ABSOLUTE LOWER BOUNDS FOR UNIFORM PROCESSES

YIANNIS N. MOSCHOVAKIS

Contents

Introduction. Our main aim is to develop, explain and discuss some applications of the *homomorphism method* for establishing complexity lower bounds for various complexity measures in arithmetic and algebra. The lower bounds derived by the homomorphism method are provably *robust* with respect to the choice of computation model, and plausibly *absolute*, i.e., they restrict all algorithms which compute a given function from specified primitives.

For a sample result which illustrates the kind of problems and algorithms with which we will be concerned, suppose

$$\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_k^{\mathbf{A}}, \phi_1^{\mathbf{A}}, \dots, \phi_l^{\mathbf{A}}) = (A, \mathbf{\Phi})$$

is a first order structure on the vocabulary $\Phi = \{R_1, \dots, R_k, \phi_1, \dots, \phi_l\}$, suppose that $P \subseteq A^n$ is an n-ary relation on A and let $\Phi_0 \subseteq \Phi$. From these data, we will define a function

$$\text{(1)} \qquad \text{calls}_{\Phi_0}(\mathbf{A}, P) : A^n \to \mathbb{N} \cup \{\infty\} = \{0, 1, \dots, \infty\},$$

such that if α is any *algorithm from* (or *relative to*) $\mathbf{\Phi}$ which decides P, then for all $\vec{x} \in A^n$,

$$\text{(2)} \quad \text{calls}_{\Phi_0}(\mathbf{A}, P, \vec{x}) \leq \text{ the number of calls to primitives in } \mathbf{\Phi}_0$$
$$\text{that } \alpha \text{ must execute to decide } P(\vec{x}).$$

This will be a theorem if α is expressed by any one of many (deterministic or non-deterministic) computation models for algorithms from specified primitives $\mathbf{\Phi}$, so that, in particular, the complexity measure on the right is precisely defined. It will be made plausible for all algorithms, by a brief conceptual analysis of what it means (minimally) to *compute from primitives.*

Also, (2) will not be trivial if P is decided by some algorithm, i.e., we will not just set $\text{calls}_{\Phi_0}(\mathbf{A}, P, \vec{x}) = 0$ and go home: it yields some of the best lower bounds that are known for arithmetic and algebraic problems, and in some cases it shows that known algorithms are *optimal* on some infinite set of inputs, exactly or up to a multiplicative constant.

For example, it implies the following two results, which we will make precise in the body of the manuscript.

(A) The *Stein* (binary) *algorithm* is a competitor to the Euclidean algorithm: it computes the greatest common divisor $\gcd(m, n)$ of two natural numbers and so decides whether they are coprime using $O(\max(\log m, \log n))$ calls to its primitives, which are *piecewise linear* (Presburger) functions and relations on $\mathbb{N}$.

Among algorithms from its primitives which decide coprimeness on $\mathbb{N}$, the Stein algorithm is optimal (up to a multiplicative constant) on an infinite set of inputs.

(B) Let $\mathbb{R}$ be the field of real numbers, and for $a_0, \dots, a_n, x \in \mathbb{R}$, let

$$N_{\mathbb{R}}(a_0, a_1, \dots, a_n, x) \iff a_0 + a_1 x + \dots + a_n x^n = 0.$$

This is the relation of *polynomial nullity* (or 0-*testing*) on $\mathbb{R}$. The classical *Horner's rule* evaluates $a_0 + a_1 x + \dots + a_n x^n$ and so decides $N_{\mathbb{R}}$ in the real field expanded by the equality relation

$$\mathbf{R} = (\mathbb{R}, 0, 1, +, -, \cdot, \div, =).$$

It makes n multiplication calls, n addition calls, and one call to $=$.

Among algorithms from the primitives of $\mathbf{R}$ *which decide polynomial nullity, Horner's rule is optimal on algebraically independent inputs, for multiplications/divisions, for multiplications/divisions/equality tests and for additions/subtractions. It is near-optimal for additions/subtractions/equality tests.*

The second result can be expressed succinctly using the notation of (1): for algebraically independent, $a_0, \dots, a_n, x$,

$$\text{calls}_{\{\cdot,\div\}}(\mathbf{R}, N_{\mathbb{R}}, \vec{a}, x) = n, \quad \text{calls}_{\{\cdot,\div,=\}}(\mathbf{R}, N_{\mathbb{R}}, \vec{a}, x) = n + 1$$
$$\text{calls}_{\{+,-\}}(\mathbf{R}, N_{\mathbb{R}}, \vec{a}, x) = n - 1 \ (n > 1), \quad \text{calls}_{\{+,-,=\}}(\mathbf{R}, N_{\mathbb{R}}, \vec{a}, x) = n.$$

Various versions of these results are known, though not in the generality in which we will formulate them. (A) is proved for recursive programs (and *inessential extensions* of them) in van den Dries and Moschovakis [2004], which also contains a much more interesting lower bound result for coprimeness from division with remainder that we will also discuss. The results in (B) are equivalent to theorems proved for algebraic decision trees in Bürgisser and Lickteig [1992] and Bürgisser, Lickteig, and Shub [1992]. Our aim here is not to look for new results, but to develop, explain and justify the broad applicability and robustness of the homomorphism method in at least two, quite different areas.

The motivation for this work came from analysing the derivations of lower bounds for decision problems in van den Dries and Moschovakis [2004, 2009], most of which are grounded on the fact that natural complexity measures on recursive programs are preserved under (suitably defined) *embeddings*. One can extract from these proofs three simple axioms, in the style of *abstract model theory*, which imply those lower bound results and are plausibly true of all algorithms from primitives. An attempt to apply the method to problems in algebra revealed that when we axiomatize it in this way, the *embedding method* of van den Dries and Moschovakis [2004, 2009] is very similar to the *substitution method* of Pan [1966] which has been widely used in algebraic complexity, cf. Winograd [1967, 1970], the articles by Bürgisser and others cited above and the textbook Bürgisser, Clausen, and Shokollahi [1997]. This common theory of lower bounds for relative complexity is then our topic.

We have included an unusually long and tedious section on preliminaries, which may be needed because it is important to formulate correctly the basic notions of equational logic for partial functions and relations. Most readers will want to skip through it quickly and return to it only when the use of some term seems unfamiliar. The homomorphism method is developed in Section 2 for *uniform processes*, which include algorithms, as they are usually specified by deterministic or non-deterministic *computation models*; this is verified for many of the standard computation models in Section 5. Finally, in Sections 3 and 4 we discuss some of the applications of uniform process theory, especially to coprimeness in arithmetic and polynomial 0-testing in algebra.

§1. Preliminaries and notation. A *partial function* $f : X \rightharpoonup W$ on a product $X = X_1 \times \cdots \times X_n$ to W is a (total) function $f : D_f \to W$, where the set $D_f \subseteq X$ is the *domain of convergence* of f. We write as usual,

$$f(\vec{x})\downarrow \iff \vec{x} \in D_f \ (f \text{ converges at } \vec{x}),$$
$$f \sqsubseteq g \iff (\forall \vec{x})[f(\vec{x})\downarrow \implies f(\vec{x}) = g(\vec{x})] \ (f \text{ is a subfunction of } g).$$

Partial functions compose strictly, i.e.,

$$\begin{aligned}&f(g_1(\vec{y}),\dots,g_n(\vec{y})) = w\\ &\iff (\exists u_1,\dots,u_n)[g_1(\vec{y}) = u_1 \;\&\; \cdots \;\&\; g_n(\vec{y}) = u_n \;\&\; f(u_1,\dots,u_n) = w].\end{aligned}$$

A *partial relation* on X is a partial function $R : X \rightharpoonup \{\mathrm{tt}, \mathrm{ff}\}$ on X to the truth set $\{\mathrm{tt}, \mathrm{ff}\}$, and we write synonymously

$$R(\vec{x}) \iff R(\vec{x}) = \mathrm{tt}, \quad \neg R(\vec{x}) \iff R(\vec{x}) = \mathrm{ff}.$$

These definitions make sense for $n = 0$, by the usual convention: the *nullary product set* is the singleton $\{(\,)\}$ of "the empty tuple", and so a nullary partial function is any $f : \{(\,)\} \rightharpoonup W$. It is identified with its unique value if it converges, $f = f((\,)) \in W$.

(Many-sorted, partial) structures. A pair (S, Φ) is a *signature* if the *set of sorts* S is not empty, containing in particular the *boolean sort* `boole`, and the *vocabulary* Φ is a set of function symbols, each with an assigned *type* of the form[1]

$$\mathrm{type}(\phi) \equiv \langle s_1,\dots,s_n,\mathrm{sort}(\phi)\rangle$$

with $s_1,\dots,s_n \in S \setminus \{\mathtt{boole}\}$ and $\mathrm{sort}(\phi) \in S$. A (partial) (S,Φ)-*structure* is a pair

$$\mathbf{A} = (\{A_s\}_{s\in S}, \mathbf{\Phi}) = (\{A_s\}_{s\in S}, \{\phi^{\mathbf{A}}\}_{\phi\in\Phi}), \tag{3}$$

where each A_s is a set; $A_{\mathtt{boole}}$ is the set of truth values $\{\mathrm{tt}, \mathrm{ff}\}$; and for each $\phi \in \Phi$,

$$\text{if } \mathrm{type}(\phi) = \langle s_1,\dots,s_n,s\rangle, \text{ then } \phi^{\mathbf{A}} : A_{s_1} \times \cdots \times A_{s_n} \rightharpoonup A_s.$$

The convergent objects $\phi^{\mathbf{A}}$ with $\mathrm{type}(\phi) = \langle s\rangle$ are the *distinguished elements of sort s* of $\mathbf{A}$.

We will adopt the natural convention about the identity symbol: if $=_s$ occurs in the vocabulary Φ, it is then interpreted in every (S,Φ)-structure $\mathbf{A}$ by *some subfunction* $=_s^{\mathbf{A}} \sqsubseteq =_{A_s}$ of the (total) identity relation $=_{A_s}$: $A_s \to \{\mathrm{tt}, \mathrm{ff}\}$ on A_s—*not necessarily by* $=_{A_s}$. We will also use the unary relations of identity with a specific object w in some basic domain A_s,

$$\mathrm{eq}_w(x) \iff x = w.$$

One-sorted, Φ-structures. Most often there is just one sort $\mathtt{a}$ (other than `boole`) and Φ is finite: we describe these structures as usual, by identifying the *universe* $A = A_{\mathtt{a}}$, listing $\mathbf{\Phi}$, and letting the notation suggest

$$\mathrm{type}(\phi) \equiv \langle n_\phi, s\rangle :\equiv \langle \underbrace{\mathtt{a},\dots,\mathtt{a}}_{n_\phi}, s\rangle$$

for every $\phi \in \Phi$. Typical are the basic structures of unary and binary arithmetic[2]

$$\mathbf{N}_u = (\mathbb{N}, 0, S, \mathrm{Pd}, \mathrm{eq}_0), \quad \mathbf{N}_b = (\mathbb{N}, 0, \mathrm{parity}, \mathrm{iq}_2, \mathrm{em}_2, \mathrm{om}_2, \mathrm{eq}_0), \tag{4}$$

[1]We use "$\equiv$" for the equality relation on syntactic objects like types, terms, etc.

[2]We use $\mathrm{iq}(x,y)$ and $\mathrm{rem}(x,y)$ for the integer quotient and remainder (partial) functions on $\mathbb{N}$, determined by the conditions

$$x = y\,\mathrm{iq}(x,y) + \mathrm{rem}(x,y), \quad (x,y \in \mathbb{N}, y \neq 0, 0 \leq \mathrm{rem}(x,y) < y).$$

where $\mathrm{parity}(x) = \mathrm{rem}(x, 2), \mathrm{iq}_2(x) = \mathrm{iq}(x, 2)$ and

$$\mathrm{em}_2(x) = 2x,\ \mathrm{om}_2(x) = 2x + 1$$

are the operations of *even* and *odd multiplication.* More generally, for any $k \geq 2$, the structure of *k-ary arithmetic* is

$$\mathbf{N}_k = (\mathbb{N}, 0, \mathrm{m}_{k,0}, \dots , \mathrm{m}_{k,k-1}, \mathrm{iq}_k, \mathrm{rem}_k, \mathrm{eq}_0), \tag{5}$$

where $\mathrm{m}_{k,i}(x) = kx + i$, $\mathrm{iq}_k(x) = \mathrm{iq}(x, k)$ and $\mathrm{rem}_k(x) = \mathrm{rem}(x, k)$. These are *total structures*, as is the standard structure of Peano arithmetic

$$\mathbf{N} = (\mathbb{N}, 0, 1, +, \cdot, =) \tag{6}$$

An example of a genuinely partial structure is a *field* (with identity)

$$\mathbf{F} = (F, 0, 1, +, -, \cdot, \div, =),$$

where the quotient $x \div y$ is defined only when $y \neq 0$.

There are many interesting examples of many-sorted structures, e.g., a *vector space V* over a field F

$$\mathbf{V} = (V, F, 0_F, 1_F, +_F, \cdot_F, 0_V, +_V, \cdot)$$

where $\cdot : F \times V \to V$ is scalar-vector multiplication and the other symbols have their natural meanings. On the other hand, dealing directly with many sorts is tedious, and we will work with one-sorted Φ-structures. The more general versions follow by "identifying" a many-sorted $\mathbf{A}$ as in (3) with the single-sorted

$$(\oplus_{s\in S'} A_s, \{A_s : s \in S'\}, \mathbf{\Phi}), \tag{7}$$

where $\oplus_{s\in S'} A_s = \bigcup\{\langle s, x\rangle : s \in S \setminus \{\mathtt{boole}\}\ \&\ x \in A_s\}$ is the *disjoint union* of the basic universes of $\mathbf{A}$, $A_s(x) \Leftrightarrow x \in A_s$ for $s \neq \mathtt{boole}$, and the primitives in $\mathbf{\Phi}$ are as before, undefined on arguments not of the appropriate type.

Restrictions. If $\mathbf{A} = (A, \mathbf{\Phi})$ is a $\mathbf{\Phi}$-structure and $U \subseteq A = A_{\mathsf{a}}$, we set

$$\mathbf{A} \restriction U = (U, \{\phi^{\mathbf{A}} \restriction U\}_{\phi\in\Phi}),$$

where, for any $f : A^n \rightharpoonup A_s$ $(s \in \{\mathtt{a}, \mathtt{boole}\})$,

$$f \restriction U(x_1, \dots , x_n) = w \iff x_1, \dots , x_n \in U, w \in U_s\ \&\ f(x_1, \dots , x_n) = w.$$

Notice that we allow $U = \emptyset$, as well as non-empty U which may not contain all the distinguished elements of $\mathbf{A}$.

Expansions and reducts. An *expansion* of a Φ-structure $\mathbf{A}$ is obtained by adding new primitives to $\mathbf{A}$,

$$(\mathbf{A}, \mathbf{\Psi}) = (A, \mathbf{\Phi} \cup \mathbf{\Psi}).$$

Conversely, the *reduct* $\mathbf{A} \restriction \Phi_0$ of a structure $\mathbf{A} = (A, \mathbf{\Phi})$ to a subset $\Phi_0 \subseteq \Phi$ of its vocabulary is defined by removing all the operations in $\mathbf{\Phi} \setminus \mathbf{\Phi}_0$. For example, the reduct of the field of real numbers to $\{0, +, -\}$ is the additive group on $\mathbb{R}$,

$$(\mathbb{R}, 0, 1, +, -, \cdot, \div) \restriction \{0, +, -\} = (\mathbb{R}, 0, +, -).$$

Two numbers are *coprime* if no number greater than 1 divides both of them, in symbols

$$x \perp\!\!\!\perp y \iff \gcd(x, y) = 1 \quad (x, y > 0).$$

Diagrams. The (equational) *diagram* of a Φ-structure $\mathbf{A}$ is the set

$$\mathrm{eqdiag}(\mathbf{A}) = \{(\phi, \vec{x}, w) : \phi \in \Phi, \vec{x} \in A^n, w \in A_{\mathrm{sort}(\phi)} \text{ and } \phi^{\mathbf{A}}(\vec{x}) = w\},$$

and its *visible universe* is the set of members of A which occur in $\mathrm{eqdiag}(\mathbf{A})$,

$$A_{\mathrm{vis}} = \{x \in A : \exists(\phi, x_0, \dots, x_{n-1}, x_n) \in \mathrm{eqdiag}(\mathbf{A})(\exists i)[x = x_i]\}.$$

It is sometimes convenient to specify a structure $\mathbf{A}$ by giving its equational diagram, which (by convention then) means that $A = A_{\mathrm{vis}}$. For example, if we set

$$\mathbf{U} = \{2+1=3,\ 2+3=5,\ 2 \le 5,\ \neg(5 \le 1)\} \tag{8}$$

with $\Phi = \{0, S, +, \le\}$, then $U = U_{\mathrm{vis}} = \{1, 2, 3, 5\}$ and $\mathbf{U}$ is a finite structure in which (among other things) S is interpreted by the empty partial function. (And we have used here the obvious convention, to write $\phi(\vec{x}) = w, R(\vec{x}), \neg R(\vec{x})$ rather than the more pedantic $(\phi, \vec{x}, w), (R, \vec{x}, \mathrm{tt}), (R, \vec{x}, \mathrm{ff})$ in diagrams.)

Substructures. A (partial) *substructure* $\mathbf{U} \subseteq_p \mathbf{A}$ of a Φ-structure $\mathbf{A}$ is a structure of the same vocabulary Φ, such that $U \subseteq A$ and for every $\phi \in \Phi$, $\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}$, i.e.,

$$\Big(\vec{x} \in U^n \ \&\ w \in U_s \ \&\ \phi^{\mathbf{U}}(\vec{x}) = w\Big) \Longrightarrow \phi^{\mathbf{A}}(\vec{x}) = w.$$

A substructure $\mathbf{U}$ is *strong* (or *induced*) if in addition

$$\Big(\vec{x} \in U^n \ \&\ w \in U_s \ \&\ \phi^{\mathbf{A}}(\vec{x}) = w\Big) \Longrightarrow \phi^{\mathbf{U}}(\vec{x}) = w,$$

in which case $\mathbf{U} = \mathbf{A} \restriction U$, the restriction of $\mathbf{A}$ to the universe of $\mathbf{U}$. At the same time, for every U, the structure $\mathbf{U}_\emptyset$ in which all symbols of Φ are interpreted by totally undefined partial functions and relations is also a substructure of $\mathbf{A}$.

Notice that

$$\mathbf{U} \subseteq_p \mathbf{A} \iff U \subseteq A \ \&\ \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A}),$$

and if $U = U_{\mathrm{vis}}$, then

$$\mathbf{U} \subseteq_p \mathbf{A} \iff \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A}).$$

Homomorphisms and embeddings. A *homomorphism* $\pi : \mathbf{U} \to \mathbf{V}$ of one Φ-structure into another is any mapping $\pi : U \to V$ such that

$$\phi^{\mathbf{U}}(x_1, \dots, x_n) = w \Longrightarrow \phi^{\mathbf{V}}(\pi(x_1), \dots, \pi(x_n)) = \pi(w). \tag{9}$$

In reading this we extend π to $\{\mathrm{tt}, \mathrm{ff}\}$ by $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$, so that for partial relations it insures

$$R^{\mathbf{U}}(x_1, \dots, x_n) \Longrightarrow R^{\mathbf{V}}(\pi(x_1), \dots, \pi(x_n)),$$
$$\neg R^{\mathbf{U}}(x_1, \dots, x_n) \Longrightarrow \neg R^{\mathbf{V}}(\pi(x_1), \dots, \pi(x_n)).$$

A homomorphism is an *embedding* $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ if it is injective (one-to-one), and it is an *isomorphism* $\pi : \mathbf{U} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \mathbf{V}$ if it is a surjective embedding and, in addition, the inverse map $\pi^{-1} : U \rightarrowtail\!\!\!\!\!\twoheadrightarrow V$ is also an embedding. Clearly

$$\mathbf{U} \subseteq_p \mathbf{V} \iff U \subseteq V \text{ and the identity } \mathrm{id}_U : U \rightarrowtail V \text{ is an embedding.}$$

If $\pi : \mathbf{U} \to \mathbf{A}$ is a homomorphism, then $\pi[\mathbf{U}]$ is the substructure of $\mathbf{A}$ with universe $\pi[U]$ and

$$\text{eqdiag}(\pi[\mathbf{U}]) = \{(\phi, \pi(x_1), \dots , \pi(x_n), \pi(w)) : (\phi, x_1, \dots , x_n), w) \in \text{eqdiag}(\mathbf{U})\}.$$

This construction is especially useful when $\pi : \mathbf{U} \rightarrowtail \mathbf{A}$ is an embedding, in which case $\pi : \mathbf{U} \rightarrowtail\!\!\!\rightarrow \pi[\mathbf{U}]$ is an isomorphism.

Φ-terms (with conditionals). These are defined by the structural recursion

$$t :\equiv \mathrm{tt} \mid \mathrm{ff} \mid \mathsf{v}_i \mid \phi(t_1, \dots , t_{n_\phi}) \mid \text{if } \ t_1 \text{ then } t_2 \text{ else } t_3 \qquad (\phi \in \Phi),$$

where $\mathsf{v}_0, \mathsf{v}_1, \dots$ is a fixed sequence of individual variables. The definition assigns to each term a sort `boole` or `a` and sets type restrictions on the formation rules in the obvious way; for the conditional it is required that $\text{sort}(t_1) \equiv \texttt{boole}$ and $\text{sort}(t_2) \equiv \text{sort}(t_3)$ and then $\text{sort}(t) \equiv \text{sort}(t_2)$. The conditional can be used to define the propositional connectives on terms of boolean sort, e.g.,

$$s \ \& \ t :\equiv \text{if } s \text{ then } t \text{ else ff}, \quad \neg\, t :\equiv \text{if } t \text{ then ff else tt}.$$

A term t is *algebraic* if no conditional occurs in it, and *closed* if it has no variables.

We will use the familiar notation

$$t(\vec{v}) \equiv t(v_1, \dots , v_n) :\equiv (t, (v_1, \dots , v_n))$$

for a pair of a term and a sequence of distinct variables which includes all the variables that occur in t, and by the usual, sloppy convention, we will also refer to these pairs as "terms". If $\mathbf{A}$ is any Φ-structure, then for any $t(\vec{v})$,

$$t^{\mathbf{A}}[\vec{x}] = t[\mathbf{A}, \vec{x}] = \text{the value of } t(\vec{v}) \text{ in } \mathbf{A}, \text{ when } \vec{v} := \vec{x}.$$

Generated structures. For a fixed Φ-structure $\mathbf{A}$, each tuple $\vec{x} \in A^n$ and each $m \in \mathbb{N}$, set

$$\begin{aligned} &G_m(\mathbf{A}, \vec{x}) = \{t^{\mathbf{A}}[\vec{x}] : t \text{ is algebraic}, \text{sort}(t) = \mathtt{a} \text{ and depth}(t(\vec{v})) \le m\},\\ &G_\infty(\mathbf{A}, \vec{x}) = \textstyle\bigcup_m G_m(\mathbf{A}, \vec{x}), \end{aligned}$$

and for the corresponding *induced substructures*,

$$\mathbf{G}_m(\mathbf{A}, \vec{x}) = \mathbf{A} \restriction G_m(\mathbf{A}, \vec{x}), \quad \mathbf{G}_\infty(\mathbf{A}, \vec{x}) = \mathbf{A} \restriction G_\infty(\mathbf{A}, \vec{x}).$$

Clearly, $G_\infty(\mathbf{A}, \vec{x})$ is the smallest subset S of A which satisfies

$$\vec{x} \in S^n \ \& \ (\forall \vec{u} \in S^{n_\phi}, w)[\phi^{\mathbf{A}}(\vec{u}) = w \Longrightarrow w \in S]. \tag{10}$$

We will often prove that *every $u \in G_\infty(\mathbf{A}, \vec{x})$ has some property P* by "induction on the generation of G_∞", i.e., by checking (10) with $S = P$.

A structure $\mathbf{A}$ is *generated by* $\vec{x}$ if $\mathbf{A} = \mathbf{G}_\infty(\mathbf{A}, \vec{x})$; if it is finite and generated by $\vec{x}$, then $\mathbf{A} = \mathbf{G}_m(\mathbf{A}, \vec{x})$ for some m. Most often we will use finite substructures of some fixed $\mathbf{A}$ which are generated by a tuple of their elements, and we set

$$\text{depth}(\mathbf{U}, \vec{x}) = \min\{m : \vec{x} \in U^n, \mathbf{U} = \mathbf{G}_m(\mathbf{U}, \vec{x})\}.$$

The *size* of a finite $\mathbf{U} \subseteq_p \mathbf{A}$ is the number of all visible elements of its universe,

$$\begin{aligned} \text{size}(\mathbf{U}) &= |U_{\text{vis}}| \\ &= \Big|\{v \in U : v \text{ occurs in some } (\phi, \vec{u}, w) \in \text{eqdiag}(\mathbf{U})\}\Big| \le |U|. \end{aligned} \tag{11}$$

We also need the *depth of an element below a tuple*,

$$\text{depth}(w;\mathbf{A},\vec{x}) = \min\{m : w \in G_m(\mathbf{A},\vec{x})\}, \quad (w \in G_\infty(\mathbf{A},\vec{x})). \tag{12}$$

Proposition 1.1. *If* $\mathbf{U}$ *is a* Φ*-structure* $\mathbf{U}$*,* $\vec{x} \in U^n$ *and* $\text{depth}(w;\mathbf{U},\vec{x}) = m$ *for some* $w \in U$*, then*

$$m \le \text{size}(\mathbf{G}_m(\mathbf{U},\vec{x})) \le |\text{eqdiag}(\mathbf{G}_m(\mathbf{U},\vec{x}))|. \tag{13}$$

It follows that if $\mathbf{U}$ *is finite and generated by* $\vec{x}$*, then*

$$\text{depth}(\mathbf{U},\vec{x}) \le \text{size}(\mathbf{U}) \le |\text{eqdiag}(\mathbf{U})|. \tag{14}$$

Proof of (13) is by induction on m, the basis being trivial since all three numbers in it are 0.

For the induction step in the first inequality, we are given some w with

$$\text{depth}(w;\mathbf{U},\vec{x}) = m + 1,$$

so that $w = \phi^{\mathbf{U}}(u_1,\dots,u_n)$ and for some i, $\text{depth}(u_i;\mathbf{U},\vec{x}) = m$. By the induction hypothesis,

$$m \le \text{size}(\mathbf{G}_m(\mathbf{U},\vec{x})) \le \text{size}(\mathbf{G}_{m+1}(\mathbf{U},\vec{x})) - 1,$$

the latter because w occurs in the equation

$$(\phi,u_1,\dots,u_n,w) \in \text{eqdiag}(\mathbf{G}_{m+1}(\mathbf{U},\vec{x}))$$

and is not a member of $G_m(\mathbf{U},\vec{x})$. So $m+1 \le \text{size}(G_{m+1}(\mathbf{U},\vec{x}))$.

For the induction step in the proof of the second inequality, notice that (skipping $\mathbf{U}$ and $\vec{x}$ which remain constant in the argument),

$$G_{m+1} = G_m \cup \{\phi^{\mathbf{U}}(u_1,\dots,u_k) : u_1,\dots,u_k \in G_m \ \&\ \phi^{\mathbf{U}}(u_1,\dots,u_k) \notin G_m\}.$$

The first of these two disjoint pieces has size $\le |\text{eqdiag}(\mathbf{G}_m)|$ by the induction hypothesis, and to each w is the second piece we can associate in a one-to-one way some equation $(\phi,\vec{u},w)$ in the diagram of $\mathbf{G}_{m+1}$ which is not in the diagram of $\mathbf{G}_m$, because $w \notin G_m$; so

$$\begin{aligned}\text{size}(G_{m+1}) \le |\text{eqdiag}(\mathbf{G}_m)| + \Big(|\text{eqdiag}(\mathbf{G}_{m+1})| - |\text{eqdiag}(\mathbf{G}_m)|\Big)& \\ = |\text{eqdiag}(\mathbf{G}_{m+1})|.& \quad \dashv\end{aligned}$$

§2. The homomorphism method.

The key notions of this section (and the manuscript) are those of a *uniform process* in 2C and *certification* in 2F, and the main result is the *Homomorphism Test*, Lemma 2.4.

2A. Axioms which capture the uniformity of algorithms. Our basic intuition is that an *n-ary algorithm* α of sort $s \in \{\texttt{a},\texttt{boole}\}$ *of a structure* $\mathbf{A} = (A,\mathbf{\Phi})$ computes (in some way) an n-ary partial function

$$\overline{\alpha} = \overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s \quad (\text{with } A_{\texttt{boole}} = \{\text{tt},\text{ff}\}, A_{\texttt{a}} = A)$$

using the primitives in $\mathbf{\Phi}$ as *oracles*. We understand this to mean that in the course of a computation of $\overline{\alpha}(\vec{x})$, the algorithm may request from the oracle for any $\phi^{\mathbf{A}}$ any particular value $\phi^{\mathbf{A}}(u_1,\dots,u_{n_\phi})$, where each u_i either is given by the input or has already been computed; and that if the oracles cooperate and respond to all requests, then this computation of $\overline{\alpha}(\vec{x})$ is completed in a finite number of steps.

The three axioms we formulate in this section capture part of this minimal understanding of how algorithms from primitives operate.

The crucial first axiom expresses the possibility that the oracles may choose not to respond to a request for $\phi^{\mathbf{A}}(u_1, \dots, u_{n_\phi})$ unless

$$u_1, \dots, u_n \in U \ \& \ \phi^{\mathbf{U}}(u_1, \dots, u_n)\downarrow$$

for some fixed substructure $\mathbf{U} \subseteq_p \mathbf{A}$: the algorithm will still compute a partial function, which simply diverges on those inputs $\vec{x}$ for which no computation of $\overline{\alpha}(\vec{x})$ by α can be executed "inside" $\mathbf{U}$ (as far as calls to the oracles are involved).

I. Locality Axiom. *An n-ary algorithm α of sort $s \in \{\mathtt{a}, \mathtt{boole}\}$ of a structure $\mathbf{A}$ assigns to each substructure $\mathbf{U} \subseteq_p \mathbf{A}$ an n-ary partial function*

$$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s.$$

We understand this axiom constructively, i.e., we claim that the *localization operation*

$$(15) \qquad (\mathbf{U} \mapsto \overline{\alpha}^{\mathbf{U}}) \quad (\text{where } \mathbf{U} \subseteq_p \mathbf{A} \text{ and } \overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s)$$

is induced naturally by a specification of α. We set

$$(16) \ \mathbf{U} \vdash \alpha(\vec{x}) = w \iff \vec{x} \in U^n \ \& \ \overline{\alpha}^{\mathbf{U}}(\vec{x}) = w \quad (\vdash \text{ is read "proves"}),$$

$$(17) \qquad \mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff (\exists w)[\mathbf{U} \vdash \alpha(\vec{x}) = w],$$

and we call $\overline{\alpha}^{\mathbf{U}}$ the partial function on U *computed* by α.

In particular, α computes on A the partial function $\overline{\alpha} = \overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$.

II. Homomorphism Axiom. *If α is an algorithm of $\mathbf{A}$ and $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism of one substructure of $\mathbf{A}$ into another, then*

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \Longrightarrow \mathbf{V} \vdash \alpha(\pi(\vec{x})) = \pi(w) \quad (\vec{x} \in U^n),$$

where $\pi(x_1, \dots, x_n) = (\pi(x_1), \dots, \pi(x_n))$. In particular, by applying this to the identity embedding $\mathrm{id}_U : \mathbf{U} \rightarrowtail \mathbf{A}$,

$$\mathbf{U} \subseteq_p \mathbf{A} \Longrightarrow \overline{\alpha}^{\mathbf{U}} \sqsubseteq \overline{\alpha}^{\mathbf{A}} = \overline{\alpha}.$$

The idea here is that the oracle for each $\phi^{\mathbf{A}}$ may consistently respond to each request for $\phi^{\mathbf{U}}(\vec{u})$ by delivering $\phi^{\mathbf{V}}(\pi(\vec{u}))$. This transforms any computation of $\overline{\alpha}^{\mathbf{U}}(\vec{x})$ into one of $\overline{\alpha}^{\mathbf{V}}(\pi(\vec{x}))$, which in the end delivers the value $\pi(w) = \pi(\overline{\alpha}^{\mathbf{U}}(\vec{x}))$.

This argument is convincing for the identity embedding $\mathrm{id}_U : \mathbf{U} \rightarrowtail \mathbf{V}$. It is not quite that simple in the general case, because α may utilize in its computations complex data structures and rich primitives, e.g., stacks, queues, trees, conditionals, the introduction of higher type objects by λ-abstraction and subsequent application of these objects to suitable arguments, etc. The claim is that any homomorphism $\pi : \mathbf{U} \to \mathbf{V}$ lifts naturally to these data structures, and so the image of a convergent computation of $\overline{\alpha}^{\mathbf{U}}(\vec{x})$ is a convergent computation of $\overline{\alpha}^{\mathbf{V}}(\pi(\vec{x}))$. Put another way: if some $\pi : \mathbf{U} \to \mathbf{V}$ does not lift naturally to a mapping of the relevant computations, then α *is using essentially some hidden primitives not included in* $\mathbf{A}$ and so *it is not an algorithm from* $\{\phi^{\mathbf{A}}\}_{\phi \in \Phi}$.

It is clear, however, that the Homomorphism Axiom demands something more of algorithms (and how they use oracles) than the Locality Axiom, and so it is important that we verify it for the standard computation models. We will do this in Section 5.

The Homomorphism Axiom is at the heart of this approach to the derivation of lower bounds.

III. Finiteness Axiom. *If $\overline{\alpha}(\vec{x}) = w$, then there is a finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x}$ such that $\mathbf{U} \vdash \alpha(\vec{x}) = w$.*

This combines two ingredients of the basic intuition: first that in the course of a computation, the algorithm may only request of the oracles values $\phi^{\mathbf{A}}(\vec{u})$ for arguments $\vec{u}$ that it has already constructed from the input, and second, that computations are finite. A suitable $\mathbf{U}$ is then determined by putting in eqdiag($\mathbf{U}$) all the *calls made by α in some computation of $\overline{\alpha}(\vec{x})$.*

This axiom implies, in particular, that partial functions computed by an $\mathbf{A}$-algorithm take values in the substructure generated by the input,

$$\overline{\alpha}^{\mathbf{A}}(\vec{x}) = w \Longrightarrow w \in G_\infty(\mathbf{A}, \vec{x}) \cup \{\mathrm{tt}, \mathrm{ff}\}.$$

This is a substantial restriction, but not for decision problems, when $\overline{\alpha}$ is the characteristic function of a relation and so for all $\vec{x}$, $\overline{\alpha}(\vec{x}) \in \{\mathrm{tt}, \mathrm{ff}\}$.

It is useful to set

(18) $\mathbf{U} \vdash_c \alpha(\vec{x}) = w \iff \mathbf{U}$ is finite, generated by $\vec{x}$, and $\mathbf{U} \vdash \alpha(\vec{x}) = w$,

(19) $\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff (\exists w)[\mathbf{U} \vdash_c \alpha(\vec{x}) = w]$.

In this notation, the Finiteness Axiom takes the form

(20) $$\overline{\alpha}(\vec{x}) = w \Longrightarrow (\exists \mathbf{U} \subseteq_p \mathbf{A})[\mathbf{U} \vdash_c \alpha(\vec{x}) = w].$$

If we read "$\vdash_c$" as *computes*, then this form of the axiom suggests that the triples $(\mathbf{U}, \vec{x}, w)$ such that $\mathbf{U} \vdash_c \alpha(\vec{x}) = w$ play the role of *computations* in this abstract setting.

2B. Concrete algorithms and the Uniformity Thesis. We have been using the word *algorithm* informally, and we will not attempt to define it precisely in this manuscript. Rigorous results in complexity theory are established for *concrete algorithms*, specified by *computation models*, e.g., *Turing machines*, *Random Access machines*, *recursive programs*, ... , and their *non-deterministic* versions.

Axioms **I** – **III** are satisfied by all concrete algorithms which compute a partial function $f : A^n \rightharpoonup A$ from specified primitives. This is plausible from the motivation for the axioms above, but a complete proof is rather messy, as it must specify rigorously and take into account the idiosyncracies of each model. Part of the difficulty comes from the fact that many computation models have some functions on the intended universe A "built-in", so to speak: e.g., Turing machines acting on $\mathbb{N}$ assume the successor and predecessor functions if we code numbers by strings in unary or the primitives of binary arithmetic if we code numbers in binary, and random access machines have the identity relation on $\mathbb{N}$ built in, in addition to whatever functions on $\mathbb{N}$ are explicitly identified in their definition. To see that these models satisfy the axioms, we must identify all the *non-logical* primitives assumed—and then the result becomes basically obvious. We leave this analysis for Section 5, where we will also establish some general results which support the following claim:

UNIFORMITY THESIS FOR ALGORITHMS. *Every algorithm which computes a partial function $f : A^n \rightharpoonup A$ or decides a relation $R \subseteq A^n$ from the primitives of a Φ-structure* **A** *satisfies axioms* **I**, **II** *and* **III**.

This is a weak Church-Turing-type assumption about algorithms which, of course, cannot be established rigorously absent a precise definition of algorithms.

2C. Uniform processes. An n-ary *uniform process* of sort $s \in \{\mathtt{a}, \mathtt{boole}\}$ of a Φ-structure $\mathbf{A}$ is any operation

$$\alpha = (\mathbf{U} \mapsto \overline{\alpha}^{\mathbf{U}}) \quad (\mathbf{U} \subseteq_p \mathbf{A},\ \overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s)$$

on the substructures of $\mathbf{A}$ which satisfies the Homomorphism and Finiteness Axioms. We say that α *computes* the partial function $\overline{\alpha} : A^n \rightharpoonup A_s$, and we will use for uniform processes the notations introduced in (16) – (19) of 2A.[3]

We have argued (briefly) that every algorithm induces a uniform process, and we will prove this for many of the standard computation models in Section 5. On the other hand, the converse is far from true: nothing in axioms **II** and **III** suggests that functions computed by uniform processes are “computable” from the primitives of $\mathbf{A}$ in any intuitive sense, and in general, they are not.

PROPOSITION 2.1. *If a Φ-structure* $\mathbf{A}$ *is generated by the empty tuple, then every $f : A^n \rightharpoonup A$ is computed by some uniform process of* $\mathbf{A}$.

In particular, every $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ is computed by some uniform process of $\mathbf{A} = (\mathbb{N}, 0, \Phi^{\mathbf{A}})$ *if $\Phi^{\mathbf{A}}$ includes either the successor function S or the primitives of binary arithmetic* $\mathrm{em}_2(x) = 2x$ *and* $\mathrm{om}_2(x) = 2x + 1$.

PROOF. Let $G_m = G_m(\mathbf{A}, ( ))$ be the set generated in $\leq m$ steps by the empty tuple, so that $G_0 = \emptyset$, G_1 comprises the distinguished elements of $\mathbf{A}$, etc. Let

$$d(\vec{x}, w) = \min\{m : x_1, \dots, x_n, w \in G_m\},$$

and define $\overline{\alpha}^{\mathbf{U}}$ for each $\mathbf{U} \subseteq_p \mathbf{A}$ by

$$\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w \iff f(\vec{x}) = w \ \&\ \mathbf{G}_{d(\vec{x},w)} \subseteq_p \mathbf{U}.$$

The Finiteness Axiom is immediate taking $\mathbf{U} = \mathbf{G}_{d(\vec{x},w)}$, and the Homomorphism Axiom holds because if $\mathbf{G}_m \subseteq_p \mathbf{U}$, then every homomorphism $\pi : \mathbf{U} \to \mathbf{V}$ fixes every $u \in G_m$. ⊣

[3]In categorical terms, an n-ary uniform process α of $\mathbf{A}$ of sort $s \in \{\mathtt{boole}, \mathtt{a}\}$ is a continuous functor on the category $H_{\mathbf{A}}$ to $P_{\mathbf{A},s}$, where:

(1) the objects of $H_{\mathbf{A}}$ are all pairs $(\mathbf{U}, \vec{x})$ where $\vec{x} \in U^n$ and $\mathbf{U}$ is generated by $\vec{x}$, and a morphism $\phi : (\mathbf{U}, \vec{x}) \to (\mathbf{V}, \vec{y})$ is any homomorphism $\pi : \mathbf{U} \to \mathbf{V}$ which carries $\vec{x}$ to $\vec{y}$; and

(2) the objects of $P_{\mathbf{A},s}$ of all n-ary partial functions on A to A_s, and a morphism $\psi : p \to q$ is any partial function $\psi : A \rightharpoonup A$ such that

$$p(u_1, \dots, u_n) = w \Longrightarrow q(\psi(u_1), \dots, \psi(u_n)) = \psi(w).$$

The axioms aim to capture the *uniformity* of algorithms—that they compute all their values following “the same procedure”— but surely do not capture their *effectiveness*. They are not useful in establishing non-computability. At the same time, uniform processes carry a rich complexity theory, which mirrors and has implications for algorithmic complexity.

2D. Complexity measures on uniform processes. A *substructure norm* on a Φ-structure $\mathbf{A}$ is a function μ which assigns to each $\vec{x} \in A$ and each finite $\mathbf{U} \subseteq_p \mathbf{A}$ which is generated by $\vec{x}$ a number $\mu(\mathbf{U}, \vec{x})$, e.g.,

$$\begin{aligned} \mathrm{depth}(\mathbf{U}, \vec{x}) &= \min\{m : \mathbf{U} = \mathbf{G}_m(\mathbf{U}, \vec{x})\}, \\ \mathrm{size}(\mathbf{U}, \vec{x}) &= |U_{\mathrm{vis}}|, \\ \mathrm{calls}_{\Phi_0}(\mathbf{U}, \vec{x}) &= |\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| \quad (\Phi_0 \subseteq \Phi). \end{aligned}$$

The *complexity measure* of a uniform process α relative to a substructure norm μ is the partial function

$$C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}, \tag{21}$$

defined on the domain of convergence of $\overline{\alpha}^{\mathbf{A}}$.

By using the norms above, we get three complexity measures on uniform processes,[4]

$$\mathrm{depth}(\alpha, \vec{x}),\ \mathrm{size}(\alpha, \vec{x}),\ \mathrm{calls}_{\Phi_0}(\alpha, \vec{x}). \tag{22}$$

The first and last of these three correspond to familiar complexity measures with roughly similar names for concrete algorithms. The middle one $\mathrm{size}(\alpha, \vec{x})$ measures “the least number of points in A that α must see to compute $\overline{\alpha}(\vec{x})$” and is not familiar, but it comes up naturally in some derivations of lower bounds for the generally larger and most natural

$$\mathrm{calls}(\alpha, \vec{x}) = \mathrm{calls}_\Phi(\alpha, \vec{x}).$$

LEMMA 2.2. *For every uniform process α of a Φ-structure $\mathbf{A}$ and every $\vec{x}$ such that $\overline{\alpha}(\vec{x})\downarrow$,*

$$\mathrm{depth}(\overline{\alpha}(\vec{x}); \mathbf{A}, \vec{x}) \le \mathrm{depth}(\alpha, \vec{x}) \le \mathrm{size}(\alpha, \vec{x}) \le \mathrm{calls}(\alpha, \vec{x}). \tag{23}$$

PROOF. The first inequality is immediate, because if $\mathbf{U} \vdash_c \alpha(\vec{x}) = w$, then $w \in U_s$ and so

$$\mathrm{depth}(w; \mathbf{A}, \vec{x}) \le \mathrm{depth}(w; \mathbf{U}, \vec{x}) \le \mathrm{depth}(\mathbf{U}, \vec{x}).$$

For the third claimed inequality, suppose $\overline{\alpha}(\vec{x}) = w$ and choose a substructure $\mathbf{U} \subseteq_p \mathbf{A}$ with least $|\mathrm{eqdiag}(\mathbf{U})|$ such that $\mathbf{U} \vdash_c \alpha(\vec{x}) = w$, so that $\mathrm{calls}(\alpha, \vec{x}) = |\mathrm{eqdiag}(\mathbf{U})|$. Now $\mathrm{size}(\mathbf{U}) \le |\mathrm{eqdiag}(\mathbf{U})|$ by (13) in Proposition 1.1, and since $\mathbf{U}$ is among the substructures considered in the definition of $\mathrm{size}(\alpha, \vec{x})$, we have

$$\mathrm{size}(\alpha, \vec{x}) \le |\mathrm{eqdiag}(\mathbf{U})| = \mathrm{calls}(\alpha, \vec{x}).$$

The second inequality is proved by a similar argument. $\dashv$

[4]The depth measure can also be relativized to arbitrary $\Phi_0 \subseteq \Phi$, but it is tedious and we have no interesting results about it.

2E. The value-depth complexity of a function. Suppose α is a uniform process of a Φ-structure $\mathbf{A}$ which computes $f = \overline{\alpha} : A^n \rightharpoonup A$, so that if $f(\vec{x})\downarrow$, then $f(\vec{x}) \in G_\infty(\mathbf{A}, \vec{x})$ by Axiom **III**. The first of the inequalities in (23) with $\overline{\alpha} = f$ takes the form

$$\text{depth}(f(\vec{x}); \mathbf{A}, \vec{x}) \le \text{depth}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow), \tag{24}$$

and so $\text{depth}(f(\vec{x}); \mathbf{A}, \vec{x})$ is a lower bound for all complexity measures of α that we have considered. This should be true for any reasonable notion of algorithm and any complexity measure which counts (among other things) the applications of primitives that must be executed in sequence, simply because an algorithm must (at least) construct from the input the value $f(\vec{x})$. It is well understood and used extensively to derive lower bounds in arithmetic and algebra.[5] The more sophisticated complexity measure $\text{depth}(\alpha, \vec{x})$ is useful when f takes simple values, e.g., when $f = R$ is (the characteristic function of) a relation. In this case $\text{depth}(f(\vec{x}); \mathbf{A}, \vec{x}) = 0$ and (24) does not give us any information.

2F. Forcing and certification. Suppose $\mathbf{A}$ is a Φ-structure, $f : A^n \rightharpoonup A_s$ (with $s \in \{\mathtt{a}, \mathtt{boole}\}$), $\mathbf{U} \subseteq_p \mathbf{A}$, and $f(\vec{x})\downarrow$. A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ *respects f at $\vec{x}$* if

$$\vec{x} \in U^n \ \&\ f(\vec{x}) \in U_s \ \&\ \pi(f(\vec{x})) = f(\pi(\vec{x})). \tag{25}$$

Next come *forcing* and *certification*, the two basic notions of this article:

$$\begin{aligned}
\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x}) = w \iff{} & f(\vec{x}) = w \\
& \&\ \text{every homomorphism } \pi : \mathbf{U} \to \mathbf{A} \text{ respects } f \text{ at } \vec{x}, \\
\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x}) = w \iff{} & \mathbf{U} \text{ is finite, generated by } \vec{x} \ \ \&\ \mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x}) = w, \\
\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow \iff{} & (\exists w)[\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x}) = w].
\end{aligned}$$

If $\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow$, we call $\mathbf{U}$ a *certificate* for f at $\vec{x}$ in $\mathbf{A}$.[6]

Notice that

$$\Big(\mathbf{U}_1 \subseteq_p \mathbf{U}_2 \ \&\ \mathbf{U}_1 \Vdash^{\mathbf{A}} f(\vec{x}) = w\Big) \Longrightarrow \mathbf{U}_2 \Vdash^{\mathbf{A}} f(\vec{x}) = w,$$

so, in particular, if $\mathbf{U}_1$ certifies $f(\vec{x}) = w$, then every $\mathbf{U}_2 \supseteq_p \mathbf{U}_1$ forces $f(\vec{x}) = w$.

2G. Example: the Euclidean algorithm. To illustrate the notions we use this most ancient method for computing the greatest common divisor using iterated division. It can be specified succinctly by the recursive equation

$$\varepsilon \ :\ \gcd(x, y) = \text{if } (\text{rem}(x, y) = 0) \text{ then } y \text{ else } \gcd(y, \text{rem}(x, y)) \ (x, y \neq 0), \tag{26}$$

[5]The most interesting result of this type that I know is Theorem 4.1 of van den Dries and Moschovakis [2009], a $\sqrt{\log\log}$-lower bound for recursive programs on $\text{depth}(\gcd, a+1, b)$ from $+, -, \div$ and $\cdot$ (multiplication) for Pell pairs (a, b). This is due to van den Dries, and it is the largest lower bound known for the gcd from primitives that include multiplication. It is not known whether it holds for coprimeness, for which the best result is a $\log\log\log$-lower bound for algebraic decision trees in Mansour, Schieber, and Tiwari [1991a].

[6]To the best of my knowledge, certificates were first introduced in Pratt [1975] in his proof that primality is NP. The present notion is model theoretic and more abstract than Pratt's, but the idea is the same.

so that it is an algorithm from rem and eq_0.[7] To decide whether two numbers are coprime, we need to add to (26) a *head equation*

$$\perp\!\!\!\perp(x,y) = \text{if } (\gcd(x,y)=1) \text{ then } \mathtt{tt} \text{ else } \mathrm{ff}, \tag{27}$$

so that as a decision process for coprimeness, ε is an algorithm of the structure

$$\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, \mathrm{eq}_0, \mathrm{eq}_1). \tag{28}$$

Given $x, y \geq 1$, the Euclidean computes $\gcd(x,y)$ by successive divisions and 0-tests (calls to the rem- and eq_0-oracles)

$$\mathrm{rem}(x,y) = r_1, r_1 \neq 0, \mathrm{rem}(y, r_1) = r_2, r_2 \neq 0,$$
$$\dots, r_{n+1} \neq 0, \mathrm{rem}(r_n, r_{n+1}) = r_{n+2}, r_{n+2} = 0$$

until the remainder 0 is obtained, at which time it is known that $\gcd(x,y) = r_{n+1}$; and if it is asked to decide whether $x \perp\!\!\!\perp y$, it must then do one more check to test whether $r_{n+1} = 1$. Suppose $x \perp\!\!\!\perp y$ and collect all these calls into a substructure, writing $u \neq 0, u = 1$ for $\mathrm{eq}_0(u) = \mathrm{ff}, \mathrm{eq}_1(u) = \mathtt{tt}$ as above:

$$\mathrm{eqdiag}(\mathbf{U}_0) = \{\mathrm{rem}(x,y) = r_1, r_1 \neq 0, \mathrm{rem}(y, r_1) = r_2, r_2 \neq 0,$$
$$\dots, r_{n+1} \neq 0, \mathrm{rem}(r_n, r_{n+1}) = r_{n+2}, r_{n+2} = 0, r_{n+1} = 1\};$$

it follows that

$$\mathbf{U}_0 \Vdash_c^{\mathbf{N}_\varepsilon} x \perp\!\!\!\perp y,$$

because if $\pi : \mathbf{U}_0 \to \mathbf{N}_\varepsilon$, then the homomorphism property guarantees that

$$\mathrm{rem}(\pi(x), \pi(y)) = \pi(r_1), \pi(r_1) \neq 0, \mathrm{rem}(\pi(y), \pi(r_1)) = \pi(r_2), \pi(r_2) \neq 0, \tag{29}$$
$$\dots, \pi(r_{n+1}) \neq 0, \mathrm{rem}(\pi(r_n), \pi(r_{n+1})) = \pi(r_{n+2}), \pi(r_{n+2}) = 0, \pi(r_{n+1}) = 1,$$

and this in turn guarantees that $\pi(x) \perp\!\!\!\perp \pi(y)$, so that π respects the coprimeness relation at x, y. This is how certificates for functions and relations can be constructed from computations, and it is the basic method of applying uniform process theory to the derivation of lower bounds for concrete algorithms.

On the other hand, $\mathbf{U}_0$ is not a *smallest* substructure of $\mathbf{N}_\varepsilon$ which certifies that $x \perp\!\!\!\perp y$. Let

$$\mathbf{U}_1 = \{\mathrm{rem}(x,y) = r_1, \mathrm{rem}(y, r_1) = r_2, \dots, \mathrm{rem}(r_n, r_{n+1}) = r_{n+2}, r_{n+1} = 1\},$$

the substructure of $\mathbf{U}_0$ with all the 0-tests deleted. We claim that $\mathbf{U}_1$ is also a certificate for $x \perp\!\!\!\perp y$, and to see this suppose that $\pi : \mathbf{U}_1 \to \mathbf{N}_\varepsilon$ is a homomorphism. To verify that π respects $x \perp\!\!\!\perp y$, check first that if $i = 1, \dots, n+1$, then $\pi(r_i) \neq 0$; otherwise $\mathrm{rem}(\pi(r_{i-1}, \pi(r_i))$ would not be defined (with $r_0 = y$), since rem requires its second argument to be non-zero, and so π would not be totally defined in $\mathbf{U}_1$. So the homomorphism property guarantees that

$$\mathrm{rem}(\pi(x), \pi(y)) = \pi(r_1), \pi(r_1) \neq 0, \mathrm{rem}(\pi(y), \pi(r_1)) = \pi(r_2), \pi(r_2) \neq 0,$$
$$\dots, \pi(r_{n+1}) \neq 0, \mathrm{rem}(\pi(r_n), \pi(r_{n+1})) = \pi(r_{n+2}), \pi(r_{n+1}) = 1.$$

[7] The Euclidean is viewed as an algorithm from rem alone in van den Dries and Moschovakis [2004], because eq_0 is built-in the recursive programs used in that paper. The setup we use here is somewhat more "fastidious" in separating logical from essential primitives, but there is little practical difference in arithmetic.

The last two of these equations mean that for some q,

$$\pi(r_n) = q \cdot 1 + \pi(r_{n+2}), \quad 0 \le \pi(r_{n+2}) < 1$$

so that we must have $\pi(r_{n+2}) = 0$; and then all the equations in (29) hold and we have the required $\pi(x) \perp \pi(y)$.[8] This is the same feature of uniform processes exploited in Proposition 2.1, and it is typical: i.e., although computations of concrete algorithms define certificates, they generally do not give *least-in-size* certificates—which is why the lower bounds for uniform processes are typically not the best (largest) lower bounds that one might be able to prove for concrete algorithms using other methods.

2H. The best uniform process. Is there a "best algorithm" which computes a given $f : A^n \rightharpoonup A$ from specified primitives on A? The question is vague, of course—and the answer is probably negative in the general case, no matter how you make it precise. The corresponding question about uniform processes has a positive (and very simple) answer.

For given $f : A^n \rightharpoonup A$, set

$$\overline{\boldsymbol{\beta}}^{\mathbf{U}}_{f,\mathbf{A}}(\vec{x}) = w \iff \mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x}) = w \quad (\mathbf{U} \subseteq_p \mathbf{A}). \tag{30}$$

THEOREM 2.3. *The following are equivalent for any Φ-structure $\mathbf{A}$ and any partial function $f : A^n \rightharpoonup A_s$, $s \in \{\mathtt{a}, \mathtt{boole}\}$.*

(i) *Some uniform process α of $\mathbf{A}$ computes f.*

(ii) $(\forall \vec{x})\Big(f(\vec{x})\downarrow \implies (\exists \mathbf{U} \subseteq_p \mathbf{A})[\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow]\Big)$.

(iii) *$\boldsymbol{\beta}_{f,\mathbf{A}}$ is a uniform process of $\mathbf{A}$ which computes f.*

Moreover, if these conditions hold, then for every uniform process α which computes f in $\mathbf{A}$ and for every complexity measure C_μ defined on $\mathbf{A}$ by a substructure norm as in (21),

$$C_\mu(\boldsymbol{\beta}_{f,\mathbf{A}}, \vec{x}) \le C_\mu(\alpha, \vec{x}) \qquad (f(\vec{x})\downarrow). \tag{31}$$

PROOF. (iii) $\implies$ (i) is immediate and (i) $\implies$ (ii) follows from

$$\mathbf{U} \vdash_c \alpha(\vec{x}) = w \Longrightarrow \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x}) = w, \tag{32}$$

which is an immediate consequence of the definitions.

(ii) $\implies$ (iii). The operation $(\mathbf{U} \mapsto \overline{\boldsymbol{\beta}}^{\mathbf{U}}_{f,\mathbf{A}})$ satisfies the Finiteness Axiom **III** by (ii). To verify the Homomorphism Axiom **II**, suppose

$$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x}) = w \ \&\ \pi : \mathbf{U} \to \mathbf{V}$$

so that $\pi(\vec{x}) \in V^n$, $\pi(w) \in V_s$, and (since $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism), $f(\pi(\vec{x})) = \pi(w)$. Let $\rho : \mathbf{V} \to \mathbf{A}$ be a homomorphism. The composition $\rho \circ \pi : \mathbf{U} \rightarrowtail \mathbf{A}$ is also a homomorphism, and so it respects f at $\vec{x}$, i.e.,

$$f(\rho(\pi(\vec{x})) = \rho(\pi(f(\vec{x})) = \rho(\pi(w)) = \rho(f(\pi(\vec{x}))).$$

So ρ respects f at $\pi(\vec{x})$, and since it is arbitrary, we have the required

$$\mathbf{V} \Vdash^{\mathbf{A}} f(\pi(\vec{x})) = \pi(w).$$

[8]It can also be verified that *no proper substructure of* $\mathbf{U}_1$ *certifies* $x \perp y$; for example, if we delete the last equation $r_{n+1} = 1$, then the function $\pi(u) = 2u$ defines a homomorphism on the resulting substructure such that $\gcd(\pi(x), \pi(y)) = 2$.

The second claim follows from the definition of $\boldsymbol{\beta}_{f,\mathbf{A}}$ and (32). ⊣

The key point here is that the uniform process $\boldsymbol{\beta}_{f,\mathbf{A}}$ is defined directly from f and $\mathbf{A}$, independently of any notion of algorithm. If we set[9]

$$C_\mu(\mathbf{A}, f, \vec{x}) = C_\mu(\boldsymbol{\beta}_{f,\mathbf{A}}, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\} \tag{33}$$

for any substructure norm on $\mathbf{A}$, then $C_\mu(\mathbf{A}, f, \vec{x})$ is a lower bound for the μ-complexity of any uniform process which computes f from the primitives of $\mathbf{A}$. It is in this sense that $\boldsymbol{\beta}_{f,\mathbf{A}}$ is *the best* uniform process which computes f. The results in Section 5 imply then that this is a lower bound for the μ-complexity of any standard computation model which computes f in $\mathbf{A}$, and the discussion in Section 2A suggests that it is, in fact, an *absolute* (*logical*) *lower bound* which restricts all algorithms that compute f from the primitives of $\mathbf{A}$.

Moreover, the definitions yield a purely algebraic method for deriving these lower bounds:

Lemma 2.4 (The Homomorphism Test). *Suppose μ is a substructure norm on a Φ-structure $\mathbf{A}$, $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, and*

(34) *for every finite $\mathbf{U} \subseteq_p \mathbf{A}$ which is generated by $\vec{x}$,*

$$\Big(f(\vec{x}) \in U_s \ \&\ \mu(\mathbf{U}, \vec{x}) < m\Big) \Longrightarrow (\exists \pi : \mathbf{U} \to \mathbf{A})[f(\pi(\vec{x})) \neq \pi(f(\vec{x}))];$$

then $C_\mu(f, \mathbf{A}, \vec{x}) \geq m$.

2I. Intrinsic complexities. For lack of a better name, we will call $C_\mu(\mathbf{A}, f, \vec{x})$ the *intrinsic μ-complexity* of f (at $\vec{x}$ in $\mathbf{A}$). It records the μ-smallest "piece of $\mathbf{A}$" that is needed to determine the value $f(\vec{x})$—not necessarily to compute it effectively; and it is most useful when it agrees (absolutely or up to some multiplicative constant) with the μ-complexity of some known algorithm.

Most of the examples in the next two sections will be about the most important special cases of intrinsic complexity for which we have lower bounds, when $\mu(\mathbf{U}, \vec{x})$ is $\text{depth}(\mathbf{U}, \vec{x})$, $\text{size}(\mathbf{U})$ or $\text{calls}(\mathbf{U} \upharpoonright \Phi_0)$, in symbols

$$\begin{aligned}
\text{depth}(\mathbf{A}, f, \vec{x}) &= \min\{\text{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\},\\
\text{size}(\mathbf{A}, f, \vec{x}) &= \min\{\text{size}(\mathbf{U}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\},\\
\text{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) &= \min\{\text{calls}(\mathbf{U} \upharpoonright \Phi_0) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}.
\end{aligned}$$

We set $\text{calls}(\mathbf{A}, f, \vec{x}) = \text{calls}_\Phi(\mathbf{U}, f, \vec{x})$ and note that by Lemma 2.2,

$$\text{depth}(\mathbf{A}, f, \vec{x}) \leq \text{size}(\mathbf{A}, f, \vec{x}) \leq \text{calls}(\mathbf{A}, f, \vec{x}).$$

[9]This defines $C_\mu(\mathbf{A}, f, \vec{x})$ only when f has a certificate at $\vec{x}$. To simplify things in the Introduction, we set $C_\mu(\mathbf{A}, f, \vec{x}) = \infty$ in the alternative, but these values are, of course, irrelevant for anything we do with these complexity measures.

2J. Deterministic uniform processes. An n-ary uniform process of a structure $\mathbf{A}$ is *deterministic* if it satisfies the following, stronger form of the Finiteness Axiom as expressed in (20):

$$\overline{\alpha}(\vec{x}) = w \Longrightarrow (\exists \mathbf{U} \subseteq_p \mathbf{A})\Big(\mathbf{U} \vdash_c \alpha(\vec{x}) = w] \\ \& \text{ (for all } \mathbf{V} \subseteq_p \mathbf{A})[\mathbf{V} \vdash_c \alpha(\vec{x}) = w \Longrightarrow \mathbf{U} \subseteq_p \mathbf{V}]\Big), \tag{35}$$

i.e., if whenever $\overline{\alpha}(\vec{x})\downarrow$, then there is a unique, $\subseteq_p$-least "abstract computation" of $\overline{\alpha}(\vec{x})$ by α. The notion is natural, and we will note in Section 5 that every concrete deterministic algorithm induces a deterministic uniform process. We put it down here for completeness, but we have no real understanding of deterministic uniform processes and no methods for deriving lower bounds for them which are better (larger) that the lower bounds for all uniform processes which compute the same function. For example, we will note in Section 5 that *there is no best deterministic uniform* $\mathbf{A}$*-process* for every $f : A^n \rightharpoonup A$, simply because there may be two deterministic algorithms which compute the same function from the same primitives in essentially different ways.

§3. Coprimeness in $\mathbb{N}$.

§3. Coprimeness in $\mathbb{N}$. We discuss here four results on the intrinsic complexity of coprimeness (and related problems), from various primitives, which include division with remainder but not multiplication. Three of them are somewhat more abstract versions of results in van den Dries and Moschovakis [2004, 2009], and the fourth is in Pratt's unpublished [2008].[10]

3A. The binary (Stein) algorithm. This well-known algorithm computes the *greatest common divisor* $\gcd(x, y)$ from relatively simple primitives, based on the following, easy fact:

Proposition 3.1 (Stein [1967], Knuth [1973], Vol. 2, Sect. 4.5.2). *The gcd satisfies the following recursive equation for $x, y \geq 1$, using which it can be computed using $O(\max\{\log x, \log y\})$ calls to its primitives:*

$$\gcd(x,y) = \begin{cases} x & \text{if } x = y, \\ 2\gcd(\mathrm{iq}_2(x), \mathrm{iq}_2(y)) & \text{ow., if Parity}(x) = \text{Parity}(y) = 0, \\ \gcd(\mathrm{iq}_2(x), y) & \text{ow., if Parity}(x) = 0, \text{Parity}(y) = 1, \\ \gcd(x, \mathrm{iq}_2(y)) & \text{ow., if Parity}(x) = 1, \text{Parity}(y) = 0, \\ \gcd(x \dot{-} y, y) & \text{ow., if } x > y, \\ \gcd(x, y \dot{-} x) & \text{otherwise.} \end{cases}$$

Here $x \dot{-} y$ is *arithmetic subtraction* (set $= 0$ if $x < y$), and the complete list of primitives of the Stein algorithm is

$$\mathbf{StPrim} = \{0, 1, =, <, \text{parity}, \text{em}_2, \text{iq}_2, \dot{-}\}. \tag{36}$$

We set

$$\mathbf{N}_{\text{st}} = (\mathbb{N}, \mathbf{StPrim}).$$

[10]The relation of these results to the earlier Mansour, Schieber, and Tiwari [1991a, 1991b] and Meidânis [1991] is discussed in van den Dries and Moschovakis [2004, 2009].

The Stein algorithm decides the coprimeness relation in $O(\max(\log x, \log y))$, steps, by computing $d = \gcd(x, y)$ and then testing whether $d = 1$. The key to its optimality is the following[11]

LEMMA 3.2. *If $b > 2$, $a = b^2 - 1$ and $m < \frac{1}{10}\log b$, then every member x of $G_m(\mathbf{N}_{\rm st}, a, b)$ can be expressed uniquely in the form*

$$x = \frac{x_0 + x_1 a + x_2 b}{2^m} \quad \text{with } x_i \in \mathbb{Z}, |x_i| \le 2^{2m} \text{ for } i \le 2.$$

PROOF. The fact that every $x \in G_m(\mathbf{N}_{\rm st}, a, b)$ can be expressed in this form needs none of the hypotheses and can be checked by a simple induction. For the uniqueness, check the result for $m = 0$, verify (easily) that if $m > 0$, then $m < \frac{1}{10}\log b \Longrightarrow 2^{2m+3} < b$, and then use this version of the hypothesis to derive a contradiction from the assumption

$$(x_0 - x_1) + x_1 b^2 + x_2 b = (x_0' - x_1') + x_1' b^2 + x_2' b \text{ but } x_i \ne x_i' \text{ for some } i. \quad \dashv$$

THEOREM 3.3 (van den Dries and Moschovakis [2004]). *If $b > 2$ and $a = b^2 - 1$, then*

$$\text{depth}(\mathbf{N}_{\rm st}, \perp\!\!\!\perp, a, b) \ge \frac{1}{10}\log a. \tag{37}$$

It follows that if $\alpha_{\rm st}$ is the Stein algorithm for coprimeness, then for some K and all $b > 2$, $a = b^2 - 1$,

$$\text{calls}(\alpha_{\rm st}, a, b) \le K\text{depth}(\mathbf{N}_{\rm st}, \perp\!\!\!\perp, a, b) \le K\text{calls}(\mathbf{N}_{\rm st}, \perp\!\!\!\perp, a, b). \tag{38}$$

PROOF. Notice first that $a \perp\!\!\!\perp b$, because $1 = b \cdot b - a$. Since

$$\text{depth}(\mathbf{U}, a, b) \le m \Longrightarrow \mathbf{U} \subseteq_p \mathbf{G}_m(\mathbf{N}_{\rm st}, a, b),$$

to infer the conclusion of the theorem by the Homomorphism Test, it is enough to show that if $m < \frac{1}{10}\log a$, then there is a homomorphism $\pi : \mathbf{G}_m(\mathbf{N}_{\rm st}, a, b) \to \mathbf{N}_{\rm st}$ such that $\pi(a)$ and $\pi(b)$ are not coprime.

Define $\pi : G_m(\mathbf{N}_{\rm st}, a, b) \to \mathbb{Q}$ by

$$\pi\Big(\frac{x_0 + x_1 a + x_2 b}{2^m}\Big) = \frac{x_0 + x_1 \lambda a + x_2 \lambda b}{2^m} \quad \text{where } \lambda = 1 + 2^m$$

using the unique representation of each x in the domain given by the Lemma, which applies because $m < \frac{1}{10}\log a \Longrightarrow 2^{2m+3} < b$. This is well defined, the choice of λ insures that $\pi(x) \in \mathbb{N}$ if $x \in \mathbb{N}$, and π easily preserves all the primitives in **StPrim**. So it is a homomorphism of $\mathbf{G}_m(\mathbf{N}_{\rm st}, a, b)$ into $\mathbf{N}_{\rm st}$ and $\pi(a) = \lambda a, \pi(b) = \lambda b$, so π does not respect coprimeness as required by the Homomorphism Test.

The second claim follows by combining (37) with the main property of the Stein algorithm, Proposition 3.1. $\dashv$

In fact, $\pi : \mathbf{G}_m(\mathbf{N}_{\rm st}, a, b) \rightarrowtail \mathbf{N}_{\rm st}$ is an embedding, since it respects the identity relation, as were all the homomorphisms constructed in van den Dries and Moschovakis [2004, 2009].

[11]We use real logarithms to the base 2, i.e.,

$$\log x = \text{ the unique } y \in \mathbb{R} \text{ such that } 2^y = x \quad (x > 0).$$

Generalizations. Simple adaptations of this method of constructing embeddings give $r \log a$ lower bounds for $\mathrm{depth}(\mathbf{N}_{\mathrm{st}}, R, a)$ on infinite sets, for many unary relations $R \subset \mathbb{N}$ which are "spoiled" when their argument is multiplied by a suitable $\lambda \in \mathbb{N}$, including

a is a prime, a is square-free, a is a perfect square.

See Section 3 of van den Dries and Moschovakis [2004] and especially Sections 6, 7 and 12 of van den Dries and Moschovakis [2009], where these results are extended to *Presburger structures* $\mathbf{A}_p = (\mathbb{N}, \mathbf{\Phi})$ with $\mathbf{\Phi}$ any finite set of *Presburger functions and relations.* A small but interesting variation of the method also yields that for every Presburger structure $\mathbf{A}_p$, some $r > 0$ and infinitely many (a, b),

$$\mathrm{depth}(\mathbf{A}_p, |, a, b) \geq r \log b,$$

where $a \mid b \iff$ *a is a factor of b.* In another direction, Busch [2007, 2009] derives by the same methods lower bounds for substantially more complex arithmetic relations and functions. Although not stated in this form, the proofs of these extensions and generalizations are also grounded on suitable versions of the Homomorphism Test, Lemma 2.4, and they establish log lower bounds for $\mathrm{depth}(\mathbf{A}, f, \vec{a})$ for the relevant f on an infinite number of inputs.

3B. Coprimeness from division. As we specified it in 2G, the Euclidean algorithm ε computes the greater common divisor function and decides the coprimeness relation in the structure $\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, \mathrm{eq}_0, \mathrm{eq}_1)$. Its complexity has been extensively analyzed, but for our purposes here it is enough to mention two, simple facts. First,

$$\text{for all } x \geq y \geq 2,\ \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, x, y) \leq 2 \log y \leq 2 \log x.$$

Second,

$$\text{for all } k \geq 2,\ \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, F_{k+1}, F_k) = k - 1, \tag{39}$$

where $\{F_k\}_k$ is the *Fibonacci sequence* defined by the recursion

$$F_0 = 0, F_1 = 1, F_{k+2} = F_k + F_{k+1}; \tag{40}$$

in fact by Lamé's Theorem,

$$\text{if } y \leq F_k \text{ and } k \geq 2, \text{ then for all } x \geq y, \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, x, y) \leq k - 1,$$

i.e., the Euclidean exhibits its least efficient behavior on successive Fibonacci numbers. Combining this with standard properties of the Fibonacci sequence (Vol. 2, Sect. 1.2.8 of Knuth [1973]), it is easy to compute a positive rational r_ε such that

$$\mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, F_{k+1}, F_k) \geq r_\varepsilon \log F_{k+1} \quad (k \geq 2). \tag{41}$$

This suggests the following, weak version of the (worst case) optimality of the Euclidean up to a multiplicative constant, among all uniform processes which decide coprimeness in $\mathbf{N}_\varepsilon$:

CONJECTURE 3.4 (Suboptimality of ε). *There is a rational number $r > 0$, such that for infinitely many $a \geq b \geq 1$,*

$$\mathrm{calls}_{\{\mathrm{rem}\}}(\mathbf{N}_\varepsilon, \perp\!\!\!\perp, a, b) \geq r \log a.$$

This was formulated (for algorithms) in van den Dries and Moschovakis [2004]. It is open and most likely requires some non-trivial arithmetic for its proof—if it is true. The main result of van den Dries and Moschovakis [2004] was a much weaker log log lower bound in the expansion of $\mathbf{N}_\varepsilon$ by the primitives of $\mathbf{N}_{\text{st}}$, $+$ and *division with remainder*,

$$\mathbf{N}_{\text{st}}[\div] = (\mathbf{N}_{\text{st}}, +, \text{rem}, \text{iq}) = (\mathbb{N}, 0, 1, =, <, \text{parity}, \text{em}_2, \text{iq}_2, +, \dot{-}, \text{rem}, \text{iq}).$$

We need three classical, elementary results from number theory, cf. Theorems 185, 191 and 177 of Hardy and Wright [1938].

(A) *For every irrational real number $\xi > 0$, there are infinitely many coprime numbers a, b such that*

$$\Big|\xi - \frac{a}{b}\Big| < \frac{1}{b^2}.$$

We call such pairs (a, b) *good approximations of ξ.*

(B) *Liouville's Theorem for degree 2: For every quadratic irrational ξ, there is a number $C > 0$ such that for all $x, y \in \mathbb{Z}$,*

$$\Big|\xi - \frac{x}{y}\Big| > \frac{1}{Cy^2},$$

where an irrational number is *quadratic* if it is a root of a quadratic equation with integer coefficients.

(C) *If $\xi > 1$ is a quadratic irrational, then there is a number $c = c(\xi) > 1$ such that every number interval $(2^k, 2^{ck})$ contains a good approximation (a, b) of ξ, i.e., $2^k < a, b < 2^{ck}$.*[12]

The restriction $\xi > 1$ is certainly not needed here or further on, but it simplifies some of the statements we want to make: it insures, for example, that for all but finitely many good approximations (a, b) of ξ, $a > b$.

Theorem 3.5 (van den Dries and Moschovakis [2004, 2009]). *If $\xi > 1$ is a quadratic irrational, then there is a rational number $r > 0$ such that for all but finitely many good approximations (a, b) of ξ,*

$$\text{depth}(\mathbf{N}[\div], \perp\!\!\!\perp, a, b) \geq r \log\log a. \tag{42}$$

Specifically, (42) holds for all (a, b) which satisfy Pell's equation $a^2 = 2b^2 + 1$ taking $\xi = \sqrt{2}$, and for all successive Fibonacci pairs (F_{k+1}, F_k) with $k \geq 3$ taking $\xi = \frac{1+\sqrt{5}}{2}$, both with the same constant $r = \frac{1}{10}$.

For the proof we need an analog of Lemma 3.2:

Lemma 3.6. *For every quadratic irrational $\xi > 1$, there is a number $\ell = \ell(\xi)$ such that for all but finitely many good approximations (a, b) of ξ and every $m < \frac{1}{2\ell}\log\log a$, every number in $G_m(\mathbf{N}[\div], a, b)$ can be expressed uniquely in the form*

$$x = \frac{x_0 + x_1 a + x_2 b}{x_3} \quad \textit{with } x_i \in \mathbb{Z}, |x_i| < 2^{2^{\ell m}} \textit{ for } i \leq 3.$$

[12]This follows easily from the fact that the continued fraction expansion of a quadratic irrational is *periodic*, Theorem 177 of Hardy and Wright [1938].

The proof is by induction on m and it involves a considerable amount of computation—but no serious number theory beyond Liouville's Theorem, which supplies the relevant ℓ. Two somewhat different arrangements of the computations are given in detail in van den Dries and Moschovakis [2004, 2009] for the specific case $\xi = \sqrt{2}$, but they can be extended routinely to arbitrary ξ.

Outline of proof of the Theorem. The argument has the same structure as that for Theorem 3.3, based on Lemma 3.6 rather than 3.2. We define $\pi : G_m(\mathbf{N}_{\rm st}[\div], a, b) \to \mathbb{N}$ by

$$\pi(x) = \frac{x_0 + x_1\lambda a + x_2\lambda b}{x_3} \quad \text{where } \lambda = 1 + a!,$$

using the unique representation supplied by the Lemma, check (by applying the Lemma again) that it is an embedding on $\mathbf{G}_m(\mathbf{N}_{\rm st}[\div], a, b)$ to $\mathbf{N}_{\rm st}[\div]$ which does not respect $a \perp\!\!\!\perp b$, and then appeal to the Homomorphism Test, which yields the required conclusion with $r = \frac{1}{2\ell}$. $\dashv$

Generalizations. As with the simpler $\mathbf{N}_{\rm st}$, the method extends easily to yield $\log\log$ lower bounds on depth$(\mathbf{N}_{\rm st}[\div], R, a)$ for many unary relations, including *a is a prime, a is square-free, a is a perfect square.* (In fact, the proofs are substantially easier for unary relations and do not require applications of Liouville's Theorem.) It also holds for expansions of $\mathbf{N}_{\rm st}[\div]$ by arbitrary Presburger primitives, and it can be extended to apply to substantially more inputs than good approximations of a quadratic irrational, cf. Theorems 3.9, 9.1 and (ii) of Theorem 12.2 in van den Dries and Moschovakis [2009]. On the other hand, the occurrence of the same constant $r = \frac{1}{10}$ in the theorem for both Pell pairs and successive Fibonacci numbers (and even Theorem 3.3!) is entirely an artifact of the proofs: it is not at all clear how one would compute "the least" r for which the theorem holds for a given ξ, although it might be possible and interesting to determine the best way that gives a relevant r from a constant C supplied by Liouville's Theorem for ξ.

3C. Pratt's non-deterministic algorithm for the gcd. The key facts about Pratts' *nuclid* (non-deterministic Euclidean) algorithm are that it is from the primitives of the Euclidean; it is no less efficient than the Euclidean ε on all inputs; and it realizes the lower bound predicted by Theorem 3.5 on successive Fibonacci numbers, beating the Euclidean on its worse behavior:

Theorem 3.7 (Pratt [2008]). *There is a non-deterministic version* ν (nuclid) *of the Euclidean algorithm with the following two properties:*

(1) *For all* $a \geq b \geq 1$, $\text{calls}(\nu, a, b) \leq \text{calls}(\varepsilon, a, b)$.
(2) *For some* K *and all* $k \geq 3$, $\text{calls}(\nu, F_{k+1}, F_k) \leq K \log\log F_{k+1}$.

Outline of proof. Nuclid is specified by a non-deterministic sequential machine with the following components:

(i) The *states* are all quadruples $(a, b, m, n) \in \mathbb{N}^4$.
(ii) The *input function* is $\text{input}(a, b) = (a, b, a, b)$.
(iii) The *terminal states* are those of the form $(a, b, m, 0)$.

(iv) There are three *transition functions*, as follows for $i \leq 2$:

$$\begin{aligned}\tau_i(a,b,m,n) = \text{if } (n \neq 0) \text{ then } (a,b,n,\sigma_i(a,b,m,n))\\ \text{else if } (n = 0 \;\&\; \text{rem}(a,m) \neq 0) \text{ then } (a,b,m,\text{rem}(a,m))\\ \text{else if } (n = 0 \;\&\; \text{rem}(b,m) \neq 0) \text{ then } (a,b,m,\text{rem}(b,m)),\end{aligned}$$

where

$$\sigma_0(a,b,m,n) = \text{rem}(m,n),\ \sigma_1(a,b,m,n) = \text{rem}(a,m),\ \sigma_2(a,b,m,n) = \text{rem}(b,m).$$

A *computation of* ν *with input* (a,b) is a maximal finite or infinite sequence of states

$$C \ : \ s_0 = (a,b,a,b) \to s_1 = (a,b,m_1,n_1) \to s_2 = (a,b,m_2,n_2) \to \cdots ,$$

such that for each k, $s_{k+1} = \tau_i(s_k)$ for some $i \leq 2$.

It is quite easy to check that every computation is finite with final state $(a,b,m,0)$ for some m, and (with a little more work, using the last two clauses in the transition functions) that $m = \text{gcd}(a,b)$; and then this computation of nuclid outputs m, if it is asked to compute $\text{gcd}(a,b)$ or checks whether $m = 1$ and gives the correct answer if it is asked to decide whether $a \perp\!\!\!\perp b$. It is clear that ν is an algorithm from the primitives of $\mathbf{N}_\varepsilon = (\mathbb{N}, \text{rem}, \text{eq}_0, \text{eq}_1)$, i.e., the input, output and all three transition functions are defined by terms of $\mathbf{N}_\varepsilon$ and the terminal states are similarly decided by an $\mathbf{N}_\varepsilon$-term. Each computation C determines a certain number of calls to the primitives $\text{calls}(\nu, C, a, b)$, and as usual with non-deterministic algorithms, we set

$$\text{calls}(\nu, a, b) = \min\{\text{calls}(\nu, C, a, b) : C \text{ is a computation}\}.$$

If a computation chooses τ_0 at every step, then it is exactly the computation that the deterministic Euclidean would produce, and so nuclid is certainly no less efficient than ε, on all inputs.

The main content of the theorem is that for some K, and every $k \geq 3$, there is a computation C such that

$$\text{calls}(\nu, C, F_{k+1}, F_k) \leq K \log\log F_{k+1}.$$

This requires some thinking and extensive use of the basic equations satisfied by the Fibonacci sequence, cf. Knuth [1973] as above. ⊣

Corollary 3.8. *For some* K *and all* $k \geq 3$,

$$\text{depth}(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, F_{k+1}, F_k) \leq \text{calls}(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, F_{k+1}, F_k) \leq K \log\log F_{k+1}.$$

Proof. We view ν as an algorithm of $\mathbf{N}_{\text{st}}[\div]$ which is an expansion of $\mathbf{N}_\varepsilon$. Let C be a nuclid computation on the input (F_{k+1}, F_k) with a least number of calls to the primitives, so that $\text{calls}(\nu, C, F_{k+1}, F_k) \leq K \log\log F_{k+1}$ by the theorem, and then construct from C a finite $\mathbf{U}_0 \subseteq_p \mathbf{N}_{\text{st}}[\div]$ exactly as we constructed $\mathbf{U}_0$ from a computation of the Euclidean in 2G. It follows by the same argument we gave in 2G that $\mathbf{U}_0 \Vdash_c^{\mathbf{N}_{\text{st}}[\div]} F_{k+1} \perp\!\!\!\perp F_k$; and then the Corollary follows by the definition of $\text{calls}(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, F_{k+1}, F_k)$. ⊣

Notes. The corollary implies that the conclusion of Theorem 3.5 is best possible from its hypotheses. It does not contradict Conjecture 3.4, because there may well be an $r > 0$ and infinitely many $a \geq b \geq 1$ such that

$$\text{calls}_{\{\text{rem}\}}(\nu, a, b) \geq r \log a,$$

and for all we know this may be true of all Pell pairs! The analysis of nuclid for arbitrary (a, b) appears to be quite difficult, and we do not know now any non-trivial results about it.

So there could be an entirely different approach to proving Conjecture 3.4 which does not involve the Fibonacci pairs. Another possibility is that the conjecture only holds for deterministic uniform processes, as we defined these in 2J:

Conjecture 3.9 (Deterministic suboptimality of ε). *There is a rational number $r > 0$, such that for every deterministic uniform process α of $\mathbf{N}_\varepsilon$ which decides coprimeness and for infinitely many $a \geq b \geq 1$,*

$$\text{calls}(\alpha, a, b) \geq r \log a.$$

This would be very interesting, especially if Conjecture 3.4 is false, so that the classical difference between determinism and non-determinism shows up at this very fundamental (and very low) complexity level. As we mentioned in 2J, we have no methods now for attacking this sort of problem.

3D. Non-uniform complexity on $\mathbb{N}$. When we look for robust lower bounds for deciding a given number theoretic relation R, it is natural to restrict the problem to some finite set of inputs, typically the *N-bit numbers*

$$[0, 2^N) = \{x \in \mathbb{N} : x < 2^N\} \quad (N > 0).$$

This "non-uniform" approach, allows us in effect to use a different algorithm to decide R on distinct initial segments of $\mathbb{N}$: the important questions are not about the behavior of algorithms which decide R on all of $\mathbb{N}$ on specific inputs, but about the behavior of algorithms which decide R only on N-bit numbers, as N becomes large. The basic definition of intrinsic complexities make sense on finite structures and so they can be used to derive lower bounds for non-uniform complexity—but there is "less room" to exploit the Homomorphism Test and so the specific applications pose problems. We will review here just two examples which illustrate the situation.

For each structure $\mathbf{A} = (\mathbb{N}, \mathbf{\Phi})$ on $\mathbb{N}$, each relation $R \subseteq \mathbb{N}^n$ and each substructure norm μ on $\mathbf{A}$, let

$$C_\mu(\mathbf{A}, R, 2^N) = \max\{C_\mu(\mathbf{A} \restriction [0, 2^N), R, \vec{x}) : x_1, \dots, x_n < 2^N\}. \tag{43}$$

Mostly we are interested in $\text{depth}(\mathbf{A}, R, 2^N)$, $\text{size}(\mathbf{A}, R, 2^N)$ and $\text{calls}(\mathbf{A}, R, 2^N)$, cf. 2I.

The results about $\mathbf{N}_{\text{st}}$ hold basically as before:

Theorem 3.10 (van den Dries and Moschovakis [2009]). *For some $r > 0$ and all sufficiently large N,*

$$\text{depth}(\mathbf{N}_{\text{st}}, \perp\!\!\!\perp, 2^N) \geq rN. \tag{44}$$

Proof. To infer this from the Homomorphism Test 2.4 by adapting the proof of Theorem 3.3, we need to find an $r > 0$ such that for all sufficiently large N, there is a $b > 2$ and an $m > 0$ satisfying the inequalities

$$2^{2m+3} < b < 2^N,\ a = b^2 - 1 < 2^N,\ \frac{x_0 + x_1\lambda a + x_2\lambda b}{2^m} < 2^N\ (|x_i| \leq 2^{2m})$$
$$\text{and } \frac{1}{10}\log b > rN$$

with $\lambda = 1 + 2^m$. The first three of these allow us to repeat the argument in Theorem 3.3 within $\mathbf{N}_{\text{st}} \upharpoonright [0, 2^N)$ and construct an embedding

$$\pi : \mathbf{G}_m(\mathbf{N}_{\text{st}} \upharpoonright [0, 2^N), a, b) \to \mathbf{N}_{\text{st}} \upharpoonright [0, 2^N)$$

such that $\pi(a) = \lambda a, \pi(b) = \lambda b$; π then does not respect $a \perp\!\!\!\perp b$, and so

$$\text{depth}(\mathbf{N}_{\text{st}}, \perp\!\!\!\perp, 2^N) \geq \text{depth}(\mathbf{N}_{\text{st}} \upharpoonright [0, 2^N), \perp\!\!\!\perp, a, b) \geq \frac{1}{10}\log b.$$

The last inequality insures that $\frac{1}{10}\log b > rN$ as required.

It is quite easy to check that these conditions are satisfied if we take

$$N > 16,\ m = \max\{k : 16k < N\},\ a = 2^{6m},\ r = \frac{11}{170},$$

but these specific constants are artifacts of one way to do the computation and there are surely better ways, probably leading to better values. The key idea is that if a is "substantially smaller" than 2^N (but not too small), then b, λ and the values of π are all below 2^N; we then choose the largest a which is "substantially smaller" than 2^N, which allows us to express the derived lower bound in terms of N. ⊣

All the lower bounds in $\mathbf{N}_{\text{st}}$ listed in 3A have similar, non-uniform versions with a linear lower bound for depth$(\mathbf{N}_{\text{st}}, R, 2^N)$ and by the same, mild modification of their proofs. In particular, no non-trivial results from number theory are needed—except, minimally, for primality: to show that for any set $\mathbf{\Phi}$ of Presburger primitives depth$((\mathbb{N}, \mathbf{\Phi}), \text{Prime}, 2^N) \geq rN$ for some r and all sufficiently large N, we need to use some estimate about the distribution of the primes not being "too thin". *Every interval $(l, 2l)$ with $l \geq 3$ contains a prime* (Bertrand's postulate, Theorem 418 of Hardy and Wright [1938]) suffices.

The corresponding generalizations of the results for $\mathbf{N}_{\text{st}}[\div]$ are not quite that simple, because the constant $\lambda = 1 + a!$ that we used in the proof of Theorem 3.5 is too large: a direct adaptation of that proof to the non-uniform case leads to a $\log\log N$ lower bound for depth$(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, 2^N)$ which is way too low. In fact, it does not seem possible to get a decent lower bound for depth$(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, 2^N)$ with this method, but a small adjustment yields a lower bound for the size- and hence the calls- intrinsic complexities:

Theorem 3.11 (van den Dries and Moschovakis [2009]). *For some rational number $r > 0$ and all sufficiently large N,*

$$\text{calls}(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, 2^N) \geq \text{size}(\mathbf{N}_{\text{st}}[\div], \perp\!\!\!\perp, 2^N) \geq r\log N. \tag{45}$$

Outline of proof. Fix a quadratic irrational $\xi > 1$ (for simplicity), e.g., $\xi = \sqrt{2}$, choose $c = c(\xi)$ by (C) in 3B and set

$$N > 4c,\ k = \max\{s : cs < \frac{N}{2}\},\ 2^k < b < a < 2^{ck}, \tag{46}$$

where (a, b) is a good approximation of ξ. Note that by the definition of k,

$$c(k+1) \geq \frac{N}{2} \text{ and so } k > \frac{1}{4c}N \tag{47}$$

using the hypothesis on N.

Fix any $\mathbf{U} \subseteq_p \mathbf{A} = \mathbf{N}_{\rm st}[\div] \restriction [0, 2^N)$ such that $\mathbf{U} \Vdash^{\mathbf{A}}_c a \perp\!\!\!\perp b$ and set

$$\nu = \mathrm{size}(\mathbf{U}, a, b) = |U_{\rm vis}|, \quad m = \mathrm{depth}(\mathbf{U}, a, b).$$

It is enough to find some $r > 0$ independent of $\mathbf{U}$ such that $\nu \geq r \log N$.

Let $\ell = \ell(\xi)$ be as in Lemma 3.6.

Case 1, $2^{2^{2\ell m}} \geq a$. By the hypothesis on a, we have $2^{2^{2\ell m}} \geq a > 2^k$, and so $2^{2\ell m} > k > \frac{1}{4c}N$, from which we get

$$\nu \geq m > r_1 \log N \text{ (for sufficiently large } N) \tag{48}$$

with a suitable $r_1 = r_1(\ell, k) = r_1(\xi)$.

Case 2, $2^{2^{2\ell m}} < a$. In this case Lemma 3.6 applies, and every $x \in G_m(\mathbf{N}_{\rm st}[\div], a, b)$ can be written uniquely in the form

$$x = \frac{x_0 + x_1 a + x_2 b}{x_3} \quad \text{with } x_i \in \mathbb{Z}, |x_i| < 2^{2^{\ell m}} \text{ for } i \leq 3. \tag{49}$$

In particular, this holds for all $x \in U \subseteq G_m(\mathbf{N}_{\rm st}[\div], a, b)$, and we can use it to define functions $\pi_\lambda : U \to \mathbb{Q}$ (the rational numbers) by

$$\pi_\lambda(x) = \frac{x_0 + x_1 \lambda a + x_2 \lambda b}{x_3}.$$

For most λ, the values $\pi(x)$ will not be natural numbers; this was the point of choosing $\lambda = 1 + a!$ in the proof of Theorem 3.5, which, however, will now give values outside $[0, 2^N)$. Instead of this, we use the uniqueness in (49), we set

$$\mathrm{denom}(x) = x_3 \quad (x \in U), \quad \lambda = 1 + \textstyle\prod_{x \in U} \mathrm{denom}(x)$$

and it follows easily that $\pi(x) \in \mathbb{N}$ for every $x \in U$. Moreover, $\pi : \mathbf{U} \to \mathbf{N}_{\rm st}[\div]$ is a homomorphism by Theorem 3.5, and λ is not too big,

$$\lambda = 1 + \left(2^{2^{\ell m}}\right)^\nu = 1 + 2^{\nu 2^{\ell m}} < 2^{2^{2\ell\nu + 1}}.$$

We can also compute an upper bound on $\pi(x)$ for each $x \in U$:

$$\pi(x) \leq \Big| x_0 + x_1 \lambda a + x_2 \lambda b \Big| \leq 2^{2^{\ell m}} + 2^{2^{\ell m}} \lambda a + 2^{2^{\ell m}} \lambda b$$
$$\leq 2^{2^{\ell\nu}} + 2 \cdot 2^{2^{\ell\nu}} \cdot 2^{2^{2\ell\nu+1}} \cdot 2^{ck} \leq 3 \cdot 2^{2^{\ell\nu + 2\ell\nu + 1} + \frac{N}{2}} \leq 2^{2^{3\ell\nu+1} + \frac{N}{2} + 2} \leq 2^{2^{4\ell\nu} + \frac{N}{2} + 2}.$$

If this last bound is $< 2^N$, then $\pi : \mathbf{U} \to \mathbf{N}_{\rm st}[\div] \restriction [0, 2^N)$ is a homomorphism which does not respect $a \perp\!\!\!\perp b$, contrary to our assumption about $\mathbf{U}$; it follows that

$2^{4\ell\nu} + \frac{N}{2} + 2 \geq N$, and so $4\ell\nu \geq \log(\frac{N}{2} - 2)$ from which we can easily find some $r_2 > 0$ such that

$$\nu \geq r_2 \log N \text{ (for sufficiently large } N).$$

The required r for the proof is the lesser of r_1 and r_2. ⊣

Non-uniform versions of the other intrinsic lower bounds mentioned above can be checked by the same method, e.g., for some $r > 0$ and all sufficiently large N, $\text{size}(\mathbf{N}_{\text{st}}[\div], \text{Prime}, 2^N) \geq r \log N$.

§4. Polynomial nullity (0-testing). In this section we will prove four results on the intrinsic complexity of evaluation and 0-testing of polynomials, all of them establishing the optimality or "near-optimality" of Horner's rule from various primitives for "generic" inputs. Polynomial evaluation is perhaps the simplest problem in algebraic complexity, and it has been much studied since its formulation as a complexity problem by Ostrowski [1954]; here we are concerned with 0-testing, a plausibly easier problem which has received considerably less attention.

For any field F, Horner's rule computes the value

$$\chi(b) = \sum_{i \leq n} a_i b^i = a_0 + a_1 b + \cdots + a_n b^n$$

of a polynomial $\chi(x)$ using no more than n multiplications and n additions in F as follows:

$$\begin{aligned}
\chi_0(b) &= a_n, \\
\chi_1(b) &= a_{n-1} + b\chi_0(b) = a_{n-1} + a_n b \\
&\vdots \\
\chi_j(b) &= a_{n-j} + b\chi_{j-1}(b) = a_{n-j} + a_{n-j+1}b + \cdots + a_n b^j \\
&\vdots \\
\chi(b) = \chi_n(b) &= a_0 + b\chi_{n-1}(b) = a_0 + a_1 b + \cdots + a_n b^n.
\end{aligned}$$

For certain values of the coefficients, using division and subtraction might lead to a more efficient computation because of identities like

$$1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1},$$

and we also need the equality relation when we want to decide whether $\chi(b) = 0$; so it is natural to consider the optimality of Horner's rule from the primitives of the expansion

$$\mathbf{F} = (F, 0, 1, +, -, \cdot, \div, =) \tag{50}$$

of the field structure by $=$. We will use throughout the standard notation

$$N_F(a_0, a_1, \ldots, a_n, b) \iff a_0 + a_1 b + \cdots + a_n b^n = 0,$$

where F is a field and $n \geq 1$.

A *partial field homomorphism*

$$\pi : F_1 \rightharpoonup F_2$$

on one field to another is a partial function whose domain of convergence is a subfield $F_1' \subseteq F_1$ and which respects (as usual) the field operations.

For any field F and indeterminates $\vec{u} = u_1, \dots , u_k$, $F[\vec{u}]$ is the ring of all polynomials with coefficients in F and $F(\vec{u})$ is the field of all rational functions in $\vec{u}$ with coefficients in F. If $\psi(v, \vec{u}) \in F(v, \vec{u})$ is a rational function, then the *substitution* $v := \psi(v, \vec{u})$ induces a partial field homomorphism

$$\pi\Big(\frac{\chi_n(v, \vec{u})}{\chi_d(v, \vec{u})}\Big) = \frac{\chi_n(\psi(v, \vec{u}), \vec{u})}{\chi_d(\psi(v, \vec{u}), \vec{u})} \quad \Big(\frac{\chi_n(v, \vec{u})}{\chi_d(v, \vec{u})} \in F(v, \vec{u})\Big) \tag{51}$$

whose domain of convergence is the field

$$\Big\{\frac{\chi_n(v, \vec{u})}{\chi_d(v, \vec{u})} : \chi_d(\psi(v, \vec{u}), \vec{u}) \neq 0\Big\} \subseteq F(v, \vec{u}).$$

We will refer to π as "the substitution" $v := \psi(v, \vec{u})$, and the only partial field homomorphisms we will need are compositions of such substitutions.

4A. Horner's rule is $\{\cdot, \div\}$-optimal for nullity. The $\{\cdot, \div\}$-optimality of Horner's rule for polynomial evaluation was proved by Pan [1966]. Pan worked with *computation sequences* (which we will define in Section 5) and introduced the *method of substitution*, i.e., the use of partial field homomorphisms induced by substitutions as above.

By Horner's rule, for all fields F and all $\vec{a} \in F^n$,

$$\text{calls}_{\{\cdot,\div\}}(\mathbf{F}, N_F, \vec{a}) \leq n.$$

THEOREM 4.1 (Bürgisser and Lickteig [1992][13]). *If F is a field of characteristic* 0*, $n \geq 1$, and $a_0, \dots , a_n, b \in F$ are algebraically independent* (over the prime field $\mathbb{Q}$)*, then*

$$\text{calls}_{\{\cdot,\div\}}(\mathbf{F}, N_F, a_0, \dots , a_n, b) = n. \tag{52}$$

In particular, (52) *holds for the reals $\mathbb{R}$ and the complexes $\mathbb{C}$ with algebraically independent $a_0, a_1, \dots , a_n, b$.*

To show the needed inequality $\text{calls}_{\{\cdot,\div\}}(\mathbf{F}, N_F, a_0, \dots , a_n, b) \geq n$ by the Homomorphism Test 2.4, we must construct for every algebraically independent tuple $\vec{a}, b \in F^{n+2}$ and every finite substructure $\mathbf{U} \subseteq_p \mathbf{F}$ such that $\mathbf{U} \Vdash_c^{\mathbf{F}} \neg N_F(\vec{a}, b)$ and $\text{calls}_{\{\cdot,\div\}}(\mathbf{U}, \vec{a}, b) < n$ a homomorphism $\pi : \mathbf{U} \to \mathbf{F}$ which does not respect $N_F(\vec{a}, b)$; and since eqdiag($\mathbf{U}$) may contain all true non-equalities $u \neq v$ for $u, v \in U$, we must make sure that π is an embedding. The construction is by induction on n, but we need a very strong "induction loading device" for it to go through. The appropriate lemma is an elaboration of the construction in Winograd [1967, 1970], which extends and generalizes Pan's results.

We will need a simple, preliminary fact.

LEMMA 4.2. *If F is infinite, $\phi_1(\vec{u}), \phi_2(\vec{u}), \phi(\vec{u}) \in F(\vec{u})$ with $\phi_1(\vec{u}) \neq 0$ or $\phi_2(\vec{u}) \neq 0$ and U is any finite subset of $F(v, \vec{u})$, then there is some $\overline{f} \in F$ such*

[13]The proof in Bürgisser and Lickteig [1992] is for algebraic decision trees, i.e., (branching) computation sequences with equality tests.

that the (partial field homomorphism $\rho_{\overline{f}}$ induced by the) *substitution*

$$v := \rho_{\overline{f}}(v) = \overline{f}\Big(\phi_1(\vec{u}) + \phi_2(\vec{u})v\Big) + \phi(\vec{u})$$

is totally defined and injective on U.

Proof. It is convenient to prove first a

Sublemma 4.2.1. *If $U' \subset F[v, \vec{u}]$ is any finite set of polynomials, then there is some $\overline{f} \in F$ such that*

$$\Big(\chi(v,\vec{u}) \in U' \text{ and } \chi(v,\vec{u}) \neq 0\Big) \Longrightarrow \chi(\rho_{\overline{f}}(v), \vec{u}) \neq 0.$$

Proof of the Sublemma. Write each $\chi(v,\vec{u}) \in U'$ as a polynomial in v,

$$\chi(v,\vec{u}) = \chi_0(\vec{u}) + \chi_1(\vec{u})v + \cdots + \chi_l(\vec{u})v^l,$$

let $\overline{F(\vec{u})}$ be the algebraic closure of $F(\vec{u})$ and consider the complete factorization

$$\chi(v,\vec{u}) = \chi_l(\vec{u})(v - \alpha_1)\cdots(v - \alpha_l)$$

of $\chi(v,\vec{u})$ in $\overline{F(\vec{u})}[v]$. Now

$$\begin{aligned}\chi(\rho_{\overline{f}}(v), \vec{u}) = \chi_l(\vec{u})(&\overline{f}\phi_1(\vec{u}) + \overline{f}\phi_2(\vec{u})v + \phi(\vec{u}) - \alpha_1)\\ &\cdots(\overline{f}\phi_1(\vec{u}) + \overline{f}\phi_2(\vec{u})v + \phi(\vec{u}) - \alpha_l)\end{aligned}$$

with each $\alpha_i \in \overline{F(\vec{u})}$. A factor in this product vanishes only if

$$\overline{f}\phi_1(\vec{u}) + \phi(\vec{u}) - \alpha_i = 0 \text{ and } \overline{f}\phi_2(\vec{u}) = 0,$$

and if we choose any $\overline{f} \neq 0$, this can only happen if $\phi_2(\vec{u}) = 0$. But then the hypothesis implies that $\phi_1(\vec{u}) \neq 0$ and so $\overline{f}\phi_1(\vec{u}) + \phi(\vec{u}) - \alpha_i = 0$ can happen for at most one value of $\overline{f}$. The conclusion is then satisfied by any (non-zero) $\overline{f}$ which is different from the (at most) one value determined from each α_i, for each of the $\chi(v,\vec{u}) \in U'$. ⊣ (Sublemma)

We now fix a specific representation of the form

$$\chi(\vec{u}) = \frac{\chi_n(\vec{u})}{\chi_d(\vec{u})} \quad (\chi_d(\vec{u}) \neq 0) \tag{53}$$

for each $\chi(v,\vec{u}) \in U$, and we apply this Sublemma to the finite set U' comprising all polynomials in one of the forms

$$(i)\ \chi_d(v,\vec{u}), \qquad (ii)\ \chi_n(v,\vec{u})\chi'_d(v,\vec{u}) - \chi'_n(v,\vec{u})\chi_d(v,\vec{u})$$

with $\chi(v,\vec{u}), \chi'(v,\vec{u}) \in U$. Clearly $\rho_{\overline{f}}(\chi(v,\vec{u}))$ is defined for every $\chi(v,\vec{u}) \in U$ because we put in U' the polys in (i), and so it is enough to check that it is injective on U; and this is true because

$$\begin{aligned}\rho_{\overline{f}}\Big(\frac{\chi_n(v,\vec{u})}{\chi_d(v,\vec{u})}\Big) = \rho_{\overline{f}}\Big(\frac{\chi_n'(v,\vec{u})}{\chi_d'(v,\vec{u})}\Big)&\\ \Longrightarrow \rho_{\overline{f}}\Big(\chi_n(v,\vec{u})\chi_d'(v,\vec{u}) - \chi_d(v,\vec{u})\chi_n'(v,\vec{u})\Big) = 0&\\ \Longrightarrow \chi_n(v,\vec{u})\chi_d'(v,\vec{u}) - \chi_d(v,\vec{u})\chi_n'(v,\vec{u}) = 0&\\ \Longrightarrow \frac{\chi_n(v,\vec{u})}{\chi_d(v,\vec{u})} = \frac{\chi_n'(v,\vec{u})}{\chi_d'(v,\vec{u})}&\end{aligned}$$

where the inclusion in U' of all the polys in (ii) and the the fact that $\rho_{\overline{f}}$ is injective (guaranteed by the Sublemma) are used in the second implication. $\dashv$

We write $\{\cdot, \div\}$ for *multiplications and divisions*, and we define the *trivial* $\{\cdot, \div\}$ (relative to $z, \vec{x}, y$) by[14]

$$\begin{aligned} a \cdot b = c \text{ is trivial } &\text{ if } a \in F \text{ or } b \in F \text{ or } a, b \in F(y);\\ a \div b = c \text{ is trivial } &\text{ if } b \in F \text{ or } a, b \in F(y).\end{aligned}$$

LEMMA 4.3. *Suppose F is an infinite field, $n \geq 1$, z, $\vec{x} = x_1, \dots, x_n$, y are distinct indeterminates,*

$$\mathbf{U} \subseteq_p \mathbf{F}(z, \vec{x}, y) = (F(z, x_1, \dots, x_n, y), 0, 1, +, -, \cdot, \div, =)$$

is finite, and $\psi_1, \dots, \psi_n \in F(y)$ so that the following conditions hold:

(1) *$\mathbf{U}$ is generated by $(F \cap U) \cup \{z, \vec{x}, y\}$.*
(2) *For any $f_1, \dots, f_n \in F$, if $f_1\psi_1 + \cdots + f_n\psi_n \in F$, then $f_1 = \cdots = f_n = 0$.*
(3) *There are no more than $n-1$ non-trivial $\{\cdot, \div\}$ in* eqdiag($\mathbf{U}$).

Then there is a partial field homomorphism

$$\pi : F(z, \vec{x}, y) \rightharpoonup F(\vec{x}, y)$$

which is the identity on $F(y)$ and satisfies

$$\pi(z) = \pi(x_1)\psi_1 + \cdots + \pi(x_n)\psi_n. \tag{54}$$

Moreover, π is total and injective on U, so that its restriction to U defines an embedding $\pi \restriction U : \mathbf{U} \rightarrowtail \mathbf{F}(\vec{x}, y)$.

PROOF OF THEOREM 4.1 FROM LEMMA 4.3. The hypothesis implies that

$$a_0 + a_1 b + \cdots + a_n b^n \neq 0,$$

and so by the Homomorphism Test 2.4 it is enough to show that for every finite $\mathbf{U} \subseteq_p \mathbf{F}$ which is generated by $a_0, \vec{a} = a_1, \dots, a_n, b$, if $|\text{eqdiag}(\mathbf{U} \restriction \{\cdot, \div\})| < n$, then there is an embedding $\pi : \mathbf{U} \rightarrowtail \mathbf{F}$ such that

$$\pi(a_0) + \pi(a_1)\pi(b) + \cdots + \pi(a_n)\pi(b)^n = 0. \tag{55}$$

We may assume that $0 - a_0 = -a_0, 0 - (-a_0) = a_0 \in \text{eqdiag}(\mathbf{U})$, by adding them if necessary, so that $\mathbf{U}$ is also generated by $-a_0, \vec{a}, b$. Moreover, $\mathbf{U} \subseteq_p \mathbb{Q}(-a_0, \vec{a}, b)$ and $\mathbb{Q}(-a_0, \vec{a}, b)$ is isomorphic with $\mathbb{Q}(z, \vec{x}, y)$ by the "relabelling" isomorphism ρ generated by $-a_0 \mapsto z, a_i \mapsto x_i, b \mapsto y$. The required π is now constructed by applying Lemma 4.3 to $\mathbf{U}' = \rho[\mathbf{U}]$ and $\psi_1 = y, \psi_2 = y^2, \dots, \psi_n = y^n$, carrying

[14]We are following closely the terminology and notation of Winograd [1967].

the embedding it gives back to an embedding $\pi : \mathbf{U} \rightarrowtail \mathbf{F}(\vec{a}, b)$ and noticing that $\pi(b) = b$. ⊣

Proof of Lemma 4.3 is by induction on n, but it is useful to consider first a case which covers the basis and also arises in the induction step.

Preliminary case: there are no non-trivial $\{\cdot, \div\}$ *in* $\mathbf{U}$. It follows that every $X \in U$ is uniquely of the form

$$X = f_0 z + \textstyle\sum_{1 \le i \le n} f_i x_i + \phi(y) \tag{56}$$

with $f_i \in F, \phi(y) \in F(y)$. If π is the partial field homomorphism induced by the substitution

$$z \mapsto \textstyle\sum_{1 \le i \le n} x_i \psi_i,$$

then π is the identity on $F(\vec{x}, y)$ and it is total on U, because the only $\{\cdot, \div\}$ in $\mathbf{U}$ are with both arguments in $F(y)$ or one of them in F. So it is enough to check that it is injective on the set of all elements of the form (56) and that it satisfies (54). To check injectivity, suppose that

$$\begin{aligned}\pi(X) = f_0 \Big(\textstyle\sum_{1 \le i \le n} x_i \psi_i \Big) &+ \textstyle\sum_{1 \le i \le n} f_i x_i + \phi(y) \\ = f_0' \Big(\textstyle\sum_{1 \le i \le n} x_i \psi_i \Big) &+ \textstyle\sum_{1 \le i \le n} f_i' x_i + \phi'(y) = \pi(X')\end{aligned}$$

so that

$$\begin{aligned}(f_0 - f_0') \textstyle\sum_{1 \le i \le n} x_i \psi_i &+ \textstyle\sum_{1 \le i \le n} (f_i - f_i') x_i + (\phi(y) - \phi'(y)) \\ &= \textstyle\sum_{1 \le i \le n} \Big((f_0 - f_0')\psi_i + (f_i - f_i') \Big) x_i + (\phi(y) - \phi'(y)) = 0.\end{aligned}$$

This yields $\phi(y) = \phi'(y)$ and for each i, $(f_0 - f_0')\psi_i + (f_i - f_i') = 0$; and since no ψ_i is a constant by (2) in the hypothesis, this implies that $f_0 = f_0'$, and finally that $f_i - f_i'$ for each i.

The identity (54) is trivial because $\pi(z) = x_1\psi_1 + \cdots + x_n\psi_n$ and $\pi(x_i) = x_i$.

Basis, $n = 1$. This is covered by the preliminary case.

Induction Step, $n > 1$. If the preliminary case does not apply, then there is at least one non-trivial $\{\cdot, \div\}$ in eqdiag($\mathbf{U}$); so there is a least $m > 0$ such that some $\chi \in G_m(\mathbf{U}, z, \vec{x}, y)$ is a non-trivial product or quotient of elements of $\mathbf{G}_{m-1}(\mathbf{U}, z, \vec{x}, y)$ in which all $\{\cdot, \div\}$ are trivial; and so there is at least one non-trivial $\{\cdot, \div\}$ in eqdiag($\mathbf{U}$) of the form

$$(f_0' z + \textstyle\sum_{1 \le i \le n} f_i' x_i + \phi'(y)) \circ (f_0 z + \textstyle\sum_{1 \le i \le n} f_i x_i + \phi(y)) = \chi \tag{57}$$

where $\circ$ is $\cdot$ or $\div$. We consider cases of how this can arise.

Case 1: There is some $i \ge 1$ *such that* $f_i \ne 0$*, and the first factor in* (57) *is not in* F. We assume without loss of generality that $f_1 \ne 0$, and then dividing the equation by f_1 we put the second factor in the form

$$f_0 z + x_1 + \textstyle\sum_{2 \le i \le n} f_i x_i + \phi(y). \tag{58}$$

Appealing to Lemma 4.2 (with $v = x_1, \vec{u} = z, x_2, \ldots, x_n, y, \phi_1 = 1, \phi_2 = 0$), choose some $\overline{f} \in F$ such that the substitution

$$\rho_1(x_1) := \overline{f} - f_0 z - \textstyle\sum_{2 \le i \le n} f_i x_i - \phi(y)$$

induces an isomorphism

$$\rho_1 : \mathbf{U} \rightarrowtail\!\!\!\!\twoheadrightarrow \rho_1[\mathbf{U}] = \mathbf{U}_1 \subseteq_p \mathbf{F}(z, x_2, \dots, x_n, y).$$

Notice that ρ_1 does not introduce any new non-trivial multiplication or division (because it leaves $F(y)$ fixed), and it turns the chosen operation in $\mathbf{U}$ into a trivial one since

$$\rho_1(f_0 z + x_1 + \textstyle\sum_{2\le i\le n} f_i x_i + \phi(y)) = \overline{f}.$$

So there are fewer than $n-1$ $\{\cdot, \div\}$ in eqdiag$(\mathbf{U}_1)$, and $\mathbf{U}_1$ is generated by $z, x_2, \dots, x_n, y$ and $\{\overline{f}\} \cup (F \cap U)$.

By Lemma 4.2 again (with $v = z, \vec{u} = x_2, \dots, x_n, y, \phi_1 = 0, \phi_2 = \frac{1}{1+f_0\psi_1}$), fix some $\overline{g} \in F$ such that the substitution

$$\rho_2(z) := \frac{1}{1+f_0\psi_1}\Big((\overline{f} - \phi)\psi_1 + \overline{g}z\Big)$$

induces an isomorphism

$$\rho_2 : \mathbf{U}_1 \rightarrowtail\!\!\!\!\twoheadrightarrow \rho_2[\mathbf{U}_1] = \mathbf{U}_2 \subseteq_p \mathbf{F}(z, x_2, \dots, x_n, y).$$

This too does not introduce any non-trivial multiplications, and $\mathbf{U}_2$ is generated by $z, x_2, \dots, x_n, y$ and $F \cap U_2$. The required partial field homomorphism is the composition

$$\pi = \sigma \circ \rho_2 \circ \rho_1 : F(z, \vec{x}, y) \rightharpoonup F(\vec{x}, y)$$

of the three substitutions, where σ is guaranteed by the induction hypothesis so that $\sigma \restriction U_2 : \mathbf{U}_2 \rightarrowtail \mathbf{F}(x_2, \dots, x_n, y)$ and

$$\overline{g}\sigma(z) = \textstyle\sum_{2\le i\le n}(\psi_i - f_i\psi_1)\sigma(x_i).$$

This exists because the functions

$$\frac{1}{\overline{g}}(\psi_i - f_i\psi_1) \quad (i = 2, \dots, n)$$

satisfy (2) in the theorem.

To see that this embedding has the required property, notice first that

$$\pi(z) = \sigma(\rho_2(z))$$

because $\rho_1(z) = z$. Using the corresponding properties of ρ_2 and σ, we get:

$$\begin{aligned}
&\pi(x_1)\psi_1 + \textstyle\sum_{2\le i\le n}\pi(x_i)\psi_i \\
&\quad= \sigma(\rho_2(\overline{f} - \phi(y) - \textstyle\sum_{2\le i\le n} f_i x_i - f_0 z))\psi_1 + \textstyle\sum_{2\le i\le n}\sigma(x_i)\psi_i \\
&\quad= \sigma\Big(\overline{f} - \phi(y) - \textstyle\sum_{2\le i\le n} f_i x_i - f_0\rho_2(z)\Big)\psi_1 + \textstyle\sum_{2\le i\le n}\sigma(x_i)\psi_i \\
&\quad= (\overline{f} - \phi(y))\psi_1 - f_0\psi_1\sigma(\rho_2(z)) + \textstyle\sum_{2\le i\le n}(\psi_i - f_i\psi_1)\sigma(x_i) \\
&\quad= (\overline{f} - \phi(y))\psi_1 - f_0\psi_1\sigma(\rho_2(z)) + \overline{g}\sigma(z).
\end{aligned}$$

So what we need to check is the equation

$$\begin{aligned}
\sigma(\rho_2(z)) &= (\overline{f} - \phi(y))\psi_1 - f_0\psi_1\sigma(\rho_2(z)) + \overline{g}\sigma(z) \\
\text{equivalently } (1 + f_0\psi_1)\sigma(\rho_2(z)) &= (\overline{f} - \phi(y))\psi_1 + \overline{g}\sigma(z) \\
\text{equivalently } (1 + f_0\psi_1)\rho_2(z) &= (\overline{f} - \phi(y))\psi_1 + \overline{g}z,
\end{aligned}$$

and the last is immediate from the definition of $\rho_2(z)$. (Note that we use repeatedly the fact that σ is injective on U_2 and the identity on $F(y)$.)

Case 2: $f_1 = \cdots = f_n = 0$, $f_0 \neq 0$, and the first factor in (57) *is not in F.* We may assume without loss of generality that $f_0 = 1$, and so the second factor has the form

$$z + \phi(y).$$

By Lemma 4.2, choose some $\overline{f} \in F$ such that the substitution

$$\rho_1(z) := \overline{f} - \phi(y)$$

induces an isomorphism

$$\rho_1 : \mathbf{U} \rightarrowtail\!\!\!\!\rightarrow \rho_1[\mathbf{U}] = \mathbf{U}_1 \subseteq_p \mathbf{F}(\vec{x}, y).$$

There is one fewer non-trivial operation in eqdiag($\mathbf{U}_1$), since ρ_1 does not introduce any new ones and $\rho_1(z + \phi(y)) = \overline{f}$. Next, choose $\overline{g} \in F$ by Lemma 4.2 again, such that the substitution

$$\rho_2(x_1) := \frac{1}{\psi_1}\Big(\overline{f} - \phi(y) - \overline{g}z\Big)$$

induces an isomorphism

$$\rho_2 : \mathbf{U}_1 \rightarrowtail\!\!\!\!\rightarrow \rho_2[\mathbf{U}_1] = \mathbf{U}_2 \subseteq_p \mathbf{F}(z, x_2, \ldots, x_n, y).$$

There are fewer than $n - 1$ non-trivial $\{\cdot, \div\}$ in $\mathbf{U}_2$, and so the induction hypothesis gives us an embedding

$$\sigma : \mathbf{U}_2 \rightarrowtail \mathbf{F}(z, x_2, \ldots, x_n, y)$$

such that

$$\overline{g}\sigma(z) = \textstyle\sum_{2 \leq i \leq n} \sigma(x_i)\psi_i.$$

The required embedding is the composition $\pi = \sigma \circ \rho_2 \circ \rho_1$. To check this, note first that

$$\pi(z) = \sigma(\rho_2(\rho_1(z))) = \sigma(\rho_2(\overline{f} - \phi(y))) = \overline{f} - \phi(y).$$

On the other hand,

$$\begin{aligned}\pi(x_1)\psi_1 + \textstyle\sum_{2 \leq i \leq n} \pi(x_i)\psi_i &= \sigma(\rho_2(x_1))\psi_1 + \textstyle\sum_{2 \leq i \leq n} \sigma(x_i)\psi_i \\ &= \sigma\Big(\frac{1}{\psi_1}\Big(\overline{f} - \phi(y) - \overline{g}z\Big)\psi_1\Big) + \textstyle\sum_{2 \leq i \leq n} \sigma(x_i)\psi_i \\ &= \overline{f} - \phi(y) - \overline{g}\sigma(z) + \overline{g}\sigma(z) = \pi(z).\end{aligned}$$

Cases 3 and 4: Cases 1 and 2 do not apply, some $f_i' \neq 0$, and the second factor in (57) *is not in F*—which means that it is in $F(y) \setminus F$. These are handled exactly like Cases 1 and 2.

This completes the proof, because if none of these cases apply, then both factors of (57) are in $F(y)$, and so the operation is trivial. ⊣

4B. Counting identity tests along with $\{\cdot, \div\}$. We outline here a proof of the following theorem, which is also implicit in Bürgisser and Lickteig [1992] for algebraic decision trees.

THEOREM 4.4. *If F is a field of characteristic 0, $n \geq 1$, and $a_0, \dots, a_n, b \in F$ are algebraically independent* (over the prime field $\mathbb{Q}$)*, then*

$$\text{calls}_{\{\cdot,\div,=\}}(\mathbf{F}, N_F, a_0, \dots, a_n, b) = n + 1. \tag{59}$$

In particular, (59) *holds for the reals $\mathbb{R}$ and the complexes $\mathbb{C}$ with algebraically independent $a_0, a_1, \dots, a_n, b$.*

This will follow from the following lemma, exactly as in the preceding section.

We define *trivial* $=$-tests exactly as for the multiplication and division operations: i.e., an entry $(=, a, b, w) \in \text{eqdiag}(\mathbf{U})$ with $\mathbf{U} \subseteq_p \mathbf{F}(z, \vec{x}, y)$ is trivial if $a \in F$, or $b \in F$, or $a, b \in F(y)$.

LEMMA 4.5. *Suppose F is an infinite field, $n \geq 1$, z, $\vec{x} = x_1, \dots, x_n$, y are distinct indeterminates,*

$$\mathbf{U} \subseteq_p \mathbf{F}(z, \vec{x}, y) = (F(z, x_1, \dots, x_n, y), 0, 1, +, -, \cdot, \div, =)$$

is finite, and $\psi_1, \dots, \psi_n \in F(y)$ so that the following conditions hold:

(1) *$\mathbf{U}$ is generated by $(F \cap U) \cup \{z, \vec{x}, y\}$.*
(2) *For any $f_1, \dots, f_n \in F$, if $f_1\psi_1 + \cdots + f_n\psi_n \in F$, then $f_1 = \cdots = f_n = 0$.*
(3) *There are no more than n non-trivial $\{\cdot, \div, =\}$ entries in* eqdiag($\mathbf{U}$)*.*

Then there is a partial field homomorphism $\pi : F(z, \vec{x}, y) \rightharpoonup F(\vec{x}, y)$ which is the identity on $F(y)$ and satisfies

$$\pi(z) = \pi(x_1)\psi_1 + \cdots + \pi(x_n)\psi_n. \tag{60}$$

Moreover, π is total on U and so $\pi \restriction U : \mathbf{U} \to \mathbf{F}(\vec{x}, y)$ is a homomorphism which satisfies (60)*.*

OUTLINE OF THE PROOF. If $\mathbf{U}$ has at least one non-trivial $=$ - test, then it has no more than $n-1$ non-trivial $\{\cdot, \div\}$ and the lemma follows from Lemma 4.3, since the π produced by it is injective on U (an embedding) and so it respects all equalities and inequalities among members of U, including those in eqdiag($\mathbf{U}$). Similarly, if $\mathbf{U}$ has fewer than n non-trivial $\{\cdot, \div\}$, no matter how many $=$ - tests it has. This leaves only one case to consider:

$(*)$ *There are exactly n non-trivial $\{\cdot, \div\}$ and no non-trivial $=$ - tests in* eqdiag($\mathbf{U}$).

This would be the case, for example, it eqdiag($\mathbf{U}$) comprises all the calls made to compute the polynomial by the Horner Rule without any $=$ - test to verify that the value is or is not 0.

We define the (non-trivial) $\mathbf{U}$-*rank* of an element $w \in U$ by the recursion below, where for any $X \subseteq U$,

$$C(X) = \text{the closure of } X \text{ under trivial operations in } \mathbf{U}.$$

This can be defined precisely as $\cup_i C_i(X)$ where

$$C_0(X) = X, \quad C_{i+i}(X) = C_i(X) \cup \{\phi(\vec{u}) : \vec{u} \in C_i(X) \ \&\ \phi \text{ is trivial}\},$$

meaning $+, -$, multiplication/division by an element of $F \cap U$ or multiplication/division of elements in $F(y)$. Now

$$\begin{aligned} R_0 &= C(F \cap U),\\ R_{m+1} &= C(\{uv, \frac{u}{v} : \text{this is a non-trivial } \{\cdot, \div\} \text{ with } u, v \in R_m\}\\ \text{rank}(w) &= \min\{m : w \in R_m\}. \end{aligned}$$

Let w be an element with maximum $\text{rank}(w)$ in $\mathbf{U}$. We may assume that

$$w = uv \text{ or } w = \frac{u}{v} \text{ in } \text{eqdiag}(\mathbf{U}),$$

with a non-trivial $\{\cdot, \div\}$, since these are the only operations which increase rank. We assume division wlog.

Case 1. There is a non-trivial $(\div, a, w, b) \in \text{eqdiag}(\mathbf{U})$, or similarly with multiplication.

Choose one such entry and let $\mathbf{U}_1$ have universe U and

$$\text{eqdiag}(\mathbf{U}_1) = \text{eqdiag}(\mathbf{U}) \setminus \{(\div, a, w, b)\}.$$

Keep in mind that a may involve w also, e.g., it might be a rational function of elements of smaller or equal rank to $rank(w)$. Also, $w \neq 0$, since $\text{rank}(0) = 0$.

Since $\text{rank}(b) \leq \text{rank}(w)$, there must be some entry in $\text{eqdiag}(\mathbf{U})$ which introduces b using elements of smaller ranks; so $\mathbf{U}_1$ is generated by $z, \vec{x}, y$. It has $n-1$ non-trivial $\{\cdot, \div\}$, and so by Lemma 4.3, there is a partial field homomorphism $\pi : F(z, \vec{x}, y) \rightharpoonup F(\vec{x})$ which is total and injective on $U = U_1$ and satisfies the relevant equation. In particular, this implies that $\pi(w) \neq 0$, and so

$$\pi(b) = \pi\Big(\frac{a}{w}\Big) = \frac{\pi(a)}{\pi(w)},$$

since π is a partial field homomorphism whose domain is a subfield of $F(z, \vec{x}, y)$ that already contains a and w, and so it contains $\frac{a}{w}$ since $\pi(w) \neq 0$. So $\pi \restriction U$ is total on $\mathbf{U}_1$, as required.

Case 2: w does not occur in an argument for a non-trivial operation, so all we know about it is from entries like

$$w = \frac{u}{v}, w = u_1 v_1, \ldots \text{with } u,\ v \text{ of smaller rank.}$$

We now define $\mathbf{U}_1$ by removing from $\text{eqdiag}(\mathbf{U})$ all of these from $\mathbf{U}$ as well as all the trivial entries in which w occurs as an argument, and taking as universe those elements of U which occur in one of the remaining entries. In particular, $w \notin U_1$. Now $\mathbf{U}_1$ is a structure with one less non-trivial $\{\cdot, \div\}$, and we can apply Lemma 4.3 again to get a partial field homomorphism which is injective on it. Notice that $v \neq 0$, since it occurs as a denominator in an entry of $\text{eqdiag}(\mathbf{U})$, and so $\pi(v) \neq 0$. So $\pi(w)$ is defined by any of the defining equations for w, e.g.,

$$\pi(w) = \frac{\pi(u)}{\pi(v)}.$$

By the same token, π is also defined on all the elements of U which are introduced by trivial operations that involve w, even if $\pi(w) = 0$: because w does not occur as a denominator in any one of these. Finally, π is total on U and still satisfies the relevant equation, which completes the proof. $\dashv$

Note that Case 2 arises if $\mathbf{U}$ is constructed by using Horner's Rule to compute

$$a_0 + a_1x + \cdots + a_nx^n = a_0 + w \text{ where } w = xv = x(a_1 + a_2x + \cdots a_nx^{n-1})$$

with w the element of largest rank introduced in U by a non-trivial multiplication. Now $U_1 = U \setminus \{w, a_0 + w\}$ and

$$\mathrm{eqdiag}(\mathbf{U}_1) = \mathrm{eqdiag}(\mathbf{U}) \setminus \{xv = w, v = a_0 + w\}$$

The partial field homomorphism defined on $\mathbf{U}_1$ and so on $\mathbf{U}$ satisfies

$$\pi(a_0) + \pi(a_1)x + \cdots + \pi(a_n)x^n = 0, \quad \pi(v) = \pi(a_1 + a_2x + \cdots + a_nx^{n-1})$$

and since it is a homomorphism with $x, a_0, \dots, a_n$ in its domain, all the terms a_ix^i are also in its domain and so it satisfies

$$\pi(v) = \pi(a_1) + \pi(a_2)x + \cdots + \pi(a_n)x^{n-1}.$$

Hence

$$\pi(w) = \pi(xv) = \pi(x)\pi(v) = \pi(a_1)x + \pi(a_2)x^2 + \cdots + \pi(a_n)x^n$$
$$\pi(v) = \pi(a_0 + w) = \pi(a_0) + \pi(w) = 0.$$

So this is a case where π is not an embedding, because $v \neq 0$ but $\pi(v) = 0$. The point is that although $\mathbf{U}$ can compute v, it does not check that it is $\neq 0$, and so the restriction of π to U is a structure homomorphism.

4C. Horner's rule is $\{+,-\}$-optimal for nullity. Notice first that we can test whether $a_0 + a_1w = 0$ by executing three multiplications, equality tests and no $\{+,-\}$ (additions/subtractions): first check if any of a_0, a_1, w is 0 and give the correct answer for these cases, and if none applies, set

$$f(a_0, a_1, w) = \text{if } \ a_0^2 \neq (a_1w)^2 \text{ then ff else if } a_0 = a_1w \text{ then ff else tt}.$$

The method works for any field with characteristic $\neq 2$ and combines with Horner's rule to decide whether $a_0 + a_1x + \cdots + a_nx^n = 0$ using $(n-1)$ additions (and $(n+2)$ multiplications) along with equality tests: apply Horner's rule to compute $w = a_1 + \cdots + a_nx^{n-1}$ using $n-1$ multiplications and additions and then use the subroutine above with this w. This gives

$$\mathrm{calls}_{\{+,-\}}(\mathbf{F}, N_F, a_0, \dots, a_n, b) \leq n - 1 \quad (\mathrm{char}(F) \geq 2, n \geq 1)$$

with $\mathbf{F}$ defined by (50), and the correct lower bound for the number of $\{+,-\}$ required to test nullity with unlimited calls to $\cdot, \div, =$ is $n-1$.[15]

Theorem 4.6. *If $n \geq 2$, $F = \mathbb{R}$ or $\mathbb{C}$ and $a_0, a_1, \dots, a_n, b \in F$ are algebraically independent* (over $\mathbb{Q}$)*, then*

$$\mathrm{calls}_{\{+,-\}}(\mathbf{F}, N_F, a_0, a_1, \dots, a_n, b) = n - 1. \tag{61}$$

[15]I do not know if Horner rule's $2n$ is the correct lower bound for the combined number of operations with unlimited equality tests in the generic case. The current results give a possibly too low total lower bound of $2n-1$.

This will follow for $\mathbb{R}$ as in 4A, from a much stronger lemma which has the appropriate induction hypotheses. The proof for $\mathbb{C}$ is similar and we will skip it.

For any natural number $k > 0$ and any $w \in \mathbb{R}^+ = \{u \in \mathbb{R} : u \geq 0\}$, let $w^{\frac{1}{k}}$ be the positive k'th root of w; for any fraction $b = \frac{m}{k} \neq 0$ and $w \in \mathbb{R}^+$, set $w^b = (w^{\frac{1}{k}})^m$; and for positive $v_1, \dots, v_n$, let

$$\text{Roots}(v_1, \dots, v_n) = \{{v_i}^b \mid i = 1, \dots, n, b \in \mathbb{Q}, b \neq 0\}.$$

Let $\mathbb{K}$ be the field of algebraic real numbers, and for any two tuples of real numbers $\vec{u} = (u_1, \dots, u_k), \vec{v} = (v_1, \dots, v_n)$ with $v_1, \dots, v_n > 0$, let

$$\begin{aligned}\mathbb{K}^*(\vec{u}; \vec{v}) &= \mathbb{K}(\{u_1, \dots, u_k\} \cup \text{Roots}(v_1, \dots, v_n)) \\ &= \text{the rational functions of algebraic numbers,} \\ &\qquad\qquad u_1, \dots, u_k \text{ and rational powers of } v_1, \dots, v_n.\end{aligned}$$

By the basic notational convention (50),

$$\mathbf{K}^*(\vec{u}; \vec{v}) = (\mathbb{K}^*(\vec{u}; \vec{v}), 0, 1, +, -, \cdot, \div, =),$$

the expansion of the field structure of $\mathbb{K}^*(\vec{u}; \vec{v})$ by the equality relation.

Suppose $\mathbf{U} \subseteq_p \mathbf{K}^*(y, z; x_1, \dots, x_n)$. An addition or subtraction $u \pm v = w$ in eqdiag($\mathbf{U}$) is *trivial* if $u, v \in \mathbb{K}(y, z)$.

Lemma 4.7. *Suppose $n \geq 2$, $\overline{g} \in \mathbb{K}$, $\overline{g} \neq 0$, $z, x_1, \dots, x_n, y$ are algebraically independent real numbers with $x_1, \dots, x_n > 0$, and $\mathbf{U}$ is a finite substructure of $\mathbf{K}^*(y, z; x_1, \dots, x_n)$ generated by*

$$(U \cap \mathbb{K}) \cup \{y, z\} \cup (U \cap \text{Roots}(x_1, \dots, x_n))$$

which has $< (n-1)$ non-trivial additions and subtractions.

Then there is a partial field homomorphism

$$\pi : \mathbb{K}^*(y, z; x_1, \dots, x_n) \rightharpoonup \mathbb{K}^*(y; x_1, \dots, x_n)$$

such that:

(a) *$\pi(a) = a$ for every $a \in \mathbb{K}$ and $\pi(y) = y$;*
(b) *π is total and injective on U, and so it induces an embedding*

$$\pi \restriction U : \mathbf{U} \rightarrowtail \mathbf{K}^*(y; x_1, \dots, x_n) \subseteq_p \mathbf{R};$$

and

(c) $\pi(z) + \overline{g}\Big(\pi(x_1)y^1 + \cdots + \pi(x_n)y^n\Big) = 0.$

Proof of Theorem 4.6 from Lemma 4.7. Towards an application of the Homomorphism Test 2.4, suppose $n \geq 2$ and $\mathbf{U} \subseteq_p \mathbf{R}$ is finite, generated by algebraically independent

$$a_0, a_1, \dots, a_k, a_{k+1}, \dots, a_n, b \quad (a_1, \dots, a_k > 0, a_{k+1}, \dots, a_n < 0)$$

which we have split in two lists according to the sign. Clearly

$$a_0 + a_1 b + \cdots + a_n b^n \neq 0. \tag{62}$$

Let $\rho : \mathbf{U} \rightarrowtail \mathbf{R}$ be the embedding defined by the "relabelling"

$$\rho(a_{k+1}) = -a_{k+1}, \dots, \rho(a_n) = -a_n$$

of the negative a_i's, which is an isomorphism of $\mathbf{U}$ with $\mathbf{U}_1 = \rho[\mathbf{U}]$. If $\mathbf{U}$ has $< n-1$ additions and subtractions, then $\mathbf{U}_1$ satisfies all the hypotheses of the Lemma with $z = \rho(a_0) = a_0, y = \rho(b) = b, x_1 = \rho(a_1), \dots, x_n = \rho(a_n)$, and so there is an embedding $\pi_1 : \mathbf{U}_1 \rightarrowtail \mathbf{R}$ such that

$$\pi_1(\rho(a_0)) + \pi_1(\rho(a_1))b + \cdots + \pi_1(\rho(a_n))b^n = 0.$$

The composition $\pi = \pi_1 \circ \rho : \mathbf{U} \rightarrowtail \mathbf{R}$ then does not respect (62), and so the Homomorphism Test gives the conclusion of the theorem. $\dashv$

Proof of Lemma 4.7 is by induction on $n \geq 2$, starting with a preliminary case which will also cover the basis of the induction.

Sublemma 4.7.1 (Preliminary case). *There are no non-trivial* $\{+,-\}$ *in* $\mathbf{U}$.

Proof. It follows that every member of U is uniquely of the form

$$M = x_1^{b_1} \cdots x_n^{b_n} p(y,z) \tag{63}$$

where $b_1, \dots, b_n \in \mathbb{Q}$ and $p(y,z) \in \mathbb{K}(y,z) \cap U$. Define

$$\pi : \mathbb{K}^*(y,z; x_1, \dots, x_n) \rightharpoonup \mathbb{K}^*(y; x_1, \dots, x_n)$$

by the substitution

$$z \mapsto -\overline{g}\Big(x_1 y^1 + \cdots + x_n y^n\Big),$$

It is enough to show that this is total and injective on the set of all numbers of the form (63), so that in particular it is total and injective on $\mathbf{U}$. We skip the argument that $\pi(M)$ is defined for all M in the form (63), as it is similar to (and a bit simpler) than the argument for injectivity which follows.

Suppose then that

$$x_1^{b_1} \cdots x_n^{b_n} p(y, -\overline{g}(x_1 y^1 + \cdots + x_n y^n)) = x_1^{b_1'} \cdots x_n^{b_n'} p'(y, -\overline{g}(x_1 y^1 + \cdots + x_n y^n))$$

where $p(y,z), p'(y,z) \in \mathbb{K}(y,z)$. By clearing the denominators of the rational functions $p(y,z), p'(y,z)$ and the negative powers by cross-multiplying and then the denominators in the exponents of $x_1, \dots, x_n$ by raising both sides to suitable integer powers, we may assume that all exponents in this equation are in $\mathbb{N}$, $b_i b_i' = 0$ for $i = 1, \dots, n$, and $p(y,z), p'(y,z)$ are polynomials in $\mathbb{K}[y,z]$. We then expand these polynomials in powers of z, so that the assumed equation takes the form

$$\begin{aligned}x_1^{b_1} \cdots x_n^{b_n}[A_0(y) + A_1(y)(-\overline{g})(x_1 y^1 + \cdots + x_n y^n) + \cdots \\ + A_k(y)(-\overline{g})^k (x_1 y^1 + \cdots + x_n y^n)^k] \\ = x_1^{b_1'} \cdots x_n^{b_n'}[B_0(y) + B_1(y)(-\overline{g})(x_1 y^1 + \cdots + x_n y^n) + \cdots \\ + B_l(y)(-\overline{g})^l (x_1 y^1 + \cdots + x_n y^n)^l]\end{aligned}$$

where $A_k(y), B_l(y) \neq 0$. The terms with the highest powers of x_1 on the (fully expanded) two sides of this equation must be equal, and so we have

$$x_1^{b_1} \cdots x^{b_n} A_k(y)(-\overline{g})^k x_1^k y^k = x_1^{b_1'} \cdots x^{b_n'} B_l(y)(-\overline{g})^l x_1^l y^l$$

from which we get immediately

$$x_2^{b_2}\cdots x_n^{b_n} = x_2^{b'_2}\cdots x_n^{b_n},$$

so that $b_i = b'_i$ for $i = 2, \dots, n$; and since $b_i b'_i = 0$, all these numbers are 0. If we repeat this argument[16] using x_n rather than x_1, we get that $b_1 = b'_1 = 0$ also, so that the original equation takes the simpler form

$$\begin{aligned}(64)\quad & A_0(y) + A_1(y)(-\overline{g})(x_1y^1 + \cdots + x_ny^n) + \cdots \\ & \qquad + A_k(y)(-\overline{g})^k(x_1y^1 + \cdots + x_ny^n)^k \\ &= B_0(y)+B_1(y)(-\overline{g})(x_1y^1+\cdots+x_ny^n)+\cdots+B_l(y)(-\overline{g})^l(x_1y^1+\cdots+x_ny^n)^l.\end{aligned}$$

If we now equate again the terms with the highest powers of x_1 in this equation, we get

$$A_k(y)(-\overline{g})^k y^k x_1^k = B_l(y)(-\overline{g})^l y^l x_1^l;$$

which gives immediately that $k = l$ and $A_k(y) = B_l(y)$. Finally, we can cancel these terms from (64) and then repeat the argument to show that $A_i(y) = B_i(y)$ for all $i \le k$, completing the proof. $\dashv$ (Sublemma 4.7.1)

The basis of the induction $n = 2$ is covered by the preliminary case.

In the induction step with $n > 2$, if the preliminary case does not apply, then there must exist a "least complex" non-trivial addition or subtraction in $\mathbf{U}$ of the form

$$(65)\qquad w = x_1^{b_1}\cdots x_n^{b_n}p(y,z) \pm x_1^{b'_1}\cdots x_n^{b'_n}p'(y,z)$$

where $p(y,z), p'(y,z) \in \mathbb{K}(y,z)$ and the component parts

$$u = x_1^{b_1}\cdots x_n^{b_n}p(y,z),\quad v = x_1^{b'_1}\cdots x_n^{b'_n}p'(y,z)$$

are also in U. We may, in fact, assume that this is an addition, by replacing $p'(y,z)$ by $-p'(y,z)$ if necessary.

Sublemma 4.7.2. *We may assume that in* (65), $b'_i = 0$ *for* $i = 1, \dots, n$ *and* $p(y,z)$, $p'(y,z)$ *are polynomials, i.e.,* (65) *is of the form*

$$(66)\qquad w = x_1^{b_1}x_2^{b_2}\cdots x_n^{b_n}p(y,z) + p'(y,z)$$

in which $p(y,z), p'(y,z) \in \mathbb{K}[y,z]$.

Proof. Let

$$W = x_1^{-b'_1}x_2^{-b'_2}\cdots x_n^{-b'_n}d(p(y,z))d(p'(y,z))$$

where $d(p(y,z)), d(p'(y,z)$ are the denominators of $p(y,z), p'(y,z)$ and replace (65) in eqdiag($\mathbf{U}$) by the operations

$$u_1 = Wu,\quad v_1 = Wv,\quad w_1 = u_1 + v_1,\quad w = \frac{w_1}{W}$$

along with all the multiplications, divisions and trivial additions and subtractions required to compute W. If $\mathbf{U}'$ is the resulting structure, then clearly $U \subseteq U'$ and the fixed, non-trivial addition in $\mathbf{U}$ has been replaced by one of the form (66).

[16]This is the part of the proof where $n \ge 2$ is used.

It is not quite true that $\mathbf{U} \subseteq_p \mathbf{U}'$, because the basic equation $w = u + v$ is in eqdiag($\mathbf{U}$) but not in eqdiag($\mathbf{U}'$). On the other hand, if

$$\pi : \mathbb{K}^*(y, z; x_1, \dots, x_n) \rightharpoonup \mathbb{K}^*(y; x_1, \dots, x_n)$$

is a partial field homomorphism which is total and injective on U', then $\pi \restriction U : \mathbf{U} \rightarrowtail \mathbf{K}^*(y; x_1, \dots, x_n)$ is an embedding, because it preserves all the other entries in eqdiag($\mathbf{U}$) and $\pi(u+v) = \frac{\pi(u_1)+\pi(v_1)}{\pi(W)} = \frac{\pi(w_1)}{\pi(W)} = \pi(w)$. $\dashv$ (Sublemma 4.7.2)

It must be the case that some $b_i \neq 0$ in (66), otherwise the chosen addition is trivial, and we may assume to simplify notation that this happens for $i = 1$. For each strictly positive $\overline{f} \in \mathbb{Q}$, let

$$\rho_{\overline{f}} : \mathbb{K}^*(y, z; x_1, \dots, x_n, z) \rightharpoonup \mathbb{K}^*(y, z; x_2, \dots, x_n, z)$$

be the partial field homomorphism induced by the substitution

$$\rho_{\overline{f}}(x_1) := \Big(\frac{\overline{f}}{x_2^{b_2} \cdots x_n^{b_n}}\Big)^{\frac{1}{b_1}}. \tag{67}$$

Sublemma 4.7.3. *For all but finitely many $\overline{f} \in \mathbb{Q}^+$, $\rho_{\overline{f}}$ is total and injective on U.*

This is proved very much like Lemma 4.2 and we will not repeat the argument. We fix one such $\overline{f}$ and we let

$$\rho_1 = \rho_{\overline{f}}.$$

Since the restriction of ρ_1 to U is injective, ρ_1 defines an isomorphism

$$\rho_1 \restriction U : \mathbf{U} \rightarrowtail\!\!\!\!\rightarrow \rho_1[\mathbf{U}] = \mathbf{U}'$$

of $\mathbf{U}$ with its image $\mathbf{U}'$, which is generated by

$$(U \cap \mathbb{K}) \cup \{\overline{f}^{\frac{1}{b_1}}\} \cup \{y, z\} \cup (U' \cap \text{Roots}(x_2, \dots, x_n)).$$

Also, ρ_1 takes trivial $\{+, -\}$ to trivial ones, because it is the identity on $\mathbb{K}(y, z)$, and it transforms the non-trivial addition in (66) into a trivial one since

$$\rho_1(x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n} p(y, z)) = \overline{f} p(y, z).$$

So there are fewer than $n - 2$ non-trivial $\{+, -\}$ in $\mathbf{U}'$, and we can apply the induction hypothesis to it, further down. Let

$$X = \rho_1(x_1) = \overline{f}^{\frac{1}{b_1}} x_2^{-\frac{b_2}{b_1}} \cdots x_n^{-\frac{b_n}{b_1}}$$

and for any $\overline{h} \in \mathbb{Q}$, $\overline{h} > 0$, define $\rho_2 : \mathbb{K}^*(y, z; x_2, \dots, x_n) \rightharpoonup \mathbb{K}^*(y, z; x_2, \dots, x_n)$ by the substitution

$$\rho_2(z) := \overline{g}(\frac{y}{\overline{h}} z - Xy).$$

Sublemma 4.7.4. *For all but finitely many $\overline{h} \in \mathbb{Q}^+$, the homomorphism ρ_2 is injective on U'.*

This, too is easily checked as before. We fix one such $\overline{h}$ and we invoke the induction hypothesis to get a partial field homomorphism

$$\sigma : \mathbb{K}^*(y, z; x_2, \dots , x_n) \rightharpoonup \mathbb{K}^*(y; x_2, \dots , x_n)$$

which fixes $\mathbb{K}$ and y, is total and injective on $\mathbf{U}'$ and satisfies

$$\sigma(z) + \overline{h}\Big(\sigma(x_2)y + \cdots + \sigma(x_n)y^{n-1}\Big) = 0.$$

We claim that the required partial field homomorphism is the composition

$$\pi = \sigma \circ \rho_2 \circ \rho_1 : \mathbb{K}^*(y, z; x_1, x_2, \dots , x_n) \rightharpoonup \mathbb{K}^*(y; x_2, \dots , x_n).$$

This is certainly total and injective on U, as the composition of three injections. To check that it has the required property, we compute, using the properties of the three homomorphisms involved—i.e., that ρ_1 fixes all the generators of $\mathbb{K}^*(y, z; x_1, x_2, \dots , x_n)$ except for x_1 and ρ_2 affects only z:

$$\begin{aligned}
&\overline{g}\Big(\pi(x_1)y + \pi(x_2)y^2 + \cdots \pi(x_n)y^n\Big)\\
&\qquad = \overline{g}\Big(\sigma(\rho_2(\rho_1(x_1))y + \sigma(x_2)y^2 + \cdots + \sigma(x_n)y^n\Big)\\
&\qquad = \overline{g}\Big(\sigma(X)y + y\Big(\sigma(x_2)y + \cdots \sigma(x_n)y^{n-1}\Big)\Big)\\
&\qquad = \overline{g}\Big(\sigma(X)y - \frac{y}{\overline{h}}\sigma(z)\Big)\\
&\qquad = \sigma\Big(\overline{g}\Big(Xy - \frac{y}{\overline{h}}z\Big)\Big)\\
&\qquad = \sigma(-\rho_2(z)) \quad \text{(by the definition of } \rho_2)\\
&\qquad = -\sigma(\rho_2(\rho_1(z))) = -\pi(z)
\end{aligned}$$

as required. ⊣

4D. Counting identity tests along with $\{+, -\}$. If we also count calls to the equality relation, then Horner's rule clearly requires n additions and one equality test to decide the nullity relation, for a total of $n + 1$. This may well be the correct lower bound for $\text{calls}_{\{+,-,=\}}(\mathbf{R}, N_{\mathbb{R}}, \vec{a}, b)$ on an infinite number of tuples $\vec{a}, b$, but we do not know how to prove this now. In any case, it fails for algebraically independent inputs:

LEMMA 4.8. *If a_0, a_1, b are algebraically independent real numbers, $\mathbf{U} \subseteq_p \mathbf{R}$ and*

$$\text{eqdiag}(\mathbf{U}) = \{u = a_1 b, w = a_0 + u, v = \frac{b}{w}\},$$

then $\mathbf{U} \Vdash^{\mathbf{R}}_{c} a_0 + a_1 b \neq 0$, and so

$$\text{calls}_{\{+,-,=\}}(\mathbf{R}, N_{\mathbb{R}}, a_1, a_2, b) \leq 1.$$

PROOF. Every homomorphism $\pi : \mathbf{U} \to \mathbf{R}$ must be defined on v and satisfy

$$\pi(v) = \frac{\pi(b)}{\pi(w)},$$

so that $\pi(w) = \pi(a_0) + \pi(a_1)\pi(b) \neq 0$. ⊣

The trick here is to use division (which we are not counting) in place of the natural identity test, so one might think that allowing only multiplications would enforce at least two $+, -$ or $=$ - tests to certify $a_0 + a_1 b \neq 0$, but this does not work either: if the single equality test $a_0^2 = (a_1 b)^2$ is in eqdiag($\mathbf{U}$), then, easily, $\mathbf{U} \Vdash a_0 + a_1 b \neq 0$. We prove the best result for the generic case and leave open the possibility that Horner's rule is optimal on infinitely many non-generic inputs.

THEOREM 4.9.[17] *If $n \in \mathbb{N}$, $F = \mathbb{R}$ or $\mathbb{C}$ and $a_0, a_1, \dots, a_n, b \in F$ are algebraically independent* (over $\mathbb{Q}$)*, then*

$$\text{calls}_{\{+,-,=\}}(\mathbf{F}, N_F, a_0, a_1, \dots, a_n, b) = n. \tag{68}$$

A partial field homomorphism $\pi : \mathbb{K}^*(y, z; x_1, \dots, x_n) \rightharpoonup \mathbb{K}^*(y, z; x_1, \dots, x_n)$ is *proper* on a set U if its kernel does not intersect $U \setminus \{0\}$, i.e.,

$$v \in U \ \& \ v \neq 0 \Longrightarrow \pi(v) \neq 0.$$

This insures that if $u \div v = w \in$ eqdiag($\mathbf{U}$), then $\pi(w)$ is defined.

Suppose $\mathbf{U} \subseteq_p \mathbb{K}^*(y, z; x_1, \dots, x_n)$. An addition $u + v$, subtraction $u - v$ or inequality test $u \neq v$[18] in eqdiag($\mathbf{U}$) is *trivial* if $u, v \in \mathbb{K}(y, z)$.

The theorem follows as before from the following lemma—and the general version of Lemma 4.8 for the upper bound.

LEMMA 4.10. *Suppose $n \geq 1$, $\overline{g} \in \mathbb{K}$, $\overline{g} \neq 0$, $z, x_1, \dots, x_n, y$ are algebraically independent real numbers with $x_1, \dots, x_n > 0$, and $\mathbf{U}$ is a finite substructure of $\mathbb{K}^*(y, z; x_1, \dots, x_n)$ generated by*

$$(U \cap \mathbb{K}) \cup \{y, z\} \cup (U \cap \text{Roots}(x_1, \dots, x_n))$$

which has $< n$ non-trivial additions, subtractions and equality tests.

Then there is a partial field homomorphism

$$\pi : \mathbb{K}^*(y, z; x_1, \dots, x_n) \rightharpoonup \mathbb{K}^*(y; x_1, \dots, x_n)$$

such that:

(a) *$\pi(a) = a$ for every $a \in \mathbb{K}$ and $\pi(y) = y$;*
(b) *π is proper on U, and so it induces a homomorphism*

$$\pi \restriction U = \pi : \mathbf{U} \to \mathbf{K}^*(y; x_1, \dots, x_n);$$

and

(c) $\pi(z) + \overline{g}\Big(\pi(x_1)y^1 + \cdots + \pi(x_n)y^n\Big) = 0.$

PROOF is by induction on $n \geq 1$. It is almost exactly (and a bit simpler) than the proof of Lemma 4.7, and we will only describe the necessary changes, mostly in the Sublemma corresponding to 4.7.1 and in the mild modification of the statements of the other Sublemmas, to include inequations.

[17] A differently formulated but equivalent result is proved for algebraic decision trees in Bürgisser, Lickteig, and Shub [1992].

[18] There is no need to include entries of the form $(=, u, v, \mathrm{tt})$ in eqdiag($\mathbf{U}$), because every homomorphism on $\mathbf{U}$ automatically respects them. So the number of significant $=$ - tests is the number of entries $(=, u, v, \mathrm{ff})$ in eqdiag($\mathbf{U}$).

Sublemma 4.10.1 (Preliminary case). *There are no non-trivial* $\{+,-,\neq\}$ *in* $\mathbf{U}$.

Proof. The members of U are uniquely of the form

$$M = x_1^{b_1} \cdot x_n^{b_n} p(y,z) \tag{69}$$

with $b_1, \dots, b_n \in \mathbb{Q}$ and $p(y,z) \in \mathbb{K}(y,z) \cap U$, and we define

$$\pi : \mathbb{K}^*(y,z;x_1,\dots,x_n) \rightharpoonup \mathbb{K}^*(y;x_1,\dots,x_n)$$

by the substitution

$$z \mapsto -\overline{g}\Big(x_1 y^1 + \cdots + x_n y^n\Big),$$

exactly as before. The difference is that we need not prove that π is an injection on elements of the form (69), which may, in fact, be false—this was the part where we used $n \geq 2$ while now n may be 1. We only need show that π is proper, i.e., that

$$x_1^{b_1} \cdot x_n^{b_n} p(y, -\overline{g}(x_1 y^1 + \cdots + x_n y^n)) = 0 \Longrightarrow x_1^{b_1} \cdot x_n^{b_n} p(y,z) = 0,$$

which is practically trivial by just the first part of the argument for Sublemma 4.7.1 which does not need the hypothesis $n > 1$. ⊣ (Sublemma 4.10.1)

A counterexample to the injectivity of π when $n = 1$ is given by the distinct polynomials z and $x_1(-y)$ with $\overline{g} = 1$, for which

$$\pi(z) = -x_1 y = \pi(x_1(-y)).$$

This Sublemma covers the basis of the induction $n = 1$. Suppose then that $n > 1$ and there is at least one entry in eqdiag($\mathbf{U}$), and hence a least-complex entry of the form

$$w = x_1^{b_1} \cdots x_n^{b_n} p(y,z) \circ x_1^{b'_1} \cdots x_n^{b'_n} p'(y,z) \tag{70}$$

where $\circ$ is $+, -$ or $\neq$, $p(y,z), p'(y,z) \in \mathbb{K}(y,z)$ and the component parts

$$u = x_1^{b_1} \cdots x_n^{b_n} p(y,z), \quad v = x_1^{b'_1} \cdots x_n^{b'_n} p'(y,z)$$

are also in U.

Sublemma 4.10.2. *We may assume that in* (70), $b'_i = 0$ *for* $i = 1, \dots, n$ *and* $p(y,z)$, $p'(y,z)$ *are polynomials, i.e.,* (70) *is of the form*

$$w = x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n} p(y,z) \circ p'(y,z) \tag{71}$$

in which $p(y,z), p'(y,z) \in \mathbb{K}[y,z]$.

The argument here is exactly like that of 4.7.2, with the extra case when $\circ$ is $\neq$ causing no problem. In fact, the rest of the proof of Lemma 4.7 goes through essentially word-for-word, as long as we change "injective" to "proper". We skip the details. ⊣

§5. Recursive programs and computation models. Most computation models which have been used to derive lower bounds in arithmetic and algebraic complexity can be viewed as *recursive programs* with various restrictions. We will establish here this representation of concrete algorithms from specified primitives and derive from it the connection between some of the natural, concrete complexity measures and the abstract measures induced by substructure norms that we have been studying. The results are very simple, natural generalizations of the construction in Section 2G, and the only subtle points are the need to identify and take account of "hidden primitives" often assumed but not explicitly identified in the specification of computation models and the use of additional "data types". These issues are especially important for the most common models of Turing and Random Access Machines (with oracles), which we will take up last.

5A. Recursive programs—syntax.[19] Let Φ be a vocabulary as in Section 1. An n-ary *recursive program* E in the signature Φ is a syntactic expression

$$E \equiv E_0(\vec{x}, \vec{p}) \text{ where } \{p_1(\vec{v}_1) = E_1(\vec{v}_1, \vec{p}), \dots , p_k(\vec{v}_k) = E_k(\vec{u}_k, \vec{p})\} \tag{72}$$

where the following conditions hold:

(RP1) $\vec{p} \equiv p_1, \dots , p_k$ is a sequence of (not necessarily distinct) function symbols, each with an assigned type

$$\text{type}(p_i) \equiv \langle k_i, s_i\rangle \equiv \langle \underbrace{\mathsf{a}, \dots , \mathsf{a}}_{k_i}, s\rangle \quad (i = 1, \dots , k).$$

(RP2) For $i = 0, \dots , k$, $E_i(\vec{v}_i, \vec{p})$ is a term in the vocabulary $\Phi \cup \{p_1, \dots , p_k\}$ whose free variables are in the list $\vec{v}_i$, with

$$\vec{v}_0 \equiv \vec{x} \equiv x_1, \dots , x_n.$$

(RP3) For $i = 1, \dots , k$, $\text{sort}(p_i(\vec{v}_i)) = \text{sort}(E_i(\vec{v}_i, \vec{p}))$.

The *sort* of E is the sort of its *head* $E_0(\vec{x}, \vec{p})$; the *free variables* of E are $x_1, \dots , x_n$; and its bound variables are those in the lists $\vec{v}_i$ and the *recursive variables* $p_1, \dots , p_k$. We identify programs which differ only by a relabelling of their bound variables.

The program E is *deterministic* if its recursive variables are all distinct, so that the *body* of E (within the braces) is a system of mutual recursive equations.

It is well known that a deterministic, n-ary Φ-program defines on every Φ-structure $\mathbf{A}$ an n-ary partial function

$$f_E : A^n \rightharpoonup A_s,$$

[19]Deterministic recursive programs were introduced by McCarthy [1963], who used them, in particular, to develop clean foundations for call-by-value computability from arbitrary, specified primitives. Especially significant was McCarthy's explicit identification of the *conditional* (branching) as an essential ingredient of computation: he used it to give an elegant characterization of the general recursive functions on ℕ which avoids the non-determinism inherent in the *Herbrand-Gödel-Kleene* systems of Kleene [1952]. Van den Dries and Moschovakis [2004] outlines briefly their applications to complexity theory, which we take up more generally (and in a different way) here.

which is determined by the *least solutions* of the recursive equations in its body. These least solutions and f_E can also be computed by various standard implementations, which also apply to non-deterministic programs and yield a suitable f_E computed by E on *some* Φ-structures—this, too is well known. In the next section we will outline a (very abstract) approach to their semantics which justifies the following identifications of several standard computation models obtained by placing syntactic restrictions on the form of E in (72).

(1) Terms: $k = 0$ in (72), i.e., the body of E is empty. In this case $E \equiv E_0(\vec{x})$ is just a Φ-term.

(2) Finite algorithms (with branching): for each $i = 1, \dots, k$, if p_j occurs in $E_i(\vec{v}_i, \vec{p})$, then $j < i$.

Among these are the *k-step algorithms* of Pan [1966] and Winograd [1967] in the vocabulary Φ (or *computation sequences* in Strassen [1990]) which are deterministic recursive programs that satisfy the following additional restrictions:

(2.1) All the bound variable lists $\vec{v}_i$ have the same length n, equivalently: for some fixed $\vec{v} \equiv v_1, \dots, v_n$ and each $i = 1, \dots, k$, $\vec{v}_i \equiv \vec{v}$.

(2.2) For each $i = 1, \dots, k$, $E_i(\vec{v}, \vec{p})$ is in one of the three forms,

$$c, \ v_j, \quad \text{or} \quad \phi(p_{j_1}(\vec{v}), \dots, p_{j_m}(\vec{v})),$$

where $1 \le j \le n$, c is a distinguished constant (a nullary function symbol) in Φ, $\phi \in \Phi$ with arity $n_\phi = m$ and $j_1, \dots, j_m < i$.

In particular, there are no conditionals in these programs. If we also allow

(2.3) $E_i(\vec{v}, \vec{p}) \equiv \text{if } p_{j_1}(\vec{v}) \text{ then } p_{j_2}(\vec{v}) \text{ else } p_{j_3}(\vec{v})$ (with $j_1, j_2, j_3 < i$),

we get what Strassen [1990] and Mansour, Schieber, and Tiwari [1991a, 1991b] call *computation trees*. These have also been called *algebraic computation trees* in the literature, or just *decision trees* when p_{j_1} is restricted to be the equality symbol $=$.

(3) Iterative (while) programs: $k = 1$ and E_1 is a *tail recursion*,

$$E_1(\vec{v}, p) \equiv \text{if test}(\vec{v}) \text{ then out}(\vec{v}) \text{ else } p(\tau(\vec{v})) \tag{73}$$

where $\vec{v} \equiv v_1, \dots, v_m$, $\text{test}(\vec{v}), \text{out}(\vec{v})$ are Φ-terms and

$$\tau(\vec{v}) \equiv (\tau_1(\vec{v}), \dots, \tau_m(\vec{v}))$$

is an m-tuple of Φ-terms which defines a *transition function.* These are "logical", recursive specifications of the classical, deterministic computation models with their *test*, *output* and *transition* functions defined by Φ-terms. Their non-deterministic versions are obtained by allowing arbitrary $k \ge 1$ but only one recursive variable p, so that every $E_i(\vec{v}, p)$ is a conditional of the form (73) with varying choices of test, out and τ.

Many more interesting classes of algorithms can be similarly specified by placing restrictions on the form of recursive programs, but these suffice for our purposes here.

5B. Recursive programs—semantics. For each recursive program E in the vocabulary Φ as in (72) and for each Φ-structure $\mathbf{A}$, let

$$\mathrm{ClTerms}(E,\mathbf{A}) = \{t : t \text{ is a closed } \Big(\Phi \cup \{p_1,\dots,p_k\} \cup A\Big)\text{-term}\},$$

where each $u \in A$ is viewed as a nullary function symbol. We define the relation

$$E,\mathbf{A} \vdash t = w \iff E \text{ proves that } t = w \text{ in } \mathbf{A}$$
$$(t \in \mathrm{ClTerms}(E,\mathbf{A}), w \in A \cup \{\mathrm{tt},\mathrm{ff}\}, \mathrm{sort}(t) = \mathrm{sort}(w))$$

by the following *generalized induction*, i.e., $\{(t,w) : E,\mathbf{A} \vdash t = w\}$ is the smallest set which satisfies the induction clauses, as in the definition of generated substructions in (10):[20]

(RP1) For each $w \in A$, $E,\mathbf{A} \vdash w = w$.
(RC2) If $E,\mathbf{A} \vdash t_1 = u_1,\dots,E,\mathbf{A} \vdash t_n = u_n$ and $\phi^{\mathbf{A}}(u_1,\dots,u_n) = w$, then $E,\mathbf{A} \vdash \phi(t_1,\dots,t_n) = w$.
(RP3) If $E,\mathbf{A} \vdash t_1 = \mathrm{tt}$ and $E,\mathbf{A} \vdash t_2 = w$, then $E,\mathbf{A} \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 = w$.
If $E,\mathbf{A} \vdash t_1 = \mathrm{ff}$ and $E,\mathbf{A} \vdash t_3 = w$, then $E,\mathbf{A} \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 = w$.
(RP4) If $E,\mathbf{A} \vdash t_1 = u_1,\dots,E,\mathbf{A} \vdash t_n = u_n$ and $E,\mathbf{A} \vdash E_i(u_1,\dots,u_n,\vec{p}) = w$ for some i such that $p \equiv p_i$, then $E,\mathbf{A} \vdash p(t_1,\dots,t_n) = w$.

Notice that if E is non-deterministic, we may well have

$$E,\mathbf{A} \vdash t = w_1 \text{ and } E,\mathbf{A} \vdash t = w_2 \text{ with } w_1 \neq w_2.$$

A partial function $f : A^n \rightharpoonup A_s$ is *computed by E in* $\mathbf{A}$ if it satisfies the equivalence

$$f(\vec{x}) = w \iff E,\mathbf{A} \vdash E_0(\vec{x},\vec{p}) = w \quad (\vec{x} \in A^n, w \in A_s). \tag{74}$$

The definition insures that each E computes *at most* one partial function in each Φ-structure $\mathbf{A}$, and it does that if for all $\vec{x}, w, w' \in A$,

$$\Big(E,\mathbf{A} \vdash E_0(\vec{x},\vec{p}) = w \ \&\ E,\mathbf{A} \vdash E_0(\vec{x},\vec{p}) = w'\Big) \Longrightarrow w = w'. \tag{75}$$

Lemma 5.1. *Suppose* $\mathbf{A}$ *is a* Φ*-structure and* E *is an* n*-ary* Φ*-recursive program.*

(1) *If* $\mathbf{U} \subseteq_p \mathbf{A}$*, then*

$$E,\mathbf{U} \vdash t = w \Longrightarrow E,\mathbf{A} \vdash t = w.$$

(2) *If E computes a partial function in* $\mathbf{A}$*, then it computes a partial function in every* $\mathbf{U} \subseteq_p \mathbf{A}$.

(3) *If E is deterministic, then E computes a partial function in* $\mathbf{A}$.

[20]There are various ways to define a notion of "computation" by E in $\mathbf{A}$, but the crucial property that any such notion must have is that

$$E,\mathbf{A} \vdash t = w \iff \text{there is a computation of } t \text{ by } E \text{ in } \mathbf{A} \text{ which outputs } w,$$

For non-deterministic programs with which we are working here, it may be that E computes two different values w_1 and w_2 of t, and so it is easier to work with this more abstract notion of "provability".

PROOF. (1) is immediate by induction on the definition of $E, \mathbf{U} \vdash t = w$. The same kind of induction verifies (75) for deterministic programs, which proves (3): the key observation is that in the deterministic case, an equation $t = w$ is provable from $E, \mathbf{A}$ in at most one way which can be read from the form of t.

(2) follows from (1), because if E computes $f : A^n \rightharpoonup A_s$ in $\mathbf{A}$, then for all $\vec{x} \in U^n$ and $w \in U$,

$$E, \mathbf{U} \vdash E_0(\vec{x}, \vec{p}) = w \Longrightarrow E, \mathbf{A} \vdash E_0(\vec{x}, \vec{p}) = w \Longrightarrow f(\vec{x}) = w,$$

and so $\{(\vec{x}, w) \in U^{n+1} : E, \mathbf{U} \vdash E_0(\vec{x}, \vec{p}) = w\}$ is the graph of a partial function $f' \sqsubseteq f$. ⊣

By a similar inductive definition, we assign to each equation $t = w$ such that $E, \mathbf{A} \vdash t = w$ the three basic complexity measures of recursive programs,

$$\mathrm{calls}_{\Phi_0}(E, \mathbf{A})(t = w) \text{ (with } \Phi_0 \subseteq \Phi),$$
$$\mathrm{size}(E, \mathbf{A})(t = w) = |\mathrm{space}(E, \mathbf{A})(t = w)|,$$
$$\mathrm{depth}(E, \mathbf{A})(t = w).$$

We skip in the definition most references to $E, \mathbf{A}$, which remain constant:

(RC1) $\mathrm{calls}_{\Phi_0}(w = w) = \mathrm{depth}(w = w) = 0$ and $\mathrm{space}(w = w) = \{w\}$ $(w \in A)$.

(RC2) If $E, \mathbf{A} \vdash t_1 = \mathrm{tt}$ and $E, \mathbf{A} \vdash t_2 = w$, then
- $\mathrm{calls}_{\Phi_0}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3 = w) = \mathrm{calls}_{\Phi_0}(t_1 = \mathrm{tt}) + \mathrm{calls}_{\Phi_0}(t_2 = w)$,
- $\mathrm{space}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3 = w) = \mathrm{space}(t_1 = \mathrm{tt}) \cup \mathrm{space}(t_2 = w)$,
- $\mathrm{depth}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3 = w) = \max(\mathrm{depth}(t_1 = \mathrm{tt}), \mathrm{depth}(t_2 = w))$,

and similarly if $E, \mathbf{A} \vdash t_1 = \mathrm{ff}$.

(RC3) If $E, \mathbf{A} \vdash t_j = u_j$ for $j = 1, \dots, n$ and $\phi^{\mathbf{A}}(u_1, \dots, u_n) = w$, then
- $\mathrm{calls}_{\Phi_0}(\phi(t_1, \dots, t_n) = w) = \sum_{j=1,\dots,n} \mathrm{calls}(t_j = u_j) + m$, where $m = 1$ if $\phi \in \Phi_0$ and $m = 0$ otherwise,
- $\mathrm{space}(\phi(t_1, \dots, t_n) = w) = \bigcup_{j=1,\dots,n} \mathrm{space}(t_j = w_j) \cup S$, where $S = \{\phi^{\mathbf{A}}(u_1, \dots, u_n)\}$ if $w \in A$ and $S = \emptyset$ if $w \in \{\mathrm{tt}, \mathrm{ff}\}$,
- $\mathrm{depth}(\phi(t_1, \dots, t_n) = w) = \max_{j=1,\dots,n}(\mathrm{depth}(t_j = u_j)) + 1$.

(RC4) If $E, \mathbf{A} \vdash t_j = u_j$ for $j = 1, \dots, n$ and $E, \mathbf{A} \vdash E_i(u_1, \dots, u_n, \vec{p}) = w$ for some i such that $p \equiv p_i$, then
- $\mathrm{calls}_{\Phi_0}(p(t_1, \dots, t_n) = w) = \sum_{j=1,\dots,n} \mathrm{calls}_{\Phi_0}(t_j = u_j)$ $+ \min\Big\{\mathrm{calls}_{\Phi_0}(E_i(u_1, \dots, u_n, \vec{p}) = w) : p(\vec{v}) = E_i(\vec{v}, \vec{p}) \text{ is in } E\Big\}$,
- $\mathrm{space}(p(t_1, \dots, t_n) = w) = \bigcup_{j=1,\dots,n} \mathrm{space}(t_j = u_j) \cup S$ where $S = \mathrm{space}(E_i(u_1, \dots, u_n, \vec{p}) = w)$ for some $p(\vec{v}) = E_i(\vec{v}, \vec{p})$ in E such that $|\bigcup_{j=1,\dots,n} \mathrm{space}(t_j = u_j) \cup S|$ is least,
- $\mathrm{depth}(p(t_1, \dots, u_n) = w) = \max_{j=1,\dots,n}(\mathrm{depth}(t_j = u_j))$ $+ \min(\mathrm{depth}(E_i(u_1, \dots, u_n, \vec{p}) = w) : p(\vec{v}) = E_i(\vec{v}, \vec{p}) \text{ is in } E)$.

Clauses (RC1) – (RC3) are quite direct, only mildly complicated by the circumstance that we do not have variables over $\{\mathrm{tt}, \mathrm{ff}\}$. The last clause (RC4) is substantially more complex because we are working with non-deterministic programs: it, too, is very simple (without the "min" operation) for deterministic programs in which the recursive variables $p_1 \dots, p_k$ are all distinct. In any case, we will assume that these conditions are satisfied by all reasonable implementations of non-deterministic recursive programs and their associated, natural complexity measures—and a fortiori, that they are satisfied by the concrete computation models whose characterization as restricted recursive programs was explained in (1) – (3) of Section 5A.

LEMMA 5.2. *Suppose all the constants from A in a term t occur in the list $\vec{x} = (x_1, \dots, x_n)$, $E, \mathbf{A} \vdash t = w$, $\Phi_0 \subseteq \Phi$ and $C = \text{calls}_{\Phi_0}, \text{size}$ or depth; then there is a finite $\mathbf{U} \subseteq \mathbf{A}$ generated by $\vec{x}$ such that $E, \mathbf{U} \vdash t = w$ and*

$$C(\mathbf{U}, \vec{x}) \leq C(E, \mathbf{A})(\vec{x}). \tag{76}$$

OUTLINE OF PROOF. We need to prove the lemma separately for each complexity measure, since a different $\mathbf{U}$ may be needed in each case. The argument is by induction on the definition of $E, \mathbf{A} \vdash t = w$, again, and it is trivial in the base: if $w = w$ is an axiom, take $\mathbf{U}$ to be the structure with universe $\{w\}$ and $\text{eqdiag}(\mathbf{U}) = \emptyset$.

We consider two of the cases in the induction step.

Suppose $t \equiv \phi(t_1, \dots, t_n)$, so that for suitable $u_1, \dots, u_n$, $E, \mathbf{A} \vdash t_j = u_j$ for $j = 1, \dots, u_n$ and $\phi^{\mathbf{A}}(u_1, \dots, u_n) = w$. The induction hypotheses gives us for each j a substructure $\mathbf{U}_j \subseteq_p \mathbf{A}$ generated by $\vec{x}$ such that $E, \mathbf{U}_j \vdash t_j = u_i$ and the relevant complexity inequality is satisfied. Define $\mathbf{U}$ by

$$U = U_1 \cup \dots \cup U_n \cup S \quad (\text{where } S = \{w\} \text{ if } w \in A \text{ and } S = \emptyset \text{ otherwise}),$$
$$\text{eqdiag}(\mathbf{U}) = \text{eqdiag}(\mathbf{U}_1) \cup \cdots \text{eqdiag}(\mathbf{U}_n) \cup \{\phi(u_1, \dots, u_n) = w\},$$

so that $\mathbf{U}$ is generated by $\vec{x}$ and $E, \mathbf{U} \vdash t = w$ by (1) of Lemma 5.1. The complexity claim follows directly from the definitions.

Suppose $t \equiv p(t_1, \dots, t_{k_i})$. The induction hypothesis gives us again for each j a suitable $\mathbf{U}_j \subseteq_p \mathbf{A}$ satisfying the conclusion of the lemma, and for each i such that $p \equiv p_i$ and $E, \mathbf{A} \vdash E_i(\vec{u}, \vec{p}) = w$ a substructure $\mathbf{U}_{E_i}$ which is generated by $\vec{u}$ such that $E, \mathbf{U}_{E_i} \vdash E_i(\vec{u}, \vec{p}) = w$ and the relevant complexity inequality is satisfied. We choose one i for which this measure is least and define $\mathbf{U}$ by

$$U = U_1 \cup \dots \cup U_{k_i} \cup U_{E_i},$$
$$\text{eqdiag}(\mathbf{U}) = \text{eqdiag}(\mathbf{U}_1) \cup \cdots \text{eqdiag}(\mathbf{U}_n) \cup \text{eqdiag}(\mathbf{U}_{E_i}).$$

The required complexity inequality must be checked separately for each complexity measure, but it is quite easy. ⊣

THEOREM 5.3. *Suppose E is an n-ary Φ-recursive program which computes a partial function $f : A^n \rightharpoonup A_s$ in the Φ-structure $\mathbf{A}$, and for each $\mathbf{U} \subseteq_p \mathbf{A}$ set*

$$\overline{\alpha}_E^{\mathbf{U}}(\vec{x}) = w \iff E, \mathbf{U} \vdash E_0(\vec{x}, \vec{p}) = w. \tag{77}$$

The operation $\mathbf{U} \mapsto \overline{\alpha}_E^{\mathbf{U}}$ is a uniform process of $\mathbf{A}$ which computes f, and for each of the complexity measure $C = \text{calls}_{\Phi_0}, \text{size}$ or depth,

$$C(\alpha^{\mathbf{U}}, \vec{x}) \leq C(E, \mathbf{A})(t = w),$$

so that the same inequality holds for the intrinsic complexities,

$$C(\mathbf{A}, f, \vec{x}) \leq C(E, \mathbf{A})(t = w).$$

PROOF. The equivalence (77) defines a partial function $\overline{\alpha}_E^{\mathbf{U}} : U^n \rightharpoonup U_s$ by (2) of Lemma 5.1, and the argument that this operation satisfies the Homomorphism Axiom **II** is a simple extension of the argument for (1) of that lemma: given $\pi : \mathbf{U} \to \mathbf{V}$, define t^π for every closed $\Big(\Phi \cup \{p_1, \dots, p_k\} \cup A\Big)$-term by replacing each $u \in U$ which occurs in t by $\pi(u)$ and then verify by an easy induction on (RC1) – (RC4) that

$$E, \mathbf{U} \vdash t = w \Longrightarrow E, \mathbf{V} \vdash t^\pi = \pi(w).$$

It follows that

$$\begin{aligned}\overline{\alpha}_E^{\mathbf{U}}(\vec{x}) = w &\Longrightarrow E, \mathbf{U} \vdash E_0(\vec{x}, \vec{p}) = w \\ &\Longrightarrow E, \mathbf{V} \vdash E_0(\vec{\pi}(\vec{x}), \vec{p}) = \pi(w) \Longrightarrow \overline{\alpha}_E^{\mathbf{V}}(\vec{\pi}(x)) = \pi(w)\end{aligned}$$

as required.

Axiom **III** and the complexity inequalities follow immediately from Lemma 5.2, because the constants in $E_0(x_1, \dots, x_n, \vec{p})$ all occur in the list $(x_1, \dots, x_n)$. $\dashv$

The upshot is that a lower bound result for the intrinsic complexities defined in Section 2I applies to all recursive programs, and so also to all the computation models identified with restricted recursive programs in Section 5A.

5C. Turing machines with oracles (missing in this draft).

5D. Random access machines (missing in this draft).

§6. Concluding remarks (missing in this draft).

REFERENCES

[1997] P. BÜRGISSER, M. CLAUSEN, and M. AMIN SHOKOLLAHI, ***Algebraic complexity theory***, Springer, 1997. *3.*

[1992] P. BÜRGISSER, T. LICKTEIG, and M. SHUB, *Test complexity of generic polynomials*, ***Journal of Complexity***, vol. 8 (1992), pp. 203–215. *3, 41.*

[1992] P. BÜRGISSER and T. LICKTEIG, *Verification complexity of linear prime ideals*, ***Journal of pure and applied algebra***, vol. 81 (1992), pp. 247–267. *3, 27, 33.*

[2007] JOSEPH BUSCH, *On the optimality of the binary algorithm for the Jacobi symbol*, ***Fundamenta Informaticae***, vol. 76 (2007), pp. 1–11. *19.*

[2009] ———, *Lower bounds for decision problems in imaginary, norm-Euclidean quadratic integer rings*, ***Journal of Symbolic Computation***, vol. 44 (2009), pp. 683–689. *19.*

[2004] LOU VAN DEN DRIES and YIANNIS N. MOSCHOVAKIS, *Is the Euclidean algorithm optimal among its peers?*, ***The Bulletin of Symbolic Logic***, vol. 10 (2004), pp. 390–418. *3, 14, 17, 18, 19, 20, 21.*

[2009] ———, *Arithmetic complexity*, ***ACM Trans. Comput. Logic***, vol. 10 (2009), no. 1, pp. 1–49. *3, 13, 17, 18, 19, 20, 21, 23, 24.*

[1990] P. VAN EMDE BOAS, *Machine models and simulations*, 1990, in van Leeuwen [1990], pp. 1–66. .

[1938] G. H. HARDY and E. M. WRIGHT, ***An introduction to the theory of numbers***, Clarendon Press, Oxford, 1938, fifth edition (2000). *20, 24.*

[1952] STEPHEN C. KLEENE, ***Introduction to metamathematics***, D. Van Nostrand Co, North Holland Co, 1952. *43.*

[1973] D. E. KNUTH, ***The Art of Computer Programming. Fundamental Algorithms***, second ed., Addison-Wesley, 1973. *17, 19, 22.*

[1990] J. van Leeuwen (editor), *Handbook of theoretical computer science*, vol. A, Algorithms and Complexity, Elsevier and the MIT Press, 1990. *48, 49.*

[1991a] Yishay Mansour, Baruch Schieber, and Prasoon Tiwari, *A lower bound for integer greatest common divisor computations*, ***Journal of the Association for Computing Machinery***, vol. 38 (1991), pp. 453–471. *13, 17, 44.*

[1991b] ———, *Lower bounds for computations with the floor operation*, ***SIAM Journal on Computing***, vol. 20 (1991), pp. 315–327. *17, 44.*

[1963] J. McCarthy, *A basis for a mathematical theory of computation*, ***Computer programming and formal systems*** (P. Braffort and D Herschberg, editors), North-Holland, 1963, pp. 33–70. *43.*

[1991] João Meidânis, *Lower bounds for arithmetic problems*, ***Information Processing Letters***, vol. 38 (1991), pp. 83–87. *17.*

[1954] A. M. Ostrowski, *On two problems in abstract algebra connected with Horner's rule*, ***Studies presented to R. von Mises***, Academic Press, New York, 1954, pp. 40–48. *26.*

[1966] V. Ya. Pan, *Methods for computing values of polynomials*, ***Russian Mathematical Surveys***, vol. 21 (1966), pp. 105 – 136. *3, 27, 44.*

[1975] Vaughan Pratt, *Every prime has a succint certificate*, ***SIAM Journal of computing***, vol. 4 (1975), pp. 214 – 220. *13.*

[2008] ———, *Euclidean gcd is exponentially suboptimal: why gcd is hard to analyse*, unpublished manuscript, 2008. *21.*

[1967] J. Stein, *Computational problems associated with Racah Algebra*, ***Journal of Computational Physics***, vol. 1 (1967), pp. 397–405. *17.*

[1990] V. Strassen, *Algebraic complexity theory*, 1990, in van Leeuwen [1990], pp. 633–672. *44.*

[1967] Shmuel Winograd, *On the number of multiplications required to compute certain functions*, ***Proceedings of the National Academy of Sciences, USA***, vol. 58 (1967), pp. 1840 – 1842. *3, 27, 29, 44.*

[1970] ———, *On the number of multiplications required to compute certain functions*, ***Communications on pure and applied mathematics***, vol. 23 (1970), pp. 165 – 179. *3, 27.*

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF CALIFORNIA, LOS ANGELES
and
DEPARTMENT OF MATHEMATICS, UNIVERSITY OF ATHENS
E-mail: ynm@math.ucla.edu