CAMBRIDGE

Cambridge University Press
978-1-108-41558-3 — Abstract Recursion and Intrinsic Complexity
Yiannis N. Moschovakis
Excerpt
More Information

INTRODUCTION

This is the (somewhat polished) present state of an evolving set of lecture notes that I have used in several courses, seminars and workshops, mostly at UCLA and in the *Graduate Program in Logic and Algorithms* (MPLA) at the University of Athens. The general subject is the theory of *abstract* (first-order) *recursion* and its relevance to the *foundations of the theory of algorithms and computational complexity*, but the work on this broad project is very incomplete and so the choice of topics that are covered is somewhat eclectic.

The preliminary Chapter 1 gives a brief, elementary exposition of some basic facts and examples and helps make the material which follows accessible to students and researchers with varying backgrounds. After that, the book naturally splits into two, roughly equal parts according to the title: Part I (Chapters 2 – 3) on *abstract recursion* and Part II (Chapters 4 – 9) on *intrinsic complexity*.

– Chapter 2 introduces *recursive* (McCarthy) *programs* on abstract structures and develops their elementary theory. There is little that is new here, other than Vaughan Pratt's very interesting nondeterministic algorithm for coprimeness in Section 2E, but I do not know of another easily accessible, self-contained and reasonably complete source for this material.

– Chapter 3 introduces the natural *complexity measures* for recursive programs and establishes their basic properties. There is some novelty in approach, especially as the complexity measures are defined directly for the programs and so are independent of any particular "implementation of recursion"; and there are also some new results, most notably Theorems 3B.9 and 3B.12 which are due to Anush Tserunyan and have (I think) substantial foundational significance.

Part II is about the derivation of robust and widely applicable lower bounds for problems (especially) in arithmetic and algebra, and perhaps the simplest way to introduce my take on this is to give a fairly precise formulation of a fundamental conjecture about an ancient object of mathematical study.

The *Euclidean algorithm* (on the natural numbers, using division) can be specified succinctly by the *recursive equation*

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

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE
Cambridge University Press
978-1-108-41558-3 — Abstract Recursion and Intrinsic Complexity
Yiannis N. Moschovakis
Excerpt
More Information

where $\mathrm{rem}(x, y)$ is the remainder in the division of x by y. It computes *the greatest common divisor* of x and y when $x, y \geq 1$ and it is an algorithm *from* (*relative to*) the remainder function rem and the relation eq_0 of equality with 0: meaning that in its execution, ε has access to "oracles" which provide on demand the value $\mathrm{rem}(s, t)$ for any s and $t \neq 0$ and the truth value of $\mathrm{eq}_0(s)$. It is not hard to prove that

$$(*) \qquad c_\varepsilon(x, y) \leq 2 \log y \leq 2 \log x \quad (x \geq y \geq 2),$$

where $c_\varepsilon(x, y)$ is the number of divisions (calls to the rem-oracle) required for the computation of $\gcd(x, y)$ by the Euclidean and logarithms are to the base 2. Much more is known about $c_\varepsilon(x, y)$, but this upper bound suggests one plausible formulation of the Euclidean's (worst-case) *weak optimality*:

Main Conjecture. *For every algorithm* α *from* rem *and* eq_0 *which computes* $\gcd(x, y)$ *when* $x, y \geq 1$, *there is a number* $r > 0$, *such that for infinitely many pairs* (x, y) *with* $x > y \geq 1$,

$$c_\alpha(x, y) > r \log x,$$

where $c_\alpha(x, y)$ is the number of calls to the rem-oracle that α makes in the computation of $\gcd(x, y)$.

This is a classical fact about the Euclidean algorithm, taking for example the pairs (F_{n+3}, F_{n+2}) of successive Fibonacci numbers, cf. Problems x1C.8, x1C.9. The general case is open, probably not easy and certainly not precise as it stands, without specifying *what algorithms it is about* and what it means for an algorithm *to call an oracle* in the course of a *computation*.

Now, there are Turing machines which compute $\gcd(x, y)$ making no oracle calls at all, simply because $\gcd(x, y)$ is Turing computable—so that's not it.

In fact, there is no generally accepted, rigorous definition of *what algorithms are*. This is not a problem when we study particular algorithms, which are typically specified precisely in some form or other without any need to investigate whether *all relevant algorithms* can be similarly specified. In *Complexity Theory*—and especially when we want to establish *lower bounds* for some measure of computational complexity—the standard methodology is to ground proofs on rigorously defined *models of computation*, such as Turing machines, register or random access machines, decision trees, straight line programs, etc., and sometimes also on specific *representations of the input*, e.g., *unary* or *binary* notation for natural numbers, *adjacency matrices* for graphs, etc. There is a problem with this practice, when we try to compare lower bound results obtained for different models, typically attacked by establishing *simulations* of one model by another, cf. van Emde Boas [1990]; and this problem becomes acute when we want to prove *absolute* (or at least widely applicable) lower bounds which are small, polynomial or even linear (in the length of the

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE
Cambridge University Press
978-1-108-41558-3 — Abstract Recursion and Intrinsic Complexity
Yiannis N. Moschovakis
Excerpt
More Information

input) as in the Main Conjecture, generally less complex than the standard simulations.

So there are two equally important aims of research in this area:

One is to derive *lower bounds for mathematical problems*; the other is to develop a foundational framework in which one may be able to prove (or at least argue convincingly) that these bounds are *absolute*, that they restrict *all relevant algorithms*. The first of these naturally requires mathematical tools from the area in which the problems arise; and the second inevitably involves logic.

Recursion gets in the picture because there are both foundational arguments and mathematical results which support the view that *all elementary*[1] *algorithms can be faithfully expressed by recursive programs*, so that lower bounds established for them should be absolute, cf. Moschovakis [1984], [1989a], [1998], [2001]. This connection has motivated much of the work reported here, but it is not our topic.

Here I will take a different approach to the derivation and justification of robust lower bounds, which is more widely applicable and does not tie us to any specific foundational view of what algorithms are.

In Chapter 4, which is the heart of this book, we formulate three simple axioms about algorithms in the style of *abstract model theory*. These are bundled into the notion of a *uniform process* of an arbitrary (first order) structure: all *concrete* algorithms specified by computation models induce uniform processes, as do their usual *nondeterministic* versions. Uniform processes can "compute" functions that are not computable, they are not about that; but they carry a rich complexity theory which, when applied to concrete algorithms yields non-trivial lower bounds, in some cases optimal, absolutely or up to a multiplicative constant.

For a sample result, 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 $P \subseteq A^n$ is an n-ary relation on A and let $\Phi_0 \subseteq \Phi$. From these data, we will define a function

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

[1] There are algorithms whose implementations print output (or drop bombs), ask "the user" if she prefers business or coach class and may never terminate. In this book we confine ourselves to pure, finitary algorithms which compute partial functions or decide relations from given partial functions and relations, for which complexity theory is most fully developed. The extension of most of what we say to algorithms with *side effects* or *interaction* requires combining the methods we will use with classical *domain theory*, introduced by Scott and Strachey [1971] and richly developed by Scott and many others since then, especially the early Plotkin [1977], [1983]. It is not as different from what we will be doing as one might think, but we will not go into it here.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE
Cambridge University Press
978-1-108-41558-3 — Abstract Recursion and Intrinsic Complexity
Yiannis N. Moschovakis
Excerpt
More Information

the *intrinsic* calls(Φ_0)*-complexity function* of P, such that *if* α *is any* (deterministic or nondeterministic) *algorithm from* $\mathbf{\Phi}$ *which decides* P*, then for all* $\vec{x} \in A^n$,

$(*)$ $c(\vec{x}) \leq$ the number of calls to primitives in $\mathbf{\Phi}_0$ that α must execute to decide $P(\vec{x})$ from $\mathbf{\Phi}$.

This is a theorem if α is expressed by a concrete algorithm from $\mathbf{\Phi}$ 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*; and it is not trivial, e.g., we will show that *if* $x \perp\!\!\!\perp y$ *is the coprimeness relation on* $\mathbb{N}$ *and*

$$c = \text{calls}(\text{rem})((\mathbb{N}, \text{rem}, \text{eq}_0), \perp\!\!\!\perp) : \mathbb{N}^2 \to \mathbb{N},$$

then for infinitely many pairs (a, b) *with* $a > b$,

$$(**) \qquad c(a, b) > \frac{1}{10} \log\log a.$$

This follows from the (much stronger) Theorem 6C.5, an abstract version of one of the main results in van den Dries and Moschovakis [2004]. It gives a (very) partial result towards the Main Conjecture, one log below what we would like to prove—but Vaughn Pratt's nondeterministic algorithm for coprimeness in Theorem 2E.2 suggests that the conjecture may hold only for deterministic algorithms.

The main tool for defining the intrinsic complexities and deriving lower bounds for them is the *homomorphism method*, an abstract and mildly extended version of the *embedding method* developed in van den Dries and Moschovakis [2004], [2009]. We will use it in Chapters 5 – 8 to get somewhat strengthened versions of some of the lower bound results about arithmetic in these two papers and then again in Chapter 9 to get similar results in algebra. Few of these applications in the last two Parts are new: my main aim here is to explain the homomorphism method, illustrate its applicability in two different areas and (primarily) to identify some basic notions of the theory of computational complexity which (perhaps) have not been noticed.

Notice that **this is not a textbook on computability and complexity**, a core part of Computer Science which is covered in many excellent books including the classical Papadimitriou [1994]; **it is not a textbook on Turing computability and recursion on** $\mathbb{N}$, a huge subject amply covered in classical texts like Kleene [1952], Davis [1958] and Rogers [1967] and many more recent ones; and **it is definitely not a textbook on arithmetic and algebraic complexity**, not even a good introduction to these vast research areas about which I really know very little. It is natural to assume for some of the discussion that the reader knows something about these subjects, but the rigorous development of the material is substantially self-contained and limited to a few results which (I hope)

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-1-108-41558-3 — Abstract Recursion and Intrinsic Complexity
Yiannis N. Moschovakis
Excerpt
More Information

throw some light on the central problem of *deriving and justifying absolute lower complexity bounds*.

The exposition is elementary, aimed at advanced undergraduates, graduate students and researchers in mathematics and computer science with some knowledge of logic, a good understanding of the basic facts about algorithms and computability and an interest in foundational questions. Many of the (roughly) 250 problems are very easy, to test understanding, but there are also more challenging ones, sometimes marked with an asterisk * and a few that I cannot do, marked "Open Problem".

I have tried hard to assign results to those who proved them and to give correct and useful references to the literature, but this is not the place to look for a history of recursion and its interaction with computability—another vast and complex topic which is way out of my expertise and certainly not of the moment.

Yiannis N. Moschovakis
Santa Monica, CA and Paleo Faliro, Greece

Acknowledgments.

My greatest debt is to Lou van den Dries, whose insights in van den Dries [2003] led to a fruitful (and very pleasant) collaboration that produced van den Dries and Moschovakis [2004], [2009] and ultimately led to Part II of this book.

I am grateful to Vaughan Pratt and Anush Tserunyan for letting me include in this book unpublished results of theirs; to Vasilis Paschalis and Tyler Arant for chasing errors and typos in Parts I and II respectively—and I know they must have missed some, it's OK; and to my wife, always, and for many things other than her help with this book.

To go farther than this, I would need to put down the many logicians, philosophers and computer scientists who have informed my understanding of logic, recursion, algorithms and the connections among these subjects, including Stephen Kleene, John McCarthy, Dana Scott and many, many others—too long a list to put down here and certainly not unique to me.

Finally, I want to thank the hundreds of students who have taken courses or wrote M.Sc. or Ph.D. Theses with me on these topics, mostly at UCLA and the University of Athens, including the *Graduate Program in Logic, Algorithms and Computation* (MPLA). It is sometimes said that we learn more from our students than they learn from us and perhaps this is true of me; in any case, there is no doubt that I have enjoyed the process, very much.

© in this web service Cambridge University Press www.cambridge.org