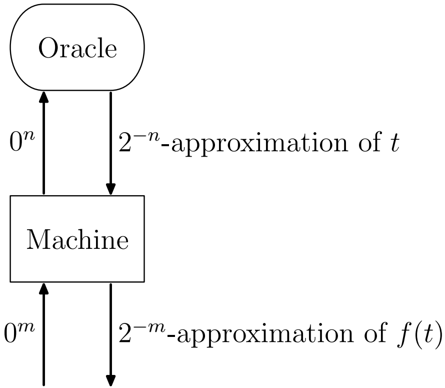
Thus, each $\varphi(0^n)$ gives a 2^{-n}-approximation of t.

Computing real functions

Use Oracle Turing Machines.

Definition (Grzegorczyk 1955)

A machine computes function $f : [0, 1] \to \mathbf{R}$ if, given any name of any $t \in [0, 1]$ as oracle, it computes some name of $f(t)$.

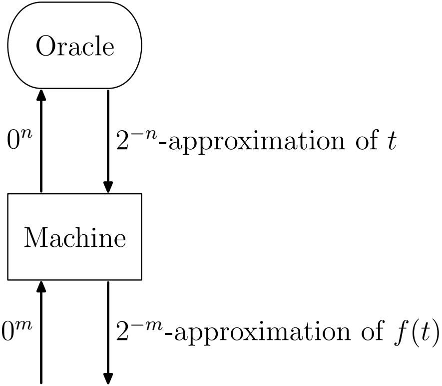


In other words:

- The machine Turing-reduces $f(t)$ to t.
- The machine produces approximations of $f(t)$ of any precision the user desires,
  assuming that the user supplies the machine with approximations of t of any precision it desires.

Computing real functions

Use Oracle Turing Machines.



Definition (Grzegorczyk 1955)

A machine computes function $f : [0, 1] \to \mathbf{R}$ if, given any name of any $t \in [0, 1]$ as oracle, it computes some name of $f(t)$.

PTIME means that the machine halts after polynomially many steps in m.

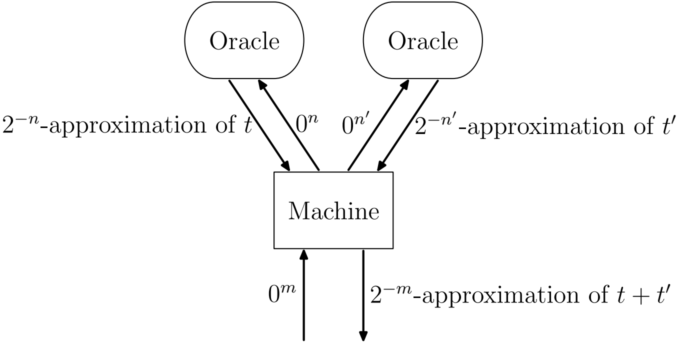
- A polynomial p and a PTIME algorithm $\hat{h}$ such that

$$u \in \{\lfloor t \cdot 2^{p(m)} \rfloor, \lceil t \cdot 2^{p(m)} \rceil\}$$
$$\implies \hat{h}(0^m, u) \in \{\lfloor h(t) \cdot 2^m \rfloor, \lceil h(t) \cdot 2^m \rceil\}.$$

Example 1

Addition $+ : [0,1] \times [0,1] \to \mathbf{R}$ is PTIME computable.

Given (functions representing) t, t' as oracles, compute a 2^{-m}-approximation of the sum $t + t'$.



Get 2^{-m-2}-approximations of t and t', compute their sum (a rational number), and output the closest 2^{-m}-approximation.

Example 2

The function $\exp : [0, 1] \to \mathbf{R}$ is PTIME computable.

To 2^{-m}-approximate

$$\exp t = \frac{1}{0!} + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots ,$$

it suffices to $2^{-m}/2$-approximate
the sum of the first $m + 2$ terms
(the remaining terms add up to $< 2^{-m}/2$).

Example 3

Comparison $\leq : [0, 1]^2 \rightarrow \mathbf{R}$ is not computable:

$$\leq(t, t') = \begin{cases} 1 & \text{if } t \leq t' \\ 0 & \text{if } t > t' \end{cases}$$

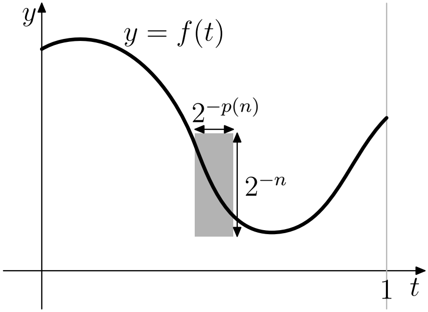
Computable functions are continuous

Theorem

- Every computable function is continuous.
- Every PTIME computable function has polynomial modulus of continuity.

Modulus of continuity p:

$$|t - t'| < 2^{-p(n)}$$
$$\Longrightarrow |f(t) - f(t')| < 2^{-n}$$



1. Complexity of real functions: How do we define (PTIME) computability of real functions?

➤ 2. Complexity of IVPs: How complex can *h* be when *g* is PTIME computable?
    - Lipschitz IVP is PSPACE complete
    - Analytic IVP is PTIME computable

3. Discussion

Summary

Let $g : [0,1] \times [-1,1] \to \mathbf{R}$ and $h : [0,1] \to \mathbf{R}$ satisfy

$$h(0) = 0, \qquad h'(t) = g\bigl(t, h(t)\bigr).$$

and suppose that g is PTIME computable. Then

- there may be infinitely many solutions h, none of which is computable (Pour-El 1979, Ko 1983);
- if we simply assume that h is the unique solution, then it is computable but can take arbitrarily long time (Miller 1970, Ko 1983);
- if we further assume that g is Lipschitz continuous, then the (unique) solution h is PSPACE computable and can be PSPACE complete;
- if we further assume that g is analytic, then the (unique) solution h is PTIME computable.

Summary

Let $g : [0,1] \times [-1,1] \to \mathbf{R}$ and $h : [0,1] \to \mathbf{R}$ satisfy

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big).$$

and suppose that g is PTIME computable. Then

- there may be infinitely many solutions h, none of which is computable (Pour-El 1979, Ko 1983);
- if we simply assume that h is the unique solution, then it is computable but can take arbitrarily long time (Miller 1970, Ko 1983);
- if we further assume that g is Lipschitz continuous, then the (unique) solution h is PSPACE computable and can be PSPACE complete;
- if we further assume that g is analytic, then the (unique) solution h is PTIME computable.

red: new results

1. Complexity of real functions: How do we define (PTIME) computability of real functions?
2. Complexity of IVPs: How complex can *h* be when *g* is PTIME computable?
   ➤ ▸ Lipschitz IVP is PSPACE complete
   - Analytic IVP is PTIME computable
3. Discussion

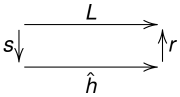
PSPACE completeness

Theorem

There is a PTIME computable, Lipschitz continuous g such that the solution h is PSPACE complete.

I.e., h is the "hardest" among PSPACE solvable problems:

> for any PSPACE predicate L, there are a polynomial p and PTIME computable functions r, s such that if $s(u) \in \{\lfloor t \cdot 2^{p(n)} \rfloor, \lceil t \cdot 2^{p(n)} \rceil\}$ and $v \in \{\lfloor h(t) \cdot 2^n \rfloor, \lceil h(t) \cdot 2^n \rceil\}$, then $r(v) = L(u)$.



Building blocks

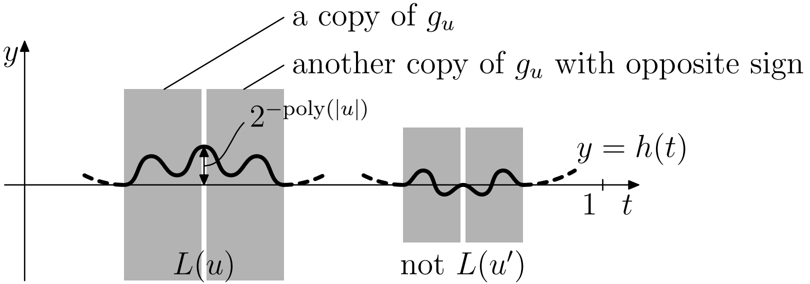
It suffices to construct $g_u : [0,1] \times [-1,1] \to \mathbf{R}$ and $h_u : [0,1] \to \mathbf{R}$ for each string u (with g_u uniformly Lipschitz and PTIME computable from u) such that

$$h_u(0) = 0, \qquad h_u'(t) = g_u\bigl(t, h_u(t)\bigr)$$

and

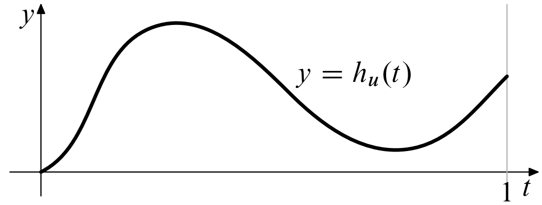
$$h_u(1) = \begin{cases} 2^{-\mathrm{poly}(|u|)} & \text{if } L(u), \\ 0 & \text{otherwise.} \end{cases}$$

Once this is done, g_u's can be put into one function g:



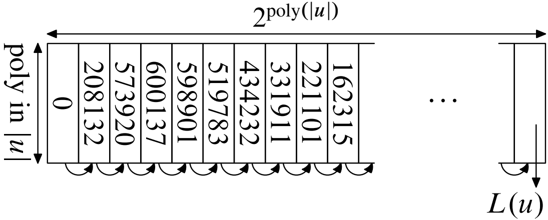
An attempt: discrete IVP

We want h_u such that:



- its slope is PTIME computed using the current value;
- $h_u(1)$ encodes the answer $L(u)$.

Every PSPACE computation has the following description:

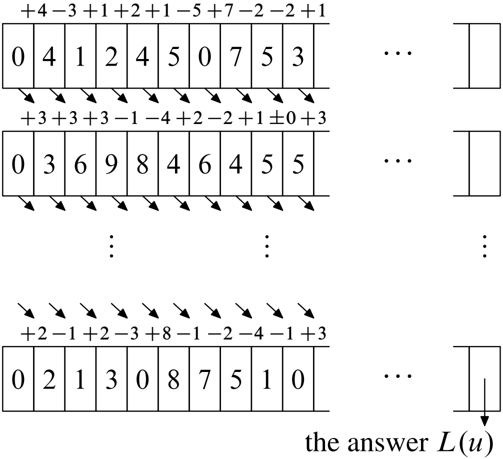


- each cell is PTIME computed from the previous cell;
- the last cell has the answer $L(u)$.

An attempt: use values of h_u to encode this table.
But this does not give Lipschitz continuous g.

Discrete Lipschitz IVP

A discrete problem simulable by Lipschitz IVP:



Each increment is PTIME computed from (u and) the upper left number.

By a more elaborate reduction, any PSPACE predicate can be simulated by this table also.

1. Complexity of real functions: How do we define (PTIME) computability of real functions?
2. Complexity of IVPs: How complex can h be when g is PTIME computable?
   - Lipschitz IVP is PSPACE complete
   - ➤ Analytic IVP is PTIME computable
3. Discussion

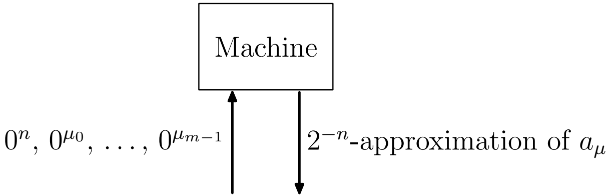
PTIME computable Taylor coefficients

Lemma

Let f be a real analytic function on
a compact subset of $\mathbf{R}^m$ around the origin:

$$f(x) = \sum_{\mu \in \mathbf{N}^m} a_\mu x_0^{\mu_0} \cdots x_{m-1}^{\mu_{m-1}}.$$

Then f is PTIME computable if and only if
the sequence $(a_\mu)_{\mu \in \mathbf{N}^m}$ is.



Solution by Taylor series

Substitute

$$g(t, y) = \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} a_{ij} t^i y^j, \qquad h(t) = \sum_{i \in \mathbf{N}} b_i t^i$$

into $h'(t) = g\big(t, h(t)\big)$ and compare coefficients.

Theorem

If $(a_{ij})_{(i,j) \in \mathbf{N}^2}$ is PTIME computable, so is $(b_i)_{i \in \mathbf{N}}$.

Corollary

If an analytic function g is PTIME computable, so is h.

1. Complexity of real functions: How do we define (PTIME) computability of real functions?
2. Complexity of IVPs: How complex can h be when g is PTIME computable?
   - Lipschitz IVP is PSPACE complete
   - Analytic IVP is PTIME computable
3. Discussion

Algorithms and complexity

- The Church–Turing Thesis:
  - effective
    = computed by a Turing machine
  - feasible
    = computed by a Turing machine in polynomial time
- In discrete problems, hardness results sometimes give useful information for algorithm design.
- What about numerical problems?

Results (again)

Let $g : [0,1] \times [-1,1] \to \mathbf{R}$ and $h : [0,1] \to \mathbf{R}$ satisfy

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big).$$

and suppose that g is PTIME computable. Then

- there may be infinitely many solutions h, none of which is computable (Pour-El 1979, Ko 1983);
- if we simply assume that h is the unique solution, then it is computable but can take arbitrarily long time (Miller 1970, Ko 1983);
- if we further assume that g is Lipschitz continuous, then the (unique) solution h is PSPACE computable and can be PSPACE complete;
- if we further assume that g is analytic, then the (unique) solution h is PTIME computable.

What do the positive results imply?

- unique solution $\Longrightarrow$ computable
- Lipschitz $\Longrightarrow$ PSPACE computable
- analytic $\Longrightarrow$ PTIME computable

What do these mean for numerical solution of IVPs?
—Not much: Just because easy g gives rise to easy h does not mean that it is easy to compute $g \mapsto h$.

The first result (computability) can be strengthened nicely to a computability result of the operator $g \mapsto h$.

Open question:
How should we formulate PTIME computable operators?

What do the negative results imply?

- non-computable solution
- arbitrarily long time, even if unique
- PSPACE-complete, even if Lipschitz

Though we have no definition of "easy operators", they certainly should not take easy functions to hard ones.

The last result (say) implies: Unless PTIME = PSPACE, any efficient algorithm for Lipschitz IVP must fail on some g (and in particular on the one we constructed).

Complexity-theoretic barriers to validated solution of initial value problems

Aki KAWAMURA

www.cs.toronto.edu/~kawamura/

Department of Computer Science
University of Toronto

May 23, 2008