Time Polynomial in Input or Output

Yuri Gurevich; Saharon Shelah

The Journal of Symbolic Logic, Vol. 54, No. 3 (Sep., 1989), 1083-1088.

Stable URL:
http://links.jstor.org/sici?sici=0022-4812%28198909%2954%3A3%3C1083%3ATPIIOO%3E2.0.CO%3B2-9

The Journal of Symbolic Logic is currently published by Association for Symbolic Logic.

THE JOURNAL OF SYMBOLIC LOGIC
Volume 54, Number 3, Sept. 1989

TIME POLYNOMIAL IN INPUT OR OUTPUT

YURI GUREVICH AND SAHARON SHELAH

Abstract. We introduce the class PIO of functions computable in time that is polynomial in max{the length of input, the length of output}, observe that there is no notation system for total PIO functions but there are notation systems for partial PIO functions, and give an algebra of partial PIO functions from binary strings to binary strings.

§1. Introduction. Bob Paige brought to our attention computability in time linear in max{the length of input, the length of output} [Pa], [CP]. He argued that it may be unreasonable to measure computational complexity in terms of input only; a very simple algorithm may spend a long time printing out the output.

Unfortunately, the notion of linear time greatly depends on the model of computation. (In this connection, we have introduced nearly linear time [GS].) On the other hand, the notion of polynomial time is very robust. Let Σ, Σ_1, Σ_2 be alphabets, and Σ^*, Σ_1^*, Σ_2^*, respectively, be the sets of strings in them.

DEFINITION. A partial function f from Σ_1^* to Σ_2^* is PIO (or *computable in time polynomial in input or output*) if there exist a Turing machine M and a polynomial p such that:

(1) given any x in the domain of f, M outputs $f(x)$ within time $\leq p(\max\{|x|, |fx|\})$, and

(2) M does not halt on any input outside the domain of f.

It does not matter whether the witnessing Turing machine has one or many tapes, whether it has a special input tape, whether it has a special output tape, whether a random access to input is allowed, and so on. To a great extent, the notion of PIO functions is machine-independent.

DEFINITION. In this paper, a function $f\colon \Sigma_1^* \to \Sigma_2^*$ will be called *honest* if there is a polynomial q such that for every x in the domain of f, $q(|fx|) \geq |x|$.

In §3, we show that there is no notation system for total PIO functions but there are notation systems for partial PIO functions. An alternative machine-independent definition for the class of partial PIO functions from binary strings to binary strings

Received January 2, 1988; revised August 30, 1988.

The work of the first author was partially supported by NSF grant DCR 85-03275. The work of both authors was partially supported by a grant of the US-Israel Binational Science Foundation. In the main, the work was done during a week in Fall 1985 when both authors visited Rutgers University.

0022-4812/89/5403-0035/$01.60

is given in §5; it is shown there that partial PIO functions form the closure of some simple initial PIO functions under some natural operations. A similar algebra of honest partial PIO functions is given in §4.

One may want to study various analogs of PIO. The two theorems of §3 survive many generalizations, but creating a reasonable algebra of partial functions is a separate problem in each case (when there are notation systems for partial functions). Here are few possible candidates for study:

(1) Functions computable in time that is polynomial in the length of output (PO functions). There are two very different classes of PO functions, depending on whether random access to input is allowed. Notice that in either case, honest PO functions are exactly honest PIO functions.

(2) Functions computable in time linear in max{the length of input, the length of output} (LIO functions); and functions computable in time linear in the length of output (LO functions). One can identify some reasonable computational models and study the proposed classes. In connection with LIO, see [CP].

§2. Composition of PIO functions. Even though it does not matter what kind of Turing machine is used to define PIO functions, one kind is especially convenient.

DEFINITION. In this paper, an *off-line Turing machine* is a Turing machine with a read-only input tape, one or several work tapes, and a write-only output tape.

A separate output tape guarantees a legal output whenever the machine halts. Every off-line Turing machine M with input alphabet Σ_1 and output alphabet Σ_2 computes some (not necessarily total) function from Σ_1^* to Σ_2^*.

DEFINITION. Suppose that an off-line Turing machine M computes a partial function f. M is PIO if there is a polynomial p such that, for every x in the domain of f, M spends at most $p(\max\{|x|, |fx|\})$ steps for computing $f(x)$.

Obviously, the function computed by any PIO off-line Turing machine is PIO, and every PIO function is computed by some PIO off-line Turing machine. We use off-line Turing machines to define versions of the classes PO, LIO and LO mentioned in §1. For the sake of consistency, we give the name PI to the class of partial functions computable in time polynomial in the length of the input.

DEFINITION. Suppose that an off-line Turing machine M computes a function f.

(1) M is PI (resp. PO) if there is a polynomial p such that, for every x in the domain of f, M spends at most $p(|x|)$ (resp. $p(|fx|)$) steps for computing $f(x)$.

(2) M is LIO (resp. LO) if there is a linear polynomial p such that, for every x in the domain of f, M spends at most $p(\max\{|x|, |fx|\})$ (resp. $p(|fx|)$) steps for computing $f(x)$.

DEFINITION. A partial function f from some Σ_1^* to some Σ_2^* is PI (resp. PO, LIO, LO) if there is a PI (resp. PO, LIO, LO) off-line Turing machine that computes f.

The empty string will be denoted e.

LEMMA **2.1.** *The composition of two PIO functions may be not PIO. Moreover, let f be any function from some Σ_1^* to some Σ_2, and let* FirstLetter$(y) =$ [*if $y = e$ then e, else the first letter of y*] *for every y in Σ_2^*. Then there is an LO function g such that $f =$ FirstLetter $\circ\, g$.*

PROOF. Let M be a Turing machine that computes f, and let $m(x)$ be the number of steps of M on inputs x. (If $f(x)$ is undefined then $m(x)$ is undefined.) The desired $g(x) = f(x)0^{m(x)}$. Q.E.D.

REMARK. The lemma reflects an observation conveyed to us by Bob Paige [Pa].

LEMMA. **2.2** *Suppose that f and g are PIO functions. Then the composition $(g \circ f)(x) = g(f(x))$ of f and g is computable (whenever it is defined) within time bounded by a polynomial of $|x|, |fx|$ and $|g(fx)|$. Hence $g \circ f$ is PIO if $|fx|$ is bounded by a polynomial of $|x|$ and $|g(fx)|$ (whenever $|g(fx)|$ is defined). In particular, $g \circ f$ is PIO if f is PI or g is honest PO.*

PROOF. Let M_1 and M_2 be off-line Turing machines that compute f and g respectively, and let M_3 be an off-line Turing machine obtained from M_1 and M_2 by identifying the output tape of M_1 with the input tape of M_2. M_3 starts by simulating M_1; if and when the output y of M_1 is computed, M_3 resets the head on the output tape of M_1 and then simulates M_2 on y. Obviously, M_3 computes $(g \circ f)(x)$ within time bounded by a polynomial of $|x|, |fx|$ and $|g(fx)|$. Q.E.D.

§3. Notation systems.

DEFINITION. Let K be a collection of partial computable functions from some Σ_1^* to some Σ_2^*. A *notation system* for K is a triple (Σ, L, F) where L is a recursive subset of Σ^* and F is a recursive function that associates every L-string x with a Turing machine M_x in such a way that:

(1) every M_x computes a K-function, and

(2) every K-function is computed by some M_x.

REMARK (triggered by a question of Andreas Blass). In the case when K is defined by a class W of Turing machines, it is natural to strengthen (1) by requiring that each M_x belongs to W; for example, if K is the class of PIO functions, one may want to require that each M_x is a PIO machine. One may also want to put some complexity restrictions on L and F. We stick to our liberal definition because our main goal in this section is to prove the following negative result.

THEOREM **3.1.** *Let K be a class of total functions from some Σ_1^* to some Σ_2^* which contains all total LO functions from Σ_1^* to Σ_2^*. There is no notation system for K. In particular, there is no notation system for the class of total PIO functions from Σ_1^* to Σ_2^*, and there is no notation system for the class of honest total PIO functions from Σ_1^* to Σ_2^*.*

PROOF. For a contradiction, let (Σ, L, F) be a notation system for K. Without loss of generality, $\Sigma = \Sigma_1$ and $L = \Sigma_1^*$.

Construct a Turing machine U that, given a Σ_1-string x, executes the following algorithm.

(1) Construct the machine $M_x = F(x)$, counting the number of steps in unary notation on the output tape.

(2) Simulate M_x on x until some result y is obtained, counting the number of steps in unary notation on the output tape.

(3) Print another character on the output tape.

(4) Print y on the output tape and halt.

The machine U is LO and computes some total function g from Σ_1^* to Σ_2^*. Hence some M_x computes g. Let y be the output of M_x on x.

On one hand, $g(x) = y$ because M_x computes g. On the other hand, $g(x)$ is a nonempty (because of (3), among other reasons) string followed by y because of the way U computes g. This is a contradiction. Q.E.D.

THEOREM **3.2.** *For all Σ_1 and Σ_2,*

(1) *there is a notation system for the class of all partial* PIO *functions from Σ_1^* to Σ_2^*, and*

(2) *there is a notation system for the class of all honest partial* PIO *functions from* Σ_1^* *to* Σ_2^*

PROOF. For i equal to 1 or 3, let K_i be the class of off-line Turing machines with input alphabet Σ_1, output alphabet Σ_2, and i work tapes with alphabet $\Sigma_1 \cup \Sigma_2 \cup \{\text{blank}\}$. Let $\Sigma = \{0, 1\}$.

(1) Let L comprise the binary codes for pairs (M, p) where M belongs to K_1 and p is a polynomial with nonnegative integer coefficients. The desired recursive function F transforms the code for (M, p) into a K_3-machine M_p which simulates M on one work tape, counts the steps of M on another, and counts the length of output on the third work tape. If and when M computes an output y on the given input x after some number m of steps, M_p checks whether $p(\max\{|x|, |y|\}) \geq m$. If yes, M_p outputs y and halts; otherwise it enters an infinite loop.

It is easy to see that every M_p computes a partial PIO function from Σ_1^* to Σ_2^*, and every partial PIO function f from Σ_1^* to Σ_2^* is computable by some M_p.

(2) The case of honest functions is similar, but the K_3-machine checks also that some polynomial q of $|y|$ exceeds $|x|$. Q.E.D.

COROLLARY. *There are notation systems for partial* PO, LIO, *and* LO *functions from* Σ_1^* *to* Σ_2^*.

PROOF. Similar to that of Theorem 3.2. Q.E.D.

§4. An algebra of honest PO functions. In the rest of this paper, a function is a partial function from $\{0, 1\}^*$ to $\{0, 1\}^*$. Let u, v, w, x, y and z be binary strings. Recall that the empty string is denoted e.

DEFINITION of the *replace-and-extend* function $y = \mathrm{RE}_{u,v,w}(x)$. If u is not a substring of x then $y = x$, and if $x = x_1 u x_2$, where the shown occurrence of u is the leftmost occurrence of u in x, then $y = x_1 v x_2 w$.

The functions $\mathrm{RE}_{e,v,e}$, $\mathrm{RE}_{e,e,w}$ and $\mathrm{RE}_{u,v,e}$ will be called $\mathrm{AddPrefix}_v$, $\mathrm{AddSuffix}_w$ and $\mathrm{Replace}_{u,v}$ respectively.

DEFINITION. The function $y = \mathrm{Truncate}_u(x)$ is given by the following program:

$y := x$;
while u is a suffix of y and $(|y| - |u|)^2 \geq |x|$ do
    $y :=$ [the unique z such that $zu = y$].

REMARK. The condition $(|y| - |u|)^2 \geq |x|$ is somewhat arbitrary. It ensures that for any u, $\mathrm{Truncate}_u$ is an honest PO function and compositions of $\mathrm{Truncate}_u$ allow us to remove polynomially-long tails of u's.

DEFINITION of the *conditional removal* function $\mathrm{CR}_u(x)$. If u is a suffix of x then remove the last letter of x, else do nothing.

DEFINITION. The *upper iteration* of a function f is a function $y = f^*(x)$ given by the following program:

while $|f(x)| > |x|$ do $x := f(x)$;
$y := x$.

LEMMA **4.1.** (1) *All functions* $\mathrm{RE}_{u,v,w}$, $\mathrm{Truncate}_u$ *and* CR_u *are honest and PO.*

(2) *The composition* $g \circ f$ *of a PIO function* f *and an honest PO function* g *is an honest PO function.*

(3) *The upper iteration* f^* *of any PIO function* f *is an honest PO function.*

PROOF. (1) is obvious. (2) follows from Lemma 2.2. To prove (3), let f be a PIO

function. Since $|f^*(x)| \geq |x|$ for all x, it suffices to prove that f^* is PO. There is a polynomial $p(i,j)$ such that p is monotone in both arguments and $f(x)$ is computable from x within time $p(|x|,|fx|)$. Let $x_0 = x$, each $x_{i+1} = f(x_i)$, and $m = \min\{i: |x_{i+1}| \leq |x_i|\}$, so that $f^*(x) = x_m$. The obvious computation of $f^*(x)$ requires time bounded by

$$\sum_{i \leq m} p(|x_i|,|x_{i+1}|) \leq \sum_{i \leq m} p(|x_m|,|x_m|) \leq p(|x_m|,|x_m|) \cdot (|x_m| + 1). \qquad \text{Q.E.D.}$$

THEOREM 4.1. *The class of honest PO functions is the closure of functions* $\mathrm{RE}_{u,v,w}$, $\mathrm{Truncate}_u$ *and* CR_u *by means of composition and upper iteration.*

PROOF. By virtue of Lemma 4.1, it suffices to prove only that every honest PO function f is constructible from functions $\mathrm{RE}_{u,v,w}$, $\mathrm{Truncate}_u$ and CR_u by means of composition and upper iteration. There exists a one-tape Turing machine M such that, for every input x,

(1) if x belongs to the domain of f then M computes $f(x)$ within time bounded by a polynomial in $|fx|$, and

(2) if f is undefined at x then M does not halt.

Without loss of generality, we may suppose that M satisfies the following conditions. Initially the tape consists of the given input x followed by one blank (which is different from either 0 or 1), and the head is in the leftmost cell. At each step (until M halts), the tape acquires another blank on the right. In the halting configuration, the tape is $f(x)$ followed by blanks and the head is in the leftmost blank cell.

It is supposed that state symbols differ from tape symbols. Let Σ be the set of all state and tape symbols of M (including the blank). Let $l = \lceil \log|\Sigma| \rceil$ and assign different binary strings of length l to Σ- symbols. If σ is assigned $b_1 \cdots b_l$, let $C(\sigma) = 11b_1 0 b_2 \cdots 0 b_l 0$.

If the tape of M is $a_1 \cdots a_k$, the current state symbol is q and the head is in the ith cell, then the string

$$C(a_1) \cdots C(a_{i-1}) C(q) C(a_i) C(a_{i+1}) \cdots C(a_k)$$

(with obvious modifications in case $i \leq 2$ or $i \geq k - 1$) will be called the *binary instantaneous description* (BID) of M.

Let α, β, B and I be the codes for 0, 1, the blank and the initial state respectively, and let IBID be the function

$$\mathrm{AddPrefix}_I \circ (\mathrm{Replace}_{B1,\beta B} \circ \mathrm{Replace}_{B0,\alpha B})^* \circ \mathrm{AddPrefix}_B,$$

so that IBID(x) is the initial BID of M on input x.

Given the transition table of M, one can construct a composition Step of RE functions such that if z is a nonhalting BID of M then Step(z) is the next BID of M, and if z is a halting BID then Step(z) = z. Let FBID = Step* $\circ$ IBID. If $f(x)$ is undefined then FBID(x) is undefined, else FBID(x) is the final BID of M on x. Let x range over the domain of f, and $y = f(x)$.

Let H be the code for the halt state and

$$g = \mathrm{RE}_{H,e,B} \circ [\mathrm{RE}_{\beta H,H1,B} \circ \mathrm{RE}_{\alpha H,H0,B}]^* \circ \mathrm{FBID}.$$

Then $g(x)$ equals y followed by some positive number of B's bounded by a polynomial in $|y|$. Recall that f is honest. Hence there is a number k, depending on f but not on x, such that if h is the composition $h = \text{Truncate}_B \circ \cdots \circ \text{Truncate}_B \circ g$ of g and k copies of Truncate_B then $h(x)$ equals either y or yB. Finally, f is the composition of h and a fixed number of CR functions. Q.E.D.

REMARK. We could define CR_u in such a way that $\text{CR}_u(zu) = z$ for all z. That would simplify the last step in the proof of Theorem 4.1 and make more complicated the proof of Theorem 5.1.

§5. An algebra of partial PIO functions.

DEFINITION of the function $y = \text{CutTail}(x)$:

> While 1 is a suffix of x do $x :=$ [the unique x_0 with $x = x_0 1$];
> if 0 is a suffix of x then $y :=$ [the unique x_1 with $x = x_1 0$].

LEMMA **5.1.** *Every PIO function f is the composition of some honest PO function g and the* CutTail *function.*

PROOF. The desired $g(x) = y01^m$, where $y = f(x)$ and m is the minimal natural number such that $|y01^m| \geq |x|$. Q.E.D.

Unfortunately, PIO is not closed under composition; see Lemma 2.1.

DEFINITION of the *upper composition* $g \uparrow f$ of functions f and g. Given x, compute $y = f(x)$. If and when y is computed, compute $z = g(y)$. If and when z is computed, check whether $(1 + |x| + |z|^2) \geq |y|$. If yes, output z; otherwise output y.

THEOREM 5.1. *PIO is the closure of functions* $\text{RE}_{u,v,w}$, Truncate_u, CR_u *and* CutTail *under upper composition and upper iteration.*

PROOF. CutTail is PI and therefore PIO. Use Lemmas 2.2 and 4.1 to check that PIO contains all functions $\text{RE}_{u,v,w}$, Truncate_u and CR_u and is closed under upper composition and upper iteration. It remains to notice that the upper composition can replace the ordinary composition in the proofs of Theorem 4.1 and Lemma 5.1. (The upper composition is not associative, but we may suppose that the proof of Theorem 4.1 uses the right associative notation: $f_3 \circ f_2 \circ f_1$ abbreviates $f_3 \circ (f_2 \circ f_1)$.) Q.E.D.

REFERENCES

[CP] JIAZHEN CAI and ROBERT PAIGE, *Binding performance at language design time*, ***Conference record of the fourteenth annual ACM symposium on principles of programming languages***, 1987, pp. 80–87.

[GS] YURI GUREVICH and SAHARON SHELAH, *Nearly linear time*, Lecture Notes in Computer Science, Springer-Verlag (to appear).

[Pa] ROBERT PAIGE, Private communication, Fall 1985.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN 48109

INSTITUTE OF MATHEMATICS
THE HEBREW UNIVERSITY
JERUSALEM, ISRAEL

DEPARTMENT OF MATHEMATICS
RUTGERS UNIVERSITY
NEW BRUNSWICK, NEW JERSEY 08903