The axiomatic derivation of absolute lower bounds

Yiannis N. Moschovakis
UCLA and University of Athens

Iowa State University, April 15, 2010

The Euclidean algorithm

For $x, y \in \mathbb{N} = \{0, 1, \ldots\},\ x \geq y \geq 1$,

$$(\varepsilon) \quad \boxed{\text{gcd}(x, y) = \text{if } (\text{rem}(x, y) = 0) \text{ then } y \text{ else } \text{gcd}(y, \text{rem}(x, y))}$$

where $\text{rem}(x, y)$ is the remainder of the division of x by y,

$$x = \text{iq}(x, y) \cdot y + \text{rem}(x, y) \quad (0 \leq \text{rem}(x, y) < y)$$

$$\begin{aligned} \text{calls}_\varepsilon(x, y) = &\text{ the number of divisions required to compute } \text{gcd}(x, y) \\ &\text{ by the Euclidean algorithm} \\ \leq &\ 2 \log(y) \quad (x \geq y \geq 2) \end{aligned}$$

- $\boxed{\text{Is the Euclidean optimal for computing gcd}(x, y) \text{ from rem?}}$
- $\boxed{\text{Is the Euclidean optimal for deciding coprimeness from rem?}}$

$$x \perp\!\!\!\perp y \iff \text{gcd}(x, y) = 1$$

A partial result

Theorem (van den Dries, ynm, 2004)

If an algorithm α decides the coprimeness relation $x \perp\!\!\!\perp y$ on $\mathbb{N}$ from the primitives $\leq, +, \dot{-}, \mathrm{iq}, \mathrm{rem}$, then for infinitely many a, b with $a > b$,

$$\mathrm{calls}_\alpha(a, b) \geq \mathrm{depth}_\alpha(a, b) > \frac{1}{10} \log \log a \qquad (*)$$

where $\mathrm{depth}_\alpha(a, b)$ is the least number of applications of the primitives which must be executed in sequence in the computation

- $\mathrm{depth}_\alpha(x, y)$ is a natural parallel time complexity
- The result is one log short of establishing the optimality of the Euclidean (and one log $= \infty$ in this context)
- (*) holds for all sufficiently large a, b such that
  - $a^2 = 1 + 2b^2$ (solutions of Pell's equation),
  - or $a = F_{n+1}, b = F_n$ (successive Fibonacci numbers)
- Claim: This applies to all algorithms from the specified primitives

Outline

Slogan: *Absolute lower bound results*
are the undecidability facts about decidable problems

. . . and so their precise formulation should be a matter of logic

(1) Tweak logic (a bit) so it applies smoothly to computation theory
(2) Three (simple) axioms for elementary algorithms,
in the style of *abstract model theory*
(3) Verify that the axioms are satisfied by all computation models
(4) Derive lower bounds from the axioms

Is the Euclidean algorithm optimal among its peers? (with vDD, 2004)
Arithmetic complexity (with vDD, 2009)

Y. Mansour, B. Schieber, and P. Tiwari (1991)
A lower bound for integer greatest common divisor computations,
Lower bounds for computations with the floor operation

J. Meidânis (1991): *Lower bounds for arithmetic problems*

(Partial) algebras

- A (partial, pointed) algebra is a structure $\mathbf{M} = (M, 0, 1, \Phi^{\mathbf{M}})$ where $0, 1 \in M$, Φ is a set of function symbols (the vocabulary) and $\Phi^{\mathbf{M}} = \{\phi^{\mathbf{M}}\}_{\phi\in\Phi}$, where each primitive $\boxed{\phi^{\mathbf{M}} : M^{n_\phi} \rightharpoonup M}$ is a partial function of arity n_ϕ determined by the symbol ϕ

$$\mathbf{N}_\varepsilon = (\mathbb{N}, 0, 1, \mathrm{rem}), \text{ the Euclidean algebra}$$

$$\mathbf{N}_u = (\mathbb{N}, 0, 1, S, \mathrm{Pd}), \text{ the } \textit{unary numbers}$$

$$\mathbf{N}_b = (\mathbb{N}, 0, 1, \mathrm{Parity}, \mathrm{iq}_2, (x \mapsto 2x), (x \mapsto 2x+1)), \text{ the } \textit{binary numbers}$$

$$\mathbf{N} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, \cdot), \text{ the full algebra of arithmetic}$$

$$\mathbf{N}_\varepsilon\restriction U = (U, 0, 1, \mathrm{rem}\restriction U) \text{ where } \{0,1\} \subseteq U \subseteq \mathbb{N} \text{ and}$$

$$(\mathrm{rem}\restriction U)(x,y) = w \iff x, y, w \in U \ \&\ \mathrm{rem}(x,y) = w$$

- The diagram of a Φ-algebra is the set of its basic identities,

$$\mathrm{diag}(\mathbf{M}) = \{(\phi, \vec{x}, w) : \phi^{\mathbf{M}}(\vec{x}) = w\}$$

- $\boxed{\mathbf{M} \text{ is completely determined by } M, 0, 1 \text{ and } \mathrm{diag}(\mathbf{M})}$

Homomorphisms, embeddings and subalgebras

- A homomorphism $\pi : \mathbf{U} \to \mathbf{M}$ of one Φ-algebra into another is any function $\pi : U \to M$ such that

$$\pi(0^{\mathbf{U}}) = y \iff y = 0^{\mathbf{M}}, \quad \pi(1^{\mathbf{U}}) = y \iff y = 1^{\mathbf{M}},$$

  and for all $\phi \in \Phi, x_1, \dots, x_n, w \in U$,

$$\phi^{\mathbf{U}}(x_1, \dots, x_n) = w \implies \phi^{\mathbf{M}}(\pi x_1, \dots, \pi x_n) = \pi w$$

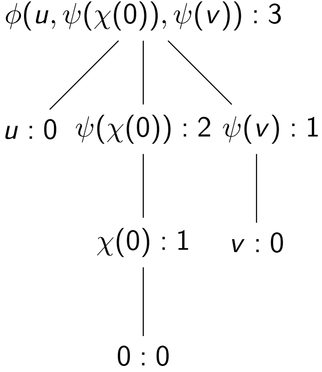
- π is an embedding if, in addition, it is one-to-one
- Subalgebras:

$$\begin{aligned} \mathbf{U} \subseteq_p \mathbf{M} \iff & \{0, 1\} \subseteq U \subseteq M \\ & \ \& \ \text{the identity } I : U \rightarrowtail M \text{ is an embedding} \\ \iff & \{0, 1\} \subseteq U \subseteq M \ \& \ \mathrm{diag}(\mathbf{U}) \subseteq \mathrm{diag}(\mathbf{M}) \end{aligned}$$

- We will use finite subalgebras $\mathbf{U} \subseteq_p \mathbf{M}$ to represent calls to the primitives executed during a computation in $\mathbf{M}$

Terms and depth

The terms of a vocabulary Φ: $\boxed{t :\equiv 0 \mid 1 \mid v \mid \phi(t_1, \ldots, t_n)}$



$$\text{depth}(0) = \text{depth}(1) = \text{depth}(v) = 0,$$
$$\text{depth}(\phi(t_1, \ldots, t_n)) = \max(\text{depth}(t_1), \ldots, \text{depth}(t_n)) + 1$$

Term evaluation and algebra generation

- If $\vec{v} = v_1, \dots, v_n$ includes all the variables which occur in t,

$$t(\vec{v}) = (t, \vec{v})$$

- For any subalgebra $\mathbf{U} \subseteq_p \mathbf{M}$ and any $\vec{x} = x_1, \dots, x_n \in U$,

$$t^{\mathbf{U}}[\vec{x}] = \text{ the value of } t \text{ in } \mathbf{U} \text{ when } \vec{v} := \vec{x} \text{ (if it converges)}$$

- $G_m[\mathbf{U}, \vec{x}] = \{t^{\mathbf{U}}[\vec{x}] : \mathrm{depth}(t(\vec{v})) \leq m \ \&\ t^{\mathbf{U}}[\vec{x}]\downarrow\}$
- $G_\infty[\mathbf{U}, \vec{x}] = \bigcup_m G_m[\mathbf{U}, \vec{x}] =$ the subalgebra of $\mathbf{U}$ generated by $\vec{x}$
- $\boxed{\mathrm{depth}(w, \mathbf{U}, \vec{x}) = \min\{m : w \in G_m[\mathbf{U}, \vec{x}]\}}$

$\mathrm{depth}(w, \mathbf{U}, \vec{x})$ is the least number of applications of the primitives which must be executed in sequence to construct w from $\vec{x}$ in $\mathbf{U}$

- $\boxed{\mathrm{depth}_{\vec{x}}(\mathbf{U}) = \max\{\mathrm{depth}(w, \mathbf{U}, \vec{x}) : w \in U\} \quad (\mathbf{U} \text{ generated by } \vec{x})}$

The depth complexity of values of a function

Basic principle: *If an algorithm α computes $f : M^n \to M$ from the primitives of* $\mathbf{M}$, *then* $f(\vec{x}) \in G_\infty[\mathbf{M}, \vec{x}]$ and

$$\boxed{\mathrm{calls}_\alpha(\vec{x}) \geq \mathrm{depth}(f(\vec{x}), \mathbf{M}, \vec{x})}$$

- The value must be constructed by the primitives from the input
- Can be used to derive lower bounds for functions which grow fast, e.g., multiplication or the Ackermann function (in unary or binary arithmetic)

We can also exploit gaps in $G_m[\mathbf{M}, a]$ when a is large compared to m:

$$G_m[\mathbf{N}_b, a] : 0 \quad 1 \quad 2 \quad \cdots \quad 2^{m+1} - 1 \quad \boxed{\quad \text{gap} \quad} \quad \mathrm{iq}(a, 2^m) \cdots \; a \; \cdots$$

Theorem (van den Dries)

If an algorithm α *computes* $\mathrm{gcd}(x, y)$ *from*

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

then for all $a > b$ *such that* $a^2 = 1 + 2b^2$ (Pell pairs)

$$\mathrm{calls}_\alpha(a+1, b) \geq \mathrm{depth}(a+1, b) \geq \frac{1}{4}\sqrt{\log\log b}$$

This is the best known lower bound for the gcd from primitives which include multiplication

- *This method cannot yield lower bounds for decision problems*

(because their value (0 or 1) is available with no computation)

I The Locality Axiom

An algorithm α *of arity n of an algebra* $\mathbf{M} = (M, 0, 1, \Phi^{\mathbf{M}})$ *assigns to each subalgebra* $\mathbf{U} \subseteq_p \mathbf{M}$ *an n-ary (strict) partial function*

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

- An $\mathbf{M}$-algorithm α "computes" a partial function $\overline{\alpha}^{\mathbf{M}} : M^n \rightharpoonup M$ using the primitives of $\mathbf{M}$ as oracles, and it can be naturally localized (restricted) to arbitrary subalgebras of $\mathbf{M}$

We write

$$\boxed{\mathbf{U} \vdash \overline{\alpha}(\vec{x}) = w \iff \vec{x} \in U^n, w \in U \text{ and } \overline{\alpha}^{\mathbf{U}}(\vec{x}) = w}$$

II The Homomorphism Axiom

If α *is an n-ary algorithm of* $\mathbf{M}$*,* $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{M}$*, and* $\pi : \mathbf{U} \to \mathbf{V}$ *is a homomorphism, then*

$$\mathbf{U} \vdash \overline{\alpha}(\vec{x}) = w \implies \mathbf{V} \vdash \overline{\alpha}(\pi\vec{x}) = \pi w \quad (x_1, \ldots, x_n, w \in U)$$

In particular, if $\mathbf{U} \subseteq_p \mathbf{M}$, then $\overline{\alpha}^{\mathbf{U}} \sqsubseteq \overline{\alpha}^{\mathbf{M}}$

- When asked for $\phi^{\mathbf{U}}(\vec{x})$, the oracle for ϕ may consistently provide $\phi^{\mathbf{V}}(\pi\vec{x})$, if π is a homomorphism.

III The Finiteness Axiom

If α is an n-ary algorithm of $\mathbf{M}$*, then*

$$\mathbf{M} \vdash \overline{\alpha}(\vec{x}) = w$$

$$\Longrightarrow \textit{ there is a finite } \mathbf{U} \subseteq_p \mathbf{M} \textit{ generated by } \vec{x} \textit{ such that } \mathbf{U} \vdash \overline{\alpha}(\vec{x}) = w$$

- The algorithm must construct the arguments $\vec{u}$ for every call $\phi(\vec{u})$ to the primitives, and so the entire computation takes place within the subalgebra generated by the input $\vec{x}$
- A computation of $\overline{\alpha}^{\mathbf{M}}(\vec{x})$ from the primitives of $\mathbf{M}$ is finite, and so it makes finitely many calls to the primitives
- Intuitively, the axiom is satisfied by any $\mathbf{U}$ whose diagram includes $(\phi, \vec{u}, \phi^{\mathbf{M}}(\vec{u}))$ for every call to $\phi^{\mathbf{M}}$ made by α during the computation

All elementary algorithms satisfy I – III (with suitable $\mathbf{M}$)

- Explicit computation: $\overline{\alpha}^{\mathbf{M}}(\vec{x}) = t^{\mathbf{M}}[\vec{x}]$, where $t(\vec{v})$ is a Φ-term
- $\overline{\alpha}^{\mathbf{M}}$ is the partial function computed a fixed recursive (McCarthy) program P in the vocabulary Φ
- $\overline{\alpha}^{\mathbf{M}}$ is computed from $\Phi^{\mathbf{M}}$ by any of the familiar machine models of computation with oracles—register machines, Random Access Machines (of all kinds), Turing machines, etc.
- $\overline{\alpha}^{\mathbf{M}}$ is computed in PCF (typed λ-calculus) above the algebra $\mathbf{M}$
- $\overline{\alpha}^{\mathbf{M}}$ by computed by non-deterministic versions of any of these

Note: For computation models (e.g., Turing machines), the functions built into the model must be included among the primitives of $\mathbf{M}$.

Abstract algorithms

- I, Locality Axiom: An abstract algorithm α of arity n of an algebra $\mathbf{M} = (M, 0, 1, \Phi^{\mathbf{M}})$ assigns to each subalgebra $\mathbf{U} \subseteq_p \mathbf{M}$ an n-ary partial function $\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U$

$$\mathbf{U} \vdash \overline{\alpha}(\vec{x}) = w \iff \overline{\alpha}^{\mathbf{U}}(\vec{x}) = w$$

- II, Homomorphism Axiom: If $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{M}$, and $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism, then

$$\mathbf{U} \vdash \overline{\alpha}(\vec{x}) = w \implies \mathbf{V} \vdash \overline{\alpha}(\pi\vec{x}) = \pi w$$

Set $\boxed{\boxed{\mathbf{U} \vdash_g \overline{\alpha}(\vec{x}) = w \iff \mathbf{U} \text{ is generated by } \vec{x} \;\&\; \mathbf{U} \vdash \overline{\alpha}(\vec{x}) = w}}$

- III, Finiteness Axiom:

$$\mathbf{M} \vdash \overline{\alpha}(\vec{x}) = w \implies \textit{there is a finite } \mathbf{U} \subseteq_p \mathbf{M} \textit{ such that } \mathbf{U} \vdash_g \overline{\alpha}(\vec{x}) = w$$

Complexity functions for abstract algorithms

Suppose α is an n-ary abstract algorithm of $\mathbf{M}$ and $\mathbf{M} \vdash \overline{\alpha}(\vec{x}) = w$

- $\boxed{\mathrm{calls}_\alpha(\vec{x}) = \min\{|\mathrm{diag}(\mathbf{U})| : \mathbf{U} \vdash_g \overline{\alpha}(\vec{x}) = w\}}$

  (the least number of calls α must execute to compute $\overline{\alpha}^{\mathbf{M}}(\vec{x})$)

- $\boxed{\mathrm{size}_\alpha(\vec{x}) = \min\{|U \setminus \{0, 1, \vec{x}\}| : \mathbf{U} \vdash_g \overline{\alpha}(\vec{x}) = w\}}$

  (the least number of elements of $\mathbf{M}$ (other than $0, 1, \vec{x}$) that α must see to compute $\overline{\alpha}^{\mathbf{M}}(\vec{x})$)

- $\boxed{\mathrm{depth}_\alpha(\vec{x}) = \min\{\mathrm{depth}_{\vec{x}}(\mathbf{U}) : \mathbf{U} \vdash_g \overline{\alpha}(\vec{x}) = w\}}$

  (the least number of calls α must execute in sequence to compute $\overline{\alpha}^{\mathbf{M}}(\vec{x})$)

Thm $\boxed{\mathrm{depth}_\alpha(\vec{x}) \le \mathrm{size}_\alpha(\vec{x}) \le \mathrm{calls}_\alpha(\vec{x})}$

These notions agree with standard definitions for concrete algorithms

⋆ The certification relation $\mathbf{U} \Vdash f(\vec{x}) = w$

Suppose $f : M^n \to M$, $\mathbf{U} \subseteq_p \mathbf{M}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{M}$ respects f at $\vec{x}$ if

$$\vec{x} \in U \ \& \ f(\vec{x}) \in U \ \& \ \pi(f(\vec{x})) = f(\pi(\vec{x}))$$

$$\boxed{\mathbf{U} \Vdash f(\vec{x}) = w \iff \text{every homomorphism } \pi : \mathbf{U} \to \mathbf{M} \text{ respects } f \text{ at } \vec{x}}$$

Lemma

(1) *If an abstract* $\mathbf{M}$*-algorithm* α *computes* $f : M^n \to M$*, then*

$$\mathbf{U} \vdash_g \overline{\alpha}(\vec{x}) = w \implies \mathbf{U} \Vdash f(\vec{x}) = w$$

(2) *If some abstract* $\mathbf{M}$*-algorithm* α *computes* $f : M^n \to M$ *and* $f(\vec{x}) = w$*, then there is a finite* $\mathbf{U}$ *such that* $\mathbf{U} \Vdash f(\vec{x}) = w$

Proof. By the Homomorphism and Finiteness axioms.

Certificates

For coprimeness in the Euclidean algebra $\mathbf{N}_\varepsilon = (\mathbb{N}, 0, 1, \mathrm{rem})$,

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

certifies that $x \perp\!\!\!\perp y$.

To prove (by this method) that the Euclidean is number-of-calls optimal for coprimeness, we would need to show that every certificate for $x \perp\!\!\!\perp y$ in $\mathbf{N}_\varepsilon$ is at least as large as $n + 2 = \mathrm{calls}_\varepsilon(x, y)$ for infinitely many (x, y).

Complexity functions for computable functions

Suppose some abstract $\mathbf{M}$-algorithm α computes $f : M^n \to M$.

- $\mathrm{calls}_f(\mathbf{M}, \vec{x}) = \min\{|\mathrm{diag}(\mathbf{U})| : \mathbf{U} \Vdash f(\vec{x}) = w\} \leq \mathrm{calls}_\alpha(\vec{x})$
- $\mathrm{size}_f(\mathbf{M}, \vec{x}) = \min\{|U \setminus \{0, 1, \vec{x}\}| : \mathbf{U} \Vdash f(\vec{x}) = w\} \leq \mathrm{size}_\alpha(\vec{x})$
- $\mathrm{depth}_f(\mathbf{M}, \vec{x}) = \min\{\mathrm{depth}_{\vec{x}}(\mathbf{U}) : \mathbf{U} \Vdash f(\vec{x}) = w\} \leq \mathrm{depth}_\alpha(\vec{x})$

Thm $\mathrm{depth}_f(\mathbf{M}, \vec{x}) \leq \mathrm{size}_f(\mathbf{M}, \vec{x}) \leq \mathrm{calls}_f(\mathbf{M}, \vec{x})$

Interpretation: $\mathrm{depth}_f(\mathbf{M}, \vec{x})$ is the least number of calls to the primitives which must be executed in sequence by any algorithm which computes $f : M^n \to M$ from the primitives of $\mathbf{M}$

- These complexities are most useful for relations $R \subseteq M^n$ whose characteristic functions take values in $\{0, 1\}$

Outline of a proof

Theorem (van den Dries, ynm)

In $\mathbf{M} = (\mathbb{N}, 0, 1, \leq, +, \dot{-}, \mathrm{iq}, \mathrm{rem})$*: if* a *is sufficiently large and* $\boxed{a^2 = 1 + 2b^2}$ *or* $\boxed{a = F_{n+1}, b = F_n}$*, then*

$$\mathrm{depth}_{\perp\!\!\!\perp}(\mathbf{M}, a, b) > \frac{1}{10} \log \log(a) \qquad (*)$$

So if α *decides coprimeness in* $\mathbf{M}$*, then* (*) *holds with* $\mathrm{depth}_\alpha(a, b)$

Pf. If $2^{2^{4m+6}} \leq a$, then every $X \in G_m[\mathbf{M}, a, b]$ can be written uniquely as

$$X = \frac{x_0 + x_1 a + x_2 b}{x_3} \qquad \text{with } x_i \in \mathbb{Z}, \quad |x_i| \leq 2^{2^{4m}}$$

and we can define $\pi : G_m[\mathbf{M}, a, b] \rightarrowtail \mathbf{M}$ letting $\lambda = 1 + a!$,

$$\pi(X) = \frac{x_0 + x_1 \lambda a + x_2 \lambda b}{x_3}, \text{ so } (\pi(a), \pi(b)) = (\lambda a, \lambda b)$$

The "universal constant" $\frac{1}{10}$

Detailed version of result

In $\mathbf{M} = (\mathbb{N}, 0, 1, \leq, +, \dot{-}, \mathrm{iq}, \mathrm{rem})$*: if* $1 < \xi < 2$*,* ξ *is a quadratic algebraic irrational,* $C > 0$*,* a *is sufficiently large, and* $a \perp\!\!\!\perp b$*, then*

$$\frac{1}{Cb^2} < \left|\xi - \frac{a}{b}\right| < \frac{1}{b^2} \implies \mathrm{depth}_{\perp\!\!\!\perp}(a, b) > \frac{1}{K} \log\log a,$$

with $K \geq 2\log(\log C + 19)$

- Liouville: for sufficiently large C, infinitely many a, b satisfy the hypothesis
- With $\xi = \sqrt{2}$ and $a^2 = 1 + 2b^2$, we can take $C = 5, K \geq 10$
- With $\xi = \frac{1}{2}(1 + \sqrt{5})$ and $a = F_{n+1}, b = F_n$, we can take again, $C = 5, K \geq 10$

$\mathbf{M} = (\mathbb{N}, 0, 1, \mathrm{Parity}, \mathrm{iq}_2, \leq, +, \dot{-}, \text{Presburger functions})$

- (van den Dries, ynm) *If $R(x)$ is one of the relations*

  $$\boxed{x \text{ is prime}}, \quad \boxed{x \text{ is a perfect square}}, \quad \boxed{x \text{ is square free}},$$

  then for some $r > 0$ and infinitely many a,

  $$\mathrm{depth}_R(\mathbf{M}, a) > r \log(a)$$

- (van den Dries, ynm) *For some $r > 0$ and infinitely many a, b,*

  $$\mathrm{depth}_{\perp\!\!\!\perp}(\mathbf{M}, a, b) > r \log(\max(a, b))$$

- (Joe Busch) *If $R(x, p) \iff \boxed{x \text{ is a square } \bmod p}$,*
  then for some $r > 0$ and a sequence (a_n, p_n) with $p_n \to \infty$,

  $$\mathrm{depth}_R(\mathbf{M}, a_n, p_n) > r \log(p_n)$$

In the last two examples, the results match up to a multiplicative constant well-known known binary algorithms, so these are optimal

Non-uniform complexity

What if you are only interested in deciding $R(\vec{x})$ for n-bit numbers ($< 2^n$) and you are willing to use a different algorithm for each n?

- The lookup algorithm: For any k-ary relation R on $\mathbb{N}$ and each n, there is an $\mathbf{N}_b$-term (with conditionals) $t_n(\vec{v})$ of depth $\leq n = \log(2^n)$ which decides $R(\vec{x})$ for all $\vec{x} < 2^n$.

- Non-uniform lower bounds on depth are never greater than log
- The best ones establish the optimality of the lookup algorithm and are most interesting when some uniform algorithm matches the lookup algorithm up to a multiplicative constant
- They are quite easy for Presburger primitives

Coprimeness from division, non-uniformly

Theorem (van den Dries, ynm)
Let $\mathbf{M} = (\mathbb{N}, 0, 1, \leq, +, \dot{-}, \mathrm{iq}, \mathrm{rem})$ *and for each* n, *let*

$$x \perp\!\!\!\perp_n y \iff x, y < 2^n \ \&\ \gcd(x, y) = 1.$$

There is some $r > 0$, *such that for all sufficiently large* n, *there are* $a, b < 2^n$ *such that*

$$\mathrm{calls}_{\perp\!\!\!\perp_n}(\mathbf{M}, a, b) \geq \mathrm{size}_{\perp\!\!\!\perp_n}(\mathbf{M}, a, b) > r \log n \qquad (**)$$

So if n *is large enough and* α *decides coprimeness in* $\mathbf{M}$ *for all* $x, y < 2^n$, *then* $(**)$ *holds with* $\mathrm{calls}_\alpha(a, b), \mathrm{size}_\alpha(a, b)$ *on the left*

- I do not know how to get the corresponding result for $\mathrm{depth}_{\perp\!\!\!\perp_n}$

Concluding remarks

(1) A technique for deriving lower bounds for decision problems which are absolute, i.e., they hold of all computational models

(2) Main limitation: in its current version, it only yields lower bounds which are no better than $O(n)$ (linear in the length of the input)

(3) *Problem*: prove that the Euclidean algorithm is optimal for computing the gcd in the algebra $\mathbf{N}_\varepsilon = (\mathbb{N}, 0, 1, \mathrm{rem})$

(4) *Problem*: prove an $O(n^2)$ lower bound for *primality* in $\mathbf{N}_b = (\mathbb{N}, 0, 1, \mathrm{Parity}, \mathrm{iq}_2, (x \mapsto 2x), (x \mapsto 2x + 1))$

Comment: (4) may need some number theory, but it will also need some logical analysis of computation (since the entire input is known in $O(n)$ steps)