LOWER BOUNDS FOR COPRIMENESS AND OTHER DECISION PROBLEMS IN ARITHMETIC

YIANNIS N. MOSCHOVAKIS

This talk was about some joint work with Lou van den Dries, in which we try to derive lower bounds for the worst-case, time complexity of functions and decision problems in arithmetic which apply to *all*—or, in any case, to as many as possible—algorithms. The relevant papers are listed in the bibliography and [3] gives a brief account of how we came to these questions, as well as a fairly complete exposition of what we have proved. Here I will confine myself to very few precise statements of specific results, since my main aim is to describe the methods that we use.[1]

1. One conjecture and two results

The ancient Euclidean algorithm computes the greatest common divisor of two natural numbers using iterated division. It can be expressed succinctly by the recursive equation

(1) $\quad \mathrm{gcd}(a,b) = \text{ if } (\mathrm{rem}(a,b)=0) \text{ then } b \text{ else } \mathrm{gcd}(b,\mathrm{rem}(a,b)) \quad (a \geq b \geq 1),$

and its natural complexity measure is

$$c_\varepsilon(a,b) = \text{the number of divisions required to compute } \mathrm{gcd}(m,n) \text{ using } (1).$$

It is easy to check that

$$c_\varepsilon(a,b) \leq 2\log_2(a) \quad (a \geq b > 1),$$

which is not the best known upper bound for the Euclidean, but is good enough for our purposes. It leads to the following, natural formulation of the Euclidean's optimality:

Main Conjecture. *For every algorithm α which computes the* gcd *from the remainder operation* rem, *there is a real constant $r > 0$ such that for infinitely many pairs (a,b) with $a > b > 1$,*

$$c_\alpha(a,b) \geq r\log_2(a),$$

where $c_\alpha(a,b)$ counts the number of calls to the remainder function required to compute $\mathrm{gcd}(a,b)$ using the algorithm α.

Extended abstract of a lecture given on October 25, 2005.

[1] I use standard notation and terminology, with $\mathbb{N} = \{0,1,\ldots\}$ the set of natural numbers, $\mathbb{Z} = \{\ldots,-1,0,1,\ldots\}$, $\mathrm{gcd}(m,n) =$ the largest k such that $k \mid m$ and $k \mid n$, and the following perhaps less common notations:

$$\text{if } m \geq n,\ m = nq + r \text{ and } 0 \leq r < n, \text{ then } \mathrm{iq}(m,n) = q,\ \mathrm{rem}(m,n) = r,$$

$$m \perp n \iff m,n \geq 1 \text{ and } \mathrm{gcd}(m,n) = 1 \quad (m \text{ and } n \text{ are coprime}),$$

$$m \dot{-} n = \text{ if } (m \leq n) \text{ then } 0 \text{ else } m - n, \qquad \log_2 m = x \iff 2^x = m \quad (m \geq 1).$$

The conjecture is vague, of course, absent a precise definition of "algorithm", but it is a useful template for the formulation of precise conjectures based on specific, precise assumptions about the behavior of algorithms. It makes clear, right at the start, that we are concerned with (relative) *algorithms from* specified *primitives* (the "givens"). With this understanding, we can formulate now, vaguely, the two basic results that I wish to discuss, and which I will make precise in the sequel:

Theorem A (van den Dries, [3])**.** *For every algorithm α which computes the greatest common divisor from the functions and relations*

$$+,\ \dot{-},\ <,\ =,\ \mathrm{iq},\ \mathrm{rem},\ \cdot,$$

there are infinitely many pairs (a, b) such that $a > b > 1$ and

$$c_\alpha(a + 1, b) \geq \frac{1}{4}\sqrt{\log_2 \log_2 b};$$

in fact the inequality holds whenever a, b are positive solutions of Pell's equation

$$a^2 = 1 + 2b^2.$$

I will argue that this result requires only a minimal, natural assumption about how algorithms operate, and so it truly holds for all algorithms (from the specified givens). The second result, however, depends on somewhat stronger assumptions about algorithms which cannot be considered obvious, and so I will label it "vague" in this first formulation:

Theorem $\mathbf{B}_1$ (vague). *For every algorithm α which decides the coprimeness relation $\perp$ from the functions and relations*

$$+,\ \dot{-},\ <,\ =,\ \mathrm{iq},\ \mathrm{rem},$$

there are infinitely many pairs (a, b) such that $a > b > 1$ and

$$c_\alpha(a, b) > \frac{1}{10}\log_2 \log_2 a;$$

in fact the inequality holds for all $a > b > 1$ such that

$$a \perp b \text{ and } \left|\frac{a}{b} - \sqrt{2}\right| < \frac{1}{b^2}$$

(including all solutions (a, b) of Pell's equation).

Both of these results assume many more primitives than the bare integer remainder function assumed by the Euclidean—including the very powerful multiplication function in the case of Theorem A—and in that sense they are stronger than they need to be to establish the Main Conjecture; on the other hand, the claimed lower bounds are one log below what we would like them to be, and in this subject one log is infinitely far away.

2. The complexity of values

A *signature* is a finite set $\Phi = \{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k\}$ of function symbols, each of a specified *arity* n_i, and a (partial) Φ-*algebra* is a structure

$$\mathbf{A} = (A, 0, 1, \phi_1, \ldots, \phi_k),$$

where $0, 1 \in A$, $0 \neq 1$, and each $\phi_i : A^{n_i} \rightharpoonup A$ is a partial function on A.[2]

Classical examples include the familiar structure of arithmetic $(\mathbb{N}, 0, 1, +, \cdot)$, but also the algebra of the Euclidean $(\mathbb{N}, 0, 1, \mathrm{rem})$, whose only given is the remainder operation. These, of course, are total. Algebras with genuinely partial givens arise most naturally as restrictions,

$$\mathbf{A} \restriction B = (B, 0, 1, \phi_1 \restriction B, \dots, \phi_k \restriction B) \qquad (\{0, 1,\} \subseteq B \subseteq A),$$

where, for any $\phi : A^n \rightharpoonup A$ and any $B \subseteq A$,

$$(\phi \restriction B)(x_1, \dots, x_n) = w \iff x_1, \dots, x_n, w \in B \;\&\; \phi(x_1, \dots, x_n) = w.$$

Especially interesting is the subalgebra generated by a sequence $\vec{a} = (a_1, \dots, a_n)$ of points in A, which is naturally ramified by depth: we set recursively,

$$\begin{aligned} G_0(\vec{a}) &= \{0, 1, a_1, \dots, a_n\}, \\ G_{m+1}(\vec{a}) &= G_m(\vec{a}) \cup \{\phi_i(x_1, \dots, x_l) \mid x_1, \dots, x_l \in G_m(\vec{a}), i = 1, \dots, k\}, \\ \mathbf{G}_m(\vec{a}) &= \mathbf{A} \restriction G_m(\vec{a}), \end{aligned}$$

and then $G_\infty(\vec{a}) = \cup_m G_m(\vec{a})$, $\mathbf{G}_\infty(\vec{a}) = \mathbf{A} \restriction G_\infty(\vec{a})$. If we define the closed (algebraic) terms with parameters in $\vec{a}$ in the obvious way,

$$E :\equiv a_i \mid 0 \mid 1 \mid \boldsymbol{\phi}_i(E_1, \dots, E_{n_i}),$$

then, easily,

$$G_m(\vec{a}) = \{\mathrm{den}(E) \mid \mathrm{depth}(E) \leq m\}.$$

When we need to note the algebra in which these sets are computed, we write $G_m^{\mathbf{A}}(\vec{a}), \mathbf{G}_m^{\mathbf{A}}(\vec{a})$.

Now, it is natural to assume that if a function $f : A^n \to A$ is computed by some algorithm from the givens $\phi_1, \dots, \phi_k$ of $\mathbf{A}$, then for every $\vec{a}$, $f(\vec{a}) \in G_\infty(\vec{a})$; and so we can associate with f its **value complexity**,

$$\mathbf{v}_f^{\mathbf{A}}(\vec{a}) = \text{the least } m \text{ such that } f(\vec{a}) \in G_m^{\mathbf{A}}(\vec{a}). \tag{2}$$

Moreover, since it evidently takes m steps to construct an object in $G_m(\vec{a}) \setminus G_{m-1}(\vec{a})$ using the givens; and since the time complexity of an algorithm must be at least as large as the number of steps it takes to construct its value; I would claim that the following principle is more-or-less obvious:

Basic Principle. *If an algorithm α computes $f : A^n \to A$ in the algebra $\mathbf{A}$ with time complexity $c_\alpha(\vec{x})$, then*

$$f(\vec{a}) \in G_\infty(\vec{a}) \text{ and } c_\alpha(\vec{a}) \geq \mathbf{v}_f^{\mathbf{A}}(\vec{a}).$$

[2] A partial function $f : A^n \rightharpoonup A$ is an ordinary function $f : S \to A$, with *domain of convergence* some subset $S \subseteq A^n$. We write $f(\vec{x}) \downarrow \iff \vec{x} \in S$ and $f(\vec{x}) \uparrow \iff \vec{x} \notin S$. Partial functions compose *strictly*:

$$\begin{aligned} &f(g_1(\vec{x}), \dots, g_m(\vec{x})) = w \\ &\qquad\qquad \iff (\exists u_1, \dots, u_m)[g_1(\vec{x}) = u_1 \;\&\; \cdots \;\&\; g_m(\vec{x}) = u_m \;\&\; f(u_1, \dots, u_m) = w]. \end{aligned}$$

We assume the existence of codes of falsity and truth ($0 \neq 1$) in A for simplicity only, so that we can restrict the givens to be partial functions: each relation $R \subseteq A^n$ on A is identified with its (total) characteristic function, $\chi_R(\vec{x}) =$ if $R(\vec{x})$ then 1 else 0.

Van den Dries' proof of Theorem A proceeds by showing that if (a,b) satisfy Pell's equation, then

$$\mathbf{v}^{\mathbf{A}}_{\mathrm{gcd}}(a+1,b) \geq \frac{1}{4}\sqrt{\log_2 \log_2 b}$$

and then applying the Basic Principle.

3. Lower bounds for recursive (McCarthy) programs

The value complexity $\mathbf{v}^{\mathbf{A}}_{R}(\vec{a}) = 0$ for every relation $R \subseteq A^n$, since $\chi_R(\vec{a}) \in G_0(\vec{a})$, and so the method of proof of Theorem A is useless for coprimeness. For this we need to make some more substantial assumptions about algorithms and it is not immediately obvious what these assumptions might be. So we follow a less direct route: first we will establish a lower bound for coprimeness for algorithms which are expressed by *recursive programs*, and then (perhaps using the insights gained by the proof) we will see how generally applicable this lower bound may be.

The *programming language* $L(\Phi) = L(\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_k)$ has

- individual variables $v_0, v_1, \dots$;
- individual constants $0, 1$;
- (partial) function variables $p^n_0, p^n_1 \dots$ of arity n, for every n; and
- (partial) function constants $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_k$.

The terms of $L(\Phi)$ are defined as usual, except that we allow conditionals:

$$(\Phi\text{-terms}) \quad E :\equiv 0 \mid 1 \mid v_i \mid \boldsymbol{\phi}_i(E_1, \dots, E_{n_i}) \mid p^n_i(E_1, \dots, E_n) \\ \mid \text{ if } (E_0 = 0) \text{ then } E_1 \text{ else } E_2$$

These are interpreted in a Φ-algebra $\mathbf{A}$ with respect to an assignment σ which associates with each individual variable v_i some $\sigma(v_i) \in A$ and each function variable p^n_i a partial function $\sigma(p^n_i) : A^n \rightharpoonup A$, as usual, but we must be careful in defining the denotation of the conditional:

$$\mathrm{den}(\text{ if } (E_0 = 0) \text{ then } E_1 \text{ else } E_2, \sigma) \\ = \begin{cases} \mathrm{den}(E_1, \sigma), & \text{if } \mathrm{den}(E_0, \sigma) = 0, \\ \mathrm{den}(E_2, \sigma), & \text{if } \mathrm{den}(E_0, \sigma)\downarrow \ \& \ \mathrm{den}(E_0, \sigma) \neq 0; \end{cases}$$

so if $\mathrm{den}(E_0, \sigma) = 0$ and $\mathrm{den}(E_1, \sigma)\downarrow$, then $\mathrm{den}(\text{ if } (E_0 = 0) \text{ then } E_1 \text{ else } E_2, \sigma)\downarrow$, even if $\mathrm{den}(E_2, \sigma)\uparrow$.[3]

A *recursive* (or *McCarthy*) Φ-*program* is a system of recursive equations[4]

$$(3) \qquad \alpha \quad : \quad \begin{cases} p_0(\vec{x}) = E_0(\vec{x}, p_1, \dots, p_K) \\ p_1(\vec{x}_1) = E_1(\vec{x}_1, p_1, \dots, p_K) \\ \vdots \\ p_K(\vec{x}_K) = E_K(\vec{x}_n, p_1, \dots, p_K) \end{cases}$$

where only the indicated individual and function variables may occur in the Φ-terms on the right; the first equation is the *head* of α (with *head symbol* p_0), while the remaining K equations comprise the *body* of α. The *arity* of α is the number of

---

[3]Which is why the conditional construct cannot be defined as composition with some partial function $c(x,y,z)$, see Footnote 2.

[4]These programs were introduced in [8], where it was shown that they yield a very simple and elegant characterization of the Turing computable functions on $\mathbb{N}$.

individual variables in the list $\vec{x} = (x_1, \dots, x_n)$ of the head equation. For example, the Euclidean algorithm is expressed by the following $\{<, \mathrm{rem}\}$-program:

$$\varepsilon \quad : \quad \begin{cases} p_0(x,y) = & \text{if } (<(x,y) = 0) \ \text{ then } p_1(x,y) \ \text{ else } p_1(y,x) \\ p_1(x,y) = & \text{if } (\mathrm{rem}(x,y) = 0) \ \text{ then } y \ \text{ else } p_1(y, \mathrm{rem}(x,y)) \end{cases}$$

To interpret α in a Φ-algebra $\mathbf{A}$, we fist check that each term $E_i(\vec{x}_i, p_1, \dots, p_K)$ defines a *monotone and continuous functional* of its individual and partial function arguments, and then we invoke a simple, classical *Fixed Point Theorem* for such functionals which insures that (as a system of equations) α has a tuple $(\overline{p}_0, \dots, \overline{p}_K)$ of least solutions and set (in several useful, alternative notations),

$$\mathrm{den}^{\mathbf{A}}(\alpha) = \overline{\alpha}^{\mathbf{A}} = \overline{p}_0 : A^n \rightharpoonup A \quad (n = \ \mathit{arity}(\alpha)).$$

For example, in the algebra $(\mathbb{N}, 0, 1, <, \mathrm{rem})$, if $x, y \neq 0$, then $\overline{\varepsilon}(x,y) = \gcd(x,y)$.

In the definition of programs we allow $K = 0$, in which case α is identified with its head term $E_0(\vec{x})$—and so every Φ-term is also a program. In fact, many of the standard properties of Φ-terms can be extended to the more general Φ-programs, and this is the key to the derivation of lower bound results for these representations of algorithms.

An *imbedding* $\pi : \mathbf{B} \to \mathbf{A}$ from one Φ-algebra into another is any one-to-one mapping $\pi : B \rightarrowtail A$ such that $\pi(0) = 1$, $\pi(1) = 1$, and for every $\boldsymbol{\phi}_i \in \Phi$ and all $x_1, \dots, x_{n_i}, w \in B$:

$$\boldsymbol{\phi}_i^{\mathbf{B}}(x_1, \dots, x_{n_i}) = w \implies \boldsymbol{\phi}_i^{\mathbf{A}}(\pi(x_1), \dots, \pi(x_{n_i})) = \pi(w).$$

If $\mathbf{B} \subseteq \mathbf{A}$ and the identity map is an imbedding, we call $\mathbf{B}$ a (partial) *subalgebra* of $\mathbf{A}$ and write $\mathbf{B} \subseteq_p \mathbf{A}$. Thus, for every m and all $\vec{a}$, $\mathbf{G}_m^{\mathbf{A}}(\vec{a}) \subseteq_p \mathbf{G}_{m+1}^{\mathbf{A}}(\vec{a})$.

We need these careful formulations of the standard notions because the givens may be partial, but with them, we can verify (very easily) the expected, standard results:

Lemma 1 (Imbedding)**.** *If $\pi : \mathbf{B} \rightarrowtail A$ is an imbedding of one Φ-algebra into another and α is a Φ-program, then*

$$\mathrm{den}^{\mathbf{B}}(\alpha)(x_1, \dots, x_n) = w \implies \mathrm{den}^{\mathbf{A}}(\alpha)(\pi(x_1), \dots, \pi(x_n)) = \pi(w).$$

In particular, if $\mathbf{B} \subseteq_p \mathbf{A}$, then $\overline{\alpha}^{\mathbf{B}} = \overline{\alpha}^{\mathbf{A}} \restriction B$.

Lemma 2 (Finiteness)**.** *Fix a Φ-program α and a Φ-algebra $\mathbf{A}$: if $\mathrm{den}(\alpha)(\vec{a}) = w$, then there is some m such that $w \in G_m(\vec{a})$ & $\mathrm{den}^{\mathbf{G}_m(\vec{a})}(\alpha)(\vec{a}) = w$.*

This second lemma says (in effect) that the "computation" of $\mathrm{den}(\alpha)(\vec{a})$ takes place in the subalgebra of some finite depth generated by $\vec{a}$. It leads directly to the following, natural complexity measure of a Φ-program α on a Φ-algebra $\mathbf{A}$:

(4) If $\mathrm{den}^{\mathbf{A}}(\alpha)(\vec{a}) = w$, set

$$C^{\mathbf{A}}(\alpha)(\vec{a}) = \text{the least } m \text{ such that } \mathrm{den}^{\mathbf{G}_m(\vec{a})}(\alpha)(\vec{a}) = w.$$

This complexity measure is preserved by imbeddings and leads to

Lemma 3 (The Imbedding Test)**.** *Suppose $\mathbf{A}$ is a Φ-algebra, $f : A^n \to A$, and for some $\vec{a} \in A^n$ and some m there is an imbedding $\pi : \mathbf{G}_m^{\mathbf{A}}(\vec{a}) \rightarrowtail \mathbf{A}$ such that $\pi(f(\vec{a})) \neq f(\pi(\vec{a}))$; then for every Φ-program α which computes f in A,*

$$C^{\mathbf{A}}(\alpha)(\vec{a}) > m.$$

Proof. If $\mathrm{den}^{\mathbf{A}}(\alpha)(\vec{a}) = w$ and $C^{\mathbf{A}}(\alpha)(\vec{a}) \le m$, then $\mathrm{den}^{\mathbf{G}_m(\vec{a})}(\alpha)(\vec{a}) = w$ by the definition of the complexity and the Imbedding Lemma 1, and then by the same Lemma, $f(\pi(\vec{a})) = \mathrm{den}^{\mathbf{A}}(\alpha)(\pi(\vec{a})) = \pi(w)$, which contradicts the hypothesis. □

Theorem B$_2$ ([4]). *For every recursive program α which decides the coprimeness relation $\perp$ in the algebra*

$$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, <, =, \mathrm{iq}, \mathrm{rem}),$$

there are infinitely many pairs (a, b) such that $a > b > 1$ and

$$C^{\mathbf{A}}(\alpha)(a, b) > \frac{1}{10} \log_2 \log_2 a;$$

in fact the inequality holds for all $a > b > 1$ such that

$$a \perp b \text{ and } \left| \frac{a}{b} - \sqrt{2} \right| < \frac{1}{b^2}$$

(including all solutions (a, b) of Pell's equation).

Method of proof. To apply the Imbedding Test, it is enough to show that if (a, b) satisfy the hypothesis and $m \le \frac{1}{10} \log_2 \log_2 a$, then there is a $\lambda > 1$ and an imbedding $\pi : \mathbf{G}_m(a, b) \rightarrowtail \mathbf{A}$ such that $\pi(a) = \lambda a$ and $\pi(b) = \lambda b$. This is done by showing first that (under the hypothesis) each $x \in G_m(a, b)$ can be expressed uniquely in the form

$$x = \frac{x_0 + x_1 a + x_2 b}{x_3}$$

with sufficiently small $x_i \in \mathbb{Z}$, and then choosing λ so that the map

$$\pi(x) = \frac{x_0 + x_1 \lambda a + x_2 \lambda b}{x_3} \quad (x \in G_m(a, b))$$

is an imbedding. □

The detailed proof uses only minimal number theory—basically Liouville's Theorem, c.f. [5]—and it is very general. It can be adjusted to derive lower bounds for recursive programs which decide many interesting relations on $\mathbb{N}$ from natural givens, see [4, 3, 1].

4. Logical extensions and widely applicable lower bounds

Theorem B$_2$ does not yield immediately a lower bound for Random Access Machines which decide the coprimeness relation from $+, \dot{-}, <, =, \mathrm{iq}, \mathrm{rem}$, because the simulation of RAMs by recursive programs has an "overhead", cf. the article by van Emde Boas in [11]. The imbedding method, however, can be extended to yield very widely applicable lower bounds, as follows.

Suppose first that $A \subseteq B$ and

$$\mathbf{A} = (A, 0, 1, \phi_1, \ldots, \phi_k), \quad \mathbf{B} = (B, 0, 1, \phi_1, \ldots, \phi_k, \psi_1, \ldots, \psi_l)$$

are algebras, where we view each ϕ_i as a partial function on B, defined only when all its arguments are in A. We call $\mathbf{B}$ a *logical extension* of A if every permutation $\pi : A \rightarrowtail\!\!\!\!\twoheadrightarrow A$ such that $\pi(0) = 0, \pi(1) = 1$ has an extension $\pi^B : B \rightarrowtail\!\!\!\!\twoheadrightarrow B$ which is an automorphism for all the "fresh givens" of $\mathbf{B}$, i.e.,

$$\pi^B(\psi_j(x_1, \ldots, x_m)) = \psi_j(\pi^B(x_1), \ldots, \pi^B(x_n)) \quad (j = 1, \ldots, l, x_1, \ldots, x_m \in B).$$

It is quite easy to verify that every standard computation model relative to some functions Φ on a set A—Turing machine, RAM, etc.—can be *faithfully represented* by a recursive program on some logical extension $\mathbf{B}$ of $(A, 0, 1, \Phi)$, so that, in particular, its time complexity is no smaller than the $C^{\mathbf{B}}$ complexity of the recursive program which represents it.

Suppose next that $\mathbf{A}$ is a Φ-algebra and $f : A^n \to A$ is such that

$$f(\vec{x}) \in G_\infty(\vec{a}) \quad (\vec{a} \in A^n). \tag{5}$$

According to the Basic Principle, this condition is satisfied by every function which is computed by an algorithm from the givens of $\mathbf{A}$. We say that an imbedding $\pi : \mathbf{G}_m(\vec{a}) \rightarrowtail \mathbf{A}$ *respects* f *at* $\vec{a}$ if

$$f(\vec{a}) \in G_m(\vec{a}) \ \& \ \pi(f(\vec{a})) = f(\pi(\vec{a})),$$

and we define the **imbedding complexity** of f in $\mathbf{A}$ by

$$\imath_f^{\mathbf{A}}(\vec{a}) = \text{the least } m \text{ such that every } \pi : \mathbf{G}_m^{\mathbf{A}}(\vec{a}) \rightarrowtail \mathbf{A} \text{ respects } f \text{ at } \vec{a}. \tag{6}$$

Theorem 4 (van den Dries,YNM, Itay Neeman)**.** *If $f : A^n \to A$ and some recursive program α computes f on a logical extension $\mathbf{B}$ of $\mathbf{A}$, then* (5) *holds, $\imath_f^{\mathbf{A}}(\vec{a})$ is defined for every $\vec{a} \in A^n$, and*

$$\imath_f^{\mathbf{A}}(\vec{a}) \leq C^{\mathbf{B}}(\alpha)(\vec{a}).$$

And a judicious combination of the proofs of this Theorem and Theorem B_2 gives the most general version of the lower bound result for coprimeness which we have been using as our basic example:

Theorem B. *If $\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, <, =, \text{iq}, \text{rem})$, $a \perp b$ and $\left|\dfrac{a}{b} - \sqrt{2}\right| < \dfrac{1}{b^2}$, then*

$$\imath_{\text{gcd}}^{\mathbf{A}}(a, b) > \frac{1}{10} \log_2 \log_2 a,$$

so that for every recursive program β which decides coprimeness in some logical extension $\mathbf{B}$ of $\mathbf{A}$, $C^{\mathbf{B}}(\beta)(a, b) > \dfrac{1}{10} \log_2 \log_2 a$.

One can give arguments in favor of a *Church-Turing Thesis for algorithms*, by which *every algorithm from given functions and relations Φ on some set A can be faithfully represented by a recursive program in some logical extension $\mathbf{B}$ of* $(A, 0, 1, \Phi)$. This would then remove the "vague" qualification from Theorem B_1. Whatever the value of such foundational arguments, however, the lower bound of Theorem B applies to the time complexities of all known computation models. The method can also be used to establish very generally applicable lower bounds from various givens for many relations on $\mathbb{N}$, including "a is prime", "a is a perfect square" or "square-free", the Jacobi symbol $(\frac{a}{n})$, etc., cf. [3, 4, 1].

Relevant results (by different methods) in the literature can be found in [7, 6, 9]. In these papers, however, multiplication is included among the givens, and so their results are not directly comparable to the theorems we analyzed here—except for van den Dries' Theorem A, which gives a better lower bound than that in [6].

References

1. J. Busch, *On the optimality of the binary algorithm for the Jacobi symbol*, Fundamenta Informaticae, to appear.
2. Lou van den Dries, *Generating the greatest common divisor, and limitations of primitive recursive algorithms*, Foundations of Computational Mathematics **3** (2003), 297–324.
3. Lou van den Dries and Yiannis N. Moschovakis, *Arithmetic complexity*, to appear in ACM Transactions on Computational Logic.

4. ———, *Is the Euclidean algorithm optimal among its peers?*, Bulletin of Symbolic Logic (2004), 390–418.
5. G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, Clarendon Press, Oxford, fifth edition (2000), originally published in 1938.
6. Yishay Mansoor, Baruch Schieber, and Prasoon Tiwari, *A lower bound for integer greatest common divisor computations*, Journal of the Association for Computing Machinery **38** (1991), 453–471.
7. ———, *Lower bounds for computations with the floor operation*, SIAM Journal on Computing **20** (1991), 315–327.
8. J. McCarthy, *A basis for a mathematical theory of computation*, Computer programming and formal systems (P. Braffort and D Herschberg, eds.), North-Holland, 1963, pp. 33–70.
9. João Meidânis, *Lower bounds for arithmetic problems*, Information Processing Letters **38** (1991), 83–87.
10. Yiannis N. Moschovakis, *On primitive recursive algorithms and the greatest common divisor function*, Theoretical Computer Science **301** (2003), 1–30.
11. P. van Emde Boas, *Machine models and simulations*, Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity (Jan van Leeuwen, ed.), Elsevier and MIT Press, 1994, pp. 1–66.

Department of Mathematics, University of California, Los Angeles, CA, USA,
and
Graduate Program in Logic, Algorithms and Computation, Department of Mathematics, University of Athens, Athens, Greece
E-mail address: ynm@math.ucla.edu