# ON PRIMITIVE RECURSIVE ALGORITHMS AND THE GREATEST COMMON DIVISOR FUNCTION

YIANNIS N. MOSCHOVAKIS

Abstract. We establish linear lower bounds for the complexity of non-trivial, primitive recursive algorithms from *piecewise linear* given functions. The main corollary is that logtime algorithms for the *greatest common divisor* from such givens (such as *Stein*'s) cannot be matched in efficiency by primitive recursive algorithms from the same given functions. The question is left open for the *Euclidean algorithm*, which assumes the remainder function.

In 1991, Colson [3][1] proved a remarkable theorem about the limitations of *primitive recursive algorithms*, which has the following consequence:

**Colson's Corollary.** *If a primitive recursive derivation of* $\min(x, y)$ *is expressed faithfully in a programming language, then one of the two computations* $\min(1, 1000)$ *and* $\min(1000, 1)$ *will take at least* $1000$ *steps.*

The point is that the natural algorithm which computes $\min(x, y)$ in $O(\min(x, y))$ steps cannot be matched in efficiency by a primitive recursive program, even though $\min(x, y)$ is a primitive recursive function; and so, as a practical and (especially) a foundational matter, we need to consider "recursive schemes" more general than primitive recursion, even if, ultimately, we are only interested in primitive recursive functions.

In this paper we consider extensions of Colson's Theorem which allow conditional definitions and *especially* calls to a rich variety of "given" functions, whose values are produced on demand in constant time. Sample, easy to state, result:

**Corollary 20.** *Consider primitive-recursive-like derivations, which in addition to composition and primitive recursion allow definition by cases and calls to the following functions and* (characteristic functions of) *relations:*

$$x + y, \quad x \dot{-} y, \quad x \div 2, \quad \text{Parity}(x), \quad x = y, \quad x < y$$

*For each such derivation of the greatest common divisor function* $\gcd(x, y)$*, there is a sequence of pairs* $\{(x_t, y_t)\}$ *and a rational constant* $r > 0$*, such that*

$$\lim_t(x_t + y_t) = \infty, \text{ and for all } t, c^*(x_t, y_t) \geq r(x_t + y_t),$$

The research reported here was partially supported by Grant #70/4/5633 from the Research Committee of the University of Athens. I am also grateful to the *Graduate Program in Logic, Algorithms and Computation* (ΜΠΛΑ), for some additional financial and much moral support.

I thank Elias Koutsoupias who was (really) a collaborator in the early stages of this work (see Footnote 5); René David, for his useful comments on early versions of the paper; and Lou van den Dries for his comments which influenced substantially the final, revised version of the paper, and (more significantly) for his subsequent contributions to the topic, see 8.1.

[1]Colson proved a general result about (absolute) call-by-name primitive recursion, which implies this Corollary, and David [4] extended Colson's result using a new method; the call-by-value version of the theorem was established by Fredholm [6, 7].

where the *essential complexity measure* $c^*(x,y)$ is lower than both the strict and non-strict (parallel) complexity measures for primitive recursive algorithms from arbitrary given functions.[2]

It follows that *Stein's algorithm* which computes $\gcd(x,y)$ with strict complexity $O(\log_2(x)+\log_2(y))$, using the givens listed in the theorem and a very simple (but not primitive) recursion scheme, cannot be matched in efficiency using only "primitive-recursive-like" recursive definitions.

We will start in Section 1 with some precise definitions of (mostly) familiar notions, and then give in Section 2 a detailed proof of the strict (call-by-value) version of Colson's Theorem, which sets the pattern for the later results. Section 3 develops some simple ideas from linear programming, which are then used in Section 4 to effect the main construction of the paper for the call-by-value case; this is strengthened by the introduction of conditionals and the *essential complexity* measure in Section 5, and again in Section 6, where it is shown that the essential complexity is no larger than the non-strict complexity measure. The main result of the paper is established in Section 7. Finally, in Section 8, we discuss briefly the connection of this work with the work of Colson, Fredholm and David which inspired it, and we formulate two relevant open problems.

## 1. Preliminaries

1.1. **Primitive recursive derivations.** We consider *primitive recursive derivations* or **prd**'s for short, from a given set of functions $\Phi$, i.e., sequences

$$\Gamma = f_0, \dots, f_k,$$

of functions on $\mathbb{N} = \{0, 1, \dots\}$ such that each $f_i$ satisfies one of the following conditions:

D1. $f_i$ is a *given* (function), either one of the *basic* functions $S(x) = x+1$, $P^n_j(\vec{x}) = x_j$, $C^n_q(\vec{x}) = q$, or an *external given*, $f_i \in \Phi$.

D2. $f_i$ is defined by *composition*

$$f(\vec{x}) = h(g_1(\vec{x}), \dots, g_m(\vec{x})) \tag{1}$$

from functions $h, g_1, \dots, g_m$ which are listed earlier in the derivation.

D3. $f_i$ is defined by *primitive recursion*

$$\left\{\begin{array}{rcl} f(0,\vec{x}) & = & g(\vec{x}) \\ f(y+1,\vec{x}) & = & h(f(y,\vec{x}),y,\vec{x}) \end{array}\right. \tag{2}$$

from functions $g, h$ which are listed earlier in the derivation. (We allow here the empty tuple $\vec{x}$, in which case $g(\ ) = C^0_q = q$ is a number.)

We will systematically confuse "functions" $f_i$ and "symbols" naming them in such programs, and we will assume that $\Gamma$ associates with each $i \leq k$ some fixed justification by D1, D2 or D3 for the inclusion of $f_i$ in the sequence.

[2]The essential complexity of a primitive recursive algorithm is an "optimized" call-by-value time complexity, in which the given functions are assumed computed in constant time (one time unit), and arguments of functions are computed first, but "only if needed". The *strict* and *non-strict* complexities count the steps of the iterations determined by the algorithm on the complete posets

$$\mathbb{N}^n \to \mathbb{N} \cup \{\bot\} \text{ and } \mathrm{Mon}((\mathbb{N}\cup\{\bot\})^n \to \mathbb{N} \cup \bot\})$$

of *strict* and *non-strict* partial functions respectively, with the given funtions available from the first stage. Precise definitions are given in Subsection 1.2 and Sections 5 and 6.

1.2. **The strict (parallel, call-by-value) complexity.** A (strict) partial function $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ is a function $f : \mathbb{N}^n \to \mathbb{N} \cup \{\bot\}$, and we write

$$f(\vec{x})\downarrow \iff f(\vec{x}) \neq \bot;$$

the "strict" means that these objects compose strictly, i.e., for $\vec{x}, w \in \mathbb{N}$,

$$f(g_1(\vec{x}), \dots, g_m(\vec{x})) = w$$
$$\iff (\exists u_1, \dots, u_m \in \mathbb{N})[g_1(\vec{x}) = u_1, \dots, g_m(\vec{x}) = u_m, f(\vec{u}) = w].$$

Each **prd** determines a system of recursive equations on partial functions, which can be iterated to "compute" the functions defined by the program in the usual way:

R1. If $f$ is a given function $\phi(\vec{x})$, then, for every $k$,

$$f^k(\vec{x}) = \phi(\vec{x}),$$

so that these functions are available immediately from stage 0.

R2. If $f$ is defined by composition as in (1), then

$$f^0(\vec{x}) = \bot, \quad f^{k+1}(\vec{x}) = g^k(h_1^k(\vec{x}), \dots, h_m^k(\vec{x})).$$

R3. If $f$ is defined by primitive recursion as in (2), then

$$f^0(y, \vec{x}) = \bot,$$
$$f^{k+1}(0, \vec{x}) = g^k(\vec{x}),$$
$$f^{k+1}(y+1, \vec{x}) = h^k(f^k(y, \vec{x}), y, \vec{x}).$$

We then set, for each function in $\Gamma$,

$$c_f^s(\vec{x}) = \text{the least } k \text{ such that } f^k(\vec{x})\downarrow, \tag{3}$$

so that the (strict, parallel) *complexity measure* $c_f^s(\vec{x})$ gives the (minimum) number of steps required for a call-by-value computation of $f(\vec{x})$ by the program $\Gamma$.

1.3. **$\Phi$-terms and $V_\Gamma$-terms.** A $\Phi$-*term* is an (explicit) term in the vocabulary

$$V_\Phi = \{0, S\} \cup \Phi \tag{4}$$

with symbols for 0, the successor and the "external givens" in $\Phi$ (if any). We also associate with $\Gamma$ the richer vocabulary

$$V_\Gamma = \{0, S, f_{k_1}, \dots, f_{k_m}\} \cup \Phi \tag{5}$$

which includes a symbol $f_{k_i}$ for each function defined in $\Gamma$ by a primitive recursion—these are the *recursive symbols* of $V_\Gamma$. If $E$ is a $V_\Gamma$-*term*—a term explicit in this vocabulary—and $\pi : \mathbb{N} \to \mathbb{N}$, we set

$$d_\Gamma(E, \pi) = d(E, \pi) = \text{the value of } E \text{ for } \mathsf{v}_i := \pi(i),$$

where $\mathsf{v}_0, \mathsf{v}_1, \dots$ are the formal variables. Updates of assignments are defined as usual,

$$\pi\{\mathsf{v}_k := y\}(j) \begin{cases} y & \text{if } j = k, \\ \pi(j) & \text{otherwise.} \end{cases}$$

It is clear that each $f_i$ with arity $n_i$ can be defined by a $V_\Gamma$-term $E_{f_i}$ whose free variables are among the first $n$ formal variables $\mathsf{v}_0, \dots, \mathsf{v}_{n_i-1}$. The precise definition is by induction on $i$:

T1. If $f_i(\vec{x}) = \phi(\vec{x})$ for some $\phi \in \Phi$, then $E_\phi \equiv \phi(\vec{\mathsf{v}})$. For the standard givens, we set (with the obvious notation)

$$E_S \equiv S(\mathsf{v}_0), \quad E_{C^n_q} \equiv \Delta q, \quad E_{P^n_j} \equiv \mathsf{v}_{j-1},$$

where the *numeral* $\Delta q \equiv S^q(0)$ is the canonical term denoting the number $q$.

T2. If $f$ is defined by composition (1), then

$$E_f \equiv E_h\{\mathsf{v}_0 :\equiv E_{g_1}, \dots, \mathsf{v}_{m-1} :\equiv E_{g_m}\},$$

where $:\equiv$ indicates the (simultaneous) formal *replacement* of variables by terms.

T3. If $f$ is defined by primitive recursion (2), we then have the recursive symbol (for) $f$ in the vocabulary $V_\Gamma$, and we simply set

$$E_f \equiv f(\mathsf{v}_0, \mathsf{v}_1, \dots, \mathsf{v}_n).$$

To simplify notation when we deal with these terms, we also set

$$(6) \quad d(E, x_0, \dots, x_{n-1}) = d(E, \pi) \qquad \text{where } \pi(0) = x_0, \dots, \pi(n-1) = x_{n-1}, \pi(n+j) = 0;$$

by an easy induction then, with this notation, for each symbol $f$ in $\Gamma$,

$$(7) \qquad f(\vec{x}) = d(E_f, \vec{x}).$$

1.4. **Ranks.** The key notion which we will use in the proofs is that of the *rank* of a $V_\Gamma$-term, which is characteristic of "primitive-recursive-like" algorithms; it makes sense only when we have a "derivation", where each function is defined from functions preceding it.

Let $f_{k_1}, \dots, f_{k_m}$ be the recursive symbols which are defined in a derivation $\Gamma$ with $k_1 < k_2 < \dots < k_m$, and for each $V_\Gamma$-term $E$, set

$$(8) \qquad \text{rank}(E) = \langle l_m, l_{m-1}, \dots, l_1 \rangle,$$

where $l_i$ is the number of occurrences of $f_{k_i}$ in $E$.

Ranks are ordered lexicographically, so that $\Phi$-terms have the least rank $\langle 0, 0, \dots, 0\rangle$, and $\langle 3, 0, \dots\rangle > \langle 2, 10, \dots\rangle$.

1.5. **The strict complexity on terms.** With each $V_\Gamma$-term $E$, we associate a number

$$c(E, \pi) = c_\Gamma(E, \pi)$$

which represents (roughly) the least number of steps required for a "fully parallel, call-by-value" computation of $d(E, \pi)$. As with denotation functions, we will occasionally simplify notation by using the convention

$$(9) \quad c(E, x_0, \dots, x_{n-1}) = c(E, \pi) \qquad \text{where } \pi(0) = x_0, \dots, \pi(n-1) = x_{n-1}, \pi(n+j) = 0,$$

which is especially useful when $E \equiv E_f \equiv f(\mathsf{v}_0, \dots, \mathsf{v}_{n-1})$.

The definition of $c(E, \pi)$ is by induction on the rank of $E$, and within this by induction on the length of $E$.

CT1. $c(\mathsf{v}_i, \pi) = c(0, \pi) = 0$.

CT2. $E \equiv f(E_1, \dots, E_m)$, where $m = 1$ and $f$ is the successor symbol $S$, or $f$ is an $m$-ary symbol in $\Phi$. Now each $E_i$ has rank no greater than rank$(E)$ and smaller length, and so $c(E_i, \pi)$ is defined; we set

$$c(f(E_1, \dots, E_m), \pi) = \max\{c(E_1, \pi), \dots, c(E_m, \pi)\}.$$

By these first two clauses, if $E$ is a $\Phi$-term, then $c(E, \pi) = 0$.

CT3. Suppose $E \equiv f(E_0, E_1, \dots, E_n)$ and $f$ is defined by primitive recursion as in (2). Now $\text{rank}(E_i) < \text{rank}(E)$, because no recursive symbol occurs more times in $E_i$ than in $E$, and $f$ occurs at least one more time in $E$ than it does in any $E_i$. Let

$$d^j = d(E_j, \pi), \quad c^j = c(E_j, \pi) \quad (j \leq n).$$

We will first define

$$c(E_f, y, x_1, \dots, x_n) = c(f(\mathsf{v}_0, \mathsf{v}_1, \dots, \mathsf{v}_n), y, x_1, \dots, x_n)$$

by induction on $y$, and then we will set

$$(10) \qquad c(f(E_0, E_1, \dots, E_n), \pi) = \max\{c^0, \dots, c^n, c(E_f, d^0, d^1, \dots, d^n)\}.$$

CT3.0. Clearly $\text{rank}(E_g) < \text{rank}(E_f) \leq \text{rank}(E)$, and we can set

$$c(E_f, 0, \vec{x}) = c(E_g, \vec{x}).$$

CT3.1. Again $\text{rank}(E_h) < \text{rank}(E_f) \leq \text{rank}(E)$, and we can set

$$c(E_f, y+1, \vec{x}) = \max\{c(E_f, y, \vec{x}) + 1, c(E_h, f(y, \vec{x}), y, \vec{x})\}.$$

**Lemma 1.** *For each symbol $f$ in a* **prd** $\Gamma$,

$$c^s_f(\vec{x}) \geq c(E_f, \vec{x}).$$

PROOF. It is enough to show that for every $k$,

$$f^k(\vec{x})\downarrow \implies c(E_f, \vec{x}) \leq k,$$

and this is simple, by induction on $k$. ⊣

The main feature of this complexity measure is that it assigns no cost to composition, but it takes recursion seriously:

**Lemma 2.** *If $f$ is defined by the primitive recursion* (2) *in $\Gamma$, then for every $y$, $c(E_f, y, \vec{x}) \geq y$.*

PROOF is by induction on $y$, with a trivial basis. In the induction step, by CT3.1 and the induction hypothesis,

$$c(E_f, y+1, \vec{x}) = \max\{c(E_f, y, \vec{x}) + 1, c(E_h, f(y, \vec{x}), y, \vec{x})\} \geq y + 1. \qquad ⊣$$

**Theorem 3.** *For any $V_\Gamma$-terms $E$, $A_1, \dots, A_k$, if $\mathsf{u}_1, \dots, \mathsf{u}_k$ occur in $E$ and $d_i = d(A_i, \pi)$, $c_i = c(A_i, \pi)$ for $i = 1, \dots, k$, then*

$$(*) \qquad c(E\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) = \max\{c_1, \dots, c_k, c(E, \pi\{\mathsf{u}_1 := d_1, \dots, \mathsf{u}_k := d_k\})\}.$$

PROOF is by induction on the length of $E$.

In the basis, $E \equiv \mathsf{v}_i$ or $E \equiv 0$, and the only non-trivial possibility is when $\mathsf{u}_1 \equiv \mathsf{v}_i$; now $E\{\mathsf{u}_1 :\equiv A\} \equiv A$ and the result is immediate.

In the induction step we distinguish cases following the definition of complexity of terms.

*Case* CT2, $E \equiv f(E_1, \dots, E_m)$ and $f$ is the successor or in $\Phi$. We set

$$c^j = c(E_j\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) \quad (j = 1, \dots, m)$$

and we compute, noting that each $\mathsf{u}_i$ occurs in some $E_j$:

$$\begin{aligned} c(E\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) &= \max\{c^1, \dots, c^m\} \quad \text{(by definition)}\\ &= \max\{c(E_1, \pi\{\vec{\mathsf{u}} := \vec{d}\}), \dots, c(E_m, \pi\{\vec{\mathsf{u}} := \vec{d}\}),\\ &\qquad c_1, \dots, c_k\} \quad \text{(by the induction hypothesis)}\\ &= \max\{c(E, \pi\{\vec{\mathsf{u}} := \vec{d}\}), c_1, \dots, c_k\}. \end{aligned}$$

*Case* CT3, $E \equiv f(E_0, E_1, \dots, E_n)$ and $f$ is defined by primitive recursion as in (2). We set

$$c^j = c(E_j\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) \quad (j \leq n),$$
$$d^j = d(E_j\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi), \quad \vec{\mathsf{u}} \equiv \mathsf{u}_1, \dots, \mathsf{u}_k,\ \vec{d} = (d^1, \dots, d^n).$$

By (10), the induction hypothesis, and (10) again,

$$\begin{aligned} c(E\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) &= \max\{c^0, \dots, c^n, c(E_f, d^0, \vec{d})\}\\ &= \max\{c_1, \dots, c_k,\\ &\qquad c(E_0, \pi\{\vec{\mathsf{u}} := \vec{d}\}), \dots, c(E_n, \pi\{\vec{\mathsf{u}} := \vec{d}\}),\\ &\qquad c(E_f, d^0, \vec{d})\}\\ &= \max\{c_1, \dots, c_k, c(E, \pi\{\vec{\mathsf{u}} := \vec{d}\})\}, \end{aligned}$$

which is the required equation. ⊣

**Corollary 4.** (1) *For any three $V_\Gamma$-terms $E$, $A$, $B$,*

$$c(A, \pi) = c(B, \pi) \Longrightarrow c(E\{\mathsf{u} :\equiv A\}, \pi) = c(E\{\mathsf{u} :\equiv B\}, \pi).$$

(2) *If* $\mathsf{u}$ *occurs in $E$, then $c(E\{\mathsf{u} :\equiv A\}, \pi) \geq c(A, \pi)$.*

Proof. Both claims follow immediately from the theorem, taking cases in (1) on whether $\mathsf{u}$ occurs in $E$ or not. ⊣

## 2. Colson's Theorem (for call-by-value)

We will give in this section a detailed proof of Colson's Theorem for call-by-value, not because it is necessarily new or the simplest, but because it will be used as a pattern for more complex arguments later in the paper. Key to the proof is the following, trivial observation:

**Lemma 5.** *If $\Phi = \emptyset$* (no external givens)*, then the $\Phi$-terms are exactly the terms of the form $S^k(\mathsf{v})$ for some variable $\mathsf{v}$ and the numerals $\Delta k \equiv S^k(0)$.*

**Theorem 6** (Fredholm [6, 7])**.** *For each absolute*($\Phi = \emptyset$) **prd** *which defines a function $f(\vec{x})$ of $n$ variables,* ***either*** *$f(\vec{x})$ is one of the trivial functions*

$$f(\vec{x}) = q, \quad f(\vec{x}) = x_i + q,$$

***or*** *there exists some $i$, $1 \leq i \leq n$, such that for all $\vec{x}$,*

$$c^s_f(\vec{x}) \geq c(E_f, \vec{x}) \geq x_i.$$

Proof. A subterm $M$ of a $V_\Gamma$-term $E$ is *critical*, if

$$M \equiv f(G, \vec{N}), \tag{11}$$

where $f$ is a recursive symbol and $G, \vec{N}$ are $\Phi$-terms, and a $V_\Gamma$-term $E$ is *reducible* if it has a critical subterm of the form

$$M \equiv f(\Delta k, \vec{N}), \tag{12}$$

otherwise it is *irreducible*.

Consider the following one-step-reduction procedure on reducible terms:

1. Choose a critical subterm $M$ of $E$ of the form (12).
2. If $f$ is defined by the primitive recursion (2) in $\Gamma$, let $E'$ be obtained from $E$ by replacing $M$ by the term $M_k$, where $M_i$ is defined by the obvious recursion:

$$M_0 \equiv E_g\{\vec{\mathsf{v}} :\equiv \vec{N}\} \tag{13}$$

$$M_{i+1} \equiv E_h\{\mathsf{v}_0 :\equiv M_i, \mathsf{v}_1 :\equiv \Delta i, \mathsf{v}_{1+1} :\equiv N_1, \ldots, \mathsf{v}_{1+n} :\equiv N_n\}. \tag{14}$$

**Sublemma 6.1.** *$d(E,\pi) = d(E',\pi)$, $c(E,\pi) \geq c(E',\pi)$ and $\operatorname{rank}(E') < \operatorname{rank}(E)$.*

*Proof*. The first claim is immediate, because

$$d(f(S^i(0), \vec{N}), \pi) = d(M_i, \pi),$$

and the second will follow from

$$c(f(\Delta i, \vec{N}), \pi) = c(E_f, i, \vec{d}) \geq c(M_i, \pi) \quad (\vec{d} = (d(N_1,\pi), \ldots, d(N_n,\pi))$$

by Theorem 3. To prove this last equation by induction on $i$, compute first for the basis:

$$\begin{aligned} c(f(0,\vec{N}),\pi) &= c(E_g, d(N_1,\pi), \ldots, d(N_n,\pi)) \\ &\qquad \text{(because the } \vec{N} \text{ are } \Phi\text{-terms)} \\ &= c(E_g\{\vec{\mathsf{v}} :\equiv \vec{N}\}, \pi) = c(M_0, \pi). \end{aligned}$$

For the induction step, we compute, using again the fact that the $\vec{N}$ are $\Phi$-terms with trivial complexities:

$$\begin{aligned} c(E_f, i+1, \vec{d}) &= \max\{c(E_f, i, \vec{d}) + 1, c(E_h, f(i,\vec{d}), i, \vec{d})\} \\ &\geq \max\{c(M_i,\pi) + 1, c(E_h, f(i,\vec{d}), i, \vec{d})\}; \end{aligned}$$

on the other hand,

$$\begin{aligned} c(M_{i+1},\pi) &= c(E_h\{\mathsf{v}_0 :\equiv M_i, \mathsf{v}_1 :\equiv \Delta i, \mathsf{v}_{1+1} :\equiv N_1, \ldots, \mathsf{v}_{1+n} :\equiv N_n\}, \pi) \\ &\leq \max\{c(M_i,\pi), c(E_h, d(M_i,\pi), i, \vec{d})\} \end{aligned}$$

by Theorem 3 (taking account of the fact that $\mathsf{v}_0$ may not occur in $E_h$), and this completes the argument.

Finally, the third claim holds because the reduction procedure replaces the critical subterm $f(S^k(0), \vec{N})$ by the term $M_k$, in which every recursive symbol occurs earlier in $\Gamma$ than $f$. ⊣ (6.1)

**Sublemma 6.2.** *For each $V_\Gamma$-term $E$, there is another $V_\Gamma$-term $E^*$ such that:*

(1) *$d(E,\pi) = d(E^*,\pi)$.*
(2) *$c(E,\pi) \geq c(E^*,\pi)$.*
(3) *$E^*$ has no critical subterm of the form* (12).

*Proof*. Apply the reduction procedure repeatedly, until you reach an irreducible term $E^*$—which will occur, since every application lowers the rank. ⊣ (6.2)

To prove the theorem, we apply this Sublemma to a term $E_f$ associated with a symbol $f$ in $\Gamma$. If the resulting term $E_f^*$ has no occurrences of recursive symbols, then $E_f^*$ (and so $f$) must be trivial, by Lemma 5. If, on the other hand, some recursive symbol occurs in $E_f^*$, then some recursive symbol $g$ must occur in an *innermost subterm*, which is then critical; and since this critical subterm cannot be of the form (12), it must be of the form $g(S^k(\mathsf{v}_i), \vec{N})$ for some variable $\mathsf{v}_i$, and hence

$$c(E_f, \vec{x}) \geq c(E_g, x_i + k, d(N_1, \vec{x}), \dots, d(N_n, \vec{x})) \geq x_i. \qquad \dashv$$

## 3. Ψ-linear and piecewise Ψ-linear partial functions

We develop here some basic notions from linear programming which we need, both to state and to prove our results. Throughout this section, $\Psi$ is a fixed set of (total) functions on $\mathbb{N}$. We are primarily interested in the "absolute" case $\Psi = \emptyset$ in this article, but the more general "relative" results are just as easy to establish, and they have interesting applications.

### 3.1. **Ψ-linear terms and partial functions.** A $\Psi$-*linear term* is an (explicit) term in the vocabulary

$$V_{\ell\Psi} = \{0, 1, +, -, m\cdot, \frac{1}{m}\cdot\} \cup \Psi, \tag{15}$$

where for each $m \in \mathbb{N}$, $m\cdot$ and $\frac{1}{m}\cdot$ are the unary operations of multiplication and division by $m$, respectively,

$$m \cdot (x) = mx, \quad \frac{1}{m} \cdot (x) = \frac{x}{m};$$

and a partial function $f(\vec{x})$ is $\Psi$-*linear* if it is defined by a $\Psi$-linear term. When $\Psi = \emptyset$, we will just say *linear* instead of $\emptyset$-linear.

The partiality here comes from subtraction and division by $m$ which are only partially defined on $\mathbb{N}$,

$$x - y\downarrow \iff x \geq y, \quad \frac{1}{m} \cdot (x)\downarrow \iff m \text{ divides } x.$$

It extends to the terms whose denotations are composed strictly, so that

$$f(x) = 2 \cdot (\frac{1}{2} \cdot (x)) = \begin{cases} x, & \text{if } x \text{ is even,} \\ \text{undefined,} & \text{otherwise,} \end{cases} \tag{16}$$

and the restriction of the identity function to the even numbers is a linear partial function.

Here are some examples of partial functions with the terms establishing their linearity, where $\chi_R$ is the *characteristic function* of a relation $R$:

| | |
|---|---|
| $S(x)$ | $x + 1$ |
| $P_i^n(\vec{x})$ | $x_i$ |
| $C_q^n(x)$ | $q \cdot (1)$ |
| $x + y$ | $x + y$ |
| $x \dot{-} y$ for $x \geq y$ | $x - y$ |
| $x \dot{-} y$ for $x < y$ | $0 \cdot (y - x - 1)$ |

| | |
|---|---|
| $\lfloor \frac{x}{2} \rfloor$ for even $x$ | $\frac{1}{2} \cdot (x)$ |
| $\lfloor \frac{x}{2} \rfloor$ for odd $x$ | $\frac{1}{2} \cdot (x - 1)$ |
| $\chi_<(x, y)$ for $x < y$ | $1 + 0 \cdot (y - x - 1)$ |
| $\chi_<(x, y)$ for $x \geq y$ | $0 \cdot (x - y)$ |
| $\chi_=(x, y)$ for $x = y$ | $1 + 0 \cdot (x - y) + 0 \cdot (y - x)$ |
| Parity$(x)$ for even $x$ | $0 \cdot (\frac{1}{2} \cdot (x))$ |
| Parity$(x)$ for odd $x$ | $1 + 0 \cdot (\frac{1}{2} \cdot (x - 1))$ |

Immediately from the definition, we get:

**Lemma 7.** *The composition of $\Psi$-linear partial functions is $\Psi$-linear.* ⊣

The absolute linear partial functions admit a very simple representation, as follows.

A *rational linear form* is an expression

$$\rho(\vec{x}) = a_0 + a_1x_1 + \cdots + a_nx_n \quad (a_0, \ldots, a_n \in \mathbb{Q}) \tag{17}$$

where $\mathbb{Q}$ is the set of (positive and negative) rational numbers, e.g., $\frac{x}{2} - 3$. These define partial functions on $\mathbb{N}$, defined when their value is a natural number.

**Lemma 8.** *A partial function $f(\vec{x})$ is linear* ($\emptyset$-linear) *if and only if for suitable rational, linear forms $\rho_1(\vec{x}), \ldots, \rho_k(\vec{x}), \rho_{k+1}(\vec{x})$,*

$$\begin{aligned} (18)\quad & f(\vec{x}) = w \\ & \iff (\exists u_1, \ldots, u_k \in \mathbb{N})[u_1 = \rho_1(\vec{x}) \mathbin{\&} \cdots \mathbin{\&} u_k = \rho_k(\vec{x}) \mathbin{\&} w = \rho_{k+1}(\vec{x})]. \end{aligned}$$

*In particular, every rational linear form defines a linear partial function, and every linear partial function agrees with a rational linear form on its domain.*

Proof. By adding the fractions, we can re-write a rational linear form as

$$\rho(\vec{x}) = \frac{p_0 + p_1x_1 + \cdots + p_nx_n}{q} \quad (q > 0, p_1, \ldots, p_n \in \mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}),$$

so that it is defined by the term

$$\frac{1}{q} \cdot (p_0 + p_1x_1 + \cdots + p_nx_n),$$

and if $f(\vec{x})$ satisfies (18), then it is defined by the term

$$\rho_{k+1}(\vec{x}) + 0 \cdot (\rho_1(\vec{x})) + \cdots + 0 \cdot (\rho_k(\vec{x})).$$

The converse is proved by an easy induction on the construction of terms. ⊣

Equation (16) gives an example of a partial function which is linear but not defined by a rational linear form.

3.2. **$\Psi$-semilinear sets.** A set $X \subseteq N^n$ is $\Psi$-*semilinear* if it is the domain of convergence of a $\Psi$-linear partial function,

$$\vec{x} \in X \iff f(\vec{x})\downarrow .$$

**Lemma 9.** (a) *The intersection of two $\Psi$-semilinear sets is semilinear.*

(b) *The restriction of a $\Psi$-linear partial function to a $\Psi$-semilinear set is $\Psi$-linear.*

(c) *If $f$ is a $\Psi$-linear partial function and $k$ is any natural number, then the set $\{\vec{x} \mid f(\vec{x}) = k\}$ is $\Psi$-semilinear.*

(d) *A set $X \subseteq \mathbb{N}^n$ is semilinear ($\Psi = \emptyset$) if an only if for suitable rational, linear forms,*

$$\vec{x} \in X \iff (\exists u_1, \ldots, u_k \in \mathbb{N})[u_1 = \rho_1(\vec{x}) \,\&\, \cdots \,\&\, u_k = \rho_k(\vec{x})].$$

Proof. (a) If $\vec{x} \in X \iff f(\vec{x})\downarrow$ and $\vec{x} \in Y \iff g(\vec{x})\downarrow$, then

$$\vec{x} \in X \cap Y \iff \Big(f(\vec{x}) + g(\vec{x})\Big)\downarrow .$$

(b) If $\vec{x} \in X \iff f(\vec{x})\downarrow$, then the restriction of $g(\vec{x})$ to $X$ is defined by the term $g(\vec{x}) + 0 \cdot f(\vec{x})$.

(c) $f(\vec{x}) = k \iff \Big((k - f(\vec{x})) + (f(\vec{x}) - k)\Big)\downarrow$.

(d) follows immediately from Lemma 8. ⊣

3.3. **Piecewise $\Psi$-linear partial functions.** A partial function is *piecewise $\Psi$-linear* if it is the "union" of a finite number of $\Psi$-linear partial functions with pairwise disjoint domains. We indicate this by writing

$$f(\vec{x}) = f_1(\vec{x}) \uplus \cdots \uplus f_\ell(\vec{x})$$

where each $f_i(\vec{x})$ is a $\Psi$-linear, somewhere defined function, and conventionally including the nowhere defined partial function in the case $\ell = 0$; thus $f(\vec{x})$ is $\Psi$-linear if it has exactly one *linear part* in some decomposition. The list above shows that the total functions $\chi_<(x, y)$, $x \dot{-} y$, $x/2$ and Parity$(x)$ are piecewise linear (in the empty $\Psi$). The characteristic function of equality $\chi_=(x, y)$ is also piecewise linear, by combining the three (linear) cases of $x = y$, $x < y$ and $x > y$, and (by Lemma 8) it is not linear.

**Lemma 10.** *The composition of piecewise $\Psi$-linear partial functions is piecewise $\Psi$-linear,* and hence, if $\Psi$ is a set of (total) piecewise $\Psi$-linear functions, then every function defined by an $\Psi$-linear term is piecewise $\Psi$-linear.

Proof. If, for example, $f(\vec{x}) = f_1(\vec{x}) \uplus f_2(\vec{x})$ and $g(\vec{x}) = g_1(\vec{x}) \uplus g_2(\vec{x})$, then

$$f(g(\vec{x})) = f_1(g_1(\vec{x})) \uplus f_1(g_2(\vec{x})) \uplus f_2(g_1(\vec{x})) \uplus f_2(g_2(\vec{x})).$$

We get a *proper* representation of the composition $f(g(\vec{x}))$ by deleting from this disjoint union the "empty" (nowhere defined) linear parts. ⊣

**Lemma 11.** *A piecewise piecewise $\Psi$-linear partial function is piecewise $\Psi$-linear.*

Proof. The assertion is that if $f$ is the disjoint finite union of piecewise $\Psi$-linear partial functions on disjoint domains, then it is piecewise $\Psi$-linear; and the proof is obvious. ⊣

## 4. The basic construction

With these definitions, we can use the construction in the proof of Theorem 6 in Section 2 to establish a similar result for **prd**'s from piecewise $\Psi$-linear (total) functions.

**Proposition 12.** *For each* **prd** *from a set* $\Phi$ *of piecewise* $\Psi$*-linear functions which defines a function* $f(\vec{x})$ *of* $n$ *variables, there is a sequence*

$$\phi_1, \dots, \phi_k, \psi_1, \dots, \psi_l$$

*of* $\Psi$*-linear partial functions with the following properties:*

(1) *The domains of the* $\phi_i$*'s and the* $\psi_j$*'s form a partition of* $\mathbb{N}^n$*, i.e., they are pairwise disjoint and their union is* $\mathbb{N}^n$*.*
(2) *If* $\phi_i(\vec{x})\downarrow$*, then* $f(\vec{x}) = \phi_i(\vec{x})$*.*
(3) *Each* $\psi_j$ *is unbounded, i.e.,* $\sup\{\psi_j(\vec{x}) \mid \psi_j(\vec{x})\downarrow\} = \infty$*.*
(4) *If* $\psi_j(\vec{x})\downarrow$*, then* $c(E_f, \vec{x}) \geq \psi_j(\vec{x})$*.*

*It follows that if* $f(\vec{x})$ *is not piecewise* $\Psi$*-linear, then there exists an unbounded* $\Psi$*-linear partial function* $\psi(\vec{x})$ *such that*

$$\psi(\vec{x})\downarrow \implies c(E_f, \vec{x}) \geq \psi(\vec{x}).$$

Proof. Here we need to deal with the fact that the piecewise $\Psi$-linear functions defined by $\Phi$-terms may be bounded without being constant, e.g., $5 \dot{-} x$. This complicates the definition of the crucial one-step-reduction procedure, which now splits into cases and, when iterated, yields a "decision tree" of sorts.

We fix a **prd** $\Gamma$, as before, which defines $f$. If $E$ is a $V_\Gamma$-term and $X$ is a $\Psi$-semilinear set, we say that $E$ *is* $X$*-reducible*, if $E$ has a critical subterm

$$M \equiv f(G, \vec{N}) \tag{19}$$

such that *either* the restriction of $d(G, \vec{x})$ to $X$ is not $\Psi$-linear, *or* it is $\Psi$-linear and bounded. Since the restriction of $d(G, \vec{x})$ to $X$ is always piecewise $\Psi$-linear, this means that either every decomposition of it has at least two linear parts, or there is a $\Psi$-linear function $\widetilde{G}(\vec{x})$ with domain $X$ and a number $k$, such that

$$\vec{x} \in X \Longrightarrow d(G, \vec{x}) = \widetilde{G}(\vec{x}) \leq k.$$

If $E$ is not $X$-reducible, then it is $X$*-irreducible.*

A *term tree* is a structure of the form

$$(T, \{X_u\}_{u \in T}, \{E_u\}_{u \in T}),$$

with the following properties:

1. $T$ is a finite (rooted) tree.
2. For each node $u \in T$, $X_u$ is a $\Psi$-semilinear set and $E_u$ is a $V_\Gamma$-term.
3. If $v$ is a child of $u$ in $T$, then
$$\vec{x} \in X_v \Longrightarrow d(E_u, \vec{x}) = d(E_v, \vec{x}) \ \text{ and } c(E_u, \vec{x}) \geq c(E_v, \vec{x}).$$
4. For each node $u \in T$, the family $\{X_v \mid v \text{ is a child of } u\}$ is a partition of $X_u$, i.e.,
$$v, v' \text{ distinct children of } u \Longrightarrow X_v \cap X_{v'} = \emptyset,$$
$$X_u = \bigcup \{X_v \mid v \text{ is a child of } u\}.$$

A term tree is *reducible* if there is some leaf $u$ such that the term $E_u$ is $X_u$-reducible.

Consider the following one-step reduction procedure which assigns to each reducible term tree $T$ another term tree $T'$, an extension of $T$ by the addition of new nodes below some leaf of $T$:

1. Choose a leaf $u$ such that $E_u$ is $X_u$-reducible, so that it has a critical subterm of the form $f(G, \vec{N})$; choose a critical subterm of $E_u$ of this kind; and let

$$\widetilde{G}(\vec{x}) = \widetilde{G}_1(\vec{x}) \uplus \cdots \uplus \widetilde{G}_\ell(\vec{x}) \tag{20}$$

   be the restriction of $d(G, \vec{x})$ to $X_u$ decomposed into (non-empty) $\Psi$-linear partial functions with the least, possible $\ell$.

2. If $\ell > 1$ in (20), introduce new nodes $u(1), \dots, u(\ell)$ below $u$ tagged with

$$X_{u(i)} = X_u \cap \{\vec{x} \mid \widetilde{G}_i(\vec{x})\downarrow\}, \quad E_{u(i)} \equiv E_u.$$

3. Otherwise, $\widetilde{G}(\vec{x})$ is $\Psi$-linear and

$$\sup\{\widetilde{G}(\vec{x}) \mid \vec{x} \in X_u\} = k$$

   for some $k \in \mathbb{N}$. Suppose $f$ is defined from $g$ and $h$ by (2); define the terms $M_i$ for $i \leq k$ by (13) and (14); and for each $i \leq k$, let

$$X_{u,i} = X_u \cap \{\vec{x} \mid \widetilde{G}(\vec{x}) = i\}.$$

   These sets are $\Psi$-semilinear by Lemma 9, they are pairwise disjoint, and their union is $X_u$. We introduce new nodes $u(0), \dots, u(k)$ tagged with $X_{u,i}$ and the term $E_{u,i}$ obtained by replacing $f(G, \vec{N})$ in $E_u$ by $M_i$.

Proof that $T'$ is a term tree is exactly as in the proof of Theorem 6.

Now, given a term $E$, start with the one-node tree $T$ with $\mathbb{N}^n$ and $E$ assigned to the root, and apply repeatedly this one-step reduction procedure, as long as it is possible.

*Lemma. The iteration cannot go on indefinitely.*

*Proof.* If it did, then it would produce a finitely splitting, infinite tree, which would then have an infinite branch, by König's Lemma. Suppose this branch is

$$(X_1, E_1) > (X_2, E_2) > \cdots$$

If all the reductions after some stage $n$ along this branch were by Case 2, then $E_{n+m} \equiv E_n$ for all $m$, and so the reductions are all triggered by subterms $f(G, \vec{N})$ of the same $E_n$; but this is not possible, since there are only finitely many subterms of $E_n$, and each of them can justify at most one reduction by Case 2. Thus there must be infinitely many reductions by Case 3 of the procedure along the branch, and each of them lowers the rank of the term, as in the proof of Theorem 6, yielding an absurd infinite, decreasing sequence of ranks. ⊣ (Lemma)

So eventually we stop at a term tree $T^*$ such that (as in the proof of Theorem 6): *For each leaf* $u$, $E_u$ is $X_u$-irreducible and

$$\vec{x} \in X_u \Longrightarrow d(E, \vec{x}) = d(E_u, \vec{x}) \text{ and } c(E, \vec{x}) \geq c(E_u, \vec{x}).$$

In addition, by the construction,

$$\mathbb{N}^n = \bigcup\{X_u \mid u \text{ is a leaf of } T^*\}.$$

The leaves of $T^*$ are divided into two groups:

(1) $E_u$ is $X_u$-irreducible because it has no critical subterm, and so it defines a $\Psi$-linear partial function function on $X_u$. We let

$$\phi_1, \dots, \phi_k$$

be an enumeration of these partial functions.

(2) $E_u$ contains a critical subterm $f(G, \vec{n})$ such that the restriction of $d(G, \vec{x})$ to $X_u$ is $\Psi$-linear and

$$\sup\{d(G, \vec{x}) \mid \vec{x} \in X_u\} = \infty;$$

now $c(E, \vec{x}) \geq c(E_u, \vec{x}) \geq d(G, \vec{x})$ as in the proof of Theorem 6, and we can complete the proof of the Proposition by letting

$$\psi_1, \dots, \psi_l$$

be an enumeration of these $\Psi$-linear partial functions. ⊣

## 5. Conditionals and the essential complexity measure

An oft-cited problem with call-by-value recursion is that it is very inefficient for "dummy" recursive definitions like that of the predecessor function

$$\mathrm{Pd}(0) = 0, \quad \mathrm{Pd}(x+1) = x;$$

this is formally expressed in a **prd** as

$$\mathrm{Pd}(0) = 0, \quad \mathrm{Pd}(x+1) = h(\mathrm{Pd}(x), x), \text{ with } h(w, x) = P^2_2(w, x) = x,$$

and, clearly, $c(E_{\mathrm{Pd}}, x) = x$, which is wasteful. We introduce in this section extended derivations which allow definitions *by cases*, and then we use these to define a very robust notion of "essential" complexity for primitive recursive programs from suitably "rich" $\Phi$'s.

5.1. **Adding conditionals.** A *primitive recursive derivation with conditionals* or **prdc** is a sequence of functions

$$\Gamma = f_0, \dots, f_k$$

satisfying one of the conditions D1–D3 in Section 1 or the following

D4. $f_i$ is defined by *cases*

$$f(t, \vec{x}) = \text{if } (t = 0) \text{ then } g(\vec{x}) \text{ else } h(t, \vec{x}) \tag{21}$$

from functions $g$, $h$ which are listed earlier in the derivation. The corresponding clause in the iteration of $\Gamma$ is

R4. $f^{k+1}(t, \vec{x}) = \begin{cases} g^k(\vec{x}) & \text{if } t = 0, \\ h^k(t, \vec{x}) & \text{otherwise.} \end{cases}$

We treat symbols that are introduced by conditionals as "recursive symbols" and they are put in the vocabulary $V_\Gamma$, so that the $V_\Gamma$-term associated with $f$ is simply

T4. $E_f \equiv f(\mathsf{v}_0, \mathsf{v}_1, \mathsf{v}_2, \dots, \mathsf{v}_{1+n})$.

We also extend the complexity measure:

CT4. Suppose $f$ is defined by (21) and $E \equiv f(E_0, E_1, \dots, E_n)$. Set first

$$d^j = d(E_j, \pi), \quad c^j = c(E_j, \pi) \quad (j \leq n),$$

and put

$$c(E,\pi) = \begin{cases} \max\{c^0, c^1, \dots, c^n, c(E_g, d^1, \dots, d^n)\}, & \text{if } d^0 = 0, \\ \max\{c^0, c^1, \dots, c^n, c(E_h, d^0, d^1, \dots, d^n)\}, & \text{otherwise.} \end{cases}$$

Lemma 1 and Theorem 3 extend easily to **prdc**'s:

**Theorem 13.** *For each* **prdc** $\Gamma$*:*

(1) *For each symbol $f$, $c^s_f(\vec{x}) \geq c(E_f, \vec{x})$.*

(2) *For any $V_\Gamma$-terms $E$, $A_1, \dots, A_k$, if $\mathsf{u}_1, \dots, \mathsf{u}_k$ occur in $E$ and for $i = 1, \dots, k$,*

$$d_i = d(A_i, \pi), \quad c_i = c(A_i, \pi),$$

*then*

$$c(E\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) = \max\{c_1, \dots, c_k, c(E, \pi\{\mathsf{u}_1 := d_1, \dots, \mathsf{u}_k := d_k\})\}.$$

Proof. (1) is immediate from the definition, and the proof of (2) is by induction on $E$, with just one more case than the narrower Theorem 3.

*Case* CT4, $E \equiv f(E_0, E_1, \dots, E_n)$ and $f$ is defined by cases as in (21). We set

$$c^j = c(E_j\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) \quad (j \leq n),$$
$$d^j = d(E_j\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi), \quad \vec{\mathsf{u}} \equiv \mathsf{u}_1, \dots, \mathsf{u}_k, \ \vec{d} = (d^1, \dots, d^n),$$

and we consider two possibilities.

If $d^0 = 0$, then, by the definition, the induction hypothesis, and the fact that $d^0 = d(E_0, \pi\{\vec{\mathsf{u}} := \vec{d}\})$,

$$\begin{aligned} c(E\{\vec{\mathsf{u}} :\equiv \vec{A}\}, \pi) &= \max\{c^0, c^1, \dots, c^n, c(E_g, \vec{d})\} \\ &= \max\{c_1, \dots, c_k, \\ &\qquad c(E_0, \pi\{\vec{\mathsf{u}} := \vec{d}\}), \dots, c(E_n, \pi\{\vec{\mathsf{u}} := \vec{d}\}), \\ &\qquad c(E_g, \vec{d})\} \\ &= \max\{c_1, \dots, c_k, c(E, \pi\{\vec{\mathsf{u}} := \vec{d}\})\}. \end{aligned}$$

The computation for the case $d^0 > 0$ is almost identical. $\dashv$

**Lemma 14.** *For each* **prdc** $\Gamma$ *from a set* $\Phi$ *which contains the function* $1 \dot{-} y$*, there is a* **prd** $\mathrm{r}\Gamma$ *whose recursive symbols include all the recursive symbols of* $\Gamma$*, and such that for every* $V_\Gamma$*-term* $E$*,*

$$d_{\mathrm{r}\Gamma}(E, \pi) = d_\Gamma(E, \pi), \quad c_{\mathrm{r}\Gamma}(E, \pi) \leq c_\Gamma(E, \pi).$$

Proof.[3] We will eliminate the definitions by cases in $\Gamma$ one-at-a-time, by iterating the following reduction procedure on **prdc**'s:

**Sublemma 14.1.** *Suppose* $\Phi$ *contains arithmetic subtraction* $1 \dot{-} y$ *and*

$$\Delta = f_0, \dots, f_k, f$$

*is a* **prdc** *from* $\Phi$*, in which the last function is the only one defined by cases,*

$$f(t, \vec{x}) = \text{if } (t = 0) \text{ then } g(\vec{x}) \text{ else } h(t, \vec{x});$$

*then there is a* **prd**

$$\Delta' = f_0, \dots, f_k, \phi, \phi_g, \phi_h, f$$

[3]This clever trick is most often used to show that *the class of partial functions primitive recursive in given partial functions is closed under definition by cases*, and it is due to Bird [2]; I learned it from Ivan Soskov.

*with complexity measure $c'$, such that for all $V_\Delta$-terms $E$,*

$$d'(E,\pi) = d(E,\pi), \quad c'(E,\pi) \le c(E,\pi). \tag{22}$$

Proof. We set

$$\begin{aligned} \phi(0,u,v) &= v, & \phi(t+1,u,v) &= u,\\ \phi_g(0,\vec{x}) &= 0, & \phi_g(t+1,\vec{x}) &= g(\vec{x}),\\ \phi_h(0,s,\vec{x}) &= 0, & \phi_h(t+1,s,\vec{x}) &= h(s,\vec{x}),\end{aligned}$$
$$f(t,\vec{x}) = \phi(1 \dot- t, \phi_g(1 \dot- t,\vec{x}), \phi_h((1 \dot- (1 \dot- t)), t, \vec{x})),$$

where, of course, the primitive recursive definitions for the new three functions and the composition defining $f$ must be "spelled out" correctly. To check first that the new definition for $f$ gives the same function as that in $\Delta$, compute with the $\Delta'$ definition of $f$:

$$\begin{aligned} f(0,\vec{x},\vec{y}) &= \phi(1,\phi_g(1,\vec{x}),\phi_h(0,0,\vec{x}))\\ &= \phi(1,g(\vec{x}),0) = g(\vec{x}),\\ f(t+1,\vec{x},\vec{y}) &= \phi(0,\phi_g(0,\vec{x}),\phi_h(1,t+1,\vec{x}))\\ &= \phi(0,0,h(t+1,\vec{x})) = h(t+1,\vec{x}).\end{aligned}$$

For the corresponding complexity values, we compute:

$$\begin{aligned} c'(E_{\phi_g},0,\vec{x}) &= 0,\\ c'(E_{\phi_g},1,\vec{x}) &= \max\{c'(E_{\phi_g},0,\vec{x})+1, c(E_g,\vec{x})\}\\ &\le c(E_g,\vec{x})+1\\ c'(E_{\phi_h},0,s,\vec{x}) &= 0,\\ c'(E_{\phi_h},1,s,\vec{x}) &= \max\{c'(E_{\phi_h},0,s,\vec{x})+1, c(E_h,s,\vec{x})\}\\ &\le c(E_h,s,\vec{x})+1\\ c'(E_\phi,0,u,v) &= 0\\ c'(E_\phi,1,u,v) &= 1,\\ c'(E_f,0,\vec{x}) &= \max\{c'(E_{\phi_g},1,\vec{x}), c'(E_{\phi_h},0,0,\vec{x}),\\ &\qquad c'(E_\phi,\phi_g(1,\vec{x}),\phi_h(0,0,\vec{x})\}\\ &\le c(E_g,\vec{x})+1\\ c'(E_f,t+1,\vec{x}) &= \max\{c'(E_{\phi_g},0,\vec{x}), c'(E_{\phi_h},t+1,\vec{x}),\\ &\qquad c'(E_\phi,\phi_g(0,\vec{x}),\phi_h(t+1,\vec{x})\}\\ &\le c(E_h,t+1,\vec{x})+1.\end{aligned}$$

From this we get immediately the required inequality (22). ⊣ (14.1)

To prove the lemma from this, set

$$\Gamma_0 = \Gamma, \Gamma_1 = \Gamma', \ldots, \Gamma_{t+1} = \Gamma'_t,$$

where $\Gamma'_i$ is constructed by 14.1 with respect to the first definition by cases in $\Gamma_i$, if one such exists. In the end we obtain a **prd** r$\Gamma = \Gamma_t$ with the required properties. ⊣

5.2. **The essential complexity.** A primitive recursive definition

$$\left\{\begin{array}{rcl} f(0,\vec{x}) & = & g(\vec{x}) \\ f(y+1,\vec{x}) & = & h(f(y,\vec{x}),y,\vec{x}) \end{array}\right. \tag{23}$$

in a **prdc** $\Delta = f_0, \dots, f_k$ is *fake* if the variable $\mathsf{v}_0$ does not occur in the term $E_h$, so that (by the definitions n Subsection 1.3) the function $h(u,y,\vec{x})$ does not depend on $u$. A *genuine* primitive recursive definition is one which is not fake, and a genuine **prdc** is one with no fake primitive recursive definitions.

Suppose $f_i$ is the first function in $\Delta$ defined by a fake primitive recursion, and let $\Delta'$ be constructed by replacing this fake recursion by the definition by cases

$$f(t,\vec{x}) = \text{if } (t=0) \text{ then } g(\vec{x}) \text{ else } h(t, t \dot{-} 1, \vec{x}); \tag{24}$$

formally this requires the introduction of two new symbols which are defined from $h(w,t,\vec{x})$ by composition (with projections), but this does not introduce recursive symbols, and so $V_\Delta = V_{\Delta'}$.

For any **prdc** $\Gamma$, set

$$\Gamma_0 = \Gamma, \Gamma_1 = \Gamma', \dots, \Gamma_{i+1} = \Gamma'_i \dots$$

and let

$$\Gamma^* = \Gamma_t, \text{ for the least } t \text{ such that } \Gamma_t \text{ is genuine,} \tag{25}$$

$$\begin{aligned} c^*(E,\pi) &= c^*_\Gamma(E,\pi) = c_{\Gamma^*}(E,\pi) \\ &= \text{the complexity of } E \text{ computed in } \Gamma^*. \end{aligned} \tag{26}$$

**Lemma 15.** *If $\Gamma$ is a* **prdc** *from a set of functions $\Phi$ which includes the predecessor $x \dot{-} 1$ and the function $1 \dot{-} y$, then $\Gamma^*$ has the following properties.*

(a) *$V_\Gamma = V_{\Gamma^*}$, i.e., $\Gamma$ and $\Gamma^*$ have the same recursive symbols.*
(b) *For each $V_\Gamma$-term $E$, $d_\Gamma(E,\pi) = d_{\Gamma^*}(E,\pi)$.*
(c) *For each $V_\Gamma$-term $E$, $c(E,\pi) \geq c^*(E,\pi)$.*
(d) *Every primitive recursion in $\Gamma^*$ is genuine.*

Proof. (a) and (d) are immediate from the definition. To verify (b) and (c), it is enough to show that the transformation $\Delta \mapsto \Delta'$ preserves denotations and does not increase complexities.

For (b), notice that if the variable $\mathsf{v}_0$ does not occur in the term $E_h$, then the value $h(w,t,\vec{x})$ is independent of $w$, because for any $w, w'$,

$$h(w,t,\vec{x}) = d(E_h, w, t, \vec{x}) = d(E_h, w', t, \vec{x}) = h(w', t, \vec{x}).$$

It follows that if $f$ is defined by a fake recursion (23) in $\Delta$, then

$$\begin{aligned} f(0,\vec{x}) &= g(\vec{x}), \\ f(t+1,\vec{x}) &= h(f(t,\vec{x}),t,\vec{x}) \\ &= h(t+1,t,\vec{x}), \end{aligned}$$

and so $f(t,\vec{x})$ satisfies its new definition (24) in $\Delta'$.

For (c), we check directly from the definitions that

$$c_\Delta(E_f,t,\vec{x}) \geq c_{\Delta'}(E_f,t,\vec{x}),$$

and then we apply (2) of Theorem 13. ⊣

The next result improves Proposition 12 in two ways, by allowing cases in the derivations, and by replacing the complexity $c(E_f,\vec{x})$ by the generally smaller $c^*(E_f,\vec{x})$. It is the main combinatorial result of this article.

**The Basic Lemma 16.** *For each* **prdc** *from a set* $\Phi$ *of piecewise* $\Psi$*-linear functions which defines a function* $f(\vec{x})$ *of* $n$ *variables, there is a sequence*

$$\phi_1, \dots, \phi_k, \psi_1, \dots, \psi_l$$

*of* $\Psi$*-linear partial functions with the following properties:*

(1) *The domains of the* $\phi_i$*'s and the* $\psi_j$*'s form a partition of* $\mathbb{N}^n$*, i.e., they are pairwise disjoint and their union is* $\mathbb{N}^n$*.*
(2) *If* $\phi_i(\vec{x})\downarrow$*, then* $f(\vec{x}) = \phi_i(\vec{x})$*.*
(3) *Each* $\psi_j$ *is unbounded, i.e.,* $\sup\{\psi_j(\vec{x}) \mid \psi_j(\vec{x})\downarrow\} = \infty$*.*
(4) *If* $\psi_j(\vec{x})\downarrow$*, then* $c^*(E_f, \vec{x}) \geq \psi_j(\vec{x})$*.*

*It follows that if* $f(\vec{x})$ *is not piecewise* $\Psi$*-linear, then there exists an unbounded* $\Psi$*-linear partial function* $\psi(\vec{x})$ *such that*

$$\psi(\vec{x})\downarrow \implies c^*(E_f, \vec{x}) \geq \psi(\vec{x}).$$

Proof. By its definition, for any $V_\Gamma$-term $E$,

$$c^*(E_f, \vec{x}) = c_{\Gamma^*}(E_f, \vec{x}),$$

where $\Gamma^*$ is some **prdc**; and then by Lemma 14, there is a **prd** $\mathrm{r}\Gamma^*$ such that

$$c_{\Gamma^*}(E_f, \vec{x}) \geq c_{\mathrm{r}\Gamma^*}(E, \vec{x}).$$

Moreover, these transformations preserve denotations, i.e.,

$$f(\vec{x}) = d_\Gamma(E_f, \vec{x}) = d_{\Gamma^*}(E_f, \vec{x}) = d_{\mathrm{r}\Gamma^*}(E_f, \vec{x}).$$

The result follows by applying the Proposition 12 to $\mathrm{r}\Gamma^*$ and using the inequalities above. ⊣

## 6. The non-strict complexity

The non-strict[4] iterates $\tilde{f}^k$ of a derivation $\Gamma$ are defined by the same clauses R1–R4 that are used for the call-by-value iteration, except—and this is a big difference—that they are monotone (not necessarily strict) functions on $(\mathbb{N} \cup \{\perp\})^n$ to $\mathbb{N} \cup \{\perp\}$. We repeat the clauses here for easy reference, using variables $\zeta$, $\xi$ over $\mathbb{N} \cup \{\perp\}$:

RN1. If $f$ is a given function $\phi(\vec{\zeta})$, then, for every $k$,

$$\tilde{f}^k(\vec{\zeta}) = \phi(\vec{\zeta}).$$

If, for example, $f(\zeta_1, \zeta_2) = P^2_2(\zeta_1, \zeta_2)$, then $\tilde{f}^k(\perp, 3) = 3$.

RN2. If $f$ is defined by composition as in (1), then

$$\tilde{f}^0(\vec{\zeta}) = \perp, \quad \tilde{f}^{k+1}(\vec{\zeta}) = \tilde{g}^k(\tilde{h}^k_1(\vec{\zeta}), \dots, \tilde{h}^k_m(\vec{\zeta})).$$

[4]In the study of recursive definitions relative to arbitrary given functions, this is sometimes referred-to as "call-by-name" iteration. I have avoided this terminology here, since the related papers by Colson, David etc., use the term "call-by-name" to refer to iteration of (absolute) primitive recursive algorithms on the complete poset of *lazy integers* rather than the flat poset $\mathbb{N} \cup \{\perp\}$ of *partial integers*. Notice that there is no natural interpretation of iteration on the poset of lazy integers in the presence of arbitrary given functions.

RN3. If $f$ is defined by primitive recursion as in (2), then

$$\begin{aligned}\tilde{f}^0(\xi,\vec{\zeta}) &= \perp,\\ \tilde{f}^k(\perp,\vec{\zeta}) &= \perp,\\ \tilde{f}^{k+1}(0,\vec{\zeta}) &= \tilde{g}^k(\vec{\zeta}),\\ \tilde{f}^{k+1}(y+1,\vec{\zeta}) &= \tilde{h}^k(\tilde{f}^k(y,\vec{\zeta}),y,\vec{\zeta}).\end{aligned}$$

RN4. If $f$ is defined by cases as in (21), then

$$\begin{aligned}\tilde{f}^k(\perp,\vec{\zeta},\vec{\xi}) &= \perp,\\ \tilde{f}^{k+1}(t,\vec{\zeta},\vec{\xi}) &= \begin{cases}\tilde{g}^k(\vec{\zeta}) & \text{if } t=0,\\ \tilde{h}^k(\vec{\xi}) & \text{otherwise.}\end{cases}\end{aligned}$$

The non-strict complexity measure is defined on natural number arguments only, by

$$c^n_f(\vec{x}) = \text{the least } k \text{ such that } \tilde{f}^k(\vec{x})\downarrow . \tag{27}$$

**Theorem 17.** *For each* **prdc** $\Gamma$*, and each function symbol* $f$*,*

$$c^s_f(\vec{x}) \geq c^n_f(\vec{x}) \geq c^*(E_f,\vec{x}).$$

Proof. For the (well-known) first inequality, we need to show that for every $k$,

$$f^k(\vec{x})\downarrow \implies \tilde{f}^k(\vec{x})\downarrow \qquad (\vec{x}\in\mathbb{N}),$$

and this is almost immediate, by induction on $k$.

For the second inequality, it is enough to prove that for every $k$,

$$\tilde{f}^k(\vec{\zeta})\downarrow \implies d(E_f,\vec{\zeta}) = \tilde{f}^k(\vec{\zeta})\,\&\, c^*(E_f,\vec{\zeta}) = c_{\Gamma^*}(E_f,\vec{\zeta}) \leq k. \tag{28}$$

We show this by induction on $k$, following the definition of $\Gamma^*$ and noting that

$$[d(E_f,\vec{\zeta})\downarrow \ \&\, \zeta_i = \perp] \implies \text{ the variable } \mathsf{v}_{i-1} \text{ does not occur in } E_f,$$

simply because terms are evaluated strictly.

In the basis $k=0$, (28) is easy to verify, taking cases on whether $f$ is the successor $S$ or in $\Phi$—and then it is evaluated strictly—or a projection $P^n_i$ or a constant $C^n_q$, when only the relevant variable occurs in $E_f$.

In the induction step, we consider three cases.

*Case* T2. $f$ is defined by composition, so

$$\tilde{f}^{k+1}(\vec{\zeta}) = \tilde{g}^k(u_1,\dots,u_m),$$

where $u_i = \tilde{h}^k_i(\vec{\zeta})$ for $i=1,\dots,m$. By the induction hypothesis,

$$d(E_g,u_1,\dots,u_m) = \tilde{g}^k(u_1,\dots,u_m),$$

and so if $u_i=\perp$, then the variable $\mathsf{v}_{i-1}$ does not occur in $E_g$; for each $i$ then such that $\mathsf{v}_{i-1}$ occurs in $E_f$, it must be that

$$u_i = \tilde{h}^k_i(\vec{\zeta}) \neq \perp,$$

and so the induction hypothesis gives again that $d(E_{h_i},\vec{\zeta})\downarrow$. This implies that

$$\begin{aligned}d(E_f,\vec{\zeta}) &= d(E_g\{\mathsf{u}_1 :\equiv E_{h_1},\dots,\mathsf{u}_m :\equiv E_{h_m}\},\vec{\zeta})\\ &= d(E_g,\vec{u})\downarrow .\end{aligned}$$

The complexity inequality follows by a direct application of Theorem 13, keeping track of the variables which actually occur in $E_g$.

*Case* T3. $f$ is defined by primitive recursion. We prove (28) by induction on $y \in \mathbb{N}$, since $\tilde{f}^{k+1}(\perp, \vec{\zeta}) = \perp$, by definition.

Basis, $y = 0$. Now $\tilde{f}^k(0, \vec{\zeta}) = \tilde{g}^k(\vec{\zeta})$, and the argument is exactly as in CT2.

Induction Step, $y = z + 1$. The hypothesis is that

$$\tilde{f}^{k+1}(z+1, \vec{\zeta}) = \tilde{h}^k(\tilde{f}^k(z, \vec{\zeta}), z, \vec{\zeta})\downarrow,$$

and so, by induction hypothesis,

$$d(E_h, \tilde{f}^k(z, \vec{\zeta}), z, \vec{\zeta})\downarrow \ . \tag{29}$$

There are two possibilities

*Case* 1. $\tilde{f}^k(z, \vec{\zeta}) = \perp$. Now (29) implies that the variable $\mathsf{v}_0$ does not occur in $E_h$, and so the primitive recursion defining $f$ is not genuine; this means that in $\Gamma^*$ it is replaced by the definition by cases

$$f(t, \vec{x}) = \text{if } (t = 0) \text{ then } g(\vec{x}) \text{ else } h(t, t \dot{-} 1, \vec{x}),$$

so that

$$\begin{aligned} d_{\Gamma^*}(E_f, z+1, \vec{\zeta}) &= d(E_h, z+1, z, \vec{\zeta}) \\ &= d(E_h, \tilde{f}^k(z, \vec{\zeta}), z, \vec{\zeta}) \\ &= \tilde{f}^{k+1}(z+1, \vec{\zeta}), \end{aligned}$$

using again the fact that $\mathsf{v}_0$ does not occur in $E_h$. Moreover,

$$c^*(E_h, z+1, \vec{\zeta}) = c_{\Gamma^*}(E_h, z+1, z, \vec{\zeta}) \leq k,$$

as required.

*Case* 2. $\tilde{f}^k(z, \vec{\zeta}) = w \in \mathbb{N}$. It may still be the case that the primitive recursion defining $f$ in $\Gamma$ is not genuine, in which case the argument is exactly like that in Case 1. If it is genuine, then $f$ is defined in $\Gamma^*$ by the same primitive recursion, and then the induction hypothesis gives the required result.

Definition by cases T4 is handled exactly like composition. ⊣

## 7. The main theorem

The key to applying the Basic Lemma 16 is the next, simple extension of *Dixon's Lemma*,[5] where for $\vec{x} = (x_1, \dots, x_n), \vec{y} = (y_1, \dots, y_n) \in \mathbb{N}^n$,

$$\vec{x} \leq \vec{y} \iff x_1 \leq y_1 \,\&\, \dots \,\&\, x_n \leq y_n.$$

**Lemma 18.** *If a set $K \subseteq \mathbb{N}^{n+1}$ is unbounded on its last coordinate, i.e.,*

$$\sup\{w \mid (\vec{x}, w) \in K\} = \infty,$$

*then it contains two points $(\vec{x}_1, w_1)$, $(\vec{x}_2, w_2)$ such that*

$$\vec{x}_1 \leq \vec{x}_2 \text{ and } w_1 < w_2. \tag{30}$$

[5] *Every set $S$ of pairwise undominated points in $\mathbb{N}^n$ is finite*, see p. 184 of [1]. This was rediscovered by Koutsoupias for the present application, and then van den Dries pointed me to its exposition in [1], where it is referred to as "the most frequently rediscovered mathematical theorem".

PROOF is by induction on $n$, the result being immediate for $n = 0$. For the induction step, fix some $(a_1, \dots, a_n, z) \in K$ and suppose the lemma fails. This implies that for every $(\vec{x}, w) \in K$,

$$\text{either } w \le z \text{ or, for some } i, x_i < a_i.$$

Since the part of $K$ with $w \le z$ is certainly not unbounded on the last coordinate and the second disjunct above splits $K$ into finitely many parts, there must exist some $i$ and some $k < a_i$ such that the set

$$\{(\vec{x}, w) \in K \mid x_i = k\}$$

is unbounded in the last coordinate; but this is a set in dimension $n$, and so it contains two points satisfying (30), by the induction hypothesis, contradicting our assumption. ⊣

**Main Theorem 19.** *For every* **prdc** *from piecewise linear* (total) *functions* $\Phi$ *which defines a function* $f(\vec{x})$ *of* $n$ *variables,* ***either*** $f$ *is piecewise linear,* ***or*** *there exist vectors* $\vec{x}_1$ *and* $\vec{d} \neq 0$ *in* $\mathbb{N}^n$*, such that for all* $t$*,*

$$c^s_f(\vec{x}_1 + t\vec{d}) \ge c^n_f(\vec{x}_1 + t\vec{d}) \ge c^*(E_f, \vec{x}_1 + t\vec{d}) \ge t.$$

*If the second alternative holds, then there is an* $s$*, such that with* $\vec{x}^t = \vec{x}_1 + (s+t)\vec{d}$*,*

$$\lim_{t\to\infty} (x^t_1 + \cdots + x^t_n) = \infty,$$

*and for a suitable, positive rational* $r$ *and all* $t$*,*

$$c^*(E_f, \vec{x}^t) \ge r(x^t_1 + \cdots + x^t_n).$$

PROOF. Suppose $f(\vec{x})$ is not piecewise linear, and let $\psi(\vec{x})$ be the linear partial function guaranteed by the Basic Lemma 16, which is unbounded and provides a lower bound for $c^*(E_f, \vec{x})$. By Lemma 8,

$$\begin{aligned} &\psi(\vec{x}) = w \\ &\qquad \iff (\exists u_1, \dots, u_k \in \mathbb{N})[u_1 = \rho_1(\vec{x})\ \&\ \cdots\ \&\ u_k = \rho_k(\vec{x})\ \&\ w = \rho_{k+1}(\vec{x})] \end{aligned}$$

for suitable rational linear forms $\rho_i$, so that

$$\rho_i(\vec{x}) = \pi_i(\vec{x}) + c_i \quad (i = 1, \dots, k+1)$$

where each $c_i$ is a (possibly 0) rational constant, and each $\pi_i(\vec{x})$ is a homogeneous linear function, i.e.,

$$\pi_i(\lambda\vec{x}_1 + \mu\vec{x}_2) = \lambda\pi_i(\vec{x}_1) + \mu\pi_i(\vec{x}_2). \tag{31}$$

The set $K$ of all $(\vec{x}, \vec{u}, w)$ in $\mathbb{N}^{n+k+1}$ which satisfies the system of equations

$$\begin{aligned} u_i &= \pi_i(\vec{x}) + c_i \quad (i = 1, \dots, k) \\ w &= \pi_{k+1}(\vec{x}) + c_{k+1} \end{aligned}$$

is unbounded in the last coordinate, and so, by Lemma 18, there exist points $(\vec{x}_1, \vec{u}_1, w_1)$ and $(\vec{x}_2, \vec{u}_2, w_2)$ in $K$, such that

$$(\vec{x}_1, \vec{u}_1) \le (\vec{x}_2, \vec{u}_2) \text{ and } w_1 < w_2.$$

**Sublemma 19.1.** *For each natural number* $t$*, the vector* $\vec{x}_1 + t(\vec{x}_2 - \vec{x}_1)$ *is in the domain of* $\psi$*, and*

$$\psi(\vec{x}_1 + t(\vec{x}_2 - \vec{x}_1)) = w_1 + t(w_2 - w_1). \tag{32}$$

*Proof*. Using (31), we first get that for each $i \le k+1$,

$$\pi_i(\vec{x}_2 - \vec{x}_1) = \pi_i(\vec{x}_2) + c_i - (\pi_i(\vec{x}_1) + c_i) = u_{2,i} - u_{1,i} \ge 0$$

(in the obvious notation) is a natural number, and so

$$\pi_i(\vec{x}_1 + t(\vec{x}_2 - \vec{x}_1)) + c_i = \pi_i(\vec{x}_1) + c_i + t(\pi_i(\vec{x}_2) - \pi_i(\vec{x}_1))$$

is the sum of two natural numbers and so also a natural number. This means that the vector $\vec{x}_1 + t(\vec{x}_2 - \vec{x}_1)$ is in the domain of $\psi(\vec{x})$, and the case $i = k+1$ shows (32). $\dashv$ (19.1)

If we set

$$\vec{d} = \vec{x}_2 - \vec{x}_1, \quad c = w_2 - w_1 > 0,$$

then

$$c^*(E_f, \vec{x}_1 + t\vec{d}) \ge g(\vec{x}_1 + t\vec{d}) = w_1 + ct \ge t$$

which completes the proof of the first claim in the theorem.

For the second claim, let first

$$\vec{y}^t = \vec{x}_1 + t\vec{d},$$

add the component equations

$$\textstyle\sum_i y_i^t = \sum_i x_{1,i} + t\sum_i d_i,$$

and solve this for $t$ to get (with $d = \sum_i d_i$)

$$t = \frac{1}{d}(\textstyle\sum_i y_i^t - \sum_i x_{1,i});$$

now $\lim_t \sum_i y_i^t = \infty$, and so there is some $s$ such that

$$t \ge s \Longrightarrow \Big(1 - \frac{\sum_i x_{1,i}}{\sum_i y_i^t}\Big)\textstyle\sum_i y_i^t > \frac{1}{2}\sum_i y_i^t;$$

the required inequality holds then with this $s$ and $r = \frac{1}{2d}$. $\dashv$

**Corollary 20.** *The function*

$$\gcd(x,y) = \text{the greatest common divisor of } x \text{ and } y$$

*is not piecewise linear and so, for every* **prdc** *which computes it from piecewise linear givens with essential complexity* $c^*(x,y)$*, there is a sequence* $(x_t, y_t)$ *such that* $\lim_t(x_t, y_t) = \infty$*, and with some rational* $r > 0$*,*

$$c^*(x_t, y_t) \ge r(x_t + y_t).$$

Proof. If $\gcd(x,y)$ is piecewise linear, then, in particular, there exists rational constants $a_i, b_i, c_i$ $(i < m)$, such that for each $(x,y)$, there is an $i < m$ satisfying

$$\gcd(x,y) = a_i x + b_i y + c_i.$$

It follows that the same $i$ will be used for infinitely many pairs of the form $(x3^x, x2^x)$, i.e., for suitable rationals $a$, $b$, $c$, the equation

$$x = \gcd(x3^x, x2^x) = ax3^x + bx2^x + c$$

has infinitely many solutions, which cannot be true. (In detail, dividing by $x3^x$ and taking the limit as $x \to \infty$, we get that $a = 0$, and so the equation

$$x = bx2^x + c$$

must have infinitely many solutions; and then, dividing by $x2^x$ and taking limits again, we get that $b = 0$, and so $x = c$ for infinitely many values of $x$, which is absurd.) ⊣

This means that fast, logtime algorithms for the computation of $\mathrm{gcd}(x, y)$ cannot be expressed by just composition and primitive recursion using piecewise linear givens. The classical, Euclidean algorithm can be defined succintly for $x \geq y \geq 1$ by the recursive equation

$$\mathrm{gcd}(x,y) = \begin{cases} y, & \text{if } \mathrm{rm}(x,y)) = 0, \\ \mathrm{gcd}(y, \mathrm{rm}(x,y)), & \text{otherwise,} \end{cases}$$

from the remainder function

$$\mathrm{rm}(x,y) = \text{the unique } r < y \text{ such that for some } q, x = yq + r \tag{33}$$

which is not piecewise linear; its complexity is $O(\min(\log_2(x), \log_2(y)))$. Stein's algorithm, on the other hand, uses only the piecewise linear functions in the list in Section 3, and works in (somewhat greater) logarithmic time. We include this result for easy reference:

**Lemma 21** (Stein's algorithm, [9], Vol. 2, Sect. 4.5.2)**.** *The following recursion computes* $\mathrm{gcd}(x, y)$ *for* $x, y \neq 0$ *in* $O(\log_2(x) + \log_2(y))$ *stages:*

$$\mathrm{gcd}(x,y) = \begin{cases} x & \text{if } x = y \text{ (or } x = 0 \text{ or } y = 0) \\ 2\,\mathrm{gcd}(x/2, y/2) & \text{otherwise, if } \mathrm{Parity}(x) = \mathrm{Parity}(y) = 0, \\ \mathrm{gcd}(x/2, y) & \text{otherwise, if } \mathrm{Parity}(x) = 0, \mathrm{Parity}(y) = 1, \\ \mathrm{gcd}(x, y/2) & \text{otherwise, if } \mathrm{Parity}(x) = 1, \mathrm{Parity}(y) = 0, \\ \mathrm{gcd}(x \dot{-} y, y) & \text{otherwise, if } x > y, \\ \mathrm{gcd}(x, y \dot{-} x) & \text{otherwise.} \end{cases}$$

Proof. That the gcd satisfies these equations and is determined by them is trivial. To check the complexity, notice that (at worst) every other application of one of the clauses involves halving one of the arguments—the worst case being subtraction, which, however must then be immediately followed by a division, since the difference of two odd numbers is even. ⊣

## 8. Comments and open problems

One can view the Main Theorem 19 as a direct generalization of Colson's Theorem to "piecewise linear primitive recursion with conditionals", especially since the Corollary 20 about $\mathrm{gcd}(x, y)$ is so much in the spirit of Colson's Corollary about $\min(x, y)$. From another point of view, these complexity results are only peripherally related to the results of Colson and David, who are primarily concerned with the analysis of absolute, sequential, call-by-name primitive recursion to which the present paper is barely relevant.

There is a third point of view however (always!), by which the work of Colson, Fredholm, David and this paper are really very much about the same thing, and that is the study of classes of algorithms which are defined by certain forms of recursion. A good name for the topic might be the *structural complexity of algorithms*, in analogy with the term *structural complexity of queries* which has been used to distinguish the classification of queries (on strings or finite structures) by the difficulty of their definition rather than by the requirement of resources (time or space) for the algorithms which decide them. The main questions in structural complexity are, of course, about its relation with traditional time-and-space complexity, and this is what we do here also—but about classes of algorithms rather than classes of queries.

If we want to study special classes of algorithms, we should single them out from "general algorithms" in some form; but what is an algorithm? Colson and David never adopt some sort of rigorous definition, but they assume that primitive recursive algorithms can be faithfully modeled by the combinators which are naturally associated with primitive recursive derivations, acting on the domain of lazy integers. Fredholm assumes an underlying theory of *partial inductive definitions* due to Hallnäs [8], which models algorithms (basically) by systems of Herbrand-Gödel-Kleene equations relating strict partial functions. My own approach is not too different (in the end) from that of Hallnäs, but it is more general, and it is based on the deterministic systems of equations with strict conditionals introduced by McCarthy [10] rather than the HGK systems; for a recent paper on the topic (with references to earlier work) see [11].

The two most obvious problems left open by this paper are the following.

**Open Problem 1.** *True or false: for every primitive recursive derivation from piecewise linear functions and the remainder function* (33) *which defines a function* $f(\vec{x})$ *of* $n$ *variables,* ***either*** $f$ *is piecewise linear from the remainder,* ***or*** *there exists some rational number* $r > 0$ *such that*

$$\textit{for infinitely many } \vec{x}, c^*(\vec{x}) \geq r(x_1 + \cdots + x_n),$$

*where* $c^*(\vec{x})$ *is the* (essential) *complexity function of the derivation.*

A positive answer would show that the most ancient Euclidean algorithm cannot be matched in efficiency by a primitive recursive algorithm from its natural givens, and the proof would likely be interesting: the obvious attempts to extend the techniques of this paper to solve the problem lead to apparently difficult algebraic and number-theoretic questions.

**Open Problem 2.** *Are there examples of recursive algorithms* (with piecewise linear givens) *whose time complexity cannot be matched by a* $\mu$*-recursive algorithm computing the same function?*

By $\mu$*-recursion* we mean here the extension of primitive recursion by the scheme for minimalization, defined on partial functions by

$$\begin{aligned} f(\vec{x}) &= \mu y[g(\vec{x}, y) = 0] \\ &= \text{the least } y[(\forall i < y)(\exists z)[g(\vec{x}, i) = z + 1 \,\&\, g(\vec{x}, y) = 0] \end{aligned}$$

It would be nice to answer this question positively simply by extending the Main Theorem 19 to $\mu$-recursion, but this does not hold because of the following, quite easy result:

**Proposition 22.** *There is a* $\mu$*-recursive algorithm from piecewise linear givens, which computes the binary logarithm* $\log_2(x)$ *in* $O(\log_2(x))$ *steps.*

8.1. **Added in proof.** Lou van den Dries [5] has now obtained a positive answer to Open Problem 1 for the most important special case of $\gcd(x, y)$, he has answered the general problem negatively, and he has established a (weaker) optimal positive proposition , along with many stronger, related results on the complexity of the greatest common divisor function. He uses non-standard models of arithmetic.

## References

[1] T. Becker and V. Weispfenning. *Grobner bases: a compuational approach to commutative algebra*. Number 141 in Graduate texts in mathematics. Springer, 1993.

[2] R. Bird. A note on definition by cases. *Zeitschrift für Mathematisches Logik und Grundlagen der Mathematik*, 19:207–208, 1973.

[3] L. Colson. About primitive recursive algorithms. *Theoretical Computer Science*, 83:57–69, 1991.

[4] R. David. On the asymptotic behaviour of primitive recursive algorithms. *Theoretical Computer Science*. to appear.

[5] Lou van den Dries. Generating the greatest common divisor, and limitations of primitive recursive algorithms. To appear in *Foundations of Computational Mathematics*, `http://www.math.uiuc.edu/People/vddries.html`.

[6] Daniel Fredholm. *Intensional aspects of function definitions*. PhD thesis, Mathematics Institute, University of Stockholm, 1994.

[7] Daniel Fredholm. Intensional aspects of function definitions. *Theoretical Computer Science*, 163:1–66, 1995.

[8] L. Hallnäs. Logical and computational invariants of programs. In L. H. Eriksson et al., editor, *Extensions of logic programming*, number 596 in Lecture Notes in Artificial Intelligence, Berlin, 1991. Springer.

[9] D. E. Knuth. *The Art of Computer Programming. Fundamental Algorithms*, volume 1. Addison-Wesley, second edition, 1973.

[10] J. McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D Herschberg, editors, *Computer programming and formal systems*, pages 33–70. North-Holland, 1963.

[11] Yiannis N. Moschovakis. What is an algorithm? In B. Engquist and W. Schmid, editors, *Mathematics Unlimited – 2001 and Beyond*. Springer, 2001.

Department of Mathematics, University of California, Los Angeles, CA 90095-1555, USA

and Department of Mathematics, University of Athens, Athens, Greece

*E-mail address*: `ynm@math.ucla.edu`

*URL*: `www.math.ucla.edu/~ynm`

Version sent to the printer, posted March 4, 2003

To appear in Theoretical Computer Science