Arnold Beckmann Christine Gaßner Benedikt Löwe (Eds.)

Logical Approaches to Barriers in Computing and Complexity

International Workshop, Greifswald, Germany, February 2010

Organized by the *Alfried Krupp Wissenschaftskolleg Greifswald* under the auspices of the *Deutsche Vereinigung für Mathematische Logik und für Grundlagen der Exakten Wissenschaften (DVMLG)*, the *Polskie Towarzystwo Logiki i Filozofii Nauki (PTLiFN)*, the *Association for Computability in Europe (ACiE)* and the *European Association for Computer Science Logic (EACSL)*

Abstract Booklet

Funded by the *Alfried Krupp von Bohlen und Halbach-Stiftung, Essen* and the *Deutsche Forschungsgemeinschaft, Bonn*

Preface

This booklet contains the extended abstracts of talks presented at the workshop on *Logical Approaches to Barriers in Computing and Complexity,* held on February 17–20, 2010, in Greifswald, Germany. The workshop was initiated by the *Deutsche Vereinigung für Mathematische Logik und für Grundlagen der Exakten Wissenschaften (DVMLG)*, the *Polskie Towarzystwo Logiki i Filozofii Nauki (PTLiFN)*, the *Association for Computability in Europe (ACiE)* and the *European Association for Computer Science Logic (EACSL)* and organized under the auspices of these four learnèd societies. The workshop was funded by the *Alfried Krupp von Bohlen und Halbach-Stiftung, Essen* and the *Deutsche Forschungsgemeinschaft, Bonn*, and it was run by and held at the *Alfried Krupp Wissenschaftskolleg* in Greifswald, Germany.

Scientific Scope. Computability theory and complexity theory have their origins in logic, with connections to foundations of mathematics. The fundamental goal in these areas is to understand the limits of computability (that is analysing which problems can be solved on nowadays and future computers in principle) and effective computability (that is understanding the class of problems which can be solved quickly and with restricted resources.) Logic provides a multifarious toolbox of techniques to analyse questions like this, some of which promise to provide a deep insight in the structure of limit of computation.

The workshop had a focus on the following related aspects: logical descriptions of complexity (e.g., descriptive complexity, bounded arithmetic), complexity classes of abstract, algebraic and infinite structures, barriers in proving complexity results, and Kolmogorov complexity and randomness. Descriptive complexity and bounded arithmetic are two complementary approaches to describe computational complexity in logical terms. The former is focused on decision problems, while the latter is more concerned with search problems. Both environments render questions about complexity classes in a natural way, leading to important open problems in their areas (e.g. finding logics to capture certain complexity classes, or the separation problem for bounded arithmetic.) Another path to gain more understanding of complexity classes is to study them in more general settings, e.g. on general algebraic structures or for computational models with infinite resources. Questions arising in this area are how complexity classes are rendered in them, and what their relationship is. It is well known that any proof of $P \neq NP$ will have to overcome two barriers: relativization and natural proofs. To this has recently been added a third barrier, called *algebraic relativization* or *algebrization.* The idea is that, when we relativize some complexity class inclusion, we should give the simulating machine access not only to an oracle A, but also to a low-degree extension of A over a finite field or ring.

The workshop intended to address current questions in all these areas, and initiate communication and research between them. In particular, the aim was to inform researcher in these related but separated areas about current approaches and ideas, to initiate new approaches to current questions. Some of these aspects were particularly timely: recently, research in these areas became more intense. Part of this is the new conference series CiE (run by the Association for Computability in Europe) whose range of interests includes those of our workshop, creating an important focus on the emerging topics of the field. This workshop was intended as a research-oriented follow-up to the CiE conferences, allowing researchers ample time for discussions and joint work.

Programme. The Programme Committee had been designed by the four organising scientific organisations and consisted of thirteen members: Zofia Adamowicz (Warschau), Franz Baader (Dresden), Arnold Beckmann (*chair*, Swansea), Sam Buss (La Jolla, CA), Manfred Droste (Leipzig), Christine Gaßner (Greifswald), Peter Koepke (Bonn), Benedikt Löwe (Amsterdam), Janos Makowsky (Haifa), Elvira Mayordomo (Zaragoza), Damian Niwinski (Warschau), Wolfgang Thomas (Aachen), and Martin Ziegler (Darmstadt). The DVMLG was represented by its vice president Löwe and members of executive board Beckmann und Makowsky, the PTLiFN by the member of executive board Niwinski, CiE by the members of the executive board Beckmann, Löwe and Mayordomo, and the EACSL by its president Makowsky, vice president Niwinski and member of executive board Löwe.

The Programme Committee has chosen six scientists as *keynote speakers*: Alessandra Carbone who spoke on *Logical structures, cyclic graphs and genus of proofs,* Lance Fortnow on *Hardness of Instance Compression and Its Applications,* Erich Grädel on *Fixed point logics, games, and polynomial-time complexity,* Pascal Koiran on *Shallow circuits with high-powered inputs,* Leszek Kołodziejczyk on *The scheme*

of induction for sharply bounded arithmetic formulas, and Antonina Kolokolova on *Axiomatic approach to barriers in complexity.* The programme was enriched through an *Öffentliche Abendveranstaltung* (public evening lecture) given by Janos Makowsky on *Spektrum Problem von H. Scholz und G. Asser: Ein halbes Jahrhundert danach.* The workshop received 45 contributed submissions which were reviewed by the programme committee. The committee decided to accept 34 talks whose abstracts are printed in this booklet.

The Programme Committee also decided to add to the workshop a Special Session on the topic *Complexity in Arbitrary Structures* which was organised by Christine Gaßner and Martin Ziegler, see the separate introduction to the Special Session on page V.

Organization and Acknowledgements. The workshop was organized by Christine Gaßner (University of Greifswald). We are delighted to acknowledge and thank the Stiftung Alfried Krupp Kolleg Greifswald and the Deutsche Forschungsgemeinschaft for their essential financial support. The high scientic quality of the workshop was possible through the conscientious work of the Programme Committee, and the Special Session organizers. We thank Andrej Voronkov for his EasyChair system which facilitated the work of the Programme Committee and the editors considerably.

Swansea, Greifswald and Amsterdam, January 2010

Arnold Beckmann
Christine Gaßner
Benedikt Löwe

Special Session on Complexity in Arbitrary Structures

The classical notion of computability is closely related to the classical theory of recursive functions. This theory deals with functions on natural numbers, on words over a finite alphabet, or on some other countable sets. For functions on real numbers and other uncountable universes, several non-equivalent definitions of computability have been introduced. Here, real closed and algebraically closed fields are the starting point for the development of new theories of computing. Analytical or algebraic properties of the ring over the real numbers are used for defining notions of computability for real functions. One of the best known models is the Type-2 Turing machine where real numbers are approximated by rational numbers and the classical Turing machine is extended to infinite converging computations. Other settings consider real numbers as entities and one assumes that it is possible to execute exact arithmetic instructions and comparisons. Algebraic properties of rings and fields yield many interesting results in models of this kind, such as algebraic circuits and the real Turing machine (i.e. the BSS machine over the reals). More general models cover computation over arbitrary fields or other structures. Most complexity classes well-known from classical (i.e. discrete) computational complexity theory have an analogue in the theories to these computation models and play partially an analogous role in the corresponding theories.

A lot of motivation to investigate these derived classes primary comes from the classical theory as well as from practical requirements. One hope is that we gain a better understanding of the difficulty of the P-versus-NP question. This famous problem is open for Turing machines as well as for many algebraic models of computation, but has been answered (in fact both in the positive and in the negative) for several structures. Further extending the list of structures with, provably, $\mathrm{P} = \mathrm{NP}$ or $\mathrm{P} \neq \mathrm{NP}$ will provide new insights in the original millennium prize problem.

Another benefit is the development and study of fast algorithms for processing real numbers. In the classical theory as well as in the theory of computation over arbitrary structures, most proof techniques are of a logical nature. For more restricted models, topological and algebraic properties are of additional relevance. Practical real number computation can be separated into a variety of subfields with distinct objectives and paradigms; e.g. exact vs. approximate input, fixed-precision (floating point) vs. unbounded precision, relative vs. absolute error goals. This diversity, and the different underlying parameters and notions of complexity, is better reflected by several models of computation than by a single one.

The special session on this field has been organised in order to foster communication and collaboration between the various sub communities. In fact, the invited and contributed works cover the topic nicely, ranging from computability over complexity theory to numeric. The following abstracts are concerned with questions like these. Which approach is suitable for modelling specific problems? How do hierarchies of degrees of unsolvability relate in the analytical approaches and the algebraic ones? Which relationships of complexity classes agree in which models of computation? Do Toda's and Ladner's Theorems hold also for computational settings other than the classical Turing machine? What about fixed point problems and total search problems? What ways are available to extend the theory to recursive definability? What role do the various aspects of computing of continuous functions on real and complex numbers play in the theories to several models?

Christine Gaßner (Greifswald)
Martin Ziegler (Darmstadt)

Table of Contents

Section 1. Regular Talks

Lower bounds for the provability of Herbrand consistency in weak arithmetics

Zofia Adamowicz[1] and Konrad Zdanowski[1]

Institute of Mathematics, Polish Academy od Science
Śniadeckich 8, Warszawa
email: {zosiaa,kz}@impan.gov.pl

One of the main methods of showing that one set of axioms, say T, is strictly stronger than the other one, say $S \subseteq T$, is to show that $T \vdash \mathrm{Con}_S$. However, as it was proved by Wilkie and Paris in [WP87] this method does not work for bounded arithmetic theories if we use the usual Hilbert style provability predicate. Indeed, they proved that even the strong arithmetic $\mathrm{I}\Delta_0 + \exp$ does not prove the Hilbert style consistency of Robinson's arithmetic Q, that is $\mathrm{I}\Delta_0 + \exp$ does not prove that there is no Hilbert prove of $0 \neq 0$ from Q. Thus, if we hope to prove that one bounded arithmetic is stronger than the other one by using consistency statements we should use some other provability notions, like tableux or Herbrand provability. Indeed, for these notions it is usually easier to show that a given theory is consistent since, e.g., Herbrand proofs are of a bigger size than Hilbert ones. Thus, it may happen in a model of $\mathrm{I}\Delta_0 + \exp$ that a theory S is inconsistent in the Hilbert sense and consistent in the Herbrand sense. Only when we know that the superexponentiation function is total we can prove the equivalence of the above notions of provability. For some time it has been even unknown whether the second Gödel incompleteness theorem holds for arithmetics $\mathrm{I}\Delta_0 + \Omega_i$ and the Herbrand style provability predicate. Adamowicz and Zbierski in [AZ01] proved, for $i \geq 2$, the second incompleteness theorem for $\mathrm{I}\Delta_0 + \Omega_i$ and the Herbrand notion of consistency and later Adamowicz in [A01] proved this result for $\mathrm{I}\Delta_0 + \Omega_1$. Recently, Kołodziejczyk showed in [K06] a strengthening of these results. He proved that there is a finite fragment S of $\mathrm{I}\Delta_0 + \Omega_1$ such that no theory $\mathrm{I}\Delta_0 + \Omega_i$ proves the Herbrand consistency of S. Thus, if one wants to prove strict hierarchy of bounded arithmetics by means of provability of Herbrand consistency one should consider a thinner notion, e.g., Herbrand proofs restricted to some cuts of a given model of a bounded arithmetic. Such a study is a main subject of our paper.

For a detailed treatment of bounded arithmetics we refer to [HP93]. We consider bounded arithmetics theories $\mathrm{I}\Delta_0 + \Omega_i$, for $i \geq 1$. $\mathrm{I}\Delta_0$ is just the first order arithmetic with the induction axioms restricted to bounded formulas i.e. formulas with quantification of the form $Qx \leq t(\bar{z})$, where $Q \in \{\exists, \forall\}$ and $x \notin \{\bar{z}\}$. For $i \geq 1$, the axiom Ω_i states the totality of the function ω_i. The functions ω_i are defined as follows. Let $\log(x)$ be the logarithm with the base 2 . Let the length function $\mathrm{lh}(x)$ be the length of the binary representation of x,

$$\mathrm{lh}(x) = \ulcorner \log(x+1) \urcorner.$$

Now,

$$\omega_1(x) = \begin{cases} 0 & \text{if } x = 0 \\ 2^{(\mathrm{lh}(x)-1)^2} & \text{if } x > 0. \end{cases}$$

and

$$\omega_{i+1}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 2^{\omega_i(\mathrm{lh}(x)-1)} & \text{if } x > 0. \end{cases}$$

By $\exp(x)$ we denote the exponentiation function 2^x.

Let $\log^n$ be a set of elements a such the n-th iteration of exp on a exists. If exp is not provably total in T then there are models of T in which not all elements are in $\log^n$. For $C > 0$, $C\log^n$ is a set of elements a such that there exists b in $\log^n$ such that a is less or equal than Cb. Let us observe that the above notions are definable by existential formulas.

For a term t we define its tree depth by an inductive condition as

$$\mathrm{tr}(f(t_1, \ldots, t_k)) = 1 + \max\{\mathrm{tr}(t_i) : i \leq k\}.$$

By the depth of a term t we define the maximum of its tree depth and the size of the greatest function symbol in t. That is

$$\mathrm{dp}(t) = \max\{f : \ f \text{ occurs in } t\} \cup \{\mathrm{tr}(t)\}.$$

For a set of terms Λ, the depth of Λ, $\mathrm{dp}(\Lambda) = \max\{\mathrm{dp}(t) : t \in \Lambda\}$.

An evaluation p on a set Λ is a function from Λ^2 into $\{0,1\}$. We define the following notion of satisfaction for evaluations by induction on the formula φ:

- $p \models t = t'$ if $p(t,t') = 1$,
- $p \models t \leq t'$ if there is $s \in \Lambda$ such that $p \models (t + s = t')$,
- for φ quantifier free, $p \models \varphi[\bar{t}]$ if p makes φ true in the sense of propositional logic,
- $p \models \exists x \varphi(\bar{x}, x)[\bar{t}]$ if $p \models \varphi(\bar{x}, x)[\bar{t}, s^{\exists\varphi}(\bar{t})]$,
- $p \models \forall x \varphi(\bar{x}, x)[\bar{t}]$ if for all terms $t \in \Lambda$, $p \models \varphi(\bar{x}, x)[\bar{t}, t]$.

For a set of axioms T an evaluation p is a T-evaluation if it satisfies all axioms from T. We say that an evaluation p on Λ decides a formula $\varphi(\bar{x})$, if for any $\bar{t} \in \Lambda$, either $p \models \varphi(\bar{t})$ or $p \models \neg\varphi(\bar{t})$. Let N be a standard integer. An evaluation is N-deciding if it decides any formula of a code less than N.

We formalize the notion of Herbrand consistency as a Π_1 arithmetical formula $\mathrm{HCons}(N, T, i)$ which states that that for each set of terms Λ of depth not greater than i, there exists an N–deciding, T–evaluation on Λ.

Let $i \geq 1$. We show that for each N there exists $\varepsilon > 0$ such that, $\mathrm{I}\Delta_0 + \Omega_i$ does not prove its N-deciding Herbrand consistency restricted to the terms of depth in $(1+\varepsilon)\log^{i+2}$, that is

$$\mathrm{I}\Delta_0 + \Omega_i \nvdash \mathrm{HCons}(N, \mathrm{I}\Delta_0 + \Omega_i, (1+\varepsilon)\log^{i+2}). \tag{1}$$

On the other hand it is known that for each N,

$$\mathrm{I}\Delta_0 + \Omega_i \vdash \mathrm{HCons}(N, \mathrm{I}\Delta_0 + \Omega_i, \log^{i+3}) \tag{2}$$

that is $\mathrm{I}\Delta_0 + \Omega_i$ proves its Herbrand consistency restricted to terms of depth $\log^{i+3}$.

It is tempting to close the gap by proving, at least for some $i \geq 1$, either for each N,

$$\mathrm{I}\Delta_0 + \Omega_i \vdash \mathrm{HCons}(N, \mathrm{I}\Delta_0 + \Omega_i, \log^{i+2}) \tag{3}$$

or

$$\mathrm{I}\Delta_0 + \Omega_i \nvdash \mathrm{HCons}(N, \mathrm{I}\Delta_0 + \Omega_i, A\log^{i+3}), \text{ for some } N, A \in \mathbb{N}. \tag{4}$$

Indeed both conjectures (3) and (4) have interesting consequences for bounded arithmetics. If (3) holds then $\mathrm{I}\Delta_0 + \Omega_{i+1}$ would not be Π_1–conservative over $\mathrm{I}\Delta_0 + \Omega_i$. This is so because $\log^{i+2}$ is closed under addition in the presence of Ω_{i+1}. Thus, in $\mathrm{I}\Delta_0 + \Omega_{i+1}$ the cuts $\log^{i+2}$ and $(1+\varepsilon)\log^{i+2}$ are the same. It follows then from (3) that $\mathrm{I}\Delta_0 + \Omega_{i+1} \vdash \mathrm{HCons}(\mathrm{I}\Delta_0 + \Omega_i, A\log^{i+2})$, for each $A \in \mathbb{N}$.

On the other hand, if (4) holds this would mean that we cannot mimic the proof of (2) for the cut $A\log^{i+3}$. But the only tool needed in that proof which is unavailable in this situation is the existence of a suitable truth definition for Δ_0 formulas. It would follow that there is no such truth definition for Δ_0 formulas whose suitable properties are provable in $\mathrm{I}\Delta_0 + \Omega_i$. This is related to a major open problem in bounded arithmetics how much exponentiation is needed for a truth definition for bounded formulas.

References

[A01] Z. Adamowicz, *On tableaux consistency in weak theories*, preprint 618, Institute of Mathematics of the Polish Academy of Sciences, 2001.

[A02] Z. Adamowicz, *Herbrand consistency and bounded arithmetic*, in Fundamenta Mathematicae 171(2002), pp. 279–292

[AZ01] Z. Adamowicz and P. Zbierski, *On Herbrand consistency in weak arithmetics*, in Archive for Mathematical Logic, 40(2001), pp. 399–413.

[HP93] P. Hájek and P. Pudlák, *Metamathematics of first–order arithmetic*, Springer–Verlag, 1993.

[K06] L. A. Kołodziejczyk, *On the Herbrand notion of consistency for finitely axiomatizable fragments of bounded arithmetic theories*, in Journal of Symbolic Logic 71(2006), pp. 624–638.

[WP87] A. J. Wilkie and J. B. Paris, *On the scheme of induction for bounded arithmetical formulas*, in Annals of Pure and Applied Logic, 35(1987), pp. 261–302.

A characterization of one-way functions based on time-bounded Komogorov complexity

Luís Antunes * André Souto ** Andreia Teixeira ***

{lfa,andresouto,andreiasofia}@dcc.fc.up.pt

Universidade do Porto
Instituto de Telecomunicações

Address:
Rua do Campo Alegre, nº 1021/1055
4169-007 Porto Portugal

The security of most cryptographic schemes is based implicitly on the security of the cryptographic primitives used, like one-way functions, i.e., functions that are "easy" to compute in polynomial time but are "hard" to invert in probabilistic polynomial time. In fact, the vast majority of the usual primitives implies the existence of one-way functions. The existence of these functions is a strong assumption as it is well known that it implies that $\mathbf{P} \neq \mathbf{NP}$, although it is a very important open question to know if this is also a sufficient condition. To emphasize the importance of the existence of one-way functions we observe that if they did not exist then pseudo-random generators, digital signatures, identification schemes and private-key encryption would not be possible, [BM84,GMR88,IL89,ILL89,Rom90].

Given the impact of one-way functions in cryptography and complexity, we believe that it is important to study these functions at the individual level in opposition to its average case behavior. In this work we give a first characterization of one-way functions based on time-bounded Kolmogorov complexity. We hope that this characterization may lead to a finer grained analysis to some cryptographic protocols, such as commitment schemes.

Classically there are two types of one-way functions: strong and weak one-way functions. In the case of a strong one-way function, it is required that the inversion happens with low probability and in the weak version the non inversion must happen with non-negligible probability. An intersecting fact about these functions is that not every weak one-way function is a strong one-way function but their existence is equivalent (see [Gol01] for details). Formally:

Definition 1 (Weak one-way function). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is a weak one-way function if it is computable in polynomial time, it is total, one-to-one, honest, and there is a polynomial p such that for every probabilistic polynomial time algorithm G and all sufficiently large n's,*

$$Pr[G(f(x)) \neq x] > \frac{1}{p(n)}.$$

Definition 2 (Strong one-way function). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is a strong one-way function if it is computable in polynomial time, it is total, one-to-one, honest, and for every probabilistic polynomial time algorithm G, every positive polynomial p, and all sufficiently large n's,*

$$Pr[G(f(x)) = x] < \frac{1}{p(n)}.$$

To give a characterization of one-way functions using individual instances we will use Kolmogorov complexity. This notion, defined independently by Kolmogorov [Kol65], Solomonoff [Sol64] and Chaitin [Cha66], is a rigorous measure of information contained in a string by the size of the smallest program producing that string.

Definition 3. *Let U be a fixed universal Turing machine with a prefix-free domain. For any strings $x, y \in \{0,1\}^*$, the Kolmogorov complexity of x given y is*

$$K(x|y) = \min_p\{|p| : U(p,y) = x\}.$$

* All the authors are partially supported by CSI^2 (PTDC/EIA- CCO/099951/2008)
** The author is also supported by the grant SFRH / BD / 28419 / 2006 from FCT
*** The author is also supported by the grant SFRH / BD / 33234 / 2007 from FCT

For any time constructible t, the t-time-bounded Kolmogorov complexity of x given y is defined by

$$K^t(x|y) = \min_p\{|p| : U(p,y) = x \text{ in at most } t(|x|) \text{ steps}\}.$$

The higher the Kolmogorov complexity the more information is necessary to recover that string. Almost all strings have nearly maximum Kolmogorov complexity, however one can easily create, with high probability, another string just as useful as the first one by using fresh random coins.

We plan to explore the idea that Kolmogorov complexity can be useful within the field of cryptography, by working at the level of the particular instances instead of the probability distributions involved. Notice that, for example in the case of zero knowledge communication protocols and in the case of commitment schemes based on one-way functions, this type of analysis gives an insight of the information that is leaked on particular communications. In this work we begin this task by analyzing one-way functions. In particular, we characterize one-way functions based on individual instances and relate it with the classical notions of one-way functions. We start by studying the expected value of Kolmogorov complexity of x given $f(x)$ and some randomness and then a characterization based on Kolmogorov complexity of the individual instances is given. In the former approach we show that to have a weak one-way function the expectation should be at least larger than any constant but if f is a strong one-way function then this value is nearly maximum which gives a huge gap between weak one-way functions and strong one-way functions. Formally we prove the following two results:

Theorem 1. *Let t be a fixed polynomial and f a polynomial time computable function. If f is an honest one-to-one function such that for all constant c the expected value of $K^t(x|f(x),f,r)$ over strings of length n is larger than c for almost all n then f is a weak one-way function.*

Theorem 2. *Let t be a fixed polynomial and f a polynomial time computable function. If f is an honest one-to-one function such that the expected value of $K^t(x|f(x),f,r)$ over strings of length n is larger than $n + O(\log n)$ for almost all n then f is a strong one-way function.*

For the later approach the intuition is that the time-bounded Kolmogorov complexity is suitable for defining one-way functions using individual instances. In this type of primitive we expect that, given x and a description of f, computing $f(x)$ will be efficiently easy, i.e., x and f give all the information to compute in polynomial time $f(x)$ and on the other hand, $f(x)$, in polynomial time, does not convey useful information about x, so the length of the program computing x given $f(x)$ and f would be approximately equal to the length of the program computing x without any auxiliary input. With this intuition in mind we define Kolmogorov one-way function in the following way:

Definition 4. *Let t be a fixed polynomial, $f : \Sigma^* \to \Sigma^*$ an honest, one-to-one function computable in polynomial time and δ a fixed constant. We say that an instance x of length n is δ-secure relatively to the random string $r \in \Sigma^{\leq t(n)}$ if:*

$$K^t(x|f,r) - K^t(x|f,f(x),r) \leq \delta.$$

We say that f is an (ε,δ)-Kolmogorov one-way function if we have:

$$Pr_{x,r}(x \text{ is } \delta\text{-secure for } r) \geq \varepsilon$$

for sufficiently large n and $r \in \Sigma^{\leq t(n)}$.

In this work we show that Kolmogorov one-way functions are more restrictive than classical one-way functions in the sense that the existence of Kolmogorov one-way functions with certain parameters implies the existence of classical one-way functions. In particular we show:

Theorem 3. *Let t be a fixed polynomial and f a function computable in polynomial time. If f is an honest, one-to-one function that is $(1 - 1/p(n), c)$-Kolmogorov one-way function for all polynomial and constant c then f is a weak one-way function.*

Theorem 4. *Let t be a fixed polynomial and f a function computable in polynomial time. If f is an honest, one-to-one function that is $(1 - 1/2^n, c)$-secure Kolmogorov one-way function for all polynomial and constant c then f is a classical one-way function.*

Notice that if the previous results are optimal then in particular we can say that a Kolmogorov one-way function is a function that the probability of inversion would not be significantly larger than guessing the inverse of each element.

In [LM93] and [LW95], the authors relate the existence of one-way functions and the conjecture of polynomial time symmetry of information. For the unbounded version of Kolmogorov complexity, symmetry of information was first proved to be true by Levin (as suggested in [ZL70]), but the proof is not valid any more when polynomial time-bounds restrictions are imposed. This conjecture has close connections to several complexity theoretic questions, similar to the ones concerning the existence of one-way functions. In order to have a full scenario about the existence of Kolmogorov one-way functions, in this work, we also study its connection to the polynomial time symmetry of information conjecture by showing that if Kolmogorov one-way functions exist, then the polynomial time symmetry of information conjecture fails.

References

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13(4):547–569, 1966.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[Gol01] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *SFCS '89*, pages 230–235. IEEE Computer Society, 1989.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *STOC '89*, pages 12–24. ACM, 1989.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[LM93] L. Longpré and S. Mocas. Symmetry of information and one-way functions. *Information processing Letters*, 46(2):95–100, 1993.

[LW95] L. Longpré and O. Watanabe. On symmetry of information and polynomial time invertibility. *Information and Computation*, 121(1):14–22, 1995.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC '90*, pages 387–394. ACM, 1990.

[Sol64] R. Solomonoff. A formal theory of inductive inference, part i. *Information and Control*, 7(1):1–22, 1964.

[ZL70] A. Zvonkin and L. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematics Surveys*, 256:83–124, 1970.

A Case Study in Graph Polynomials: The Subgraph Component Polynomial

I. Averbouch[(1), *], J.A. Makowsky[(1), **], and P. Tittmann[(2)]

[(1)]Department of Computer Science,
Technion–Israel Institute of Technology, 32000 Haifa, Israel
[(2)] Fachbereich Mathematik–Physik–Informatik,
Hochschule Mittweida, Mittweida, Germany

Abstract. Inspired by the study of community structure in social networks, we introduce the graph polynomial $Q(G; x, y)$, as a bivariate generating function which counts the number of connected components in induced subgraphs. In this case study we analyze the features of the new polynomial. First, we re-define it as a subset expansion formula. Second, we give a recursive definition of $Q(G; x, y)$ using vertex deletion, vertex contraction and deletion of a vertex together with its neighborhood, and prove a universality property. We relate $Q(G; x, y)$ to the universal edge elimination polynomial introduced by I. Averbouch, B. Godlin and J.A. Makowsky (2008), which subsumes other known graph invariants and graph polynomials, among them the Tutte polynomial, the independence and matching polynomials, and the bivariate extension of the chromatic polynomial introduced by K. Dohmen, A. Pönitz, and P. Tittmann (2003). Finally we show that the computation of $Q(G; x, y)$ is $\sharp\mathbf{P}$-hard, but Fixed Parameter Tractable for graphs of bounded tree-width and clique-width.

1 Introduction

There is a variety of graph polynomials studied in the literature. Those polynomials differ in many aspects: their origins come from different research areas, they are defined by different ways, and their authors use different terminology in their papers. In [12] the second author outlined a general framework for studying graph polynomials, further developed in [10]. This framework allows to compare graph polynomials with respect to their ability to distinguish graphs, to encode other graph polynomials or numeric graph invariants, and their computational complexity. In [17] we introduced a new graph polynomial, the *subgraph component polynomial* $Q(G; x, y)$, which arises naturally from studying community structures in social networks.

In this case study, we present the results from [17] according to the following checklist:

- Can the polynomial be presented as a subset expansion formula?
  What logic formalism is needed to define this formula?
- Does this polynomial satisfy some linear recurrence relation?
  Has it some universality property with respect to that recurrence relation?
- Is it definable as a partition function using counting of weighted graph homomorphisms?
- How hard is the polynomial to compute?
  Does it have a dichotomy property, cf. the *Difficult Point Conjecture* of [12])?
- What is its connection to known graph polynomials?
- And finally: is it really new?

2 Motivation: Community Structure in Networks

In the last decade stochastic social networks have been analyzed mathematically from various points of view. Understanding such networks sheds light on many questions arising in biology, epidemology, sociology and large computer networks. Researchers have concentrated particularly on a few properties

* Partially supported by a grant of the Graduate School of the Technion–Israel Institute of Technology
** Partially supported by a grant of the Fund for Promotion of Research of the Technion–Israel Institute of Technology and grant ISF 1392/07 of the Israel Science Foundation (2007-2010)

that seem to be common to many networks: the small-world property, power-law degree distributions, and network transitivity. For a broad view on the structure and dynamics of networks, see [15]. M. Girvan and M.E.J. Newman, [9], highlight another property that is found in many networks, the property of *community structure*, in which network nodes are joined together in tightly knit groups, between which there are only looser connections.

Motivated by [14], and the third author's involvement in a project studying social networks, we were led to study the graph parameter $q_{ij}(G)$, the number of vertex subsets $X \subseteq V$ with i vertices such that $G[X]$ has exactly j components. $q_{ij}(G)$, counts the number of degenerated communities which consist of i members, and which split into j isolated subcommunities.

The ordinary bivariate *generating function* associated with $q_{ij}(G)$ is the two-variable graph polynomial

$$Q(G;x,y) = \sum_{i=0}^{n}\sum_{j=0}^{n} q_{ij}(G)\, x^i y^j.$$

We call $Q(G;x,y)$ the *subgraph component polynomial* of G. The coefficient of y^k in $Q(G;x,y)$ is the ordinary generating function for the number of vertex sets that induce a subgraph of G with exactly k components.

3 $Q(G;x,y)$ as a Graph Polynomial

In this paper we study the *s*ubgraph component polynomial $Q(G;x,y)$ as a graph polynomial in its own right and explore its properties along the checklist from Section 1.

Like the bivariate Tutte polynomial, see [6, Chapter 10], the polynomial $Q(G;x,y)$ has several remarkable properties. However, its distinguishing power is quite different from the Tutte polynomial and other well studied polynomials.

Our main findings are [1]:

- The Tutte polynomial satisfies a linear recurrence relation with respect to edge deletion and edge contraction, and is universal in this respect. $Q(G;x,y)$ also satisfies a linear recurrence relation, but with respect to three kinds of vertex elimination, and is universal in this respect.
- A graph polynomial in three indeterminates, $\xi(G;x,y,z)$, which satisfies a linear recurrence relation with respect to three kinds of edge elimation, and which is universal in this respect, was introduced in [2]. It subsumes both the Tutte polynomial and the matching polynomial. For line graph $L(G)$ of a graph G, we have $Q(L(G);x,y)$ is a substitution instance of $\xi(G;x,y,z)$.
- Distinguishing power of $Q(G;x,y)$ is incomparable with that of the Tutte polynomial, the characteristic polynomial and the bivariate chromatic polynomial introduced in [7].
- Also like for the Tutte polynomial, cf. [11], $Q(G;x_0,y_0)$ has the *Difficult Point Property*, i.e. it is $\sharp\mathbf{P}$-hard to compute for all fixed values of $(x_0,y_0) \in \mathbb{R}^2 - E$ where E is a semi-algebraic set of lower dimension. In [12] it is conjectured that the Difficult Point Property holds for a wide class of graph polynomials, the graph polynomials definable in Monadic Second Order Logic. The conjecture has been verified for various special cases, [3–5].
- $Q(G;x,y)$ is fixed parameter tractable in the sense of [8] when restricted to graphs classes of bounded tree-width or even to classes of bounded clique-width. For the Tutte polynomial, this is known only for graph classes of bounded tree-width, [16, 1, 13].

References

1. A. Andrzejak. Splitting formulas for Tutte polynomials. *Journal of Combinatorial Theory, Series B*, 70.2:346–366, 1997.
2. I. Averbouch, B. Godlin, and J.A. Makowsky. An extension of the bivariate chromatic polynomial. *European Journal of Combinatorics*, 31.1:1–17, 2010.
3. M. Bläser and H. Dell. Complexity of the cover polynomial. In L. Arge, C. Cachin, T. Jurdziński, and A. Tarlecki, editors, *Automata, Languages and Programming, ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*, pages 801–812. Springer, 2007.

[1] More results regarding $Q(G;x,y)$ are available in preprint [17].

4. M. Bläser, H. Dell, and J.A. Makowsky. Complexity of the Bollobás-Riordan polynomia. exceptional points and uniform reductions. In Edward A. Hirsch, Alexander A. Razborov, Alexei Semenov, and Anatol Slissenko, editors, *Computer Science–Theory and Applications, Third International Computer Science Symposium in Russia*, volume 5010 of *Lecture Notes in Computer Science*, pages 86–98. Springer, 2008.
5. Markus Bläser and Christian Hoffmann. On the complexity of the interlace polynomial. In *STACS*, volume 08001 of *Dagstuhl Seminar Proceedings*, pages 97–108. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.
6. B. Bollobás. *Modern Graph Theory*. Springer, 1999.
7. K. Dohmen, A. Pönitz, and P. Tittmann. A new two-variable generalization of the chromatic polynomial. *Discrete Mathematics and Theoretical Computer Science*, 6:69–90, 2003.
8. R.G. Downey and M.F Fellows. *Parametrized Complexity*. Springer, 1999.
9. M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.
10. B. Godlin, E. Katz, and J.A. Makowsky. Graph polynomials: From recursive definitions to subset expansion formulas. arXiv `http://uk.arxiv.org/pdf/0812.1364.pdf`, to appear in the Journal of Logic and Computation, 2008.
11. F. Jaeger, D.L. Vertigan, and D.J.A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Camb. Phil. Soc.*, 108:35–53, 1990.
12. J.A. Makowsky. From a zoo to a zoology: Towards a general theory of graph polynomials. *Theory of Computing Systems*, 43:542–562, 2008.
13. J.A. Makowsky, U. Rotics, I. Averbouch, and B. Godlin. Computing graph polynomials on graphs of bounded clique-width. In F. V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-23, 2006, Revised Papers*, volume 4271 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 2006.
14. M.E.J. Newman. Detecting community structure in networks. *Eur. Phys. J.B.*, 38:321–330, 2004.
15. M.E.J. Newman, A.L. Barabasi, and D. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.
16. S.D. Noble. Evaluating the Tutte polynomial for graphs of bounded tree-width. *Combinatorics, Probability and Computing*, 7:307–321, 1998.
17. P. Tittmann, I. Averbouch, and J.A. Makowsky. The enumeration of vertex induced subgraphs with respect to the number of components. *European Journal of Combinatorics*, 3x.x:xx–yy, 2010.

Proof Complexity of Propositional Default Logic*

Olaf Beyersdorff[1], Arne Meier[2], Sebastian Müller[3], Michael Thomas[2], and Heribert Vollmer[2]

[1] Institute of Computer Science, Humboldt University Berlin, Germany beyersdo@informatik.hu-berlin.de
[2] Institute of Theoretical Computer Science, Leibniz University Hanover, Germany
{meier,thomas,vollmer}@thi.uni-hannover.de
[3] Faculty of Mathematics and Physics, Charles University Prague, Czech Republic
smueller@informatik.hu-berlin.de

Abstract. Default logic is one of the most popular and successful formalisms for non-monotonic reasoning. In 2002, Bonatti and Olivetti introduced several sequent calculi for credulous and skeptical reasoning in propositional default logic. In this paper we examine these calculi from a proof-complexity perspective. In particular, we show that the calculus for credulous reasoning obeys almost the same bounds on the proof size as Gentzen's system LK. Hence proving lower bounds for credulous reasoning will be as hard as proving lower bounds for LK. On the other hand we show an exponential lower bound to the proof size in Bonatti and Olivetti's enhanced calculus for skeptical default reasoning.

1 Barriers in Computing: Lower Bounds to Lengths of Proofs

Proving lower bounds for propositional proof systems constitutes one of the major objectives in propositional proof complexity. By a classical result of Cook and Reckhow [9], showing super-polynomial lower bounds to the lengths of proofs in increasingly stronger propositional proof systems is one approach toward separating NP from coNP, and consequently also P from NP. Frege systems currently form a strong barrier with respect to lower bounds to proof lengths. Though exponential lower bounds have been shown for bounded-depth Frege systems [1, 3, 4, 18], no non-trivial lower bound is known for the general system.

While there is a rich body of results for propositional proof systems (cf. [17]), proof complexity of non-classical logics has only recently attracted more attention, and a number of exciting results have been obtained for modal and intuitionistic logics [14–16]. In particular, Hrubeš [14] has shown exponential lower bounds for the proof size for Frege systems in many modal and intuitionistic logics. As non-classical logics are often more expressive than propositional logic, they are usually associated with large complexity classes like PSPACE. Intuitively therefore, lower bounds to the lengths of proofs in non-classical logic should be easier to obtain, as they "only" target at separating NP and PSPACE. The results of Hrubeš [14] and Jeřábek [15, 16] on non-classical Frege systems are very interesting to contrast with our knowledge on classical Frege as they shed new light on this topic from a different perspective.

In this paper, we investigate the proof complexity of propositional default logic. In particular, we analyse the sequent calculi of Bonatti and Olivetti [5] for propositional default logic. We show that the sequent calculus for credulous default reasoning has almost the same upper and lower bounds on lengths of proofs as classical Frege systems. Thus, the current barrier in classical proof complexity admits a natural restatement in terms of non-monotonic logic.

2 Default Logic

Trying to understand the nature of human reasoning has been one of the most fascinating adventures since ancient times. It has long been argued that due to its monotonicity, classical logic is not adequate to express the flexibility of commonsense reasoning. To overcome this deficiency, a number of formalisms have been introduced (cf. [21]), of which Reiter's default logic [22] is one of the most popular and widely used systems. Default logic extends the usual logical (first-order or propositional) derivations by patterns for default assumptions. These are of the form "in the absence of contrary information, assume ...".

* Supported in part by DFG grants KO 1053/5-2 and VO 630/6-1 and by a grant from the John Templeton Foundation.

Reiter argued that his logic adequately formalizes human reasoning under the *closed world assumption*. Today default logic is widely used in artificial intelligence and computational logic.

The semantics and the complexity of default logic have been intensively studied during the last decades (cf. [8] for a survey). In particular, Gottlob [13] has identified and studied two reasoning tasks for propositional default logic: the *credulous* and the *skeptical* reasoning problem which can be understood as analogues of the classical problems SAT and TAUT. Due to the stronger expressibility of default logic, however, credulous and skeptical reasoning become harder than their classical counterparts—they are complete for the second level Σ_2^{p} and Π_2^{p} of the polynomial hierarchy, see [13].

Less is known about the complexity of proofs in default logic. Starting with Reiter's work [22], several proof-theoretic methods have been developed for default logic (cf. [2,12,19,20,23] and [10] for a survey). However, most of these formalisms employ external constraints to model non-monotonic deduction and thus cannot be considered purely axiomatic (cf. [11] for an argument). This was achieved by Bonatti and Olivetti [5] who designed simple and elegant sequent calculi for credulous and skeptical default reasoning. Subsequently, Egly and Tompits [11] extended Bonatti and Olivetti's calculi to first-order default logic and showed a speed-up of these calculi over classical first-order logic, i. e., they construct sequences of first-order formulae which need long classical proofs but have short derivations using default rules.

3 Our Results

In the present paper we investigate the original calculi of Bonatti and Olivetti [5] from a proof-complexity perspective. Apart from some preliminary observations in [5], this comprises, to our knowledge, the first comprehensive study of lengths of proofs in propositional default logic. Our results can be summarized as follows. Bonatti and Olivetti's *credulous default calculus* BO_{cred} obeys almost the same bounds to the proof size as Gentzen's propositional sequent calculus LK, i. e., we show that upper bounds to the proof size in both calculi are polynomially related. The same result also holds for the proof length (the number of steps in the system). Thus, proving lower bounds to the size of BO_{cred} will be as hard as proving lower bounds to LK (or, equivalently, to Frege systems), which constitutes a major challenge in propositional proof complexity [6, 17]. This result also has implications for automated theorem proving. Namely, we transfer the non-automatizability result of Bonet, Pitassi, and Raz [7] for Frege systems to default logic: BO_{cred}-proofs cannot be efficiently generated, unless factoring integers is possible in polynomial time.

While already BO_{cred} appears to be a strong proof system for credulous default reasoning, admitting very concise proofs, we also exhibit a general method of how to construct a proof system $Cred(P)$ for credulous reasoning from a propositional proof system P. This system $Cred(P)$ bears the same relation to P with respect to proof size as BO_{cred} does to LK. Thus, choosing for example P as extended Frege might lead to stronger proof systems for credulous reasoning.

For *skeptical reasoning*, the situation is different. Bonatti and Olivetti [5] construct two proof systems for this task. While they already show an exponential lower bound for their first skeptical calculus, we obtain also an exponential lower bound to the proof length in their enhanced skeptical calculus. Thus, searching for natural proof systems for skeptical default reasoning with more concise proofs will be a rewarding task for future research.

Acknowledgement

The first author wishes to thank Neil Thapen for interesting discussions on the topic of this paper during a research visit to Prague.

References

1. M. Ajtai. The complexity of the pigeonhole-principle. *Combinatorica*, 14(4):417–433, 1994.
2. G. Amati, L. C. Aiello, D. M. Gabbay, and F. Pirri. A proof theoretical approach to default reasoning I: Tableaux for default logic. *Journal of Logic and Computation*, 6(2):205–231, 1996.
3. P. W. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák, and A. Woods. Exponential lower bounds for the pigeonhole principle. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 200–220, 1992.
4. P. W. Beame, T. Pitassi, and R. Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.

5. P. A. Bonatti and N. Olivetti. Sequent calculi for propositional nonmonotonic logics. *ACM Transactions on Computational Logic*, 3(2):226–278, 2002.
6. M. L. Bonet, S. R. Buss, and T. Pitassi. Are there hard examples for Frege systems? In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 30–56. Birkhäuser, 1995.
7. M. L. Bonet, T. Pitassi, and R. Raz. On interpolation and automatization for Frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.
8. M. Cadoli and M. Schaerf. A survey of complexity results for nonmonotonic logics. *Journal of Logic Programming*, 17(2/3&4):127–160, 1993.
9. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.
10. J. Dix, U. Furbach, and I. Niemelä. Nonmonotonic reasoning: Towards efficient calculi and implementations. In *Handbook of Automated Reasoning*, pages 1241–1354. Elsevier and MIT Press, 2001.
11. U. Egly and H. Tompits. Proof-complexity results for nonmonotonic reasoning. *ACM Transactions on Computational Logic*, 2(3):340–387, 2001.
12. D. Gabbay. Theoretical foundations of non-monotonic reasoning in expert systems. In *Logics and Models of Concurrent Systems*, pages 439–457. Springer-Verlag, Berlin Heidelberg, 1985.
13. G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, 1992.
14. P. Hrubeš. On lengths of proofs in non-classical logics. *Annals of Pure and Applied Logic*, 157(2–3):194–205, 2009.
15. E. Jeřábek. Frege systems for extensible modal logics. *Annals of Pure and Applied Logic*, 142:366–379, 2006.
16. E. Jeřábek. Substitution Frege and extended Frege proof systems in non-classical logics. *Annals of Pure and Applied Logic*, 159(1–2):1–48, 2009.
17. J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.
18. J. Krajíček, P. Pudlák, and A. Woods. Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1):15–39, 1995.
19. S. Kraus, D. J. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1–2):167–207, 1990.
20. D. Makinson. General theory of cumulative inference. In *Proc. 2nd International Workshop on Non-Monotonic Reasoning*, pages 1–18. Springer-Verlag, Berlin Heidelberg, 1989.
21. V. W. Marek and M. Truszczyński. *Nonmonotonic Logics—Context-Dependent Reasoning*. Springer-Verlag, Berlin Heidelberg, 1993.
22. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
23. V. Risch and C. Schwind. Tableaux-based characterization and theorem proving for default logic. *Journal of Automated Reasoning*, 13(2):223–242, 1994.

Tractable constructions of finite automata from monadic second-order formulas

Bruno Courcelle, Irène Durand

Université Bordeaux-1, LaBRI, CNRS
351, Cours de la Libération
33405, Talence cedex, France
courcell@labri.fr ; idurand@labri.fr

It is well-known from [DF, FG, CMR] that the model-checking problem for MSO logic on graphs is fixed-parameter tractable (FPT) with respect to tree-width and clique-width. The proof uses tree-decompositions (for tree-width as parameter) and k-expressions (for clique-width as parameter; see below), and the construction of a finite tree-automaton from an MSO sentence, expressing the property to check.

These two points are difficult : although tree-width $\leq k$ can be checked in linear time, the corresponding algorithm (by Bodlaender, see [DF]) is not practically usable. The situation is similar for bounded clique-width (see [HliOum]). Graphs can be given *with* their decompositions witnessing tree-width or clique-width $\leq k$, but another difficulty arises : the automata to be constructed are much too large and computations abort by lack of memory space. This is actually unavoidable if one wants an algorithm taking as input any MSO sentence (see, e.g., [FriGro]). One possibility is to forget the idea of implementing the general theorem, and to work directly on particular problems: see [G1,G2] or [GH]. Another one, explored here (also in [KL] in a different way) consists in finding fragments of MSO logic having an interesting expressive power, and for which automata constructions (or other constructions) are tractable.

What we propose is based on the following ideas :
(1) Do not alternate quantifiers and do not determinize automata.
(2) Write MSO formulas with Boolean set terms (see below definitions).
(3) Precompile basic graph properties into "small" finite automata.

We do not capture all MSO graph properties, but we can formalize in this way coloring and partitionning problems, and also some domination problems to take a few examples. We only discuss graphs of bounded clique-width, but the ideas extend to graphs of bounded tree-width and MSO formulas with edge set quantifications. The problems considered successfully in [BK] use automata with smaller numbers of states than what we need.

Definition 1 : Clique-width and k-expressions.
Graphs are finite, simple, directed, loop-free. Each vertex has a label in $[k] := \{1, ..., k\}$. The operations on graphs are $\oplus$, the union of disjoint graphs, the unary edge-addition $\overrightarrow{add}_{a,b}$ that adds the missing edges from every vertex labelled a to every vertex labelled b, the relabelling $relab_{a\to b}$ that changes a to b (with $a \neq b$ in both cases). The constant a denotes one vertex (with no edge) labelled by $a \in [k]$.Let F_k be the set of these operations and constants. Every term t in $T(F_k)$ called a *k-expression* defines a graph $G(t)$ with vertex set equal to the set of occurrences of constants in t. A graph has *clique-width* at most k if it is defined by some t in $T(F_k)$.

As a logical structure, a graph is defined as $\langle V_G, edg_G\rangle$ where V_G is the vertex set and edg_G the binary relation that describes edges.

Definition 2 : **The set of terms representing a graph property.**
Let $P(X_1, ..., X_n)$ be a property of sets of vertices $X_1, ..., X_n$ of a graph G denoted by a term t in $T(F_k)$. Examples are : $E(X, Y)$: there is an edge from some x in X to some y in Y; $H(X, Y)$: for every x in X, there is an edge from some y in Y to x ; $Path(X, Y)$: X has two vertices linked by a path in $G[Y]$, the subgraph of G induced by Y and $Conn(X)$: $G[X]$ is connected.

Let $F_k^{(n)}$ be obtained from F_k by replacing each constant a by the constants (a, w) where $w \in \{0, 1\}^n$. For fixed k, let $L_{P,(X_1,...,X_n),k}$ be the set of terms t in $T(F_k^{(n)})$ such that $P(A_1, ..., A_n)$ is true in $G(t)$, where A_i is the set of vertices which are occurrences of constants (a, w) where the i-th component of w

is 1. Hence t in $T(F_k^{(n)})$ defines a graph $G(t)$ and an assignment of sets of vertices to the set variables $X_1, ..., X_n$.

Definition 3 : $\exists MSO(\mathcal{P})$ **sentences**

We let $\mathcal{P}$ be a set of basic graph properties like those of Definition 2 together with the atomic formulas $X_1 \subseteq X_2$, $X_1 = \emptyset$, $Sgl(X_1)$ (the last one means that X_1 denotes a singleton set). Let $\{X_1, ..., X_n\}$ be a set of set variables. A *Boolean set term* is a term over these variables, $\cap, \cup$ and complementation (example below). A $\mathcal{P}$ *-atomic formula* is a formula of the form $P(S_1, ..., S_m)$ where $S_1, ..., S_m$ are Boolean set terms and P belongs to $\mathcal{P}$. An $\exists MSO(\mathcal{P})$ sentence is a sentence of the form $\exists X_1, ..., X_n.\varphi$ where φ is a positive Boolean combination of $\mathcal{P}$ -atomic formulas.

Examples 4 :

We give some examples of $\exists MSO(\mathcal{P})$ sentences.

(1) The property of *p-vertex colorability* can be expressed as follows :

$$\exists X_1, ..., X_p.(Part(X_1, ..., X_p) \wedge St(X_1) \wedge \cdots \wedge St(X_p)),$$

where $Part(X_1, ..., X_p)$ expresses that $X_1, ..., X_p$ define a partition of the vertex set and $St(X_i)$ expresses that X_i is *stable*, i.e., that the induced graph $G[X_i]$ has no edge.

A p-vertex coloring defined by $X_1, ..., X_p$ is *acyclic* if furthermore, each induced graph $G[X_i \cup X_j]$ is acyclic (is a forest). These properties are thus in $\exists MSO(\mathcal{P})$ if we let in $\mathcal{P}$ the properties $Part(X_1, ..., X_p)$, $St(X_1)$ and $NoCycle(X_1)$ expressing that $G[X_1]$ is acyclic.

(2) *Minor inclusion.* That a graph G contains a fixed simple loop-free graph H with vertex set $\{v_1, ..., v_p\}$ as a minor, can be expressed by the sentence μ :

$$\exists X_1, \cdots, X_p.(Disjoint(X_1, ..., X_p) \wedge Conn(X_1) \wedge \cdots \wedge Conn(X_p) \wedge \cdots Link(X_i, X_j) \wedge \cdots)$$

where $Disjoint(X_1, ..., X_p)$ expresses that $X_1, ..., X_p$ are pairwise disjoint, $Conn(X_i)$ expresses that $G[X_i]$ is connected and $Link(X_i, X_j)$ expresses that there exists an edge between a vertex of X_i and one of X_j; in μ, there is one formula $Link(X_i, X_j)$ for each edge $\{v_i, v_j\}$ of H.

(3) *Constrained domination.* The sentence $\exists X_1.(P(X_1) \wedge Dom(\overline{X_1}, X_1))$ expresses that there exists a set X_1 satisfying a property P and which also *dominates all other vertices.* The formula $Dom(Y, X)$ expresses that every vertex of Y is linked by an edge to some vertex of X.

(4) Many vertex partitionning problems considered by Rao in [Rao] can be expressed in this way.

Definition 5 : From $\exists MSO(\mathcal{P})$ **sentences to "reasonable sized" automata.**

Let us assume that for each basic property $P(X_1, ..., X_m)$ we have constructed a finite automaton $\mathcal{A}_{P,(X_1,...,\ X_m),k}$ that accepts the set $L_{P,(X_1,...,X_m),k}$.

Claim 6 : For set terms $S_1, ..., S_m$ over $\{X_1, ..., X_n\}$, the set of terms $L_{P(S_1,...,S_m),(X_1,...,X_n),k}$ is $h^{-1}(L_{P,(X_1,...,X_m),k})$ where h is the *alphabetic homomorphism* : $T(F_k^{(n)}) \rightarrow T(F_k^{(m)})$ that replaces a constant symbol (a, w) for $w \in \{0,1\}^n$ by (a, w') for some $w' \in \{0,1\}^m$ and does not modify the nonnullary function symbols. We give an example : consider $P(X_1)$, $n = 3$ and $S = X_1 \cup \overline{X_3}$. Then $L_{P(S),(X_1,X_2,X_3),k} = h^{-1}(L_{P,(X_1),k})$ where $h(1x0) = h(1x1) = 1$, $h(0x0) = 1, h(0x1) = 0$, for every $x = 0, 1$, i.e., $h(x_1, x_2, x_3) = x_1 \vee \neg x_3$.

Claim 7 : From an automaton $\mathcal{A}_{P,(X_1,...,X_m),k}$ that accepts $L_{P,(X_1,...,X_m),k}$ one gets an automaton $\mathcal{A}_{P(S_1,...,S_m),(X_1,...,X_n),k}$ with same number of states that accepts $L_{P(S_1,...,S_m),(X_1,...,X_n),k}$. If $\mathcal{A}_{P,(X_1,...,X_m),k}$ is deterministic, then $\mathcal{A}_{P(S_1,...,S_m),(X_1,...,X_n),k}$ is also deterministic.

Claim 8 : If φ is a positive Boolean combination of $\mathcal{P}$ -atomic formulas $\alpha_1, ..., \alpha_d$ for which we have constructed non-deterministic (resp. deterministic) automata $\mathcal{A}_1, ..., \mathcal{A}_d$ with respectively $N_1, ..., N_d$ states, one can construct a "product" non-deterministic (resp. deterministic) automaton for φ with $N_1 \times ... \times N_d$ states (perhaps less after deletion of useless states).

Claim 9 : If θ is the sentence $\exists X_1, ..., X_n.\varphi$, and we have constructed an automaton $\mathcal{A}$ for φ, we obtain one, usually not deterministic even if $\mathcal{A}$ is, for θ with same number of states, by the classical "projection" that deletes the Boolean sequences from the constant symbols of $F_k^{(n)}$.

Theorem 10 : Let $\mathcal{P}$ be a set of basic graph properties. For each $P \in \mathcal{P}$ and for each k, let a nondeterministic automaton $\mathcal{A}_{P,(X_1,...,X_m),k}$ with at most $N(k)$ states be known. For every sentence θ of the form $\exists X_1, ..., X_n.\varphi$ where φ is a positive Boolean combination of d $\mathcal{P}$-atomic formulas, a nondeterministic automaton $\mathcal{A}_{\theta,\varepsilon,k}$ (over the signature F_k) with at most $N(k)^d$ states can be constructed.

In many cases (see below) deterministic automata for the properties of $\mathcal{P}$ are such that $N(k) = 2^{O(k^2)}$. The only nondeterministic transitions of $\mathcal{A}_{\theta,\varepsilon,k}$ are those associated with the constants. Then, with these hypotheses and the notation of Theorem 6:

Theorem 11 : For every term t in $T(F_k)$, one can decide in time $O(\mid t \mid .N(k)^{2d})$ if the graph $G(t)$ satisfies θ.

Proof : One can decide in time $O(\mid t \mid .N^2)$ if a nondeterministic automaton with N states over a binary signature and nondeterministic transitions limited to constants accepts a term t. In this computation, one considers θ as fixed, and the time to fire a transition constant.

Application 12: Some basic graph properties and their automata; experiments[1]

We classify graph properties in terms of the numbers of states $N(k)$ of deterministic automata that check them.

Polynomial-sized automata.

The automata for $X_1 \subseteq X_2$, $X_1 = \emptyset$, $Sgl(X_1)$ have no more than 4 states. For $E(X_1, X_2)$ we can build an automaton with $k^2 + k + 2$ states.

Single-exponential sized automata.

For $Part(X_1, ..., X_p)$, $Disjoint(X_1, ..., X_p)$ and $St(X_i) : 2^k$ states.

For $Link(X_i, X_j)$ and $Dom(X_i, X_j) : 2^{2k}$ states.

For $Path(X_1, X_2)$, we have constructed a (non-minimal) automaton with 2^{k^2} states. Its minimization (with the software ATUOWRITE [Dur]) has given the results shown in the following table.

$Path(X_1, X_2)$		
cwd	A	min(A)
2	25	12
3	214	128
4	3443	2197

For $NoCycle(X_1) : 2^{O(k^2)}$ states.

For d-vertex coloring, we get, by Theorem 10, a nondeterministic automaton with 2^{kd} states.

We have been able to construct minimal deterministic automata for the following values of (k, d) shown in the following table.

Vertex coloring				
cwd	d	A	det(A)	min(A)
2	2	16	12	8
2	3	64	37	14
2	4	256	96	23
2	5	1024	213	36
3	2	64	406	56
3	3	512	∞	

[1] The tables contain the number of states of the considered automata. The symbol ∞ means that the computation did not finish but did not run out of memory.

Double-exponential sized automata.

For checking connectivity, one can build a deterministic automaton with 2^{2^k} states. If $k = 2p + 1$, the corresponding minimal deterministic automaton has more than 2^{2^p} states. However, for connectivity of graphs of degree at most d, which may be enough in practice, we can build a single-exponential sized automaton with $2^{O((dk)^2)}$ states.

For having a circuit, we get 9 states for $cwd = 2$ and 81 for $cwd = 3$ and the program runs out of memory for $cwd = 4$. For indegree at most 1, we get :

cwd	2	3	4	5
A	24	123	621	3120

Perspectives: To make these constructions usable, we will not try to tabulate and minimize automata, but rather, we will describe their transitions by clauses. We will compute a transition each time it is necessary for running the automaton on a given term.

References

[BK] D.Basin, N. Klarlund, Automata based reasoning in hardware verification, *J. of Formal Methods in System Design*, 13 (1998) 255-288.

[CMR] B. Courcelle, J. A. Makowsky, U. Rotics, On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics* 108 (2001) 23-52

[DF] R. Downey et M. Fellows, *Parameterized complexity*, Springer-Verlag, 1999

[Dur] I. Durand, Autowrite: A Tool for Term Rewrite Systems and Tree Automata, *Electronic Notes TCS* 124(2005) 29-49

[FG] J. Flum, M. Grohe, *Parametrized complexity theory*, Springer, 2006.

[FriGro] M. Frick, M. Grohe: The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic* 130 (2004) 3-31

[GH] R. Ganian, P. Hlineny, On Parse Trees and Myhill-Nerode-type Tools for handling Graphs of Bounded Rank-width.

Discrete Applied Maths, In press.

[G1] G. Gottlob, R. Pichler, F. Wei: Abduction with Bounded Treewidth: From Theoretical Tractability to Practically Efficient Computation. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, July 2008, AAAI Press, 2008, pp. 1541-1546

[G2] G. Gottlob, R. Pichler, F. Wei: Monadic Datalog over Finite Structures with Bounded Treewidth, 2008, ArXiv, CoRR abs/0809.3140

[HliOum] P. Hlineny, S. Oum: Finding Branch-Decompositions and Rank-Decompositions. *SIAM J. Comput.* 38(2008) 1012-1032.

[KL] J. Kneis, A. Langer, A Practical Approach to Courcelle's Theorem, *Electronic Notes TCS*, 251 (2009) 65-81.

[Rao] M. Rao, MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theor. Comput. Sci.* 377(2007) 260-267.

Logical structures, cyclic graphs and genus of proofs

A. Carbone

Department of Computer Science, Université Pierre et Marie Curie-Paris 6
Alessandra.Carbone@lip6.fr

Abstract. We report on two recent results concerning the logical structure and the underlying graphs of propositional proofs.

First, we consider the genus of a proof as a measure of proof complexity and we discuss a few geometrical properties of logical flow graphs of proofs, with and without cuts, with the purpose in mind to represent how complicated a cut-free proof can be. The main result says that arbitrarily complicated non oriented graphs, that is graphs of arbitrarily large genus, can be encoded in a cut-free proof. This fact was proved by Richard Statman in his thesis written in the early seventies and never published. We reformulate Statman's result in a purely graph theoretical language and give a proof of it. We show that there are several ways to embed non oriented graphs of arbitrary complexity into cut-free proofs and provide some other direct embeddings of arbitrarily complex non oriented graphs into proofs possibly with cuts. We also show that given any formal circuit, we can codify it into a proof in such a way that the graph of the circuit corresponds to the logical flow graph of the encoding proof [1].

Second, we look at propositonal proofs by reformulating them in the more general framework of combinatorial mappings. Combinatorial proofs are abstract invariants for sequent calculus proofs, similarly to homotopy groups which are abstract invariants for topological spaces. Starting from the observation that sequent calculus fails to be surjective onto combinatorial proofs, we extract a syntactically motivated closure of sequent calculus from which there is a surjection onto a complete set of combinatorial proofs. We characterize a class of canonical sequent calculus proofs for the full set of propositional tautologies and derive a new completeness theorem for combinatorial propositions. The result is based on a definition of a mapping between combinatorial proofs and sequent calculus proofs which explicitly links the logical flow graph of a proof to a skew fibration between graphs of formulas. The categorical properties relating the original and the new mappings are explicitly discussed [2].

References

1. A. Carbone (2009) Logical structures and genus of proofs. *Annals of Pure and Applied Logic*, 161(2), 139-149.
2. A. Carbone (2009) A new mapping between combinatorial proofs and sequent calculus proofs read out from logical flow graphs, *Information and Computation*, 1-15, doi:10.1016/j.ic.2009.01.007

On Optimal Probabilistic Algorithms for SAT

Yijia Chen[1], Jörg Flum[2], and Moritz Müller[3]

[1] Shanghai Jiaotong University
BASICS, Department of Computer Science, Dongchuan Road 800, Shanghai 200240, China
Email: yijia.chen@cs.sjtu.edu.cn
[2] Albert-Ludwigs-Universität Freiburg
Abteilung für Mathematische Logik, 79104 Freiburg, Eckerstr. 1, Germany
Email: joerg.flum@math.uni-freiburg.de
[3] Centre de Recerca Matemàtica
Universitat Autònoma de Barcelona, Facultat de Ciències, 08193 Bellaterra, Spain
Email: mmueller@crm.cat

1. Introduction. A major aim in the development of algorithms for hard problems is to decrease the running time. In particular one asks for algorithms that are optimal: A deterministic algorithm $\mathbb{A}$ deciding a language $L \subseteq \Sigma^*$ is *optimal* (or *(polynomially) optimal* or *p-optimal*) if for any other algorithm $\mathbb{B}$ deciding L there is a polynomial p such that

$$t_{\mathbb{A}}(x) \leq p(t_{\mathbb{B}}(x) + |x|) \tag{1}$$

for all $x \in \Sigma^*$. Here $t_{\mathbb{A}}(x)$ denotes the running time of $\mathbb{A}$ on input x. If (1) is only required for all $x \in L$, then $\mathbb{A}$ is said to be an *almost optimal algorithm for* L (or to be *optimal on positive instances of* L).

Various recent papers address the question whether such optimal algorithms exist for NP-complete or coNP-complete problems (cf. [1]), even though the problem has already been considered in the seventies when Levin [4] observed that there exists an optimal algorithm that finds a witness for every satisfiable propositional formula. Furthermore the relationship between the existence of almost optimal algorithms for a language L and the existence of "optimal" proof systems for L has been studied [3, 5].

Here we present a result (see Theorem 2.1) that can be interpreted as stating that (under the assumption of the existence of one-way functions) there is no optimal *probabilistic* algorithm for SAT.

2. Probabilistic speed-up. For a propositional formula α we denote by $\|\alpha\|$ the number of literals in it, counting repetitions. Hence, the actual length of any reasonable encoding of α is polynomially related to $\|\alpha\|$.

The main result of this abstract reads as follows:

Theorem 2.1. *Assume one-way functions exist. Then for every probabilistic algorithm* $\mathbb{A}$ *deciding* SAT *there exists a probabilistic algorithm* $\mathbb{B}$ *deciding* SAT *such that for all* $d \in \mathbb{N}$ *and sufficiently large* $n \in \mathbb{N}$

$$\Pr\Big[\textit{there is a satisfiable } \alpha \textit{ with } \|\alpha\| = n \textit{ such that } \mathbb{A} \textit{ does not accept } \alpha \textit{ in at most } (t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d \textit{ steps}\Big] \geq \frac{1}{5}$$

Note that $t_{\mathbb{A}}(\alpha)$ *and* $t_{\mathbb{B}}(\alpha)$ *are random variables, and the probability is taken over the coin tosses of* $\mathbb{A}$ *and* $\mathbb{B}$ *on* α.

Here we say that a probabilistic algorithm $\mathbb{A}$ decides SAT if it decides SAT as a nondeterministic algorithm, that is

$$\alpha \in \text{SAT} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] > 0,$$
$$\alpha \notin \text{SAT} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] = 0.$$

In particular, $\mathbb{A}$ can only err on 'yes'-instances.

Note that in the first condition the error probability is not demanded to be bounded away from 0, say by a constant $\varepsilon > 0$. As a more usual notion of probabilistic decision, say $\mathbb{A}$ *decides* SAT *with one-sided error* ε if

$$\alpha \in \text{SAT} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] > 1 - \varepsilon,$$
$$\alpha \notin \text{SAT} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] = 0.$$

For this concept we get

Corollary 2.2. *Assume one-way functions exist and let $\varepsilon > 0$. Then for every probabilistic algorithm $\mathbb{A}$ deciding* SAT *with one-sided error ε there exists a probabilistic algorithm $\mathbb{B}$ deciding* SAT *with one-sided error ε such that for all $d \in \mathbb{N}$ and sufficiently large $n \in \mathbb{N}$*

$$\Pr\Big[\textit{there is a satisfiable } \alpha \textit{ with } \|\alpha\| = n \textit{ such that } \mathbb{A} \textit{ does not accept } \alpha \textit{ in at most } (t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d \textit{ steps}\Big] \geq \frac{1}{5}$$

This follows from the fact that in the proof of Theorem 2.1 we choose the algorithm $\mathbb{B}$ in such way that on any input α the error probability of $\mathbb{B}$ on α is not worse than the error probability of $\mathbb{A}$ on α.

3. Witnessing failure. The proof of Theorem 2.1 is based on the following result.

Theorem 3.1. *Assume that one-way functions exist. Then there is a probabilistic polynomial time algorithm $\mathbb{C}$ satisfying the following conditions.*
(1) On input $n \in \mathbb{N}$ in unary the algorithm $\mathbb{C}$ outputs with probability one a satisfiable formula β with $\|\beta\| = n$.
(2) For every $d \in \mathbb{N}$ and every probabilistic algorithm $\mathbb{A}$ deciding SAT *and sufficiently large $n \in \mathbb{N}$*

$$\Pr\big[\mathbb{A} \textit{ does not accept } \mathbb{C}(n) \textit{ in } n^d \textit{ steps}\big] \geq \frac{1}{3}.$$

In the terminology of fixed-parameter tractability this theorem tells us that the parameterized construction problem associated with the following parameterized decision problem p-COUNTEREXAMPLE-SAT is in a suitably defined class of randomized nonuniform fixed-parameter tractable problems.

Instance:	An algorithm $\mathbb{A}$ *deciding* SAT and $d, n \in \mathbb{N}$ in unary.
Parameter:	$\|\mathbb{A}\| + d$.
Problem:	Does there exist a satisfiable CNF-formula α with $\|\alpha\| = n$ such that $\mathbb{A}$ does not accept α in n^d many steps?

Note that this problem is a promise problem. We can show:

Theorem 3.2. *Assume that one way functions exist. Then p-*COUNTEREXAMPLE-SAT *is nonuniformly fixed-parameter tractable.*[4]

This result is an immediate consequence of the following result interesting in its own right:

Theorem 3.3. *Assume that one way functions exist. For every infinite $I \subseteq \mathbb{N}$ the problem*

SAT_I
Instance: A CNF-formula α with $\|\alpha\| \in I$.
Problem: Is α satisfiable?

is not *in* PTIME.

We consider the construction problem associated with p-COUNTEREXAMPLE-SAT, that is, the problem

Instance:	An algorithm $\mathbb{A}$ *deciding* SAT and $d, n \in \mathbb{N}$ in unary.
Parameter:	$\|\mathbb{A}\| + d$.
Problem:	Construct a satisfiable CNF-formula α with $\|\alpha\| = n$ such that $\mathbb{A}$ does not accept α in n^d many steps, if one exists.

We do not know anything on its (deterministic) complexity; its nonuniform fixed-parameter tractability would rule out the existence of strongly almost optimal algorithms for SAT. By definition, an algorithm $\mathbb{A}$ deciding SAT is a *strongly almost optimal algorithm for* SAT if there is a polynomial p such that for any other algorithm $\mathbb{B}$ deciding SAT

$$t_{\mathbb{A}}(\alpha) \leq p(t_{\mathbb{B}}(\alpha) + |\alpha|)$$

for all $\alpha \in$ SAT. Then the precise statement of the result just mentioned reads as follows:

[4] This means, there is a $c \in \mathbb{N}$ such that for every algorithm $\mathbb{A}$ deciding SAT and every $d \in \mathbb{N}$ there is an algorithm that decides for every $n \in \mathbb{N}$ whether $(\mathbb{A}, d, n)$ is a positive instance of p-COUNTEREXAMPLE-SAT in time $O(n^c)$; here the constant hidden in $O(\ )$ may depend on $\mathbb{A}$ and d.

Proposition 3.4. *Assume that* P $\neq$ NP. *If the construction problem associated with* p-COUNTEREXAMPLE-SAT *is nonuniformly fixed-parameter tractable, then there is no strongly almost optimal algorithms for* SAT.

4. Some Proofs. We now show how to use an algorithm $\mathbb{C}$ as in Theorem 3.1 to prove Theorem 2.1.

Proof of Theorem 2.1 from Theorem 3.1: Let $\mathbb{A}$ be an algorithm deciding SAT. We choose $a \in \mathbb{N}$ such that for every $n \geq 2$ the running time of the algorithm $\mathbb{C}$ (provided by Theorem 3.1) on input n is bounded by n^a. We define the algorithm $\mathbb{B}$ as follows:

$\mathbb{B}(\alpha)$ // $\alpha \in$ CNF

1. $\beta \leftarrow \mathbb{C}(\|\alpha\|)$.
2. **if** $\alpha = \beta$ **then** accept and halt.
3. **else** Simulate $\mathbb{A}$ on α.

Let $d \in \mathbb{N}$ be arbitrary. Set $e := d \cdot (a+2) + 1$ and fix a sufficiently large $n \in \mathbb{N}$. Let S_n denote the range of $\mathbb{C}(n)$. Furthermore, let $T_{n,\beta,e}$ denote the set of all strings $r \in \{0,1\}^{n^e}$ that do not determine a (complete) accepting run of $\mathbb{A}$ on β that consists in at most n^e many steps. Observe that a (random) run of $\mathbb{A}$ does not accept β in at most n^e steps if and only if $\mathbb{A}$ on β *uses* $T_{n,\beta,e}$, that is, its first at most n^e many coin tosses on input β are described by some $r \in T_{n,\beta,e}$. Hence by (2) of Theorem 3.1 we conclude

$$\sum_{\beta \in S_n} \big(\Pr[\beta = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\beta,e}] \big) \geq \frac{1}{3}. \tag{2}$$

Let $\alpha \in S_n$ and apply $\mathbb{B}$ to α. If the execution of $\beta \leftarrow \mathbb{C}(\|\alpha\|)$ in Line 1 yields $\beta = \alpha$, then the overall running time of the algorithm $\mathbb{B}$ is bounded by $O\big(n^2 + t_{\mathbb{C}}(n)\big) = O(n^{a+1}) \leq n^{a+2}$ for n is sufficiently large. If in such a case a run of the algorithm $\mathbb{A}$ on input α uses an $r \in T_{n,\alpha,e}$, then it does not accept α in time $n^e = n^{(a+2)\cdot d+1}$ and hence not in time $(t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d$. Therefore,

$$\begin{aligned}
&\Pr\Big[\text{there is a satisfiable } \alpha \text{ with } \|\alpha\| = n \text{ such that } \mathbb{A} \text{ does not accept } \alpha \text{ in at most } (t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d \text{ steps}\Big] \\
&\geq 1 - \Pr\big[\text{for every input } \alpha \in S_n \text{ the algorithm } \mathbb{B} \text{ does not generate } \alpha \text{ in Line 3, or } \mathbb{A} \text{ does not use } T_{n,\alpha,e}\big] \\
&= 1 - \prod_{\alpha \in S_n} \big((1 - \Pr[\alpha = \mathbb{C}(n)]) + \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \notin T_{n,\alpha,e}] \big) \\
&= 1 - \prod_{\alpha \in S_n} \big(1 - \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\alpha,e}] \big) \\
&\geq 1 - \left(\frac{\sum_{\alpha \in S_n} \big(1 - \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\alpha,e}]\big)}{|S_n|} \right)^{|S_n|} \\
&= 1 - \left(1 - \frac{\sum_{\alpha \in S_n} \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\alpha,e}]}{|S_n|} \right)^{|S_n|} \\
&\geq 1 - \left(1 - \frac{1}{3 \cdot |S_n|} \right)^{|S_n|} \qquad \text{(by (2))} \\
&\geq \frac{1}{5}.
\end{aligned}$$

□

Theorem 3.1 immediately follows from the following lemma.

Lemma 4.1. *Assume one-way functions exist. Then there is a randomized polynomial time algorithm* $\mathbb{H}$ *satisfying the following conditions.*

(H1) *Given* $n \in \mathbb{N}$ *in unary the algorithm* $\mathbb{H}$ *computes with probability one a* satisfiable CNF α *of size* $\|\alpha\| = n$.

(H2) *For every* probabilistic *algorithm* $\mathbb{A}$ *deciding* SAT *and every* $d, p \in \mathbb{N}$ *there exists an* $n_{\mathbb{A},d,p} \in \mathbb{N}$ *such that for all* $n \geq n_{\mathbb{A},d,p}$

$$\Pr\big[\mathbb{A} \textit{ accepts } \mathbb{H}(n) \textit{ in time } n^d\big] \leq \frac{1}{2} + \frac{1}{n^p},$$

where the probability is taken uniformly over all possible outcomes of the internal coin tosses of the algorithms $\mathbb{A}$ *and* $\mathbb{H}$.

(H3) *The cardinality of the range of (the random variable)* $\mathbb{H}(n)$ *is superpolynomial in* n.

Sketch of proof: We present the construction of the algorithm $\mathbb{H}$. By the assumption that one-way functions exist, we know that there is a pseudorandom generator (e.g. see [2]), that is, there is an algorithm $\mathbb{G}$ such that:

(G1) For every $s \in \{0,1\}^*$ the algorithm $\mathbb{G}$ computes a string $\mathbb{G}(s)$ with $|\mathbb{G}(s)| = |s| + 1$ in time polynomial in $|s|$.

(G2) For every probabilistic polynomial time algorithm $\mathbb{D}$, every $p \in \mathbb{N}$, and all sufficiently large $\ell \in \mathbb{N}$ we have

$$\left| \Pr_{s\in\{0,1\}^\ell} \big[\mathbb{D}(\mathbb{G}(s)) = 1\big] - \Pr_{r\in\{0,1\}^{\ell+1}} \big[\mathbb{D}(r) = 1\big] \right| \leq \frac{1}{\ell^p}$$

(In the above terms, the probability is also taken over the internal coin toss of $\mathbb{D}$.)

Let the language Q be the range of $\mathbb{G}$,

$$Q := \big\{\mathbb{G}(s) \;\big|\; s \in \{0,1\}^*\big\}.$$

Q is in NP by (G1). Hence, there is a polynomial time reduction $\mathbb{S}$ from Q to SAT, which we can assume to be injective. We choose a constant $c \in \mathbb{N}$ such that $\|\mathbb{S}(r)\| \leq |r|^c$ for every $r \in \{0,1\}^*$. For every propositional formula β and every $n \in \mathbb{N}$ with $n \geq \|\beta\|$ let $\beta(n)$ be an equivalent propositional formula with $\|\beta(n)\| = n$. We may assume that $\beta(n)$ is computed in time polynomial in n.

One can check that the following algorithm $\mathbb{H}$ has the properties claimed in the lemma.

$\mathbb{H}(n)$ // $n \in \mathbb{N}$

1. $m \leftarrow \big\lfloor \sqrt[c]{n-1} \big\rfloor - 1$
2. Choose an $s \in \{0,1\}^m$ uniformly at random.
3. $\beta \leftarrow \mathbb{S}(\mathbb{G}(s))$.
4. Output $\beta(n)$

□

Acknowledgements. We wish to thank an anonymous referee for pointing out a mistake in an earlier formulation of Theorem 2.1. The third author thanks the John Templeton Foundation for its support under Grant #13152, *The Myriad Aspects of Infinity.*

References

1. O. Beyersdorff and Z. Sadowski. Characterizing the existence of optimal proof systems and complete sets for promise classes. Electronic Colloquium on Computational Complexity, Report TR09-081, 2009.
2. O. Goldreich. *Foundations of Cryptograph, Volume 1 (Basic Tools)*. Cambridge University Press, 2001.
3. J. Krajicèk and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. Jour. Symb. Logic, 54(3):1063–1079, 1989.
4. L. Levin. Universal search problems (in russian). Problemy Peredachi Informatsii 9(3):115-116, 1973.
5. J. Messner. On optimal algorithms and optimal proof systems. STACS'99, LNCS 1563:541–550 1999.

Pebble Games for Rank Logics

Anuj Dawar and Bjarki Holm

University of Cambridge Computer Laboratory
{anuj.dawar, bjarki.holm}@cl.cam.ac.uk

Abstract. We show that equivalence in finite-variable infinitary logic with rank operators can be characterised in terms of pebble games based on set partitions. This gives us a game-based method for proving lower bounds for FOR and IFPR, the extensions of first-order and fixed-point logic with rank operators, respectively. As an illustration of the game method, we establish that over finite structures, $\mathrm{IFPR}_p^{[2]} \neq \mathrm{IFPR}_q^{[2]}$ for distinct primes p and q, where $\mathrm{IFPR}_p^{[m]}$ is the restriction of IFPR that only has operators for defining rank of matrices of arity at most m over GF_p.

1 Introduction

The question of whether there is a logical characterisation of the complexity class PTIME remains the fundamental open problem of descriptive complexity. Most attempts to answer this question have focused on finding suitable extensions of first-order logic that can describe exactly all properties decidable in PTIME. In this way, Immerman and Vardi independently showed that on inputs equipped with a linear order, inflationary fixed-point logic (IFP) expresses exactly the properties in PTIME [6, 9]. In the absence of an order, IFP is too weak to express all properties in PTIME. In particular, it fails to define very simple cardinality properties. This immediate deficiency is easily solved by extending the logic with counting terms, which gives us fixed-point logic with counting (IFPC), which was at one time conjectured to be a logic for PTIME. However, Cai, Fürer and Immerman later showed that this logic still falls short of capturing PTIME [2].

Since the result of Cai *et al.*, a number of examples have been constructed of polynomial-time decidable properties that are not expressible in IFPC. Recently it was observed that all these examples can be reduced to the problem of determining the solvability of systems of linear equations (see [1, 3, 4]). Over finite fields, this can be further reduced to the problem of computing the *rank* of a definable (unordered) matrix. Computing rank can be understood as a generalised form of counting where, rather than counting the cardinality of a definable set, one is allowed to count the dimension of a definable vector space. This suggests that the key weakness of IFPC is that the form of counting it incorporates is too weak. In [4], Dawar *et al.* proposed fixed-point logic with rank (IFPR), an extension of IFP with operators for computing the rank of a matrix over a fixed prime field. It is shown in [4] that IFPR can express various polynomial-time properties known to separate IFPC from PTIME. It is an open question whether IFPR captures PTIME.

Despite some positive results on the expressive power of logical rank operators, not much is known about their limitations. For instance, it is not even known whether first-order logic with rank (FOR) is strictly less expressive than IFPR over finite structures, although that would seem likely. To establish such separations we seem to lack general methods for proving inexpressiblity in FOR and IFPR. This leads us to consider variations of Ehrenfeucht-Fraïssé-style pebble games, which form an essential tool for analysing expressiveness of other extensions of first-order logic, such as IFP and IFPC.

In this abstract we introduce a new pebble game based on set partitions that characterises expressivity in an infinitary logic with rank quantifiers, which subsumes both FOR and IFPR. This type of game turns out to be quite generic, with standard games for both IFP and IFPC occurring as special cases. As an illustration of the game method, we establish that over finite structures, $\mathrm{IFPR}_p^{[2]} \neq \mathrm{IFPR}_q^{[2]}$ for distinct primes p and q, where $\mathrm{IFPR}_p^{[m]}$ is the restriction of IFPR that only has operators for defining rank of matrices of arity at most m over GF_p. This partially resolves one of the open questions posed in [4]. Due to space constraints, details of all proofs are omitted.

2 Rank Logics

We assume that all structures are finite and all vocabularies are finite and relational. For a logic $\mathcal{L}$, we write $\mathbf{A} \equiv^{\mathcal{L}} \mathbf{B}$ to denote that the structures $\mathbf{A}$ and $\mathbf{B}$ are not distinguished by any sentence of $\mathcal{L}$. We write $|\mathbf{A}|$ for the universe of a structure $\mathbf{A}$ and write $\|\mathbf{A}\|$ for the cardinality of $|\mathbf{A}|$. We often denote tuples $(v_1, \ldots, v_k)$ by $\boldsymbol{v}$ and denote their length by $|\boldsymbol{v}|$.

Inflationary fixed-point logic (IFP) is obtained by adding to first-order logic the ability to define inflationary fixed-points of inductive definitions. It is easily shown that on finite structures, IFP fails to express very simple cardinality queries. We define counting terms $\#x\varphi$ to denote the number of elements that satisfy the formula φ. By adding to IFP rules for building counting terms, we obtain inflationary fixed-point logic with counting (IFPC). For a detailed discussion of these logics we refer to the standard literature [5, 7].

Definable matrices over finite fields. We write $[m]$ to denote the set $\{0, \ldots, m-1\}$, for $m \geq 1$. For sets I and J, an $I \times J$ matrix over the prime field GF_p can be seen as a function $M : I \times J \to [p]$. Here the rows of M are indexed by I and the columns of M are indexed by J. Observe that the sets I and J are not necessarily ordered. Natural matrix properties, such as singularity and rank, are invariant under permutations of rows and columns, and are therefore well-defined in the context of unordered row and column sets.

Using our notation for describing matrices, a formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ interpreted in a structure $\mathbf{A}$ defines a GF_2 matrix $M_\varphi^{\mathbf{A}} : A^{|\boldsymbol{x}|} \times A^{|\boldsymbol{y}|} \to \{0, 1\}$ given by $M_\varphi^{\mathbf{A}}(\boldsymbol{a}, \boldsymbol{b}) = 1$ if, and only if, $(\mathbf{A}, \boldsymbol{a}, \boldsymbol{b}) \models \varphi$. More generally, let $\Phi = (\varphi_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, \varphi_l(\boldsymbol{x}, \boldsymbol{y}))$ be an l-tuple of formulas, with $1 \leq l < p$ and p prime. Interpreted in a structure $\mathbf{A}$, these formulas define a matrix $M_\Phi^{\mathbf{A}} : A^{|\boldsymbol{x}|} \times A^{|\boldsymbol{y}|} \to [p]$ given by

$$M_\Phi^{\mathbf{A}}(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{l} i M_{\varphi_i}^{\mathbf{A}}(\boldsymbol{a}, \boldsymbol{b}) \pmod{p}.$$

For example, for any formula $\varphi(x)$, the formula $(x = y \wedge \varphi(x))$ interpreted in a structure $\mathbf{A}$ defines a square diagonal matrix, with 1 in position $(a, a) \in A \times A$ on the diagonal if, and only if, $(\mathbf{A}, a) \models \varphi$.

Fixed-point logic with rank. We recall the basic definition of rank logics. To simplify the transition to infinitary rank logics later, our presention of rank operators differs slightly from that of Dawar *et al.* [4], although the two definitions can be seen to be equivalent. Specifically, in [4] we consider matrices over GF_p defined by a single number term modulo p, instead of looking at tuples of formulas as we do below.

Inflationary fixed-point logic with rank (IFPR) has two sorts of variables: $x_1, x_2, \ldots$ ranging over the domain elements of the structure, and $\nu_1, \nu_2, \ldots$ ranging over the non-negative integers. All quantification of number variables has to be bounded. Thus, is if ν is a number variable, its binding quantifier must appear in the form $(\forall \nu \leq t\ \varphi)$ or $(\exists \nu \leq t\ \varphi)$ for a numeric term t and a formula φ. In addition, we also have second-order variables $X_1, X_2, \ldots$, each of which has a type which is a finite string in $\{\mathrm{element}, \mathrm{number}\}^*$. Thus, if X is a variable of type (element,number), it is to be interpreted by a binary relation relating elements to numbers. We write $\mathbf{ifp}_{X \leftarrow \boldsymbol{x}\boldsymbol{\nu} \leq \boldsymbol{t}} \varphi$ for the inflationary fixed-point of φ over the relation variable X of type $(\mathrm{element}^{|\boldsymbol{x}|}, \mathrm{number}^{|\boldsymbol{\nu}|})$, where the number variables in $\boldsymbol{\nu}$ are bounded by the numeric terms in $\boldsymbol{t}$. By closing first-order logic under the formation of inflationary fixed-points, we get IFP in a two-sorted setting. The logic IFPR is obtained by extending the formula-formation rules of IFP with a rule for building *rank terms* in the following way:

> for prime p and $l \in \{1, \ldots, p-1\}$, if $\Phi = (\varphi_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, \varphi_l(\boldsymbol{x}, \boldsymbol{y}))$ is an l-tuple of formulas and $\boldsymbol{x}$ and $\boldsymbol{y}$ are tuples of variables of the first sort, then $\mathrm{rk}_p(\boldsymbol{x}, \boldsymbol{y})\Phi$ is a term.

The intended semantics is that $\mathrm{rk}_p(\boldsymbol{x}, \boldsymbol{y})\Phi$ denotes the rank (i.e. the member of the number sort) over GF_p of the matrix defined by the formulas Φ. More generally, we can define rank terms for formulas Φ with number variables. In this case, all free number variables have to be bounded by number terms, as is described in more detail in [4]. The arity of a rank operator $\mathrm{rk}_p(\boldsymbol{x}, \boldsymbol{y})$ is $|\boldsymbol{x}| + |\boldsymbol{y}|$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are assumed to be tuples of distinct variables.

We write $\mathrm{IFPR}^{[m]}$ for the fragment of IFPR in which all rank operators have arity at most m and write IFPR_p to denote the fragment where only rank operators rk_p are allowed. Putting the two together, we obtain logics $\mathrm{IFPR}_p^{[m]}$ where only rank operators rk_p of arity at most m are allowed.

It is easy to see that rank logics can express the cardinality of any definable set. Indeed, for a formula $\varphi(x)$ and prime p, the rank term $\mathrm{rk}_p(x,y)(x = y \wedge \varphi(x))$ is equivalent to the counting term $\#x\varphi$, as the rank of a diagonal matrix is exactly the number of non-zero entries along the diagonal. This immediately implies that each of the rank logics IFPR_p is at least as expressive as IFPC.

Infinitary rank logics. For each natural number i and prime p, we consider a quantifier rk_p^i where $\mathbf{A} \models \mathrm{rk}_p^i \boldsymbol{xy}\,(\varphi_1, \ldots, \varphi_{p-1})$ if, and only if, the rank of the $|\mathbf{A}|^{|\boldsymbol{x}|} \times |\mathbf{A}|^{|\boldsymbol{y}|}$ matrix defined by $(\varphi_1(\boldsymbol{x},\boldsymbol{y}), \ldots, \varphi_{p-1}(\boldsymbol{x},\boldsymbol{y}))$ over $\mathbf{A}$ is i. Here the rank is taken over GF_p. Let R^k denote k-variable infinitary logic with rank quantifiers. The logic R^ω is given by $R^\omega = \bigcup_{k\in\omega} R^k$. That is, R^ω consists of infinitary rank formulas in which each formula has only finitely many variables. We let R_p^k denote the sublogic of R^k where only rank quantifiers of the form rk_p^i are allowed. We also write $R^{k;[m]}$ and $R_p^{k;[m]}$ to denote the fragments of R^k and R_p^k, respectively, with rank quantifiers of arity at most m, where the arity of a quantifier $\mathrm{rk}_p^i \boldsymbol{xy}$ is $|\boldsymbol{x}| + |\boldsymbol{y}|$. Clearly, $m \leq k$. It can be shown that every formula of $\mathrm{IFPR}_p^{[m]}$ is equivalent to one of $R_p^{\omega;[m]} = \bigcup_{k\in\omega} R_p^{k;[m]}$. Hence, $\mathrm{IFPR} \subseteq \mathrm{R}^\omega$. It is shown in [4] that for any $m \geq 2$, $R^{k;[m]}$ is strictly less expressive than $R^{k;[m+1]}$. Hence also $\mathrm{IFPR}^{[m]} \subsetneq \mathrm{IFPR}^{[m+1]}$.

3 Games for Logics with Rank

We give a game characterisation of equivalence in the logics $R_p^{k;[m]}$. To describe the game we will use the following notation. Let I and J be finite sets, $\mathbf{P}$ a set partition of $I \times J$, and $\gamma : \mathbf{P} \to [p]$ a labeling of the parts in $\mathbf{P}$, with p prime. Then $M_\gamma^{\mathbf{P}}$ denotes the $I \times J$ matrix over GF_p defined by

$$M_\gamma^{\mathbf{P}}(i,j) = \alpha \in [p] \Leftrightarrow \exists P \in \mathbf{P}\ \big((i,j) \in P \wedge \gamma(P) = \alpha\big).$$

We first consider the game for $R_p^{k;[m]}$ when $m = 2$. The game board of the k-pebble 2-ary rank partition game over GF_p consists of two structures $\mathbf{A}$ and $\mathbf{B}$ and k pairs of pebbles $(a_i, b_i), 1 \leq i \leq k$. The pebbles $a_1, \ldots, a_l$ are initially placed on the elements of an l-tuple $\boldsymbol{s}$ of elements in $\mathbf{A}$, and the pebbles $b_1, \ldots, b_l$ on an l-tuple $\boldsymbol{t}$ in $\mathbf{B}$, $l \leq k$. There are two players, Spoiler and Duplicator. At each round, Spoiler picks up two pairs of corresponding pebbles (a_i, b_i) and (a_j, b_j) for some i and j. Duplicator has to respond by choosing (a) partitions $\mathbf{P}$ of $A \times A$ and $\mathbf{Q}$ of $B \times B$, with $|\mathbf{P}| = |\mathbf{Q}|$; and (b) a bijection $f : \mathbf{P} \to \mathbf{Q}$, such that for all labelings $\gamma : \mathbf{P} \to [p]$,

$$\mathrm{rk}_p(M_\gamma^{\mathbf{P}}) = \mathrm{rk}_p(M_{f(\gamma)}^{\mathbf{Q}}).$$

Here $f(\gamma) : \mathbf{Q} \to [p]$ is the labeling of $\mathbf{Q}$ defined by $f(\gamma)(Q) = \gamma(f^{-1}(Q)$ for all $Q \in \mathbf{Q}$. Spoiler next picks a part $P \in \mathbf{P}$, and places the pebbles (a_i, a_j) on an element in $P \subseteq A \times A$ and places the pebbles (b_i, b_j) on an element in $f(P) \subseteq B \times B$. This completes one round in the game. If, after this exchange, the partial map $f : \mathbf{A} \to \mathbf{B}$ given by $a_i \mapsto b_i$ is not a partial isomorphism, or Duplicator is unable to produce the required partitions, then Spoiler has won the game; otherwise it can continue for another round.

For the more general case of m-ary rank quantifiers over GF_p, we modify the above game so that at each round, Spoiler starts by choosing two integers r and s with $r + s = m$. He then picks up m pebbles in some order from $\mathbf{A}$ and the m corresponding pebbles in the same order from $\mathbf{B}$. Duplicator has to respond by choosing partitions $\mathbf{P}$ and $\mathbf{Q}$ of $A^r \times A^s$ and $B^r \times B^s$, respectively, and a bijection $f : \mathbf{P} \to \mathbf{Q}$ between the two partitions. The rest of the round proceeds in exactly the same way as above, with Spoiler finally choosing a part $P \in \mathbf{P}$ and placing the m pebbles in $\mathbf{A}$ on an element in P (in the order they were chosen earlier) and the corresponding m pebbles in $\mathbf{B}$ on an element in $f(P)$ (in the same order). We denote the *k-pebble m-ary rank partition game over* GF_p played on structures $\mathbf{A}$ and $\mathbf{B}$ by $\mathcal{G}_p^{k;[m]}(\mathbf{A}, \mathbf{B})$.

Theorem 1. *Duplicator has a strategy for playing $\mathcal{G}_p^{k;[m]}(\mathbf{A}, \mathbf{B})$ forever if, and only if, $\mathbf{A} \equiv^{R_p^{k;[m]}} \mathbf{B}$.*

Write L^k to denote k-variable infinitary logic and C^k to denote the extension of L^k with counting quantifiers (see [7] for more details). The idea behind the rank partition game can also be used to give alternative characterisations of the relations $\equiv^{L^k}$ and $\equiv^{C^k}$. At each round in the *k-pebble cardinality partition game* on $\mathbf{A}$ and $\mathbf{B}$, the Spoiler picks up a pair of pebbles (a_i, b_i) for some i. Duplicator has to respond by choosing (a) partitions $\mathbf{P}$ of A and $\mathbf{Q}$ of B, with $|\mathbf{P}| = |\mathbf{Q}|$; and (b) a bijection $f : \mathbf{P} \to \mathbf{Q}$, such that for all parts $P \in \mathbf{P}$: $|P| = |f(P)|$. Spoiler then picks a part $P \in \mathbf{P}$, and places a_i on an element in $P \subseteq A$ and places b_i on an element in $f(P) \subseteq B$. This completes one round in the game. If Duplicator fails to produce the required partitions or the partial map defined by the pebbled elements is not a partial isomorphism, then Spoiler wins the game. Otherwise it can continue for another round. It can be shown that Duplicator has a strategy to play this game forever if, and only if, $\mathbf{A} \equiv^{C^k} \mathbf{B}$. Similarly, we can define the *k-pebble partition game* in exactly the same way as above, except we drop the requirement that the corresponding parts have to have the same size, i.e. Duplicator does not have to show that $|P| = |f(P)|$ for all $P \in \mathbf{P}$. It can be shown that Duplicator has a strategy to play this game forever if, and only if, $\mathbf{A} \equiv^{L^k} \mathbf{B}$. These two games can be seen as special cases of the generic rank partition game, which of course reflects the fact that the corresponding infinitary logics are both certain restrictions of infinitary rank logic.

4 Separation Results

The rank partition game can be used to delimit the expressive power of the rank logics restricted to a fixed arity and prime p. Specifically, using the game, we can show the following.

Theorem 2. *For all primes p and q where $q \equiv 1 \pmod p$, there is a property of finite graphs which is definable in $\mathrm{FOR}_q^{[2]}$ but not in $R_p^{\omega;[2]}$.*

The basic idea of the proof is as follows. For all primes p and q where $q \equiv 1 \pmod p$, and each $k \geq 2$, we construct a pair of non-isomorphic graphs $(\mathbf{A}_k^q, \mathbf{B}_k^q)$ which can be separated by a sentence of $\mathrm{FOR}_q^{[2]}$. We then show that Duplicator has a winning strategy in the game $\mathcal{G}_p^{k;[2]}(\mathbf{A}_k^q, \mathbf{B}_k^q)$, which shows that the classes of graphs $(\mathbf{A}_k^q)_{k\geq 2}$ and $(\mathbf{B}_k^q)_{k\geq 2}$ are not definable in $R_p^{\omega;[2]}$. The graphs $(\mathbf{A}_k^q, \mathbf{B}_k^q)$ are based on a construction of Torán [8]. This is essentially a way of encoding an arithmetic circuit modulo q into a given graph G. For instance, for $q = 2$ we get the graphs defined by Cai *et al.* [2] used to separate IFPC from PTIME. By starting with graphs G of large enough treewidth, we can ensure that for each k, Duplicator can hide the difference between $\mathbf{A}_k^q$ and $\mathbf{B}_k^q$ when playing the k-pebble rank partition game. Note that $q \equiv 1 \pmod p$ is required only for technical reasons in the proof; we believe the same method can be generalised for all distinct primes p and q. This gives us the following corollary, which partially resolves one of the open questions posed in [4].

Corollary 1. *For all primes p and q where $q \equiv 1 \pmod p$, $\mathrm{IFPR}_p^{[2]} \neq \mathrm{IFPR}_q^{[2]}$.*

References

1. A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.*, 410:1666–1683, 2009.
2. J-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.
3. A. Dawar. On the descriptive complexity of linear algebra. In *WoLLIC '08*, volume 5110 of *LNCS*, pages 17–25. Springer, 2008.
4. A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *Proc. 24th IEEE Symp. on Logic in Computer Science*, pages 113–122, 2009.
5. H. D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1999.
6. N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.
7. M. Otto. *Bounded Variable Logics and Counting — A Study in Finite Models*, volume 9 of *LNL*. Springer, 1997.
8. J. Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004.
9. M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM Symp. on the Theory of Computing*, pages 137–146, 1982.

How definitions, equivalent for Turing machines, cease to be equivalent, when generalized to Ordinal Time Turing Machines

Barnaby Dawson

University of Bristol

1 Extended Abstract

Mathematical models of hypercomputation, generalise Turing machines to allow for calculations that can be rigorously defined, but which could not be performed on a Turing machine, regardless of the time or memory it is assumed to possess. Various mathematical models of hypercomputation have been proposed in the past, notably the infinite time Turing machines of Hamkins & Lewis [1] and the Ordinal Time Turing Machines (OTTMs) of Koepke, Dawson and Bissell-Siders [2]. In recent years, other machine configurations have been proposed, including several bounded versions of OTTMs and ordinal time register machines. Koepke and Bissell-Siders [3] have even defined a programming language for an ordinal time register machine [4].

The OTTM is particularly interesting from a set theoretic point of view, as it is potentially unbounded in its operation, takes sequences of ordinal length as input, and returns such sequences as output. Its operation is pleasingly concrete, and has been linked to constructibility in set theory.

Unlike the situation for finite binary sequences, not all infinite binary sequences can be generated by an OTTM. Those that can, are called computable. Some sets are not computable, but can be recognised using a computable test. Those sets are called recognisable. The study of which sets are computable, and which recognisable, is of interest in applying ideas from Hypercomputation in set theory. OTTMs may also be run for α much time where α is a limit ordinal with weak closure properties (for instance admissibility). An admissible ordinal α is an ordinal with the property $L_\alpha \models KP$ (where KP is Kripke-Platek set theory).

Furthermore for Turing computation many concepts have multiple equivalent definitions. In particular, computing from an oracle, and the Turing computability relation, both admit several apparently distinct, yet nevertheless equivalent definitions. We define various α-computers from OTTMs operating with an α length tape (where α is admissible). We show that many definitions that are equivalent for Turing machines are no longer equivalent, when generalized to α-time.

For the α-computers we present the varying alternative definitions and investigate their differing consequences. We also give example sets which are recognisable for the OTTM. Finally we will describe several basic features a computability notion should be expected to possess and discuss which of the varying α-computers deliver these features.

(1) J. Hamkins and A. Lewis. Infinite time turing machines. Journal of Symbolic Logic, 65(2):567-604, 2000.
(2) P. Koepke. Turing computations on ordinals. Bulletin of Symbolic Logic, 11(3):377-397, 2005.
(3) R. Bissell-Siders P. Koepke. Register computations on ordinals. 47(6):529-548, 2008.
(4) R. Miller P. Koepke. An enhanced theory of infinite time register machine, In C. Dimitracopoulos A. Beckmann and B. Loewe, editors, Logic and theory of algorithms, volume 5028 of Lecture. Notes in Computer Science, pages 306-315. Athens University, Springer. 2008.

An Analogue of the Church-Turing Thesis for Computable Ordinal Functions

Tim Fischbach[1], Peter Koepke[2]

[1] Mathematisches Institut, Universität Bonn, Germany, fischbach@uni-bonn.de
[2] Mathematisches Institut, Universität Bonn, Germany, koepke@math.uni-bonn.de

1 Introduction

Classical computability theory introduces a wide range of computational models which attempt to formalise the intuitive notion of effective computability. Despite their vast dissimilarity all these approaches turn out to define the same class of computable functions. This equivalence gives rise to the Church-Turing thesis proposing the notion of recursiveness as the formal counterpart of the concept of effective computability.

Theorem 1 (P. Odifreddi [7]). *For $f : \omega \to \omega$ the following are equivalent:*

1. *f is recursive.*
2. *f is finitely definable.*
3. *f is* Turing *machine computable.*
4. *f is register machine computable.*
5. *f is* Gödel-Herbrand *computable.*
6. *f is λ-definable.*

Ordinal computability generalises these models of computation from a set theoretical point of view. In this paper we present ordinal versions of the notions referred to in Theorem 1 and establish their equivalence. Just like in the finitary case the identified class of functions proves resistant to technical variations of its definition. It is thus a stable and well-characterised class of effectively computable ordinal functions.

2 Ordinal Recursive Functions

The following definition proceeds in close analogy to the definition of (primitive) recursive functions on the natural numbers.

Definition 1 (R. Jensen, C. Karp [2]). *$f : Ord \to Ord$ is a* primitive recursive ordinal function *iff it is generated by the following scheme.*

- *Initial functions $P_{n,i}(x_1, \ldots, x_n) = x_i$ (for $1 \leq i \leq n < \omega$), $F(x) = 0$, $F(x) = x + 1$,*
- *Case distinction $C(x, y, u, v) = x$, if $u < v$, $= y$ otherwise.*
- *Substitution $F(\boldsymbol{x}, \boldsymbol{y}) = G(\boldsymbol{x}, H(\boldsymbol{y}), \boldsymbol{y})$, $F(\boldsymbol{x}, \boldsymbol{y}) = G(H(\boldsymbol{y}), \boldsymbol{y})$*
- *Primitive recursion $F(z, \boldsymbol{x}) = G(\bigcup\{F(u, \boldsymbol{x}) \mid u \in z\}, z, \boldsymbol{x})$*

$f : Ord \to Ord$ is an ordinal recursive function *iff it is generated by the above scheme together with:*

- *Minimisation rule $F(\boldsymbol{x}) = \min\{\xi \mid G(\xi, \boldsymbol{x}) = 0\}$, provided $\forall \boldsymbol{x} \exists \xi \; G(\xi, \boldsymbol{x}) = 0$*

Theorem 2 (R. Jensen, C. Karp [2]). *$f : Ord \to Ord$ is ordinal recursive iff it is $\triangle_1(L)$.*

3 Ordinal Turing Machines

We use the notion of *ordinal* TURING *machines (OTM)* introduced in [3]. A standard TURING program operates on a tape of ordinal length for possibly ordinal many steps. The behaviour at successor stages resembles that of finite TURING machines. At limit times the configuration is determined by a limit rule: Program state, position of the read-write head and cell contents are set to inferior limits of their respective prior values.

Ordinals α can be represented on the tape as characteristic functions $\chi_{\{\alpha\}}$ of singleton sets. A function $f : Ord \to Ord$ is said to be *OTM computable* if there is a TURING program which generates the tape content $\chi_{\{f(\alpha)\}}$ from such a representation of α.

In view of the desired equivalence we state:

Theorem 3 (P. Koepke, B. Seyfferth [5]). *$f : Ord \to Ord$ is OTM computable iff it is $\triangle_1(L)$.*

Since any OTM computation can be carried out inside the constructible universe L, the first half of the proof amounts to verifying absoluteness properties of OTM computations. The proof of the other implication proceeds by computing a bounded truth predicate for the constructible universe L.

4 Ordinal Register Machines

Pursuing the strategy of the previous section *ordinal register machines (ORM)* are obtained by generalising standard register machines. We permit arbitrary ordinals as contents for the finitely many registers and again use inferior limit rules to allow for computations of ordinal length.

Theorem 4 (P. Koepke, R. Siders [4]). *$f : Ord \to Ord$ is ORM computable iff it is $\triangle_1(L)$.*

The proof again involves the recursive computation of a bounded truth predicate. Coding stacks of ordinals as ordinal values this recursion can be implemented in the seemingly restricted setting of having only finitely many registers at one's disposal.

5 Ordinal Gödel-Herbrand Computability

Section 5 and 6 represent work in progress. Standard GÖDEL-HERBRAND computability applies substitution and replacement rules to finite systems of equations. In close analogy to the definitions of KRIPKE [6] we allow assignments of ordinal values and obtain the notion of ordinal GÖDEL-HERBRAND computability.

Conjecture 1. *$f : Ord \to Ord$* is ordinal GÖDEL-HERBRAND computabable iff it is $\triangle_1(L)$.

6 Ordinal λ-Calculus

Ongoing joint work with B. Seyfferth suggests the possibility to extend the λ-calculus to transfinite strings. The generalisations are motivated by the aim to make λ-computability as strong as the other notions of ordinal computability.

Conjecture 2. *$f : Ord \to Ord$* is ordinal λ-definable iff it is $\triangle_1(L)$.

7 Conclusion

Summarising the above results we formulate an ordinal version of Theorem 1:

Corollary 1. *For $f : Ord \to Ord$ the following are equivalent:*

1. *f is ordinal recursive.*
2. *f is $\triangle_1(L)$.*
3. *f is OTM computable.*
4. *f is ORM computable.*

The conjectures of Section 5 and 6 would complete the picture. The models presented in the previous sections give different characterisations of effectively computable ordinal functions. Their equivalence suggests to accept the following analogue of the classical CHURCH-TURING thesis:

Thesis 1. *Every effectively computable ordinal function is ordinal recursive.*

References

[1] Keith Devlin. *Constructibility*. Perspectives in Mathematical Logic, Berlin, 1984.
[2] Ronald B. Jensen and Carol Karp. *Primitive recursive set functions*. In: Axiomatic Set Theory, Proceedings of Symposia in Pure Mathematics, Volume XIII, Part I, American Mathematical Society, Providence, Rhode Island, 1971, 143-176.
[3] Peter Koepke. *Turing computations on ordinals*. The Bulletin of Symbolic Logic 11 (2005), 377-397.
[4] Peter Koepke and Ryan Siders. *Register computations on ordinals*. Archive for Mathematical Logic 47 (2008), 529-548.
[5] Peter Koepke and Benjamin Seyfferth. *Ordinal Machines and Admissible Recursion Theory*. Annals of Pure and Applied Logic, Volume 160, Issue 3, September 2009, Computation and Logic in the Real World: CiE 2007, 310-318.
[6] Saul Kripke. *Transfinite recursion on admissible ordinals I,II (abstracts)*. Journal of Symbolic Logic 29 (1964), 161-162.
[7] Piergiorgio Odifreddi. *Classical Recursion Theory*. Studies in Logic and the Foundations of Mathematics, Volume 125, North-Holland, 1992.

Hardness of Instance Compression and Its Applications*

Lance Fortnow**

Northwestern University

Bodlaender, Downey, Fellows and Hermelin [2] and Harnik and Naor [3] raise the following question which has relevance to a wide variety of areas, including parameterized complexity, cryptography, probabilistically checkable proofs and structural complexity. This question asks whether the OR-SAT problem (given a list of formulae, is at least one satisfiable) is compressible.

Question 1. Is there a function f that, given as input m Boolean formula $\phi_1, \ldots, \phi_m$ where each ϕ_i has length at most n (possibly much less than m), has the following properties?

- f is computable in time polynomial in m and n,
- $f(\phi_1, \ldots, \phi_m)$ is satisfiable if and only if at least one of the ϕ_i are satisfiable, and
- $|f(\phi_1, \ldots, \phi_m)|$ is bounded by a polynomial in n.

Fortnow and Santhanam [1] essentially settle this question in the negative by showing that a positive answer to Question 1 implies that the polynomial-time hierarchy collapses. They actually show the following stronger statement, in which f is allowed to map to an arbitrary set.

Theorem 1. *If* NP *is not contained in* coNP/poly *then there is no set A and function f such that given m Boolean formula $\phi_1, \ldots, \phi_m$ where each ϕ_i has length at most n, f has the following properties*

- *f is computable in time polynomial in m and n,*
- *$f(\phi_1, \ldots, \phi_m) \in A$ if and only if at least one of the ϕ_i are satisfiable, and*
- *$|f(\phi_1, \ldots, \phi_m)|$ is bounded by a polynomial in n.*

Bodlaender et al. [2] arrive at Question 1 from the perspective of parameterized complexity. Parameterized complexity asks about the complexity of NP problems based on some inherent parameter, like the number of variables in a formula or clique size in a graph. In parameterized complexity, feasibility is identified with *fixed-parameter tractability*. A problem is fixed-parameter tractable (FPT) if it has an algorithm running in time $f(k)n^{O(1)}$ where n is the input size and $f(k)$ is an arbitrary function of the parameter k. One technique to show fixed-parameter tractability is *kernelization*, where one reduces the solution of the given instance to the solution of an instance of size depending only on the parameter; this new instance is called a "problem kernel". Chen et al. [4] show that a problem is FPT if and only if it has a kernelization. However, in general, the kernel can be arbitrarily long as a function of the parameter.

It is a fundamental question in parameterized complexity as to which problems have polynomial-size kernels [5, 6]. Bodlaender et al. [2] develop a theory of polynomial kernelizability, and define a notion of *strong distillation* which is useful in this context. A problem L has a strong distillation function if there is a polynomial-time computable function f that takes inputs $x_1, \ldots, x_m$ and outputs a y with $|y|$ bounded by a polynomial in $\max_i |x_i|$, such that y is in L if and only if at least one of the x_i's are in L. Question 1 is equivalent to asking if SAT has a strong distillation. Bodlaender et al. conjectured that the answer to Question 1 is negative, and under that conjecture showed that the parameterized versions of several NP-complete problems do not have polynomial kernelizations, including k-Path and k-Cycle. Theorem 1 confirms the conjecture of Bodlaender et al. modulo a widely-believed complexity-theoretic assumption, a rare connection between parameterized complexity and a traditional complexity-theoretic hypothesis.

Harnik and Naor [3] arrived at essentially Question 1 with a very different motivation, *cryptographic* in nature. An NP language L is *instance compressible* if there is some polynomial-time computable function f and a set A in NP such that x is in L if and only if $(f(x), 1^{|x|})$ is in A, and $|f(x)|$ is bounded by a polynomial in the length of a *witness* for x.[1] They showed that if the Satisfiability problem is compressible

* Much of this extended abstract draws from the paper by Fortnow and Santhanam [1].

** Supported in part by NSF grants CCF-0829754 and DMS-0652521.

[1] Harnik and Naor actually allow $|f(x)|$ to be bounded by a polynomial in both the length of the witness and $\log |x|$ though Fortnow and Santhanam [1] observe this variation is actually equivalent to the above formulation.

then collision resistant hash functions can be constructed from one-way functions. If an even stronger compressibility assumption holds, then oblivious transfer protocols can be constructed from one-way functions. This would imply that public-key cryptography can be based on one-way functions, solving one of the outstanding open problems in theoretical cryptography.

The cryptographic reductions of Harnik and Naor also follow from a weaker assumption than compressibility of SAT, namely the compressibility of the OR-SAT problem. In the OR-SAT problem, a list of m formulae $\phi_1 \dots \phi_m$ each of size at most n is given as input, and a "yes" instance is one in which at least one of these formulae is satisfiable. Note that the size of a witness for OR-SAT is bounded above by n, hence compressibility of OR-SAT implies a positive answer to Question 1. Harnik and Naor describe a hierarchy of problems including Satisfiability, Clique, Dominating Set and Integer Programming, the compression of any of which would imply the compression of the OR-SAT problem. Thus Theorem 1 shows that none of these problems are compressible unless the polynomial-time hierarchy collapses. From a cryptographic point of view, this result indicates that the approach of Harnik and Naor may not be viable in its current form.

Theorem 1 is directly relevant to the question of whether there are succinct PCPs for NP, which has been raised recently by Kalai and Raz [7]. A succinct PCP for an NP language L is a probabilistically checkable proof for L where the size of the proof is polynomial in the witness size n rather than in the instance size m. Current proofs of the PCP theorem [8–10] do not yield such PCPs. Kalai and Raz state that the existence of succinct PCPs "would have important applications in complexity theory and cryptography, while a negative result would be extremely interesting from a theoretical point of view". Fortnow and Santhanam [1] show such a negative result: unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ and the Polynomial Hierarchy collapses, SAT does not have succinct PCPs, nor do problems like Clique and DominatingSet. On the other hand, polynomially kernelizable problems such as VertexCover do have succinct PCPs.

Chen, Flum and Müller [11] give further applications to kernelization.

Buhrman and Hitchcock [12] use Theorem 1 to show implications of NP-complete problems reducing to subexponential-size sets. Mahaney [13] showed that if there are NP-complete sparse sets (polynomial number of strings at every length) then P=NP. Karp and Lipton [14] show that if NP has a Turing-reduction to sparse sets than NP is in $\mathsf{P/poly}$. But we had no known polynomial-time consequences of reductions to larger sets. Buhrman and Hitchcock use the proof of Theorem 1 to show that there are no NP-complete sets of size $2^{n^{o(1)}}$ unless NP is in $\mathsf{coNP/poly}$. More generally they show that if NP is not in $\mathsf{coNP/poly}$ then for every set A of size $2^{n^{o(1)}}$, there is no reduction from Satisfiability to A using $n^{1-\epsilon}$ adaptive queries to A for any fixed $\epsilon > 0$.

Dell and van Melkebeek [15] generalize Theorem 1 and show limitations on getting even small improvements on certain NP-complete problems. For example they show that one cannot compress k-CNF satisfiability or vertex cover on k-uniform hypergraphs to inputs of length $n^{k-\epsilon}$ for any $\epsilon > 0$.

Proof. (Theorem 1) Let ϕ be any formula of size at most m consisting of the disjunction of formulae each of size at most n. By the assumption on compressibility of OR-SAT, there is a language A and a function f computable in deterministic time $poly(m)$ such that $|f(\phi, 1^n)| \leqslant O(poly(n, \log(m)))$, and ϕ is satisfiable iff $f(\phi, 1^n) \in A$. Let c be a constant such that the length of compressed instances on OR-SAT formulae of size at most m and parameter at most n is at most $k = (n + \log(m))^c$.

Now let S be the set of unsatisfiable formulae of size at most n and T be the set of strings in $\bar{A}$ of length at most k. The function f induces a map $g : S^{m/n} \to T$, since a tuple of m/n formulae of size n can be represented in size m in a natural encoding scheme, and the correctness of the reduction implies that a disjunction of m/n unsatisfiable formulae maps to a string in $\bar{A}$ of length at most k.

Our strategy will be as follows: we will attempt to find a $poly(n)$ size set C of strings in T, such that any formula in S is contained in at least one tuple that maps to a string in C under the mapping g. If such a set C exists, then we have a *proof with advice* of unsatisfiability of a formula z of size n, by guessing a tuple of m/n formulae of size at most n such that z belongs to the tuple, and then checking if the tuple maps to a string in C. The check whether the tuple maps to a string in C can be done with polynomial advice, by enumerating the strings in C in the advice string. Any unsatisfiable formula will have such a proof with advice, just by the definition of C. Conversely, any tuple containing a satisfiable formula will map to a string in A and hence to a string in $\bar{C}$, implying that no satisfiable formula will have such a proof. If $m = poly(n)$, then the proof is polynomial-size, and since the advice is polynomial-size as well by assumption on C, we get that $\mathsf{coNP} \subseteq \mathsf{NP}/poly$.

Thus the proof reduces to showing the existence of a set C with the desired properties. The proof is via a purely combinatorial argument. We employ a greedy strategy, trying to "cover" as many strings in S as possible with each string we pick in C. We prove that such a greedy strategy terminates after picking polynomially many strings.

We pick the set C in stages, with one string picked in each stage. Let C_i be the set of strings picked at or before stage i, $|C_i| = i$. Let S_i denote the set of strings y in S, such that y is not part of a tuple that maps to a string in C_i under g. Let $X = S^{m/n}$, and $X_i \subseteq X$ be the set of tuples that do not belong to the pre-image set of S_i (under the mapping g).

At stage 0, C_i is the empty set, $S_i = S$ and $X_i = X$. We proceed iteratively as follows. If S_i is empty, we stop. Otherwise, at stage i, we pick the string in T with the maximum number of pre-images in X_{i-1}, and add it to C_i.

We show that if m is picked appropriately as a function of n, then this process concludes within $poly(n)$ stages. It is enough to show that the size of S_i decreases by at least a constant factor in each stage. Since $|S| \leqslant 2^{n+1}$, this implies that the process concludes after $O(n)$ stages.

Now we analyze the decrease in the size of S_i. By the pigeonhole principle, at least $|X_{i-1}|/2^{k+1}$ tuples are in $X_{i-1} - X_i$, i.e., are pre-images of the newest string in C_i. This implies that at least $|X_{i-1}|^{n/m}/2^{kn/m}$ elements are in $S_{i-1} - S_i$, since the set of all m/n-tuples with elements in $S_{i-1} - S_i$ is contained in $X_{i-1} - X_i$ and has cardinality $|S_{i-1} - S_i|^{m/n}$. But we have $|X_{i-1}|^{n/m} \geqslant |S_{i-1}|$, since the set of m/n-tuples of elements in S_{i-1} is contained in X_{i-1}. Hence $|S_{i-1} - S_i| \geqslant |S_{i-1}|/2^{kn/m}$. Since $k \leqslant (\log(m) + n)^c$ for some constant c, we can pick a constant $c' > c$ large enough so that $kn < m$ when $m = n^{c'}$. For this choice of m, we have that $|S_{i-1} - S_i| \geqslant |S_{i-1}|/2$, and therefore that $|S_i| \leqslant |S_{i-1}|/2$.

Thus, for this choice of m, we have that the set C has size $O(n)$ and that m is polynomially bounded in n. By the argument given earlier, this gives polynomial-sized proofs with polynomial advice for unsatisfiable formulae, and implies that $\mathsf{coNP} \subseteq \mathsf{NP}/poly$. From a result of Yap [16], it follows that PH collapses to the third level. □

Two open questions: The first is whether some negative results can be shown under a standard assumption for the probabilistic or closely-related average-case version of compression. Such results would have relevance to parametric hardness of approximation, and provide further evidence for the inviability of certain approaches to cryptographic constructions. The second is the general question of characterizing for which functions f, the compressibility of f-SAT implies collapse of PH. Here f-SAT is the natural generalization of the OR-SAT problem to Boolean functions other than OR. This question basically reduces to the question of whether AND-SAT is compressible, since the compression of f-SAT for non-monotone f directly implies $\mathsf{NP} \subseteq \mathsf{coNP}/\text{poly}$, and every monotone f that depends on all its m variables embeds either a $\sqrt{m}$ sized OR or AND. Thus if we can show that (assuming NP not in co-NP/poly) AND-SAT is not compressible, then under that same assumption f-SAT is not compressible for any f that has no useless variables.

References

1. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: Proceedings of the 40th ACM Symposium on the Theory of Computing. ACM, New York (2008) 133–142
2. Bodlaender, H., Downey, R., Fellows, M., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences **75**(8) (December 2009) 423–434
3. Harnik, D., Naor, M.: On the compressibility of NP instances and cryptographic applications. In: Proceedings if the 47th Annual IEEE Symposium on Foundations of Computer Science. (2006) 719–728
4. Cai, L., Chen, J., Downey, R., Fellows, M.: Advice classes of parameterized tractability. Annals of Pure and Applied logic **84**(1) (1997) 119–138
5. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
6. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. ACM SIGACT News **38**(1) (2007) 31–45
7. Kalai, Y.T., Raz, R.: Interactive PCP. Electronic Colloquium on Computational Complexity **7**(31) (2007)
8. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. Journal of the ACM **45**(1) (1998) 70–122
9. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM **45**(3) (1998) 501–555
10. Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM **54**(3) (2007)

11. Chen, Y., Flum, J., Müller, M.: Lower bounds for kernelization. In: Proceedings of the 5th Computability in Europe, Mathematical Theory and Computational Practice. Volume 5635 of Lecture Notes in Computer Science. Springer (2009) 118–28
12. Buhrman, H., Hitchcock, J.: NP-complete sets are exponentially dense unless NP $\subseteq$ co-NP/poly. In: Proceedings of 23rd Annual IEEE Conference on Computational Complexity. (2008) 1–7
13. Mahaney, S.: Sparse complete sets for NP: Solution of a conjecture of berman and hartmanis. Journal of Computer and System Sciences **25**(2) (1982) 130–143
14. Karp, R., Lipton, R.: Turing machines that take advice. L'Enseignement Mathématique **28**(2) (1982) 191–209
15. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. Manuscript (2009)
16. Yap, C.K.: Some consequences of non-uniform conditions on uniform classes. Theoretical Computer Science **26** (1983) 287–300

Efficiently Inverting the L^2-Invariant through Stability Theory.

Cameron Donnay Hill

University of California, Berkeley
Department of Mathematics
970 Evans Hall #3840, Berkeley, CA 94720-3840 USA
chill@math.berkeley.edu

Abstract. In [4], an efficient algorithm for inverting the L^2-invariant is presented, which amounts to a reduction to resolving a certain combinatorial-design problem, the so-called magic square. While elegant, this approach does not lend itself to generalization to greater numbers of variables. In this paper, I approach the L^2-canonization problem by explicitly seeking to exploit the well-behaved geometry of L^2-theories. Specifically, I present an algorithm to invert the L^2-invariant which relies on the fact that all complete L^2-theories are stable with "trivial" forking-dependence.

1 Introduction

Classically, forking-dependence is a means for developing a geometric structure theory for the class of models of a given complete first-order theory. A structure theory generally amounts to a recipe for building models of the theory from a reasonably succinct parametric specification – e.g. a system of cardinal invariants – and the parameters are commonly dimensions of certain "uncomplicated building blocks." Thus, we can think of a "structure theory" as the infinitary analog of a model-building algorithm. Analysis of forking-dependence is a pre-requisite of understanding the "uncomplicated building blocks" and their interactions, and in some cases, this analysis comprises essentially all of the interesting/hard work to be done.

In [4], an efficient algorithm for inverting the L^2-invariant is presented, which amounts to a reduction to resolving a certain combinatorial-design problem, the so-called magic square. While elegant, this approach does not lend itself to generalization to greater numbers of variables – a resolvable higher-dimensional magic "square" problem seems rather difficult to characterize in this case. By contrast, the algorithm I have in mind is an adaptation of early work on ω-stable and superstable theories. Ultimately, the efficient construction of models is made possible by the availability of a uniform coordinatization of (some of) the models over the geometries of strongly-minimal types. It then suffices to show how to recover such a coordinatization efficiently from the L^2-invariant – for this, I draw an analogy with one of the several characterizations of forking in stable theories, the machinery of heirs and the fundamental order on types. In the L^2-scenario, one can get away with defining the fundamental order exclusively for a certain small collection of formulas in such a way that (i) a complete analysis of forking for a given L^2-theory can be performed efficiently using only these formulas, and (ii) a coordinatization can be recovered efficiently from the input data $\mathbf{I}^2(\mathcal{M})$.

My approach does not yield a more efficient algorithm for inverting the L^2-invariant, but it does shed some light on the underlying structural properties of L^2 that make efficient inversion possible. In particular, it is possible to generalize the analysis to invert the L^k-invariant, $k \geq 2$, for certain well-behaved classes of finite structures – namely, those finite structures $\mathcal{M}$ such that forking-dependence is trivial in $Th^k(\mathcal{M})$ and in which forking can be analyzed uniformly in terms of a fixed class of "simplicial" configurations of L^k-k-types over $\emptyset$. Going forward, this suggests that a parametrized-complexity analysis of the L^k-canonization problem for well-behaved class of structures will be fruitful. It also suggests that more extensive investigation of the notion of "triviality" is warranted for L^k-theories. (This generalization is pursued as a coda in the full paper, but not in this extended abstract.)

1.1 Preliminaries and notation

Throughout, we assume that T is the complete L^2-theory of a finite ρ-structure $\mathcal{M}_0$, where ρ is a finite relational language such that $ari(R) \leq 2$ for all $R \in \rho$. For brevity, set $S = S_2^2(T)$. We consider a structure

$$\mathbf{G} = \mathbf{G}^2(\mathcal{M}_0) = (S, <, E, R_\tau, \Delta)$$

where

- $<$ is a linear order of the universe
- $(p,q) \in E \iff (\forall a_1, a_2 \in M_0)[\mathcal{M}_0 \models p(a_1,a_2) \implies (\exists b \in M_0)(\mathcal{M}_0 \models q(a_1,b)]$
- $(\forall p)(\exists! q)(R_\tau(p,q))$; thus, we write $q = p^\tau$ to mean that $R_\tau(p,q)$ holds.
- $(p,q) \in R_\tau \iff (\forall a_1, a_2 \in M_0)[\mathcal{M}_0 \models p(a_1,a_2) \iff \mathcal{M}_0 \models q(a_2,a_1)]$
- $p \in \Delta \iff (\forall a_1, a_2 \in M_0)[\mathcal{M}_0 \models p(a_1,a_2) \implies a_1 = a_2]$

The structure $\mathbf{G}^2(\mathcal{M}_0)$ is called the game-tableau of $\mathcal{M}_0$. Given new unary predicates, $\{P_\theta\}_\theta$, where θ ranges over complete quantifier-free 2-types, the natural expansion $\mathbf{I}^2(\mathcal{M}_0)$ of $\mathbf{G}$, in which $P_\theta = \{p \in S : p \models \theta\}$, is often called the complete L^2-invariant of $\mathcal{M}_0$. That $Th^2(\mathcal{M}_0) = Th^2(\mathcal{M}_1)$ if and only if $\mathbf{I}^2(\mathcal{M}_0) = \mathbf{I}^2(\mathcal{M}_1)$ was established in [1]. Thus, $\mathbf{G}^2(\mathcal{M}_0)$ and $\mathbf{I}^2(\mathcal{M}_0)$ are actually invariants of $T = Th^2(\mathcal{M}_0)$, so I am justified in writing $\mathbf{G}^2(T)$ and $\mathbf{I}^2(T)$.

If A is a non-empty set, then a map $h : A \times A \to S$ is called a *realization*[1] if for all $a_1, a_2 \in A$,

- $(h(a_1,a_2), q) \in E \iff (\exists b \in A)(h(a_1,b) = q)$
- $h(a_1,a_2)^\tau = h(a_2,a_1)$
- $h(a_1,a_2) \in \Delta \iff a_1 = a_2$

Clearly, the map $h_{\mathcal{M}_0} : M_0 \times M_0 \to S$, $h_{\mathcal{M}_0}(a_1,a_2) = tp^2(a_1a_2; \mathcal{M}_0)$ is a realization, and it is not difficult to verify that if $h : A \times A \to S$ is a realization, then the structure $\mathcal{A}_h$ on A given by $(a_1,a_2) \in R^{\mathcal{A}_h} \iff R(x_1,x_2) \in \theta \in h(a_1,a_2)$ (again, θ ranges over complete quantifier-free 2-types), is a model of T. In particular, it's easy to see that $\mathcal{M}_0 = (M_0)_{h_{\mathcal{M}_0}}$. If $h : M \times M \to S$ is a realization and $C \subseteq M$, then $h \restriction (C \times C)$ corresponds to the (quantifier-free) diagram $\mathrm{diag}(C; \mathcal{M})$ of $(C; \mathcal{M})$ in the language obtained by naming each L^2-2-type as a binary predicate; in the specification of the algorithm, I have found it more convenient to speak of diagrams rather than realizations.

Again following [4], the set $\mathfrak{o}_T = \{q(x,x) \in S_1^2(T) : T \models \exists_{\leq 1} x\, q(x,x)\}$ can be read off from $\mathbf{G}^2(T)$, and T has a unique model just in case $\mathfrak{o}_T = S_1^2(T)$. For brevity, I will assume that T has arbitrarily large finite models – the alternative case being "easy." I will also identify $\mathfrak{o}_T$ with the diagram comprising the union $\{q(c_q) : q \in \mathfrak{o}_T\}$ and $\{tp^2(c_q, c_{q'}) : q, q' \in \mathfrak{o}_T\}$, and further I will assume that for every model $\mathcal{M}$ of T, $c_q \in M$ and $\mathcal{M} \models q(c_q)$ for all $q \in \mathfrak{o}_T$ – that is, $\mathcal{M}$ contains the diagram $\mathfrak{o}_T$. Finally, the following proposition is extremely useful in the stability-theoretic analysis:

Proposition 1 (Amalgamation). *Suppose $h : A \times A \to S$ and $h' : A' \times A' \to S$ are realizations such that $h \restriction (A_0 \times A_0) = h' \restriction (A_0 \times A_0)$, where $\mathfrak{o}_T \subseteq A_0 = A \cap A'$. Then there is a realization $H : B \times B \to S$ such that $h \cup h' \subseteq H$.*

2 Non-forking independence

Following [2], [3], I define forking in terms of an adaptation of Morley rank (and of the local ranks; these notions essentially coincide in the context of finite-variable logics on finite structures). Suppose $\mathcal{M}_0$ is a model, $A \subseteq M_0$ and π is a partial type over A.

- $R(\pi) \geq 0$ just in case there is a model $\mathcal{M}$ containing $(dom(\pi); \mathcal{M}_0)$ in which p is realized.
- $R(\pi) \geq n+1$ iff for every $m < \omega$, there are a model $\mathcal{M}$ containing $(dom(\pi); \mathcal{M}_0)$, $dom(\pi) \subseteq B \subseteq M$, and types $q_0, ..., q_{m-1} \in S^2(B; \mathcal{M})$ such that
  - $\pi \subseteq q_i$ and $R(q_i) \geq n$ for each $i < m$;
  - For all $i < j < m$, there are $\phi(x,y) \in S_2^2(T)$ and $b \in B$ such that $\phi(x,b) \in q_i$ and $\neg\phi(x,b) \in q_j$.
- $R(\pi) = \infty$ just in case $R(\pi) \geq n$ for all $n < \omega$.

Assuming $R(\pi)$ is finite, I define $\deg(\pi)$ to be the smallest number $m < \omega$ such that there are $\mathcal{M}$ containing $(dom(\pi); \mathcal{M})$, $dom(\pi) \subseteq B \subseteq M$, and pairwise (explicitly) contradictory extensions $q_0, ..., q_{m-1} \in S^2(B; \mathcal{M})$ of π such that $R(q_i) = R(\pi)$ for each $i < m$. I abbreviate $R(tp^2(\bar{b}/A; \mathcal{M}))$ by $R(\bar{b}/A)$. If $A \subseteq B \subseteq M$ and $p \in S^2(B)$, I say that p *does not fork over* A just in case $R(p) = R(p \restriction A)$ – otherwise, of course, p forks over A. As is usual, I define

$$A \underset{C}{\downarrow} B \iff R(\bar{a}/BC) = R(\bar{a}/C) \text{ for all } \bar{a} \in A^{<\omega}. \tag{1}$$

[1] This notion is due to [4]

Naturally, the relation $\downarrow$ is called "non-forking independence." In [2] and [3], the following standard facts about non-forking independence are recovered for stable finite-variable theories (with finite models). (Total triviality does not hold for all stable theories, but all L^2-theories do have this property.)

Theorem 1. *For any finite structure $\mathcal{N}$, if $T = Th^2(\mathcal{N})$, then $R(-)$ is always finite, and $\downarrow$ has the following properties:*

1. *Invariance: If $(A, B, C) \equiv^2 (A', B', C')$, then $A \downarrow_C B$ iff $A' \downarrow_{C'} B'$.*
2. *Existence: $A \downarrow_B B$.*
3. *Symmetry: $A \downarrow_C B$ iff $B \downarrow_C A$.*
4. *Transitivity: $A \downarrow_C B_1 B_2$ iff $A \downarrow_C B_1$ and $A \downarrow_{CB_1} B_2$.*
5. *Extension: If $A \downarrow_C B$ and $B \subseteq B_1$, then $A' \downarrow_C B_1$ for some $A' \equiv^2_{BC} A$.*
6. *Total triviality: If $A \not\downarrow_C B_1 B_2$, then $A \not\downarrow_C B_1$ or $A \not\downarrow_C B_2$.*

2.1 Triangles

A *triangle* is simply a set $\{\phi_1(x, y_1), \phi_2(x, y_2), \phi_3(y_1, y_2)\}$, where x, y_1, y_2 are pairwise distinct variables and $\phi_1, \phi_2, \phi_3 \in S_2^2(T)$, such that (i) for some $\mathcal{M} \models T$, there are $b, a_1, a_2 \in M$ such that $\mathcal{M}$ satisfies $\phi_1(b, a_1)$, $\phi_2(b, a_2)$, and $\phi_3(a_1, a_2)$, and (ii) ϕ_3 induces the same L^2-1-type on both y_1 and y_2. Let $\mathrm{Tri}(T)$ be the set of triangles with respect to T. We sometimes identify $\{\phi_1(x, y_1), \phi_2(x, y_2), \phi_3(y_1, y_2)\}$ with the formula $\phi_1(x, y_1) \wedge \phi_2(x, y_2) \wedge \phi_3(y_1, y_2)$, or with the expression $(\phi)(x, y_1, y_2)$, suppressing the subscripts.

Now, suppose $\mathcal{M} \models T$, $A \subseteq M$, and $p(x) \in S_1^2(A; \mathcal{M})$. Then

$$\mathrm{Tri}(p) = \{(\phi)(x, y_1, y_2) \in \mathrm{Tri}(T) : (\exists a_1, a_2 \in A \cup \mathfrak{o}_T)(\{\phi_1(x, a_1), \phi_2(x, a_2), \phi_3(a_1, a_2)\} \subseteq p)\} \,. \quad (2)$$

I define the fundamental (pre-)order, and the associated equivalence relation, as follows: for any pair of types p, q, not necessarily over the same domain,

$$p \leq q \iff \mathrm{Tri}(p) \supseteq \mathrm{Tri}(q) \;;\; p \sim q \iff p \leq q \wedge q \leq p \iff \mathrm{Tri}(p) = \mathrm{Tri}(q) \,. \quad (3)$$

If $\mathcal{M}$ is a model, $A \subseteq M$ and $p \in S^2(A)$, I define $\langle p \rangle$ to be the set of types $q \in S^2(N)$ such that $p \subseteq q$ and $\mathcal{N}$ is a model containing $(A; \mathcal{M})$. The following theorem is proved in much the same way as the analogous theorem of infinitary stability theory (cf. [5], proposition 17.25).

Theorem 2. *For every type p, $\langle p \rangle / \sim$ has a unique $\leq$-maximal class, denoted $\beta(p)$. Moreover, if $A \subseteq B \subseteq M$, where $\mathcal{M}$ is a model, and $p \in S_1^2(B; \mathcal{M})$, then p forks over A if and only if $\beta(p) \neq \beta(p \restriction A)$.*

3 Strongly-minimal types and the algorithm

I feel it adds a bit of clarity to the presentation to expose, first, an inadequate approach to model-building – the moving parts of this method are essentially the same as those of the correct solution, but there is less bookkeeping to contend with. Thus, we first consider the situation of a single strongly-minimal type

Abusing notation somewhat, I define a *strongly-minimal type* to be a pair $(\mathbf{C}, p(x))$, where $\mathbf{C}$ is the diagram $\mathrm{diag}(C; \mathcal{M}_1)$ and $p(x) \in S_1^2(C)$, such that $R(p) = \deg(p) = 1$. Suppose $(\mathbf{C}, p(x))$ is a strongly-minimal type. If $\mathcal{M}$ is a model containing $(C; \mathcal{M}_1)$, where $\mathbf{C} = \mathrm{diag}(C; \mathcal{M}_1)$, and $A \subseteq M$, then

$$cl_p(A) = \{b \in p(M) : R(b/AC) = 0\} = \left\{b \in p(M) : b \not\downarrow_C A\right\} \quad (4)$$

When $b \in cl_p(a)$, I say that b is a p-coordinate of a.

Lemma 1. *Suppose $(\mathbf{C}, p)$ is a strongly-minimal type, $\mathbf{C} = diag(C; \mathcal{M}_1)$. Suppose $\mathcal{M} \models T$ contains $(C; \mathcal{M}_1)$.*

1. *If $C \subseteq A \subseteq M$, then $p(x)$ has a unique non-forking extension to $(A; \mathcal{M})$. In particular, for any $a_1, a_2, b_1, b_2 \in p(M)$, if $a_2 \notin cl_p(a_1)$ and $b_2 \notin cl_p(b_1)$, then $tp^2(a_1 a_2) = tp^2(b_1 b_2)$.*
2. *$(p(M), cl)$ is pregeometry/matroid.*
3. *If $A \subseteq M$, then $cl_p(A) = \bigcup_{a \in A} cl_p(a)$.*

4. *If* $\{d_1, d_2, d_3\} \subseteq p(M)$ *is independent, then* $cl_p(d_1, d_2) \cap cl_p(d_1, d_3) = cl_p(d_1)$.
5. *If* $B \subseteq p(M)$ *is a basis of* $p(M)$ *(i.e. a maximal independent subset), then every permutation of* B *extends to an automorphism of a model* $\mathcal{M}'$ *such that* $p(M') = p(M)$.

Proposition 2. *There is an algorithm* Φ_0 *which solves the following problem:*

GIVEN: $\mathbf{G}^2(T)$; RETURN: *A strongly-minimal type* $(\mathbf{C}, p)$.

Moreover, the running time of Φ_0 *is bounded by poly*$(|\mathbf{G}^2(T)|)$.

Proof (Sketch). Starting with $C_0 = \mathfrak{o}_T$ and $p_{0,0}(x)$,..., $p_{0,t-1}(x)$, an enumeration of 1-types over $\mathfrak{o}_T$, the algorithm greedily builds $(C_{i+1}, p_{i+1,j} : j < t)$ from $(C_i, p_{i,j} : j < t)$: obtain $C_{i+1} \supsetneq C_{i+1}$ by adding realizations of each $p_{i,j}$, $j < t$, and then, using theorem 2, choose a forking extension $p_{i+1,j} \in S_1^2(C_{i+1})$ of $p_{i,j}$, $j < t$, such that $R(p_{i+1,j}) \geq 1$ if $R(p_{i,j}) > 1$ – choosing a non-forking extension $p_{i+1,j} \in S_1^2(C_{i+1})$ of $p_{i,j}$ when $R(p_{i,j}) = 1$. Then $R(p_{n,0}) = 1$ after $n \leq |\mathrm{Tri}(T)| \leq |S_2^2(T)|^3$ steps. Some additional processing is required to choose an non-forking extension $p(x)$ of $p_{0,n}(x)$ such that $\deg(p) = 1$, but I omit this for the sake of brevity.

Proposition 3. *There is an algorithm* Φ_1 *which solves the following problem:*

GIVEN: $\mathbf{G}^2(T)$, *a strongly-minimal type* $(\mathbf{C}, p)$ *and a number* $2 \leq m < \omega$;
RETURN: *The diagram* $\mathbf{D}$ *of a set* $(D; \mathcal{M}_1)$ *such that* $D = C \dot{\cup} p(M_1)$ *and* $\dim(p(M_1)) = m$, *together with an ordered basis* $(B, <)$ *of* $p(M)$.

Moreover, the running time of Φ_1 *is bounded by poly*$(|\mathbf{G}^2(T)|, |C|, m)$.

Proof (Sketch). The algorithm first invents an ordered basis $b_0, ..., b_{m-1}$, asserting $p(b_i)$ for each $i < m$ and $b_i \mathop{\smile\!\!\!\!\!\!|}_C b_j$ whenever $i < j < m$ (by part 1 of lemma 1, there is a unique 2-type over C associated with this independence statement). Next, the procedure invents elements to realize all of the dependent types in $S_2^2(Cb_0b_1)$ extending $p(x) \cup p(y)$ – thus, inventing the substructure $cl_p(b_0, b_1)$. Note that $S_2^2(Cb_0b_1)$ is (essentially) a subset of $\{q(x, y) \in S_2^2(T) : q_x = q_y = p\}$. By parts 4 and 5 of lemma 1, it then suffices to "copy" $cl_p(b_0, b_1)$ to $cl_p(b_i, b_j)$, $i \neq j$, by labeling the elements of $cl_p(b_0, b_1)$ by permutations of m. (Some care is needed in the bookkeeping here.) Further, it's only necessary to consider "short" products of transpositions, and this observation yields the bound on the running time.

From this point, the next logical step would be to simply complete the pair $((\mathbf{C}, p), p(M))$ returned by the composition $\Phi_1 \circ \Phi_0$ to a model, perhaps in a manner similar to that of proposition 3. Unfortunately, it's not obvious (to me, in any case), how to do this efficiently. The difficulty lies in the fact that $S_2^2(Cb_0b_1)$ may be exponentially large compared to $S_2^2(T)$. To deal with this, I introduce the somewhat more complicated notion of a *coordinate system*. A coordinate system is a tuple $(\mathbf{C}, p_0, ..., p_{t-1})$ (say $\mathbf{C} = \mathrm{diag}(C; \mathcal{M}_1)$) such that

- $(\mathbf{C}, p_i)$ is a strongly-minimal type for each $i < t$;
- $\{p_0, ..., p_{t-1}\}$ is $\subseteq$-maximal subject to the following *orthogonality* condition:
  For any model $\mathcal{M}$ containing $\mathbf{C}$, $C \subseteq D \subseteq M$ and any $i < j < t$, if $a \in p_i(M)$, $b \in p_j(M)$, $a \mathop{\smile\!\!\!\!\!\!|}_C D$ and $b \mathop{\smile\!\!\!\!\!\!|}_C D$, then $a \mathop{\smile\!\!\!\!\!\!|}_C b$.

The last condition allows the algorithm to ignore the C-part of a type when trying to realize it. Thus, it should suffice to keep t small, and in fact, it does. The algorithm Ψ_0 of the next proposition is largely the same as Φ_0 above.

Proposition 4. *There is an algorithm* Ψ_0 *which solves the following problem:*

GIVEN: $\mathbf{G}^2(T)$; RETURN: *A coordinate system* $(\mathbf{C}, p_0, ..., p_{t-1})$.

Moreover, the running time of Ψ_0 *is bounded by poly*$(|\mathbf{G}^2(T)|)$.

Proposition 5. *There is an algorithm* Ψ *which solves the following problem:*

GIVEN: $\mathbf{G}^2(T)$, *a coordinate system* $(\mathbf{C}, p_0, ..., p_{t-1})$ *of* T *and parameters* $2 \leq m_0, ..., m_{t-1} < \omega$.
RETURN: *A model* $\mathcal{M}$ *containing* $\mathbf{C}$ *such that* $\dim(p_i(M)) = m_i$ *for each* $i < t$.

Moreover, the running time of Φ is bounded by poly$(|\mathbf{G}^2(T)|, |C|, \overline{m})$.

Proof (Sketch). Running Φ_0 and then running Φ_1 on each instance $(\mathbf{G}^2(T), (\mathbf{C}, p_i), m_i)$, $i < t$, recover the data $p_i(M)$ – with $\dim(p_i(M)) = m_i$ – and ordered bases B_i of $p_i(M)$ for some model $\mathcal{M}$ containing $(C; \mathcal{M}_1)$. Say $B_i = \{b^i_0, ..., b^i_{m_i-1}\}$, and let $B^* = \{b^i_0, b^i_1 : i < t\}$. Subsequently, the method is a perturbation of the method in proposition 3. That is, the algorithm invents realizations of those types $q(x, y)$ in $S^2_2(CB^*)$ such that if $a_1 a_2 \models q$, then $a_j \not\perp_C b^i_0 b^i_1$, for each $j = 1, 2$ and some $i < t$ (so either b^i_0 or b^i_1 is a coordinate of a_j); it then uses the same copying-via-permutations method of Φ_1 to invent realizations having coordinates in $\{b^i_r, b^i_s\}$, $i < t$, $r < s < m$. It can then be shown that the resulting diagram is a model of the game tableau of T, and this amounts to a model of T. The fact of searching only over forking extensions yields the bound on the running time.

References

1. S. Abiteboul and V. Vianu. *Computing with first-order logic.* Journal of Computer and System Sciences, Vol. 50 (1995) pp. 309-335.
2. T. Hyttinen. *On stability in nite models.* Archive for Mathematical Logic, Vol. 39 (2000), 89-102.
3. T. Hyttinen. *Forking in finite models.* (Unpublished manuscript).
4. M. Otto. *Canonization for Two Variables and Puzzles on the Square.* Annals of Pure and Applied Logic, Vol. 85, 1997, pp. 243-282.
5. B. Poizat. *A Course in Model Theory: An Introduction to Contemporary Mathematical Logic.* Tr. by M. Klein. Universitext. Springer-Verlag, New York, 2000.

Randomness and the ergodic decomposition

Mathieu Hoyrup

LORIA
615, rue du jardin botanique
BP 239
54506 Vandoeuvre-lès-Nancy
France mathieu.hoyrup@loria.fr

Abstract. We briefly present ongoing work about Martin-Löf randomness and the ergodic decomposition theorem.

In [ML66], Martin-Löf defined the notion of an algorithmically random infinite binary sequence. The idea was to have an individual notion of a random object, an object that is acceptable as an outcome of a probabilistic process (typically, the tossing of a coin). This notion has proved successful, as it turns out that random sequences in the sense of Martin-Löf have all the usual properties that are proved to hold almost surely in classical probability theory. A particularly interesting property is typicalness in the sense of Birkhoff's ergodic theorem. This theorem embodies many probability theorems (strong law of large numbers, diophantine approximation, e.g.). Whereas the algorithmic versions of many probability theorems are straightforward to derive from their classical proofs, whether the ergodic theorem has a version for random elements has been an open problem for years, finally proved by V'yugin [V'y97] (improvements of this result, extending the class of functions for which the algorithmic version holds, have been established later in [Nan08], [HR09]). The reason for this difficulty is that the classical proof of Birkhoff's theorem is in some sense nonconstructive. In [V'y97], V'yugin gave it a precise meaning: the speed of convergence is not computable in general.

Recently, Avigad, Gerhardy and Towsner [AGT10] proved that the speed of convergence is actually computable in the ergodic case, i.e. when the system is undecomposable. As a result, the nonconstructivity of the theorem lies in the ergodic decomposition. Let us recall two ways of talking about the ergodic decomposition (see Section 1 for definitions and details):

1. Let E be the set of points x such that P_x is ergodic. E has measure one for every invariant measure.
2. For each invariant measure μ, there is a measure m supported on the ergodic measures, such that μ is the barycenter of m, i.e. for every Borel set A, $\mu(A) = \int \nu(A)\, \mathrm{d}m(\nu)$.

One can easily derive each formulation from the other, but not constructively.

We are interested in the precise extent to which the ergodic decomposition is non-constructive, or non-computable. We briefly survey a few investigations about this problem, especially from the point of view of Martin-Löf randomness.

1 The ergodic decomposition

We work with a topological dynamical system, i.e. a compact metric space X and a continuous transformation $T : X \to X$. $\mathscr{B}$ is the σ-field of Borel subsets of X. Let μ be a T-invariant Borel probability measure over X (i.e., $\mu(A) = \mu(T^{-1}A)$ for $A \in \mathscr{B}$). Birkhoff's ergodic theorem states that for every observable $f \in L^1(X, \mathscr{B}, \mu)$, the following limit exists

$$f^*(x) := \lim_{n\to\infty} \frac{1}{n} \sum_{i<n} f \circ T^i(x) \quad \text{for } \mu\text{-almost every } x \in X. \tag{1}$$

Moreover, the function f^* is integrable and $\int f^* \,\mathrm{d}\mu = \int f \,\mathrm{d}\mu$.

In general one cannot expect, for a single point x, convergence (1) for every $f \in L^1(X, \mathscr{B}, \mu)$. Nevertheless, using a separability argument it is easy to see that for μ-almost every x, convergence holds for *every* bounded continuous f. Let us denote the set of continuous functions from X to $\mathbb{R}$ by $C(X)$.

A point is called ***generic*** for T if $f^*(x)$ exists for all $f \in C(X)$ (this notion does not depend on μ). Now, given a generic point x, the functional which maps $f \in C(X)$ to $f^*(x)$ is positive and linear, so it extends to a probability measure, denoted P_x. In other words,

$$f^*(x) = \int f \, \mathrm{d}P_x$$

for every $f \in C(X)$.

From the definition of $f^*(x)$, one can prove that for every generic x, P_x is T-invariant. Now, the ergodic decomposition theorem states if μ is T-invariant then P_x is ergodic for μ-almost every x. This theorem admits (at least) two different proofs, one is an application of Choquet's theorem from convex analysis: the set of invariant measures is a compact convex set whose extreme points are the ergodic measures; hence, any invariant measure μ can be expressed as a barycenter of the ergodic measures, i.e. there is a probability measure m over the set of probability measures over X assigning full weight to the set $\mathcal{E}(T)$ of ergodic invariant measures, such that for every Borel set A,

$$\mu(A) = \int \nu(A) \, \mathrm{d}m(\nu). \tag{2}$$

2 Martin-Löf random points

A natural way of investigating the constructivity of the ergodic decomposition is the following: given a topological system with appropriate computability assumptions, given an invariant measure μ, is P_x ergodic for every μ-ML random point x?

Precisely we will assume X is a computable metric space that is compact in an effective way, $T : X \to X$ a computable map and μ a Borel probability measure over X that is T-invariant. Observe that it is a consequence of V'yugin's theorem that if x is μ-ML random then x is generic, so P_x is well-defined. It is important to note that while V'yugin's theorem was originally proved when X is the Cantor space and μ is computable, it can be extended to arbitrary computable metric spaces and arbitrary Borel probability measures [GHR], [Hoy08].

We briefly investigate the question: is P_x ergodic?

Remark 1. First, if μ is itself ergodic, then from V'yugin's theorem, $P_x = \mu$, so P_x is ergodic.

2.1 Naive strategy

Let x be a generic point and $\mathcal{F}_x$ be the class of Borel sets A such that $\mathbf{1}_A^*(x) = \mu(A)$ ($\mathbf{1}_A$ is the indicator of A and $\mathbf{1}_A^*$ is defined by (1)). All T-invariant sets $A \in \mathcal{F}_x$ have P_x-measure 0 or 1, as the whole trajectory of x lies either inside A, or outside A. If $\mathcal{F}_x$ were the whole class of Borel sets (which is almost never the case) then P_x would be automatically ergodic.

Hence this approach (which does not work) raises two questions:

1. Find a characterization of $\mathcal{F}_x$ for μ-random x. For instance, does $\mathcal{F}_x$ contain all the r.e. open sets? We know that it contains all the r.e. open sets whose μ-measure is computable (see [HR09]).
2. Which classes $\mathcal{F}$ are sufficient to characterize ergodicity? A class $\mathcal{F}$ of Borel sets characterizes ergodicity if the system is ergodic as soon as all the T-invariant sets in $\mathcal{F}$ have measure 0 or 1. In particular, can we restrict to sets that are *constructive* in some sense?

2.2 Other strategy

The following proposition is an easy consequence of classical results.

Proposition 1. *Every m-random measure is ergodic.*

Proof. It is known that the set of ergodic measures is a G_δ-set. When (X, T) is a computable system, it is a computable G_δ-set. As it has measure one, it contains all the random elements, by a result proved by Kurtz (actually, its relativized version for non-computable measures).

We prove the following result:

Theorem 1. *Assume m and μ are computable. The following are equivalent:*

1. *x is μ-random,*
2. *there exists a m-random measure ν such that x is ν-random.*

Hence, if m is computable the problem is solved: if x is a μ-random point, then there is a m-random measure ν such that x is ν-random. By the preceding proposition, ν is ergodic, so by Remark 1, $P_x = \nu$.

The problem boils down to the computability of m and μ. As μ can be simply derived from m by equality (2), it follows that μ is always computable relative to m (this can be formalized using Weirauch's type-two machines or Ko's oracle machines for instance, see [Wei00], [Ko91]). But it might be possible that a computable measure μ induces a non-computable m.

When m is not computable, a relativized version of Theorem 1 gives that if x is μ-random *relative to* m, then there exists a m-random measure ν such that x is ν-random, so as above, ν is ergodic and $\nu = P_x$. But we do not know what happens for μ-random points that are no more random when m is added as an oracle.

3 Summary

We can identify three ways of defining the computability of the ergodic decomposition:

1. Given a generic point x, is P_x computable from x? In what sense?
   - The point in V'yugin's counter-example is that the mapping $x \mapsto P_x$ is non-effective, in some sense. In particular, there is a computable observable f such that f^* is not L^1-computable.
2. Is the measure m computable?
   - In V'yugin's counter-example, even if $x \mapsto P_x$ is non-effective, the measure m is computable.
3. Given a μ-random point x, is P_x ergodic?

For the moment we only know that 1. $\Longrightarrow$ 2. $\Longrightarrow$ 3., for a suitable notion of computability of the mapping $x \mapsto P_x$.

In the introduction, we said that in V'yugin's example the ergodic decomposition is non-constructive. Actually, it is computable in the sense of 2. (and hence 3.) but not in the sense of 1. We still do not know whether the ergodic decomposition is always computable in the sense of 2. (while we conjecture it is not the case) and 3.

References

AGT10. J. Avigad, P. Gerhardy, and H. Towsner. Local stability of ergodic averages. *Trans. Amer. Math. Soc.*, 362:261–288, 2010. 39

GHR. S. Galatolo, M. Hoyrup, and C. Rojas. Effective symbolic dynamics, random points, statistical behavior, complexity and entropy. *Information and Computation*. To appear. 40

Hoy08. M. Hoyrup. *Computability, Randomness and ergodic theory on metric spaces*. PhD thesis, Université Denis-Diderot – Paris VII, 2008. 40

HR09. Mathieu Hoyrup et Cristóbal Rojas. Applications of Effective Probability Theory to Martin-Löf Randomness. *ICALP 2009*. LNCS, 5555:549-561, 2009. 39, 40

Ko91. K.-I Ko. *Complexity Theory of Real Functions*. Birkhauser Boston Inc., Cambridge, MA, USA, 1991. 41

ML66. P. Martin-Löf. The definition of random sequences. *Information and Control*, 9(6):602–619, 1966. 39

Nan08. Satyadev Nandakumar. An effective ergodic theorem and some applications. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 39–44, New York, NY, USA, 2008. ACM. 39

V'y97. V. V. V'yugin. Effective convergence in probability and an ergodic theorem for individual random sequences. *SIAM Theory of Probability and Its Applications*, 42(1):39–50, 1997. 39

Wei00. K. Weihrauch. *Computable Analysis*. Springer, Berlin, 2000. 41

An Axiomatic Approach to Barriers in Complexity

Russell Impagliazzo[1], Valentine Kabanets[2], Antonina Kolokolova[3]

[1] UC San Diego
[2] Simon Fraser University
[3] Memorial University of Newfoundland

Abstract. In late 1980s Russell Impagliazzo, Sanjeev Arora and Umesh Vazirani came up with a logic framework for formalization of a major complexity barrier, relativization of [BGS75]. That is, Arora, Impagliazzo and Vazirani defined a theory of arithmetic such that, informally, a complexity result relativizes iff it can be proved within this theory. For various reasons, this paper which we reference as [AIV92], was never published (there are drafts available on Impagliazzo's and Vazirani's webpages).

Relativizing techniques cannot resolve many major complexity questions such as P vs. NP. So do we know a type of technique that can be used to resolve such questions? In [AIV92], they argue that we do have such techniques already, and these are the techniques that make use of the local checkability of computation, Cook-Levin theorem being the prime example.

Recently another complexity barrier became the focus of attention: Aaronson and Wigderson [AW08] defined the barrier they called "algebrization", refining the notion of relativization to incorporate the use of arithmetization techniques. They show that majority of non-relativizing results can be shown to be algebrizing, and yet no algebrizing technique can resolve main complexity questions. In [IKK09] we show how to adapt the framework of [AIV92] to axiomatize the notion of algebrization barrier, and show that the theory formalizing algebrization is stronger than the relativization theory, but weaker than the local checkability theory.

In this talk, I will describe the [AIV92] logic framework for axiomatizing barriers in complexity theory, and show how it can be adapted to formalize other barriers of this type.

1 Introduction

In the mid-1970's, Baker, Gill and Solovay [BGS75] used *relativization* as a tool to argue that techniques like simulation and diagonalization cannot, by themselves, resolve the "P vs. NP" question. Intuitively, a technique relativizes if it is insensitive to the presence of oracles (thus, a result about complexity classes holds for all oracle versions of these classes). If there are oracles giving a contradictory resolution of a complexity question (e.g., $\mathsf{P}^A = \mathsf{NP}^A$, but $\mathsf{P}^B \neq \mathsf{NP}^B$), then no relativizing technique can resolve this question. This method of relativization has been brought to bear upon many other open questions in Complexity Theory, for example, P vs. PSPACE [BGS75], NP vs. EXP [Dek69,GH83,Lis86], BPP vs. NEXP [Hel86], IP vs. PSPACE [FS88], and a long list of other classes.

In an informal sense, contrary relativizations of a complexity theory statement have been viewed as a *mini-independence result*, akin to the independence results shown in mathematical logic. But what *independence* is implied by contradictory relativizations, and what are the *proof techniques* from which this independence is implied? This was made precise in [AIV92]. There the authors introduced a theory $\mathcal{RCT}$ (which stands for Relativized Complexity Theory) based on Cobham's axiomatization of polynomial-time computation [Cob64]. Roughly speaking, $\mathcal{RCT}$ has standard axioms of arithmetic (Peano axioms), and an axiomatic definition of the class of functions that is supposed to correspond to the class P. This class is defined as the closure of a class of some basic functions under composition and limited recursion (as in Cobham's paper [Cob64]), plus there is an extra axiom postulating the existence of a universal function for that class. $\mathcal{RCT}$'s view of the complexity class P is "black-box": the axioms are satisfied not only by the class P, but also by every relativized class P^O, for every oracle O. In fact, [AIV92] shows that the (standard) models of $\mathcal{RCT}$ are exactly the classes P^O, over all oracles O.[4] It follows that, for any complexity statement S about P, this statement S is true relative to every oracle A (i.e., S relativizes) iff S is provable in $\mathcal{RCT}$. On the other hand, a non-relativizing statement is precisely a statement independent of $\mathcal{RCT}$. Thus, e.g., the "P vs. NP" question is independent of $\mathcal{RCT}$.

[4] Cobham [Cob64] gets the exact characterization of P by considering the *minimal* model for his theory.

[AIV92] also shows that extending $\mathcal{RCT}$ with another axiom, which captures the "local checkability" of a Turing machine computation in the style of the Cook-Levin theorem (the computation tableau can be checked by checking that all 2×3 "windows" are correct), almost exactly characterizes the class P in the following sense: the models for the resulting theory, denoted by $\mathcal{LCT}$ (for Local Checkability) in [AIV92], are necessarily of the form P^O with $O \in \mathsf{NP} \cap \mathsf{co\text{-}NP}$. Thus, $\mathcal{LCT}$ is so strong that resolving most complexity questions in $\mathcal{LCT}$ is essentially equivalent to resolving them in the non-relativized setting.

In 1990s, a sequence of important non-relativizing results was proved using arithmetization techniques, most notably $\mathsf{IP} = \mathsf{PSPACE}$ by [LFKN92,Sha92]. Was this class of techniques sufficient to resolve questions like P vs. NP, or was there still another barrier? Fortnow [For94] have presented a way to construct a class of oracles with respect to which $\mathsf{IP} = \mathsf{PSPACE}$ and similar results hold, however he did not give an indication whether there are two such oracles giving contrary answers to questions like P vs. NP. Later, Aaronson and Wigderson [AW08] defined the "algebrization barrier": algebrizing separations and collapses of complexity classes, by comparing classes relative to one oracle to classes relative to an algebraic extension of that oracle. Using these definitions, they show that the standard collapses "algebrize" and that many of the open questions in complexity fail to "algebrize", suggesting that the arithmetization technique is close to its limits. However, in their framework it is unclear how to formalize algebrization of more complicated complexity statements than collapses or separations, and it is unclear whether the algebrizing statements are, e.g., closed under *modus ponens*. So, in particular, it is conceivable that several algebrizing premises could imply (in a relativizing way) a non-algebrizing conclusion.

In [IKK09] we provide an axiomatic framework for algebraic techniques in the style of [AIV92]. We extend their theory $\mathcal{RCT}$ with an axiom capturing the notion of arithmetization: the *Arithmetic Checkability* axiom. Intuitively, Arithmetic Checkability postulates that all NP languages have verifiers that are polynomial-time computable families of *low-degree polynomials*; a verifier for an NP language is a polynomial f (say, over the integers) such that the verifier accepts a given witness y on an input x iff $f(x, y) \neq 0$. Standard techniques (the characterization of "real world" non-deterministic Turing Machines in terms of consistent tableaux, and algebraic interpolations of Boolean functions) show that this axiom holds for unrelativized computation.

The models for the resulting theory, which we call $\mathcal{ACT}$ (for Arithmetic Checkability Theory), are, essentially, all relativized classes P^O with oracles O such that Arithmetic Checkability holds relative to this O, i.e., all NP^O languages have P^O-computable families of low-degree polynomials as verifiers. Arithmetic Checkability is implied by, yet is strictly weaker than Local Checkability, since $\mathcal{ACT}$ has models P^O for arbitrarily powerful oracles O (in particular, any oracle from Fortnow's [For94] recursive construction). Thus, $\mathcal{ACT}$ is a theory that lies between the [AIV92] theories $\mathcal{RCT}$ and $\mathcal{LCT}$. Moreover, both inclusions are proper: $\mathcal{RCT} \subsetneq \mathcal{ACT} \subsetneq \mathcal{LCT}$. That is, there are statements provable in $\mathcal{ACT}$ that can't be proved in $\mathcal{RCT}$, and there are statements provable in $\mathcal{LCT}$ that can't be proved in $\mathcal{ACT}$.

2 The axiomatic framework of [AIV92]

Consider an unrestricted theory of arithmetic (e.g., Peano arithmetic). Now, extend the language by introducing a class of function symbols (so the language is two-typed). These function symbols are allowed to appear in the induction axiom. Now, the question to ask is what can be proved about this class of function symbols if all we state about them is that they satisfy certain axioms.

The theory $\mathcal{RCT}$ is the base theory of [AIV92]. It consists of axioms of Peano Arithmetic over the language with additional function symbols, together with the statement that these function symbols satisfy Cobham's axioms without the minimality restriction, and some extra axioms. Call this class of functions FP' (*quasi*-FP in the original [AIV92]). That is, FP' includes basic functions (including the $\#$ function, defined as $2^{|x|^2}$), and is closed under composition and limited recursion on notation (defining recursively $g'(x, k)$ as a repetition of $g(x)$ for $|k|$ steps, $g'(x, k) = g(g'(x, k/2))$ for $k > 1$, $g'(x, 1) = x$). Two additional axioms are:

$$\text{Length axiom:} \quad \forall f \in \mathsf{FP}', \exists c > 0 \ |f(x)| < |x|^c$$

$$\text{Universal function:} \quad \exists U(i, t, x) \in \mathsf{FP}' \text{ such that } \forall f \in \mathsf{FP}' \exists i, c \ f(x) = U(i, 2^{|x|^c})$$

It is shown in [AIV92] that standard models of $\mathcal{RCT}$ correspond to relativized classes of polynomial time functions. Indeed, for an oracle O, FP^O can be represented as a class of polynomial time functions with an undefined symbol O, satisfying the axioms. For the other direction, construct an oracle O returning the bits of the universal function output. For this oracle O, $\mathsf{FP}' \subseteq \mathsf{FP}^O$.

To encode complexity-theoretic questions we need to talk about classes other than the polynomial-time computable functions. The approach chosen by [AIV92] is to define these classes in terms of polynomial-time computable functions rather than introducing additional function classes. Thus, the statement $\mathsf{P} = \mathsf{NP}$ (relativizing) can be phrased as $\forall c, \forall f \in \mathsf{FP}' \exists g \in \mathsf{FP}'\ g(x) = 1 \Leftrightarrow \exists y, |y| \leq |x|^c \wedge f(x,y) = 1$.

Now, to show that a complexity statement is not relativizing the standard approach is to present two different oracles for which the outcome of this statement is different (e,g, $\mathsf{P}^A = \mathsf{NP}^A$, but $\mathsf{P}^B \neq \mathsf{NP}^B$.) In this framework, this is a model-theoretic proof of an independence of such a statement from $\mathcal{RCT}$: the two oracles correspond to two standard models of $\mathcal{RCT}$. Alternatively, if a statement is provable in $\mathcal{RCT}$ then it definitely does relativize, since then it is true in all models of $\mathcal{RCT}$ and thus for all oracle worlds.

Note that this approach to characterizing polynomial time is different from the bounded arithmetic approach: here, the induction is unlimited, and all power of Peano Arithmetic can be used in the proof. However, all that is known about the functions in FP' is that they satisfy Cobham's axioms (without minimality), have a universal function in the class and have polynomially-bounded values. This is a "black-box" view of polynomial-time computation, and by [AIV92] it does not give us enough information about polynomial time to resolve main complexity-theoretic questions such as P vs NP.

3 Strengthening $\mathcal{RCT}$ with the Local Checkability Axiom

So what other information about FP' can be used? In [AIV92], Arora, Impagliazzo and Vazirani show that knowing that computation is locally checkable suffices to resolve the questions like P vs. NP. Thus, Cook-Levin theorem does not relativize in a very strong sense, and techniques using locality of computation in this manner avoid known oracle-based barriers.

Definition 1. [5] *Let* $\mathsf{PF\text{-}CHK}[poly, log]$ *be the class of languages L so that there is a polynomial $p(n)$ and a polynomial-time computable verifier $V^{x,\pi}(1^n, r)$ with random access to the input x and a proof $\pi \in \{0,1\}^{p(n)}$ so that V makes at most $O(\log n)$ queries to the input and proof, and so that $x \in L$ iff $\exists \pi \in \{0,1\}^{p(|x|)} \forall r \in [1, \ldots, p(n)]\ V^{x,\pi}(1^n, r) = 1$.*

The Local Checkability axiom *is the statement* $\mathsf{NP} = \mathsf{PF\text{-}CHK}[poly, log]$*, which we also denote by* LCT *(for Local Checkability Theorem). The theory $\mathcal{LCT}$ is $\mathcal{RCT} + LCT$. An oracle O is* consistent with LCT *if* $\mathsf{NP}^O = \mathsf{PF\text{-}CHK}[poly, log]^O$.

[AIV92] show that the oracles consistent with LCT are very weak (in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$) and resolving questions like P vs. NP with respect to such oracles will resolve them in the non-relativized setting.

4 $\mathcal{RCT}$ with Arithmetic Checkability

Relativizing techniques are "black-box", and locally checkable techniques are non-black-box in a strong sense. A class of technique that lies between them is arithmetization techniques: here, we do state that every easily computable function can be interpolated into an easily computable low-degree polynomial, but we do not assume much else.

Definition 2. *The class* $\mathsf{ALG\text{-}PF}$ *(algebraically checkable proof systems) is the class of languages L such that there is a polynomial-time computable polynomial family $\{f_n\}$ and a polynomially bounded polynomial-time computable function $m = m(n)$ so that $x = x_1 \ldots x_n \in L$ iff $\exists y_1 \ldots y_m \in \{0,1\}^m$ such that*
$f_{n+m}(x_1, \ldots, x_n, y_1, \ldots, y_m) \neq 0$.

[5] There are several other ways (equivalent and not) to define local checkability, for example, requiring varying degree of uniformity. The strongest of them precisely captures polynomial-time.

In [IKK09], two versions of this axiom (strong and weak) are defined. Here, we will just define the strong version, and mean it when talking about $\mathcal{ACT}$. The class $\mathsf{ALG\text{-}PF}^*$ *is the class of languages L such that there are a polynomially bounded polynomial-time computable function $m = m(n)$, a polynomial-time computable function family $\{g_n : \{0,1\}^n \to \{0,1\}^m\}$, and a polynomial-time computable polynomial family $\{f_n\}$ so that* (i) *if $x = x_1 \dots x_n \in L$ then $f_{n+m}(x, g_n(x)) = 1$,* (ii) *if $x = x_1 \dots x_n \notin L$ then $f_{n+m}(x, g_n(x)) = 0$, and* (iii) *for all $y \in \{0,1\}^m$, $y \neq g_n(x) \implies f(x, y) = 0$. The* strong ACT*, denoted by ACT^*, is the statement* $\mathsf{P} = \mathsf{ALG\text{-}PF}^*$*; the corresponding strong version of $\mathcal{ACT}$ is denoted $\mathcal{ACT}^*$. An oracle A is* consistent with ACT^* *if* $\mathsf{P}^A = \mathsf{ALG\text{-}PF}^{*A}$.

Clearly, all relativizing results are provable in $\mathcal{ACT}$; also, all results provable in $\mathcal{ACT}$ are provable in $\mathcal{LCT}$. It can be shown [IKK09] that most known non-relativizing results such as ones shown to be algebrizing in [AW08] are provable in (at least the strong) $\mathcal{ACT}$, and many open complexity questions such as P vs. NP, P vs. BPP, etc are independent of (even strong) $\mathcal{ACT}$. There is a known result though, $\mathsf{NEXP} = \mathsf{MIP}$, for the proof of which arithmetic checkability is insufficient (unless NEXP^O is restricted to have poly-length queries).

5 Conclusions

The Arora, Impagliazzo, Vazirani framework allows us to formalize a class of barriers in computational complexity, ones that show that techniques using only specific restricted information about computation cannot resolve many major open problems in complexity. In particular, the notion of relativization is formalized as provability in the theory $\mathcal{RCT}$ (all we know about polynomial-time functions is that they satisfy Cobham's axioms). Provability in $\mathcal{ACT}$ is meant to capture algebrizing statements, and $\mathcal{LCT}$ has as much information as there is about polynomial-time computable functions. This makes formal the intuitive notion that a complexity barrier "looks like" an independence of some logical theory.

Although the algebrization barrier has been defined fairly recently, there are already results such that $\mathsf{MIP} = \mathsf{NEXP}$ that avoid it. Thus, while a statement failing to algebrize shows a broad range of techniques that will fail to resolve it, it certainly does not mean that it is beyond the scope of all current techniques in complexity. We should use algebrization as a tool for homing in on the correct proof techniques to solve open problems, not as an alibi for failing to solve them.

References

[AIV92] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: The role of local checkability. Manuscript, 1992.

[AW08] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 731–740, 2008.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1964.

[Dek69] M. Dekhtiar. On the impossibility of eliminating exhaustive search in computing a function relative to its graph. *DAN SSSR = Soviet Math. Dokl.*, 14:1146–1148, 1969.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the European Association for Theoretical Computer Science*, 52:229–244, February 1994. Columns: Structural Complexity.

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP Languages? *Information Processing Letters*, 28:249 –251, 1988.

[GH83] I. Gasarch and S. Homer. Relativizations comparing NP and EXP. *Information and Control*, 58:88–100, 1983.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Computation*, 71(3):231 –243, 1986.

[IKK09] R. Impagliazzo, V. Kabanets, and A. Kolokolova. An Axiomatic Approach to Algebrization. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pages 695–704, 2009.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, 1992.

[Lis86] G. Lischke. Relationships between relativizations of P, NP, EL, NEL, EP and NEP. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 2:257 –270, 1986.

[Sha92] A. Shamir. IP=PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, 1992.

Lower bounds for width-restricted clause learning

Jan Johannsen

Institut für Informatik, LMU München, Munich, Germany

Introduction

In the past decades, a considerable amount of research has been devoted to the satisfiability problem for classical propositional logic (SAT). Besides its central role in computational complexity theory, programs for this problem, so-called SAT solvers, are of increasing importance for practical applications in various domains.

Most SAT solvers are based on extensions of the basic backtracking procedure known as the DLL algorithm [4]. The recursive procedure is called for a formula F in conjunctive normal form and a partial assignment α (which is empty in the outermost call). If α satisfies F, then it is returned, and if α causes a conflict, i.e., falsifies a clause in F, then the call fails. Otherwise a variable x not set by α is chosen according to some heuristic, and the procedure is called recursively twice, with α extended by $x := 1$ and by $x := 0$. If one recursive call returns a satisfying assignment, then it is returned, otherwise — if both recursive calls fail — the call fails.

Contemporary SAT solvers employ several refinements and extensions of the basic DLL algorithm. One of the most successful of these extensions is *clause learning* [8], which works roughly as follows: When the procedure encounters a conflict, then a sub-assignment of the current partial assignment is computed, that suffices to cause the conflict. This sub-assignment can then be stored in form of a new clause C added to the formula, viz. the unique largest clause falsified by it. This clause then serves to prune the search by causing conflicts in later branches containing the same partial assignment.

When clause learning is implemented, a heuristic is needed to decide which learnable clauses to keep in memory, because learning a large number of clauses leads to excessive consumption of memory, which slows the solver down rather than helping it. Many early heuristics for clause learning were such that the *width*, i.e., the number of literals, of learnable clauses was restricted, so that the solver learned only clauses whose width does not exceed a certain threshold.

Experience has shown that such heuristics are not very helpful, i.e., learning only short clauses does not significantly improve the performance of a DLL algorithm for hard formulas. We support this experience with rigorous mathematical analyses in the form of lower bound theorems.

The lower bounds are shown by proving the same lower bounds on the length of refutations in a certain resolution based propositional proof system. The relationship of this proof system to the DLL algorithm with clause learning has been established in earlier work [2].

Preliminaries

We consider resolution-based refutation systems for formulas in CNF, which are strongly related to DLL algorithms. These proof systems have two inference rules: the *weakening rule* , which allows to conclude a clause D from any clause C with $C \subseteq D$, and the *resolution rule.* which allows to infer the clause $C \vee D$ from the two clauses $C \vee x$ and $D \vee \bar{x}$, provided that the variable x does not occur in either C or D, pictorially:

$$\frac{C \vee x \qquad D \vee \bar{x}}{C \vee D}$$

We say that the variable x is *eliminated* in this inference.

It is well-known that resolution is related to the DLL algorithm by the following correspondence: the search tree produced by the run of a DLL algorithm on an unsatisfiable formula F forms a tree-like resolution refutation of F, and vice versa. In order to define proof systems that correspond to the DLL algorithm with clause learning in the same way, we define *resolution trees with lemmas* (RTL).

The post-ordering $\prec$ of an ordered binary tree is the order in which the nodes of the tree are visited by a post-order traversal, i.e., $u \prec v$ holds for nodes u, v if u is a descendant of v, or if there is a common

ancestor w of u and v such that u is a descendant of the left child of w and v is a descendant of the right child of w.

An RTL-derivation of a clause C from a CNF-formula F is an ordered binary tree, in which every node v is labeled with a clause C_v such that:

- The root is labeled with C.
- If a node v has one child u, then C_v follows from C_u by weakening.
- If a node v has two children u_1, u_2, then C_v follows from C_{u_1} and C_{u_2} by resolution.
- A leaf v is labeled by a clause D in F, or by a clause C labelling some node $u \prec v$. In the latter case we call C a *lemma*.

An RTL-derivation is an RTL(k)-derivation if every lemma C is of width $w(C) \leq k$. An RTL-refutation of F is an RTL-derivation of the empty clause from F.

A subsystem WRTI of RTL was defined by Buss et al. [2], which corresponds to a general formulation of the DLL algorithm with clause learning in the following sense: the size of a refutation of an unsatisfiable formula F in WRTI has been shown [2] to be polynomially related to the runtime of a schematic algorithm DLL-L-UP on F. This schema DLL-L-UP subsumes all clause learning strategies commonly used in practice.

It follows that if an unsatisfiable formula F can be solved by a DLL algorithm with clause learning in time t, then it has an RTL-refutation of size polynomial in t. Moreover, if the algorithm learns only clauses of width at most k, then the refutation is in RTL(k). In this work we prove lower bounds on the size of RTL(k)-refutations, which thus yield lower bounds on the runtime of DLL algorithms with width-restricted clause-learning.

The lower bounds The pigeonhole principle states that there can be no 1-to-1 mapping from a set of size $n+1$ into a set of size n. In propositional logic, the negation of this principle gives rise to an unsatisfiable set of clauses PHP_n in the variables $x_{i,j}$ for $1 \leq i \leq n+1$ and $1 \leq j \leq n$. The set PHP_n consists of the following clauses:

- $\bigvee_{1 \leq j \leq n} x_{i,j}$ for every $1 \leq i \leq n+1$.
- $\bar{x}_{i,k} \vee \bar{x}_{j,k}$ for every $1 \leq i < j \leq n+1$ and $k \leq n$.

It is well-known that the pigeonhole principle PHP_n requires exponential size resolution proofs: Haken [5] shows that every dag-like resolution refutation of PHP_n is of size $2^{\Omega(n)}$. We will show that for $k \leq n/2$, RTL(k)-refutations of PHP_n need to be asymptotically larger :

Theorem 1. *For every $k \leq n/2$, every RTLW(k)-refutation of PHP_n is of size $2^{\Omega(n \log n)}$.*

In fact, this lower bound is of the same size as the lower bound shown by Iwama and Miyazaki [6] for tree-like refutations. On the other hand, it is known [3] that there exist dag-like regular refutations of PHP_n of size $2^n \cdot n^2$. A general simulation [2] yields WRTI-refutations of PHP_n of the same size, and hence PHP_n can be solved in time $2^{O(n)}$ by a DLL algorithm with learning arbitrary long clauses, whereas our lower bound shows that any DLL algorithm that learns only clauses of size at most $n/2$ needs time $2^{\Omega(n \log n)}$.

The ordering principle expresses the fact that every finite linear ordering has a maximal element. Its negation is expressed in propositional logic by the following set of clauses Ord_n over the variables $x_{i,j}$ for $1 \leq i, j \leq n$ with $i \neq j$:

$$
\begin{array}{ll}
\bar{x}_{i,j} \vee \bar{x}_{j,i} & \text{for } 1 \leq i < j \leq n \\
x_{i,j} \vee x_{j,i} & \text{for } 1 \leq i < j \leq n \\
\bar{x}_{i,j} \vee \bar{x}_{j,k} \vee \bar{x}_{k,i} & \text{for } 1 \leq i, j, k \leq n \text{ pairwise distinct} \\
\displaystyle\bigvee_{j \in [n] \setminus \{i\}} x_{i,j} & \text{for } 1 \leq i \leq n
\end{array}
$$

The formulas Ord_n were introduced by Krishnamurthy [7] as potential hard example formulas for resolution, but short regular resolution refutations of size $O(n^3)$ for them were constructed by Stålmarck [9]. These refutations thus can be simulated by WRTI, thus Ord_n has WRTI-refutations of polynomial size. On the other hand, we show a lower bound for RTL(k)-refutations of Ord_n:

Theorem 2. *For $k < n/4$, every* RTL(k)*-refutation of* Ord_n *is of size* $2^{\Omega(n)}$.

Thus this lower bound shows the necessity to use wide lemmas to refute them efficiently.

In both these lower bounds, the hard example formulas themselves contain clauses of large width. Since it is conceivable that the necessity to learn wide clauses is merely due to the presence of these wide initial clauses, the question arose whether similar lower bounds can be shown for formulas of small width. We now answer this question by showing that lower bounds on width-restricted clause learning for small width formulas are implied by resolution width lower bounds:

Theorem 3. *If F is a d-CNF that requires resolution width w to refute, then for any k, every* RTL(k)*-refutation of F is of size at least*

$$2^{w-(k+\max\{d,k\})} \geq 2^{w-(2k+d)}.$$

We instantiate this general lower bound to prove a lower bound for RTL(k)-refutations of certain concrete formulas. The most straightforward way to obtain a formula of small width from any formula is to expand it into a 3-CNF, as described below:

For a CNF-formula F, the 3-CNF-expansion $E_3(F)$ of F is obtained as follows: for every clause $C = a_1 \vee \ldots \vee a_k$ in F of width $w(C) = k \geq 4$, introduce $k+1$ new *extension variables* $y_{C,0}, \ldots, y_{C,k}$, and replace C by the clauses:

$$y_{C,0} \qquad \bar{y}_{C,i} \vee a_i \vee y_{C,i} \quad \text{for } 1 \leq i \leq k \qquad \bar{y}_{C,k}$$

The formula $E_3(F)$ is obviously in 3-CNF and is satisfiable if and only if F is satisfiable.

Bonet and Galesi [1] show a lower bound of $n/6$ on the resolution width of the 3-CNF expansion $E_3(\mathrm{Ord}_n)$ of the ordering principle, which can be slighltly improved to $n/2$. Thus a lower bound for RTL(k)-refutations of $E_3(Ord_n)$ follows by use of the lower bound theorem, by choosing $k = n/6$ and observing that for $n \geq 18$ we get $k \geq 3$, we obtain from Theorem 3 a lower bound of $2^{n/2-2n/6} = 2^{n/6}$.

Corollary 4. *For $n \geq 18$, every* RTL($n/6$)*-refutation of $E_3(\mathrm{Ord}_n)$ is of size* $2^{n/6}$.

It follows that a DLL algorithm with clause learning requires exponential time to solve the formulas $E_3(\mathrm{Ord}_n)$ when only clauses of width $n/6$ are learned. On the other hand, from the short regular resolution refutations of Ord_n, one can construct a run of a DLL algorithm with arbitrary clause learning on $E_3(\mathrm{Ord}_n)$ in polynomial time.

This talk is partially based on joint work with Eli Ben-Sasson, Samuel R.Buss and Jan Hoffmann.

References

1. M. L. Bonet and N. Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001.
2. S. R. Buss, J. Hoffmann, and J. Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL algorithms with clause learning. *Logical Methods in Computer Science*, 4(4), 2008.
3. S. R. Buss and T. Pitassi. Resolution and the weak pigeonhole principle. In M. Nielsen and W. Thomas, editors, *Computer Science Logic, 11th International Workshop CSL '97*, pages 149–156. Springer LNCS 1414, 1998.
4. M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
5. A. Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.
6. K. Iwama and S. Miyazaki. Tree-like resolution is superpolynomially slower than dag-like resolution for the pigeonhole principle. In *Proceedings of the 10th International Symposium on Algorithms and Computation (ISAAC)*, pages 133–142, 1999.
7. B. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–274, 1985.
8. J. P. M. Silva and K. A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 220–227, 1996.
9. G. Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996.

A Characterization of Δ^1_2 Pointclasses Via Ordinal Machines

Peter Koepke and Benjamin Seyfferth

University of Bonn, Mathematical Institute
Endenicher Allee 60, D-53115 Bonn, Germany
koepke@math.uni-bonn.de, seyfferth@math.uni-bonn.de

1 Introduction

Ordinal computability studies generalized computability theory by means of classical machine models that operate on ordinals instead of natural numbers. Starting with Joel Hamkins' and Andy Lewis' Infinite Time Turing Machines (ITTM) [1], recent years have seen several of those models which provided alternate approaches and new aspects for various ideas from logic, set theory and classical areas of generalized computability theory. With ITTMs, the machine may carry out a transfinite ordinal number of steps while writing 0s and 1s on tapes of length ω. This is achieved by the addition of a limit rule that governs the behavior of the machine at limit times. The 0s and 1s on the ω-long tape are interpreted as subsets of ω (*reals*). It turns out that the sets of reals semi-decidable by these machines form a subset of Δ^1_2. Similar studies have been carried out for infinite time register machines (ITRMs) whose computable reals are exactly the reals in $L_{\omega^{CK}_{\omega}}$ [4].

Another direction of ordinal computability lifts classical computability to study not the subsets of ω, but of an arbitrary ordinal α, or even the class Ord of all ordinals. In this case, both space and time are set to that ordinal α, i.e. in the Turing context, we deal with machines that utilize a tape of length α and either stop in less than α many steps or diverge. The computation is steered by a standard Turing program and a finite number of ordinal parameters less than α. This approach unveils strong connections to Gödels universe of constructible sets and the classical work on α-recursion theory [5].

In the present paper, we aim between these two approaches by analyzing the computable sets of reals of Turing machines with Ord space *and* time but without allowing arbitrary ordinal parameters.

2 The machine model

We work with *ordinal Turing machines (OTMs)*, the machine model introduced by the first author in [3]. We briefly review the basic features, for more detail and background the reader is referred to the original paper.

An OTM uses the binary alphabet on a one-sided infinite tape whose cells are indexed by ordinal numbers. At any ordinal point in time, the machine head is located on one of these cells and the machine is in one of finitely many machine states indexed by natural numbers. Since we utilize both Ord space and time, there is no need to use multiple tapes in our definition; any fixed finite number of tapes can be simulated by interleaving the tapes into one. A typical program instruction has the form $(a, s, a', s', d) \in \{0,1\} \times \omega \times \{0,1\} \times \omega \times \{-1,1\}$ and is interpreted as the instruction *"If the symbol currently read by the machines read-write head is a and the machine is currently in state s, then overwrite a with the symbol a', change the machine state to s', and move the head according to d either to the left or to the right"*. At successor times in the course of the computation, the machine behaves like a standard Turing machine, with the following exception: If the machine head rests on a cell indexed by a limit ordinal or 0 and a *"move left"*-instruction is carried out, then the head is set to position 0. The machine accesses the transfinite by the following $\liminf$-rule:

At a limit time λ set the machine state to the $\liminf$ of the states of previous time, i.e., the least state that was assumed cofinally often in the previous steps. Similarly we set the tape content for each cell individually to the $\liminf$ of the previous cell contents; in other words, a cell contains a 0 at time λ if it contained a 0 cofinally often before λ, and it contains a 1 at time λ otherwise. It is natural to set the head position to the cell indexed by the $\liminf$ over the indices of the cells visited at previous steps in which the machine's state was the same as in the limit.

For the present purpose, we allow a real coded as an ω-long sequence of 0s and 1s written on the first ω many cells of the tape as input to an OTM.

A set of reals $A \subset 2^\omega$ is called *OTM-enumerable* iff there is a program that halts exactly for the inputs $a \in A$. The *OTM-computable* sets of reals are those whose characteristic functions are computable by an OTM.

3 Definable subsets of real numbers

We classify subsets of the real numbers with respect to the structure of quantifiers ranging over reals required for their definition.

A set $A \subset 2^\omega$ is called Σ^1_1 iff there is a (in the classical sense) recursive relation R such that:

$$x \in A \leftrightarrow \exists y \in 2^\omega \forall n \in \omega R(x \restriction n, y \restriction n)$$

The Π^1_n sets are the complements of Σ^1_n sets. I.e., a Π^1_1 set A' has a definition of the following form (where R' is some recursive relation):

$$x \in A' \leftrightarrow \forall y \in 2^\omega \exists n \in \omega R'(x \restriction n, y \restriction n)$$

We call A a Σ^1_2 set iff there is a $B \subseteq 2^\omega \times 2^\omega$ that is Π^1_n (via an analogous definition) and we have:

$$x \in A \leftrightarrow \exists y \in 2^\omega (x, y) \in B$$

Again we define the Π^1_2 sets as complements of Σ^1_2 sets. This construction of alternating projection and complementation can be carried on further to define the pointclasses Σ^1_n and Π^1_n for $n \in \omega$.

We set $\Delta^1_n = \Sigma^1_n \cap \Delta^1_n$.

4 Computing $\boldsymbol{\Delta^1_2}$

Theorem 1. *A set A of reals is OTM-enumerable iff it is Σ^1_2.*

Theorem 2. *A set A of reals is OTM-computable iff it is Δ^1_2.*

We sketch the proof of Theorem 1. Let A be OTM-enumerable, i.e., there is a Turing program P such that $a \in A$ iff P halts on input a. Now the latter condition is Σ^1_2: *There is* a real coding a halting computation on input a; to express coding a computation of an ordinal machine requires to check the wellfoundedness of the "time-axis"; this can be done by another *for all* quantifier.

Conversely, if A is Σ^1_2 by the formula ϕ, then by the Schoenfield absoluteness theorem about the absoluteness of Σ^1_2 properties we get: $a \in A$ iff $L[a] \models \phi(a)$ [2, Theorem 25.20]. In models of set theory, ϕ is uniformly equivalent to a Σ_1 formula ψ of set theory (standard result from descriptive set theory). So $a \in A$ iff $L[a] \models \psi(a)$. But the latter can be (semi-)computed by an OTM: successively build up the $L[a]$-levels on the ordinal tape and check whether $\psi(a)$ is true in the level; if yes, then stop, otherwise continue. So A is OTM-enumerable.

Theorem 2 follows immediately from Theorem 1.

It remains to be seen whether the computability characterization of Σ^1_2 and Δ^1_2 has some applications in descriptive set theory.

References

1. Joel D. Hamkins and Andy Lewis. Infinite time Turing machines. *The Journal of Symbolic Logic*, 65(2):567–604, 2000.
2. Thomas Jech. *Set Theory.* Springer Monographs in Mathematics. Springer-Verlag, Berlin, Heidelberg, the third millennium edition, revised and expanded edition, 1997, 2003.
3. Peter Koepke. Turing computations on ordinals. *The Bulletin of Symbolic Logic*, 11:377–397, 2005.
4. Peter Koepke. Ordinal computability. In Klaus Ambos-Spies, Benedikt Löwe, and Wolfgang Merkle, editors, *Mathematical Theory and Computational Practice*, volume 5635 of *Lecture Notes in Computer Science*, pages 280–289. Springer-Verlag, Berlin, Heidelberg, 2009.
5. Peter Koepke and Benjamin Seyfferth. Ordinal machines and admissible recursion theory. *Annals of Pure and Applied Logic*, 160(3):310–318, 2009. Computation and Logic in the Real World: CiE 2007.

Ordinal Register Machines and Combinatorial Principles in the Constructible Universe

Peter Koepke and Gregor Weckbecker

Mathematisches Institut, Universität Bonn, Germany,
koepke@math.uni-bonn.de, gregorw@uni-bonn.de

Abstract. Ordinal Register Machines (ORMs) are register machines which act on ordinal numbers and are allowed to run for arbitrarily many ordinal steps. In [4] Peter Koepke showed that the class of ordinal register computable sets corresponds to the class of constructible sets of ordinals. This motivates to give alternative proofs of certain facts on the constructible universe like GCH and combinatorial principles $\diamondsuit_\kappa$ for a regular cardinal κ. A Silver Machine like structure M, based on ORMs, is introduced to prove the principle (global) $\square$. Such a structure is called Ordinal Register Silver Machine. In our talk we will focus on the construction of such a structure.

1 Introduction

Ordinal Register Machines (ORMs) were introduced by Peter Koepke and Ryan Siders in [5]. They generalize finite Register Machines, so that they act on ordinals and can run for arbitrarily many ordinal steps.

A register program P is a finite list $P = P_0, \dots P_{k-1}$ of instructions acting on registers $R_0, R_1, \dots$. The instructions of the machine are:

(i) The zero instruction `Z(n)` changes the contents of R_n to 0.
(ii) The successor instruction `S(n)` increases the ordinal contained in R_n by one.
(iii) The transfer instruction `T(m,n)` sets the content of R_n to the content of R_m.
(iv) The jump instruction $P_i =$ `J(m,n,l)` is carried out as follows: If $R_n = R_m$ then the next instruction is P_l, if not the next is P_{i+1}.

A computation

$$I \,:\, \theta \to \omega,\ R \,:\, \theta \to \mathrm{Ord}^{<\omega}$$

of $P = P_0 \dots P_{k-1}$ with input $\beta_0, \dots, \beta_{l-1}$ is recursively defined by initially setting $I(0) = 0$ and $R(0)(i) = \beta_i$ for $i < l$. At successor times $\alpha = \beta + 1$ the machine carries out the steps as in the finite case. At limit times λ the machine configuration is determined by taking inferior limits

$$(\forall i < \omega)\ R(\lambda)(i) = \liminf_{\alpha \to \lambda} R(\alpha)(i)$$
$$I(\lambda) = \liminf_{\alpha \to \lambda} I(\alpha).$$

A partial function $F : \mathrm{Ord}^m \rightharpoonup \mathrm{Ord}$ is ordinal register computable, if there is a register program P and ordinals $\delta_0, \dots, \delta_{n-1}$ such that for every tuple $(\alpha_0, \dots, \alpha_{m-1}) \in \mathrm{dom}(F)$ holds

$$P : (\alpha_0, \dots, \alpha_{m-1}, \delta_0, \dots, \delta_{n-1}, 0, \dots) \mapsto F(\alpha_0, \dots, \alpha_{m-1})$$

A subset $x \subseteq \mathrm{Ord}$ is ordinal register computable, if its characteristic function χ_x is ordinal register computable.

It it possible to formalize the language of set theory by ordinals and define a recursive truth predicate $F \,:\, \mathrm{Ord} \to 2$ by

$$F(\ulcorner L_\gamma \models \varphi(x, y_1, \dots, y_{n-1})\urcorner) \text{ iff } L_\gamma \models \varphi(x, y_1, \dots, y_{n-1}).$$

This predicate is ordinal register computable. Furthermore, it codes a model of set theory. This implies

Theorem 1. *A set $x \subseteq \mathrm{Ord}$ is ordinal register computable if and only if $x \in L$.*

2 Applications to the Structure of L

As an application of theorem 1 we can give alternative proofs for certain facts on the structure of the constructible universe.

The following lemma follows from theorem 1 by a Löwenheim-Skolem type argument.

Lemma 1. *Assume that every set of ordinals is ordinal register computable. Let $\kappa \geq \omega$ be an ordinal and $x \subseteq \kappa$. Then there are ordinals $\beta_0, \ldots, \beta_{n-1} < \kappa^+$ and a register program P such that for all ordinals α $P(\alpha, \beta_0, \ldots, \beta_{n-1}) = \chi_x(\alpha)$ holds. Furthermore, κ^+ is an upper bound for the length of the computation.*

This proves the following result by Kurt Gödel (cf. [2]):

Theorem 2. *Assume that all sets of ordinals are ordinal register computable, then* GCH *holds.*

The principle $\diamondsuit_\kappa$ is a generalization of the generalized continuum hypothesis. It is applied in numerous infinite constructions like that of an ω_1-Suslin tree. The principle was introduced by Ronald B. Jensen in [3].

Definition 1. *Let κ be a regular uncountable cardinal. Then $\diamondsuit_k$ is the principle: there is a sequence $(S_\alpha \mid \alpha < \kappa)$ such that*

$$(\forall S \subseteq \kappa)\ \{\alpha < \kappa \mid S \cap \alpha = S_\alpha\}$$

is stationary in κ.

Under the assumption that all sets of ordinals are ordinal register computable, it is possible to prove the following theorem.

Theorem 3. *Assume that all sets of ordinals are ordinal register computable. Let κ be a regular uncountable cardinal. Then $\diamondsuit_\kappa$ holds.*

3 Silver Machines and Ordinal Register Machines

In [3] Ronald B. Jensen introduced the finestructural analysis of the constructible universe L. This sophisticated theory allowed him to prove strong results on L, for example the combinatorial principle (global) $\square$. In the 1970s Jack H. Silver developed Silver Machines (cf. [7]) to avoid the use of Jensen's fine structure theory for example in the proof of (global) $\square$.

A complete reference for the theory of Silver Machines is the PhD. thesis of Thomas Lloyd Richardson [6]. For an introduction into Jensen's fine structure theory and Silver Machines see the monograph of Keith J. Devlin [1]. The following definitions follow Devlin's book.

Definition 2. *An ordinal α is $*$-definable from $X \subseteq$ Ord iff there exists $\gamma, \beta_0, \ldots, \beta_{n-1} \in X$ and an $\in$-Formula φ such that α is the unique ordinal that satisfies*

$$L_\gamma \models \varphi(\alpha, \beta_0, \ldots, \beta_{n-1}).$$

Definition 3. *A structure $N = (X, <, (h_i)_{i\in\omega})$ is eligible iff*

(i) $X \subseteq$ Ord.
(ii) $<$ *is the usual ordering of the ordinals restricted to* X.
(iii) For all $i \in \omega$, $k(i) \in \omega$, h_i is a partial function from $X^{k(i)}$ into X.

If N is an eligible structure and $\lambda \in$ Ord, then we set

$$N_\lambda = (X \cap \lambda, <\restriction \lambda \times \lambda, (h_i \cap (\lambda^{k(i)} \times \lambda))_{i\in\omega})$$

Further if $A \subseteq X \cap \lambda$, then $N_\lambda[A]$ denotes the closure of A under functions of N_λ. We call $N_\lambda[A]$ the hull of A in N_λ.

If $N = (X, <, (h_i)_{i\in\omega})$ and $N' = (X', <, (h'_i)_{i\in\omega})$ are eligible structures, then we write $N \triangleleft N'$ iff $X \subseteq X'$ and for all $i \in \omega$ and $x \in X^{k(i)}$, then $h_i(x) \simeq h'_i(x)$ holds.

Definition 4. *A (standard) Silver Machine is an eligible structure*

$$M = (\mathrm{Ord}, <, (h_i)_{i\in\omega})$$

such that

(i) (Condensation Principle) If $N \triangleleft M_\lambda$, then there is an α such that $N \cong M_\alpha$.
(ii) (Finiteness Principle) For each λ there is a finite set $H \subseteq \lambda$ such that for every set $A \subseteq \lambda + 1$

$$M_{\lambda+1}[A] \subseteq M_\lambda[(A \cap \lambda) \cup H] \cup \{\lambda\}.$$

(iii) (Skolem Property) If α is $*$-definable from the set $X \subseteq \mathrm{Ord}$, then $\alpha \in M[X]$. Moreover there is an ordinal $\lambda < (\sup(X) \cup \alpha)^+$, uniformly Σ_1 definable from $X \cup \{\alpha\}$, such that $\alpha \in M_\lambda[X]$.

Definition 5. *An Ordinal Register Silver Machine is an eligible structure*

$$M = (\mathrm{Ord}, <, (h_i)_{i\in\omega})$$

such that the properties (i) and (ii) hold. (iii) is replaced with the following property

(iii') (Computable Closure Property) Let $X \subseteq \mathrm{Ord}$. If P is register program and $\beta_0, \ldots, \beta_{n-1} \in X$ and $\alpha = P(\beta_0, \ldots, \beta_{n-1})$, then $\alpha \in M[X]$.

Using the constructible truth predicate and the upper bounds in lemma 1, it is possible to connect the two notions of Silver Machine.

Theorem 4. *Let $M = (\mathrm{Ord}, <, (h_i)_{i\in\omega})$ be a Ordinal Register Silver Machine. Then M is a Silver Machine.*

For the construction of an Ordinal Register Silver Machine we fix a well-order $<^*$ of $\mathrm{Ord}^{<\omega}$ such that for $s, t \in \mathrm{Ord}^{<\omega}$ $\max(s) < \max(t)$ implies $s <^* t$. Furthermore, we fix the corresponding rank function $G : \mathrm{Ord}^{<\omega} \to \mathrm{Ord}$ defined by $G(t) = \mathrm{otp}(\{s <^* t \mid s \in \mathrm{Ord}^{<\omega}\}, <^*)$.

Using the function G, we have to formalize register programs. For $i, j, k \in \omega$ we set $\ulcorner \texttt{Z(i)} \urcorner = G(0, i)$, $\ulcorner \texttt{S(i)} \urcorner = G(1, i)$, $\ulcorner \texttt{T(i,j)} \urcorner = G(2, i, j)$ and $\ulcorner \texttt{J(i, j, k)} \urcorner = G(3, i, j, k)$. If P is a register program and S is a single instruction then the concatenation PS is formalized by $\ulcorner PS \urcorner = G(4, \ulcorner P \urcorner, \ulcorner S \urcorner)$.

The machine that is constructed needs to be able to carry out the syntactical operations on programs. For this and the handling of finite sequences of parameters, it is necessary to add the functions $G_i : \mathrm{Ord}^i \to \mathrm{Ord}$ with $G_i(\beta_0, \ldots, \beta_i) = G((\beta_0, \ldots, \beta_i))$ for $i \in \omega$, the (partial) inverse functions $S_{i,j} : \mathrm{Ord} \to \mathrm{Ord}$ with $G_i(S_{i,0}(\alpha), \ldots, S_{i,k-1}(\alpha)) = \alpha$ and the constant functions c_v for all $v \in \omega$.

With this structure it is definable that P computes α from input $\beta_0, \ldots \beta_{n-1}$ in less than δ steps. So define the function

$$E(\gamma) = \begin{cases} \alpha & \text{if } \gamma = G(\ulcorner P \urcorner, \delta, \beta_0, \ldots, \beta_{n-1}) \text{ and } P^\delta(\beta_0, \ldots, \beta_{n-1}) = \alpha \\ 0 & \text{else.} \end{cases}$$

Where $P^\delta(\beta_0, \ldots, \beta_{n-1}) = \alpha$ iff $P(\beta_0, \ldots, \beta_{n-1}) = \alpha$ and the length of the computation is less then δ.

Theorem 5. *The structure*

$$M = (\mathrm{Ord}, <, E, (G_i)_{i\in\omega}, (S_{i,j})_{i\in\omega, j\in\omega}, (c_i)_{i\in\omega})$$

is an Ordinal Register Silver Machine and hence a Silver Machine.

The function E ensures that M has the *Computable Closure Property* holds. The proof of the *Finiteness Principle* and the *Condensation Principle* uses a careful analysis of the functions of the machine.

Using an Ordinal Register Silver Machine, it is possible to prove the principle (global) $\square$.

Theorem 6. *Assume that all sets of ordinals are ordinal register computable. Let S be the class of singular limit ordinals and let A be a class of limit ordinals. Then there is a class $E \subseteq A$ such that*

- If $\kappa > \omega$ is regular and $A \cap \kappa$ is stationary in κ, then $E \cap \kappa$ is stationary in κ.
- $\square(E)$ holds, where $\square(E)$ is defined as
(i) For every $\alpha \in S$, C_α is closed unbounded in α.
(ii) For every $\alpha \in S$, $\mathrm{otp}(C_\alpha) < \alpha$.
(iii) (coherency) If $\bar{\alpha}$ is a limit point of C_α, then $\bar{\alpha}$ is singular and $C_{\bar{\alpha}} = C_\alpha \cap \bar{\alpha}$.

References

1. Keith J. Devlin, *Constructibility*, Springer-Verlag Berlin Heidelberg New York Tokyo, 1984.
2. Kurt Gödel, *The consistency of the continuum hypothesis*, vol. 3, Princeton University Press, Princeton, 1940.
3. Ronald B. Jensen, *The fine structure of the constructible hierarchy*, Annals of Mathematical Logic **4** (1972), 229–308.
4. Peter Koepke, *Computing a model of set theory*, S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526 (2005), 223–232.
5. Peter Koepke and Ryan Siders, *Computing the recursive truth predicate on ordinal register machines*, Beckmann, A., et al. (eds.) Logical approaches to computational barriers. Computer Science Report Series **7** (2006), 160–169.
6. Thomas Lloyd Richardson, *Silver machine approach to the constructible universe*, Ph.D. thesis, University of California, Berkeley, 1979.
7. Jack H. Silver, *How to eliminate the fine structure from the work of jensen*, handwritten manuscript, 197?

Shallow Circuits with High-Powered Inputs

Pascal Koiran

LIP*, École Normale Supérieure de Lyon, Université de Lyon
Department of Computer Science, University of Toronto**
Pascal.Koiran@ens-lyon.fr

1 Introduction

A polynomial identity testing algorithm must determine whether an input polynomial (given for instance by an arithmetic circuit) is identically equal to 0. Following [5], it has become increasingly clear in recent years that efficient *deterministic* algorithms for polynomial identity testing would imply strong lower bounds (the connection between arithmetic circuit lower bounds and derandomization of polynomial identity testing was foreshadowed in a 30 years old paper by Heintz and Schnorr [4]). This approach to lower bounds was advocated in particular by Agrawal [1]. In my talk I will present some further results along those lines. The present abstract will not describe these results in their maximal generality. Instead, I would like to highlight three open problems that arose along the way. These problems can be viewed as refinements of the τ-conjecture of Shub and Smale on integer roots of univariate polynomials [9, 10]. A positive answer to any of these three problems would imply a superpolynomial lower bound on the arithmetic complexity of the permanent polynomial.

According to the τ-conjecture, the number of integer roots of a univariate polynomial $f \in \mathbb{Z}[X]$ should be bounded by a polynomial function of its arithmetic circuit size (the inputs to the circuit are the constant 1, or the variable X). It was already shown in [3] that the τ-conjecture implies a lower bound for the permanent. We show that to obtain such a lower bound one does not have to bound the number of integer roots of polynomials computed by arbitrary arithmetic circuits as in the τ-conjecture: it suffices to consider a restricted class of circuits, namely, sums of products of sparse polynomials. As explained below, these circuits can be viewed as depth-4 circuits whose inputs are (possibly very high) powers of a variable X or of the constant 2. We also raise the intriguing possibility that tools from real analysis might become applicable (indeed, a bound on the number of real roots of a polynomial is a fortiori a bound on its number of integer roots). It is known that this approach cannot work for the original τ-conjecture because the number of real roots of a univariate polynomial can grow exponentially as a function of its arithmetic circuit size (Chebyshev polynomials provide such an example [10]).

Compared to [3], the main new technical ingredient is the recent depth reduction theorem due to Agrawal and Vinay [2]: any multilinear polynomial in n variables which has an arithmetic circuit of size $2^{o(n)}$ also has a depth-4 arithmetic circuit of size $2^{o(n)}$.

2 Sums of Products of Sparse Polynomials

A sums of products of sparse polynomials is an expression of the form $\sum_i \prod_j f_{ij}$ where each f_{ij} is a sparse univariate polynomial in $\mathbb{Z}[X]$. Here "sparse" means as usual that we only represent the nonzero monomials of each f_{ij}. As a result one can represent concisely polynomials of very high degree. We define the size of such an expression as the sum of the number of monomials in all the f_{ij}. Note that this measure of size does not take into account the size of the coefficients of the f_{ij}, or their degrees. These relevant parameters are taken into account in the following definition.

Definition 1. *We denote by* $\mathrm{SPS}_{s,e}$ *the set of all polynomials in* $\mathbb{Z}[X]$ *which can be represented by an expression of the form* $\sum_i \prod_j f_{ij}$ *so that:*

- *The size of the expression as defined above is at most* s.
- *Each coeffficient of each* f_{ij} *has at most* s *nonzero digits in its binary representation (as a result,* f_{ij} *can be thought of as a sparse polynomial with sparse coefficients).*

* UMR 5668 ENS Lyon, CNRS, UCBL, INRIA.
** A part of this work was done during a visit to the Fields Institute.

– *These coefficients are of absolute value at most* 2^e*, and the* f_{ij} *are of degree at most* e*.*

As explained in Section 3, a polynomial upper bound on the number of integer roots of SPS polynomials would imply a lower bound on the permanent. By "polynomial upper bound", we mean an upper bound which is polynomial in $s + \log e$. It is quite natural to insist on an upper bound polynomial in s and $\log e$: s is an arithmetic circuit size bound, and $\log e$ can also be interpreted as an arithmetic cost since each power X^α in an f_{ij} can be computed from X in $O(\log e)$ operations by repeated squaring. Likewise, each of the $\leq s$ powers of 2 occuring in a coefficient can be computed from the constant 2 in $O(\log e)$ operations. As a result, a polynomial in $\mathrm{SPS}_{s,e}$ can be evaluated from the constant 1 and the variable X in a number of arithmetic operations which is polynomial in $s + \log e$.

Remark 1. The size of a SPS polynomial as we have defined it is essentially the size of a depth three arithmetic circuit (or more precisely of a depth three arithmetic formula) computing the polynomial. In this depth three circuit each input gate carries a monomial; each addition gate at level 1 computes a f_{ij}; each multiplication gate at level 2 computes a product of the form $\prod_j f_{ij}$; and the output gate at level 3 computes the final sum.

We can further refine this representation of SPS polynomials by arithmetic circuits. Namely, instead of viewing the monomial aX^β as an atomic object which is fed to an input gate, we can decompose it as a sum of terms of the form $\pm 2^\alpha X^\beta$; and each term can be further decomposed as a product of factors of the form $\pm 2^{2^i}$ and X^{2^j}. The resulting object is a depth four circuit where each input gate carries an expression of the form $\pm 2^{2^i}$ or X^{2^j} (note the symmetry between variables and constants in this representation). This connection between depth four circuits and SPS polynomials plays a crucial role in our results.

3 Variations on the τ-conjecture

We state three increasingly stronger conjectures which would all imply that the permanent polynomial does not belong to the complexity class VP^0. This constant-free version of Valiant's class VP was studied by Malod [8] (see also [7]). Very briefly, a family (f_n) of polynomials belongs to VP^0 if can be computed by a family of constant-free arithmetic circuits of polynomial size and polynomially bounded formal degree. *Constant-free* means that 1 is the only constant allowed as an input to the ciruit. In Valiant's original setting [11, 12] arbitrary constants from the underlying field are allowed.

Conjecture 1 (τ-conjecture for SPS polynomials). There is a polynomial p such that any nonzero polynomial in $\mathrm{SPS}_{s,e}$ has at most $p(s + \log e)$ integer roots.

We show in the full paper that this conjecture implies that the permanent polynomial does not belong to VP^0. Conjecture 1 follows from the τ-conjecture of Shub and Smale on integer roots of univariate polynomials [9, 10] since polynomials in $\mathrm{SPS}_{s,e}$ can be evaluated by constant-free arithmetic circuits of size polynomial in s and $\log e$. It was already shown in [3] that the τ-conjecture implies a lower bound for the permanent. The point of Conjecture 1 is that to obtain such a lower bound we no longer have to bound the number of integer roots of arbitrary arithmetic circuits: we need only do this for sums of products of sparse polynomials. This looks like a much more manageable class of circuits, but the question is of course still wide open. As announced in the introduction, another related benefit of SPS polynomials in this context is that techniques from real analysis might become applicable. Before explaining this in more detail we formulate a somewhat stronger conjecture. The idea is that the parameter e in Conjecture 1 as well as the sparsity hypothesis on the integer coefficients might be irrelevant. This leads to:

Conjecture 2 (τ-conjecture for SPS polynomials, strong form). Consider a polynomial of the form

$$f(X) = \sum_{i=1}^{k} \prod_{j=1}^{m} f_{ij}(X),$$

where each $f_{ij} \in \mathbb{Z}[X]$ has at most t monomials. The number of integer roots of f is bounded by a polynomial function of kmt.

Note that the size of f as defined at the beginning of Section 2 is upper bounded by kmt. Finally, we formulate an even stronger conjecture.

Conjecture 3 (real τ-conjecture). Consider a polynomial of the form

$$f(X) = \sum_{i=1}^{k} \prod_{j=1}^{m} f_{ij}(X),$$

where each $f_{ij} \in \mathbb{R}[X]$ has at most t monomials. The number of real roots of f is bounded by a polynomial function of kmt.

One could also formulate a weak version of the real τ-conjecture where the parameters s and e would play the same role as in Conjecture 1.

At present there isn't a lot of evidence for or against Conjecture 3. We do know that the conjecture holds true when $k = 1$: by Descarte's rule each polynomial f_{1j} has at most $2t - 2$ nonzero real roots, so f has at most $2m(t - 1) + 1$ real roots. The case $k = 2$ already looks nontrivial. In the general case we can expand f as a sum of at most kt^m monomials, so we have at most $2kt^m - 1$ real roots. A refutation of the conjecture would be interesting from the point of view of real algebra and geometry as it would yield examples of "sparse like" polynomials with many real roots. Of course, a proof of the conjecture would be even more interesting as it would yield a lower bound for the permanent. The theory of fewnomials [6] provides finiteness results and sometimes quantitative estimates on the number of real roots in very general "sparse like" situations. Unfortunately the existing estimates, at least when applied in a straightforward manner, do not seem to yield any useful bound in this case.

Finally we point out that if true, Conjecture 3 would be a property that is really specific to SPS polynomials: as explained in the introduction, it is known that for general arithmetic circuits the number of real roots cannot be bounded by a polynomial function of the arithmetic circuit size.

References

1. M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proc. 25th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105, 2005. Invited survey.
2. M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proc. 49th IEEE Symposium on Foundations of Computer Science*, pages 67–75, 2008.
3. P. Bürgisser. On defining integers in the counting hierarchy and proving lower bounds in algebraic complexity. *Computational Complexity*, 18:81–103, 2009. Conference paper in Proc. STACS 2007.
4. J. Heintz and C.-P. Schnorr. Testing polynomials which are easy to compute. In *Logic and Algorithmic (an International Symposium held in honour of Ernst Specker)*, pages 237–254. Monographie n^o 30 de L'Enseignement Mathématique, 1982. Preliminary version in *Proc. 12th ACM Symposium on Theory of Computing*, pages 262-272, 1980.
5. V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity test means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
6. A. G. Khovanskii. *Fewnomials*, volume 88 of *Translations of Mathematical Monographs*. American Mathematical Society, 1991.
7. P. Koiran. Valiant's model and the cost of computing integers. *Computational Complexity*, 13:131–146, 2004.
8. G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claude Bernard - Lyon 1, 2003.
9. M. Shub and S. Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of "P=NP". *Duke Mathematical Journal*, 81(1):47–54, 1995.
10. S. Smale. Mathematical problems for the next century. *Mathematical Intelligencer*, 20(2):7–15, 1998.
11. L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.
12. L. G. Valiant. Reducibility by algebraic projections. In *Logic and Algorithmic (an International Symposium held in honour of Ernst Specker)*, pages 365–380. Monographie n^o 30 de L'Enseignement Mathématique, 1982.

The Scheme of Induction for Sharply Bounded Arithmetic Formulas

Leszek Aleksander Kołodziejczyk *

Institute of Mathematics, University of Warsaw, Banacha 2, 02-097 Warszawa, Poland
email: lak@mimuw.edu.pl

1 Introdction and notation

Bounded arithmetic (BA) is a theory axiomatized by the induction scheme for bounded formulas in the language $L_{BA} = \{+, \cdot, 0, 1, \leq, |x|, \#, \lfloor \frac{x}{2} \rfloor\}$. Here $|x|$ is interpreted as $\lceil \log_2(x+1) \rceil$, and elements of the range of $|x|$ are thought as "logarithmically small"; $x\#y$ is interpreted as $2^{|x|\cdot|y|}$, and elements in the range of $\#$ are thought of as powers of 2. Sometimes the language is additionally extended by the function $MSP(x,y) = \lfloor \frac{x}{2^{|y|}} \rfloor$. BA was introduced in [Bus86] and is studied largely because of its close connections to computational complexity theory. For example, BA is finitely axiomatizable if and only if BA proves that the polynomial hierarchy collapses ([Bus95], [Zam96]).

Bounded formulas of L_{BA} form a natural quantifier alternation hierarchy. The bottom levels of the hierarchy are: the class Σ^b_0 of *sharply bounded formulas*, in which all quantifier bounds are of the form $|t|$, and the class Σ^b_1 of sharply bounded formulas preceded by bounded $\exists$ quantifiers (in the standard model $\mathbb{N}$, Σ^b_1 formulas define exactly the NP properties). There is a corresponding hierarchy of fragments of BA. The best known of these is the theory S^1_2, given by a restricted induction scheme for Σ^b_1 formulas. The provably total Σ^b_1 definable functions of S^1_2 are exactly the polytime computable functions. For more on the hierarchies of formulas and theories and their relation to complexity theory, see e.g. [Kra95] or [Bus98].

Separating S^1_2 from BA is one of the fundamental problems in the area, but it seems to be beyond the reach of current methods. On the other hand, the theory T^0_2, or induction for sharply bounded formulas, has traditionally been suspected of being pathologically weak. Actually settling the status of T^0_2, however, remained an open problem for quite long. The last few years have finally seen progress on this problem, leading to the conclusion that the strength of T^0_2 depends crucially on the presence of the MSP function in the language. We review this progress (Sections 2 and 3) and prove some new results (Sections 4 and 5).

Some notational conventions: we write $||x||$ for absolute value to distinguish it from the $|x|$ symbol of L_{BA}. If $\mathcal{N}, \mathcal{N}'$ are models of arithmetic, $\mathcal{N} \preceq_J \mathcal{N}'$ means that $\mathcal{N}'$ is an elementary extension of $\mathcal{N}$ and in both models J is an initial segment. Depending on the context, models of arithmetic are viewed as non-negative parts of ordered rings or as the rings themselves, and we do not take care to distunguish between the two cases. rcl denotes real closure (of a given field or the fraction field of a given ring); for basic information on real closed fields and real closure, see e.g. [BCR98].

2 The strength of sharply bounded induction with MSP

In 2006, Jeřábek proved the following theorem:

Theorem 1. *[Jeř06]* $T^0_2(MSP)$ *proves all* $\forall\Sigma^b_1$ *consequences of* S^1_2.

In particular, $T^0_2(MSP)$ proves the totality of all polytime computable functions, and hence is a relatively strong theory, strong enough that separating $T^0_2(MSP)$ from BA now seems a distant goal. The main ingredient of Jeřábek's argument is the observation that T^0_2 defines the relation "the i-th bit of x is 1", or $i \in x$, and that the amount of induction available in $T^0_2(MSP)$ suffices to prove the comprehension scheme

$$\forall x \exists! y (|y| \leq |x| \wedge \forall i < |x| (i \in y \Leftrightarrow \varphi(i,y)))$$

* Partially supported by grant N N201 382234 of the Polish Ministry of Science and Higher Education.

for a large class of Σ_0^b formulas (called *safe for bit-recursion*).

This makes it possible to show that the class of Σ_1^b definable functions provably total in $T_2^0(MSP)$ contains some basic functions and is closed under composition and a type of recursion known as bounded recursion on notation. By a result of [Cob65], this implies that all polynomial-time functions are provably Σ_1^b definable in $T_2^0(MSP)$, which leads to Theorem 1.

It is worth pointing out that the class of properties definable by $\Sigma_0^b(MSP)$ formulas is a proper subclass of uniform TC^0 (by [Man91], it does not contain $PARITY$). Hence, even induction for a rather weak class of formulas can yield a reasonably strong theory.

3 Sharply bounded induction and open induction

The proof presented in [Jeř06] relies heavily on the presence of the MSP function in the language, and it turns out that Theorem 1 itself requires MSP. Inspired by work of Boughattas and Ressayre [BR] on models of weak subtheories of BA built from "nonstandard reals" instead of just "nonstandard integers", the paper [BK] showed that T_2^0 in the original language L_{BA} is a very weak theory. In particular:

Theorem 2. *[BK] T_2^0 does not prove that a power of 2 has no non-trivial odd divisors.*

Since T_2^0 proves the termination of Euclid's algorithm and hence excludes odd divisors of powers of 2, we get $T_2^0 \subsetneq T_2^0(MSP)$.

The argument of [BK] suggests that T_2^0 is in some ways similar to the theory $IOpen$, axiomatized by the induction scheme for open (i.e. quantifier-free) formulas in the basic arithmetical language of $+, \cdot, 0, 1, \leq$. This similarity is worth exploring further, especially since $IOpen$ and some of its extensions are among the few subtheories of Peano Arithmetic for which it is known how to construct nonstandard models violating rather basic statements of number theory.

Below, we modify methods used in the study of $IOpen$ to show that a well-known nonstandard model of $IOpen$ can be embedded in a model of T_2^0, and to prove an independence result for a slight extension of T_2^0.

4 The Shepherdson model

The following criterion, due to Shepherdson, is used in one way or another in almost all constructions of models of $IOpen$:

Theorem 3. *[She64] Let M be a discrete ordered ring. Then $M \models IOpen$ iff for every $\alpha \in rcl(M)$ there exists $x \in M$ such that $||x - \alpha|| < 1$.*

Definition 1. *The* Shepherdson model M_{Shep} *for $IOpen$ consists of sums of the form $\sum_{l=0}^{k} \alpha_l X^{l/n}$, where $n > 0$ is a natural number, $\alpha_l \in rcl(\mathbb{Q})$ and $\alpha_0 \in \mathbb{Z}$. The ordering is defined by setting $X > \mathbb{Z}$.*

M_{Shep} was originally defined in [She64]. We prove:

Theorem 4. *M_{Shep} can be extended to a model of T_2^0.*

Corollary 1. *T_2^0 does not prove:*

(i) a power of 2 has no non-trivial odd divisors,
(ii) $\sqrt{2}$ is irrational,
(iii) $(x > 0 \wedge y > 0) \Rightarrow x^3 + y^3 \neq z^3$.

Parts (ii) and (iii) of the corollary follow directly from the properties of M_{Shep}. Part (i), which was the main result of [BK], is an obvious consequence of the proof of Theorem 4, sketched below.

Theorem 4 is proved by adapting a classical method originally used by Wilkie in the proof of:

Theorem 5. *[Wil78] Every $\mathbb{Z}$-ring can be extended to a model of $IOpen$.*

(A $\mathbb{Z}$-ring is a discretely ordered ring in which division with remainder by elements of $\mathbb{Z}$ is possible). Wilkie's argument involves embedding a given $\mathbb{Z}$-ring M in a sufficiently saturated real closed field $\mathcal{R}$ and building a chain of $\mathbb{Z}$-rings $M = M_0 \subseteq M_1 \subseteq M_2 \subseteq \ldots$ contained in $\mathcal{R}$ sch that $\bigcup_{n\in\mathbb{N}} M_n \models IOpen$. M_{n+1} is obtained from M_n by adding an element $x \in \mathcal{R}$ which is a witness for some instance of open induction and has the property that all non-constant polynomials in $M_n[x]$ have infinite values. The existence of x follows from the fact that M_n is a $\mathbb{Z}$-ring and the saturation of $\mathcal{R}$.

We adapt Wilkie's proof to a setting in which the role of $\mathbb{Z}$ is played by a nonstandard initial segment of a model of Peano Arithmetic. The initial segment is intended to be the logarithm (i.e. the range of $|\cdot|$) of a model of T_2^0 we will eventually build. So, as a starting point, we fix a countable nonstandard model $\mathcal{N}_0 \models PA$ and an initial segment $J \subseteq_e \mathcal{N}_0$ closed under multiplication.

Definition 2. *Let $\mathcal{N} \succeq_J \mathcal{N}_0$ be such that J is an initial segment of $\mathcal{N}$ and let M be a discretely ordered subring of $rcl(\mathcal{N})$. M is* J-euclidean *if $J \subseteq M$ and division with remainder by elements of J is possible in M, i.e. for each $x \in M$, $0 \neq i \in J$, there exists $y \in M$ such that $\frac{x}{i} - 1 < y \leq \frac{x}{i}$.*

Let $M_0 \subseteq rcl(\mathcal{N}_0)$ consist of all sums of the form $\sum_{l=1}^{k} \alpha_l 2^{i_l}$, where for each l, $i_l \in J$, $\alpha_l \in rcl(J)$, and additionally $\pm\alpha_l \in J$ if $2^{i_l} \in J$. Clearly, M_0 contains an isomorphic copy of M_{Shep}. We also have:

Lemma 1. *M_0 is a J-euclidean subring of $rcl(\mathcal{N}_0)$.*

A J-euclidean ring which contains $\{2^i : i \in J\}$ as a cofinal subset may be regarded as a structure for L_{BA}. Moreover, using Shepherdson's criterion (Theorem 3) and the analysis of sets definable by Σ_0^b formulas from [BK] (carried out earlier, outside the context of nonstandard models, in [Man91]), it is possible to show:

Lemma 2. *Assume that $\mathcal{N} \succeq_J \mathcal{N}_0$ and $M \subseteq rcl(\mathcal{N})$ is J-euclidean. If $M \models IOpen$ and M contains $\{2^i : i \in J\}$ as a cofinal subset, then $M \models T_2^0$.*

To be able to use Lemma 2, we have to know that J-euclidean rings can be extended by adding witnesses for instances of $IOpen$. Unfortunately, we have no obvious analogue of the saturated real closed field from Wilkie's construction. We deal with this issue by extending the ambient model of PA:

Lemma 3. *Assume $\mathcal{N} \succeq_J \mathcal{N}_0$ is countable and M is a J-euclidean subring of $rcl(\mathcal{N})$. Let $\alpha \in rcl(M) \subseteq rcl(\mathcal{N})$. Then there exists a countable model $\mathcal{N}' \succeq_J \mathcal{N}$ and a J-euclidean subring $M' \supseteq M$ of $rcl(\mathcal{N}')$ which contains an element x with $||x - \alpha|| < 1$.*

Lemma 3 is proved by a variant of a standard method from the theory of models of PA, typically used to construct models with a fixed common initial segment (cf. Chapter 2 of [KS06]). Once we have Lemma 3, we can build a chain $\mathcal{N}_0 \prec_J \mathcal{N}_1 \prec_J \ldots$ of models of PA and a corresponding chain $M_0 \subseteq M_1 \subseteq \ldots$ of J-euclidean rings. $\bigcup_{n\in\mathbb{N}} M_n$ is a model of $IOpen$, and by Lemma 2, its restriction to elements bounded by 2^i for some $i \in J$ is a model of T_2^0. Since this model contains $M_0 \supseteq M_{\mathrm{Shep}}$, Theorem 4 follows.

5 An extension of T_2^0

The results of [BK] and of Section 4 reveal a fundamental weakness of T_2^0: it is not able to deal with numbers viewed as binary strings. As noted, T_2^0 can define the relation "the y-th bit of x is 1", but it cannot prove even such basic facts as "every number has a least significant 1 bit", or, equivalently, "for every x, there is a least power of 2 not dividing x". Actually, it is not hard to observe that:

Proposition 1. *T_2^0 + "every number has a least significant 1 bit" proves:*

(i) a power of 2 has no non-trivial odd divisors,
(ii) $\sqrt{2}$ is irrational.

In light of Proposition 1, it is natural to ask whether any interesting independence results can be proved for extensions of T_2^0 which are better at handling bits. In the simple case of T_2^0 + "every number has a least significant 1 bit", we provide a positive answer.

Theorem 6. *T_2^0 + "every number has a least significant 1 bit" does not prove the existence of infinitely many primes.*

The proof is based on a method used by Smith in [Smi93] to show a number of independence results for the theory *IOpen* + "every two numbers have a GCD". Smith applies the basic machinery of Wilkie (with some refinements due to [MM89]), but he additionally makes sure that all elements of the $\mathbb{Z}$-rings built during his chain construction are divisible only by finitely many integers. This guarantees good control over common divisors.

An analogue in our context would be to construct chains of J-euclidean rings none of whose elements are divisible by all powers of 2 from J. This is clearly impossible (a power of 2 above J is always divisible by all powers of 2 from J), but a well-chosen weakening of this requirement works.

The existence of infinitely many primes may well be independent even of $T_2^0(MSP)$ and S_2^1, as the only known proof in BA uses a variant of the pigeonhole principle which is likely to be unprovable in S_2^1. However, even extending Theorem 6 to e.g. the (possibly somehow restricted) open induction scheme in the language with MSP or $x \in y$ is an open problem. Another related challenging open problem concerns the strength of $IOpen(\lfloor \frac{x}{y} \rfloor)$. This theory was shown to be strictly stronger than *IOpen* in [Kay93], but no non-trivial independence results for $IOpen(\lfloor \frac{x}{y} \rfloor)$ are known.

References

[BCR98] J. Bochnak, M. Coste, and M.-F. Roy, *Real Algebraic Geometry*, Springer, 1998.

[BK] S. Boughattas and L. A. Kołodziejczyk, *The strength of sharply bounded induction requires MSP*, Annals of Pure and Applied Logic, in press.

[BR] S. Boughattas and J. P. Ressayre, *Bootstrapping, part I*, Annals of Pure and Applied Logic, in press.

[Bus86] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986.

[Bus95] ———, *Relating the bounded arithmetic and polynomial time hierarchies*, Annals of Pure and Applied Logic **75** (1995), 67–77.

[Bus98] ———, *First-order proof theory of arithmetic*, Handbook of Proof Theory (S. R. Buss, ed.), Elsevier, 1998, pp. 79–147.

[Cob65] A. Cobham, *The intrinsic computational difficulty of functions*, Proc. 2nd International Congress of Logic, Methodology, and Philosophy of Science (Y. Bar-Hillel, ed.), North-Holland, 1965, pp. 24–30.

[Jeř06] E. Jeřábek, *The strength of sharply bounded induction*, Mathematical Logic Quarterly **52** (2006), 613–624.

[Kay93] R. Kaye, *Open induction, Tennenbaum phenomena, and complexity theory*, Arithmetic, proof theory, and computational complexity (P. Clote and J. Krajiček, eds.), Oxford University Press, 1993, pp. 222–237.

[Kra95] J. Krajiček, *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, Cambridge University Press, 1995.

[KS06] R. Kossak and J. Schmerl, *The Structure of Models of Peano Arithmetic*, Oxford University Press, 2006.

[Man91] S. G. Mantzivis, *Circuits in bounded arithmetic part I*, Annals of Mathematics and Artificial Intelligence **6** (1991), 127–156.

[MM89] A. Macintyre and D. Marker, *Primes and their residue rings in models of open induction*, Annals of Pure and Applied Logic **43** (1989), 57–77.

[She64] J. C. Shepherdson, *A non-standard model for a free variable fragment of number theory*, Bulletin de l'Académie Polonaise des Sciences. Série des Sciences Mathématiques, Astronomiques et Physiques **12** (1964), 79–86.

[Smi93] S. T. Smith, *Building discretely ordered Bezout domains and GCD domains*, Journal of Algebra **159** (1993), 191–239.

[Wil78] A. J. Wilkie, *Some results and problems on weak systems of arithmetic*, Logic Colloquium '77 (A. Macintyre et al., ed.), North-Holland, 1978, pp. 285–296.

[Zam96] D. Zambella, *Notes on polynomially bounded arithmetic*, Journal of Symbolic Logic **61** (1996), 942–966.

Complexity of Problems of Commutative Grammars

Eryk Kopczyński

Institute of Informatics, Warsaw University
erykk@mimuw.edu.pl

Abstract. We consider Parikh images of languages accepted by non-deterministic finite automata and context-free grammars; in other words, we treat the languages in a commutative way — we do not care about the order of letters in the accepted word, but rather how many times each one of them appears. In most cases we assume that the alphabet is of fixed size. We show tight complexity bounds for problems like membership, equality, and disjointness. In particular, we show polynomial algorithms for membership and disjointness in case of finite automata, and that inclusion and equality are Π_2^{P} complete in case of context-free grammars over a two letter alphabet.

1 Introduction

We consider languages accepted by regular and context-free grammars, except that we treat the language in a commutative way — we do not care about the order of letters in the accepted word, but rather how many times each one of them appears. In this setting, usual problems, like membership and equality, have different complexities than in the non-commutative case. We show tight complexity bounds for those problems.

It turns out that, contrary to the non-commutative case, the size of alphabet is very important. In the non-commutative case, we can use strings a, ab, and abb to encode a three letter alphabet $\{a, b, c\}$ using two letters. Trying to do this in the commutative case fails, since two different words ac and bb are mapped to $aabb$ and $abab$, which are commutatively the same word. There is no way to map a three letter alphabet to a two letter one which does not collapse anything. Each new letter adds a new dimension to the problem in the commutative case — literally: commutative words (multisets) over an alphabet of size d are better viewed as points in a d-dimensional space, rather than strings.

A well known classic result in this area is the result of Parikh [Par66] that, for a context-free grammar G over alphabet Σ, the Parikh image of G, i.e., the set $\mathrm{out}(G) \subseteq \mathbb{N}^\Sigma$ of such multisets M that, in some word $w \in L(G)$, each letter x appears $M(x)$ times, is a semilinear set. Some complexity results regarding semilinear sets and commutative grammars have been obtained by D. Huynh [Hu80,Hu83], who has shown that equality for semilinear sets and commutative grammars is Π_2^{P}-hard (where Π_2^{P} is the dual of the second level of the polynomial-time hierarchy, [Sto77]).

Some research has also been done in the field of communication-free Petri nets, or Basic Parallel Processes (BPP). A Petri net is communication-free if each transition has only one input. This restriction means that such a Petri net is essentially equivalent to a commutative context-free grammar. [Yen96] shows that the equivalence problem (or, in terms of grammars, equality) for BPP-nets can be solved in $DTIME\left(2^{2^{ds^3}}\right)$.

Contrary to most previous papers on commutative grammars, in most cases we assume that our alphabet is of fixed size. Sometimes we even assume alphabet of size 2, since some algorithms are very simple in this case. We show a polynomial algorithm deciding membership for regular languages (i.e., whether a multiset (given in binary) is in the Parikh image of a regular language).

We also improve upon Parikh's result in two ways, assuming that the alphabet is of size 2. First, $\mathrm{out}(G)$ is produced as a union of linear sets with only two periods (whose magnitude is single exponential in size of G); second, these linear sets can be grouped in a polynomial number of bundles such that each bundle shares the pairs of periods used (we call such a bundle an A, B-frame). Unfortunately, such simple presentation is impossible for alphabets of size greater than 2.

The following table summarizes our results. Alphabet size F means that alphabet is of fixed size, and U means unfixed size. We use c as an abbreviation for complete. Our main results — our most important algorithms — are marked with bold (polynomial algorithms for checking membership and disjointness for regular grammars over alphabets of fixed size, Π_2^{P} completeness of the inclusion problems for context-free

grammars over alphabets of fixed size). Problems which have been shown to be hard are marked with stars (NP-completeness of membership checking for regular grammars over alphabets of unfixed size and context-free grammars over alphabets of size 1, coNP-completeness of universality checking for regular grammars over alphabets of size 1, Π_2^{P}-completeness of inclusion checking for context-free grammars); the reductions are omitted for space reasons. Other results are mostly simple applications of these.

problem	regular				context-free			
alphabet size	1	2	F	U	1	2	F	U
membership	P	P	**P**	NPc*	NPc*	NPc	NPc	NPc
universality	coNPc*	coNPc	coNPc	?	coNPc	coNPc	coNPc	?
inclusion	coNPc	coNPc	coNPc	?	Π_2^{P}c*	$\mathbf{\Pi_2^P c}$	$\mathbf{\Pi_2^P c}$	?
equality	coNPc	coNPc	coNPc	?	Π_2^{P}c	Π_2^{P}c	Π_2^{P}c	?
disjointness	P	P	**P**	coNPc	coNPc	coNPc	coNPc	?

2 Overview

In this section, we present our techniques and results in an informal way. The formal version can be found in the following sections.

Our main observation is that we can treat our runs (or derivations in CF grammars) completely commutatively: we just count how many times each transition (rule) has been used. In both cases, the validity of such a ,,commutative run" can be checked by checking two very simple conditions: Euler condition (each state is entered as many times as it is used) and connectedness (there are no unconnected loops). From this, we immediately get that checking membership of a given multiset in a Parikh image of a context-free language is in NP.

The second observation is that we can decompose a run into smaller parts, ultimately obtaining its *skeleton*, to which we add some (simple) *cycles*. Since the skeleton uses all states that the original run used (which we call its *support*), we can add these cycles in arbitrary numbers to our skeleton, always getting valid runs. Moreover, in case of finite automata (regular grammars), both the skeleton and the cycles are bounded polynomially (in the size of the automaton, $|G|$).

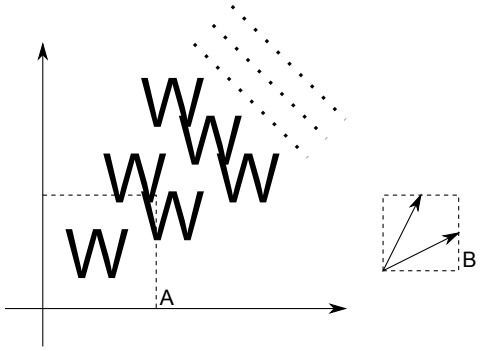
Now, linear algebraic considerations come into play. Whenever we have d linearly independent vectors $v_1, \ldots, v_d$ with integer coordinates in a d-dimensional space, for each other vector v, there is a constant C such that Cv can be written as a linear combination of $v_1, \ldots, v_d$ with integer coefficients. This C is bounded polynomially by coordinates of $v_1, \ldots, v_d$ (d appears in the exponent). In our case, our vectors will be the Parikh images of our cycles, with d letters in our alphabet.

Thus, whenever we have a non-negative integer combination of more than d cycles, where the multiplicities of cycles are big enough, we can reconstruct our Parikh image using different multiplicities of these cycles, and do such ,,shifting" until the multiplicities of some cycles drop. Thus, there are at most d cycles which we are using in large quantities. From this, we get an algorithm for membership: we can guess the small run and the d cycles (making sure that the run crosses these cycles), and then just check whether we obtain our result by adding the cycles in non-negative integer amounts to our run — which boils down to solving a system of equations. This algorithm is polynomial, since everything is bounded polynomially.

The situation is the simplest in the case of $d = 2$, where instead of guessing the two cycles, we can always use the two extreme ones v_a, v_b — i.e., the ones which proportionally contain the greatest quantities of the two letters a and b in our alphabet. Each other cycle can be written as a non-negative combination of these two. Still, we have to take the extreme cycles which cross our small run — which means that we have to guess the two states where they cross, and then take extreme cycles crossing these states. For unfixed d, or for context free grammars, the problem is NP complete.

Now, what about context free grammars? Generally, we can use the same techniques, but now, skeletons and cycles are bounded exponentially. In case of $d = 2$, we get the following Theorem: a Parikh image of G is a union of a polynomial number of A, B-*frames*; where an A, B-frame is a set of vectors defined by some W (a subset of $\mathbb{N}^d$ bounded by A) and two vectors v_a, v_b (bounded by B), consisting of all vectors of form $w + n_a v_a + n_b v_b$ (where $w \in W$). The number of A, B-frames is polynomial, because our two vectors will always correspond to extreme cycles from some state. A and B are exponential.

The following picture shows geometrically what an A, B-frame is: a set W sitting inside of a box of size A (drawn as the letter W) is copied by shifting it in two directions. The vectors by which we are shifting are bounded by B.



It turns out that two such unions of A, B-frames are equal iff they are equal in the exponentially bounded region close to 0. Together with the fact that membership checking is in NP, we get a Π_2^{P}algorithm for checking equality (inclusion) of Parikh images of context-free grammars (for $d = 2$).

For $d > 2$, it may be impossible to get a polynomial number of A, B-frames (for which we have a nice counterexample), which means that the approach outlined above fails (the region would be bounded double exponentially). However, we can circumvent this by splitting $\mathbb{N}^d$ into *regions* — when restricted to a single region, the number of A, B-frames will be polynomial, thus allowing us to use our methods successfully in each region separately, again getting a Π_2^{P} algorithm for deciding equality. The proof is too long and technical, and we had to omit it from this paper.

The exact complexity of deciding inclusion for alphabets of unbounded size remains open.

References

[Hu80] Thiet-Dung Huynh. *The Complexity of Semilinear Sets.* ICALP 1980, LNCS 85, pages 324–337.

[Hu83] Thiet-Dung Huynh. *Complexity of equivalence problems for commutative grammars.* Inform. and Comput. 66 (1985), pages 103–121.

[Par66] Rohit J. Parikh. *On context-free languages.* Journal of the Association for Computing Machinery, 13(4):570-581, 1966.

[Sto77] L. Stockmeyer, *The polynomial-time hierarchy.* Theoret. Comput. Sci. 3 (1977), pages 1-22.

[Yen96] Hsu-Chun Yen, *On reachability equivalence for BPP-nets.* Theoret. Comput. Sci. 179 (1996), pages 301–317.

Definability of Combinatorial Functions and Their Linear Recurrence Relations

T. Kotek and J.A. Makowsky

Department of Computer Science
Technion — Israel Institute of Technology
Haifa, Israel
{tkotek,janos}@cs.technion.ac.il

Abstract. We consider functions of natural numbers which allow a combinatorial interpretation as density functions (speed) of classes of relational structures, such as Fibonacci numbers, Bell numbers, Catalan numbers and the like. Many of these functions satisfy a linear recurrence relation over $\mathbb{Z}$ or $\mathbb{Z}_m$ and allow an interpretation as counting the number of relations satisfying a property expressible in Monadic Second Order Logic (MSOL).
C. Blatter and E. Specker (1981) showed that if such a function f counts the binary relations satisfying a property expressible in MSOL then f satisfies for every $m \in \mathbb{N}$ a linear recurrence relation over $\mathbb{Z}_m$.
In this abstract we give a complete characterization in terms of definability in MSOL of the combinatorial functions which satisfy a linear recurrence relation over $\mathbb{Z}$, and discuss various extensions and limitations of the Specker-Blatter theorem.

1 The Speed of a Class of Finite Relational Structures

Let $\mathcal{P}$ be a graph property, and $\mathcal{P}^n$ be the set of graphs with vertex set $[n]$. We denote by $sp_{\mathcal{P}}(n) = |\mathcal{P}^n|$ the number of labeled graphs in $\mathcal{P}^n$. By labeled graphs we mean the vertices are considered distinct, as oppose to the case of counting unlabeled graphs where we would only be interested in counting isomorphism types of graphs of size n. The function $sp_{\mathcal{P}}(n)$ is called the *speed of* $\mathcal{P}$, or in earlier literature the *density of* $\mathcal{P}$. Instead of graph properties we also study classes of finite relational structures $\mathcal{K}$ with relations $R_i : i = 1, \ldots, s$ of arity ρ_i. For the case of $s = 1$ and $\rho_1 = 1$ such classes can be identified with binary words over the positions $1, \ldots, n$. This ties our results with the study of formal power series. We will not discuss this in depth in this abstract.

The study of the function $sp_{\mathcal{K}}(n)$ has a rich literature concentrating on two types of behaviours of the sequence $sp_{\mathcal{K}}(n)$: recurrence relations and growth rate. Clearly, the existence of recurrence relations limits the growth rate.

In C. Blatter and E. Specker [5] the case of $\mathcal{K}$ was studied, where $\rho_i \leq 2$ for all $i \leq s$ and $\mathcal{K}$ definable in MSOL. They showed that in this case for every $m \in \mathbb{N}$, the sequence $sp_{\mathcal{K}}(n)$ is ultimately periodic modulo m, or equivalently, that the sequence $sp_{\mathcal{K}}(n)$ satisfies a linear recurrence relation over $\mathbb{Z}_m$.

In E.R. Scheinerman and J. Zito [15] the function $sp_{\mathcal{P}}(n)$ was studied for *hereditary* graph properties $\mathcal{P}$, i.e., graph properties closed under induced subgraphs. They were interested in the growth properties of $sp_{\mathcal{P}}(n)$. The topic was further developed by J. Balogh, B. Bollobas and D. Weinreich in a sequence of papers, [7, 1, 2], which showed that only six classes of growth of $sp_{\mathcal{P}}(n)$ are possible, roughly speaking, constant, polynomial, or exponential growth, or growth in one of three factorial ranges. They also obtained similar results for monotone graph properties, i.e., graph properties closed under subgraphs, [3].

In this abstract we concentrate on the relationship between definability of a class $\mathcal{K}$ of relational structures and the various linear recurrence relations $sp_{\mathcal{K}}(n)$ satisfy.

2 Combinatorial Functions and Specker Functions

We would like to say that a function $f : \mathbb{N} \to \mathbb{N}$ is a combinatorial function if it has a combinatorial interpretation. One way of making this more precise is the following. We say that $\mathcal{K}$ is definable in $\mathcal{L}$ if there is a $\mathcal{L}$-sentence ϕ such that for every $\bar{R}$-structure $\mathfrak{A}$, $\mathfrak{A} \in \mathcal{K}$ iff $\mathfrak{A} \models \phi$. Then a function $f : \mathbb{N} \to \mathbb{N}$ is a combinatorial function if $f(n) = sp_{\mathcal{K}}(n)$ for some class of finite structures $\mathcal{K}$ definable in a suitable logical formalism $\mathcal{L}$. Here $\mathcal{L}$ could be FOL, MSOL or any interesting fragment of Second Order Logic, SOL. We assume the reader is familiar with these logics, cf. [10].

Definition 1 (Specker[1] function).

[1] E. Specker studied such functions in the 1970ties in his lectures on topology at ETH-Zurich.

*A function $f : \mathbb{N} \to \mathbb{N}$ is called a $\mathcal{L}^k$-*Specker function *if there is a finite set of relation symbols $\bar{R}$ of arity at most k and a class of $\bar{R}$-structures $\mathcal{K}$ definable in $\mathcal{L}$ such that $f(n) = sp_{\mathcal{K}}(n)$.*

A typical non-trivial example is given by A. Cayley's Theorem from 1889, which says that $T(n) = n^{n-2}$ can be interpreted as the number of labeled trees on n vertices. Another example are the Bell numbers B_n which count the number of equivalence relations on n elements.

In this paper we study under what conditions the Specker function given by the sequence $sp_{\mathcal{K}}(n)$ satisfies a linear recurrence relation.

We say a function $f : \mathbb{N} \to \mathbb{N}$ is ultimately periodic (u.p.) over $\mathcal{R} = \mathbb{Z}$ or over $\mathcal{R} = \mathbb{Z}_m$ if there exist $i, n_0 \in \mathbb{N}$ such that for every $n \geq n_0$, $f(n + i) = f(n)$ over $\mathcal{R}$. It is well-known that f is u.p. over $\mathbb{Z}_m$ iff it satisfies a linear recurrence relation with constant coefficients over $\mathbb{Z}_m$. We note that if f satisfies a linear recurrence over $\mathbb{Z}$ then it also satisfies a linear recurrence over $\mathbb{Z}_m$ for every m. C. Blatter and E. Specker proved the following remarkable but little known theorem in [5],[6],[16].

Theorem 1 (Specker-Blatter Theorem). *If $f(n) = sp_{\mathcal{K}}(n)$ is definable in* MSOL^2, MSOL *with unary and binary relation symbols only, then for every $m \in \mathbb{N}$, $f(n)$ satisfies a linear recurrence relation with constant coefficients*

$$sp_{\mathcal{K}}(n) \equiv \sum_{j=1}^{d_m} a_j^{(m)} sp_{\mathcal{K}}(n-j) \ (mod\, m)$$

and hence is ultimately periodic over $\mathbb{Z}_m$.

In [12] it was shown that in Theorem 1 the logic MSOL can be augmented by modular counting quantifiers. Furthermore, E. Fischer showed in [11] that this cannot be extended to quaternary relations.

3 Extending MSOL and Order Invariance

In this paper we investigate the existence of linear and modular linear recurrence relations of Specker functions for the case where $\mathcal{K}$ is definable in logics $\mathcal{L}$ which are sublogics of SOL and extend MSOL. We now look at the case where $[n]$ is equipped with a linear order.

Definition 2 (Order invariance). *A class $\mathcal{D}$ of* ordered $\bar{R}$-structures *is a class of $\bar{R} \cup \{<_1\}$-structures, where for every $\mathfrak{A} \in \mathcal{D}$ the interpretation of the relation symbol $<_1$ is always a linear order of the universe of $\mathfrak{A}$. An $\mathcal{L}$ formula $\phi(\bar{R}, <_1)$ for ordered $\bar{R}$-structures is* truth-value order invariant (t.v.o.i.) *if for any two structures $\mathfrak{A}_i = \langle [n], <_i, \bar{R}\rangle$ $(i = 1, 2)$ we have that $\mathfrak{A}_1 \models \phi$ iff $\mathfrak{A}_2 \models \phi$. Note $\mathfrak{A}_1$ and $\mathfrak{A}_2$ differ only in the linear orders $<_1$ and $<_2$ of $[n]$. We denote by* TV$\mathcal{L}$ *the set of $\mathcal{L}$-formulas for ordered $\bar{R}$-structures which are t.v.o.i. We consider* TV$\mathcal{L}$ *formulas as formulas for $\bar{R}$-structures.*

For a class of ordered structures $\mathcal{D}$, let $osp_{\mathcal{D}}(n, <_1) =$

$$|\{(R_1, \ldots, R_s) \subseteq [n]^{\rho(1)} \times \ldots \times [n]^{\rho(s)} : \langle [n], <_1, R_1, \ldots, R_s\rangle \in \mathcal{D}\}|$$

*A function $f : \mathbb{N} \to \mathbb{N}$ is called an $\mathcal{L}^k$-*ordered Specker function *if there is a class of ordered $\bar{R}$-structures $\mathcal{D}$ of arity at most k definable in $\mathcal{L}$ such that $f(n) = osp_{\mathcal{D}}$. A function $f : \mathbb{N} \to \mathbb{N}$ is called a* counting order invariant (c.o.i.) $\mathcal{L}^k$-Specker function *if there is a finite set of relation symbols $\bar{R}$ of arity at most k and a class of ordered $\bar{R}$-structures $\mathcal{D}$ definable in $\mathcal{L}$ such that for all linear orders $<_1$ and $<_2$ of $[n]$ we have $f(n) = osp_{\mathcal{D}}(n, <_1) = osp_{\mathcal{D}}(n, <_2)$.*

Every formula $\phi(\bar{R}, <_1) \in$ TVSOL^k is equivalent to the SOL^k formula $\psi(\bar{R}) = \exists <_1 \phi(\bar{R}, <_1) \wedge \phi_{linOrd}(<_1)$, where $\phi_{linOrd}(<_1)$ says $<_1$ is a linear ordering of the universe. Moreover, every TVMSOL^k-Specker function is also a counting order invariant MSOL^k-Specker function. However, there are counting order invariant MSOL^2-definable Specker functions which are not TVMSOL^2-definable.

It is folklore that every formula in $C_{a,b}$MSOL^k, the logic MSOL extended with modular counting quantifiers, is equivalent to a formula in TVMSOL^k.

4 Main Results

Our first result is a characterization of functions over the natural numbers which satisfy a linear recurrence relation over $\mathbb{Z}$.

Theorem 2. *Let f be a function over $\mathbb{N}$. Then f satisfies a linear recurrence relation over $\mathbb{Z}$ iff $f = f_1 - f_2$ is the difference of two c.o.i* MSOL^1*-Specker functions.*

Viewing unary structures as words, we obtain one direction of the proof of Theorem 2 via the logical characterization of regular languages by R. Büchi (and later but independently of C. Elgot and B. Trakhtenbrot), cf. [13, 9], and using [8]. For the other direction we introduce Specker polynomials, which can be thought of as a special case of graph polynomials where graphs are replaced by linear orders.

We may derive a corollary in the terminology of rational functions, cf. [14, 4]:

Corollary 1. *Let f be a function $\mathbb{N} \to \mathbb{N}$. Then f is $\mathbb{Z}$-rational iff f is the difference of two $\mathbb{N}$-rational functions.*

We show that the Specker-Blatter Theorem cannot be extended to c.o.i Specker functions which are definable in MSOL^2 by considering the Catalan numbers. However, if we require that the defining formula ϕ of a Specker function is itself order invariant, i.e. $\phi \in \mathrm{TVMSOL}^2$, then the Specker-Blatter Theorem still holds.

Theorem 3. *Let f be a TVMSOL^2-Specker function. Then, for all $m \in \mathbb{N}$ the function f satisfies a modular linear recurrence relation modulo m.*

References

1. J. Balogh, B. Bollobás, and D. Weinreich. The speed of hereditary properties of graphs. *J. Comb. Theory, Ser. B*, 79(2):131–156, 2000.
2. J. Balogh, B. Bollobas, and D. Weinreich. The penultimate rate of growth for graph properties. *EUROCOMB: European Journal of Combinatorics*, 22, 2001.
3. J. Balogh, B. Bollobás, and David Weinreich. Measures on monotone properties of graphs. *Discrete Applied Mathematics*, 116(1-2):17–36, 2002.
4. J. Berstel and C. Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1988.
5. C. Blatter and E. Specker. Le nombre de structures finies d'une th'eorie à charactère fin. *Sciences Mathématiques, Fonds Nationale de la rechercheScientifique, Bruxelles*, pages 41–44, 1981.
6. C. Blatter and E. Specker. Recurrence relations for the number of labeled structures on a finite set. In E. Börger, G. Hasenjaeger, and D. Rödding, editors, *In Logic and Machines: Decision Problems and Complexity*, volume 171 of *Lecture Notes in Computer Science*, pages 43–61. Springer, 1984.
7. B. Bollobas and A. Thomason. Projections of bodies and hereditary properties of hypergraphs. *J. London Math. Soc*, 27, 1995.
8. N. Chomsky and M.P. Schützenberger. The algebraic theory of context free languages. In P. Brafford and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 118–161. North Holland, 1963.
9. H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, 1995.
10. H.D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic, 2nd edition*. Undergraduate Texts in Mathematics. Springer-Verlag, 1994.
11. E. Fischer. The Specker-Blatter theorem does not hold for quaternary relations. *Journal of Combinatorial Theory, Series A*, 103:121–136, 2003.
12. E. Fischer and J.A. Makowsky. The Specker-Blatter theorem revisited. In *COCOON'03*, volume 2697 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2003.
13. L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
14. A. Salomaa and M. Soittola. *Automata theoretic aspects of formal power series*. Springer, 1978.
15. E.R. Scheinerman and J.S. Zito. On the size of hereditary classes of graphs. *J. Comb. Theory, Ser. B*, 61(1):16–39, 1994.
16. E. Specker. Application of logic and combinatorics to enumeration problems. In E. Börger, editor, *Trends in Theoretical Computer Science*, pages 141–169. Computer Science Press, 1988. Reprinted in: Ernst Specker, Selecta, Birkhäuser 1990, pp. 324-350.

A Logic to Capture P-Time Computability on Cantor Space

Oleg V. Kudinov[1⋆] and Victor L. Selivanov[2⋆⋆]

[1] S.L. Sobolev Institute of Mathematics, 4 Acad. Koptyug avenue, 630090 Novosibirsk, Russia,
kud@math.nsc.ru
[2] A.P. Ershov Institute of Informatics Systems, 6 Acad. Lavrentjev pr., 630090 Novosibirsk, Russia,
vseliv@iis.nsk.su

Abstract. We propose some class of formulas similar to Σ-formulas in admissible set theory which captures the complexity class of polynomial-time computable closed subsets of Cantor space. As a corollary, similar characterizations can be deduced for polynomial-time computable functions on Cantor space.
Key words. Descriptive complexity, P-time computability, bounded quantification, definability.

1 Introduction

After essential progress in finite model theory resulted in beautiful semantic characterizations of the most important complexity classes (see e.g.[3, 5]), the search for suitable logics to capture natural complexity classes for popular topological spaces becomes actual and natural. The general aim here is to bridge the gap between numerical analysis and computability/complexity theory for continuous structures. However, in this search one has to make a choice between existing non-equivalent approaches to computability on continuous structures. We choose the approach of computable analysis [4, 6] because this approach seems to be the adequate foundation for numerical analysis.

As also in discrete complexity theory, computability in polynomial time (P-time for short) on computable metric spaces is especially important because it is a good candidate for formalizing the notion of feasible computation in analysis [4, 6]. In this abstract we discuss a possibility to find a logical characterization of P-time computability for continuous structures in the spirit of finite model theory.

We concentrate here on the Cantor space. Among reasons in favor of the Cantor space is its compactness and the fact that the P-time computable closed subsets form a lattice, in contrast with the space of reals where the corresponding class of subsets is not closed under intersection [2, 1]. As a result, the logical characterization of the closed sets in Cantor space looks more easy and natural than in some other spaces.

We hope to consider logical characterizations of P-time computability (and of other natural complexity classes) in some other important spaces in a subsequent publication.

2 Preliminaries

Keeping in mind the Cantor space $C = 2^\omega$, we extend our consideration to the structure $A = C \cup W$, joining the set $W = 2^{<\omega}$ of finite binary strings. We recall some standard relations and operations on A.

For $x \in W$ and $y \in A$, let $x \sqsubset y$ denote that x is a proper initial segment of y. For $x, y \in A$, let $x < y$ denote that x is lexicographically less than y, in particular the situation $x \sqsubseteq y \wedge (x < y \vee y < x)$ is impossible. If $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$, let $x \wedge y$ denote the greatest z with $z \sqsubseteq x, y$, so $z \in W$. For $f, g \in C$ let $f \oplus g$ denote the function h such that $h(2n) = f(n)$ and $h(2n+1) = g(n)$ for each $n < \omega$. For $f \in C$ and $n < \omega$, let $f|_n$ denote the word $f(0) \dots f(n-1)$. For $x, y \in W$, $x * y$ denotes the result of their concatenation, and $lh(x)$ denotes the length of x; so x may be written as the word $x(0) \dots x(lh(x)-1)$.

We equip the set A with the topology, which base consists of the sets $\{y \in W \mid x \sqsubseteq y\}$ and the sets $\{y \in C \mid x \sqsubseteq y\}$, where $x \in W$. This topology is not standard one, but it induces the Cantor topology on C, since the considered space A is topological sum of standard spaces W and C.

Next we list the predicates in our language (signature):

⋆ Supported by DFG-RFBR (Grant 436 RUS 113/1002/01, 09-01-91334) and by RFBR Grant 07-01-00543a.
⋆⋆ Supported by DFG-RFBR (Grant 436 RUS 113/1002/01, 09-01-91334) and by RFBR Grant 07-01-00543a.

1. The predicate $<$ on A.
2. The unary predicates C for 2^ω and W for $2^{<\omega}$.
3. The predicate $P_0(f,g)$ which is true iff the word $f \wedge g$ is defined and has 0 at the end; the predicate P_1 is defined similarly with 1 instead of 0.
4. The predicate $R_<(f,g,h)$ which is true iff the word $f \wedge g$ exists and is $< h$.
5. The predicate $R_\sqsubseteq(f,g,h)$ which is true iff the word $f \wedge g$ exists and is $\sqsubseteq h$.
6. The predicate defined via the condition $f_1 \wedge f_2 < g_1 \wedge g_2$
7. The predicate defined via the condition $f_1 \wedge f_2 \sqsubseteq g_1 \wedge g_2$.
8. The predicate defined via the condition $(f_1 \wedge f_2) * (g_1 \wedge g_2) < h$.
9. The predicate defined via the condition $(f_1 \wedge f_2) * (g_1 \wedge g_2) \sqsubseteq h$.
10. The predicate defined via the condition $lh(f_1 \wedge f_2) = lh(g_1 \wedge g_2)$.
11. The predicate defined via the condition $lh(f_1 \wedge f_2) \cdot lh(g_1 \wedge g_2) = lh(h_1 \wedge h_2)$.
12. The predicate defined via the condition $f \oplus g < h$. The predicate defined via condition $f \oplus g > h$.
13. The predicate defined via the condition $(f \wedge g) * 0 \sqsubseteq f$.
14. The predicate defined via the condition $(f \wedge g) * 1 \sqsubseteq f$.
15. The constant symbol $\perp$ for the empty word.

We are ready to explain the important notion of Δ_0-formula of our language. These formulas are constructed starting from the atomic formulas using the connectives $\vee, \wedge$ and the bounded quantifiers defined as follows: if $\psi(x,f,g)$ is a Δ_0-formula then so are the expressions $\exists_b x|(f,g)\psi$ and $\forall_b x|(f,g)\psi$. Semantically, the existential bounded quantification means that for some x it holds

$$W(x) \wedge (x \wedge f) \sqsubset (f \wedge g) \wedge (x = (x \wedge f) * 0 \vee x = (x \wedge f) * 1) \wedge \psi(x,f,g).$$

The universal bounded quantification is defined similarly in such a way that $\forall_b x|(f,g)\psi$ is semantically equivalent to $\neg\exists_b x|(f,g)\neg\psi$. Note that any signature predicate symbol should occur only positively in Δ_0-formulas, the negation is not permitted! This is the main reason why the following assertion holds.

Lemma 1. *1. Any Δ_0-definable relation on A is monotone (with respect to $\sqsubseteq$) and open.*
2. The restriction of any Δ_0-definable relation to W is P-time computable in the sense of discrete complexity theory.

Proof is straightforward.
Another useful property of Δ_0-definable relation is the following

Lemma 2. *If $\Phi(x_1,\ldots,x_s)$ is a Δ_0-formula then the set (of tuples of finite words)*

$$\{(h_1,\ldots,h_s) \in W^s | \forall f_1,\ldots,f_s((h_1 \sqsubseteq f_1 \wedge \ldots h_s \sqsubseteq f_s) \to \Phi(f_1,\ldots,f_s))\}$$

is Δ_0-definable and P-time computable in the sense of discrete complexity theory.

Proof (sketch). To construct the intermediate formula $\Phi'(x_1,\ldots,x_s)$, one need to replace any occurrences of subformulas $C(x_i)$ by formulas $(C(x_i) \vee W(x_i))$ for $i = 1,\ldots,s$. The final formula is

$$\Phi'(x_1,\ldots,x_s) \wedge W(x_1) \ldots \wedge W(x_s),$$

the required property is verified by standard induction on Δ_0-formulas.

3 Main result

The main result of this paper is the following

Theorem 1. *Let $R \subseteq 2^\omega$ be a closed subset. Then R is P-time computable iff for some Δ_0-formula $\Phi(x_1,\ldots,x_s,y)$ it holds*

$$f \in R \leftrightarrow \exists x_1,\ldots,x_s(C(x_1) \wedge \ldots \wedge C(x_s) \wedge \neg\Phi(x_1,\ldots,x_s,f)$$

.

Proof (sketch). One part is not hard since it is based on mentioned above lemmas related to properties of Δ_0-formulas. Dealing with P-time computable closed $R \subseteq C$, we consider the language R^*, consisting of all finite initial segments of elements of R. This language is computable in polynomial time and we make use of famous characterization of P-time computable query languages (see e.g.[3, 5]) in modern terms of appropriate LFP-operator, considering words as structures. Class of Δ_0-formulas is rather powerful to encode usual quantification on these structures, and appropriate elements of C can be chosen to encode calculations of corresponding LFP-operator.

Of course, the result may be trivially reformulated for the important particular case of P-time computable elements of Cantor space.

References

1. V. Brattka and K. Weihrauch. Computability on subsets of euclidean space I: Closed and compact sets. *Theoretical Computer Science*, 219:65–93, 1999.
2. P. Hertling. Is the Mandelbrot set computable? *Mathematical Logic Quarterly*, 51, No. 1, 5–18 (2005).
3. N. Immerman. *Descriptive Complexity.* Springer Verlag, New-York, 1999.
4. Ker-I Ko. *Complexity Theory of Real Functions.* Birkhäuser, Boston, 1991.
5. M. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM Symposium on the Theory of Computing*, pages 37–146, 1982.
6. K. Weihrauch. *Computable Analysis.* Springer Verlag, Berlin, 2000.

Complete Problems and Bounded Arithmetc for LOGCFL

Satoru Kuroda

Gunma Prefectural Women's University, 1395-1, Kaminote, Tamamura, Gunma, Japan, 370-1193,
satoru@gpwu.ac.jp

1 Introduction

In this paper we give a general framework for building bounded arithmetic theories using complete problems for certain complexity classes. As an application we define a bounded arithmetic theory for LOGCFL based on the acyclic conjunctive query problem.

Formulations and techniques used here based mainly on previous works by Kolokolova [3] and Nguyen [5]. Nevertheless, we will treat complexity classes which are seemingly unable to handled directly by their formulations.

This can be illustrated by comparing two complexity classes, namely NL and LOGCFL. These two classes both contain the concept of nondeterminism. However, they have rather different nature.

Consider the st-connectivity problem which is complete for NL. The witness for an instance of it is a path from the start node to the goal node.

On the other hand, it can also be witnessed by the set of nodes which are reachable from the node s and this can be computed in deterministic polynomial time. This means that the nondeterminism of NL has a deterministic alternative with high feasibility.

On the contrary, it seems unlikely that most complete problems of LOGCFL also has this property. For instance, to find a solution of an acyclic query on a database can be done nondeterministically. But finding all solutions in brute force manner requires exponential time and thus not feasible in general.

Since Nguyen's system **VNL** for NL [5] heavily depends on the property, the above argument suggests that it is hard to construct a similar system for LOGCFL. So instead we consider another property of when a complexity class of predicates has a nice counterpart in function classes.

Our approach is based on the Lindström quantifier expressing a complete problem for a given complexity class. In the previous work, the author defined a system $\mathbf{V}$-Q^{SAC} which captures LOGCFL. In this paper we generalize the method used there and give a framework for proving the witnessing theorem of a system defined via Lindström quantifier.

2 Witnessing theorem via complete problems

We assume basic knowledge about two-sort bounded arithmetic, see [3] for details.

A signature is a finite collection of relation symbols and constant symbols. For a signature σ, we define $Struct(\sigma)$ to be the set of finite structures over σ. Let $\sigma = \langle P_1, \cdots, P_s \rangle$ be a signature where $P_1, \ldots, P_s$ are relation symbols and $\mathcal{K} \subseteq Struct(\sigma)$ be a set which is complete for a given complexity class $\mathcal{C}$. We define $Q_{\mathcal{K}}[P_1, \ldots, P_s]$ to be a formula in the signature σ, where $Q_{\mathcal{K}}$ is called the Lindström quantifier.

Let $\tau = \langle R_1, \ldots, R_k, c_1, \ldots, c_l \rangle$ be a signature where $R_1, \ldots, R_k$ are relation symbols and $c_1, \ldots, c_l$ are constant symbols. For τ formulae $\phi_1, \ldots, \phi_s$ we define

$$\mathcal{A} \models Q_{\mathcal{K}}[\phi_1, \ldots, \phi_s] \Leftrightarrow (univ(\mathcal{A}), \phi_1^{\mathcal{A}}, \ldots, \phi_s^{\mathcal{A}}) \in \mathcal{K}.$$

for any $\mathcal{A} \in Struct(\tau)$, where $univ(\mathcal{A})$ is the universe of $\mathcal{A}$.

When we consider the subclass $\mathcal{C}$ of P, the membership relation $(univ(\mathcal{A}), \phi_1^{\mathcal{A}}, \ldots, \phi_s^{\mathcal{A}}) \in \mathcal{K}$ can be described by some Σ_1^B relation. More precisely, we define

Definition 1. *The set $\mathcal{K} \subseteq Struct(\sigma)$ has a Σ_1^B description $\varphi(n, P_1, \ldots, P_s) \in \Sigma_1^B$ if for all $n, P_1, \ldots, P_s$,*

$$\varphi(n, P_1, \ldots, P_s) \Leftrightarrow (\{0, \ldots, n-1\}, P_1, \ldots, P_s) \in \mathcal{K}$$

holds in the standard model.

Now we shall give a presentation of Lindström quantifier in two sort systems. The language L_2 of two sort systems consists of number variables $x, y, \ldots$ and string variables $X, Y, \ldots$ together with symbols $Z(x) = 0$, $x + y$, $x \cdot y$, $x \leq y$, $x \in Y$. The idea is to give a description of the relation $\mathcal{A} \models Q_{\mathcal{K}}[\phi_1, \ldots, \phi_s]$ in the language L_2. Note that a τ-structure is coded by a tuple $\langle n, c_1, \ldots, c_l, R_1, \ldots, R_k \rangle$ where n is the size of the universe. The description of the above satisfaction relation is obtained by replacing $P_1, \ldots, P_s$ by $\phi_1^{\mathcal{A}}, \ldots, \phi_s^{\mathcal{A}}$ respectively.

Definition 2. *Let $\varphi_{\mathcal{K}}(n, P_1, \ldots, P_s)$ be a Σ_1^B description of $\mathcal{K}$ for some relational signature σ. Let $\phi_1, \ldots, \phi_s$ be L_2 formulae. Then we define $\varphi_{\mathcal{K}}(n, c_1, \ldots, c_l, R_1, \ldots, R_k, \phi_1, \ldots, \phi_s)$ as the L_2 formula which is obtained from $\varphi_{\mathcal{K}}$ by replacing all occurrences of $P_i(x_1, \ldots, x_{t_i})$ by $\phi_i(x_1, \ldots, x_{t_i}, c_1, \ldots, c_l, R_1, \ldots, R_k)$ for $1 \leq i \leq s$. We call this scheme the Σ_1^B description of $Q_{\mathcal{K}}$ over the signature τ. For a class Φ of L_2 formulae, let $\varphi(\Phi)$ be the class of formulae of the form $\varphi_{\mathcal{K}}(n, c_1, \ldots, c_l, R_1, \ldots, R_k, \phi_1, \ldots, \phi_s)$ where $\phi_1, \ldots, \phi_s \in \Phi$.*

Based on this argument we define a L_2-system as follows:

Definition 3. *Let $\mathcal{K} \subseteq Struct(\sigma)$ for some relational signature σ and $\varphi_{\mathcal{K}}$ be the Σ_1^B description of $Q_{\mathcal{K}}$ over the signature $\tau = \langle i, c_1, \ldots, c_l, R_1, \ldots, R_k \rangle$. The L_2 theory $\mathbf{V}$-$Q_{\mathcal{K}}$ consists of the axioms BASIC together with $\varphi_{\mathcal{K}}(\Sigma_0^B)$-COMP.*

We say that an L_2-theory T captures a complexity class $\mathcal{C}$ if the Σ_1^B definable functions of T coincides with those computable in $\mathcal{C}$.

Our formulation of $\mathbf{V}$-$Q_{\mathcal{K}}$ resembles to the theory defined by Kolokolova [3]. So we claim that Kolokolova's criteria for $\mathbf{V}$-Φ to capture a complexity class applies to our system as well with a slight modification and by way of another theory.

Definition 4. *The L_2-theory $\mathbf{V}$-$\mathcal{K}$ is the system V^0 extended by the axiom stating the existence of witnesses for $\mathcal{K} \cup \mathcal{K}^c$:*

$$(\forall n)(\forall P_1) \cdots (\forall P_s)\, (\varphi_{\mathcal{K}}(n, P_1, \ldots, P_s) \vee \psi_{\mathcal{K}}(n, P_1, \ldots, P_s))$$

We say that $\mathbf{V}$-$\mathcal{K}$ is *properly defined* if there exist Σ_1^B descriptions for both $\mathcal{K}$ and $\mathcal{K}^c$.

We define the following criteria for a system to capture a complexity class.

- **Strong closure** : $\varphi_{\mathcal{K}}(\Sigma_0^B)$ is *strongly closed* if for any $\psi \in \Sigma_0^B(\varphi_{\mathcal{K}}(\Sigma_0^B))$ there exists $\eta \in \varphi_{\mathcal{K}}(\Sigma_0^B)$ such that
$$\mathbf{V}\text{-}Q_{\mathcal{K}} \vdash \psi \leftrightarrow \eta.$$
- **Self-witnessing** : $\varphi_{\mathcal{K}}(\Sigma_0^B)$ is *self-witnessing* if for $(\exists Z < t)\varphi_{\mathcal{K}}^0(\bar{x}, \bar{X}, Z) \in \varphi_{\mathcal{K}}(\Sigma_0^B)$ where $\varphi_{\mathcal{K}}^0 \in \Sigma_0^B$ if $\mathbf{V}$-$Q_{\mathcal{K}} \vdash (\forall \bar{x})(\forall \bar{X})(\exists Z)\varphi_{\mathcal{K}}^0(\bar{x}, \bar{X}, Z)$ then there exists a polynomial growth function $F(\bar{x}, \bar{X})$ which is bitwise Σ_1^B definable in $\mathbf{V}$-$Q_{\mathcal{K}}$ such that
$$\mathbf{V}\text{-}Q_{\mathcal{K}} \vdash (\forall \bar{x})(\forall \bar{X})\varphi_{\mathcal{K}}^0(\bar{x}, \bar{X}, F(\bar{x}, \bar{X})).$$

The reason for adopting the self-witnessing condition rather than the constructiveness condition of Kolokolova is mainly due to technical reasons. However we point out that the self-witnessing condition is closely related to the concept of computing a certificate of a predicate in a given nondeterministic class. In fact, Gottlob et.al. [2] proved that the certificate for LOGCFL predicates can be computed within the class and it is used to prove that the system defined in the next section captures LOGCFL.

Now we have the following witnessing theorem.

Theorem 1. *Let $Q_{\mathcal{K}}(FO)$ capture the complexity class $\mathcal{C}$ over a given signature τ and has a Σ_1^B description $\varphi_{\mathcal{K}}(\Sigma_0^B)$. Suppose that $\varphi_{\mathcal{K}}(\Sigma_0^B)$ is strongly closed and self-witnessing. If $\mathbf{V}$-$Q_{\mathcal{K}}$ contains V^0 then $\mathbf{V}$-$Q_{\mathcal{K}}$ is equivalent to $\mathbf{V}$-$\mathcal{K}$. Furthermore, both $\mathbf{V}$-$Q_{\mathcal{K}}$ and $\mathbf{V}$-$\mathcal{K}$ captures the class $\mathcal{C}$.*

(Proof Sketch).

First we sketch the proof of the first part of the theorem. To prove that $\mathbf{V}$-$Q_{\mathcal{K}}$ contains $\mathbf{V}$-$\mathcal{K}$, we show that $\mathbf{V}$-$Q_{\mathcal{K}}$ proves

$$(\forall n)(\forall P_1) \ldots (\forall P_s)(\varphi_{\mathcal{K}}(n, P_1, \ldots, P_s) \vee \psi_{\mathcal{K}}(n, P_1, \ldots, P_s)).$$

Since $\mathbf{V}$-$Q_{\mathcal{K}}$ is strongly closed, it proves

$$(\forall n)(\forall P_1)\dots(\forall P_s)(\varphi_{\mathcal{K}}(n,P_1,\dots,P_s) \leftrightarrow \psi_{\mathcal{K}}(n,P_1,\dots,P_s)).$$

Therefore we have $\neg\psi_{\mathcal{K}} \rightarrow \varphi_{\mathcal{K}}$ which is equivalent to $\psi_{\mathcal{K}} \vee \varphi_{\mathcal{K}}$.

For the converse inclusion we need the conservative universal extension $\overline{\mathbf{V}\text{-}\mathcal{K}}$ of $\mathbf{V}$-$\mathcal{K}$ which is defined in an analogous manner as in [5]. Since $\overline{\mathbf{V}\text{-}\mathcal{K}}$ is conservative over $\mathbf{V}$-$\mathcal{K}$, it suffices to show that $\overline{\mathbf{V}\text{-}\mathcal{K}}$ proves $\varphi_{\mathcal{K}}(\Sigma_0^B)$-COMP. Let $\phi_1,\dots,\phi_s \in \Sigma_0^B$. First we use Σ_0^B-COMP to convert each occurrence of ϕ_i in $\varphi_{\mathcal{K}}$ into a string P_i such that

$$(\forall x < t_i)(x \in P_i \leftrightarrow \phi_i(x)).$$

Thus it suffices to show the COMP axiom for $\varphi_{\mathcal{K}}(n,P_1,\dots,P_s)$. By the Skolemized version of the axiom

$$(\forall n)(\forall P_1)\dots(\forall P_s)(\varphi_{\mathcal{K}}(n,P_1,\dots,P_s) \vee \psi_{\mathcal{K}}(n,P_1,\dots,P_s))$$

we have

$$(\forall n)(\forall P_1)\dots(\forall P_s)(\varphi^0_{\mathcal{K}}(n,P_1,\dots,P_s,F_{\mathcal{K}}(n,P_1,\dots,P_s)) \vee \psi_{\mathcal{K}}(n,P_1,\dots,P_s,F_{\mathcal{K}}(n,P_1,\dots,P_s)))$$

for a term of $\overline{\mathbf{V}\text{-}\mathcal{K}}$, where $\varphi^0_{\mathcal{K}},\psi^0_{\mathcal{K}} \in \Sigma_0^B$. Using the COMP axiom we obtain a string $Y < a$ such that

$$(\forall n < a)(n \in Y \leftrightarrow \varphi^0_{\mathcal{K}}(n,P_1,\dots,P_s,F_{\mathcal{K}}(n,P_1,\dots,P_s)).$$

Therefore

$$(\exists Y < a)(\forall n < a)(n \in Y \leftrightarrow \varphi_{\mathcal{K}}(n,P_1,\dots,P_s,)).$$

is provable in $\overline{\mathbf{V}\text{-}\mathcal{K}}$ and thus in $\mathbf{V}$-$\mathcal{K}$.

Now the second part of the theorem follows from the witnessing theorem of $\overline{\mathbf{V}\text{-}\mathcal{K}}$, namely,

Theorem 2. *Let $Q_{\mathcal{K}}(FO)$ capture the complexity class $\mathcal{C}$ over a given signature τ and has a Σ_1^B description $\varphi_{\mathcal{K}}(\Sigma_0^B)$. If $\varphi_{\mathcal{K}}(\Sigma_0^B)$ is strongly closed and self-witnessing then $\overline{\mathbf{V}\text{-}\mathcal{K}}$ captures $\mathcal{C}$.*

3 A theory based on acyclic conjunctive queries

As an application of the witnessing argument in the previous section, we will construct a theory based on the acyclic boolean conjunctive query problem (ABCQ) for which the following is proved:

Theorem 3 (Gottlob et.al. [1]). *ABCQ is complete for LOGCFL via AC^0 reductions. Furthermore, it remains complete even if all relations are restricted to binary, where LOGCFL is the class of predicates LOGSPACE reducible to context-free languages.*

We will use the binary version of ABCQ to formulate our theory. First we shall formulate the problem over finite structures of some signature. Let $\sigma_{db} = \{D,Q\}$ where D and Q are binary predicates. Intuitively, D and Q represent a database and a query respectively such that

$$D(x,y) \Leftrightarrow \langle x,y\rangle \text{ is a record for the relation in } D$$

and

$$Q(i,j) \Leftrightarrow \langle i,j\rangle \text{ is a conjunct in } Q.$$

Thus we will define a complete set

$$\mathcal{K}_{ABCQ} = \{\langle [n],Q,D\rangle \ : \ \text{the query } Q \text{ has a solution in the database } D\}.$$

Let Q_{ABCQ} be the Lindström quantifier for the set $\mathcal{K}_{ABCQ}$. Observing that ABCQ is complete for LOGCFL, it is not difficult to see that

Proposition 1. *The logic $Q_{ABCQ}(FO)$ captures LOGCFL over ordered structure.*

We can define the Σ_1^B description of $\mathcal{K}_{ABCQ}$ in the language L_2 as follows:

$$\begin{aligned}\varphi_{ABCQ}(n,\bar{c},\bar{R},\varphi_D,\varphi_Q) \Leftrightarrow\ & (\exists C)(\forall i)(\varphi_Q(C[i],C[i+1],n,\bar{c},\bar{R}) \wedge (\exists i)(\exists j)(C[i]=C[j]))\vee\\ & (\exists S)(\forall i)(\forall j)(\forall x)(\forall y)((\varphi_Q(i,j,n,\bar{c},\bar{R}) \rightarrow \varphi_D(S[i],S[j],n,\bar{c},\bar{R})).\end{aligned}$$

Definition 5. *The L_2-theory $\mathbf{V}$-Q_{ABCQ} consists of axioms BASIC together with $\varphi_{ABCQ}(\Sigma^B_0)$-COMP.*

Using the technique presented in the previous section we have the following:

Theorem 4. $\mathbf{V}$-Q_{ABCQ} *contains V^0 and has self-witnessing and strong closure properties. Thus $\mathbf{V}$-Q_{ABCQ} LOGCFL.*

The proof of the first part of Theorem 4 is obtained by formalizing some complexity theoretical results in $\mathbf{V}$-$Q_{\mathcal{K}}$. For instance, the closure of $Q_{ABCQ}(\Sigma^B_0)$ under complementation is obtained by defining SAC^1 circuits in $\mathbf{V}$-Q_{ABCQ}. From this we have

Proposition 2. $\mathbf{V}$-Q_{ABCQ} *contains* $\mathbf{V}$-Q_{SAC}.

where $\mathbf{V}$-Q_{SAC} is the theory for LOGCFL defined in [4] in which the closure of SAC^1 under complementation is proved.

For the self-witnessing property, we formalize the following theorem of Gottlob et.al.

Theorem 5 (Gottlob et.al. [2]). *Let M be a bounded treesize logspace ATM recognizing A. It is possible to construct a L^{LOGCFL} transducer T which outputs an accepting tree for M on input $w \in A$.*

This theorem implies that the witness for acyclic conjunctive query instances can be computed by functional LOGCFL.

In $\mathbf{V}$-Q_{ABCQ}, we can define the computation of a bounded treesize logspace ATM, and the argument in [2] can be formalized inside $\mathbf{V}$-Q_{ABCQ}.

4 Future Research

It was proved by Pollett [6] that the theory TLS for LOGSPACE cannot prove a statement which roughly means NLIN=co-NLIN. and conjectured that the result can be extended to a system for LOGCFL. So our system $\mathbf{V}$-Q_{ABCQ} shall be a nice candidate for Pollett's conjecture.

However, Pollett's proof depends on Nepomnjaščiĭ's Theorem which implies that $LOGSPACE \subseteq LinH$. Thus we may need $LOGSPACE$ to be replaced by $LOGCFL$ in order to apply Pollett's proof for $\mathbf{V}$-Q_{ABCQ}. But we do not know whether this is possible. So we may need a different technique to prove Pollett's conjecture.

References

1. G. Gottlob, N. Leone and F. Scarcello, The complexity of acyclic conjunctive queries. Journal of the ACM, 48(3). pp.431–498 (2001)
2. G. Gottlob, N. Leone, and F. Scarcello. Computing LOGCFL Certificates. Theoretical Computer Science, 270(1-2), pp.761-777, (2002)
3. A. Kolokolova, Systems of bounded arithmetic from descriptive complexity. PhD Thesis, Toronto University (2005).
4. S. Kuroda, Generalized quantifier and a bounded arithmetic theory for LOGCFL. Archive for Mathematical Logic. 46(5-6) pp.489–516 (2007)
5. P. Nguyen, Bounded Reverse Mathematics, PhD Thesis, Toronto University (2008).
6. C. Pollett, A Theory for Logspace and NLIN versus co-NLIN. Journal of Symbolic Logic. 68(4), pp.1082–1090 (2003).

The Isomorphism Problem On Classes of Automatic Structures

Dietrich Kuske[1], Jiamou Liu[2], and Markus Lohrey[2,] *

[1] LaBRI, CNRS and Université Bordeaux I, France
[2] Universität Leipzig, Institut für Informatik, Germany
kuske@labri.fr, liujiamou@gmail.com, lohrey@informatik.uni-leipzig.de

1 Introduction

The idea of an automatic structure goes back to Büchi and Elgot who used finite automata to decide, e.g., Presburger arithmetic [3]. A systematic study was initiated by Khoussainov and Nerode [7] who also coined the name "*automatic structure*". In essence, a structure is automatic if the elements of the universe can be represented as strings from a regular language and every relation of the structure can be recognized by a finite state automaton with several heads that proceed synchronously. Automatic structures received increasing interest over the last years, see e.g. the recent survey [16]. One of the main motivations for investigating automatic structures is that every automatic structure has a decidable first-order theory, this holds even for the extension of first-order logic by the certain generalized quantifiers (infinity [1], modulo [10], Ramsey [16]) and a severely restricted form of second-order quantification [11].

Automatic structures form a subclass of recursive (or computable) structures. A structure is recursive, if its domain as well as all relations are recursive sets of finite words (or naturals). A well-studied problem for recursive structures is the isomorphism problem, where it is asked whether two given recursive structures over the same signature (written down by Turing-machines for the domain and all relations) are isomorphic. It is well known that the isomorphism problem for recursive structures is complete for the first level of the analytical hierarchy Σ^1_1. In fact, Σ^1_1-completeness holds for many subclasses of recursive structures, e.g., for linear orders, trees, undirected graphs, Boolean algebras, Abelian p-groups, see e.g. [2, 4].

In [8], it was shown that also for automatic structures the isomorphism problem is Σ^1_1-complete. By a direct interpretation, it follows that for the following classes the isomorphism problem is still Σ^1_1-complete [13]: automatic successor trees, automatic undirected graphs, automatic commutative monoids, automatic partial orders, automatic lattices of height 4, and automatic 1-ary functions. On the other hand, the isomorphism problem is decidable for automatic ordinals [9] and automatic Boolean algebras [8]. An intermediate class is the class of all locally-finite automatic graphs, for which the isomorphism problem is Π^0_3-complete [15].[3]

For many interesting classes of automatic structures, the exact status of the isomorphism problem is still open. In the recent survey [16] it was asked for instance, whether the isomorphism problem is decidable for automatic equivalence relations and automatic linear orders. The same question was already asked in [9] for linear orders. In this abstract, we answer these questions negatively. Our main results are:

- The isomorphism problem for automatic equivalence relations is Π^0_1-complete.
- The isomorphism problem for automatic successor trees of finite height $k \geq 2$ is Π^0_{2k-3}-complete.
- The isomorphism problem for automatic linear orders is hard for level ω of the hyperarithmetical hierarchy (and therefore for every level of the arithmetical hierarchy).

Most hardness proofs from the literature for automatic structures, in particular the Σ^1_1-hardness proof for the isomorphism problem of automatic structures from [8], use transition graphs of Turing-machines, which are automatic. This technique seems to fail for inherent reasons, when trying to prove our news results. The reason is most obvious for equivalence relations and linear orders. These structures are transitive but the transitive closure of the transition graph of a Turing-machine cannot be automatic in general (it's first-order theory is undecidable in general). Hence, we have to use a new strategy. Our proofs are based on the undecidability of Hilbert's 10th problem. Recall that Matiyasevich showed that

* The second and third author are supported by the DFG research project GELO.
[3] For background on the arithmetical hierarchy see [14].

every recursively enumerable set of natural numbers is Diophantine [12]. This fact was used by Honkala to show that it is undecidable whether the range of a rational power series is $\mathbb{N}$ [5]. Using a similar encoding, we show that the isomorphism problem for automatic equivalence relations is Π^0_1-complete. Next, we extend our technique in order to show that the isomorphism problem for automatic successor trees of height $k \geq 2$ is Π^0_{2k-3}-complete. Finally, using a similar but technically more involved reduction, we can show that the isomorphism problem for automatic linear orders is hard for every level of the arithmetical hierarchy. In fact, since our proof is uniform in the level for the arithmetical hierarchy, we immediately get hardness for level Σ^0_ω of the hyperarithmetical hierarchy. In other words: the isomorphism problem for automatic linear orders is at least as hard as the the first-order theory of $(\mathbb{N}; +, \times)$. At the moment it remains open whether the isomorphism problem for automatic linear orders is Σ^1_1-complete.

2 Preliminaries

We use *synchronous n-tape automata* to recognize n-ary relations. Such automata have n input tapes, each of which contains one of the input words. The n tapes are read in parallel until all input words are processed. Formally, let $\Sigma_\diamond = \Sigma \cup \{\diamond\}$ where $\diamond \notin \Sigma$. For words $w_1, w_2, \ldots, w_n \in \Sigma^\star$, their *convolution* is a word $w_1 \otimes \cdots \otimes w_n \in (\Sigma^n_\diamond)^\star$ with length $\max\{|w_1|, \ldots, |w_n|\}$, and the k^{th} symbol of $w_1 \otimes \cdots \otimes w_n$ is $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the k^{th} symbol of w_i if $k \leq |w_i|$, and $\sigma_i = \diamond$ otherwise. An n-ary relation R is *FA recognizable* if the set of convolutions of all tuples $(w_1, \ldots, w_n) \in R$ is regular.

A *relational structure* $\mathcal{S}$ consists of a *domain* D and atomic relations on the set D. A structure is *automatic* if its domain is a regular language and each of its atomic relations is FA recognizable. If an automatic structure $\mathcal{A}$ is isomorphic to a structure $\mathcal{B}$, then $\mathcal{A}$ is called an *automatic presentation* of $\mathcal{B}$ and $\mathcal{B}$ is *automatically presentable*. In this paper we sometimes abuse the terminology and refer to $\mathcal{B}$ as simply automatic. The structures $(\mathbb{N}; \leq, +)$ and $(\mathbb{Q}; \leq)$ are both automatic. On the other hand, e.g., $(\mathbb{N}; \times)$ and any free monoid of rank at least 2 have no automatic presentation, see e.g. [16].

Let $\mathcal{K}$ be a class of (finitely presented) structures. The *isomorphism problem* for $\mathcal{K}$ asks whether $\mathcal{A} \cong \mathcal{B}$ for two given structures $\mathcal{A}, \mathcal{B} \in \mathcal{K}$.

3 Main results

A structure $(D; E)$ with E an equivalence relation on D is an *equivalence structure*.

Theorem 1. *The isomorphism problem for automatic equivalence structures is Π^0_1-complete.*

Proof. For the upper bound, let $\mathcal{E}$ be an automatic equivalence structure. Define the function $h_\mathcal{E} : \mathbb{N} \cup \{\infty\} \to \mathbb{N} \cup \{\infty\}$ such that for all $n \in \mathbb{N} \cup \{\infty\}$, $h_\mathcal{E}(n)$ equals the number of equivalence classes (possibly infinite) in $\mathcal{E}$ of size n. Note that for given $n \in \mathbb{N} \cup \{\infty\}$, the value $h_\mathcal{E}(n)$ can be computed effectively: one can define in first-order logic with the infinity quantifier the set of all $\leq_{llex}$-least elements[4] that belong to an equivalence class of size n. Hence this set is effectively regular and its size can be computed. Now, given two automatic equivalence structures $\mathcal{E}_1 = (D_1; E_1)$ and $\mathcal{E}_2 = (D_2; E_2)$, deciding if $\mathcal{E}_1 \cong \mathcal{E}_2$ amounts to checking if $h_{\mathcal{E}_1} = h_{\mathcal{E}_2}$. Therefore, the isomorphism problem for automatic equivalence structures is in Π^0_1.

We prove Π^0_1-hardness using Hilbert's 10th problem. Let $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. For a polynomial $p(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$ let $\mathrm{Img}_+(p) = \{p(n_1, \ldots, n_k) \mid n_1, \ldots, n_k \in \mathbb{N}_+\}$. Matiyasevich constructed from a given (index of a) recursively enumerable set $X \subseteq \mathbb{N}_+$ a polynomial $p(x_1, \ldots, x_k) \in \mathbb{Z}[x_1, \ldots, x_k]$ such that $X = \{n \in \mathbb{N}_+ \mid \exists y_2, \ldots, y_k \in \mathbb{N}_+ : p(n, y_2, \ldots, y_k) = 0\}$, see e.g. [12]. Hence, the following problem is Π^0_1-complete: Given two polynomials $p_1(x_1, \ldots, x_k), p_2(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$, is $p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$ for all $x_1, \ldots, x_k \in \mathbb{N}_+$?

Before we can reduce this problem to the isomorphism problem of automatic equivalence relations, we first construct, from a polynomial $p \in \mathbb{N}[x_1, \ldots, x_k]$ an automatic equivalence relation $\mathcal{E}(p)$ such that

$$\forall n \in \mathbb{N}_+ : h_{\mathcal{E}(p)}(n) > 0 \iff n \in \mathrm{Img}_+(p). \tag{1}$$

By induction on the structure of the polynomial $p(\overline{x})$ one can show:

[4] $\leq_{llex}$ denotes the length-lexicographical order on words.

Lemma 1. *There exists an algorithm that, given a polynomial $p(\overline{x}) \in \mathbb{N}[\overline{x}]$, constructs a nondeterministic automaton $\mathcal{A}[p(\overline{x})]$ on alphabet $\{a_1, \ldots, a_k\}$ such that $\mathcal{A}[p(\overline{x})]$ has exactly $p(x_1, \ldots, x_k)$ many accepting runs on input $a_1^{x_1} a_2^{x_2} \cdots a_k^{x_k}$.*

Let $\mathcal{A} = (S, I, \Delta, F)$ be a nondeterministic finite automaton with alphabet Σ ($\Delta \subseteq S \times \Sigma \times S$ is the set of transitions, $I \subseteq S$ is the set of initial states, and $F \subseteq S$ is the set of final states). We define an automaton $\mathrm{Run}_{\mathcal{A}} = (S, I, \Delta', F)$ with alphabet Δ such that the transition relation $\Delta' \subseteq S \times \Delta \times S$ is defined as:

$$\Delta' = \{(p, (p, a, q), q) \mid (p, a, q) \in \Delta\}.$$

Let $\pi : \Delta^* \to \Sigma^*$ be the projection morphism with $\pi(p, a, q) = a$. The following lemma is immediate from the definition.

Lemma 2. *For $u \in \Delta^+$ we have: $u \in L(\mathrm{Run}_{\mathcal{A}})$ if and only if $\pi(u) \in L(\mathcal{A})$ and u forms an accepting run of $\mathcal{A}$ on $\pi(u)$.*

This lemma implies that for all words $w \in \Sigma^+$, $|\pi^{-1}(w) \cap L(\mathrm{Run}_{\mathcal{A}})|$ equals to the number of accepting runs of $\mathcal{A}$ on w. Note that this does not hold for $w = \varepsilon$.

Consider now a non-zero polynomial $p(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$. Let the automaton $\mathcal{A} = \mathcal{A}[p(\overline{x})]$ be as defined in Lemma 1 and $\mathrm{Run}_{\mathcal{A}}$ be as defined above. Define an automatic equivalence structure $\mathcal{E}(p)$ whose domain is $\pi^{-1}(a_1^+ a_2^+ \cdots a_k^+) \cap L(\mathrm{Run}_{\mathcal{A}})$. Moreover, two words $u, v \in \pi^{-1}(a_1^+ a_2^+ \cdots a_k^+) \cap L(\mathrm{Run}_{\mathcal{A}})$ are equivalent if and only if $\pi(u) = \pi(v)$. By definition and Lemma 1, a natural number y belongs to $\mathrm{Img}_+(p)$ if and only if $\mathcal{E}(p)$ contains an equivalence class of size y, i.e., $\mathcal{E}(p)$ satisfies Condition (1).

After these preparations, we can prove Π_1^0-hardness. The function $C : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with $C(x, y) = (x + y)^2 + 3x + y$ is injective ($C(x, y)/2$ is a well-known pairing function). For $p_1(\overline{x}), p_2(\overline{x}) \in \mathbb{N}[\overline{x}]$, define the following three polynomials over $\mathbb{N}$:

- $S_1(\overline{x}) = C(p_1(\overline{x}), p_2(\overline{x}))$
- $S_2(x_1, x_2) = C(x_1 + x_2, x_1)$
- $S_3(x_1, x_2) = C(x_1, x_1 + x_2)$

Let $\mathcal{E}(S_1)$, $\mathcal{E}(S_2)$, $\mathcal{E}(S_3)$ be automatic equivalence structures as defined above. Let $\mathcal{E}_1$ be the disjoint union of these three equivalence structures and let $\mathcal{E}_2$ be the disjoint union of $\mathcal{E}(S_2)$ and $\mathcal{E}(S_3)$ only. For $i \in \{1, 2\}$, the ω-sum of $\mathcal{E}_i$ is the automatic equivalence structure $\mathcal{E}_i^\omega$ consisting of $\aleph_0$ many copies of $\mathcal{E}_i$.

If $p_1(x_1, \ldots, x_k) = p_2(x_1, \ldots, x_k)$ for some $x_1, \ldots, x_k \in \mathbb{N}_+$, then there is $y \in \mathbb{N}_+$ such that $C(y, y)$ belongs to $\mathrm{Img}_+(S_1)$. Therefore in $\mathcal{E}_1$ there is an equivalence class of size $C(y, y)$. On the other hand, for any $z \in \mathbb{N}_+$, $C(z, z)$ does not belong to $\mathrm{Img}_+(S_2) \cup \mathrm{Img}_+(S_3)$ and therefore $\mathcal{E}_2$ does not contain an equivalence class of size $C(y, y)$. Hence $\mathcal{E}_1^\omega \ncong \mathcal{E}_2^\omega$.

Conversely, suppose that $p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$ for all $x_1, \ldots, x_k \in \mathbb{N}_+$. Then $C(y, z)$ belongs to $\mathrm{Img}_+(S_1) \cup \mathrm{Img}_+(S_2) \cup \mathrm{Img}_+(S_3)$ if and only if $y \neq z$, if and only if $C(y, z)$ belongs to $\mathrm{Img}_+(S_2) \cup \mathrm{Img}_+(S_3)$. Therefore for any $s \in \mathbb{N}_+$, $\mathcal{E}_1$ contains an equivalence class of size s if and only if $\mathcal{E}_2$ contains an equivalence class of size s. Hence $\mathcal{E}_1^\omega \cong \mathcal{E}_2^\omega$. In summary, we have shown that $\forall x_1, \ldots, x_k \in \mathbb{N}_+^k$: $p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$ if and only if $\mathcal{E}_1^\omega \cong \mathcal{E}_2^\omega$. Hence Theorem 1 is proved. □

A *tree* is a structure $T = (V; \leq_T)$, where $\leq_T$ is a partial order with a least element, called the *root*, and such that for every $x \in V$, $\leq_T$ restricted to $\{y \mid y \leq_T x\}$ is a finite linear order. One may also view a tree as a directed graph (V, E) with an edge $(u, v) \in E$ if and only if u is the largest element in $\{x \mid x <_T v\}$. The edge relation E is FO-definable in $(V; \leq_T)$. The *level* of a node $u \in V$ is the number of E-edges along the path from the root to u. The *height* of T is the supremum of all levels of nodes in V; it may be infinite. Here we only deal with trees of finite height. We use $\mathcal{T}_n$ to denote the class of automatic trees with height at most n. Note that when n is fixed, the tree order $\leq_T$ when restricted to the class $\mathcal{T}_n$ is FO-definable in (V, E).

Using the uniform decidability of first-order logic with the infinity-quantifier over automatic structures, it is easy to show decidability of the isomorphism problem for the class $\mathcal{T}_1$. The case $n = 2$ of the following theorem can be deduced from Theorem 1 using an automaticity-preserving interpretation, the remaining proof works by induction on n.

Theorem 2. *For all $n \geq 2$, the isomorphism problem for the class $\mathcal{T}_n$ of automatic trees is Π_{2n-3}^0-complete.*

Our construction in the proof of Theorem 2 is uniform in the sense that given $i \in \mathbb{N}$ and a Π^0_i set R, for any $x \in \mathbb{N}$, we can effectively construct two automatic trees $T_1(R,x)$ and $T_2(R,x)$ of finite height, such that $x \in R$ if and only if $T_1(R,x) \cong T_2(R,x)$. This uniformity gives us the following corollary. (For a precise definition of the hyperarithmetical hierarchy see for instance [14]. A typical problem on level Σ^0_ω of the hyperarithmetical hierarchy is the first-order theory of $(\mathbb{N}; +, \times)$.)

Corollary 1. *The isomorphism problem for automatic trees of finite height is Σ^0_ω-complete.*

Given a recursive tree T without infinite paths, one can assign a recursive ordinal $\xi(v)$ to every node v as follows: Every leaf of T is labeled with the ordinal 0. A node with children $\{v_i \mid i \in I\}$ is labeled with the ordinal $\sup\{\xi(v_i) + 1 \mid i \in I\}$. Let $\mathcal{T}_\xi$ the class of all automatic trees, for which the root is labeled with an ordinal $\alpha < \xi$. We conjecture that for every recursive ordinal ξ, the isomorphism problem for $\mathcal{T}_\xi$ is complete for a suitable level of the hyperarithmetical hierarchy.

The following theorem can be shown using similar but technically more involved arguments than those used for the proof of Theorem 2.

Theorem 3. *The isomorphism problem for automatic linear orders is Σ^0_ω-hard.*

Proofs for Theorem 2 and 3 will appear in a long version of this abstract.

References

1. A. Blumensath and E. Grädel. Automatic Structures. In *LICS'00*, pages 51–62. IEEE Computer Society Press, 2000.
2. W. Calvert and J. F. Knight. Classification from a computable viewpoint. *Bull. Symbolic Logic*, 12(2):191–218, 2006.
3. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98:21–51, 1961.
4. S. S. Goncharov and J. F. Knight. Computable structure and antistructure theorems. *Algebra Logika*, 41(6):639–681, 2002.
5. J. Honkala. On the problem whether the image of an N-rational series equals N. *Fund. Inform.*, 73(1-2):127–132, 2006.
6. B. Khoussainov and M. Minnes. Model theoretic complexity of automatic structures. In *Proc. TAMC 08*, LNCS 4978, 514–525, Springer, 2008.
7. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC'95*, LNCS 960, 367–392, Springer, 1995.
8. B. Khoussainov, A. Nies, S. Rubin, and F. Stephan. Automatic structures: richness and limitations. *Logical Methods in Computer Science*, 3(2):2:2, 18 pp. (electronic), 2007.
9. B. Khoussainov, S. Rubin, and F. Stephan. Automatic linear orders and trees. *ACM Transactions on Computational Logic*, 6(4):675–700, 2005.
10. B. Khoussainov, S. Rubin, and F. Stephan. Definability and regularity in automatic structures. In *STACS'04*, Lecture Notes in Comp. Science vol. 2996, pages 440–451. Springer, 2004.
11. D. Kuske and M. Lohrey. Some natural problems in automatic graphs. *Journal of Symbolic Logic*, 2009. Accepted.
12. Y. V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, Cambridge, Massachusetts, 1993.
13. A. Nies. Describing groups. *Bull. Symbolic Logic*, 13(3):305–339, 2007.
14. H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1968.
15. S. Rubin. *Automatic Structures*. PhD thesis, University of Auckland, 2004.
16. S. Rubin. Automata presenting structures: A survey of the finite string case. *Bull. Symbolic Logic*, 14:169–209, 2008.

Classification of the classes of finite models in Tarski' style

Marcin Mostowski

Department of Logic,
Institute of Philosophy, Warsaw University
m.mostowski@uw.edu.pl

This work is devoted to the truth definitions method of notions in finite models. The general method — working in infinite models only — was invented by Alfred Tarski [Tar33]. The method of FM–truth definitions [1] was introduced in [Mos01], and further investigated in [Mos03], [Koł04b], [Koł04a], [Koł05].

We discuss here some improvements of the method and we give a few important examples of FM–truth definitions.

1 Background

1.1 Logics

We compare logics on ordered finite models by the relation $\leq$ as follows. $L \leq L'$ if for each L–sentence φ there is L'–sentence ψ such that φ and ψ are equivalent on finite ordered models. We say that L is strictly contained in L' ($L < L'$) if $L \leq L'$ but not $L' \leq L$. We say that L and L' are equivalent ($L \equiv L'$) if $L \leq L'$ and $L' \leq L$.

1.2 Finite arithmetics

We consider arithmetical models of finite initial segments of natural numbers. [2]

We need the following two results.

Theorem 1. *For each n, the arithmetical model on the universe $\{0, 1, \ldots, n^k\}$ is first order definable in the model on the universe $\{0, 1, \ldots, n\}$, for any $k \geq 1$.*

Theorem 2. *All the arithmetical relations are definable in finite models in terms of* **BIT** *relation (***BIT**(x, y) *means that y–th bit in binary representation of x is 1). Moreover* **BIT** *is arithmetically definable in finite models.*

Combining these theorems we can think of arithmetical models in higher order logics as larger arithmetical models, but in first order logic. For instance a finite model of the size n considered from the point of view of monadic second order logic can be considered as an arithmetical model of the size 2^n from the point of view of first order logic.

1.3 FM–representability

The key notion in applications of the method considered is the notion of **FM**–representability. A relation $R \subseteq \mathbb{N}^k$ is **FM**–representable by an arithmetical formula $\varphi(x_1, \ldots, x_k)$ if for each k–tuple $a_, \ldots, a_k \in \mathbb{N}$ there is m such that the following two conditions are satisfied:

- if $R(a_, \ldots, a_k)$ then $\varphi(a_, \ldots, a_k)$ is true in all finite models greater than m;
- if $\neg R(a_, \ldots, a_k)$ then $\varphi(a_, \ldots, a_k)$ is false in all finite models greater than m.

Each function choosing m for a given k–tuple $a_, \ldots, a_k \in \mathbb{N}$ is called a witnessing function.

R is **FM**–representable if it is **FM**–representable by some arithmetical formula.

[1] FM is taken here from Finite Models.

[2] All the relevant results can be found in [KMK07].

Theorem 3 (FM–representability theorem, [Mos01]). *For each $R \subseteq \mathbb{N}^k$, R is* **FM***–representable if and only if R is recursive with recursively enumerable oracle, or equivalently: R is Δ^0_2 arithmetically definable.*

Moreover if R is functional then R is **FM***–representable by a formula $\varphi(x_1, \ldots, x_k)$ satisfying the uniqueness condition:*

$$\forall x_1, \ldots, \forall x_k \forall x'_k(\varphi(x_1, \ldots, x_k) \wedge \varphi(x_1, \ldots, x'_k) \Rightarrow x_k = x'_k).$$

A relation $R \subseteq \mathbb{N}^k$ is arithmetically definable in finite models if there is a formula $\varphi(x_1, \ldots, x_k)$ if for each m and $a_, \ldots, a_k < m$:

$R(a_, \ldots, a_k)$ if and only if $M \models \varphi(a_, \ldots, a_k)$,

provided the size of M is at least m.

From theorem 1 the following follows.

Theorem 4. *If $R \subseteq \mathbb{N}^k$ is* **FM***–representable with a polynomial witnessing function then R is arithmetically definable*

As a corollary we obtain that all context free languges as well as a suitable substitution function (treated as a ternary relation) are arithmetically definable in finite models.

2 FM truth definitions

All the finite models considered here are equipped with the standard arithmetical notions, such as addition, multiplication, ordering, zero element, and the successor operation. The successor S loops at the maximal element.[3]

We say that a formula $\varphi(x)$ is **FM** truth definition for a set of sentences Z if for each $\psi \in Z$ the equivalence $(\psi \equiv \varphi(\ulcorner\psi\urcorner))$ holds in almost all finite ordered models.[4] The expression $\ulcorner\psi\urcorner$ denotes here the Gödel number of ψ.

If there is **FM** truth definition in L' for all L–sentences then we write $L \ll L'$. Of course concrete truth definitions depend on vocabularies. If there is **FM** truth definition of vocabulary σ in L' for all L–sentences of vocabulary σ then we write $L \ll_\sigma L'$.

All the logics considered in this paper are closed on first order constructions. Additionally for all of them the relation *truth in a finite model* is recursive.

The main applications of **FM** truth definitions are based on the following.

Theorem 5 ([Mos01]). *Let L' be any logic and L be a logic closed on first order constructions. If $L \ll L'$ then $L < L'$.*[5]

Some non–equivalence results can be obtained by means of a slightly weaker notion. We say that a formula $\varphi(x)$ is a weak **FM** truth definition for a set of sentences Z if for each $\psi \in Z$ the equivalence $(\psi \equiv \varphi(\ulcorner\psi\urcorner))$ holds in infinitely many finite ordered models. The difference is that we do not require here that the equivalence $(\psi \equiv \varphi(\ulcorner\psi\urcorner))$ has only finitely many exceptions. Moreover, for different formulae ψ it can hold in different finite models. For any logics L and L', if there is a weak FM–truth definition in L' for L–sentences then we write $L \ll_w L'$.

Theorem 6. *Let L' be any logic and L be a logic closed on first order constructions. If $L \ll_w L'$ then $L \not\equiv L'$.*

Essentially the proof is the same as that of theorem 5. We use the following finite version of the diagonalization lemma. We give it here in an improved form based on theorem 4.

Theorem 7. (The finite version of the diagonalization lemma, [Mos01]) *For each logic L closed on first order constructions, and each arithmetical L–formula $\varphi(x)$ with one free variable x there is L–sentence ψ such that the equivalence $(\psi \equiv \varphi(\ulcorner\psi\urcorner))$ is true in all models of the size at least $\ulcorner\psi\urcorner$.*

Morover the formula $\varphi(x)$ occurs only once in ψ, and this occurence is positive. More precisely, ψ can be taken as a formula of the form $\exists x(\xi(x) \wedge \varphi(x))$, where $\xi(x)$ is a first order arithmetical formula.

[3] This is only for treating S as a real function. Addition and multiplication can be treated as ternary relations.

[4] For broader discussion of this notion see [Mos01].

[5] The ambiguity here is not essential because if $L < L'$ holds on ordered models then it holds on arbitrary models. The opposite implication does not hold in general.

3 Existence of FM–truth definitions

As a general application of theorem 3 we obtain the observation by Leszek Kołodziejczyk that the relation $\ll$ is essentially syntax independent.[6]

Theorem 8 ([Koł04b], [Koł05]). *Let L, L_1, L_2 be logics with recursive relation* truth in a finite model. *If $L \leq L_1$ and $L_1 \ll L_2$ then $L \ll L_2$.*

It follows from theorem 3 and the fact that if $L \leq L_1$ then there is Δ^0_2 arithmetically definable function f from L–sentences to L_1–sentences such that φ and $f(\varphi)$ are equivalent on finite models.

In this section we consider a few interesting examples.

3.1 First order hierarchy

For the standard first order hierarchy we have the following.

Theorem 9.
- $\Sigma^0_1 \ll \Sigma^0_1$,
- $\Pi^0_1 \ll \Pi^0_1$,
- $Bool(\Sigma^0_1) \ll \Sigma^0_2$,
- $Bool(\Sigma^0_1) \ll \Pi^0_2$,
- $Bool(\Sigma^0_1) \ll \Delta^0_2$.

4 Transitive Closure logics

It is known that $FO(\mathsf{PFP})$ captures **PSPACE**. However, in this work, instead of $FO(\mathsf{PFP})$, we operate on $FO(\mathsf{TC}_2)$ — the first order logic enriched by second order transitive closure operators TC^k_2 bounding two second order variables P_1 and P_2 of arity k and giving the transitive closure of the relation determined by a formula between the relations denoted by P_1 and P_2. The second order transitive closure operator on ordered models captures **PSPACE** by the same argument as that for $FO(\mathsf{PFP})$, see [Imm99]. Therefore we have the following.

Theorem 10. $FO(\mathsf{TC}_2)$ *captures* **PSPACE** *on ordered models, and therefore it is equivalent to* $FO(\mathsf{PFP})$.

By a natural construction we obtain the following.

Theorem 11.

$$FO(\mathsf{TC}) \ll FO(pos\mathsf{TC}^1_2).$$

Considering suitable refinements of the above idea we consider sublogics of the finite order logic with transitive closure operators of arbitrary finite types. These sublogics capture all natural space complexity classes over **NLOGSPACE**. Then, by suitable refinements of theorem 11, we obtain the best known version of the space hierarchy theorem.

4.1 $FO \ll FO(\mathsf{TC})$?

The problem "$FO \ll FO(\mathsf{TC})$?' seems to be very difficult. Nevertheless we will consider consequences of positive and negative answers.

assuming $FO \not\ll FO(\mathsf{TC})$ In this case **NLOGSPACE** $<$ **NP**. This is so because the logic $FO(\mathsf{TC}_2)$ cannot define truth for SO. Assuming **NLOGSPACE** $=$ **NP**, the logics $FO(\mathsf{TC})$ and Σ^1_1 should be equivalent. However this is impossible because $FO(\mathsf{TC})$ can be separated from $FO(\mathsf{TC}_2)$.

[6] Originally, the notion of FM–representability was motivated by the proof of the diagonalization lemma. The improved version presented here uses simple definability instead. Nevertheless the theorem of Kołodziejczyk uses the full power of FM–represantibility.

assuming $FO \ll FO(\mathsf{TC})$ In this case we have *min–max* $FO(\mathsf{TC})$ formula defining FM–truth for all first order formulae in empty vocabulary. This formula uses TC operator bounding $2k$ variables. It can be transformed into a *min–max* formula with the TC operator bounding 2 variables which define FM–truth on models of k–th power. Then $FO \ll_w FO(\mathsf{TC}^1_1)$.

5 Some Philosophical Remarks

Our way of thinking about semantics in finite conforms with Mycielski's approach to foundations of mathematics as presented in [Myc81]. In this paper he considers foundations of mathematics without actual infinity.

Potentially infinite domain of natural numbers consists with

$\{0\}$
$\{0,1\}$
$\{0,1,2\}$
$\{0,1,2,3\}$ ...

On the other hand actually infinite domain of natural numbers is

$$\{0,1,2,3,\dots\}.$$

We consider semantics in Tarski' style in potentially infinite domain. Originally Tarski [Tar33,Tar56] considered his truth definitions in actually infinte domains. Nevertheless it was discussed in his times how far the assuption of actual infinity was essential for defining truth. Our consideration prove that actual infinity can be removed in a reasonable way.

References

[Imm99] N. Immerman. *Descriptive Complexity.* Springer Verlag, 1999.

[KMK07] M. Krynicki, M. Mostowski, and Zdanowski K. Finite arithmetics. *Fundamenta Informaticae*, 81(1–3):183–202, 2007.

[Koł04a] L. Kołodziejczyk. A finite model-theoretical proof of a property of bounded query classes within ph. *The Journal of Symbolic Logic*, 69:1105–1116, 2004.

[Koł04b] L. Kołodziejczyk. Truth definitions in finite models. *The Journal of Symbolic Logic*, 69:183–200, 2004.

[Koł05] L. Kołodziejczyk. *Truth definitions and higher order logics in finite models.* PhD thesis, Warsaw University, 2005.

[Mos01] M. Mostowski. On representing concepts in finite models. *Mathematical Logic Quarterly*, 47:513–523, 2001.

[Mos03] M. Mostowski. On representing semantics in finite models. In A. Rojszczak†, J. Cachro, and G. Kurczewski, editors, *Philosophical Dimensions of Logic and Science*, pages 15–28. Kluwer Academic Publishers, 2003.

[Myc81] J. Mycielski. Analysis without actual infinity. *Journal of Symbolic Logic*, 46:625–633, 1981.

[Tar33] A. Tarski. *Pojęcie prawdy w językach nauk dedukcyjnych.* Nakładem Towarzystwa Naukowego Warszawskiego, 1933. English version in [Tar56].

[Tar56] A. Tarski. The concept of truth in formalized languages. In J. H. Woodger, editor, *Logic, semantics, metamathematics*, pages 152 – 278. Oxford at The Clarendon Press, 1956. translated from German by J. H. Woodger.

Some Results on Complexity of μ-calculus Evaluation in the Black-box Model

Paweł Parys*

University of Warsaw
ul. Banacha 2, 02-097 Warszawa, Poland
parys@mimuw.edu.pl

Abstract. We consider μ-calculus formulas in a normal form: after a prefix of fixed-point quantifiers follows a quantifier-free expression. We are interested in the problem of evaluating (model checking) of such formula in a powerset lattice. Assumptions about the quantifier-free part of the expression are the weakest possible: it can be any monotone function given by a black-box—we may only ask for its value for given arguments. As a first result we prove that when the lattice is fixed, the problem becomes polynomial. As a second result we show that any algorithm solving the problem has to ask at least about n^2 (namely $\Omega\left(\frac{n^2}{\log n}\right)$) queries to the function, even when the expression consists of one μ and one ν.

1 Introduction

Fast evaluation of μ-calculus expressions is one of the key problems in theoretical computer science. Although it is a very important problem and many people were working on it, no one could show any polynomial time algorithm. On the other hand the problem is in NP∩co-NP, so it may be very difficult to show any lower bound on the complexity. In such situation a natural direction of research is to slightly modify the assumptions and see whether the problem becomes easier.

We restrict ourselves to expressions in a quantifier-prefix normal form, namely

$$\theta_1 x_1.\theta_2 x_2 \dots \theta_d x_d.F(x_1,\dots,x_d), \tag{1}$$

where $\theta_i = \mu$ for odd i and $\theta_i = \nu$ for even i. We want to evaluate such expression in the powerset model or, equivalently, in the lattice $\{0,1\}^n$ with the order defined by $a_1 \dots a_n \le b_1 \dots b_n$ when $a_i \le b_i$ for all i. The function $F\colon \{0,1\}^{nd} \to \{0,1\}^n$ is an arbitrary monotone function and is given by a black-box (oracle) which evaluates the value of the function for given arguments.

First concentrate on the problem of polynomial *expression complexity*, i.e. complexity for fixed size of the model. We assume that the oracle representing the function answers in time t_F (in other words it is a computational procedure calculating the function in time t_F). To simplify the complexity formulas assume that $t_F \ge O(nd)$, i.e. that the procedure at least reads its arguments. A typical complexity, in which one can evaluate the expression (1) is $O(n^d \cdot t_F)$; this can be done by naive iterating [1]. We show that, using a slightly modified version of the naive iterating algorithm, the complexity can be $O\left(\binom{n+d}{d} \cdot t_F\right)$. For big n it does not improve anything, however for fixed n the complexity is equal to $O(d^n \cdot t_F)$, hence is polynomial in d. This is our first result, described in Section 2.

Theorem 1. *There is an algorithm which for any fixed model size n calculates the value of expression (1) in time polynomial in d and t_F, namely $O(d^n \cdot t_F)$.*

Our result slightly extends an unpublished result in [2]. The authors also get polynomial expression complexity, however using completely different techniques. Our result is stronger, since they consider only expressions in which F is given by a vectorial Boolean formula, not as an arbitrary function. Moreover their complexity is slightly higher: $O(d^{2n} \cdot |F|)$.

Our second result is an almost quadratic lower bound for $d = 2$. It was possible to achieve any lower bound thanks to the assumption that the algorithm may access the function F in just one way, by evaluating its value for given arguments. Moreover, we are not interested in the exact complexity,

* Work supported by Polish government grant no. N206 008 32/0810.

only in the number of queries to the function F. In other words we consider decision trees: each internal node of the tree is labeled by an argument, for which the function F should be checked, and each its child corresponds to a possible value of F for that argument. The tree has to determine the value of the expression (1): for each path from the root to a leaf there is at most one possible value of (1) for all functions which are consistent with the answers on that path. We are interested in the height of such trees, which justifies the following definition.

Definition 1. *For any natural number d and finite lattice L we define $num(d, L)$ as the minimal number of queries, which has to be asked by any algorithm correctly calculating expression (1) basing only on queries to the function $F\colon L^d \to L$.*

In this paper we consider only the case $d = 2$. We show that almost n^2 queries are necessary in that case. Precisely, we have the following result, described in Section 3.

Theorem 2. *For any natural n it holds $num(2, \{0,1\}^n) = \Omega\left(\frac{n^2}{\log n}\right)$.*

This result is a first step towards solving the general question, for any d. It shows that in the black-box model something may be proved. Earlier it was unknown even if for any d there are needed more than nd queries. Note that $num(1, \{0,1\}^n)$ is n and that in the case when all d fixed-point operators are μ (instead of alternating μ and ν) it is enough to do n queries. So the result gives an example of a situation where the alternation of fixed-point quantifiers μ and ν is provably more difficult than just one type of quantifiers μ or ν. Although it is widely believed that the alternation should be a source of algorithmic complexity, the author is not aware of any other result showing this phenomenon, except the result in [3].

Let us comment the way how the function F is given. We make the weakest possible assumptions: the function can be given by an arbitrary program. This is called a black-box model, and was introduced in [4]. In particular our formulation covers vectorial Boolean formulas, as well as modal formulas in a Kripke structure of size n. Moreover our framework is more general, since not every monotone function can be described by a modal formula of small size, even when it can be computed quickly by a procedure. Note that the algorithm in [4], working in time $O(n^{\lfloor d/2 \rfloor + 1} \cdot t_F)$, can also be applied to our setting. On the other hand the recent algorithms, from [5] working in time $O(m^{d/3})$ and from [6] working in time $m^{O(\sqrt{m})}$ (where $m \geq n$ depends on the size of F), use the parity games framework, hence require that F is given by a Boolean or modal formula of small size. This can be compared to models of sorting algorithms. One possible assumption is that the only way to access the data is to compare them. Then an $\Omega(n \log n)$ lower bound can be proved. Most of the sorting algorithms work in this framework. On the other hand, when the data can be accessed directly, faster algorithms are possible (like $O(n)$ for strings and $O(n \log \log n)$ for integers).

It is known that for a given structure an arbitrary μ-calculus formula can be converted to a formula of form (1) in polynomial time, see Section 2.7.4 in [7]. Hence, a polynomial algorithm evaluating expressions of form (1) immediately gives a polynomial algorithm for arbitrary expressions. However during this conversion one also needs to change the underlying structure to one of size nd, where d is the nesting level of fixed-point quantifiers. So, even when the original model has fixed size n, after the normalization the model can become very big, and our algorithm from Theorem 1 gives exponential complexity.

2 The algorithm with polynomial expression complexity

Below we present a general version of the well known iterating algorithm. The algorithm can be described by a series of recursive procedures, one for each fixed-point operator; the goal of a procedure $\mathsf{Calculate}_i(X_1, \ldots, X_{i-1})$ is to calculate $\theta_i x_i . \theta_{i+1} x_{i+1} \ldots \theta_d x_d . F(X_1, \ldots, X_{i-1}, x_i, \ldots, x_d)$.

```
Calculate_i(X_1, ..., X_{i-1}):
    X_i = Initialize_i(X_1, ..., X_{i-1})
    repeat
      X_i = Calculate_{i+1}(X_1, ..., X_i)
    until X_i stops changing
    return X_i
```

Moreover the most internal procedure $\mathsf{Calculate}_{d+1}(X_1, \ldots, X_d)$ simply returns $F(X_1, \ldots, X_d)$. To evaluate the whole expression we simply call $\mathsf{Calculate}_1()$.

Till now we have not specified the $\mathsf{Initialize}_i$ procedures. When they always return $00\ldots0$ for odd i and $11\ldots1$ for even i, we simply get the naive iterating algorithm from [1]. However we would like to make use of already done computations and start a iteration from values which are closer to the fixed-point. Of course we can not start from an arbitrary value. The following standard lemma gives conditions under which the computations are correct.

Lemma 1. *Assume that* $\mathsf{Initialize}_i(X_1, \ldots, X_{i-1})$ *for odd* i *returns either* $00\ldots0$ *or a result of a previous call to* $\mathsf{Calculate}_i(X'_1, \ldots, X'_{i-1})$ *for some* $X'_1 \leq X_1, \ldots, X'_{i-1} \leq X_{i-1}$ *and for even* i *either* $11\ldots1$ *or a result of a previous call to* $\mathsf{Calculate}_i(X'_1, \ldots, X'_{i-1})$ *for some* $X'_1 \geq X_1, \ldots, X'_{i-1} \geq X_{i-1}$. *Then the function* $\mathsf{Calculate}_i(X_1, \ldots, X_{i-1})$ *returns the correct result.*

So to speed up the algorithm we need to somehow remember already calculated values of expressions and use them later as a starting value, when the same expression for greater/smaller arguments is going to be calculated. Instead of remembering all the results calculated so far in some sophisticated data structure, we do a very simple trick. We simply take

$$\mathsf{Initialize}_i(X_1, \ldots, X_{i-1}) = \begin{cases} 00\ldots0 \text{ for } i = 1, \\ 11\ldots1 \text{ for } i = 2, \\ X_{i-2} \quad \text{for } i \geq 3. \end{cases} \tag{2}$$

It turns out that $\mathsf{Initialize}_i$ defined this way satisfies assumptions of Lemma 1, so the algorithm is correct. The complexity bound follows from a simple observation that arguments of each call to $\mathsf{Calculate}_{d+1}$ satisfy

$$X_1 \leq X_3 \leq \cdots \leq X_{d-3} \leq X_{d-1} \leq X_d \leq X_{d-2} \leq \cdots \leq X_4 \leq X_2.$$

The same chain of inequalities is true for the numbers b_i of bits of X_i set to 1. Moreover the sequence $b_1, \ldots, b_d$ during each call to $\mathsf{Calculate}_{d+1}$ differs from stage to stage, it always increases in some appropriately defined order. There are $\binom{n+d}{d}$ such sequences, hence the complexity is $O\left(\binom{n+d}{d} \cdot t_F\right)$.

3 Quadratic lower bound

In order to prove Theorem 2 we first introduce a lattice which is more convenient than $\{0,1\}^n$. Take the alphabet Γ_n consisting of letters γ_i for $1 \leq i \leq n^2$ and the alphabet $\Sigma_n = \{0,1\} \cup \Gamma_n$, with the following partial order on it: the letters γ_i are incomparable; the letter 0 is smaller than all other letters; the letter 1 is bigger than all other letters. We will be considering sequences of n such letters, i.e. the lattice is Σ_n^n. The order on the sequences is defined as previously: $a_1 \ldots a_n \leq b_1 \ldots b_n$ when $a_i \leq b_i$ for all i. The idea is that one letter of Σ_n^n may be encoded in $O(\log n)$ bits of $\{0,1\}^m$. Hence to show Theorem 2 it is enough to prove $num(d, \Sigma_n^n) \geq \Omega(n^2)$.

To prove it we define a family of monotone functions, which will be difficult to distinguish by the algorithm. A function $F_{z,\sigma} \colon \Sigma_n^{2n} \to \Sigma_n^n$ is parametrized by a sequence $z \in \Gamma_n^n$ and by a permutation $\sigma \colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ (note that z is from Γ_n^n, not from Σ_n^n, so it can not contain 0 or 1, just the letters γ_i). The result of $\mu y.\nu x.F_{z,\sigma}(y, x)$ will be z. In the following the i-th element of a sequence $x \in \Sigma_n^n$ is denoted by $x[i]$. A pair z, σ defines a sequence of values $y_0, \ldots, y_n$:

$$y_k[i] = \begin{cases} z[i] \text{ for } \sigma^{-1}(i) \leq k \\ 0 \quad \text{otherwise.} \end{cases}$$

In other words y_k is equal to z, but with some letters covered: they are 0 instead of the actual letter of z. In y_k there are k uncovered letters; the permutation σ defines the order, in which the letters are uncovered. Using this sequence of values we define the function. In some sense the values of the function are meaningful only for $y = y_k$, we define them first (assuming $y_{n+1} = y_n$):

$$F_{z,\sigma}(y_k, x)[i] = \begin{cases} 0 & \text{if } \forall_{j>i} x[j] \leq y_{k+1}[j] \text{ and } x[i] \not\geq y_{k+1}[i] \text{ (case 1)} \\ y_{k+1}[i] & \text{if } \forall_{j>i} x[j] \leq y_{k+1}[j] \text{ and } x[i] \geq y_{k+1}[i] \text{ (case 2)} \\ x[i] & \text{if } \exists_{j>i} x[j] \not\leq y_{k+1}[j] \text{ (case 3).} \end{cases}$$

For any other value y we look for the lowest possible k such that $y \leq y_k$ and we put $F_{z,\sigma}(y,x) = F_{z,\sigma}(y_k,x)$. When such k does not exists ($y \not\leq z$), we put $F_{z,\sigma}(y,x)[i] = 1$.

The intuition behind the functions is as follows. At each moment the algorithm knows only some k letters of z (at the beginning it do not know any letter). Then it may decide which letter of z it want to uncover in the next step. When it tries to uncover a letter at position $\sigma(k)$, it is successful; otherwise it has to try again at another position. For the worst function it has to try all possible $n-k$ positions to uncover one letter of z. This gives a quadratic bound. The algorithm may also try to guess a value of letter on some position; however there are n^2 different γ_i, so in the worst case it has to guess n^2 times until it will discover a correct letter. A more detailed analysis shows that the algorithm is not able to do anything else.

4 Concluding remarks

The detailed proofs of the results are contained in [8, 9], available on the author's web page.

There are two natural future directions of research. First, it is very interesting to study whether the polynomial expression complexity can be shown for arbitrary formulas (not being in the normalized form (1)), or whether the problem is then equivalent to model checking in an arbitrary model. The second goal is to get an exponential lower bound for an arbitrary number of fixed-point operator alternations in the formula.

References

1. Emerson, E.A., Lei, C.L.: Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In: LICS. (1986) 267–278
2. Niwiński, D.: Computing flat vectorial Boolean fixed points. Unpublished manuscript
3. Dawar, A., Kreutzer, S.: Generalising automaticity to modal properties of finite structures. Theor. Comput. Sci. **379**(1-2) (2007) 266–285
4. Long, D.E., Browne, A., Clarke, E.M., Jha, S., Marrero, W.R.: An improved algorithm for the evaluation of fixpoint expressions. In: CAV. (1994) 338–350
5. Schewe, S.: Solving parity games in big steps. In: FSTTCS. (2007) 449–460
6. Jurdzinski, M., Paterson, M., Zwick, U.: A deterministic subexponential algorithm for solving parity games. SIAM J. Comput. **38**(4) (2008) 1519–1532
7. Arnold, A., Niwiński, D.: Rudiments of μ-calculus. Elsevier (2001)
8. Parys, P.: Evaluation of normalized μ-calculus formulas is polynomial for fixed structure size. Unpublished manuscript
9. Parys, P.: Lower bound for evaluation of $\mu\nu$ fixpoint. In: FICS. (2009) 86–92

Triangular perplexity and a stairway to heaven

Mihai Prunescu[1;2]

(1) Brain Products, Freiburg, Germany; and (2) Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Bucharest, Romania.

Abstract. Sections 1 and 2 contain a survey of author's recent results in recurrent double sequences over finite sets. This class of objects is a Turing complete model of computation. Special recurrent double sequences can be deterministically produced by expansive systems of context-free substitutions. In Section 3 we present examples of such recurrent double sequences which are non-Frobenius and which incidentally contain a lot of triangles. The fact that those examples are non-Frobenius completes our very young sight over those structures. Section 4 deals with a recurrent double sequence called Stairway to Heaven. The sequence contains a repetitive pattern growing in arithmetic progression. This property permits us to prove that the sequence cannot be generated using expansive systems of context-free substitutions. This seems to be the first known counterexemple of such recurrent double sequence.

Key Words: recurrent double sequence, context-free substitution, Turing complete models of computation, Frobenius' automorphism, homomorphisms of finite abelian groups. A.M.S.-Classification: 05B45, 28A80, 03D03.

1 Introduction

This article belongs to a series dedicated to the study of recurrent double sequences over finite sets and their power of expression.

Definition 1. Let $(A, f, 1)$ be a finite structure with one ternary function f, and one constant 1. The recurrent double sequence $a : \mathbb{N} \times \mathbb{N} \to A$ starts with the values $a(i, 0) = a(0, j) = 1$ and satisfies the recurrence $a(i, j) = f(a(i-1, j), a(i-1, j-1), a(i, j-1))$.

In [6] the author studied the problem to decide if recurrent double sequences are ultimately zero or not, where $0 \in A$ is some other fixed constant. It is proved that this problem is undecidable even if restricted to binary functions with the recurrence $a(i, j) = f(a(i-1, j), a(i, j-1))$ which are, moreover, commutative. In the next statement we call an instance of the Halting Problem a pair (M, w) where M is a Turing machine and w is a word in M's alphabet.

Theorem 1. *To every instance (M, w) of the Halting Problem one can algorithmically associate a commutative finite algebra $\mathcal{A} = (A, f, 0, 1)$ such that the recurrent double sequence defined by $a(i, 0) = a(0, j) = 1$ and $a(i, j) = f(a(i-1, j), a(i, j-1))$ is ultimately zero if and only if for input w: (the machine M stops with cleared tape without having done any step in the negative side of the tape) or (the machine M makes at least one step in the negative side of its tape and the first time when M makes such a step the tape of M is cleared). Consequently, by the Theorem of Rice, it is undecidable if such (or the more general) recurrent double sequences are ultimately zero.*

The starting point of this research was an open problem related to a very special form of linear recurrent double sequences over prime finite fields, posed by Lakhtakia and Passoja in [5]. The author proved in [7] that if $A = \mathbb{F}_q$ is the finite field with q elements and $f(x, y, z) = x + my + z$, where $m \in \mathbb{F}_q$ is an arbitrary fixed element, $f(x, y, z)$ generates a self-similar pattern $(a(i, j))$. In the case when q is prime and so $\mathbb{F}_q = \mathbb{Z}/q\mathbb{Z}$ as ring of classes of remainders modulo q, the pattern can be also obtained by substitutions of type $x \to xB$, where B is the $q \times q$ matrix occurring as left upper minor in the recurrent double sequence. This fact is not explicitly stated in [7], but is very easy to see it applying the Kronecker product representation theorem from [7] in the case q prime, where the only one automorphism of Frobenius is the identity.

The most classical example of such recurrent double sequence is Pascal's Triangle modulo 2, called also Sierpinski's Gasket. As one remark, this recurrent double sequence given by $f(x, y, z) = x + z$ over

$\mathbb{F}_2$ can be obtained by substitutions starting with 1 at stage 0 and applying the rules $1 \to \begin{matrix} 1\,1 \\ 1\,0 \end{matrix}$ and $0 \to \begin{matrix} 0\,0 \\ 0\,0 \end{matrix}$, such that one substitutes all elements of stage n in order to achieve the stage $n+1$. The matrix of stage k is a square and has dimension 2^k.

The case analyzed above is "regular" in the sense that the substitution rules have the special form element $\to$ matrix. The author has been surprised to observe that a lot of other repetitive phaenomena in recurrent double sequences come from a more general kind of context-free substitution, matrix $\to$ matrix. This makes the subject of the next section. *However, we must observe here that the analogy between these recurrent double sequences and the context-free languages as defined by Noam Chomsky in [1] is quite limited, because our generation procedure is* deterministic.

For other results concerning self-similarity and automata see [4], [2], [9]. To visualize recurrent double sequences we use images obtained by interpreting the values as different colours. The list of colour correspondencies will be concretely given here only if is important for understanding some proof.

2 Expansive systems of context-free substitutions

The definitions and the results of this section appeared the first time in author's recent paper [8].

Definition 2. Let $(A, 1, f)$ be a finite structure with ternary function. Denote by R a recurrent double sequence according to Definition 1. Suppose that two natural numbers $x \geq 1$ and $s \geq 2$ have been fixed. For $n \geq 1$ denote by $R(n)$ the matrix $(a(i,j))$ with $0 \leq i,j \leq xs^{n-1} - 1$. Call f-matrix every $u \times v$ matrix $(b(i,j))$ with elements in A such that for all $1 \leq i < u$ and $1 \leq j < v$ one has $b(i,j) = f(b(i-1,j), b(i-1,j-1), b(i,j-1))$, where the indexes start with 0. A $xs^{n-1} \times xs^{n-1}$ matrix $(b(i,j))$ is $R(n)$ if and only if is a f-matrix and fulfills $b(i,0) = b(0,j) = 1$.

Definition 3. The fixed natural numbers $x \geq 1$ and $s \geq 2$ used in this definition can be considered to be the same as x and s used in the Definition 2. Let $y = xs$. Let $\mathcal{X}$ be a finite set of $x \times x$ matrices over A and $\mathcal{Y}$ be a set of $y \times y$ matrices over A such that every $Y \in \mathcal{Y}$ has a $s \times s$ block matrix representation $(X(i,j))_{0 \leq i,j < s}$ and all blocks $X(i,j) \in \mathcal{X}$. We call system of (context-free) substitutions of type $x \to y$ the tuple $(\mathcal{X}, \mathcal{Y}, \Sigma, X_1)$, where $\Sigma : \mathcal{X} \to \mathcal{Y}$ is a fixed function and $X_1 \in \mathcal{X}$ is a fixed element of $\mathcal{X}$, called start-symbol. If a $u \times v$ matrix Z consists only of neighbouring blocks $X(i,j) \in \mathcal{X}$, $Z = (X(i,j))_{0 \leq i < u,\ 0 \leq j < v}$, we define $\Sigma(Z)$ to be the $su \times sv$ matrix with block representation $(\Sigma(X(i,j)))$. We define the sequence of matrices $(S(n))$ by $S(1) = X_1$ and $S(n) = \Sigma^{n-1}(X_1)$. The number s is called scaling factor of the system of substitutions.

Definition 4. We call the system of substitutions $(\mathcal{X}, \mathcal{Y}, \Sigma, X_1)$ expansive if the block representation of the matrix $\Sigma(X_1) = (X(i,j))$ using matrices in $\mathcal{X}$ fulfills the condition $X(0,0) = X_1$. To be more clear: $X(0,0)$ is the $x \times x$ left upper block of $\Sigma(X_1)$.

Lemma 1. *Let $(\mathcal{X}, \mathcal{Y}, \Sigma, X_1)$ be an expansive system of substitutions. Then for all $n > 0$ the matrix $S(n)$ is $xs^{n-1} \times xs^{n-1}$ left upper minor of the matrix $S(n+1)$.*

Definition 5. We say that a $u \times v$ matrix $K = (k(\alpha, \beta))$ occurs in the $w \times z$ matrix $T = (t(a,b))$ if for some $0 \leq i < w$ and $0 \leq j < z$ one has $i + u \leq w$, $j + v \leq z$ and for all $0 \leq \alpha < u$ and $0 \leq \beta < v$ one has $t(i + \alpha, j + \beta) = k(\alpha, \beta)$. We say that K occurs in x-position in T if moreover $x \,|\, i$ and $x \,|\, j$.

Definition 6. Let x be a fixed natural number and T be a $wx \times zx$ matrix over A. We denote $\mathcal{N}_x(T)$ the set of all $2x \times 2x$ matrices occurring in x-position in T.

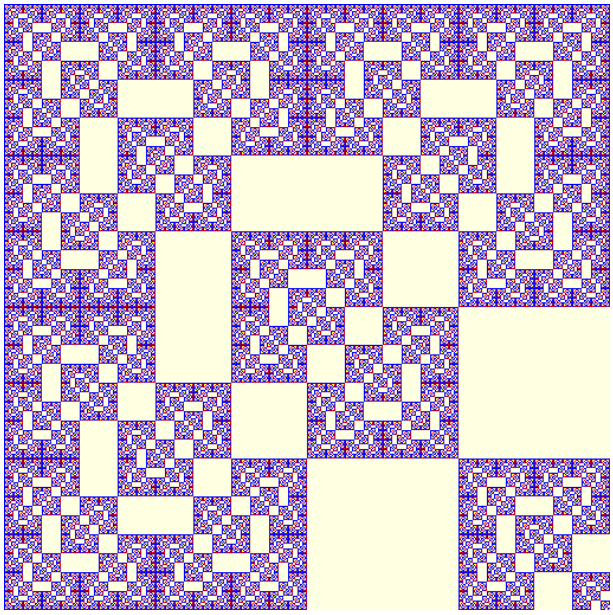
Definition 7. Let x be a fixed natural number and T be a $wx \times zx$ matrix over A. We denote by $\mathcal{J}_x(T)$ the set of all $x \times x$ matrices occurring in T in the positions $\{(0, kx) \,|\, k \in \mathbb{N}\}$. Analogously, we denote by $\mathcal{I}_x(T)$ the set of all $x \times x$ matrices occurring in T in the positions $\{(kx, 0) \,|\, k \in \mathbb{N}\}$.

Theorem 2. *Let $(A, f, 1)$ be a finite structure with ternary function f and let $(\mathcal{X}, \mathcal{Y}, \Sigma, X_1)$ be an expansive system of substitutions of type $x \to y$ over A. We define the matrices $(R(n))_{n \geq 1}$ according to x and $s = y/x$ given by the system of substitutions. Suppose that for some $m > 1$ following conditions hold:*

– $R(m) = S(m)$.
– $\mathcal{N}_x(S(m-1)) = \mathcal{N}_x(S(m))$.
– $\mathcal{J}_x(S(m-1)) = \mathcal{J}_x(S(m))$ *and* $\mathcal{I}_x(S(m-1)) = \mathcal{I}_x(S(m))$.

Then for all $n \geq 1$ *one has* $R(n) = S(n)$.

This result says essentially that if a recurrent double sequence and a double sequence produced by an expansive system of context-free substitutions are identical in a starting minor, then they are identical everywhere.

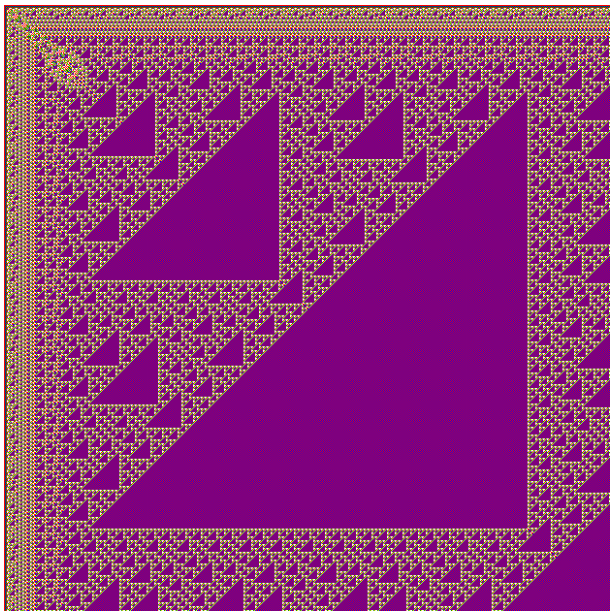


(1) Open Peano Curve, 512×512, $(\mathbb{F}_4, 2x^2 + 2y + 2z^2, 3)$.

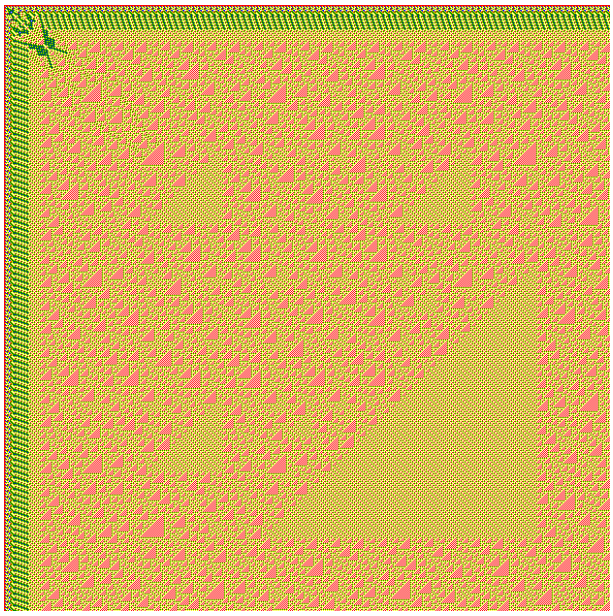
Corollary 1. *The Open Peano Curve given by* $(\mathbb{F}_4, 2x^2 + 2y + 2z^2, 3)$ *in Figure 1 can be generated with an expansive system of substitutions of type* $2 \to 4$ *with* 7 *rules. This is an example of Frobenius polynomial. Observe that we denoted by* 2 *and* 3 *elements* ϵ *and* $\epsilon + 1$ *in* $\mathbb{F}_4$ *such that* $\epsilon^2 + \epsilon + 1 = 0$.

The author found hundreds of situations in which Theorem 2 applies. Most of them are produced by linear combinations of Frobenius polynomials. Recall that for a finite field $\mathbb{F}_q$, with $q = p^k$, the automorphism of Frobenius $x \to x^p$ generates the cyclic Galois group of the algebraic extension $\mathbb{F}_q/\mathbb{F}_p$. The other automorphisms are given by the polynomials $x^p, x^{p^2}, \ldots, x^{p^{k-1}}, x^{p^k} = x = \mathrm{id}$. We call Frobenius polynomial every linear combinations of Frobeni with coefficients in $\mathbb{F}_q$. All Frobenius polynomials are additive homomorphisms of $\mathbb{F}_q$ in itself. I conjecture that all functions $f(x, y, z) = e(x) + g(y) + h(z)$, where $e(x)$, $g(y)$ and $h(z)$ are Frobenius polynomials, produces recurrent double sequences which are substitution patterns. A lot of such Frobenius examples can be found in [8]. The next section presents some examples of recurrent double sequences that can be produced by expansive systems of substitution but are *not* Frobenius.

Conjecture 1. Let G be a finite abelian group, $f : G^3 \to G$ a homomorphism of groups, and $a \in G \setminus \{0\}$ an arbitrary element. Then the recurrent double sequence defined by (G, f, a) can be also constructed using an expansive system of context-free substitutions.



(2) Pascal's Table Runner, 625×625. $(\mathbb{F}_5, x^2y^4z^2 + x^2y^4 + y^4z^2 + 4xz + 1, 1)$.



(3) Pascal's Garden, 625×625. $(\mathbb{F}_5, 4x^3z^3 + 4xy^2 + 4y^2z + x^4yz^4 + 1, 1)$.

3 Triangular perplexity

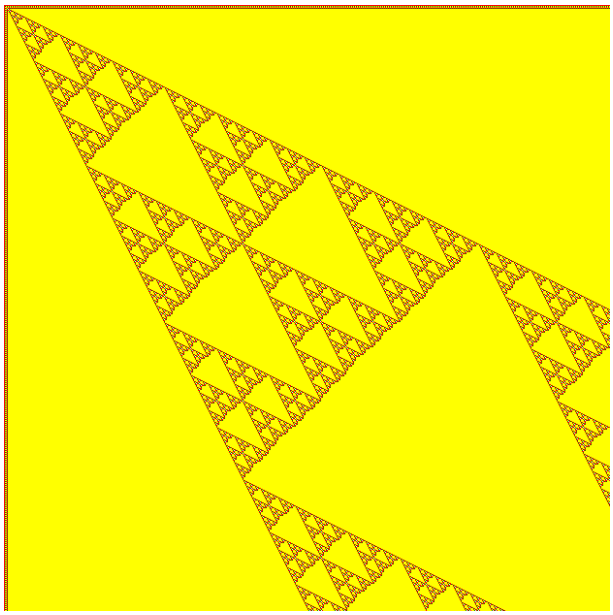
After the definitive acceptance of the paper [8] by the journal Fractals, the author accidentally found also a lot of recurrent double series which can be generated by expansive systems of context-free substitutions although they are not given by sums of Frobenius terms.

Corollary 2. *Pascal's Table Runner given by* $(\mathbb{F}_5, x^2y^4z^2 + x^2y^4 + y^4z^2 + 4xz + 1, 1)$ *in Figure 2 can be generated with an expansive system of substitutions of type* $64 \to 128$ *with* 38 *rules.*

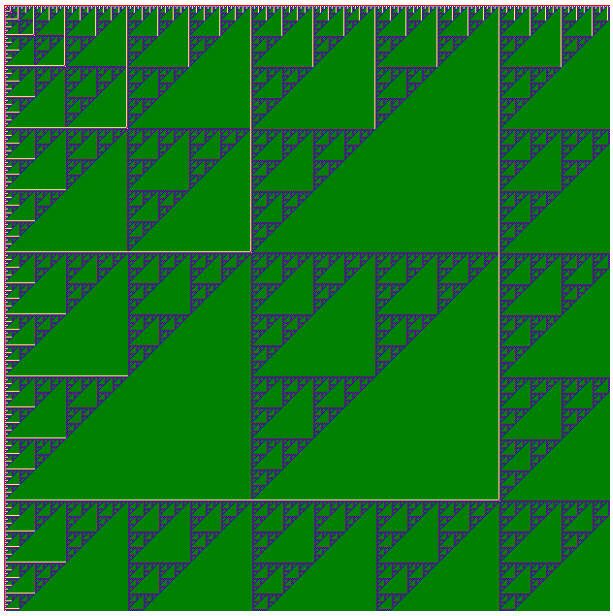
Corollary 3. *Pascal's Garden given by* $(\mathbb{F}_5, 4x^3z^3 + 4xy^2 + 4y^2z + x^4yz^4 + 1, 1)$ *in Figure 3 can be generated with an expansive system of substitutions of type* $64 \to 128$ *with* 109 *rules.*

Definition 8. Let $(A, f, 1)$ be a recurrence producing the recurrent double sequence $a(i, j)$). A minimal representation of this sequence is a recurrence $(\mathbb{F}_q, g, 1)$ and an embedding $\eta : A \to \mathbb{F}_q$ such that q is the

least prime power $\geq |A|$ and for all $i, j \in \mathbb{N}$, $\eta(a(i,j)) = b(i,j)$, where $(b(i,j)$ is the recurrent double sequence produced by $(\mathbb{F}_q, g, 1)$.



(4) Molten Radioactive Mark, 625×625. $(\mathbb{F}_5, 4x^2y^4z^2 + 4x^2y^3 + 4y^3z^2 + x^2yz^2 + 4, 1)$.



(5) Pascal's Straightedge, 625×625. $(\mathbb{F}_5, 2x^2y^4z^2 + 2x^2 + 2z^2 + 2xy^4z + 1, 1)$.

Remark 1. The recurrent double sequences defined and presented in the Figures 2, 3, 4 and 5 do not accept any minimal representation with Frobenius polynomials.

Proof. All four recurrent double sequences contain five different values inside, so they cannot be represented over any field smaller than $\mathbb{F}_5$. On the other hand, the only Frobenius automorphism of $\mathbb{F}_5$ is the identity, so the only Frobenius polynomials in question have the form $ax + by + cz + d$. But none of those polynomials generates any of the patterns (2), (3), (4) or (5), as one can see after a short computer exploration.

However, it remains an open question if those recurrent double sequences and many other examples of this kind accept some Frobenius representation in bigger fields, or representations using finite abelian groups and homomorhism of groups, as in the conjecture above.

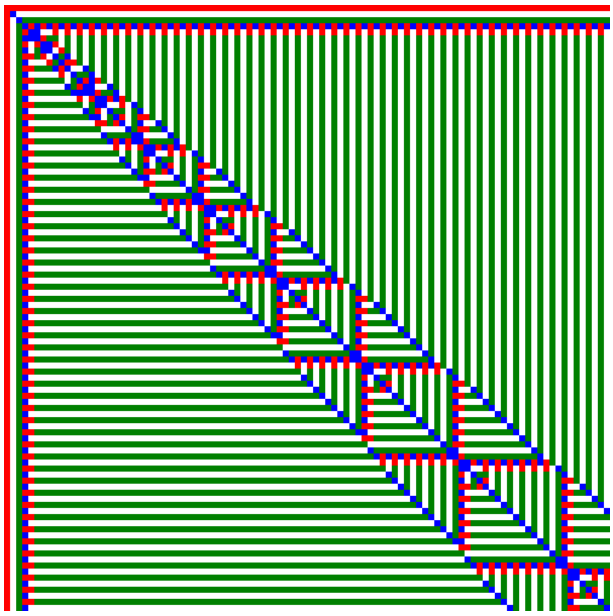
Corollary 4. *Molten Radioactive Mark given by* $(\mathbb{F}_5, 4x^2y^4z^2 + 4x^2y^3 + 4y^3z^2 + x^2yz^2 + 4, 1)$ *in Figure 4 can be generated with an expansive system of substitutions of type* $10 \to 30$ *with* 40 *rules.*

Corollary 5. *Pascal's Straightedge given by* $(\mathbb{F}_5, 2x^2y^4z^2 + 2x^2 + 2z^2 + 2xy^4z + 1, 1)$ *in Figure 5 can be generated with an expansive system of substitutions of type* $4 \to 8$ *with* 29 *rules.*

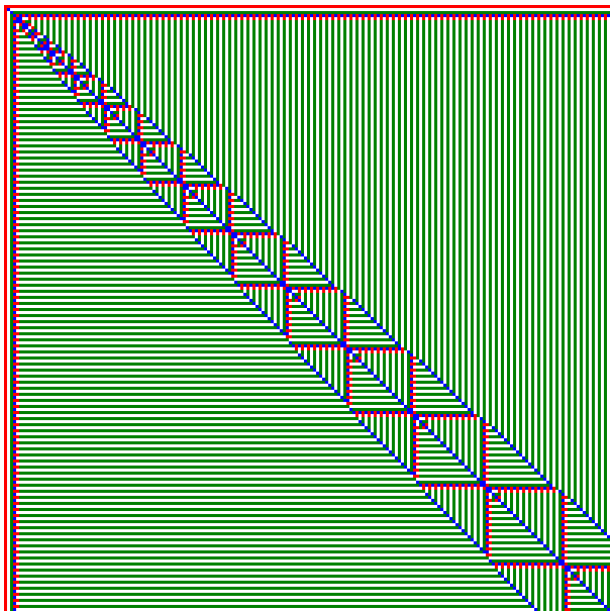
The title of the section is an homage of the celebrated article Pentaplexity by Sir Roger Penrose, see [3]. In this paper the Penrose Tiling was described for the first time, together with its substitution rules. The examples presented here contain a lot of triangles in their patterns. It is not said that all examples are of this kind. All sequences that become periodic can be also produced by expansive systems of substitutions.

4 Stairway to heaven

All these examples lead to the following natural question: Is it true that all recurrent double sequences over finite sets are generated by context-free substitutions? In this section we study the recurrent double sequence Stairway to Heaven SH given by the recurrent rule $(\mathbb{F}_5, 4x^2y^4z^2 + 4x^4y^3 + 4y^3z^4 + 4y^2 + 2, 1)$. This recurrent double sequence contains only four elements of five and accepts minimal representations over the field $\mathbb{F}_4$. However, the recurrent sequence applies only 52 triples $(x, y, z) \to f(x, y, z)$ from the 64 triples in $\mathbb{F}_4^3$, and it is quite difficult to find a nice definition in the set of 4^{12} possible minimal representations. The shortest definition found by the author has 10 polynomial terms.



(6) Stairway to Heaven, 100×100. $(\mathbb{F}_5, 4x^2y^4z^2 + 4x^4y^3 + 4y^3z^4 + 4y^2 + 2, 1)$.



(7) Stairway to Heaven, 200×200. $(\mathbb{F}_5, 4x^2y^4z^2 + 4x^4y^3 + 4y^3z^4 + 4y^2 + 2, 1)$.

We shall prove here that this sequence cannot be generated by expansive systems of context free substitutions of any type. According to Figure (6) we recall the elemets of $\mathbb{F}_5$ occurring in SH: white $= 0$, red $= 1$, green $= 2$, blue $= 3$.

Theorem 3. *SH given by $(\mathbb{F}_5, 4x^2y^4z^2+4x^4y^3+4y^3z^4+4y^2+2, 1)$ cannot be generated using expansive systems of context-free substitutions.*

Lemma 2. *Suppose that a double sequence can be generated by an expansive system of context-free substitutions of some type $x \to y$. Then for all $k \geq 2$ the double sequence can be generated by expansive systems of context-free substitutions of type $kx \to ky$.*

Proof. Let $(\mathcal{X}, \mathcal{Y}, \Sigma, X_1)$ be an expansive system of substitutions of type $x \to y$. We define the new expansive system of substitutions $(\mathcal{X}', \mathcal{Y}', \Sigma', X_1')$ in the following way: The set $\mathcal{X}'$ consists of all $kx \times kx$ matrices consisting of k^2 many $x \times x$ blocks, where every element of $\mathcal{X}$ may occur as a block. Σ' is the natural block-wise extension of Σ. Let $\mathcal{Y}'$ be $\Sigma'(\mathcal{X}')$. Let X_1' be the $kx \times kx$ left upper minor of the double sequence generated by $(\mathcal{X}, \mathcal{Y}, \Sigma, X_1)$. Using Lemma 1 one gets that the new system of context-free substitutions is also expansive.

In the next statements the word *minor* will be used for *connected minor of SH*.

Definition 9. For $i \in \mathbb{N}$: α_i is the 5×5 minor starting with $a(c(i), c(i))$, where $c(i) = i^2 + 7i + 15$, $\alpha := \alpha_0$; β_i is the 2×2 minor $a(c(i+1) - 2, c(i+1) - 2)$, $\beta := \beta_0$; D_i is the $(8 + 2i) \times (8 + 2i)$ minor starting with $a(c(i), c(i))$.

Lemma 3. *For $i \geq 1$ all elements $a(i, i)$ are blue. All minors α_i are translated copies of α. All minors β_i are translated copies of β. The squares D_i cover the first diagonal for $i \geq 15$. D_i has α_i as a left upper minor and β_i as a right lower minor. Between α_i and β_i one finds $i+1$ white stripes and i green stripes. SH interprets the set $\{n \in \mathbb{N} \mid n \geq 2\}$ in the folowing way: For all $i \geq 0$, the square D_i has exactly $i+2$ many red unit-squares on every edge.*

Proof. By induction. The crucial configuration to look at is the configuration around $a(c(i) - 1, c(i) - 2i - 10)$. This configuration adds 2 to the edge of D_{i-1} to get the edge of D_i.

Lemma 4. *Consider $u \in \mathbb{N}$ even, $u \geq 8$. Let $N \in \mathbb{N}$ even, $N \geq 16$, such that a $u \times u$ minor starting with $a(N, N)$ does not contain any α_i and does not intersect any β_j. Suppose that SH is decomposed in $u \times u$ minors $U(k, h)$, where $U(k, h)$ starts with $a(ku, hu)$. Let M_u be the set of $u \times u$ matrices occurring in SH as minors $U(n, n)$ with $nu \geq N$. Then:*

- *M_u contains $u/2+3$ elements.*
- *Every element of M_u occurs infinitely often on the main diagonal of SH, as $U(n,n)$ with $nu \geq N$.*

Proof. $U(n,n)$ may contain at most one starting point for a β_i. Counting the possible starting points, we get $u/2$ cases. Further one has two cases where $U(n,n)$ intersects an α_i and the case where $U(n,n)$ does not intersect any α_i and any β_j.

Proof (Theorem 3). Suppose that SH can be produced by some expansive system of context-free substitutions of type $x \to mx$. By Lemma 2 we can suppose that x is even and $x \geq 8$. Choose N good for $u = mx$ in the sense of Lemma 4 and observe that this value N is also good for $u = x$ in the same sense. Every element of M_{mx} occurs in SH infinitely often as minor starting in some $a(kmx, hmx)$, so all these minors must be right hand side in a substitution rule with left hand side in M_x. So $(x/2)+3 \geq m(x/2)+3$ which is possible only for $m = 1$. Contradiction.

References

1. **Noam Chomsky**: (1956). *Three models for the description of language.* IRE Transactions on Information Theory, 2, 113 - 124, 1956.
2. **Jarkko Kari**: *Theory of cellular automata: A survey.* Theoretical Computer Science 334, 3 - 33, 205.
3. **Roger Penrose**: *Pentaplexity.* Mathematical Intelligencer 2 (1), 32 - 37, 1979.
4. **Benoit B. Mandelbrot**: *The fractal geometry of nature.* W. H. Freeman and Company, San Francisco, 1977, 1982.
5. **Dann E. Passoja, Akhlesh Lakhtakia**: *Carpets and rugs: an exercise in numbers.* Leonardo, 25, 1, 1992, 69 - 71.
6. **Mihai Prunescu**: *Undecidable properties of recurrent double sequences.* Notre Dame Journal of Formal Logic, 49, 2, 143 - 151, 2008.
7. **Mihai Prunescu**: *Self-similar carpets over finite fields.* European Journal of Combinatorics, 30, 4, 866 - 878, 2009.
8. **Mihai Prunescu**: *Recurrent double sequences that can be produced by context-free substitutions.* accepted for publication in by Fractals, 2009.
9. **Stephen J. Willson**: *Cellular automata can generate fractals.* Discrete Applied Mathematics, 8, 1984, 91 - 99.

Herbrand Consistency of $\mathbf{I\Delta_0}$ and $\mathbf{I\Delta_0 + \Omega_1}$

Saeed Salehi

Department of Mathematics
University of Tabriz
29 Bahman Boulevard
51666-17766 Tabriz – Iran
root@saeedsalehi.ir
http://saeedsalehi.ir/

1 Notation and Basic Definitions

Robinson's Arithmetic Q is a finitely axiomatized first-order theory that states some very basic facts of arithmetic, like the injectivity of the successor function or the inductive definitions of addition and multiplication. Peano's arithmetic PA is the first-order theory that extends Q by the induction schema for any arithmetical formula $\varphi(x)$: $\varphi(0) \,\&\, \forall x[\varphi(x) \to \varphi(x+1)] \to \forall x \varphi(x)$. Fragments of PA are extensions of Q with the induction schema restricted to a class of formulas. A formula is called bounded if its every quantifier is bounded, i.e., is either of the form $\forall x \leq t(\ldots)$ or $\exists\, x \leq t(\ldots)$ where t is a term; they are read as $\forall x(x \leq t \to \ldots)$ and $\exists x(x \leq t \wedge \ldots)$ respectively. It is easy to see that bounded formulas are decidable. The theory $\mathrm{I\Delta_0}$, also called bounded arithmetic, is axiomatized by Q plus the induction schema for bounded formulas. The exponentiation function exp is defined by $\exp(x) = 2^x$; the formula Exp expresses its totality: $(\forall x \exists y[y = \exp(x)])$. The inverse of exp is log; and the cut $\mathfrak{log}$ consists of the logarithms of all elements: $\mathfrak{log} = \{x \mid \exists y[\exp(x) = y]\}$. The superscripts above the function symbols indicate the iteration of the functions: $\exp^2(x) = \exp(\exp(x))$, $\log^2 x = \log\log x$; similarly the cut $\mathfrak{log}^n$ is $\{x \mid \exists y[\exp^n(x) = y]\}$. Let us recall that Exp is not provable in $\mathrm{I\Delta_0}$; and sub-theories of $\mathrm{I\Delta_0} + \mathrm{Exp}$ are called weak arithmetics. Between $\mathrm{I\Delta_0}$ and $\mathrm{I\Delta_0} + \mathrm{Exp}$ a hierarchy of theories is considered in the literature, which has close connections with computational complexity. Let $\omega_0(x) = x^2$ and $\omega_{\mathrm{n}+1} = \exp(\omega_{\mathrm{n}}(\log \mathrm{x}))$ be defined inductively, and let Ω_{m} express the totality of ω_{m} (i.e., $\Omega_{\mathrm{m}} \equiv \forall x \exists y[y = \omega_{\mathrm{m}}(x)]$). We have $\mathrm{I\Delta_0} + \Omega_{\mathrm{n}} \subsetneqq \mathrm{I\Delta_0} + \Omega_{\mathrm{n}+1}$ for every $n \geq 0$.

Skolemized form of a (preferably prenex normal) formula is obtained by removing the existential quantifiers and replacing their corresponding variables with new (Skolem) function symbols on the universal variables that precedes the quantifier. The resulted universal formula is equi-consistent with the original formula, in the sense that the formula is consistent if and only if the set of instances of its Skolemized form (its Skolem instances) is consistent. This can be generalized to theories (i.e., sets of sentences) as well. Herbrand Consistency of a theory is defined to be the (propositional) consistency of the set of its all Skolem instances. This is a weaker notion than the standard (Hilbert style) consistency, resembling much to cut-free consistency. Thus, for a theory T, Herbrand Consistency, $\mathrm{HCon}(T)$, of T is equivalent to Hilbert Consistency, $\mathrm{Con}(T)$, of T in sufficiently strong theories such as Peano's Arithmetic PA (or even $\mathrm{I\Delta_0} + \mathrm{SuperExp}$). But in weak arithmetics, like $\mathrm{I\Delta_0}$ or $\mathrm{I\Delta_0} + \Omega_1$ these are quite different consistency predicates; which makes it difficult to deal with $\mathrm{HCon}(-)$ in them.

2 Background History

Two interesting theorems were proved by Z. Adamowicz in [1] about Herbrand Consistency of the theories $\mathrm{I\Delta_0} + \Omega_{\mathrm{m}}$ for $m \geq 2$:

Theorem 1. *For a bounded formula $\theta(\overline{x})$ and $m \geq 2$, if the theory*

$$\mathrm{I\Delta_0} + \Omega_{\mathrm{m}} \;+\; \exists \overline{x} \in \mathfrak{log}^{m+1} \theta(\overline{x}) \;+\; \mathrm{HCon}_{\mathfrak{log}^{m-2}}(\mathrm{I\Delta_0} + \Omega_{\mathrm{m}})$$

is consistent, then so is the theory

$$\mathrm{I\Delta_0} + \Omega_{\mathrm{m}} \;+\; \exists \overline{x} \in \mathfrak{log}^{m+2} \theta(\overline{x}),$$

where $\mathrm{HCon}_{\mathfrak{log}^{m-2}}(-)$ is the relativization of $\mathrm{HCon}(-)$ to the cut $\mathfrak{log}^{m-2}$.

Theorem 2. *For natural $m, n \geq 0$ there exists a bounded formula $\eta(\overline{x})$ such that*

$$\mathrm{I}\Delta_0 + \Omega_{\mathrm{m}} \ + \ \exists \overline{x} \in \mathfrak{log}^n \eta(\overline{x})$$

is consistent, but the theory

$$\mathrm{I}\Delta_0 + \Omega_{\mathrm{m}} \ + \ \exists \overline{x} \in \mathfrak{log}^{n+1} \eta(\overline{x})$$

is not consistent.

These two theorems (by putting $n = m + 1$ for $m \geq 2$) imply together that

$$(*) \qquad \mathrm{I}\Delta_0 + \Omega_{\mathrm{m}} \nvdash \mathrm{HCon}_{\mathfrak{log}^{m-2}}(\mathrm{I}\Delta_0 + \Omega_{\mathrm{m}}) \ \text{ for } m \geq 2.$$

This gives a partial answer to the question of holding Gödel's Second Incompleteness Theorem for cut-free consistency and weak arithmetics, which was answered in full by D.E. Willard in [4].

The proof of Theorem 1 was adapted to the case of $\mathrm{I}\Delta_0 + \Omega_1$ in Chapter 5 of [3] by modifying the definition of $\mathrm{HCon}(-)$. Thus, one gets another model-theoretic proof of the unprovability of Herbrand Consistency of $\mathrm{I}\Delta_0 + \Omega_1$ in itself. Note that Theorem 2 holds for $\mathrm{I}\Delta_0 + \Omega_1$ and $\mathrm{I}\Delta_0$ as well. Later, L.A. Kołodziejczyk extended $(*)$ above to the following results in [2]:

Main Theorem (of [2]): *There exists n such that the theory $\bigcup_m S_m$ (or, equivalently $\mathrm{I}\Delta_0 + \bigwedge_m \Omega_{\mathrm{m}}$) does not prove the Herbrand Consistency of S_3^n (where S_3^n is defined like S_2^n but with the language expanded by a function symbol for $\#_3$).*

Also the following weaker result is proved in [2]:

Theorem 4.1 (of [2]): *For every $m \geq 3$ there exists an n such that $S_m \nvdash \mathrm{HCon}(S_m^n)$.*

These results are taken to show that the Herbrand notion of Consistency cannot serve for Π_1-seperating the theories $\{\mathrm{I}\Delta_0 + \Omega_{\mathrm{k}}\}_k$ or $\{S_k\}_k$.

3 New Results

In the paper version of this extended abstract, we would like to polish the results of [3] (Ch. 5) and make them readable to a wider audience. The newest result is that Theorem 1 can be extended (almost) to $\mathrm{I}\Delta_0$.

Let $\mathscr{I}$ be the cut $\{x \mid \exists y[y = \exp(\omega_1{}^2(x))]\}$ and put $\mathscr{T} = \log \mathscr{I} = \{x \mid \exists y[y = \exp(x) \wedge y \in \mathscr{I}]\}$. Another way of defining $\mathscr{T}$ is $\{x \mid \exists y[y = \exp^2(x^4)]\}$. Then we can modify the above theorems of Z. Adamowicz to the following:

Theorem 3. *For a bounded formula $\theta(\overline{x})$, if the theory*

$$\mathrm{I}\Delta_0 + \mathrm{HCon}(\mathbb{I}\triangle_0) \ + \ \exists \overline{x} \in \mathscr{I} \theta(\overline{x})$$

is consistent, then so is the theory

$$\mathrm{I}\Delta_0 \ + \ \exists \overline{x} \in \mathscr{T} \theta(\overline{x}),$$

where $\mathbb{I}\triangle_0$ is the theory $\mathrm{I}\Delta_0$ augmented with the additional axiom $\forall x \exists y (y = x \cdot x)$.

Theorem 4. *There exists a bounded formula $\eta(\overline{x})$ such that*

$$\mathrm{I}\Delta_0 \ + \ \exists \overline{x} \in \mathscr{I} \eta(\overline{x})$$

is consistent, but the theory

$$\mathrm{I}\Delta_0 \ + \ \exists \overline{x} \in \mathscr{T} \eta(\overline{x})$$

is not consistent.

These two theorems give rise to a model-theoretic proof of $\mathrm{I}\Delta_0 \nvdash \mathrm{HCon}(\mathbb{I}\triangle_0)$.

We note that the proof of Theorem 1 in [1] goes (roughly) as follows: for a model

$$\mathcal{M} \models \mathrm{I}\Delta_0 + \Omega_{\mathrm{m}} \ + \ \exists \overline{x} \in \mathfrak{log}^{m+1} \theta(\overline{x}) \ + \ \mathrm{HCon}_{\mathfrak{log}^{m-2}}(\mathrm{I}\Delta_0 + \Omega_{\mathrm{m}})$$

we construct an inner model $\mathcal{N} \models \mathrm{I}\Delta_0 + \Omega_{\mathrm{m}}$ (by $\mathcal{M} \models \mathrm{HCon}_{\mathfrak{log}^{m-2}}(\mathrm{I}\Delta_0 + \Omega_{\mathrm{m}})$) which is definable in $\mathcal{M}$, and then *squeeze* a witness for $\overline{x} \in \mathfrak{log}^{m+1} \& \theta(\overline{x})$ logarithmically to get a witness for $\overline{x} \in \mathfrak{log}^{m+2} \& \theta(\overline{x})$ in $\mathcal{N}$, hence we obtain the desired model

$$\mathcal{N} \models \mathrm{I}\Delta_0 + \Omega_{\mathrm{m}} \ + \ \exists \overline{x} \in \mathfrak{log}^{m+2} \theta(\overline{x}).$$

In the squeezing process, we need to demonstrate a large (non-standard) number with a relatively small code (Gödel) number – as small as the logarithm of the number. Thus we added the redundant (and already Q−provable) sentence $\forall x \exists y (y = x \cdot x)$ to the set of axioms of $\mathrm{I}\Delta_0$ to get a Skolem function symbol like $\mathfrak{f}$ with the interpretation $\mathfrak{f}(x) = x^2$. Now we have $\mathfrak{f}^m(2) = 2^{2^m}$ and we can code this number by $\mathcal{O}(2^m)$. This is the reason we could formulate Theorem 3 for $\mathbb{I}\triangle_0$ instead of $\mathrm{I}\Delta_0$, and then get its corollary as $\mathrm{I}\Delta_0 \nvdash \mathrm{HCon}(\mathbb{I}\triangle_0)$, though it would have been more desirable to give this kind of proof for the unprovability $\mathrm{I}\Delta_0 \nvdash \mathrm{HCon}(\mathrm{I}\Delta_0)$.

Also, L.A. Kołodziejczyk notes in [2] that his main theorem could be extended to the case of S_2^n by either having a function symbol for ω_1 in the language or adding the seemingly irrelevant axiom $\forall x \exists y (y = x \# x)$. He then mentions that $\bigcup_m S_m \nvdash \mathrm{HCon}^*(S_2^m + [\forall x \exists y (y = x \# x)])$.

This goes to say that by having an additional formula in the list of axioms, though it may seem too trivial to be an axiom, one can shorten the cut-free proofs exponentially (cf. Clarification on p. 474 in [4] and Remark on p. 636 in [2]).

To see why the proof of Theorem 2 in [1] works for Theorem 4 as well, we note that the proof does not explicitly construct a bounded formula $\eta(\overline{x})$ which satisfies the conditions of the theorem. The proof is by contradiction: assume for all bounded formulas $\theta(\overline{x})$, the consistency of $\mathrm{I}\Delta_0 + \Omega_{\mathrm{n}} + \exists \overline{x} \in \mathfrak{log}^m \theta(\overline{x})$ implies the consistency of $\mathrm{I}\Delta_0 + \Omega_{\mathrm{n}} + \exists \overline{x} \in \mathfrak{log}^{m+1} \theta(\overline{x})$, and then we get a contradiction. The essential relevance of the cuts $\mathfrak{log}^m$ and $\mathfrak{log}^{m+1}$ in the proof is the equivalence $2^{\mathtt{x}} \in \mathfrak{log}^m \iff \mathtt{x} \in \mathfrak{log}^{m+1}$. Indeed, any two cuts $\mathscr{I}, \mathscr{T}$ which satisfy this equivalence ($2^{\mathtt{x}} \in \mathscr{I} \iff \mathtt{x} \in \mathscr{T}$) will work in the proof of Theorem 2 in [1]. And our cuts $\mathscr{I}, \mathscr{T} = \log \mathscr{I}$ defined above, just have this relation with each other.

Finally, we believe that with the modified definition and formalization of Herbrand Consistency in [3], the above mentioned results of [2] can be improved considerably – for the case of S_2^n and a little beyond. This is yet to be seen.

References

1. Adamowiz, Z.: Herbrand Consistency and Bounded Arithmetic. Fund. Math. 171, 279–292 (2002) `http://journals.impan.pl/fm/Inf/171-3-7.html`
2. Kołodziejczyk, L.A.: On the Herbrand Notion of Consistency for Finitely Axiomatizable FGragments of Bounded Arithmetic Theories. J. Symb. Log. 71, 624–638 (2006) `http://projecteuclid.org/euclid.jsl/1146620163`
3. Salehi, S.: Herbrand Consistency in Arithmetics with Bounded Induction. Ph.D. Dissertation in Institute of Mathematics, Polish Academy of Sciences (2002) `http://saeedsalehi.ir/pphd.html`
4. Willard, D. E.: How to Extend the Semantic Tableaux and Cut-Free Versions of the Second Incompleteness Theorem Almost to Robinson's Arithmetic Q. J. Symb. Log. 67, 465–496 (2002) `http://projecteuclid.org/euclid.jsl/1190150055`

Unbounded arithmetic

Sam Sanders[1] and Andreas Weiermann[1]

[1]University of Ghent, Department of Pure Mathematics and
Computer Algebra, Krijgslaan 281, B-9000 Gent (Belgium),
{sasander,weierman}@cage.ugent.be

1 Introduction

When comparing the different axiomatizations of bounded arithmetic and Peano arithmetic, it becomes clear that there are similarities between the fragments of these theories. In particular, it is tempting to draw an analogy between the hierarchies of bounded arithmetic and Peano arithmetic. However, one cannot deny that there are essential and deeply rooted differences and the most one can claim is a weak analogy between these hierarchies. The following quote by Kaye (see [2]) expresses this argument in an elegant way.

> Many authors have emphasized the analogies between the fragments Σ_n^b-IND of $I\Delta_0+(\forall x)(x^{\log x}$ exists) and the fragments $I\Sigma_n$ of Peano arithmetic. Sometimes this is helpful, but often one feels that the bounded hierarchy of theories is of a rather different nature and new techniques must be developed to answer the key questions concerning them.

In this paper, we propose a (conjectured) hierarchy for Peano arithmetic which is much closer to that of bounded arithmetic than the existing one. In light of this close relation, techniques developed to establish properties of the new hierarchy should carry over naturally to the (conjectured) hierarchy of bounded arithmetic. As the famous P vs. NP problem is related to the collapse of the hierarchy of bounded arithmetic, the new hierarchy may prove particularly useful in solving this famous problem.

2 Preliminaries

We assume that the reader is familiar with the fundamental notions concerning bounded arithmetic and Peano arithmetic. For details, we refer to the first two chapters of [1]. For completeness, we mention the Hardy hierarchy and some of its essential properties. Let α be an ordinal and let λ be a limit ordinal and λ_n its n-th predecessor.

$$\begin{aligned} H_0(x) &:= x, \\ H_{\alpha+1}(x) &:= H_\alpha(x+1), \\ H_\lambda(x) &:= H_{\lambda_x}(x). \end{aligned}$$

The well-known Ackermann function $A(x)$ corresponds to $H_{\omega^\omega}(x)$. For a given function $H_\alpha(x)$, the inverse $H_\alpha^{-1}(x)$ is defined as $(\mu m \leq x)(H_\alpha(m) \geq x)$. In general, the function $H_\alpha^{-1}(x)$ is of much lower complexity than $H_\alpha(x)$. Indeed, it is well-known that $A(x)$ is not primitive recursive and that $A^{-1}(x)$ is. For brevity, we sometimes write $|x|_\alpha$ instead of $H_\alpha^{-1}(x)$.

3 Two fundamental differences

In this section, we point out two fundamental differences between bounded arithmetic and Peano arithmetic. In section 4, we attempt to overcome these differences.

3.1 The logarithmic function

In bounded arithmetic, the log function is defined as $|x| := \lceil \log_2(x+1) \rceil$. Because the inverse of log, i.e. the exponential function, is not total in bounded arithmetic, the log does not have its 'usual' properties. The following theorem illustrates this claim.

Theorem 1 *The theory of bounded arithmetic does not prove that the* log *function is unbounded, i.e.* $S_2 \nvdash (\forall x)(\exists y)(|y| > x)$.

Proof. Assume S_2 proves $(\forall x)(\exists y)(|y| > x)$. By Parikh's theorem, there is a term t such that S_2 proves $(\forall x)(\exists y \leq t(x))(|y| > x)$. As $|x|$ is weakly increasing, there follows $(\forall x)(|t(x)| > x)$. However, this implies that $t(x)$ grows as fast as the exponential function, which is impossible.

By completeness, there is a model of S_2 in which $|x|$ is bounded. At the very least, this theorem shows that one should be careful with 'visual' proofs. Indeed, even most mathematicians would claim that it is clear from the graph of $\log x$ that this function is unbounded. However, by itself, the previous theorem is not a big revelation. Indeed, the same theorem (and proof) holds for PRA and $A^{-1}(x)$ instead of S_2 and $|x|$. It is easy to verify that the function $H_{\varepsilon_0}^{-1}(x)$ has the same property for Peano arithmetic.

So far, we showed that the log function has unusual properties in bounded arithmetic, but there seem to be similarly 'strange' functions in Peano arithmetic. However, the axioms of Peano arithmetic do not involve $H_{\varepsilon_0}^{-1}(x)$, whereas the log function is used explicitly in the axiomatization of bounded arithmetic. Indeed, consider the following axiom schema.

Axiom schema 2 (Φ-LIND) *For every $\varphi \in \Phi$, we have*

$$\big[\varphi(0) \wedge (\forall n)(\varphi(n) \to \varphi(n+1))\big] \to (\forall n)\varphi(|n|).$$

This axiom schema is called 'length induction'. The theory S_2^i of bounded arithmetic consists of the basic theory BASIC plus the Σ_n^b-LIND schema. Furthermore, the theory T_2^i consists of BASIC plus the Σ_n^b-induction schema and the (conjectured) hierarchy of bounded arithmetic is as follows, for $i \geq 2$,

$$S_2^1 \subseteq T_2^1 \subseteq \cdots \subseteq S_2^i \subseteq T_2^i \subseteq S_2^{i+1} \subseteq T_2^{i+1} \subseteq \cdots \subseteq S_2 = T_2. \tag{1}$$

Thus, the log function appears in a non-trivial way in the axiomatization of bounded arithmetic, although it has unusual properties (see theorem 1). By contrast, the function $H_{\varepsilon_0}^{-1}(x)$ does not appear in the axioms of Peano arithmetic.

Finally, it is worth mentioning that in the presence of the exponential function, which is available in $I\Sigma_1$, Σ_n-IND and Σ_n-LIND coincide. Thus, at first glance there is no analogue of the length induction axioms for Peano arithmetic. In section 4, we shall fill this gap.

3.2 The 'smash' function

In bounded arithmetic, Nelson's 'smash' function $x\#y := 2^{|x|.|y|}$ plays an important role. The presence of this function guarantees that Gödel numbering can be done elegantly, that sharply bounded quantifiers can be pushed into bounded quantifiers and that there is a natural correspondence between the polynomial time hierarchy and the hierarchy of bounded arithmetical formulas (see [1, p. 100] for details).

However, the smash function is not Σ_1-definable in $I\Delta_0$. Thus, it is added to $I\Delta_0$, either by the axiom Ω_1 which defines the function $\omega_1(x,y) = x^{|y|}$, or through the axioms BASIC which guarantee that $x\#y = 2^{|x|.|y|}$. The natural counterparts for this function in PRA and Peano arithmetic are

$$x\%y := A\big[A^{-1}(x).A^{-1}(y)\big] \text{ and } x@y := H_{\varepsilon_0}\big[H_{\varepsilon_0}^{-1}(x).H_{\varepsilon_0}^{-1}(y)\big].$$

It is easily verified that $x\%y$ is not primitive recursive and that $x@y$ is not provably total in Peano arithmetic. It should be noted that the latter function has recently been considered in [3] in the context of 'Ackermannian degrees'.

4 A new hierarchy for Peano arithmetic

In this section we introduce a new (conjectured) hierarchy of Peano arithmetic, inspired by the (conjectured) hierarchy of bounded arithmetic. Thus, we refer to these theories as 'unbounded arithmetic'. The following axiom schema plays a central role.

Axiom schema 3 (Φ-$\mathbb{LIND}$) *For every $\varphi \in \Phi$, we have*

$$\big[\varphi(0) \wedge (\forall n)(\varphi(n) \to \varphi(n+1))\big] \to (\forall n)\varphi(|n|_{\varepsilon_0}).$$

Thus, we have introduced the function $H_{\varepsilon_0}^{-1}(x)$ explicitly and the Σ_n-$\mathbb{LIND}$ axioms are the natural counterpart for the Σ_n^b-LIND axioms of bounded arithmetic.

However, the theory $Q{+}\Sigma_i$-$\mathbb{LIND}$ is not a good counterpart for S_2^i. Indeed, recall that S_2^i consists of the axiom schema Σ_i^b-LIND plus the axiom set BASIC. The latter makes sure that $x\#y = 2^{|x|\cdot|y|}$ is available. Thus, the natural counterpart of the smash function in Peano arithmetic, namely $x@y = H_{\varepsilon_0}(|x|_{\varepsilon_0}.|y|_{\varepsilon_0})$, is missing from $Q + \Sigma_n$-$\mathbb{LIND}$. Thus, we define $\mathbb{BASIC}$ as Robinson's theory Q plus the statement that $x@y = H_{\varepsilon_0}(|x|_{\varepsilon_0}.|y|_{\varepsilon_0})$ is total. Next, we define $\mathbb{S}_2^i$ as the theory $\mathbb{BASIC}$ plus Σ_i-$\mathbb{LIND}$ and $\mathbb{T}_2^i$ as the theory $\mathbb{BASIC}$ plus Σ_i-IND. Finally, we define $\mathbb{T}_2$ (respectively $\mathbb{S}_2$) as the union of all theories $\mathbb{T}_2^i$ (respectively $\mathbb{S}_2^i$). It is immediate that $\mathbb{T}_2$ is very close to Peano Arithmetic. We have partial proofs for the following theorem.

Theorem 4 *For $i \geq 2$, we have*

$$\mathbb{S}_2^1 \subseteq \mathbb{T}_2^1 \subseteq \cdots \subseteq \mathbb{S}_2^i \subseteq \mathbb{T}_2^i \subseteq \mathbb{S}_2^{i+1} \subseteq \mathbb{T}_2^{i+1} \subseteq \cdots \subseteq \mathbb{S}_2 = \mathbb{T}_2 = \mathrm{PA} + \mathbb{BASIC}.$$

The ubiquity of fast growing functions in PA allows us to give an alternative hierarchy. The following axiom schema is fundamental.

Axiom schema 5 (Φ-$\mathcal{LIND}$) *For every $\varphi \in \Phi$, we have*

$$\big[\varphi(0) \wedge (\forall n)(\varphi(n) \to \varphi(n+1))\big] \to (\forall n)\varphi\big(|n|_{(\varepsilon_0)_n}\big).$$

As in the previous, we define $\mathcal{BASIC}_n$ as Robinson's theory Q plus the statement that $H_{(\varepsilon_0)_n}(|x|_{(\varepsilon_0)_n}.|y|_{(\varepsilon_0)_n})$ is total. Next, we define $\mathcal{S}_2^i$ as the theory $\mathcal{BASIC}_i$ plus Σ_i-$\mathcal{LIND}$ and $\mathcal{T}_2^i$ as the theory $\mathcal{BASIC}_i$ plus Σ_i-$\mathcal{LIND}$. Finally, we define $\mathcal{T}_2$ (respectively $\mathcal{S}_2$) as the union of all theories $\mathcal{T}_2^i$ (respectively $\mathcal{S}_2^i$). It is immediate that $\mathcal{T}_2$ is essentially Peano Arithmetic. We have partial proofs for the following theorem.

Theorem 6 *For $i \geq 2$, we have*

$$\mathcal{S}_2^1 \subseteq \mathcal{T}_2^1 \subseteq \cdots \subseteq \mathcal{S}_2^i \subseteq \mathcal{T}_2^i \subseteq \mathcal{S}_2^{i+1} \subseteq \mathcal{T}_2^{i+1} \subseteq \cdots \subseteq \mathcal{S}_2 = \mathcal{T}_2 = \mathrm{PA}.$$

Incidentally, if we replace the ordinal ε_0 in schema 3 with an ordinal parameter α, then $\alpha = \varepsilon_0$ corresponds to $\mathbb{LIND}$, $\alpha = (\varepsilon_0)_n$ to $\mathcal{LIND}$ and $\alpha = \omega^2$ essentially to LIND. Thus, all the above length induction schemas are 'branches of the same tree'.

5 Some time functions

The attentive reader has noted that the length induction axioms of bounded arithmetic is not the only place where the log-function is used explicitly. Indeed, the latter function is also used explicitly in the definition of the polynomial time functions. In this section, we introduce two additional function classes which play the role of the polynomial time functions in our two new hierarchies of Peano Arithmetic.

The class FP of the polynomial time functions is obtained by closing a certain set of initial functions under projection, composition and a restricted version of primitive recursion, called 'limited iteration'. Essentially, primitive recursion is allowed as long as the resulting function $f(z, \boldsymbol{x})$ 'does not grow too fast'. In particular, f has to satisfy the following growth condition:

$$|f(z, \boldsymbol{x})| \leq p(|z|, |\boldsymbol{x}|), \text{ for all } z, \boldsymbol{x},$$

where p is some polynomial.

Analogously, the class $\mathbb{FP}$ is defined by closing the same initial functions under projection, composition and a restricted version of double recursion. In particular, double recursion is allowed if the resulting function $f(z, \boldsymbol{x})$ satisfies

$$|f(z, \boldsymbol{x})|_{\varepsilon_0} \leq h\big(|z|_{\varepsilon_0}, |\boldsymbol{x}|_{\varepsilon_0}\big),$$

where h is some primitive recursive function.

Analogously, the class $\mathcal{FP}$ is defined by closing the same initial functions under projection, composition and a restricted version of double recursion. In particular, double recursion is allowed if the resulting function $f(z, \boldsymbol{x})$ satisfies

$$A^{-1}\big[f(z, \boldsymbol{x})\big] \leq h\big(A^{-1}(z), A^{-1}(\boldsymbol{x})\big),$$

where h is some primitive recursive function.

The functions in $\mathcal{FP}$ and $\mathbb{FP}$ may be called 'primitive recursive time' functions. The class $\mathbb{FP}$ is closely related to the total functions of $\mathbb{S}^1_2$ and the class $\mathcal{FP}$ is closely related to the total functions of $\mathcal{S}^1_2$.

Bibliography

[1] Samuel R. Buss, *An introduction to proof theory*, Handbook of proof theory, Stud. Logic Found. Math., vol. 137, North-Holland, Amsterdam, 1998, pp. 1–78.

[2] Richard Kaye, *Using Herbrand-type theorems to separate strong fragments of arithmetic*, Arithmetic, proof theory, and computational complexity (Prague, 1991), Oxford Logic Guides, vol. 23, Oxford Univ. Press, New York, 1993, pp. 238–246. MR1236465 (94f:03067)

[3] H. Simmons, *The Ackermann functions are not optimal, but by how much?*, to appear in Journal of Symbolic Logic (2010).

Fine Hierarchies via Priestley Duality

Victor Selivanov *

A.P. Ershov Institute of Informatics Systems
Siberian Division Russian Academy of Sciences
vseliv@iis.nsk.su

Abstract. In applications of the fine hierarchies their characterizations in terms of the so called alternating trees is of principal importance. Also, in many cases a suitable version of many-one reducibility exists that fits a given fine hierarchy. With a use of Priestley duality we obtain a surprising result that suitable versions of alternating trees and of m-reducibilities may be found for any given fine hierarchy, i.e. the methods of alternating trees and m-reducibilities are quite general, which is of some methodological interest.
Along with hierarchies of sets, we consider also more general hierarchies of k-partitions and in this context propose some new notions and establish new results, in particular extend the results mentioned above for hierarchies of sets.
Key words. Hierarchy, m-reducibility, Boolean algebra, bounded distributive lattice, Priestley space, alternating tree, k-partition.

1 Introduction

In several applications of fine hierarchies (see [Se08] for a recent survey), their characterisations in terms of the so called alternating trees is of principal importance. Also, in many cases a suitable version of m-reducibility exists that fits a given fine hierarchy (FH). Here we show a surprising result that suitable versions of alternating trees and of m-reducibilities may be found for any given fine hierarchy, i.e. the methods of alternating trees and m-reducibilities are quite general, which is of some methodological interest for the hierarchy theory initiated in [Ad62]. The result is naturally described in terms of Priestley duality [DP94]. For simplicity, we discuss our results in this introduction only for the difference hierarchy (DH) which is the simplest and most important version of a FH; for the DH the alternating trees are simplified to alternating chains.

Let $B = (B; \cup, \cap, ^-, 0, 1)$ be a Boolean algebra and L a sublattice of $(B; \cup, \cap, 0, 1)$. Let $L(k)$ be the set of all elements $\bigcup_i (a_{2i} \setminus a_{2i+1})$ where $a_i \in L$ satisfy $a_0 \supseteq a_1 \supseteq \cdots$ and $a_k = 0$. The sequence $\{L(k)\}_{k<\omega}$ is called *the difference hierarchy over* L. It is well known that $L(k) \cup \check{L}(k) \subseteq L(k+1)$ and $\bigcup_{k<\omega} L(k)$ is the Boolean algebra generated by L, where $\check{L}(k) = \{\overline{x} \mid x \in L(k)\}$. Let us mention a couple of examples:

1. Let $\mathcal{L}$ be the class of open sets in an ω-algebraic domain X. Then $\mathcal{L}(n)$ is the class of approximable sets $A \subseteq X$ such that there is no sequence $a_0 \leq \cdots \leq a_n$ of compact elements with $a_{2i} \in A$, $a_{2i+1} \notin A$. Here $\leq$ is the specialization order. A suitable reducibility is the Wadge reducibility (i.e., the m-reducibility by continuous functions).
2. Let $\mathcal{L}$ be the level $1/2$ of the Straubing-Thérien hierarchy of regular languages over a given alphabet. Then $\mathcal{L}(n)$ is the class of languages A such that there is no sequence $a_0 \leq \cdots \leq a_n$ of words with $a_{2i} \in A$, $a_{2i+1} \notin A$. Here $\leq$ is the subword order. A suitable reducibility was not known, though for the closely related Brzozowski hierarchy such a reducibility was found in [SW05].
3. Let L be the set of existential sentences of a signature σ. Then $L(n)$ is the set of σ-sentences φ such that there is no sequence $A_0 \subseteq \cdots \subseteq A_n$ of σ-structures with $A_{2i} \models \varphi$, $A_{2i+1} \not\models \varphi$. A suitable reducibility is not known, to my knowledge.

Are there similar characterizations for an arbitrary DH? It is not obvious because there are also many examples of DH's in the literature for which the chain-characterisation was not known. Nevertheless, we show that the answer is positive. Is there a suitable reducibility that fits arbitrary given DH? Again, the answer is positive, at least for a rather broad natural class of DH's.

Along with the DH, we establish similar results for a rich class of the FH's. We also establish similar results for the hierarchies of k-partitions $A : X \to k = \{0, \ldots, k-1\}$ of a given set X to $k \geq 2$ parts

* Supported by DFG-RFBR (Grant 436 RUS 113/1002/01, 09-01-91334) and by RFBR Grant 07-01-00543a.

$(A_0, \ldots, A_{k-1})$ which were recently considered by several researchers in different fields of hierarchy theory and of computable analysis [Ko00,KW00,HW94,Hem04,Se08]. The extension to k-partitions is non-trivial because even the "right" definitions of some corresponding notions were not known.

2 Difference Hierarchy

We assume the reader to be acquainted with basic notions and facts about Stone and Priestley dualities [DP94]. Recall that a *Priestley space* $(X; \leq)$ is a compact topological space X equipped with a partial order $\leq$ such that for any $x, y \in X$ with $x \not\leq y$ there is a clopen up-set U with $x \in U \not\ni y$ (a subset U of X is up if $x \in U$ and $x \leq y$ imply $y \in U$). The *Priestley duality* states the dual equivalence between the category of bounded distributive lattices and the category of Priestley spaces as objects and the continuous monotone mappings as morphisms.

From Priestley duality it follows that for any bounded distributive lattice L the DH's over L and over the lattice $\mathcal{L}$ of clopen up-sets in the dual space $(X; \leq)$ are isomorphic (in a natural sense). By an *alternating chain* of length k for $A \subseteq X$ we mean a sequence $(x_0, \ldots, x_k)$ of elements of X such that $x_0 \leq \cdots \leq x_k$ and $x_i \in A$ iff $x_{i+1} \notin A$ for each $i < k$. Such a chain is called a *1-chain* if $x_0 \in K$. The next result shows that the method of alternating chains [Ad65] is completely general.

Theorem 1. *Let $(X; \leq)$ be a Priestley space, $A \subseteq X$ and $k \geq 0$. Then $A \in \mathcal{L}(k)$ iff A is clopen and has no 1-alternating chain of length k.*

Let $(X; \leq)$ be a Priestley space and $A, B \subseteq X$. We say that A is M-reducible to B if $A = f^{-1}(B)$ for some monotone continuous function $f : X \to X$.

Theorem 2. *Let $(X; \leq)$ be a Priestley space.*
1. *For any $n \geq 0$, $\mathcal{L}(n)$ is closed under M-reducibility.*
2. *If $\mathcal{C} = \mathcal{L}(n) \setminus \check{\mathcal{L}}(n)$ is non-empty then $\mathcal{L}(n)$ has an M-complete set and $\mathcal{C}$ forms an M-degree.*
3. *If $\check{\mathcal{L}}$ has the separation property and $\mathcal{C} = (\mathcal{L}(n+1) \cap \check{\mathcal{L}}(n+1)) \setminus (\mathcal{L}(n) \cup \check{\mathcal{L}}(n))$ is non-empty then $\mathcal{L}(n+1) \cap \check{\mathcal{L}}(n+1)$ has an M-complete set and $\mathcal{C}$ forms an M-degree.*

Remark 1. For a given DH over L, the chain characterisation and the appropriate reducibility are not unique. E.g., $(A^*; \subseteq)$ is not isomorphic to the Priestley poset for the level $1/2$ of the Straubing-Thérien hierarchy (see Introduction, Example 1). Different chain characterisations of a given DH may provide different useful information on the hierarchy. E.g., in [GS01,GSS08] two different chain characterisations of the DH over the level $1/2$ of the Brzozowski's dot-depth hierarchy were found which yield, respectively, polynomial-space and nondeterministic log-space algorithms deciding the levels of the hierarchy. qf-Reducibility from [SW05] fits the DH over the level $1/2$ of the Brzozowski's dot-depth hierarchy but it is distinct from the corresponding M-reducibility. In several cases (say in Example 1 from Introduction or for the Wagner hierarchy [Wag79]) M-reducibility coincides with reducibilities already known from the literature.

3 Fine Hierarchy

By an ω-base we mean a sequence $L = \{L_n\}_{n<\omega}$ of bounded distributive lattices such that, for any $n < \omega$, L_n is a sublattice of both L_{n+1} and $\check{L}_{n+1}$. For an ω-base L, $L^* = \bigcup_{n<\omega} L_n$ is a Boolean algebra. Let $M = \{M_n\}_{n<\omega}$ be another ω-base. By a morphism $f : L \to M$ we mean a homomorphism $f : L^* \to M^*$ of Boolean algebras such that $f(L_n) \subseteq M_n$ for each $n < \omega$. Let $\mathbf{B}_\omega$ be the category of ω-bases.

By an ω-space we mean a compact topological space X equipped with a sequence $\{\leq_n\}_{n<\omega}$ of preorders such that: for all $n < \omega$ and $x, y \in X$, $x \not\leq_n y$ implies that $x \in U \not\ni y$ for a clopen $\leq_n$-up set $U \subseteq X$ (the class of such sets is denoted $\mathcal{L}_n$); for any $n < \omega$, $x \leq_{n+1} y$ implies $x \equiv_n y$; if $x \equiv_n y$ for all $n < \omega$ then $x = y$. We denote such a space as $(X; \leq_0, \ldots)$ or, abusing notation, just by X. Let $\mathbf{S}_\omega$ denote the category with the ω-spaces as objects and continuous functions between ω-spaces which are monotone with respect to all the preorders $\leq_n$, $n < \omega$, as morphisms. We will need the following easily verified extension of Priestley duality: the categories $\mathbf{B}_\omega$ and $\mathbf{S}_\omega$ are dually equivalent.

To any ω-base $L = \{L_n\}$ one can associate (using suitable Boolean operations) in a natural way classes $S^n_\alpha \subseteq B(n < \omega, \alpha < \varepsilon_0)$ (we omit the rather technical definition from [Se08]). We call the sequence of sets

$\{S_\alpha\}_{\alpha<\varepsilon_0}$, where $S_\alpha = S^0_\alpha$, *the fine hierarchy over* L. These sets satisfy $S_\alpha \cup \check{S}_\alpha \subseteq S_\beta$ for $\alpha < \beta < \varepsilon_0$. Let $(X; \leq_0, \ldots)$ be the dual ω-space for L and let $\mathcal{L} = \{\mathcal{L}_n\}_n$ be the induced ω-base in X. Then L and $\mathcal{L}$ are canonically isomorphic, and in fact this isomorphism extends to an isomorphism of the FH's over L and $\mathcal{L}$. The FH over $\mathcal{L}$ has a characterisation that extends Theorem 1.

The alternating chains are now extended to alternating trees as follows (cf. [Se08]). Let $A \subseteq X$ and $\tau \in \omega^*$. By a *τ-aternating tree* for A we mean a family $\{p_\sigma \mid \sigma \in 2^*, |\sigma| \leq |\tau|\}$ of elements of X such that $p_\emptyset \notin A$ and $p_{\sigma 0} \notin A$, $p_{\sigma 1} \in A$, $p_\sigma \leq_{\tau(|\sigma|)} p_{\sigma k}$ for $|\sigma| < |\tau|$ and $k < 2$. Define strings $\tau^n_\alpha (n < \omega)$ by induction on $\alpha < \varepsilon_0$ as follows: $\tau^n_0 = \emptyset$, $\tau^n_{\alpha+1} = n\tau^n_\alpha$, $\tau^n_{\omega^\gamma} = \tau^{n+1}_\gamma$ for $\gamma > 0$, and $\tau^n_{\delta+\omega^\gamma} = \tau^n_{\omega^\gamma} n \tau^n_\delta$ for $\delta = \omega^\gamma \cdot \delta' > 0$, $\gamma > 0$. Let $\tau_\alpha = \tau^0_\alpha$.

Theorem 3. *Let $(X; \leq_0, \ldots)$ be an ω-space and $\alpha < \varepsilon_0$. Then the level $\mathcal{S}_\alpha$ of the fine hierarchy over $\mathcal{L}$ coincides with the class of clopen subsets of X that do not have τ_α-alternating trees.*

Also Theorem 2 has a suitable extension to the FH's; instead of M-reducibility above one has now to take the many-one reducibility by morphisms of the category $\mathbf{S}_\omega$.

4 Hierarchies of k-Partitions

In this section we briefly discuss extensions of the results about hierarchies of sets above to hierarchies of k-partitions. Levels of our hierarchies of k-partitions will be well partial ordered by inclusion. A *well preorder* (wqo) is a preorder P that has neither infinite descending chains nor infinite antichains. Well preorder theory is widely known as the wqo-theory. Recall that two preorders are equivalent if the corresponding quotient posets are isomorphic.

A k-poset is an object $(P; \leq, c)$ consisting of a poset $(P; \leq)$ and a labeling $c : P \to k$. A *morphism* $f : (P; \leq, c) \to (P'; \leq', c')$ of k-posets is a monotone function $f : (P; \leq) \to (P'; \leq')$ respecting the labelings, i.e. satisfying $c = c' \circ f$. Let $\mathcal{F}_k$ and $\mathcal{P}_k$ be the classes of all finite k-forests and finite k-posets, respectively. Define [Ko00,KW00] the preorder $\leq_h$ on $\mathcal{P}_k$ as follows: $(P, c) \leq_h (P', c')$, if there is a morphism from (P, c) to (P', c').

Let $P = (P; \leq)$ be a finite k-poset, $(X; \leq)$ be a Priestley space and $\mathcal{L}$ the class of clopen up-sets in X. Let $\mathcal{L}(P)$ be the class of partitions $A : X \to k$ such that for some P-family $\{S_p\}_{p\in P}$ of $\mathcal{L}$-sets we have $A_i = \bigcup\{\tilde{S}_p \mid c(p) = i\}$ for all $i < k$ where $\tilde{S}_p = S_p \setminus \bigcup_{q>p} S_q$. The family $\{\mathcal{L}(P)\}_{P\in\mathcal{P}_k}$ is called the DH of k-partitions over $\mathcal{L}$. It is easy to show that $P \leq_h P'$ implies $\mathcal{L}(P) \subseteq \mathcal{L}(P')$ and that $\mathcal{L}(P) = \mathcal{L}(F(P))$ where $F(P)$ is the unfolding of P to a k-forest defined in [Se04] ($F(P)$ coincides with a greatest element in $(\{F \in \mathcal{F}_k \mid F \leq_h P\}; \leq_h)$.

An advantage of our definition of the DH compared with the definition from the DH of k-partitions over posets from [Ko00] (denoted in this paragraph by $\{\mathcal{L}'(P)\}_{P\in\mathcal{P}_k}$) is that the results of Section 2 may be naturally extended to the DH of k-partitions (we omit exact formulation). Our DH of k-partitions is in fact a coarsification of the hierarchy from [Ko00]: $\mathcal{L}(P) \supseteq \bigcup\{\mathcal{L}'(Q) \mid F(Q) \leq_h P\}$. Another advantage is the fact that the collection of levels of our DH of k-partitions is well partial ordered by inclusion, in contrast to the hierarchy in [Ko00]. For the important particular case when $\mathcal{L}$ has the reduction property both definitions are equivalent (this is proved using an argument from [Se04]).

Also the results of Section 3 have natural extensions to the case of k-partitions. Because of lack of space we only give the definition of partial orders which describe the order (under inclusion) of levels of the FH of k-partitions. For a preorder $(Q; \leq)$, let $\mathcal{T}_Q$ denote the preorder formed by the finite Q-labeled trees (T, c), $c : T \to Q$, preordered by: $(T, c) \leq_h (T', c')$ iff there is a monotone function $c : T \to T'$ such that $c(x) \leq c'(f(x))$ for each $x \in T$. As is well known from the wqo-theory, if Q is a wqo then so is also $\mathcal{T}_Q$.

Define the sequence $\{\mathcal{T}_k(n)\}_{n<\omega}$ of preorders by induction on n as follows: $\mathcal{T}_k(0)$ is the antichain $\overline{k}$ on the set k, and $\mathcal{T}_k(n+1) = \mathcal{T}_{\mathcal{T}_k(n)}$. Identifying $\overline{k}$ with the minimal elements of $\mathcal{T}_k(1)$, we may assume that $\mathcal{T}_k(0)$ is an initial segment of $\mathcal{T}_k(1)$ and, moreover, $\mathcal{T}_k(n)$ is an initial segment of $\mathcal{T}_k(n+1)$ for each $n \geq 0$. Therefore, $\mathcal{T}_k(\omega) = \bigcup_{n<\omega} \mathcal{T}_k(n)$ is a wqo and $\mathcal{T}_{\mathcal{T}_k(\omega)}$ coincides with $\mathcal{T}_k(\omega)$. For any $n \leq \omega$, let $\mathcal{T}^*_k(n)$ be the class of finite subsets of $\mathcal{T}_k(n)$ preordered by: $F \leq G$ iff for any $T \in F$ there is $S \in G$ with $T \leq G$. It is easy to check that $\mathcal{T}^*_k(1)$ is equivalent to $\mathcal{F}_k$ with the h-preorder.

For an ω-space $(X; \leq_0, \ldots)$, the levels of the FH of k-partitions over the induced ω-base $\mathcal{L}$ in X are ordered by inclusion as the quotient-poset of $\mathcal{T}^*_k(\omega)$, and the results of Section 3 may be extended to this case. There are several interesting examples of such FH's. E.g., with a heavy use of one such FH we can

extend the main facts on the Wagner hierarchy of regular sets [Wag79] to the case of regular k-partitions (completing thus the results in [Se07]). In particular, we have:

Theorem 4. *The Wadge preorder on the set of ω-regular k-partitions is equivalent to $\mathcal{T}_k^*(2)$.*

References

[Ad62] J.W. Addison. The theory of hierarchies. *Logic, Methodology and Philosophy of Science*, Proc. of 1960 Int. Congress, Stanford, Palo Alto, 1962, 26–37.

[Ad65] J.W. Addison. The method of alternating chains. In: *The theory of models*, Amsterdam, North Holland, 1965, p.1–16.

[DP94] B.A. Davey and H.A. Pristley. *Introduction to Lattices and Order*. Cambridge, 1994.

[GS01] C. Glaßer amd H. Schmitz. The Boolean Structure of Dot-Depth One. *J. of Automata, Languages and Combinatorics*, 6 (2001), 437–452.

[GSS08] C. Glaßer, H. Schmitz and V. Selivanov. Efficient algorithms for membership in Boolean hierarchies of regular languages. Proceedings of STACS-2008, p. 337-348. Dagstuhl Seminar Proceedings 08001 (full version in ECCC Report TR07-094).

[Hem04] A. Hemmerling. Hierarchies of function classes defined by the first-value operator. Universität Greifswald Preprint-Reihe Mathematik 12/2004.

[HW94] P. Hertling and K. Weihrauch. Levels of degeneracy and exact lower complexity bounds for geometric algorithms. Proc. of the 6th Canadian Conf. on Computational Geometry, Saskatoon 1994, 237–242.

[Ko00] S. Kosub. *Complexity and Partitions*. PhD Thesis, Würzburg, 2000.

[KW00] S. Kosub and K. Wagner. The boolean hierarchy of NP-partitions. *STACS-2000 proceedings, Lecture Notes of Computer Science*, 1770 (2000), 157–168, Berlin, Springer.

[Se04] V.L. Selivanov. Boolean hierarchies of partitions over reducible bases. *Algebra and Logic*, 43, N 1 (2004), 44–61.

[Se07] V.L. Selivanov. Classifying omega-regular partitions. Preproceedings of LATA-2007, Universitat Rovira i Virgili Report Series, 35/07, 529–540.

[Se08] V. L. Selivanov. Fine hierarchies and m-reducibilities in theoretical computer science. *Theoretical Computer Science*, 405 (2008), 116–163.

[SW05] V.L. Selivanov and K.W. Wagner. A reducibility for the dot-depth hierarchy. *Theoretical Computer Science*, 345, N 2-3 (2005), 448–472.

[Wag79] K. Wagner. On ω-regular sets. *Inform. and Control*, 43 (1979), 123–177.

On Transitive Closure Operators in Finite Order Logic

Artur Wdowiarski

Department of Logic, Institute of Philosophy, Warsaw University, Krakowskie Przedmieście 3, 00-047 Warsaw, Poland
arturwdowiarski@uw.edu.pl

The approach presented in this paper is a generalization of Immerman's result from [Imm87], i.e. the fact that first order logic with first order transitive closure operator captures NLOGSPACE. We discuss logics of finite orders with transitive closure operators of various orders. These logics are intuitive and productive tools for describing some interesting space complexity classes. The general inspiration follows from Mostowski's work presented in [Most01] and devoted to studying sublogics of finite order logic in finite models. The ideas of that work were further developed in application to time complexity classes in Kołodziejczyk's papers [Koł04a] and [Koł04b].

We consider arithmetical models on finite initial segments $U = \{0, \ldots, m-1\}$. These models are supplied with standard arithmetical operations such as addition and multiplication, treated as ternary relations. Let us observe that addition and multiplication are definable by means of the successor relation in the logic of first order transitive closure. In the models under consideration, the greatest element $m-1$ is denoted by max.

We define types as follows. ι is a type (the basic type) and, if $\tau_1, \ldots, \tau_k$ are types, then $\tau = (\tau_1, \ldots, \tau_k)$ is also a type. The corresponding universes are defined as follows. $U_\iota = U$, and $U_\tau = P(U_{\tau_1} \times \ldots \times U_{\tau_k})$. Each type τ determines the corresponding function $h_\tau \colon \mathbb{N} \to \mathbb{N}$. $h_\iota(m) = m$, and $h_\tau(m) = 2^{h_{\tau_1} \cdot \ldots \cdot h_{\tau_k}}$. Then we show, using a technique suggested by Marcin Mostowski, that finite arithmetic over $U = \{0, \ldots, m-1\}$ in the logic with quantification restricted to a type τ and its subtypes is equivalent to first order arithmetic over $U' = \{0, \ldots, h_\tau(m) - 1\}$. That technique involves an alternative way of proving one of Zdanowski's results from [Zd05].

We define the order of a type τ, denoted by $ord(\tau)$, as follows. $ord(\iota) = 1$, and if $\tau = (\tau_1, \ldots, \tau_k)$, then $ord(\tau) = max\{ord(\tau_i) : 1 \leq i \leq k\}$. Finite order logic, L_ω, is the logic that allows quantification over variables of arbitrary types. The n-th order logic, L_n, is said to be the sublogic of L_ω with quantification restricted to viariables of types with orders of at most n.

For each type τ and each binary relation between objects of type τ (or k-tuples of such objects) we define the transitive closure of that relation. We introduce transitive closure operators that applied to a formula defining the relation R denote the transitive closure of R. We say that a transitive closure operator is of order n and that it has arity k iff it is applicable to formulas describing relations R of order n and with arity $2k$, i.e. such that $R \in U_\tau$, for some type τ of order n, where $\tau = (\tau_1, \ldots, \tau_{2k})$.

We consider logics $L_n(TC_{n+1})$ or order n with added transitive closure operator of order $n+1$. The syntax for such logics is similar to that of the first order logic of transitive closure, as presented in, for example, [Imm87]. We also consider sublogics of $L_n(TC_{n+1})$ consisting of only positive formulas. They are of the form

$$[TC\varphi(P_1, P_2)](0, \mathrm{MAX}) \quad ,$$

where φ defines a relation of type (τ, τ), 0 is the empty relation of type τ and MAX is the full relation of type τ (they can be defined by „$x \neq x$" and „$x = x$", respectively). Such sublogic of $L_n(TC_{n+1})$ is denoted by $L_n(posTC_{n+1})$.

We also introduce transitive closure operators restricted by some functions – TC^f. Such operators ignore bits beyond $f(m)$ in models of cardinality m. We achieve this by defining inductively orders $<_\tau$ for each type τ. $<_\iota$ is simply $<$ and if x and y are variables of type $\tau = (\tau_1, \ldots, \tau_k)$, then

$$x <_\tau y \equiv_{df} \exists \overline{z}\, [\forall \overline{w} <_{\tau_1, \ldots, \tau_k} \overline{z}\, (x(\overline{w}) \equiv y(\overline{w})) \wedge \neg x(\overline{z}) \wedge y(\overline{z})] \quad ,$$

where $\overline{v} = (v_1, \ldots, v_k)$ and $\overline{w} <_{\tau_1, \ldots, \tau_k} \overline{z}$ is a shortcut for

$$\begin{gathered} w_1 <_{\tau_1} z_1 \vee \\ \vee (w_1 =_{\tau_1} z_1 \wedge w_2 <_{\tau_2} z_2) \vee \\ \vee (w_1 =_{\tau_1} z_1 \wedge w_2 =_{\tau_2} z_2 \wedge w_3 <_{\tau_3} z_3) \vee \ldots \\ \ldots \vee (w_1 =_{\tau_1} z_1 \wedge w_2 =_{\tau_2} z_2 \wedge \ldots \wedge w_k <_{\tau_k} z_k) \quad , \end{gathered}$$

where $v =_{\tau_i} v'$ is a shortcut for $\neg (v <_{\tau_i} v') \wedge \neg (v' <_{\tau_i} v)$. Having defined such orders for all types, we say that a transitive closure operator is restricted by some function f such that f is everywhere-dominated by the function h_τ, for some type τ, and denote it by TC^f, iff TC^f can be applied only to formulas with all the quantifiers restricted to the $f(max)$-th element of the order $<_\tau$. We denote the n-th order logic with the $n+1$-th order transitive closure operator of arity k and restricted to f by $L_n\left(TC^{f,k}_{n+1}\right)$ and its sublogic consisting only of positive formulas by $L_n\left(posTC^{f,k}_{n+1}\right)$. If we allow arbitrary arity k, we get $L_n\left(TC^{f,\omega}_{n+1}\right)$ and $L_n\left(posTC^{f,\omega}_{n+1}\right)$, respectively.

We consider space complexity classes as classes of finite arithmetical models recognized by Turing machines with binary memory of the size $\leq f(m)$, for a proper function f. Thus $NSPACE(f)$ is smaller than $NSPACE(\mathcal{O}(f))$. Essentially we consider nondeterministic space complexity classes.

The generalization of Immerman's result mentioned in the first paragraph consists in following the idea visualized in the figure below.

Input data:
$\overbrace{\qquad\qquad}^{n}$
Available working memory:
$\overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{h_\tau(n)}$
Restricted working memory:
$\overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{f(n)}$

The idea is that if we have an input model of size n (its universe being the initial segment of natural numbers) and allow the transitive closure operator acting on relations R between objects of type τ, we operate in logic $FO(TC)$ over models of size $h_\tau(n)$. Then, if we limit formulas defining relations R to those with quantifiers restricted to $f(max)$, we move to models of size $f(n)$. Having established that, we can prove the following theorems, using well known techniques from, for example [EF95].

Theorem 1. *For every arithmetically definable and space constructible function f such that there is a function h_τ such that for all $m \in \mathbb{N}$ $g(m) \leq h_\tau(m)$, where $g(x) = f(x) + \log f(x) + c + d\log(x)$, for some c and d, if $\tau = (\tau_1, \ldots, \tau_k)$ and $ord(\tau) = n$, then:*

$$NSPACE(f) \subseteq L_n\left(posTC^{g,1}_{n+1}\right)$$

Theorem 2. *For every arithmetically definable and space constructible function f such that there is a function h_τ such that for all $m \in \mathbb{N}$ $f(m) \leq h_\tau(m)$, if $ord(\tau) = n$, then:*

$$L_n\left(posTC^{f,1}_{n+1}\right) \subseteq SPACE(2f + \mathcal{O}(logf))$$

Theorem 3. *For f,h_τ and g like in Theorem 1,*

$$L_n\left(TC^{f,1}_{n+1}\right) \subseteq SPACE(\mathcal{O}(f)) \subseteq L_n\left(TC^{g,\omega}_{n+1}\right),$$

References

[EF95] Ebbinghaus H.D. and Flum J.: Finite Model Theory. Springer Monographs in Mathematics. Springer, Berlin, 1995.

[Imm87] Immerman N.: Languages That Capture Complexity Classes. SIAM J. of Computing 16:4 (1987), 760-778.

[Koł04a] Kołodziejczyk L.A.: A finite model-theoretical proof of a property of bounded query classes within PH. JSL 69 (2004), 1105-1116.

[Koł04b] Kołodziejczyk L.A.: Truth definitions in finite models. JSL 69 (2004), 183-200.

[Most01] Mostowski M.: On representing concepts in finite models. MLQ 47(2001), 513-523;

[Zd05] Zdanowski K.: Arithmetics in finite but potentially infinite worlds. Ph.D. Thesis, Warsaw University, 2005

Polynomial Hierarchy, Betti Numbers and a Real Analogue of Toda's Theorem *

Saugata Basu[1] and Thierry Zell[2]

[1] Department of Mathematics, Purdue University, West Lafayette, IN 47906, U.S.A. sbasu@math.purdue.edu
[2] School of Mathematics and Computing Sciences, Lenoir-Rhyne University, Hickory, NC 28603
thierry.zell@lr.edu

Abstract. Toda [10] proved in 1989 that the (discrete) polynomial time hierarchy, **PH**, is contained in the class $\mathbf{P}^{\#\mathbf{P}}$, namely the class of languages that can be decided by a Turing machine in polynomial time given access to an oracle with the power to compute a function in the counting complexity class $\#\mathbf{P}$.
We prove an analogous result in the complexity theory over the reals. Unlike Toda's proof in the discrete case, which relied on sophisticated combinatorial arguments, our proof is topological in nature.
Keywords: Polynomial hierarchy, Betti numbers, Semi-algebraic sets, Toda's theorem.

1 Introduction

The primary motivation for our result comes from classical (i.e. discrete) computational complexity theory. In that setting, a seminal result due to Toda [10] links the complexity of counting with that of deciding sentences with a fixed number of quantifier alternations.

Theorem 1 (Toda [10]). $\mathbf{PH} \subset \mathbf{P}^{\#\mathbf{P}}$.

Here, **PH** denotes the (discrete) polynomial hierarchy, and $\#\mathbf{P}$ is the counting class associated with the decision problems in **NP**: it can be defined as the set of functions $f(x)$ which, for any input x, return the number of accepting paths for the input x in some non-deterministic Turing machine. Thus, Toda's theorem asserts that any language in the polynomial hierarchy can be decided by a Turing machine in polynomial time, given access to an oracle with the power to compute a function in $\#\mathbf{P}$. (Only one call to the oracle is required in the proof.)

While it is obvious that the classes $\mathbf{P}, \mathbf{NP}, \mathbf{coNP}$ are contained in $\mathbf{P}^{\#\mathbf{P}}$, the proof for the higher levels of the polynomial hierarchy is quite non-trivial, and requires previous results of Schöning [8], and Valiant and Vazirani [11]. We prove the following real analogue of Theorem 1.

Theorem 2 (Real analogue of Toda's theorem). $\mathbf{PH}_{\mathrm{R}}^{c} \subset \mathbf{P}_{\mathrm{R}}^{\#\mathbf{P}_{\mathrm{R}}^{\dagger}}$.

Establishing such a result is not straightforward, for the following reasons.

- While the problem of defining the real polynomial hierarchy $\mathbf{PH}_{\mathrm{R}}$ is straightforward, there is no consensus on the "right" generalization of the counting class $\#\mathbf{P}$ to the reals. Our choice, which we denoted $\#\mathbf{P}_{\mathrm{R}}^{\dagger}$ (Definition 3), is crucial to the proof.
- Because of the presence of an intermediate complexity class in Toda's original proof, there seems to be no direct way of extending such a proof to real complexity classes in the sense of Blum-Shub-Smale model of computation [2, 9].

Thus, the proof of Theorem 2 proceeds along completely different lines; it is mainly topological in nature, and technical reasons require us to consider a compact restriction $\mathbf{PH}_{\mathrm{R}}^{c}$ of the real polynomial hierarchy.

* A more complete extended abstract appears in the proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '09). The full paper will appear in *Foundations of Computational Mathematics*.

2 Real Machines and Complexity Classes

We let R be the real field (or, more generally, any real-closed field).

2.1 Real Turing Machines

In the late eighties Blum, Shub and Smale [2, 9] introduced the notion of Turing machines over more general fields, thereby generalizing the classical problems of computational complexity theory such as **P** vs **NP** to corresponding problems over arbitrary fields (such as the real, complex, p-adic numbers etc.) In particular, Blum, Shub, and Smale (henceforth B-S-S) proved the $\mathbf{NP}_{\mathrm{R}}$-completeness of the problem of deciding whether a real polynomial equation in many variables of degree at most four has a real solution (this is the real analogue of Cook-Levin's theorem that the satisfiability problem is **NP**-complete in the discrete case), and subsequently through the work of several researchers (Koiran, Bürgisser, Cucker, Meer to name a few) a well-established complexity theory over the reals has been built up, mirroring closely the discrete case.

2.2 Real Analogue of P

The analogue of the polynomial class for the reals is well known.

Definition 1. *Let $k(n)$ be any polynomial in n. A sequence of semi-algebraic sets $(T_n \subset \mathrm{R}^{k(n)})_{n>0}$ is said to belong to the class $\mathbf{P}_{\mathrm{R}}$ if there exists a Turing machine M over* R *(see [2, 1]), such that for all $\mathbf{x} \in \mathrm{R}^{k(n)}$, the machine M tests membership of $\mathbf{x}$ in T_n in time bounded by a polynomial in n.*

2.3 The Compact Hierarchy $\mathbf{PH}^c_{\mathrm{R}}$

For technical reasons linked to the topological methods used we need to restrict to compact semi-algebraic sets, and for this purpose, we will now define a compact analogue of $\mathbf{PH}_{\mathrm{R}}$ that we will denote $\mathbf{PH}^c_{\mathrm{R}}$. Unlike in the non-compact case, we will assume all variables vary over certain compact semi-algebraic sets (namely spheres of varying dimensions).

Notation 3. We denote by $\mathbf{S}^k(0, r)$ the sphere in R^{k+1} of radius r centered at the origin, and by $\mathbf{S}^k$ the unit sphere $\mathbf{S}^k(0, 1)$.

Definition 2 (Compact real polynomial hierarchy). *Let*

$$k(n), k_1(n), \ldots, k_\omega(n)$$

be polynomials in n. A sequence of semi-algebraic sets $(S_n \subset \mathbf{S}^{k(n)})_{n>0}$ is in the complexity class $\mathbf{\Sigma}^c_{\mathrm{R},\omega}$, if for each $n > 0$ the semi-algebraic set S_n is described by a first order formula

$$(Q_1 \mathbf{Y}^1 \in \mathbf{S}^{k_1(n)}) \cdots (Q_\omega \mathbf{Y}^\omega \in \mathbf{S}^{k_\omega(n)}) \phi_n(X_0, \ldots, X_{k(n)}, \mathbf{Y}^1, \ldots, \mathbf{Y}^\omega), \tag{1}$$

with ϕ_n a quantifier-free first order formula defining a closed *semi-algebraic subset of $\mathbf{S}^{k_1(n)} \times \cdots \times \mathbf{S}^{k_\omega(n)} \times \mathbf{S}^{k(n)}$ and for each $i, 1 \leq i \leq \omega$, $\mathbf{Y}^i = (Y^i_0, \ldots, Y^i_{k_i})$ is a block of $k_i(n)+1$ variables, $Q_i \in \{\exists, \forall\}$, with $Q_j \neq Q_{j+1}, 1 \leq j < \omega$, $Q_1 = \exists$, and the sequence of semi-algebraic sets $(T_n \subset \mathbf{S}^{k_1(n)} \times \cdots \times \mathbf{S}^{k_\omega(n)} \times \mathbf{S}^{k(n)})_{n>0}$ defined by the formulas $(\phi_n)_{n>0}$ belongs to the class $\mathbf{P}_{\mathrm{R}}$.*

We define the class $\mathbf{\Pi}^c_{\mathrm{R},\omega}$ as above, with the exception that the alternating quantifiers in (1) *start with $Q_1 = \forall$. Finally define the* ***compact real polynomial time hierarchy*** *to be the union*

$$\mathbf{PH}^c_{\mathrm{R}} \stackrel{def}{=} \bigcup_{\omega \geq 0} (\mathbf{\Sigma}^c_{\mathrm{R},\omega} \cup \mathbf{\Pi}^c_{\mathrm{R},\omega}) = \bigcup_{\omega \geq 0} \mathbf{\Sigma}^c_{\mathrm{R},\omega} = \bigcup_{\omega \geq 0} \mathbf{\Pi}^c_{\mathrm{R},\omega}.$$

Notice that the semi-algebraic sets belonging to any language in $\mathbf{PH}^c_{\mathrm{R}}$ are all semi-algebraic compact (in fact closed semi-algebraic subsets of spheres). Also, note the inclusion

$$\mathbf{PH}^c_{\mathrm{R}} \subset \mathbf{PH}_{\mathrm{R}}.$$

Remark 1. Even though the restriction to compact semi-algebraic sets might appear to be only a technicality at first glance, this is actually an important restriction. For instance, it is a long-standing open question in real complexity theory whether there exists an $\mathbf{NP}_{\mathrm{R}}$-complete problem which belongs to the class $\mathbf{\Sigma}^c_{\mathrm{R},1}$ (the compact version of the class $\mathbf{NP}_{\mathrm{R}}$).

2.4 Example

The following is an example of a language in $\mathbf{\Sigma}^c_{\mathrm{R},1}$ (i.e. the compact version of $\mathbf{NP}_{\mathrm{R}}$). Let $k(n) = \binom{n+4}{4} - 1$ and identify $\mathrm{R}^{k(n)+1}$ with the space of *homogeneous* polynomials in $\mathrm{R}[X_0, \ldots, X_n]$ of degree 4. Let $S_n \subset \mathbf{S}^{k(n)} \subset \mathrm{R}^{k(n)+1}$ be defined by

$$S_n = \{P \in \mathbf{S}^{k(n)} \mid \exists \mathbf{x} = (x_0 : \cdots : x_n) \in \mathbb{P}^n_{\mathrm{R}} \text{ with } P(\mathbf{x}) = 0\};$$

in other words S_n is the set of (normalized) real forms of degree 4 which have a zero in the real projective space $\mathbb{P}^n_{\mathrm{R}}$. Then

$$(S_n \subset \mathbf{S}^{k(n)})_{n>0} \in \mathbf{\Sigma}^c_{\mathrm{R},1},$$

since it is easy to see that S_n also admits the description:

$$S_n = \{P \in \mathbf{S}^{k(n)} \mid \exists \mathbf{x} \in \mathbf{S}^n \text{ with } P(\mathbf{x}) = 0\}.$$

Note that it is *not known* if $(S_n \subset \mathbf{S}^{k(n)})_{n>0}$ is $\mathbf{NP}_{\mathrm{R}}$-complete while the non-compact version of this language i.e. the language consisting of (possibly non-homogeneous) polynomials of degree at most four having a zero in $\mathbb{A}^n_{\mathrm{R}}$ (instead of $\mathbb{P}^n_{\mathrm{R}}$), has been shown to be $\mathbf{NP}_{\mathrm{R}}$-complete [1].

2.5 Real Analogue of #P

In order to define real analogues of counting complexity classes of discrete complexity theory, it is necessary to identify the proper notion of "counting" in the context of semi-algebraic geometry. Counting complexity classes over the reals have been defined previously by Meer [7], and studied extensively by other authors [4]. These authors used a straightforward generalization to semi-algebraic sets of counting in the case of finite sets – namely the counting function took the value of the cardinality of a semi-algebraic set if it happened to be finite, and ∞ otherwise. This is in our view not a fully satisfactory generalization since the count gives no information when the semi-algebraic set is infinite, and most interesting semi-algebraic sets have infinite cardinality. Moreover, no real analogue of Toda's theorem has been proved using this definition of counting.

Notation 4. For any semi-algebraic set $S \subset \mathrm{R}^k$ we denote by $b_i(S)$ the i-th Betti number (that is the rank of the singular homology group $\mathrm{H}_i(S) = \mathrm{H}_i(S, \mathbb{Z})$) of S. We also let $P_S \in \mathbb{Z}[T]$ denote the ***Poincaré polynomial*** of S, namely $P_S(T) \stackrel{\text{def}}{=} \sum_{i \geq 0} b_i(S)\, T^i$.

The polynomial $P_S(T)$ clearly generalizes counting. Indeed, $P_S(0)$ counts the number of connected components of S, and thus the cardinality of S when S is a finite set of points.

Remark 2. The problems of "counting" varieties and computing their Betti numbers are connected at a deeper, more direct level over fields of positive characteristic via the zeta function. Thus our choice of definition for a real analogue of $\#\mathbf{P}$ is not altogether ad hoc.

The above considerations motivate us to depart from the definition of $\#\mathbf{P}_{\mathrm{R}}$ considered previously in [7, 4]. We denote our class $\#\mathbf{P}^{\dagger}_{\mathrm{R}}$ to avoid any possible confusion with these authors' work.

Definition 3 (The class $\#\mathbf{P}^{\dagger}_{\mathrm{R}}$). *We say a sequence of functions $(f_n : \mathrm{R}^n \to \mathbb{Z}[T])_{n>0}$ is in the class $\#\mathbf{P}^{\dagger}_{\mathrm{R}}$, if there exists a language $(S_n \subset \mathrm{R}^n)_{n>0} \in \mathbf{P}_{\mathrm{R}}$, as well as a polynomial $m(n)$, such that*

$$f_n(\mathbf{x}) = P_{S_{m+n,\mathbf{x}}}(T)$$

for each $\mathbf{x} \in \mathrm{R}^n$, where $S_{m+n,\mathbf{x}}$ is the fiber $S_{m+n} \cap \pi^{-1}(\mathbf{x})$ of the projection along the first m co-ordinates $\pi : \mathrm{R}^{m+n} \to \mathrm{R}^n$, and $P_{S_{m+n,\mathbf{x}}}(T)$ is the corresponding Poincaré polynomial.

Remark 3. Note that the class $\#\mathbf{P}^{\dagger}_{\mathrm{R}}$ is quite robust. For instance, given two sequences $(f_n)_{n>0}, (g_n)_{n>0} \in \#\mathbf{P}^{\dagger}_{\mathrm{R}}$ it follows (by taking disjoint union of the corresponding semi-algebraic sets) that $(f_n + g_n)_{n>0} \in \#\mathbf{P}^{\dagger}_{\mathrm{R}}$, and also $(f_n g_n)_{n>0} \in \#\mathbf{P}^{\dagger}_{\mathrm{R}}$ (by taking Cartesian product of the corresponding semi-algebraic sets and using the multiplicative property of the Poincaré polynomials, which itself is a consequence of the Kunneth formula in homology theory.)

3 Main Ideas and Techniques

3.1 Topological Constructions

Our main tool is a topological construction which, given a semi-algebraic set $S \subset \mathrm{R}^{m+n}$, $p \geq 0$, and $\pi_{\mathbf{Y}} : \mathrm{R}^{m+n} \subset \mathrm{R}^n$ the projection along (say) the $\mathbf{Y}$ co-ordinates, constructs *efficiently* a semi-algebraic set, $D^p_{\mathbf{Y}}(S)$, such that

$$b_i(\pi_{\mathbf{Y}}(S)) = b_i(D^p_{\mathbf{Y}}(S)), \quad 0 \leq i < p \tag{2}$$

A precursor to this construction appears in [6], based on the so called "descent" spectral sequence which, under the right conditions, converges to the homology of a set given by a quantified formula.

In this paper we need to be able to recover exactly (not just bound) the Betti numbers of $\pi_{\mathbf{Y}}(S)$ from those of $D^p_{\mathbf{Y}}(S)$. Moreover, it is very important in our context that membership in the semi-algebraic set $D^p_{\mathbf{Y}}(S)$ should be checkable in polynomial time, given that the same is true for S. Notice that even if there exists an efficient (i.e. polynomial time) algorithm for checking membership in S, the same need not be true for the image $\pi_{\mathbf{Y}}(S)$.

3.2 Case of One Quantifier

First consider the class $\mathbf{\Sigma}^c_{\mathrm{R},1}$. Consider a closed semi-algebraic set $S \subset \mathbf{S}^k \times \mathbf{S}^\ell$ defined by a quantifier-free formula $\phi(\mathbf{X}, \mathbf{Y})$ and let $\pi_{\mathbf{Y}} : \mathbf{S}^k \times \mathbf{S}^\ell \to \mathbf{S}^k$ be the projection map along the $\mathbf{Y}$ variables.

Then the formula $\Phi(\mathbf{X}) = \exists \mathbf{Y}\ \phi(\mathbf{X}, \mathbf{Y})$ is satisfied by $\mathbf{x} \in \mathbf{S}^k$ if and only if $b_0(S_{\mathbf{x}}) \neq 0$, where $S_{\mathbf{x}} = S \cap \pi_{\mathbf{Y}}^{-1}(\mathbf{x})$. Thus, the problem of deciding the truth of $\Phi(\mathbf{x})$ is reduced to computing a Betti number (the 0-th) of the fiber of S over $\mathbf{x}$.

Now consider the class $\mathbf{\Pi}^c_{\mathrm{R},1}$. Using the same notation as above we have that the formula $\Psi(\mathbf{X}) = \forall \mathbf{Y}\ \phi(\mathbf{X}, \mathbf{Y})$ is satisfied by $\mathbf{x} \in \mathbf{S}^k$ if and only if the formula $\neg\Psi(\mathbf{X}) = \exists \mathbf{Y}\ \neg\phi(\mathbf{X}, \mathbf{Y})$ does not hold, which means, according to the previous case, that we have $b_0(\mathbf{S}^\ell \setminus S_{\mathbf{x}}) = 0$, which is equivalent to $b_\ell(S_{\mathbf{x}}) = 1$. Notice that, as before, the problem of deciding the truth of $\Psi(\mathbf{x})$ is reduced to computing a Betti number (the ℓ-th) of the fiber of S over $\mathbf{x}$.

3.3 Case of Two Quantifiers

Consider the class $\mathbf{\Pi}^c_{\mathrm{R},2}$ and let $S \subset \mathbf{S}^k \times \mathbf{S}^\ell \times \mathbf{S}^m$ be a closed semi-algebraic set defined by a quantifier-free formula $\phi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ and let $\pi_{\mathbf{Z}} : \mathbf{S}^k \times \mathbf{S}^\ell \times \mathbf{S}^m \to \mathbf{S}^k \times \mathbf{S}^\ell$ be the projection map along the Z variables, and $\pi_{\mathbf{Y}} : \mathbf{S}^k \times \mathbf{S}^\ell \to \mathbf{S}^k$ be the projection map along the Y variables as before. Consider the formula

$$\Phi(\mathbf{X}) = \forall \mathbf{Y}\ \exists \mathbf{Z}\ \phi(\mathbf{X}, \mathbf{Y}, \mathbf{Z}).$$

This formula can be recast as:

$$\Phi(\mathbf{X}) = \forall \mathbf{Y}\ (\mathbf{X}, \mathbf{Y}) \in \pi_{\mathbf{Z}}(S).$$

For any $\mathbf{x} \in \mathbf{S}^k$, $\Phi(\mathbf{x})$ holds if the $\pi_{\mathbf{Y}}$ fiber $(\pi_{\mathbf{Z}}(S))_{\mathbf{x}}$ is equal to $\mathbf{S}^\ell$. This can be formulated in terms of Betti numbers by the condition: $b_\ell\left((\pi_{\mathbf{Z}}(S))_{\mathbf{x}}\right) = 1$. The construction mentioned in (2) gives, for $p = \ell + 1$, the existence of a semi-algebraic set $D^{\ell+1}_{\mathbf{Z}}(S)$ such that $b_\ell(D^{\ell+1}_{\mathbf{Z}}(S)) = b_\ell(\pi_{\mathbf{Z}}(S))$. Fortunately, the construction of the set $D^{\ell+1}_{\mathbf{Z}}(S)$ is compatible with taking fibers, so that we have, for all $\mathbf{x} \in \mathbf{S}^k$,

$$b_\ell\left((\pi_{\mathbf{Z}}(S))_{\mathbf{x}}\right) = b_\ell\left(D^{\ell+1}_{\mathbf{Z}}(S)_{\mathbf{x}}\right).$$

Thus for any $\mathbf{x} \in \mathbf{S}^k$, the truth or falsity of $\Phi(\mathbf{x})$ is determined by a certain Betti number of the fiber $D^{\ell+1}_{\mathbf{Z}}(S)_{\mathbf{x}}$ over $\mathbf{x}$ of a certain semi-algebraic set $D^{\ell+1}_{\mathbf{Z}}(S)$ which can be constructed efficiently in terms of the set S.

3.4 General Case

The idea behind the proof of the main theorem is a recursive application of the above argument in case when the number of quantifier alternations is larger (but still bounded by some constant) while keeping track of the growth in the sizes of the intermediate formulas and also the number of quantified variables.

References

1. L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and real computation*, Springer-Verlag, New York, 1998, With a foreword by Richard M. Karp. MR 99a:68070
2. L. Blum, M. Shub, and S. Smale, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*, Bull. Amer. Math. Soc. (N.S.) **21** (1989), no. 1, 1–46. MR 90a:68022
3. Peter Bürgisser and Felipe Cucker, *Variations by complexity theorists on three themes of Euler, Bézout, Betti, and Poincaré*, Complexity of computations and proofs (Jan Krajicek, ed.), Quad. Mat., vol. 13, Dept. Math., Seconda Univ. Napoli, Caserta, 2004, pp. 73–151. MR 2131406 (2006c:68053)
4. ———, *Counting complexity classes for numeric computations. II. Algebraic and semialgebraic sets*, J. Complexity **22** (2006), no. 2, 147–191. MR 2200367 (2007b:68059)
5. Peter Bürgisser, Felipe Cucker, and Martin Lotz, *Counting complexity classes for numeric computations. III. Complex projective sets*, Found. Comput. Math. **5** (2005), no. 4, 351–387. MR 2189543 (2006h:68039)
6. A. Gabrielov, N. Vorobjov, and T. Zell, *Betti numbers of semialgebraic and sub-Pfaffian sets*, J. London Math. Soc. (2) **69** (2004), no. 1, 27–43. MR 2025325 (2004k:14105)
7. Klaus Meer, *Counting problems over the reals*, Theoret. Comput. Sci. **242** (2000), no. 1-2, 41–58. MR 1769145 (2002g:68041)
8. Uwe Schöning, *Probabilistic complexity classes and lowness*, J. Comput. System Sci. **39** (1989), no. 1, 84–100. MR 1013721 (91b:68041a)
9. Michael Shub and Steve Smale, *On the intractability of Hilbert's Nullstellensatz and an algebraic version of "*NP $\neq$ P*?"*, Duke Math. J. **81** (1995), no. 1, 47–54 (1996), A celebration of John F. Nash, Jr. MR 1381969 (97h:03067)
10. Seinosuke Toda, *PP is as hard as the polynomial-time hierarchy*, SIAM J. Comput. **20** (1991), no. 5, 865–877. MR 1115655 (93a:68047)
11. L. G. Valiant and V. V. Vazirani, *NP is as easy as detecting unique solutions*, Theoret. Comput. Sci. **47** (1986), no. 1, 85–93. MR 871466 (88i:68021)

Noncomputable Functions in the Blum-Shub-Smale Model

Wesley Calvert[1], Ken Kramer[2] and Russell Miller[2⋆]

[1] Murray State University
Murray, Kentucky 42071 USA
wesley.calvert@murraystate.edu
http://campus.murraystate.edu/academic/faculty/wesley.calvert
[2] Queens College of CUNY
65-30 Kissena Blvd., Flushing, NY 11367 USA
and the CUNY Graduate Center
365 Fifth Avenue, New York, NY 10016 USA
kkramer@qc.cuny.edu & Russell.Miller@qc.cuny.edu
http://qcpages.qc.cuny.edu/~rmiller

Abstract. We answer several questions of Meer and Ziegler about the Blum-Shub-Smale model of computation on $\mathbb{R}$: the set $\mathbb{A}_d$ of algebraic numbers of degree $\leq d$ is not decidable in $\mathbb{A}_{d-1}$, and the BSS halting problem is not decidable in any countable oracle.

Key words: Blum-Shub-Smale model, computability, real computation.

1 Introduction

Blum, Shub, and Smale introduced in [2] a notion of computation with full-precision real arithmetic, in which the ordered field operations are axiomatically computable, and the computable functions are closed under the usual operations. A more complete account of this model is given in [1].

The key question for this paper was posed by Meer and Ziegler in [5]. Section 2 gives the basic technical result, Lemma 1, applied in Section 3 to Question 1.

Question 1 (Meer-Ziegler). Let $\mathbb{A}_d$ be the set of algebraic numbers with degree (over $\mathbb{Q}$) at most d. Then is it true that

$$\mathbb{A}_0 \lneq_{BSS} \mathbb{A}_1 \lneq_{BSS} \cdots \mathbb{A}_d \lneq_{BSS} \cdots?$$

$\mathbb{A}_{d-1} \leq_{BSS} \mathbb{A}_d$ is clear: if $x \in \mathbb{A}_d$, find its minimal polynomial in $\mathbb{Q}[X]$; while if $x \notin \mathbb{A}_d$ then $x \notin \mathbb{A}_{d-1}$. The question asks if $\mathbb{A}_d \leq_{BSS} \mathbb{A}_{d-1}$.

2 BSS-Computable Functions At Transcendentals

Here we introduce our basic method for showing that various functions on the real numbers fail to be BSS-computable. In many respects, it is equivalent to the method, used by many others (see for example [1]), of considering BSS computations as paths through a finite-branching tree of countable height, branching whenever there is a forking instruction in the program. However, we believe our method can be more readily understood by a mathematician unfamiliar with computability theory.

Lemma 1. *Let M be a BSS-machine, and $\boldsymbol{z}$ the finite tuple of real parameters mentioned in the program for M. Suppose that $\boldsymbol{y} \in \mathbb{R}^{m+1}$ is a tuple of real numbers algebraically independent over the field $Q = \mathbb{Q}(\boldsymbol{z})$, such that M converges on input $\boldsymbol{y}$. Then there exists $\epsilon > 0$ and rational functions $f_0, \ldots, f_n \in Q(\boldsymbol{Y})$, (that is, rational functions of the variables $\boldsymbol{Y}$ with coefficients from Q) such that for all $\boldsymbol{x} \in \mathbb{R}^{m+1}$ in the ϵ-ball $B_\epsilon(\boldsymbol{y})$, M converges on input $\boldsymbol{x}$ with output $\langle f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\rangle \in \mathbb{R}^{n+1}$.*

⋆ The corresponding author was supported by Grants # 90927-08 08 from the Queens College Research Enhancement Program, # 61467-00 39 and # 62632-00 40 from the PSC-CUNY Research Award Program, and (along with the first author) # 13397 from the Templeton Foundation.

Proof. The intuition is that by choosing $\boldsymbol{x}$ sufficiently close to $\boldsymbol{y}$, we can ensure that the computation on $\boldsymbol{x}$ branches in exactly the same way as the computation on $\boldsymbol{y}$, at each of the (finitely many) branch points in the computation on $\boldsymbol{y}$. Say that the run of M on input $\boldsymbol{y}$ halts at stage t, and that at each stage $s \leq t$, the non-blank cells contain the reals $\langle f_{0,s}(\boldsymbol{y}), \ldots, f_{n_s,s}(\boldsymbol{y})\rangle$. Each $f_{i,s}$ is a rational function in $Q(\boldsymbol{Y})$, uniquely determined, since $\boldsymbol{y}$ is algebraically independent over Q. Let $F = \{f_{i,s}(\boldsymbol{Y}) : s \leq t \ \&\ i \leq n_s \ \&\ f_{i,s} \notin Q\}$ be the finite set of nonconstant rational functions used in the computation. For each $f_{i,s} \in F$, the preimage $f_{i,s}^{-1}(0)$ is closed in $\mathbb{R}^{m+1}$, and therefore so is the finite union U of all these $f_{i,s}^{-1}(0)$. By algebraic independence, $\boldsymbol{y} \notin U$, so there exists an $\epsilon > 0$ with $B_\epsilon(\boldsymbol{y}) \cap U = \emptyset$. Indeed, for all $f_{i,s} \in F$ and all $\boldsymbol{x} \in B_\epsilon(\boldsymbol{y})$, $f_{i,s}(\boldsymbol{x})$ and $f_{i,s}(\boldsymbol{y})$ must have the same sign. Therefore, for any $\boldsymbol{x} \in B_\epsilon(\boldsymbol{y})$, it is clear that in the run of M on input $\boldsymbol{x}$, at each stage $s \leq t$, the cells will contain precisely $\langle f_{0,s}(\boldsymbol{x}), \ldots, f_{n_s,s}(\boldsymbol{x})\rangle$ and the machine will be in the same state in which it was at stage s on input $\boldsymbol{y}$. Therefore, at stage t, the run of M on input $\boldsymbol{x}$ must also have halted, with $\langle f_{0,t}(\boldsymbol{x}), \ldots, f_{n_t,t}(\boldsymbol{x})\rangle$ in its cells as the output. □

Lemma 1 provides quick proofs of several known results, including the undecidability of every proper subfield $F \subset \mathbb{R}$.

Corollary 1 *No BSS-decidable set $S \subseteq \mathbb{R}^n$ is both dense and co-dense in $\mathbb{R}^n$.*

Proof. If the characteristic function χ_S were computed by some BSS machine M with parameters $\boldsymbol{z}$, then by Lemma 1, it would be constant in some neighborhood of every $\boldsymbol{y} \in \mathbb{R}^n$ algebraically independent over $\boldsymbol{z}$. □

Corollary 2 *Define the boundary of a subset $S \subseteq \mathbb{R}^n$ to be the intersection of the closure of S with the closure of its complement. If S is BSS-decidable, then there is a finite tuple $\boldsymbol{z}$ such that every point on the boundary of S has coordinates algebraically dependent over $\boldsymbol{z}$.* □

Of course, Corollaries 1 and 2 follow from other results that have been established long since, in particular from the Path Decomposition Theorem described in [1]. We include them here because of the simplicity of these proofs, and because they introduce the method to be used in the following section.

3 Application to Algebraic Numbers

Here we modify the method of Lemma 1 to answer Question 1.

Theorem 1 *For all $d > 0$, $\mathbb{A}_d \not\leq_{BSS} \mathbb{A}_{d-1}$.*

Proof. Suppose that M is an oracle BSS machine with real parameters $\boldsymbol{z}$, such that $M^{\mathbb{A}_{d-1}}$ computes the characteristic function of $\mathbb{A}_d$. Fix any $y \in \mathbb{R}$ which is transcendental over the field $Q = \mathbb{Q}(\boldsymbol{z})$, and run $M^{\mathbb{A}_{d-1}}$ on input y. As in the proof of Lemma 1, we set F to be the finite set of all nonconstant rational functions $f \in Q(Y)$ such that $f(y)$ appears in some cell during this computation. Again, there is an $\epsilon > 0$ such that all x within ϵ of y satisfy $f(x) \cdot f(y) > 0$ for all $f \in F$. However, when $M^{\mathbb{A}_{d-1}}$ runs on an arbitrary input $x \in B_\epsilon(y) \cap \mathbb{A}_d$, it may have a different computation path, because such an x might lie in $\mathbb{A}_{d-1}$, or might have $f(x) \in \mathbb{A}_{d-1}$ for some $f \in F$, and in this case the computation on input x might ask its oracle whether $f(x) \in \mathbb{A}_{d-1}$ and would then branch differently from the computation on input y. (Of course, for all $f \in F$, $f(y) \notin \mathbb{A}_{d-1}$, since $f(y)$ must be transcendental over $\mathbb{Q}$ for nonconstant f.) So we must establish the existence of some $x \in B_\epsilon(y) \cap \mathbb{A}_d$ with $f(x) \notin \mathbb{A}_{d-1}$ for all $f \in F$. Of course, we do not need to give any effective procedure which produces this x; its existence is sufficient.

We will need the following lemma from calculus. The lemma uses complex numbers, but only for mathematical results about $\mathbb{R}$; no complex number is ever an input to M.

Lemma 2. *If ζ is a primitive k-th root of unity and $f \in \mathbb{R}(Y)$ and there are positive real values of v arbitrarily close to 0 for which at least one of $f(b+\zeta v), f(b+\zeta^2 v), \ldots, f(b+\zeta^{k-1}v)$ has the same value as $f(b+v)$, then $f'(b) = 0$.* □

Fix ζ to be a primitive d-th root of unity. We choose $b \in \mathbb{Q}$ such that $|y - b| < \frac{\epsilon}{2}$ and such that b lies in the domain of every $f \in F$, with all $f'(b) \neq 0$. Such a b must exist, since all $f \in F$ are differentiable and nonconstant. Now Lemma 2 yields a $\delta > 0$, such that every $v \in \mathbb{R}$ with $0 < v < \delta$ satisfies $f(b+v) \neq f(b+\zeta^m v)$ for every $f \in F$ and every m with $0 < m < d$. So fix $x = b + \sqrt[d]{u}$ for

some $u \in \mathbb{Q}$ with $0 < \sqrt[d]{u} < \min(\delta, \frac{\epsilon}{2})$, for which $(X^d - u)$ is irreducible in $Q[X]$. (This ensures $\sqrt[d]{u} \notin Q$, of course. If there were no such u, then Q could not be finitely generated over $\mathbb{Q}$; this follows from the criterion for irreducibility of $(X^d - u)$ in [4, Thm. 9.1, p. 331], along with [6, Thm. 3.1.4, p. 82].) Thus $|x - y| < \epsilon$ and all $f \in F$ satisfy $f(b + \sqrt[d]{u}) \neq f(b + \zeta^m \sqrt[d]{u})$ for all $0 < m < d$.

Suppose that $f(x) = a \in \mathbb{A}_{d-1}$. Then $Q \subseteq Q(a) \subseteq Q(x)$, and a has degree $< d$ over Q (since $\mathbb{Q} \subseteq Q$), while $[Q(x) : Q] = d$, so $Q(a)$ is a proper subfield of $Q(x)$. Indeed $[Q(x) : Q(a)] \cdot [Q(a) : Q] = [Q(x) : Q] = d$, so the degree of a over Q is some proper divisor of d. Now let $p(X)$ be the minimal polynomial of x over the field $Q(a)$. Of course $p(X)$ may fail to lie in $\mathbb{Q}[X]$, but $p(X)$ must divide the minimal polynomial of x in $\mathbb{Q}[X]$, and so the roots of $p(X)$ are x and some of the $\mathbb{Q}$-conjugates $(b + \zeta^m \sqrt[d]{u})$ of x. At least one $(b + \zeta^m \sqrt[d]{u})$ with $0 < m < d$ must be a root of $p(X)$, since $\deg(p(X)) = [Q(x) : Q(a)] > 1$. We fix this m and let $\overline{x} = b + \zeta^m \sqrt[d]{u}$, and also fix $k = \deg(p(X))$.

Now we apply the division algorithm to write

$$f(X) = \frac{g(X)}{h(X)} = \frac{q_g(X) \cdot p(X) + r_g(X)}{q_h(X) \cdot p(X) + r_h(X)}$$

with $r_g(X)$ and $r_h(X)$ both in $Q(a)[X]$ of degree $< k$. We write $r_g(X) = g_{k-1}X^{k-1} + \cdots + g_1 X + g_0$ and $r_h(X) = h_{k-1}X^{k-1} + \cdots + h_1 X + h_0$, with all coefficients in $Q(a)$. Then $r_g(x) = g(x) = ah(x) = ar_h(x)$, since $p(x) = p(\overline{x}) = 0$. The equation $0 = r_g(x) - ar_h(x)$ can then be expanded in powers of $\sqrt[d]{u}$:

$$\begin{aligned}
0 &= \sum_{j<k} \left(g_j \cdot (b + \sqrt[d]{u})^j - ah_j \cdot (b + \sqrt[d]{u})^j \right) \\
&= \Big[(g_{k-1}b^{k-1} + g_{k-2}b^{k-2} + \cdots + g_1 b + g_0) \\
&\quad - a(h_{k-1}b^{k-1} + h_{k-2}b^{k-1} + \cdots + h_1 b + h_0) \Big] \\
&\quad + \sqrt[d]{u} \cdot \left[\left(\binom{k-1}{1} g_{k-1}b^{k-2} + \binom{k-2}{1} g_{k-2}b^{k-3} + \cdots + \binom{1}{1} g_1 b^0 \right) \right. \\
&\qquad \left. -a \left(\binom{k-1}{1} h_{k-1}b^{k-2} + \binom{k-2}{1} h_{k-2}b^{k-3} + \cdots + \binom{1}{1} h_1 b^0 \right) \right] \\
&\qquad\qquad \vdots \\
&\quad + (\sqrt[d]{u})^{k-2} \left[\left(\binom{k-1}{k-2} g_{k-1}b + g_{k-2} \right) - a \left(\binom{k-1}{k-2} h_{k-1}b + h_{k-2} \right) \right] \\
&\quad + (\sqrt[d]{u})^{k-1} \Big[g_{k-1} - ah_{k-1} \Big]
\end{aligned}$$

Here all bracketed expressions lie in $Q(a)$. However, $x = b + \sqrt[d]{u}$ has degree k over $Q(a)$, and therefore so does $\sqrt[d]{u}$. It follows that $\{1, \sqrt[d]{u}, (\sqrt[d]{u})^2, \ldots, (\sqrt[d]{u})^{k-1}\}$ forms a basis for $Q(x)$ as a vector space over $Q(a)$, and hence, in the equation above, all bracketed expressions must equal 0. One then proceeds inductively: the final bracket shows that $g_{k-1} = ah_{k-1}$, and plugging this into the second-to-last bracket shows that $g_{k-2} = ah_{k-2}$, and so on up. Thus $r_g(X) = ar_h(X)$, and so

$$f(x) = \frac{r_g(x)}{r_h(x)} = a = \frac{r_g(\overline{x})}{r_h(\overline{x})} = f(\overline{x}),$$

contradicting the choice of δ above. This contradiction shows that $f(x) \notin \mathbb{A}_{d-1}$, for every $f \in F$, and as in Lemma 1, it follows immediately that the computations by the machine M with oracle $\mathbb{A}_{d-1}$ on inputs x and y proceed along the same path and result in the same output. Since $x \in \mathbb{A}_d$ and $y \notin \mathbb{A}_d$, this proves the theorem. □

4 Further Results

We state here a few further results we have recently proven. For these we extend the notation: given any subset $S \subseteq \mathbb{N}$, write $\mathbb{A}_S = \cup_{d \in S} \mathbb{A}_{=d}$.

Theorem 2 *For sets $S, T \subseteq \mathbb{N}$, if $\mathbb{A}_S \leq_{BSS} \mathbb{A}_T$, then there exists $M \in \mathbb{N}$ such that all $p \in S$ satisfy $\{p, 2p, 3p, \ldots, Mp\} \cap T \neq \emptyset$. As a near-converse, if $(S - T)$ is finite and $(\forall p \in S - T)(\exists q > 0)[pq \in T]$, then $\mathbb{A}_S \leq_{BSS} \mathbb{A}_T$.*

Corollary 3 *There exists a subset $\mathcal{L}$ of the BSS-semidecidable degrees such that $(\mathcal{L}, \leq_{BSS}) \cong (\mathcal{P}(\mathbb{N}), \subseteq)$.*

Proof. We may replace the power set $\mathcal{P}(\mathbb{N})$ by the power set $\mathcal{P}(\{\text{primes}\})$. The latter maps into the BSS-semidecidable degrees via $S \mapsto \mathbb{A}_S$, and Theorem 2 shows this to be an embedding of partial orders. (The same map on all of $\mathcal{P}(\mathbb{N})$ is not an embedding.) In particular, if S and T are sets of primes and $n \in S - T$, then no multiple of n can lie in T; thus, by the theorem, $S \not\subseteq T$ implies $\mathbb{A}_S \not\leq_{BSS} \mathbb{A}_T$. The converse is immediate (for subsets of $\mathbb{N}$ in general, not just for prime numbers): if $S \subseteq T$, then ask whether an input x lies in the oracle set $\mathbb{A}_T$. If not, then $x \notin \mathbb{A}_S$; if so, find the minimal polynomial of x over $\mathbb{Q}$ and check whether its degree lies in S. (This program requires one parameter, to code the set S.) □

Theorem 3 *If $C \subseteq \mathbb{R}^\infty$ is a set to which the Halting Problem for BSS machines is BSS-reducible, then $|C| = 2^\omega$. Indeed, $\mathbb{R}$ has finite transcendence degree over the field K generated by (the coordinates of the tuples in) C.*

For the definition of the Halting Problem, see [1, pp. 79-81]. Since a program is allowed finitely many real parameters, it must be coded by a tuple of real numbers, not merely by a natural number. Theorem 3 is a specific case of a larger result on cardinalities, which is a rigorous version of the vague intuition that a set of small cardinality cannot contain enough information to compute a set of larger cardinality.

Definition 4 A set $S \subseteq \mathbb{R}$ is *locally of bicardinality $\leq \kappa$* if there exist two open subsets U and V of $\mathbb{R}$ with $|\mathbb{R} - (U \cup V)| \leq \kappa$ and and $|U \cap S| \leq \kappa$ and $|V \cap \overline{S}| \leq \kappa$. (Here $\overline{S} = \mathbb{R} - S$.)

This definition roughly says that up to sets of size κ, each of S and $\overline{S}$ is equal to an open subset of $\mathbb{R}$. For example, the BSS-computable set $S = \{x \in \mathbb{R} : (\exists m \in \mathbb{N})\ 2^{-(2m+1)} \leq x \leq 2^{-(2m)}\}$, containing those x which have a binary expansion beginning with an even number of zeroes, is locally of bicardinality ω. The property of local bicardinality $\leq \kappa$ does not appear to us to be equivalent to any more easily stated property, but it is exactly the condition needed in our general theorem on cardinalities.

Theorem 5 *If $C \subseteq \mathbb{R}^\infty$ is an oracle set of infinite cardinality $\kappa < 2^\omega$, and $S \subseteq \mathbb{R}$ is a set with $S \leq_{BSS} C$, then S must be locally of bicardinality $\leq \kappa$. The same holds for oracles C of infinite co-cardinality $\kappa < 2^\omega$.*

References

1. L. Blum, F. Cucker, M. Shub, and S. Smale; *Complexity and real computation* (Berlin: Springer-Verlag, 1997).
2. L. Blum, M. Shub, and S. Smale; On a theory of computation and complexity over the real numbers, *Bulletin of the A.M.S. (New Series)* **21** (1989), 1–46.
3. C. Gassner; A hierarchy below the halting problem for additive machines, *Theory of Computing Systems* **43** (2008) 3–4, 464–470.
4. S. Lang; *Algebra* (second edition) (Menlo Park, CA: Addison-wesley Publishing Co., Inc., 1984).
5. K. Meer & M. Ziegler; An explicit solution to Post's Problem over the reals, *Journal of Complexity* **24** (2008) 3–15.
6. M. Nagata; *Theory of Commutative Fields*, English trans. (American Mathematical Society, 1993).
7. Y. Yonezawa; The Turing degrees for some computation model with the real parameter, *J. Math. Soc. Japan* **60** 2 (2008), 311-324.

Representation Theorems for Analytic Machines*

Tobias Gärtner

Universität des Saarlandes, Saarbrücken

Abstract. The well-known representation theorem, or path decomposition theorem, for Blum-Shub-Smale machines (BSS machines) characterizes the functions computable by those machines. The domain of a BSS computable function decomposes into countably many semi-algebraic sets, and on each of those sets the function is a polynomial, or a rational function if division is allowed. The model of analytic machines is an extension of the BSS model, admitting infinite converging computations. In this work, we ask the natural question: To which extent does the representation theorem generalize to analytic machines, and are functions computable by analytic machines representable by power series, the generalization of polynomials? We show that over the real numbers, there is no such representation theorem. On the other hand, we show that over the complex numbers, functions computable by machines with 'not to many branching operations' are indeed representable by power series on parts of their domain.

1 Introduction

There are several different approaches for defining computable functions on the real and complex numbers. The two most prominent of those can roughly be divided into the analytic and the algebraic approach. The analytic approach, *recursive analysis*, extends the classic Turing machine by infinite converging computations. The machine model used here is the Type 2 Turing machine, see e.g. [5]. In the algebraic approach, register machine models are used that deal with real numbers as atoms and are able to perform exact arithmetic and comparisons. The main machine model in this approach is the *BSS machine* [1]. A computation of a BSS machine is a finite sequence of arithmetic operations and conditional branches.

The model of *analytic machines*, see e.g. [2], which is the model this work is based on, is an extension of the BSS model and can be regarded as a kind of synthesis of the analytic and the algebraic approach, since it extends BSS machines by infinite converging computations. For a comparison and classification of many different approaches see [6].

The functions computable by BSS machines are characterized by the *representation theorem for BSS computable functions* (often called *path decomposition theorem*, [1]): The domain of a BSS computable function decomposes into countably many semi-algebraic sets, and on each of these sets the function is a polynomial or rational function. Considering that a computation of a BSS machine is a sequence of arithmetic operations and branches, this theorem can be appreciated quite intuitively. This theorem is of central importance for the computability and complexity theory of BSS machines.

Since BSS computable functions are representable by polynomials (or rational functions) on parts of their domain and since analytic machines generalize the finite computations of BSS machines to infinite computations, the question arises to which extent functions computable by analytic machines are representable by the generalization of polynomials, namely power series on parts of their domains. This is the topic of this work.

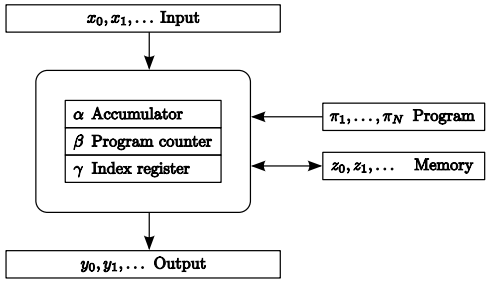
It turns out that it is important to distinguish between functions defined over the real numbers on the one hand and the complex numbers on the other. Over the real numbers, we show that the representation theorem does not generalize to analytic machines. On the other hand, over the complex numbers, we show that a function that is computable by an analytic machine and which does not branch infinitely often is always representable by a power series on a part of its domain.

* This work has been included in a paper that has been submitted for publication in the TOCS CiE 09 special issue (together with Günter Hotz).

2 Analytic Machines

The machine model underlying this work is a register machine model, which is an extension of the model of BSS machines. For finite computations, the models are equivalent. Analytic machines extend BSS machines by infinite converging computations.
We briefly present the model of analytic machines but assume familiarity with the BSS model. For more precise and formal definitions and also for properties of analytic machines, cf. [2] and [3].

The model is a register machine model defined over the base field of the real or complex numbers $\mathbb{R}$ or $\mathbb{C}$, respectively. A machine consists of a control unit and a finite program with instructions from an instruction set consisting of assignments, arithmetics, and conditional branches. Registers, memory, input and output contain real or complex numbers, which are regarded as atoms. It is important to note that a machine can perform *exact arithmetic* and comparisons on real or complex numbers.



A computation of a machine is a sequence of configurations of such a machine. We distinguish finite and infinite converging computations. Infinite computations are only regarded as valid if an output is written infinitely often. If the sequence of outputs of an infinite computation is convergent, we call such a computation *analytic*. We call the n-th output of a machine $\mathcal{M}$ with input x the *n-th approximation* of the computation of $\mathcal{M}$ and write $\mathcal{M}^{(n)}(x)$. Therefore, a computation is analytic iff $\lim_{n\to\infty} \mathcal{M}^{(n)}(x)$ exists.
We call a function f (finitely) $\mathbb{R}$-computable ($\mathbb{C}$-computable) if there is a $\mathbb{R}$-machine ($\mathbb{C}$-machine) that computes f with finite computations for each input. The $\mathbb{R}$-computable functions are exactly the BSS computable functions. We call a function f *analytically* $\mathbb{R}$-computable ($\mathbb{C}$-computable) if there is an $\mathbb{R}$-machine ($\mathbb{C}$-machine) that computes f with finite or analytic computations.
An important notion we use throughout the text is the notion of a *computation path* σ: The computations of a machine can be represented in a computation tree in a natural manner (see e.g. [1]), where a node is labeled by the instruction that is performed at the corresponding step of the computation. For finite machines, a computation tree has only paths of finite length (but possibly countably many). For a computation path σ, we denote by D_σ its *domain*, i.e., the set of all inputs that result in the computation path σ.

3 Representation Theorems

We now address the main question of this work: Are there generalizations of Blum, Shub and Smale's representation theorem for analytic machines? We begin with the following observation: An analytic machine can have infinitely many branching operations on each computation path, and therefore it is possible that each input has its own computation path. A decomposition in the original sense is hence impossible, and since the notion of power series only makes sense on sets with nonempty open interior, in this case there cannot be a representation by power series. Therefore, we restrict our consideration to computation paths σ whose domain D_σ has nonempty open interior, or in the simplest case, functions which are computable with a single computation path.

3.1 Representation over $\mathbb{R}$

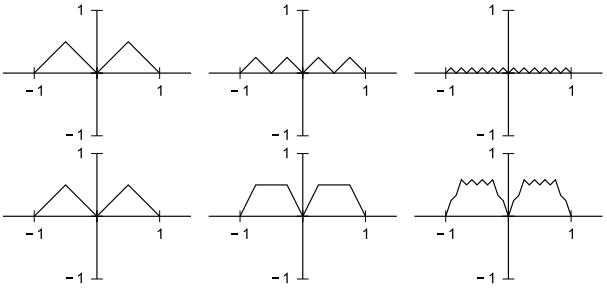
In this paragraph we will show that over the real numbers, the representation theorem for finite $\mathbb{R}$-machines does not generalize to analytic machines. More precisely, we will show that there are functions over $\mathbb{R}$ which are nowhere representable by power series and which are yet computable by an analytic machine with a single computation path.

Proposition 1. *Suppose $f : [0,1] \to \mathbb{R}$ is a continuous function with the property $f(\mathbb{Q} \cap [0,1]) \subseteq \mathbb{Q}$, i.e. f maps rationals on rationals. Then f is analytically $\mathbb{R}$-computable with the use of general constants such that all inputs $x \in [0,1]$ have the same computation path.*

Proof. If f is a function with the required property, then by the Weierstrass approximation theorem, f can be uniformly approximated on $[0,1]$ by the *Bernstein polynomials* $B_{n,f}(x) := \sum_{m=0}^{n} f\left(\frac{m}{n}\right) \binom{n}{m} x^m (1 -$

$x)^{n-m}$. Given the values $f\left(\frac{m}{n}\right)$ of f on the rationals, arbitrary approximations of f can be computed without branching operations *that depend on the input*, since the sequence $\left(f\left(\frac{m}{n}\right)\right)_{m\leq n}$ can be coded in a single real constant c_f. The extraction of $f\left(\frac{m}{n}\right)$ from the coded constant can be done without branching operations depending on the input x.

Example 1. We now give an example for a function with the properties of Prop. 3.1 which is nowhere representable by a power series. Moreover, we can show that this function can be computed with a single computation path even *without* the use of irrational constants.



Let g be the periodic continuation of the function $x \mapsto \begin{cases} x: & 0 \leq x \leq \frac{1}{2} \\ 1-x: & \frac{1}{2} \leq x \leq 1 \end{cases}$ and define G by $G(x) := \sum_{k=1}^{\infty} \frac{1}{k!} g(k!x)$. In the figure, the first three steps of the formation of G are shown, summands above and sums below. G is a continuous but nowhere differentiable function. Hence G is nowhere representable by a power series. The sum in the definition of G is *finite* for all rational inputs $\frac{p}{q}$. Therefore, the function G is *(finitely)* $\mathbb{R}$-*computable* on the rationals, and the encoding of the rational values in a real constant c_G is not necessary. Combined with Prop. 3.1, we obtain that G is an example of a function that is computable by an analytic machine with a single computation path, and which is nowhere representable by a power series.

3.2 Representation over $\mathbb{C}$

Over the complex numbers, regularity of functions is often much easier obtained than over the real numbers. For example, it is well-known that over the reals, a function that is differentiable has a continuous derivative. In contrast, a function that is complex differentiable is already differentiable infinitely often and even analytic. Therefore, the question arises whether in the setting of analytic machines, there are also stronger regularity properties of functions computed by machines over $\mathbb{C}$. As before, we restrict our consideration to computation paths σ that have domains D_σ with nonempty open interior.
The following theorem shows that over the complex numbers, machines with such open domains of computation paths indeed admit a representation by power series on a subset of these domains.

Theorem 1. *Suppose $f : D \to \mathbb{C}$ is analytically computable by a $\mathbb{C}$-analytic machine $\mathcal{M}$. Suppose further that there is a computation path σ of $\mathcal{M}$, such that its domain D_σ has nonempty interior. Then there is an open and dense subset $D' \subset D_\sigma$ such that f is an analytic function on D'.*

Proof. Consider the sequence of n-th approximations $f_n(z) := \mathcal{M}^{(n)}(z)$ of the machine for inputs $z \in D_\sigma$ for a computation path as in the theorem. By the representation theorem for BSS machines, each f_n is a rational function, and, therefore, a holomorphic function on the open interior of D_σ. The sequence (f_n) converges pointwise on D_σ to the function f. Osgood's theorem [4], a classical theorem from complex analysis, can be directly applied in this context. It implies that there is an open and dense subset $D' \subset D_\sigma$ such that the convergence of (f_n) is uniform on compact subsets of D'. From this it follows immediately that the limit f is an analytic function on D'.

The existence of a computation path that has a domain with nonempty open interior is, for example, fulfilled if each computation path of a machine $\mathcal{M}$ has only finitely many branching operations. This follows with Baire's category theorem, since $\mathcal{M}$ has only countably many computation paths. Therefore we have:

Corollary 1. *Let $f : D \to \mathbb{C}$ be computable by a $\mathbb{C}$-analytic machine such that each computation path of $\mathcal{M}$ has only finitely many branching operations. Then there is a nonempty open set on which f is representable by a power series.*

References

1. Leonore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and Real Computation.* Springer, New York, 1998.

2. Thomas Chadzelek and Günter Hotz. Analytic machines. *TCS*, 219:151–167, 1999.
3. Tobias Gärtner and Günter Hotz. Computability of analytic functions with analytic machines. In *Proc. CiE 2009*, *LNCS* 5635, 250–259. Springer-Verlag, 2009.
4. William F. Osgood. Note on the functions defined by infinite series whose terms are analytic functions of a complex variable. *Ann. of Math.*, 2. Ser. 3:25–34, 1902.
5. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.
6. Martin Ziegler. Real computability and hypercomputation. TR C-07013, KIAS, 2007.

Computability over Positive Predicate Structures ⋆

Margarita Korovina[1] and Oleg Kudinov[2]

[1] The University of Manchester and IIS SB RAS Novosibirsk,
Margarita.Korovina@manchester.ac.uk,
[2] Sobolev Institute of Mathematics, Novosibirsk
kud@math.nsc.ru

The main goal of the research presented in this paper is to provide a logical framework for studying computability over discrete and continuous data in a common language. Since discrete and continuous structures are different by nature, formalisation of computability over such structures in a common language is a challenging research problem. In order to archive this goal we represent data as a structure which could not have effective equality and employ Σ–definability theory. Our approach is based on representations of data (discrete or continuous) by a suitable structure $\mathcal{A} = \langle A, \sigma_P, \neq \rangle$, where A contains more than one element, and σ_P is a set of basic predicates. *We assume that all predicates $Q_i \in \sigma_P$ and $\neq$ occur only positively in Σ-formulas and do not assume that the language σ_P contains equality.* We call such structures as *positive predicate structures*. As examples we can consider

- (The natural numbers) $\mathbb{N} = \langle \mathbb{N}, 0, s, < \rangle$;
- (The real numbers) $\mathbb{R} = \langle \mathbb{R}, 0, 1, +, \cdot, < \rangle$;
- (The complex numbers) $\mathbb{C} = \langle \mathbb{C}, 0, 1, +, \cdot, \neq \rangle$.
- (The real-valued function defined on compact intervals)
  $C[0,1] = \langle C[0,1], P_1, \ldots, P_{10}, \neq \rangle$ where the predicates $P_1, \ldots, P_{10}$ have the following meanings for every $f,\ g \in C[0,1]$:
  The first group formalises relations between infimum and supremum of two functions.

$$C[0,1] \models P_1(f,g) \leftrightarrow \sup f < \sup g;$$
$$C[0,1] \models P_2(f,g) \leftrightarrow \sup f < \inf g;$$
$$C[0,1] \models P_3(f,g) \leftrightarrow \sup f > \inf g;$$
$$C[0,1] \models P_4(f,g) \leftrightarrow \inf f > \inf g.$$

  The second group formalises properties of operations on $C[0,1]$.

$$C[0,1] \models P_5(f,g,h) \leftrightarrow f(x) + g(x) < h(x);\ \text{for every } x \in [0,1];$$
$$C[0,1] \models P_6(f,g,h) \leftrightarrow f(x) \cdot g(x) < h(x) \text{ for every } x \in [0,1];$$
$$C[0,1] \models P_7(f,g,h) \leftrightarrow f(x) + g(x) > h(x) \text{ for every } x \in [0,1];$$
$$C[0,1] \models P_8(f,g,h) \leftrightarrow f(x) \cdot g(x) > h(x) \text{ for every } x \in [0,1].$$

  The third group formalises relations between functions f and the identity function $\lambda x.x$.

$$C[0,1] \models P_9(f) \ \leftrightarrow f > \lambda x.x;$$
$$C[0,1] \models P_{10}(f) \leftrightarrow f < \lambda x.x.$$

In order to logically characterise computability over positive predicate structures we employ Σ-definability where elements of a structure and computational processes involving these elements can be defined using finite formulas. Definability is a very successful framework for generalised computability theory, descriptive complexity, set-theoretic specifications, and databases. One of the most interesting and practically important types of definability is Σ-definability, which generalises recursive enumerability over the natural numbers [1, 4, 5]. However, the most developed part of definability and Σ-definability theories deal with abstract structures with equality (i.e., the natural numbers, trees, automata, etc.) which are not appropriate for representation of continuous data.

It turns out that Σ-definability without equality is rather different from Σ-definability with equality. It has been shown in [12] that there is no effective procedure which given a Σ-formula with equality defining an open set produces a Σ-formula without equality defining the same set. Therefor it is important

⋆ This research was partially supported by EPSRC grant EP/E050441/1, DFG-RFBR (grant No 436 RUS 113/1002/01, grant No 09-01-91334), RFBR grant 08-01-00336.

to figure out which properties of Σ-definability hold on structures with equality likewise on structures without equality. We prove the following properties of Σ–definability which are necessary to make effective reasoning about computability in logical terms.

Theorem 1. *For any positive predicate structure $\mathcal{A} = \langle A, \sigma_P, \neq \rangle$ the following properties holds*

1. *The least fixed point of an effective operator is Σ–definable.*
2. *There exists a universal Σ-predicate, i.e., for every $n \in \omega$ there exists a Σ-formula $Univ_{n+1}(m, x_0, \ldots, x_n)$ such that for any Σ-formula $\Phi(x_0, \ldots, x_n)$*

$$\mathbf{HF}(A) \models \Phi(r_0, \ldots, r_n) \leftrightarrow Univ_{n+1}(\ulcorner \Phi \urcorner, r_0, \ldots, r_n).$$

3. *A set $B \subset A^n$ is Σ-definable if and only if it is definable by a disjunction of a recursively enumerable set of existential formulas.*

We also show links between positive predicate structures and topological spaces. for an positive predicate structure $\mathcal{A}$, we introduce a topology, called τ_Σ^A, with a base consisting of the subsets of the carrier set defined by existential formulas. In this topology *Σ-definability coincides with effective openness*. On several examples we illustrate how to pick an appropriate finite language in such way that the τ_Σ^A-topology coincides with the usual topology.

In order to introduce a reasonable notion of computability on positive predicate structures including continuous ones we have to take into account the following necessary conditions. One of them is logical which says that $Th_\exists(\mathcal{A})$ should be computably enumerable. This condition provides tools for effective reasoning about computable continuous data based on Σ-definability. Another one is topological which says that computable functions should be continuous. This condition provides correct approximating computation of continuous data.

In order to address these conditions we introduce and investigate *effectively enumerable and strongly effectively enumerable topological spaces and computability* over them [7]. It is worth noting that the notion of effectively enumerable topological spaces is a generalisation of the notion of computable topological spaces introduced in [6]. We show the following proposition.

Theorem 2. *For every positive predicate structure $\mathcal{A}$ the following properties hold.*

1. *The topological space $\left(A, \tau_\Sigma^{\mathcal{A}}\right)$ is effectively enumerable if and only if $Th_\exists(A)$ is computable enumerable.*
2. *If $Th_\exists(A)$ is decidable then $\left(A, \tau_\Sigma^{\mathcal{A}}\right)$ is strongly effectively enumerable.*

This properties allow us to study computability over positive predicate structures.

Theorem 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be structures with computably enumerable existential theories and $\left(B, \tau_\Sigma^{\mathcal{B}}\right)$ be T_0-space. Then for every total function $F : A \to B$ the following are equivalent.*

1. *F is computable;*
2. *There exists a computable function $h : \omega \to \omega$ such that for all $j \in \omega$ the F-preimage of the set definable by a Σ-formula with the Gödel number j coincides with the set definable by a Σ-formula with the Gödel number $h(j)$.*

As a corollary we get that for any positive predicate structure with computably enumerable existential theory *computability coincide with effective continuity*. Now we make comparative analysis of computability and majorant-computability over positive predicate structures which are effectively enumerable topological spaces. Assume $\mathcal{A} = \langle A, \sigma_P, \neq \rangle$ is a structure with finite σ_P and $\mathbb{R} = \langle \mathbb{R}, 0, 1, +, \cdot, < \rangle$. We use the criteria of majorant-computability of a function $F : A \to \mathbb{R}$ (c.f. [9]) which we recall below.

We extend the structure $\mathcal{R} = \mathbb{R} \cup A$ by the set of hereditarily finite sets $\mathrm{HF}(\mathcal{R})$ and consider Σ-definability in $\mathrm{HF}(\mathcal{R})$.

Below we will write $\varphi_1(a, \cdot) < \varphi_2(a, \cdot)$ if $\mathbf{HF}(\mathcal{R}) \models \varphi_1(a, y) \wedge \varphi_2(a, z) \to y < z$ for all real numbers y, z. The following theorem connects a majorant-computable function with validity of finite formulas in the set of hereditarily finite sets, $\mathbf{HF}(\mathcal{R})$.

Theorem 4. *[9] For every function $F : A \to \mathbb{R}$ the following assertions are equivalent:.*

1. *The functional F is majorant-computable.*

2. *There exists Σ–formulas $\varphi_1(a,y)$, $\varphi_2(a,y)$ such that $\varphi_1(a,\cdot) < \varphi_2(a,\cdot)$ and*

$$F(a) = y \leftrightarrow \forall z_1 \forall z_2 \left(\varphi_1(a,z_1) < y < \varphi_2(a,z_2)\right) \wedge$$
$$\{z \mid \varphi_1(a,z)\} \cup \{z \mid \varphi_2(a,z)\} = \mathbb{IR} \setminus \{y\}.$$

Theorem 5. *Let $\mathcal{A}$ be a structure such that $\left(A, \tau_\Sigma^{\mathcal{A}}\right)$ is effectively enumerable. A function $F : A \to \mathbb{IR}$ is majorant-computable if and only if there is a computable total function $G : A \to \mathcal{I}_{\mathbb{IR}}$, where $\mathcal{I}_{\mathbb{IR}}$ is the interval domain, such that*

1. *If $x \in \mathrm{dom}(F)$ then $G(x) = \{F(x)\}$.*
2. *If $x \notin \mathrm{dom}(F)$ then $G(x) \neq \{y\}$ for all $y \in \mathbb{IR}$.*

In [8] we proved the Uniformity principle for Σ-definability over the real numbers. We employed the Uniformity principle to show that quantifiers bounded by computable compact sets, rational numbers, polynomials, and computable functions as well can be used in Σ-formulas without enlarging the class of Σ-definable sets. It will be interesting to find requirements on a positive predicate structure under which the Uniformity principle holds.

References

1. J. Barwise. *Admissible sets and Structures.* Springer Verlag, Berlin, 1975.
2. Vasco Brattka, Gero Presser, *Computability on subsets of metric spaces.*, Theor. Comput. Sci. 305(1-3): 43-76 (2003)
3. V. Brattka and K. Weihrauch, Computability on subsets of euclidean space I: Closed and compact sets. *TCS*, 219:65–93, 1999.
4. Yu. L. Ershov, *Definability and computability.* Plenum, New-York, 1996.
5. W. Hodges, *The meaning of specifications : Set-theoretical specification*, Semantics of Programming Languages and Model Theory, Algebra, Logic and Applications Series, v. 5, pp 43–68, 1993.
6. Tanja Grübba, Mattias Schröder and Klaus Weihrauch, *Computable metrization*, Mathematical Logic Quaterly, **53(4-5)**, 2007, 595-604.
7. Margarita Korovina, Oleg Kudinov, Comparative Analysis of Some Models of Computation over Effectively Enumerable Topological Spaces, In Proceedings CCA'08, Computability and Complexity in Computable Analysis, FernUniversitat Hagen, Bericht Nr. 346-8, pages 129-141, 2008.
8. Margarita Korovina and Oleg Kudinov. The Uniformity Principle for Σ-definability with Applications to Computable Analysis. In S.B. Cooper, B. Löwe, and A. Sorbi, editors, *CiE'07, Lecture Notes in Computer Science vol. 4497*, pages 416–425, Springer, 2007.
9. Margarita V. Korovina and Oleg V. Kudinov. Towards computability of higher type continuous data. In S. Barry Cooper, Benedikt Löwe, and Leen Torenvliet, editors, *CiE*, volume 3526 of *Lecture Notes in Computer Science*, pages 235–241. Springer, 2005.
10. Margarita V. Korovina. Computational aspects of sigma-definability over the real numbers without the equality test. In Matthias Baaz and Johann A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 330–344. Springer, 2003.
11. Margarita V. Korovina. Gandy's theorem for abstract structures without the equality test. In Moshe Y. Vardi and Andrei Voronkov, editors, *LPAR*, volume 2850 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2003.
12. Andrei Morozov and Margarita Korovina, Remarks on Σ-definability without the equality test over the Reals. Electronic Notes in Theoretical Computer Science, N 202, Elsevier, 2008.
13. Y. N. Moschovakis, Abstract first order computability I, II. *Transactions of the American Mathematical Society*, 138:427–504, 1969.

Undecidability in Weihrauch Degrees

Oleg V. Kudinov[1⋆], Victor L. Selivanov[2⋆⋆] and Anton V. Zhukov[3⋆⋆⋆]

[1] S.L. Sobolev Institute of Mathematics, 4 Acad. Koptyug avenue, 630090 Novosibirsk, Russia
[2] A.P. Ershov Institute of Informatics Systems, 6 Acad. Lavrentjev pr., 630090 Novosibirsk, Russia
[3] Novosibirsk State Pedagogical University, 28 Vilyuiskaya ul., 630126 Novosibirsk, Russia

Abstract. We prove that the 1-quasiorder and the 2-quasiorder of finite k-labeled forests and trees have hereditarily undecidable first-order theories for $k \geq 3$. Together with an earlier result of P. Hertling, this implies some undecidability results for Weihrauch degrees.
Keywords. Weihrauch reducibility, labeled forest, 1-quasiorder, 2-quasiorder, undecidability, theory.

1 Introduction and preliminaries

As is well-known, different notions of hierarchies and reducibilities serve as useful tools for understanding the complexity (or non-computability) of decision problems on discrete structures. In computable analysis, many problems of interest turn out to be non-computable (even non-continuous), so there is a need for tools to measure their non-computability. Accordingly, also in this context of decision problems on continuous structures, people employed several hierarchies and reducibilities closely related to descriptive set theory.

In [10, 11] K. Weihrauch introduced some notions of reducibility for functions on topological spaces which turned out useful for understanding the non-computability and non-continuity of interesting decision problems in computable analysis [7, 6, 2] and constructive mathematics [1]. In particular, the following three notions of reducibilities between functions $f, g : X \to Y$ on topological spaces were introduced: $f \leq_0 g$ (resp. $f \leq_1 g$, $f \leq_2 g$) iff $f = g \circ H$ for some continuous function $H : X \to X$ (resp. $f = F \circ g \circ H$ for some continuous functions $H : X \to X$ and $F : Y \to Y$, $f(x) = F(x, gH(x))$ for some continuous functions $H : X \to X$ and $F : X \times Y \to Y$).

The notions are nontrivial even for the case of discrete spaces $Y = k = \{0, \ldots, k-1\}$ with $k < \omega$ points (we call functions $f : X \to k$ *k-partitions of X*). E.g., for $k = 2$ the relation $\leq_0$ coincides with the classical Wadge reducibility.

In [5, 6] (see also [8]) P. Hertling gave a useful "combinatorial" characterisation of important initial segments of the degree structures under Weihrauch reducibilities on k-partitions of the Baire space (note that the Baire space is important because it is commonly used in computable analysis for representing many other spaces of interest). From this characterisation and results in [9] it follows that for any $k \geq 3$ the first-order theory of this segment for $\leq_0$ is undecidable (and it is even computably isomorphic to the first-order arithmetic). The main result of this work states similar undecidability properties for the other two reducibilities.

We use some standard notation and terminology on posets which may be found e.g. in [3]. Sometimes we apply notions concerning posets also to quasiorders (known also as preorders); in such cases we mean the corresponding quotient-poset of the preordered set. Throughout this paper, k denotes an arbitrary integer, $k \geq 2$, which is identified with the set $\{0, \ldots, k-1\}$. By a *forest* we mean a finite poset in which every lower cone $\hat{x}$ is a chain. A *tree* is a forest that has a least element (called the *root* of the tree).

A *k-labeled poset* (or just a k-poset) is an object $(P; \leq, c)$ consisting of a poset $(P; \leq)$ and a labeling $c : P \to k$. A k-poset P can be seen as a node-weighted directed graph where every $x \in P$ carries the label $c(x)$. We call a k-poset $(P; \leq, c)$ *repetition-free* iff $c(x) \neq c(y)$ whenever x is an immediate predecessor of y in P. We usually simplify the notation of a k-poset to (P, c) or even P. We are mainly interested in the set $\mathcal{F}_k$ of finite k-labeled forests, but also in the set $\mathcal{T}_k$ of finite k-labeled trees. For any k-forest $F \in \mathcal{F}_k$ and $i < k$, let $p_i(F)$ denote a k-tree obtained from F by adding a new smallest element with the label i.

⋆ Supported by DFG-RFBR (Grant 436 RUS 113/1002/01, 09-01-91334), and by RFBR Grant 07-01-00543a.
⋆⋆ Supported by DFG-RFBR (Grant 436 RUS 113/1002/01, 09-01-91334) and by RFBR Grant 07-01-00543a.
⋆⋆⋆ Supported by RFBR Grant 07-01-00543a.

A *homomorphism* (or 0-morphism) $f : (P; \leq_P, c_P) \to (Q; \leq_Q, c_Q)$ between k-posets is a monotone function $f : (P; \leq_P) \to (Q; \leq_Q)$ respecting the labelings, i.e. satisfying $c_P = c_Q \circ f$.

A 1-*morphism* $f : (P; \leq_P, c_P) \to (Q; \leq_Q, c_Q)$ between k-posets is a monotone function $f : (P; \leq_P) \to (Q; \leq_Q)$ for which there exists a mapping $g : k \to k$ such that $c_P = g \circ c_Q \circ f$.

A 2-*morphism* $f : (P; \leq_P, c_P) \to (Q; \leq_Q, c_Q)$ between k-posets is a monotone function $f : (P; \leq_P) \to (Q; \leq_Q)$ which maps comparable elements (nodes) with different labels to elements with different labels, i.e.

$$\forall x, y \in P((x \leq_P y \wedge c_P(x) \neq c_P(y)) \to c_Q(f(x)) \neq c_Q(f(y))).$$

We say that a k-poset P is 0-*morphic* (resp. 1-*morphic*, 2-*morphic*) to a k-poset Q, denoted $P \leq_0 Q$ (respectively $P \leq_1 Q$, $P \leq_2 Q$), iff there exists a 0-morphism (resp. 1-morphism, 2-morphism) $f : P \to Q$. It is easy to prove that any 0-morphism is a 1-morphism and any 1-morphism is a 2-morphism in turn. Therefore $\leq_0$ implies $\leq_1$ and $\leq_1$ implies $\leq_2$.

Let $\equiv_0$ (0-equivalence, h-equivalence or just equivalence), $\equiv_1$ (1-equivalence) and $\equiv_2$ (2-equivalence) denote the equivalence relations induced by the quasiorders $\leq_0$, $\leq_1$ and $\leq_2$, respectively. The quotient set of $\mathcal{F}_k$ ($\mathcal{T}_k$) under $\equiv_i$, $i \leq 2$, is denoted by $\mathfrak{F}_k^i$ (respectively $\mathfrak{T}_k^i$). We use the same symbols $\leq_0$, $\leq_1$ and $\leq_2$ to denote the partial orders induced by the corresponding quasiorders on the corresponding quotient sets.

2 Properties of the preorders

Here we will observe only properties used further in this paper.

Lemma 1. *Let $A \leq_2 B$ for $A, B \in \mathcal{T}_k$. Then there exists a 2-morphism from A to B which maps the root of A to the root of B.*

Proof: induction on the height of A.

It is a simple and well-known fact that $(\mathfrak{F}_k^0, \leq_0)$ is an upper semilattice. For $a, b \in \mathfrak{F}_k$, their supremum $a \sqcup b$ is the 0-equivalence class of the join (i.e. disjoint union) $A \sqcup B$ of any k-forests $A \in a$ and $B \in b$.

Proposition 1. *The structures $(\mathfrak{F}_k^2, \leq_2)$ and $(\mathfrak{T}_k^2, \leq_2)$ are upper semilattices.*

Proof. It is easy to verify that the supremums in $(\mathfrak{F}_k^2, \leq_2)$ are defined in the same way as for $(\mathfrak{F}_k^0, \leq_0)$, so $(\mathfrak{F}_k^2, \leq_2)$ is an upper semilattice. Now let $a, b \in \mathfrak{T}_k^2$. Consider arbitrary disjoint k-trees $A \in a$ and $B \in b$. We can assume w.l.o.g. that the roots of A and B are labeled with 0. Let C be the k-tree that is obtained from the union of A and B by identifying their roots, $C \equiv_0 p_0(A \sqcup B)$. By the previous lemma, it is straightforward to prove that $[C]_{\equiv_2}$ is the supremum of a and b in $(\mathfrak{T}_k^2, \leq_2)$.

As for $(\mathfrak{F}_k^1, \leq_1)$ and $(\mathfrak{T}_k^1, \leq_1)$, we show in the next proposition that these structures are not semilattices. Let $(P, \leq)$ be an arbitrary poset or preordered set and $a, b \in P$. As usual, $c \in P$ is called a *minimal upper bound* of a and b iff

$$a \leq c \wedge b \leq c \wedge \forall x((a \leq x \wedge b \leq x \wedge x \leq c) \to c \leq x),$$

we call c also a *quasijoin* of a and b in P.

For any permutation $\alpha \in S_k$ and any k-forest $F = (F; \leq_F, c_F) \in \mathcal{F}_k$, let $\alpha(F)$ denote the k-forest $(F; \leq_F, \alpha \circ c_F)$ (i.e. all labels in F are replaced by their images via α).

Proposition 2. *(i) For any k-forests $F, G \in \mathcal{F}_k$, a k-forest H is a minimal upper bound of F and G in $(\mathcal{F}_k, \leq_1)$ iff $H \equiv_1 \alpha(F) \sqcup \beta(G)$ for some $\alpha, \beta \in S_k$.*

(ii) For any k-trees $U, V \in \mathcal{T}_k$, a k-tree W is a minimal upper bound of U and V in $(\mathcal{T}_k, \leq_1)$ iff $W \equiv_1 p_0(\alpha(U) \sqcup \beta(V))$ for some $\alpha, \beta \in S_k$.

Proof is a simple routine omitted due to the restrictions on the size of the paper.

We will use the same symbol "$\sqcup$" for the supremums in all upper semilattices under consideration. For an arbitrary semilattice $(S; \sqcup, \leq)$ where a partial order $\leq$ corresponds to $\sqcup$, $x \in S$ is called *join-irreducible* iff

$$\forall y \forall z(x \leq y \sqcup z \to (x \leq y \vee x \leq z)).$$

For an arbitrary poset $(P;\leq)$ and $x, y \in P$, let $mub_P(x,y)$ denote the set of all minimal upper bounds of x and y in P. We call $x \in P$ *quasijoin-irreducible* iff

$$\forall y \forall z \forall t((t \in mub_P(y,z) \wedge x \leq t) \rightarrow (x \leq y \vee x \leq z)).$$

Since the relation $t \in mub_P(y,z)$ can be defined by a formula of signature $\{\leq\}$, there exists a formula $ir(x)$ of signature $\{\leq\}$ which defines (in every poset) exactly the non-zero quasijoin-irreducible elements. If P is an upper semilattice then $mub_P(x,y) = \{x \sqcup y\}$ for all $x, y \in P$, so quasijoin-irreducible elements are exactly the join-irreducible elements. Therefore $ir(x)$ defines exactly the non-zero join-irreducible elements in any upper semilattice. Let $ir(P)$ denote the set of all non-zero quasijoin-irreducible elements of P.

Proposition 3. *(i)* $ir(\mathfrak{F}_k^1) = \mathfrak{T}_k^1$ *and* $ir(\mathfrak{F}_k^2) = \mathfrak{T}_k^2$.

(ii) $t \in ir(\mathfrak{T}_k^1)$ *iff* t *includes a repetition-free* k*-tree whose root has exactly one (immediate) successor. The same holds for* $t \in ir(\mathfrak{T}_k^2)$

Proof is a simple routine omitted due to the restrictions on the size of the paper.

We will need also the following technical lemma which shows that the structures $(\mathfrak{T}_k^1, \leq_1)$ and $(\mathfrak{T}_k^2, \leq_2)$ have infinite width.

Lemma 2. *For all* $k > 2$ *and* $n \in \omega$ *there exist pairwise* 2*-incomparable (i.e. incomparable in* $\leq_2$*) repetition-free* k*-chains* $C_0, \ldots, C_n$.

Proof. For every repetition-free k-word α whose first symbol is not $k-1$, let D_α denote the repetition-free k-chain presented by the k-word $0 \ldots (k-1)\alpha$. It is straightforward to check that D_α and D_β are 2-incomparable if $|\alpha| = |\beta|$ and $\alpha \neq \beta$ (consider non-equal labels on the same place in α and β). It remains to note that $|\{D_\alpha : |\alpha| = n\}| = (k-1)^n$ for any $n \in \omega$.

3 Interpretation scheme

We use the same interpretation scheme as in [9]. As is well known [4], for establishing hereditary undecidability of the first-order theory of a structure, say of a partial order $(P;\leq)$, it suffices to show that the class of finite models of the theory of two equivalence relations is relatively elementarily definable in $(P;\leq)$ with parameters [4]. It turns out that the following very particular case of the last notion is sufficient for our proof.

It suffices to find first-order formulas $\phi_0(x,\bar{p})$, $\phi_1(x,y,\bar{p})$ and $\phi_2(x,y,\bar{p})$ of signature $\{\leq\}$ (where x, y are variables and $\bar{p}$ is a string of variables called parameters) with the following property:

(*) for every $n < \omega$ and for all equivalence relations ξ, η on $\{0, \ldots, n\}$ there are values of parameters $\bar{p} \in P$ such that the structure $(\{0, \ldots, n\}; \xi, \eta)$ is isomorphic to the structure $(\phi_0(P,\bar{p}); \phi_1(P,\bar{p}), \phi_2(P,\bar{p}))$.

Here

$$\phi_0(P,\bar{p}) = \{a \in P | (P;\leq) \models \phi_0(a,\bar{p})\},$$
$$\phi_1(P,\bar{p}) = \{(a,b) \in P | (P;\leq) \models \phi_1(a,b,\bar{p})\}$$

and similarly for ϕ_2. In other words, for all n, ξ, η as above there are parameter values $\bar{p} \in P$ such that the set $\{0, \ldots, n\}$ and the relations ξ, η are first-order definable in $(P;\leq)$ with parameters $\bar{p}$.

So for each of the quotient structures $(\mathfrak{F}_k^1, \leq_1)$, $(\mathfrak{F}_k^2, \leq_2)$, $(\mathfrak{T}_k^1, \leq_1)$, $(\mathfrak{T}_k^2, \leq_2)$ it remains only to find suitable formulas ϕ_0, ϕ_1, ϕ_2 and to specify parameter values as described in (*).

4 Main Result

The result of P. Hertling [5, 6, 8] states that the quotient-structure of the preorder $\leq_0$ (resp. $\leq_1$, $\leq_2$) on the k-partitions of the Baire space on subsets whose components are boolean combinations of open sets is isomorphic to $(\mathfrak{F}_k^0; \leq_0)$ (resp. to $(\mathfrak{F}_k^1; \leq_1)$, $(\mathfrak{F}_k^2; \leq_2)$). Using this result, we may consider the last structures instead of the corresponding structures of Weihrauch degrees.

The main result of this paper is formulated as follows.

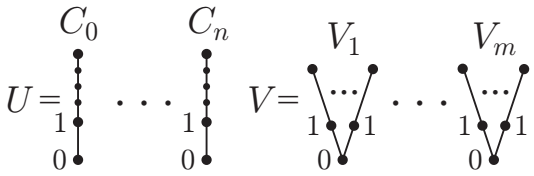
Theorem 1. *For any $k \geq 3$, the first-order theories of the structures $(\mathfrak{F}^1_k, \leq_1)$, $(\mathfrak{F}^2_k, \leq_2)$, $(\mathfrak{T}^1_k, \leq_1)$, $(\mathfrak{T}^2_k, \leq_2)$ are hereditarily undecidable.*

Scetch of the proof. Let $\tau(x,u)$ be the formula $x \leq u \wedge ir(x) \wedge \neg\exists y > x(y \leq u \wedge ir(y))$ which means that x is a maximal non-zero quasijoin-irreducible element below u. Let $\bar{p} = (u,v,w)$ and $\phi_0(x,\bar{p}) = \tau(x,u)$. Let $\phi_1(x,y,\bar{p})$ be the formula $\tau(x,u) \wedge \tau(y,u) \wedge \exists t(\tau(t,v) \wedge x \leq t \wedge y \leq t)$ and $\phi_2(x,y,\bar{p})$ is obtained from $\phi_1(x,y,\bar{p})$ by substituting w in place of v.

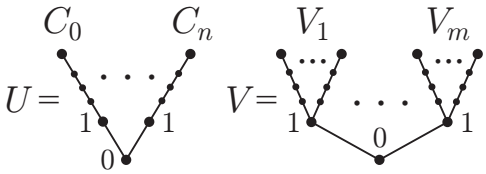
Assume that $\xi, \eta \subseteq (n+1)^2$ are arbitrary equivalence relations on $n+1 = \{0,\ldots,n\}$ and $\{\xi_0,\ldots,\xi_m\}$, $\{\eta_0,\ldots,\eta_l\}$ are the quotient sets $n+1/\xi$ and $n+1/\eta$, respectively.

We use the same formulas ϕ_0, ϕ_1 and ϕ_2 for all structures in Theorem 1 (with $\leq_1$ or $\leq_2$ instead of $\leq$) with parameters u, v and w depending on ξ. For interpreting the elements $0,\ldots,n$ from the set $n+1$, we use 2-incomparable k-chains $C_0,\ldots,C_n$ from Lemma 2. We can assume w.l.o.g. that all k-chains $C_0,\ldots,C_n$ are disjoint, their roots are labeled with 0 and the successors of the roots are labeled with 1 (by the proof of Lemma 2).

We construct parameters u, v and w from $C_0,\ldots,C_n$ in the following way. For the case of $(\mathfrak{F}^1_k, \leq_1)$ and $(\mathfrak{F}^2_k, \leq_2)$, let $U = C_0 \sqcup \ldots \sqcup C_n$ and $V = V_1 \sqcup \ldots \sqcup V_m$ where V_i is obtained from $\{C_j | j \in \xi_i\}$ by identifying the roots, $V_i \equiv_0 p_0(\bigsqcup\{C_j | j \in \xi_i\})$ (pic.1). Define W as V with η instead of ξ. Namely, $W = W_1 \sqcup \ldots \sqcup W_l$ where W_i is obtained from $\{C_j | j \in \eta_i\}$ by identifying the roots, $W_i \equiv_0 p_0(\bigsqcup\{C_j | j \in \eta_i\})$. Now take $u = [U]_{\equiv_i}$, $v = [V]_{\equiv_i}$ and $w = [W]_{\equiv_i}$ for $\mathfrak{F}^i_k$.



Pic.1. Parameters U and V for $\mathfrak{F}^1_k$ and $\mathfrak{F}^2_k$.



Pic.2. Parameters U and V for $\mathfrak{T}^1_k$ and $\mathfrak{T}^2_k$.

For the case of $(\mathfrak{F}^1_k, \leq_1)$ and $(\mathfrak{F}^2_k, \leq_2)$, the parameters are a bit more complicated than in the previous case, see pic. 2. Let U be obtained from $C_0,\ldots,C_n$ by identifying the roots, $U \equiv_0 p_0(C_0 \sqcup \cdots \sqcup C_n)$. For each i from 1 to l, let V_i be obtained from $\{C_j | j \in \xi_i\}$ by identifying the roots and their successors. Now V is constructed by identifying the roots of all V_i, $1 \leq i \leq l$. Define W as V with η instead of ξ. Take $u = [U]_{\equiv_i}$, $v = [V]_{\equiv_i}$, $w = [W]_{\equiv_i}$ for $\mathfrak{T}^i_k$.

References

1. Brattka, V., Gherardi, G.: Weihrauch degrees, omniscience principles and weak computability. http://arxive.org/abs 0905.4679 (2009)
2. Brattka, V., Gherardi, G.: Effective choice and boundedness principles in computable analysis. http://arxive.org/abs 0905.4685 (2009)
3. Davey, B.A., Pristley, H.A.: Introduction to Lattices and Order. Second edition, Cambridge University Press, 2002
4. Ershov, Yu.L., Lavrov, T.A., Taimanov, A.D., Taitslin, M.A.: Elementary theories. Uspechi Mat. Nauk **20** N 4 (1965) 37–108 (in Russian)
5. Hertling, P.: Topologische Komplexitätsgrade von Funktionen mit endlichem Bild. Informatik-Berichte 152, Fernuniversität Hagen, 1993
6. Hertling, P.: Unstetigkeitsgrade von Funktionen in der effektiven Analysis, Informatik Berichte **208-11/1996**, Fernuniversität Hagen, 1996
7. Hertling, P., Weihrauch, K.: Levels of degeneracy and exact lowercomplexity bounds for geometric algorithms. Proc. of the 6th Canadian Conf. on Computational Geometry Saskatoon (1994) 237-24
8. Selivanov, V.L.: Hierarchies of $\mathbf{\Delta}^0_2$-measurable k-partitions. Math. Logic Quarterly **53** (2007) 446–461
9. Kudinov, O.V., Selivanov, V.L.: Undecidability in the homomorphic quasiorder of finite labelled forests. Journal of Logic and Computation **17** (2007) 1135–1151
10. Weihrauch, K.: The degrees of discontinuity of some translators between representations of the real numbers. Technical Report TR-92-050, International Computer Science Institute, Berkeley, 1992.
11. Weihrauch, K.: Computable Analysis. Springer Verlag, Berlin (2000)

Diagonal sets for real number complexity classes

Klaus Meer

Computer Science Institute
BTU Cottbus
Konrad-Wachsmann-Allee 1
D-03046 Cottbus, Germany
meer@informatik.tu-cottbus.de

Abstract. Ladner has shown that there are non-complete problems in NP \ P assuming NP ≠ P. We survey results of similar type in computational settings different from the classical Turing machine. The latter include real and complex Turing machines and several of its variants as well as Valiant's complexity theory for families of polynomials over infinite fields.

Keywords: Complexity, real number model, diagonal problems

1 Introduction

Starting point of our investigations is the following classical result by Ladner [6]:

Theorem 1. *Suppose* NP ≠ P. *Then there are problems in* NP \ P *which are not* NP*-complete under polynomial time many-one reductions.*

Its proof relies intrinsically on the countability of both the family $\{P_1, P_2, \ldots\}$ of polynomial time machines and the family $\{R_1, R_2, \ldots\}$ of polynomial time reduction machines. A diagonalization argument is performed to fool one after the other each machine in the two sets. This is briefly done as follows. Given an NP-complete problem L one constructs a problem $\tilde{L} \in$ NP such that all machines R_i fail to reduce L to $\tilde{L}$ on some input and all P_i fail to decide $\tilde{L}$ correctly on some input. Towards this aim the definition of $\tilde{L}$ proceeds dimensionwise while intending to fool step by step $P_1, R_1, P_2, R_2, \ldots$. In order to fool an P_i the language $\tilde{L}$ is taken to look like L for inputs of sufficiently large size. Conversely, in order to fool reduction algorithms R_i for sufficiently many of the following input-sizes $\tilde{L}$ looks like an easy problem. In addition, a typical padding argument guarantees $\tilde{L} \in NP$.

Extensions of Ladner's result can, f.e., be found in [11].

Considering computational models over uncountable structures like $\mathbb{R}$ and $\mathbb{C}$ the above diagonalization argument - at least at a first sight - fails since the corresponding algorithm classes become uncountable.

In this presentation we survey what has been done in the last 10 years in order to obtain Ladner-type results in different frameworks dealing with uncountable structures.

2 Some uncountable settings

We are interested in the following computational settings. For more severe introductions see [2, 4].

The Blum-Shub-Smale model (BSS for short) over $\mathbb{R}$ or $\mathbb{C}$, respectively, treats reals or complex numbers as basic entities. It performs basic operations among $\{+, -, \cdot\}$ together with a test $x \geq 0$? (for $\mathbb{R}$) or $x = 0$? (for $\mathbb{C}$) at unit cost. If only $\{+, -\}$-operations are allowed we get the *additive* BSS model.We denote the resulting complexity classes analogue to P and NP in these models by adding self-explaining sub- and superscripts like $\mathrm{P}_{\mathbb{R}}, \mathrm{P}_{\mathbb{C}}, \mathrm{P}_{\mathbb{R}}^{add}$ etc.

Of particular interest below will be the set of machine constants used by a BSS algorithm. Each uniform algorithm M has access to finitely many constants $c_1, \ldots, c_k$ from the underlying structure. These constants can be used within the algorithm. The way how the c_i's occur in M's computation leads to two more definitions important later on.

If all intermediate results in any computation of M depend linearly on the c_i only we get the BSS model with *restricted use of contants* introduced in [8]. As we shall discuss later on in this model the

currently strongest uniform Ladner type result for real Turing machines is known. We denote complexity classes in the restricted model by adding an superscript *rc*.

The notion of basic machines was introduced by Michaux [9]. It is important in order to perform diagonalization arguments like Ladner's one as well in the frameworks we are interested in.

Definition 1. *A basic machine over $\mathbb{R}$ in the BSS-setting is a BSS-machine M with rational constants and with two blocks of parameters. One block x stands for a concrete input instance and takes values in $\mathbb{R}^\infty$, the other one c represents real constants used by the machine and has values in some $\mathbb{R}^k$ ($k \in \mathbb{N}$ fixed for M). Here $\mathbb{R}^\infty := \bigcup_{n\geq 1} \mathbb{R}^n$ denotes the set of all finite sequences of real numbers.*

Basic machines for the other models mentioned above are defined similarly.

The above definition of a basic machine intends to split the discrete skeleton of an original BSS machine from its real machine constants. That is done by regarding those constants as a second block of parameters. Fixing c we get back a usual BSS machine $M(\bullet, c)$ that uses the same c as its constants for all input instances x. If below we speak about the machine's constants we refer to the potentially real ones only.

Basic machines give rise to define a non-uniform complexity class P/const for the different model variants we consider. The non-uniformity is literally weaker than the well-known $P/poly$ class since the non-uniform advice has fixed dimension for all inputs.

Definition 2. *(cf. [9]) A problem L is in class $\mathrm{P}_\mathbb{R}$/const if and only if there exists a polynomial time basic BSS machine M and for every $n \in \mathbb{N}$ a tuple $c^{(n)} \in \mathbb{R}^k$ of real constants for M such that $M(\bullet, c^{(n)})$ decides L for inputs up to size n.*

Similarly for $\mathrm{P}_\mathbb{C}$/const, $P_\mathbb{R}^{\mathrm{add}}$/const, and $P_\mathbb{R}^{\mathrm{rc}}$/const.

The final setting we consider below is Valiant's complexity theory for families of polynomials with coefficients in a field k together with the corresponding complexity classes VP and VNP, see [4] for a detailed introduction.

3 Diagonal problems in uncountable structures: Survey of results

We next discuss which results of Ladner type have been established for these settings.

3.1 Complex Turing machines

The first result in the BSS framework was shown in [7] for the complex classes $\mathrm{P}_\mathbb{C}$ and $\mathrm{NP}_\mathbb{C}$:

Theorem 2 ([7]). *Suppose $NP_\mathbb{C} \neq P_\mathbb{C}$. Then there are problems in $NP_\mathbb{C} \setminus P_\mathbb{C}$ which are not $NP_\mathbb{C}$-complete under polynomial time many-one reductions.*

The proof relies on a theorem by Shub and Smale, see chapter 6 in [2]. It basically says that for a problem in $\mathrm{NP}_\mathbb{C}$ that can be defined without complex constants the use of complex machine constants does not speed up any decision algorithm significantly. Since there exist $\mathrm{NP}_\mathbb{C}$-complete problems having the above property it can be shown that the theorem allows to transfer Ladner's problem over $\mathbb{C}$ to the analogue problem over $\bar{\mathbb{Q}}$, the algebraic closure of $\mathbb{Q}$ in $\mathbb{C}$. Here, the classical proof can be performed.

3.2 Unifying argument using P/const

The next step was done in [1] by using arguments on the class P/const in the corresponding models. The class of basic machines clearly is countable as long as the particular choice of machine constants is not fixed. Thus, in principle we can diagonalize over P/const decision and reduction machines in the different models.

Theorem 3. *[1] Suppose $NP_\mathbb{R} \not\subseteq P_\mathbb{R}$/const. Then there exist problems in $NP_\mathbb{R} \setminus P_\mathbb{R}$/const not being $NP_\mathbb{R}$-complete under $P_\mathbb{R}$/const reductions.*

Similarly for the other model variants.

The proof again uses the usual padding argument along the classical line. The main new aspect, however, is the necessity to establish that for each basic machine M which is supposed to decide the intended diagonal problem $\tilde{L}$ an input-dimension where M's result disagrees with $\tilde{L}$'s definition can be computed effectively. The condition that M disagrees with $\tilde{L}$ for all possible choices of machine constants can be expressed via a quantified first-order formula. Deciding the latter then is possible due to the existence of quantifier elimination algorithms in the respective structures.

Since the assumption of Theorem 3 deals with $\mathrm{P}_{\mathbb{R}}/\mathrm{const}$ instead of $\mathrm{P}_{\mathbb{R}}$ it gives a non-uniform version of Ladner's result. Note that because of $\mathrm{P}_{\mathbb{R}} \subseteq \mathrm{P}_{\mathbb{R}}/\mathrm{const}$ the theorem's implication also holds for uniform reductions. In order to achieve stronger versions one next has to study the relation between the classes P and P/const. If both are equal, then a uniform version of the theorem follows.

Theorem 4 ([9],[1]). *In recursively saturated structures it is* $P = \mathrm{P}/\mathrm{const}$. *As a consequence, Ladner's results holds uniformly over structures like* $\{0,1\}$ *and* $\mathbb{C}$.

Thus, Ladner's original result as well as Theorem 2 are reproved. Since $\mathbb{R}$ is not recursively saturated the theorem's consequence does not apply to $\mathbb{R}$.

Remark 1. Similar results were independently obtained by B. Poizat in [10].

3.3 Additive and restricted BSS model

Due to its importance for the above questions Chapuis and Koiran in [5] have undertaken a deep model-theoretic analysis of P/const and related classes. They argue that for the full real model already the equality $\mathrm{P}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}/1$ is highly unlikely unless some major complexity theoretic conjecture is violated. Here, $\mathrm{P}_{\mathbb{R}}/1$ is defined by means of basic machines which use a finite number of uniform and a single non-uniform machine constant only. Nevertheless, for the reals with addition and order (additive model) they were able to show once again $\mathrm{P}_{\mathbb{R}}^{add} = \mathrm{P}_{\mathbb{R}}^{add}/\mathrm{const}$ and thus

Theorem 5 ([5]). *Suppose* $NP_{\mathbb{R}}^{add} \neq P_{\mathbb{R}}^{add}$. *Then there are problems in* $NP_{\mathbb{R}}^{add} \setminus P_{\mathbb{R}}^{add}$ *which are not* $NP_{\mathbb{R}}^{add}$*-complete.*

Their proof for showing the inclusion $\mathrm{P}_{\mathbb{R}}^{add}/\mathrm{const} \subseteq \mathrm{P}_{\mathbb{R}}^{\mathrm{add}}$ makes use of the moderate growth of intermediate results in an additive computation. This allows to bound the size of and compute efficiently and uniformly for each input dimension n a set of rational machine constants $c^{(n)}$ such that the given $\mathrm{P}_{\mathbb{R}}^{add}/\mathrm{const}$-machine works correctly on $\mathbb{R}^{\leq n}$ if $c^{(n)}$ is taken as vector of constants.

This idea is one of the starting points in [8] to extend the result to the restricted model. We point out that this model is closer to the original full real BSS model than the additive one is. As one indication for this fact note that the $\mathrm{NP}_{\mathbb{R}}$-complete feasibility problem $FEAS$ which asks for solvability of a system of polynomial equations over $\mathbb{R}$ is $\mathrm{NP}_{\mathbb{R}}^{\mathrm{rc}}$-complete as well in the restricted model. Since Theorem 3 holds as well here the main task is to analyze the relation between $\mathrm{P}_{\mathbb{R}}^{\mathrm{rc}}$ and $\mathrm{P}_{\mathbb{R}}^{\mathrm{rc}}/\mathrm{const}$.

Theorem 6 ([8]). *It is* $P_{\mathbb{R}}^{\mathrm{rc}} = P_{\mathbb{R}}^{\mathrm{rc}}/\mathrm{const}$. *As a consequence, supposing* $FEAS \notin NP_{\mathbb{R}}^{\mathrm{rc}}$ *there exist non-complete problems in* $NP_{\mathbb{R}}^{\mathrm{rc}} \setminus P_{\mathbb{R}}^{\mathrm{rc}}$.

Crucial for showing $\mathrm{P}_{\mathbb{R}}^{\mathrm{rc}} = \mathrm{P}_{\mathbb{R}}^{\mathrm{rc}}/\mathrm{const}$ is a certain convex structure underlying the set of suitable machine constants. Given a problem $L \in \mathrm{P}_{\mathbb{R}}^{\mathrm{rc}}/\mathrm{const}$ and a corresponding basic machine M using k constants define $E_n \subset \mathbb{R}^k$ as set of constants that can be used by M in order to decide $L \cap \mathbb{R}^{\leq n}$ correctly. It can be shown that without loss of generality the $\{E_n\}_n$ build a nested sequence of convex sets with empty intersection. The main point now is to establish by a limit argument in affine geometry the following: There exist three vectors $c^*, d^*, e^* \in \mathbb{R}^k$ such that for all $n \in \mathbb{N}$ and small enough $\mu_1 > 0, \mu_2 > 0$ (μ_2 depending on μ_1 and both depending on n) machine M correctly decides $L \cap \mathbb{R}^{\leq n}$ when using $c^* + \mu_1 \cdot d^* + \mu_2 \cdot e^*$ as its constants.

This is sufficient to change M into a polynomial time restricted machine that decides L and uses $c^*, d^*, e*$ as its *uniform* machine constants.

Let us summarize the methods described so far. The diagonalization technique used above allows some degree of freedom as to how to define $\mathrm{P}_{\mathbb{R}}/\mathrm{const}$. This means that we can put some additional conditions onto the set of constants that we allow for a fixed dimension to work. To make the diagonalization

work there are basically two aspects that have to be taken into account. First, the resulting class has to contain $\mathrm{P}_{\mathbb{R}}$. Secondly, the conditions we pose on the constants have to be semi-algebraically definable without additional real constants. Playing around with suitable definitions might be a way to attack Ladner's problem as well in the full real number model. However, for a problem L in $\mathrm{P}_{\mathbb{R}}/\mathrm{const}$ the topological structure of the set of suitable constants is more complicated since now each branch results in a (potentially infinite) intersection of semi-algebraic conditions. Then one has to study how the topology of the sets $\bigcap_{i=1}^{N} E_i$ evolves for increasing N. For example, could one guarantee the existence of say a semi-algebraic limit curve along which one could move from a point c^* into an E_n? In that case, a point on the curve might only be given by a semi-algebraic condition. As consequence, though one would likely not be able to show $\mathrm{P}_{\mathbb{R}}/\mathrm{const} \subseteq \mathrm{P}_{\mathbb{R}}$ may be at least a weaker uniform version of Ladner's result could be settled.

3.4 Valiant's model

The final Ladner type result to be briefly discussed here deals with Valiant's classes VP and VNP and was shown by Bürgisser in [3]. For the definition of these classes and p-projections as a reduction notion see [4].

Theorem 7 ([3]). *Suppose $VNP \neq VP$ in Valiant's model over a field k. Then there exists a family of polynomials in $VNP \setminus VP$ that is not VNP-complete under p-projections.*

As in Theorem 3 non-uniformity again plays an important role in proving the theorem. Note that Valiant's model already by definition is non-uniform since the straight-line programs that define the members of a polynomial family in class VNP are not necessarily related to each other. This fact again allows a particular enumeration of all straight-line programs of polynomial size over an uncountable field "in blocks", similar in spirit to the enumeration of basic machines.

For finite coefficient fields [3] moreover gives concrete problems of such an intermediate complexity. Since all the problems constructed above for the different BSS models are quite artificial it remains an interesting open question to find more natural problems satisfying Ladner's theorem as well in these settings.

References

1. S. Ben-David, K. Meer, and C. Michaux. A note on non-complete problems in $\mathrm{NP}_{\mathbf{R}}$. *J. Complexity*, 16(1):324–332, 2000.
2. Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998.
3. Peter Bürgisser. On the structure of Valiant's complexity classes. In *STACS 98 (Paris, 1998)*, volume 1373 of *Lecture Notes in Comput. Sci.*, pages 194–204. Springer, Berlin, 1998.
4. Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig.
5. Olivier Chapuis and Pascal Koiran. Saturation and stability in the theory of computation over the reals. *Ann. Pure Appl. Logic*, 99(1-3):1–49, 1999.
6. Richard E. Ladner. On the structure of polynomial time reducibility. *J. Assoc. Comput. Mach.*, 22:155–171, 1975.
7. Gregorio Malajovich and Klaus Meer. On the structure of $\mathrm{NP}_{\mathbb{C}}$. *SIAM J. Comput.*, 28(1):27–35, 1999.
8. Klaus Meer. On Ladner's result for a class of real machines with restricted use of constants. In *Computability in Europe*, volume 5635 of *Lecture Notes in Comput. Sci.*, pages 352–361. Springer, 2009.
9. Christian Michaux. $\mathrm{P} \neq \mathrm{NP}$ over the nonstandard reals implies $\mathrm{P} \neq \mathrm{NP}$ over $\mathbf{R}$. *Theoret. Comput. Sci.*, 133(1):95–104, 1994.
10. Bruno Poizat. Gonflette dans les modèles ω-saturés. Exposé au groupe de travail LIP-IGD "Complexité Algébrique", 1996.
11. Uwe Schöning. A uniform approach to obtain diagonal sets in complexity classes. *Theoret. Comput. Sci.*, 18(1):95–103, 1982.

Nash Equilibria and Fixed Points in the BSS-Model

Arno Pauly*

University of Cambridge
Computer Laboratory
Cambridge CB3 0FD
United Kingdom

1 Introduction

Some computational problems elude a natural formulation as decision problems, and are therefore not adequately covered in the study of complexity classes such as P or NP. Motivated by game theoretic problems in particular, and fixed point problems in general, complexity classes of total search problems have been studied in the classical framework. A famous example is the class PPAD introduced in [10], which has computing a Nash equilibrium in a two player game, an ε-Nash equilibrium in an n-player game or a Brouwer fixed point of a suitably represented function as complete problems ([10], [7], [8]).

A unifying property of these problems is their continuous nature, which immediately suggests to study them in a framework of real-number computation. In the computable analysis approach ([14]), none of these problems is computable, as they are all discontinuous. The exact degree of incomputability (i.e. the Weihrauch-degree [5], [4], [12]) of finding equilibria in two-player games was classified in [11].

A hybrid approach was chosen in [9], by assuming that the instances (such as games or functions) are given as finite sequences of bits, while the solutions are vectors of real numbers. This solves e.g. the obstacle that a three-player game with rational payoffs can have purely irrational equilibria. By including a suitable notion of reducibility, equivalences shown in this approach directly transfer to the associated discrete problems. As a result, the class FIXP is defined, which has finding Nash equilibria in n-player games (or in 3-player games) and several generic fixed point problems as complete problems.

The goal of the present paper is to initiate the study of the corresponding problems in the BSS-model ([3]). A number of interesting equivalences between problems can be obtained by inspection of the reductions given in [9].

2 $\mathbf{FIXP}_{\mathbb{R}}$

Our starting point to define the class $\mathrm{FIXP}_{\mathbb{R}}$ will be circuits defining continuous functions from either the unit cube, the unit ball or the unit simplex into itself. By Brouwer's Fixed Point Theorem, such a function will have a fixed point, so finding a fixed point for each circuit is a total search problem. The circuits shall be coded in a way that admits (potential) evaluation in polynomial time, e.g. by using a real number for each individual gate. Then we obtain the following result:

Theorem 1. *The following problems are equivalent under polynomial time BSS-reductions:*

1. *Finding a fixed point for a circuit using gates from* $\{+, *, /, \sqrt[n]{\ }, \max\}$.
2. *Finding a fixed point for a circuit using gates from* $\{+, *, \max\}$.
3. *Finding an exact Nash equilibrium for a* 3*-player game.*
4. *Finding an exact Nash equilibrium for an* n*-player game.*

Proof. 4. $\Rightarrow$ 3. *The construction given in [6] can be executed by a polynomial time BSS-machine.*
3. $\Rightarrow$ 2. *The function used in the proof of Nash's Theorem by Brouwer's Fixed Point Theorem is the sought reduction.*
2. $\Rightarrow$ 1. *Trivial.*
1. $\Rightarrow$ 4. *The reduction defined in the proof of [9, Theorem 18] does not depend on the inputs being rational numbers, and the number of algebraic operations executed during the reduction only depends on the number of coefficients present in the input, not on their size.*

* Arno.Pauly@cl.cam.ac.uk

Now we can define $\mathrm{FIXP}_\mathbb{R}$ as the class of problems reducible to any of the problems above. By the first problem, it is obvious that a $\mathrm{FIXP}_\mathbb{R}$-complete problem cannot be solved by a deterministic BSS-machine, as finding square-roots is reducible to it. Examination of the second problem shows $\mathrm{FIXP}_\mathbb{R} \subseteq \mathrm{FNP}_\mathbb{R}$: As we can evaluate all the gates in constant time, the circuit itself can be evaluated in polynomial time. If we guess a fixed point, we can verify it.

3 $PPAD_\mathbb{R}$

As we have seen, roots and division can be removed from the set of available circuit gates without altering the difficulty of finding fixed points. In a similar way, we can reduce the number of players in the games down to 3. Restricting the problems further will yield a new equivalence class, and the class $PPAD_\mathbb{R}$:

Theorem 2. *The following problems are equivalent under polynomial time BSS-reductions:*

1. *Finding a fixed point for a circuit using gates from* $\{+, *c, \max\}$[1].
2. *Finding an exact Nash equilibrium for a* 2*-player game.*

Proof. The corresponding reductions given in [9] can be extended to polynomial time BSS-reductions.

Obviously, we have $\mathrm{PPAD}_\mathbb{R} \subseteq \mathrm{FIXP}_\mathbb{R}$. In the discrete setting, nothing more is known (although a strict inclusion is strongly expected). Here, however, we can prove the separation of the two classes, as a consequence of the following theorem:

Theorem 3. *All problems in* $\mathrm{PPAD}_\mathbb{R}$ *can be solved by a deterministic BSS-machine.*

Proof. It is sufficient to show that a deterministic BSS-machine can find Nash equilibria in two player games. The reduction from NASH *to* $\overline{\text{RDIV}}$ *given in [11] can be interpreted as an algorithm for the BSS-model, establishing the claim.*

Corollary 1. $\mathrm{PPAD}_\mathbb{R} \subsetneq \mathrm{FIXP}_\mathbb{R}$.

4 Outlook

The reductions presented here can be adapted quickly to link decision problems derived from the original search problems. However, it might be more promising to extend the study of search problems to include further interesting applications. Linear programming seems to be a straight-forward choice. Besides being an open problem of considerable importance ([13], [2]), the search problem is more relevant than the decision problem: If a solution exists, it should be returned. Game theory and linear programming are sufficiently related to expect a link between the two problems.

Another line of reasoning leads back into the classical setting. If PPAD and FIXP should be identical, the corresponding reduction must fail to transfer to the reals in some way. This constitutes a new type of barrier, in addition to relativization, natural proofs and algebrization ([1]); although we have to admit, one of lesser importance. Nevertheless, the identification of this barrier might be useful to study the relationship between PPAD and FIXP.

This significantly restricts the available options, as purely algebraic reductions are ruled out. As an alternative to non-algebraic operations, a different treatment of rational and irrational numbers might be involved. It is even conceivable that this restriction might be useful to prove the separation in the bit model.

Finally, the original definition of PPAD given in [10] is of a different nature than the one we transferred here. If we wish to stay close to the first one, we arrive at the following problem: An instance is a directed graph with in- and out-degree bounded by 1 for each vertex in the form of a BSS-machine that computes the adjacent vertices for each vertex, where vertices are denoted by real vectors of a fixed dimension (dependent on the machine), and the 0-vector has no incoming edges. The tasks is to find another vertex v with $\text{in-degree}(v) + \text{out-degree}(v) \leq 1$. How is this problem related to the problems studied in the present paper?

[1] $*c$ denotes multiplication by constants only.

References

1. S. Aaronson and A. Wigderson. Algebrization: A New Barrier in Complexity Theory. *ACM Trans. Comput. Theory*, 1(1):1–54, 2009.
2. G. Bär. On the Complexity of Linear Programming in the BSS-Model. *Discrete Applied Mathematics*, 95:35–40, 1999.
3. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer, 1998.
4. V. Brattka and G. Gherardi. Effective Choice and Boundedness Principles in Computable Analysis. arXiv:0905.4685v1, May 2009.
5. V. Brattka and G. Gherardi. Weihrauch Degrees, Omniscience Principles and Weak Computability. arXiv:0905.4679v1, May 2009.
6. V. Bubelis. On Equilibria in Finite Games. *International Journal of Game Theory*, 8(2):65–79, 1979.
7. Xi Chen and Xiaotie Deng. Settling the Complexity of 2-Player Nash-Equilibrium. Technical Report 134, Electronic Colloquium on Computational Complexity, 2005.
8. Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Computing Nash Equilibria: Approximation and Smoothed Complexity. Technical Report 23, Electronic Colloquium on Computational Complexity, 2006.
9. K. Etessami and M. Yannakakis. On the Complexity of Nash Equilibria and other Fixed Points. to appear.
10. C. Papadimitriou. On the Complexity of the Parity Argument and other Inefficient Proofs of existence. *Journal of Computer and Systems Science*, 48(3):498–532, 1994.
11. A. Pauly. How Discontinuous is Computing Nash Equilibria? arXiv:0907.1482v1, July 2009.
12. A. Pauly. On the (semi)lattices induced by continuous reducibilities. arXiv:0903.2177v1, March 2009.
13. S. Smale. Mathematical Problems for the Next Century. *Mathematical Intelligencer*, 20:7–15, 1998.
14. K. Weihrauch. *Computable Analysis*. Springer-Verlag, 2000.

Une dualité entre fonctions booléennes

Bruno POIZAT
poizat@math.univ-lyon1.fr

(To be published in the Journal de Jussieu)

Summary of results

We consider a finitely generated ring A, which will be mainly a finite field k, or the ring Z of the integers, or the ring Z/nZ of the integers modulo n. What we call a function, in m variables, $f(x_1, \ldots x_m)$ is an application from $\{0, 1\}^m$ to A.

Any function can be written in a unique way as a polynomial of degree zero or one in each of its variables, with coefficients in k ; the dual function f^* of f is the function, in the same number of variables, whose f expresses the coefficients of the canonical polynomial ; in other terms, $f^*(x_1, \ldots x_m) = \Sigma_{\underline{u} \text{ boolean}} f(u_1,\ldots u_m).x_1^{u1}. \ldots .x_m^{um}$, where $x^y = y.x - y + 1$ ($x^y = 1$ if $y = 0$, $x^y = x$ if $y = 1$).

Duality is a special case of a transformation T, that is the data, for each m, of an inversible linear map T_m between functions of arity m satisfying the Fubini condition : $T_m(f).T_{m'}(g) = T_{m+m'}(f.g)$ when the m-tuple of the variables of f is disjoint from the m'-tuple of the variables of g.

A transformation is determined by its restriction T_1 on the space of functions in one variable, so that the group of transformations is isomorphic to the group $GL2(k)$ of two by two invertible matrices with coefficients in k.

In fact, to any transformation T is associated a function $\nu(y,x)$ in two variables, that we call its kernel, such that $T_1(f)(x) = \Sigma_{t \in \{0,1\}} f(t).\nu(t,x)$, and $T_m(f)(x_1, \ldots x_m) = \Sigma_{\underline{t} \text{ booléen}} f(t_1,t_2,\ldots t_m).\nu(t_1,x_1). \ldots .\nu(t_m,x_m)$ for each integer m, thanks to Fubini condition.

Our aim is to evaluate the effect of transformations on the complexity of functions. By definition, the complexity of f is the number of arithmetic operations (additions and multiplications) that are necessary to obtain f

starting from variables and constants choosen in a fixed generating system of A . The class $P(A)$ is the set of sequences of functions whose complexity grows polynomially ; $Pt(A)$ is the set of sequence of functions computed by arithmetics terms of polynomial size, or, equivalently, by arithmetic circuits of logarithmic depth. An intermediate an less robust class is Malod's $Pmd(A)$ associated to multiplicatively disjoint circuits. No separation results is known between $Pt(A)$ and $Pmd(A)$, whatever be the ring A .

The classes $SP(A)$, $SPmd(A)$ and $SPt(A)$ are obtained by summation respectively before $P(A)$, $Pmd(A)$ and $Pt(A)$; in fact, $SPt(A) = SPmd(A)$. Moreover, when the ring A is finite, the parcimonious reduction of circuits to terms implies that $SP(A) = SPt(A)$.

The complexity classes that we consider are non uniform ; we could give them a polynomial uniformity as well, without affecting our results.

From the existence of the kernel, it is clear that $SP(A)$ and $SPt(A)$ are closed under transformations.

Theorem. (*i) Whatever be the ring* A , $SPt(A) = Pt(A)$ *if and only if* $Pt(A)$ *is closed under transformations, if and only if* $Pt(A)$ *is closed by duality, if and only if* $Pt(A)$ *is closed by inverse duality.*
(ii) Whatever be the ring A , $SPmd(A) = Pmd(A)$ *if and only if* $Pmd(A)$ *is closed under transformations, if and only if* $Pmd(A)$ *is closed by duality, if and only if* $Pmd(A)$ *is closed by inverse duality.*
(iii) If the ring A *is finite ,* $SP(A) = P(A)$ *if and only if* $P(A)$ *is closed under transformations, if and only if* $P(A)$ *is closed by duality, if and only if* $P(A)$ *is closed by inverse duality.*

Sketch of the proof. A sequence of functions in $SPt(A)$ is obtained by projection from a sequence of functions which have a simple polynomial expression, i.e. an inverse dual in $Pt(A)$.

Now we consider the case where the ring A is a finite field k . When two finite fields k and k' have the same characteristic p , then $SP(k) = P(k)$ is equivalent to $SP(k') = P(k')$, and this is also equivalent to the well-known hypothesis $\#pP = P$ (concerning boolean classes of complexity). Similarly, $SPt(k) = Pt(k)$ if and only if $SPt(k') = Pt(k')$, and $SPmd(k) = Pmd(k)$ if and only if $SPmd(k') = Pmd(k')$.

Among the transformations, we distinguish the begnin ones, which for obvious reasons have a moderate effect on the complexity : $P(A)$, $Pmd(A)$ and $Pt(A)$ are therefore closed under begnin transformations. They are the

diagonal transformations, which multiply the functions in one variable by some $a.x + b$, where b and $a+b$ are invertible, and the antidiagonal transformations, obtained by composition of the diagonal transformations with the transformation corresponding to the exchange of the two values 0 and 1.

Theorem. *(i) If* k *is a finite field,* $SPt(k) = Pt(k)$ *if and only if* $Pt(k)$ *is closed under any non-begnin transformation.*
(ii) If k *is a finite field,* $SPmd(k) = Pmd(k)$ *if and only if* $Pmd(k)$ *is closed under any non-begnin transformation.*
(iii) If k *is a finite field,* $SP(k) = P(k)$ *if and only if* $P(k)$ *is closed under any non-begnin transformation.*

Sketch of the proof. Except when the field has three or five elements, the group of begnin transformations is maximal in $GL2(k)$; in the two exceptionnal cases, jump to the quadratic extension of k.

The paper concludes with considerations on characteristic zero. It is observed that the hypothesis $\#P = P$ corresponds to the possibility of making summation in figures.

The connexions between the two (unlikely) hypothesis $SP(Z) = P(Z)$ and $SPmd(Z) = Pmd(Z)$ are very mysterious ; they both imply $SPt(Z) \subseteq P(Z)$, an hypothesis on computations with integers that turns out to be equivalent to $\#P = P$, which is an hypothesis concerning boolean ; it implies in turn each of the $\#pP = P$, as well as $NP = P$.

It can be shown that $\#nP = P$ is equivalent to the conjunction of the $\#pP = P$ for each of the prime divisors p of n.

Une dualité entre fonctions booléennes

Bruno POIZAT

Résumé. Si k est un corps fini, toute fonction $f(x_1, \ldots x_m)$ de $\{0, 1\}^m$ dans k s'écrit de manière unique comme un polynôme, à coefficients dans k, de degré un ou zéro en chacune de ses variables ; on peut donc lui associer une fonction $f^*(x_1, \ldots x_m)$, sa duale, qui exprime les coefficients de son polynôme canonique. Nous considérons l'improbable hypothèse que la classe $P(k)$, formée des suites de fonctions calculables en un nombre d'opérations (additions et multiplications) de croissance polynomialement bornée, soit close par dualité ; nous montrons qu'elle équivaut à une hypothèse bien connue en Théorie de la Complexité sous le nom de $P = \#pP$, où p est la caractéristique de k.

Dans une première section, nous exposons ce résultat lorsque $k = Z/2Z$, c'est-à-dire dans le cadre des calculs booléens classiques ; sa démonstration évite l'emploi d'un polynôme universel comme le hamiltonien : ses ingrédients sont d'une part la réduction parcimonieuse des circuits aux termes, et d'autre part la constatation que les expressions arithmétiques ont une duale très facile à calculer.

Dans la deuxième section, nous traitons le cas général, en introduisant une classe $SP(k)$ obtenue par sommation à partir de la classe $P(k)$; nous vérifierons dans la quatrième section l'équivalence des hypothèses $SP(k) = P(k)$ et $\#pP = P$. Nous y définissons également une notion de transformation, dont la dualité est un cas particulier. Les transformations forment un groupe isomorphe à $GL_2(k)$, avec un sous-groupe $B(k)$ de transformations que nous qualifions de bénignes, car elles n'ont que peu d'effet sur la complexité des fonctions ; nous montrons que toutes les transformations non-bénignes ont à peu près la même influence sur la complexité des fonctions, sauf si $k = F_3$ ou $k = F_5$; dans ces deux cas exceptionnels, la transformation de Fourier joue un rôle particulier.

Dans la troisième section, nous considérons des fonctions de k^m dans k ; nous n'y trouvons pas des classes de complexité vraiment nouvelles, mais seulement un groupe de transformations plus riche.

La quatrième section introduit l'égalité $\#P = P$ dans le paysage ; quant à la cinquième et dernière, elle examine le lien entre nos résultats et ceux de Guillaume Malod concernant la clôture par fonction-coefficient de diverses classes de complexité pour le calcul des polynômes à la manière de Valiant.

Nous nous sommes efforcés de rédiger cet article de manière à ce qu'il puisse être lu par des personnes non spécialisées en Algorithmie.

Mots clefs. Complexité des calculs, circuits et termes arithmétiques, expressions arithmétiques, parallélisation, réduction parcimonieuse, hamiltonien, Valiant, Malod.

1. Fonctions booléennes

Dans cette première section, nous nous plaçons dans le cadre usuel de l'étude de la complexité algorithmique, qui est celui des fonctions booléennes. Nous introduisons une transformation involutive entre fonctions booléennes, que nous appelons dualité, et nous évaluons son influence sur la complexité des calculs de parité.

1.1. Une fonction et sa duale

Nous qualifions de booléens deux éléments notés 0 et 1, et nous appelons fonction booléenne en m variables une application de $\{0, 1\}^m$ dans $\{0, 1\}$. Nous munissons $\{0, 1\}$ de la structure du corps $Z/2Z = F_2$ à deux éléments : $0 + 0 = 1 + 1 = 0$, $0 + 1 = 1 + 0 = 1$, $0.0 = 1.0 = 0.1 = 0$, $1.1 = 1$.

Toute fonction booléenne s'écrit comme un polynôme à coefficients dans F_2 ; on le voit par une induction aisée sur son nombre de variables, en utilisant la formule $f(x_1, ... x_m,x) = x.f(x_1, ... x_m,1) + (x+1).f(x_1, ...x_m,0)$. Comme 0 et 1 satisfont à l'égalité $x^2 = x$, on peut supposer ce polynôme de degré au plus un par rapport à chacune de ses variables ; l'écriture devient alors unique, car il y autant de fonctions booléennes en m variables que de polynômes en m variables de cette forme. Nous appellerons ce polynôme univoquement attaché à f le polynôme canonique de f.

Nous appelons duale f^* de f la fonction en le même nombre de variables dont f décrit les coefficients des monômes du polynôme canonique ; plus précisément, si nous notons $x^y = y.x + y + 1$ la fonction qui vaut 1 si $y = 0$, et qui vaut x si $y = 1$, $f^*(x_1, ... x_m) = \Sigma_{\underline{u} \text{ booléen}} f(u_1,... u_m).x_1^{u1}. ... .x_m^{um}$.

On voit que, dans cette écriture, un monôme en $(x_1, ... x_n)$, de degré un ou zéro en chacune de ses variables, est associé au uplet booléen $(u_1, ... u_n)$ tel que $u_i = 1$ si x_i figure dans le monôme, et $u_i = 0$ sinon.

Considérons par exemple la fonction ternaire $\mu(x,y,z)$ de majorité, qui vaut 1 s'il y a au moins deux 1 parmi les valeurs données aux variables x, y et z, et qui vaut 0 sinon. Son expression polynomiale canonique est $\mu(x,y,z) = x.y + y.z + z.x$. Comme la fonction de trois variables $x.y$ vaut 1 dans exactement deux cas, lorsque $x = y = 1$ et z vaut soit 0, soit 1, sa duale est $\Sigma_{u,v,w} u.v.x^u.y^v.z^w = x.y + x.y.z$; de même les duales de $y.z$ et de $z.x$ valent respectivement $y.z + x.y.z$ et $z.x + x.y.z$. Comme la dualité est de toute évidence linéaire, $\mu^*(x,y,z) = x.y + y.z + z.x + x.y.z$, qui vaut 1 si

exactement deux de ses variables prennent la valeur 1 ; en conséquence, μ^* est la fonction qui exprime les coefficients du polynôme canonique de μ. Autrement dit $(\mu^*)^* = \mu$; voici l'explication de ce miracle :

Théorème 1. *La dualité est involutive :* $(f^*)^* = f$ *, et* f^* *décrit les coefficients des monômes du polynôme canonique de* f.

Démonstration. $(f^*)^*(x_1, \ldots x_m) = \Sigma_{\underline{u}} f^*(u_1, \ldots u_m).x_1^{u1}. \ldots .x_m^{um} = \Sigma_{\underline{u}} [\Sigma_{\underline{v}} f(v_1,\ldots v_m).u_1^{v1}. \ldots .u_m^{vm}].x_1^{u1}. \ldots .x_m^{um} = \Sigma_{\underline{v}} f(v_1,\ldots v_m).[\Sigma_{\underline{u}} u_1^{v1}. \ldots .u_m^{vm}. x_1^{u1}. \ldots .x_m^{um}]$, la dernière égalité étant obtenue par l'échange des sommations. La somme entre crochets se calcule grâce au Théorème de Fubini, qui déclare que $\Sigma_{\underline{u},\underline{u}'} \varphi(\underline{u}).\psi(\underline{u}') = [\Sigma_{\underline{u}} \varphi(\underline{u})].[\Sigma_{\underline{u}'} \psi(\underline{u}')]$ si les uplets de variables $\underline{u}$ et $\underline{u}'$ sont disjoints. Elle vaut donc $\Pi_i (\Sigma_{ui} u_i^{vi}.x_i^{ui}) = \Pi_i (0^{vi}.x_i^0 + 1^{vi}.x_i^1) = \Pi_i (1 + v_i + x_i)$, soit encore 1 si $v_i = x_i$ pour chaque i , et 0 sinon. **Fin**

Remarque 1. La dualité est définie à nombre de variables m fixé ; si une fonction dépend de m variables, sa duale n'est plus la même quand on la considère comme une fonction en une variable muette de plus.

1.2. Questions de complexité

Notre problème est de comparer la complexité de f et celle de f^* .

Par définition la complexité d'une fonction (booléenne) est le nombre d'opérations nécessaires pour la calculer à partir de variables et des constantes 0 et 1 . Comme le choix d'une base (finie) d'opérations ne modifie la complexité qu'à une constante multiplicative près, nous élirons la somme et le produit modulo deux. La complexité d'une fonction booléenne est donc la complexité minimale d'un circuit arithmétique modulo deux qui l'exprime.

Rappelons qu'un circuit arithmétique est la chose bien naturelle suivante : un graphe fini orienté, sans cycles orientés, dont les points sont appelés portes ; il n'a qu'une seule porte qui n'émet pas de flèche, sa sortie ; ses portes qui ne reçoivent pas de flèches sont appelées entrées ; chaque entrée est étiquetée par une variable ou par une constante ; les points qui ne sont pas des entrées reçoivent exactement deux flèches, d'origines distinctes ou confondues, et sont étiquetées par l'addition, ou bien par la multiplication ; la complexité du circuit est le nombre de ces portes d'opération ; sa profondeur est la longueur maximale d'un chemin joignant une de ses entrées à sa sortie. On voit facilement qu'un circuit de complexité c n'a pas plus de $c+1$ entrées. Un

trait important des circuits, c'est qu'on ne limite pas le nombre de flèches que peut émettre une porte, ce qui signifie qu'une fois qu'un calcul intermédiaire est fait, on peut l'utiliser autant de fois qu'on veut. Si vous avez soif de plus de détails, vous pouvez vous abreuver chez [Bürgisser 2000] ou/et chez [Poizat 1995].

Pour éviter des problèmes idiots avec les variables muettes, nous convenons que, dans un circuit qui calcule $f(x_1, \ldots x_m)$, chacune des variables x_i figure à au moins une entrée du circuit : si x_i est absente d'un calcul de f , on peut toujours l'introduire artificiellement en ajoutant $0.x_i$ au résultat. Avec cette convention, une fonction de complexité c n'a pas plus de $c+1$ variables.

On ne confondra pas la complexité de f et celle de son polynôme canonique : un circuit de complexité minimale calculant f calcule aussi un polynôme à coefficients dans F_2 qui a même restriction à F_2 que le polynôme canonique de f ; il n'y a pas de raison pour que ces deux polynômes soient égaux !

Nous noterons P la classe des suites f_n de fonctions booléennes dont la complexité c_n est majorée par un polynôme en n (à coefficients réels) ; le nombre m_n de variables de f_n est donc polynomialement borné, étant majoré par $c_n + 1$. Comme nous n'introduisons pas d'uniformité, cette classe est souvent notée $P/poly$; on pourrait l'uniformiser en exigeant que les fonctions de la suite soient calculées par une suite de circuits produits par une machine de Turing travaillant en temps polynomial ; nous ne le faisons pas, car il n'est pas coutume d'uniformiser les classes de Valiant auxquelles nous allons comparer nos classes booléennes. De toutes façons, l'uniformité n'aurait qu'un effet décoratif, étranger à la nature des problèmes que nous posons.

Nous notons $SP(F_2)$ la classe des suites de fonctions de la forme $f_n(\underline{x}) = \Sigma_{\underline{u}\ \text{booléen}}\ g_n(\underline{x},\underline{u})$, où la suite g_n est dans P . Cette classe, qui correspond aux comptage de la parité, à $\underline{x}$ fixé, du nombre de solutions (booléennes) de l'équation $g_n(\underline{x},\underline{u}) = 1$, est usuellement notée $\#2P$; si nous changeons la notation, c'est que nous la considérerons comme un cas particulier d'une classe définie sur un corps fini arbitraire. En convenant que $\Sigma_{\emptyset}\ f_n(\underline{x}) = f_n(\underline{x})$, ou bien en observant que $f_n(\underline{x}) = \Sigma_{\underline{u}}\ u.f_n(\underline{x})$, on voit que P est inclus dans $SP(F_2)$. Puisque la longueur du uplet $\underline{u}$ peut être un polynôme en n , les sommations portent sur un nombre a priori exponentiel de termes, et on s'attend à ce qu'elles fassent exploser la complexité ; on pense donc qu'il très probable que l'inclusion de P dans $SP(F_2)$ soit stricte, mais, comme d'habitude en l'état présent de la Théorie de la Complexité des calculs, on est incapable de le démontrer.

Il est évident, d'après la définition de f^* , que la classe $SP(F_2)$ est close par dualité : la suite f_n est dans $SP(F_2)$ si et seulement si la suite f_n^* est dans $SP(F_2)$. Plus précisément, on voit que si f s'obtient par sommation devant une fonction de complexité c , f^* s'obtient par sommation devant une fonction de complexité inférieure à $5.c + 4$.

Un terme est un circuit d'architecture arborescente, dans lequel chaque porte n'est autorisée à émettre qu'une seule flèche ; Pt est la classe des suites de fonctions booléennes calculées par une suite de termes de complexité polynomiale ; grâce au lemme de parallélisation de Spira (voir [Spira 1971], [Poizat 1995]), Pt est aussi la classe des suites de fonctions booléennes calculées par une suite de circuits de profondeur logarithmique, c'est-à-dire bornée par $A.\log(n)$ pour une certaine constante réelle A . En effet, d'une part un circuit de profondeur p est de complexité au plus 2^{p+1} ; d'autre part le Lemme de Spira affirme qu'un terme de complexité c peut se remplacer par un terme équivalent, c'est-à-dire calculant la même fonction, de profondeur majorée par $4.\log(2.c + 1)$ (le logarithme est en base deux). Comme ce lemme n'utilise aucune propriété algébrique de la somme et du produit, mais seulement le fait qu'on travaille dans un ensemble fini ($\{0, 1\}$ dans le cas présent), la classe Pt ne dépend pas de la base d'opérations choisie.

Rappelons qu'on ne sait pas séparer P de Pt (c'est-à-dire qu'on ne sait pas montrer que l'inclusion de Pt dans P est stricte). On transforme un circuit en un terme équivalent en répétant les calculs à chaque niveau ; cette manipulation conserve la profondeur, mais a un effet exponentiel sur la complexité ; cela laisse penser que l'inclusion de Pt dans P est stricte, mais un sentiment n'est pas une démonstration !

La classe $SPt(F_2)$ est définie à partir de Pt comme $SP(F_2)$ l'est à partir de P . Le lemme suivant est bien connu, sous cette forme ou sous une autre :

Lemme 2, de réduction parcimonieuse. *Si* $f(\underline{x})$ *est une fonction booléenne calculée par un circuit arithmétique de complexité* c *, on peut trouver une fonction* $g(\underline{x},\underline{u})$ *en* c *variables de plus, calculée par un terme arithmétique de complexité* $4.c$ *, ayant la propriété suivante : si* $f(\underline{x}) = 0$, $g(\underline{x},\underline{u}) = 0$ *pour toute valeur de* $\underline{u}$ *; si* $f(\underline{x}) = 1$, $g(\underline{x},\underline{u}) = 0$ *pour toute les valeurs de* $\underline{u}$ *, à l'exception d'une seule où elle vaut* 1 .

Démonstration. A chaque porte d'entrée du circuit est attachée une variable ou une constante (valant 0 ou 1) ; à chacune de ses portes d'opération π nous associons une nouvelle variable u . Si π est une porte d'addition nous

considérons le polynôme $1 + u + u' + u''$, où u' et u'' sont les variables ou les constantes associées aux portes dont π reçoit ses flêches : ce polynôme vaut 1 si $u = u' + u''$, et 0 sinon ; si π est une porte de multiplication, nous considérons pareillement le polynôme $1 + u + u'.u''$, qui vaut 1 si $u = u'.u''$, et 0 sinon ; $g(\underline{x},\underline{u})$ est le produit de tous ces polynômes par la variable associée à la sortie du circuit. **Fin**

Corollaire 3. $SP(F_2) = SPt(F_2)$.

Démonstration. Si $f(\underline{x},\underline{u})$ est de complexité c , le Lemme 2 donne un terme $g(\underline{x},\underline{u},\underline{v})$ de complexité $4.c$ tel que $f(\underline{x},\underline{u}) = \Sigma_{\underline{v}}\, g(\underline{x},\underline{u},\underline{v})$, et alors $\Sigma_{\underline{u}}\, f(\underline{x},\underline{u}) = \Sigma_{\underline{u},\underline{v}}\, g(\underline{x},\underline{u},\underline{v})$. **Fin**

1.3. Sommations et projections

On dit qu'une fonction est projection d'une autre si la première s'obtient à partir de la seconde en remplaçant les variables de cette dernière par des variables ou des constantes ; comme nos fonctions ont pour domaine $\{0, 1\}^m$, cette notion n'a pour nous de sens que si les constantes de projection sont 0 ou 1 . Une projection ne peut que diminuer la complexité.

On dit qu'un terme de complexité c est une expression si ses $c+1$ entrées sont affectées de variables (pas de constantes) qui sont toutes distinctes ; on voit que tout terme est projection d'une expression. Le résultat qui suit vient de ce que la duale d'une expression est facile à calculer :

Théorème 4. *Si* $f(\underline{x}) = \Sigma_{\underline{u}}\, g(\underline{x},\underline{u})$ *, où* g *est un terme de profondeur* p *,* $f(\underline{x})$ *est projection d'une fonction* $f_1(\underline{y}) = \Sigma_{\underline{u}}\, h(\underline{y},\underline{u})$ *dont la duale se calcule par un terme de profondeur inférieure à* $3.p + 2$.

Démonstration. A partir du terme $g(\underline{x},\underline{u})$, nous en fabriquons un autre à peine plus profond, $h(\underline{y},\underline{u})$, en modifiant les entrées de la manière suivante : nous introduisons autant de variables y_j qu'il y a d'entrées ; si la j-ième entrée est indexée par une constante ε , nous la remplaçons par $\varepsilon\, .y_j$; si la j-ième entrée est indexée par une variable u du uplet $\underline{u}$, nous la remplaçons par $u.y_j$; si la j-ième entrée est indexée par une variable x qui n'est pas une variable de sommation, nous la remplaçons par y_j . Il est clair que $f(\underline{x})$ est une projection de $f_1(\underline{y}) = \Sigma_{\underline{u}}\, h(\underline{y},\underline{u})$.

Si nous considérons $h(\underline{y},\underline{u})$ comme un polynôme en $\underline{y}$ à coefficients dans l'anneau $F_2[\underline{u}]$, il n'a pas de terme constant, et les coefficients de ses

monômes sont très faciles à calculer.

Plus précisément, comme ce polynôme est du premier degré en chacune de ses variables y_j , nous pouvons représenter chaque monôme en $\underline{y}$ par un uplet booléen $\underline{v}$ de même longueur. L'application qui se calcule aisément, c'est celle, à valeurs dans $F_2[\underline{u}]$, qui à $\underline{v}$ associe le coefficient du monôme représenté par $\underline{v}$. Cela se fait par une induction sur la complexité des termes ; considérons une porte d'opération π calculant un sous-terme $t(\underline{z},\underline{u})$ de $h(\underline{y},\underline{u})$, où nous notons $\underline{z}$ le uplet de variables de $\underline{y}$ dont dépend t ; soit $\underline{w}$ le uplet de variables de $\underline{v}$ correspondant à $\underline{z}$; π reçoit ses flêches de portes π' et π'' , calculant respectivement $t'(\underline{z}',\underline{u})$ et $t''(\underline{z}'',\underline{u})$ dont les uplets de variables $\underline{z}'$ et $\underline{z}''$ sont disjoints ; si π est une porte d'addition, la décomposition en monômes de t est une simple juxtaposition de celle de t' et de celle de t'' , tandis que si π est une porte de multiplication, les monômes de t sont les produits de ceux de t' et de ceux de t'' . On remarque que le calcul du coefficient d'un monôme de $h(\underline{y},\underline{u})$ ne demande pas d'additions : ce coefficient, s'il n'est pas nul, est un monôme en $\underline{u}$; par ailleurs, la sommation d'un monôme booléen est une chose très simple, puisqu'elle vaut 1 si toutes les variables sont présentes, et 0 sinon !

Donnons plus de détails ; si $t(\underline{z},\underline{u})$ est un sous-terme de $h(\underline{y},\underline{u})$, calculé à la porte π , le coefficient de son monôme en $\underline{z}$ représenté par $\underline{w}$ est le produit d'un monôme en $\underline{u} = (u_1, \ldots u_n)$ par un coefficient valant 0 ou 1 ; nous lui associons $n+1$ fonctions $\theta_0(\underline{w})$, $\theta_1(\underline{w})$, ... $\theta_n(\underline{w})$; $\theta_0(\underline{w})$ est le coefficient ; pour $i > 0$, $\theta_i(\underline{w}) = 1$ si u_i figure dans le monôme, et 0 s'il n'y figure pas. On remarque que $\theta_0(\underline{0}) = 0$ puisque l'expression de $t(\underline{z},\underline{u})$ comme polynôme en $\underline{z}$ n'a pas de terme constant. Quand $\theta_0(\underline{w}) = 0$, les valeurs de $\theta_1(\underline{w})$, ... $\theta_n(\underline{w})$ importent peu, ce qui nous donnera la liberté de choisir des fonctions $\theta_i(\underline{w})$ faciles à calculer

Comme nous l'avons remarqué, la duale de $f_1(\underline{y}) = \Sigma_{\underline{u}}\, h(\underline{y},\underline{u})$ est le produit $\theta_0(\underline{v}).\theta_1(\underline{v}). \ldots .\theta_n(\underline{v})$ associé au terme final $h(\underline{y},\underline{u})$.

Toutes ces fonctions se calculent facilement, par récurrence sur la complexité du terme $g(\underline{x},\underline{u})$. En effet :

- si π est la j-ième entrée et est associée à la constante ε , $t(y_j) = \varepsilon .y_j$, $\theta_0(v_j) = \varepsilon .v_j$, $\theta_1(v_j) = \ldots = \theta_n(v_j) = 0$

- si π est la j-ième entrée et est associée à la variable u_i , $t(y_j) = u_i.y_j$, $\theta_0(v_j) = \theta_i(v_j) = v_j$, et pour $k \neq 0$, i $\theta_k(v_j) = 0$

- si π est la j-ième entrée et est associée à une variable qui n'est pas une variable de sommation, $t(y_j) = y_j$, $\theta_0(v_j) = v_j$, $\theta_1(v_j) = ... = \theta_n(v_j) = 0$

- si π est une porte d'addition, recevant ses flèches de π' et de π'' , le terme calculé est $t(\underline{z},\underline{u}) = t'(\underline{z}',\underline{u}) + t''(\underline{z}'',\underline{u})$, où $\underline{z}$ est la concaténation de $\underline{z}'$ et de $\underline{z}''$, qui sont disjoints ; nous considérons le produit $\alpha(\underline{w}'')$ des $1+v''$ où v'' parcourt $\underline{w}''$, ainsi que le produit $\beta(\underline{w}')$ des $1+v'$ où v' parcourt $\underline{w}'$; si $\alpha(\underline{w}'') = \beta(\underline{w}') = 0$, on calcule le coefficient d'un monôme qui a des variables à la fois dans $\underline{z}'$ et dans $\underline{z}''$: il est donc nul, et on peut prendre $\theta_0(\underline{w}) = \theta_1(\underline{w}) = ... = \theta_n(\underline{w}) = 0$; si $\alpha(\underline{w}'') = \beta(\underline{w}') = 1$, on doit prendre $\theta_0(\underline{w}) = 0$ puisqu'il n'y a pas de terme constant, tandis que les valeurs de $\theta_1(\underline{w})$, ... $\theta_n(\underline{w})$ peuvent être choisies arbitrairement ; dans le dernier cas, de $\alpha(\underline{w}'')$ et de $\beta(\underline{w}')$, un seul des deux n'est pas nul, et le monôme vient de son côté ; autrement dit, dans tous les cas, on peut prendre pour chaque i , $\theta_i(\underline{w}) = \alpha(\underline{w}'').\theta_i'(\underline{w}') + \beta(\underline{w}').\theta_i''(\underline{w}'')$

- si π est une porte de multiplication, recevant ses flèches de π' et de π'' , le terme calculé est $t(\underline{z},\underline{u}) = t'(\underline{z}',\underline{u}).t''(\underline{z}'',\underline{u})$, où $\underline{z}'$ et $\underline{z}''$ sont disjoints ; le monôme de t représenté par $\underline{w}$ est le produit du monôme de t' représenté par $\underline{w}'$ et de celui de t'' représenté par $\underline{w}''$; autrement dit, $\theta_0(\underline{w}) = \theta_0'(\underline{w}').\theta_0''(\underline{w}'')$, et, pour $i \neq 0$, $\theta_i(\underline{w}) = \theta_i'(\underline{w}') + \theta_i''(\underline{w}'') + \theta_i'(\underline{w}').\theta_i''(\underline{w}'')$.

On montre facilement, par induction sur la profondeur p de la porte π , que le calcul des θ se fait en profondeur $2.p + 1$; en effet, si $p = 0$, π est une entrée, et le calcul se fait en profondeur un ou zéro ; si π est une porte d'addition, comme les uplets $\underline{w}'$ et $\underline{w}''$ sont de longueur au plus 2^{p-1} , le calcul de α et de β se fait en profondeur $p - 1 + 1 = p \leq 2.(p-1) + 1$, qui majore la profondeur des θ' et des θ'' par hypothèse de récurrence, si bien que celui des θ se fait en profondeur $2.(p-1) + 1 + 2 = 2.p + 1$; il en est de même si π est une porte de multiplication.

Quant à la multiplication finale, comme $n \leq 2^p$, elle se fait en profondeur $p+1$, si bien que la profondeur totale est majorée par $3.p + 2$. **Fin**

Remarque 2. Comme les constantes valent 0 ou 1 , on aurait pu traiter les constantes ε comme les variables x ; si nous avons fait ainsi, c'est en prévision du futur. Remarquons par ailleurs que l'application brutale de la formule $\theta = \theta' + \theta'' + \theta'.\theta''$, où θ' et θ'' apparaissent chacun deux fois, fait exploser la taille des termes ; c'est pour cela qu'il faut paralléliser.

Théorème 5. $P = SP(F_2)$ *(c'est-à-dire* $P = \#2P$ *) si et seulement si* P *est close par dualité ;* $Pt = SP(F_2)$ *si et seulement si* Pt *est close par dualité.*

Démonstration. Puisque $SP(F_2) = SPt(F_2)$ est close par dualité, P et Pt le sont aussi quand elles lui sont égales. Réciproquement, si P est clos par dualité, d'après le théorème précédent toute suite de fonctions dans SP s'obtient par projection à partir d'une suite de fonctions dans P, et est donc elle-même dans P ; le même argument vaut pour Pt. **Fin**

2. Fonctions à valeurs dans un anneau

Dans cette deuxième section, nous étendons nos calculs aux fonctions toujours d'arguments booléens, mais à valeur dans un anneau A. *Nous définissons une notion générale de transformation, satisfaisant une condition de type Fubini, dont la dualité est un cas particulier. Nous considérons la classe* $P(A)$ *des suites de fonctions à valeurs dans* A *calculables en un nombre fini d'opérations ; quand* A *est fini, nous montrons que la clôture de* $P(A)$ *par dualité équivaut à sa clôture par sommation. Nous étudions plus précisément le groupe des transformations quand* A *est un corps fini* k *, de caractéristique* p *; nous vérifierons dans la section 4 que la clôture de* $P(k)$ *par sommations (ou transformations) correspond à une hypothèse ouverte bien connue en théorie de la complexité classique, concernant le comptage modulo* p.

2.1. Dualité et transformations

Nous considérons maintenant des fonctions de m variables booléennes à valeur dans un anneau A, c'est-à-dire des applications de $\{0, 1\}^m$ dans A, qui sera le plus souvent l'anneau Z des entiers relatifs, ou bien le corps $F_p = Z/pZ$ des restes des entiers modulo un nombre premier p. Nous persistons à qualifier 0 et 1 de booléens bien que, du point de vue opérationnel, nous les considérons maintenant comme des éléments de A et non pas de $Z/2Z$: $1 + 1$ est égal à 2 et non pas à 0.

Les fonctions en m variables booléennes forment un module libre E_m, de dimension 2^m sur A. De ce module nous considérons principalement deux bases, la base canonique et la base des monômes.

La base canonique de E_1 est composée de la fonction caractéristique $c_0(x)$ de $\{0\}$ et de la fonction caractéristique $c_1(x)$ de $\{1\}$; celle de E_m est composée des fonctions de la forme $c_{\varepsilon_1}(x_1).c_{\varepsilon_2}(x_2). \ldots .c_{\varepsilon_m}(x_m)$, où les ε_i

valent 0 ou 1 : ce sont les fonctions caractéristiques des singletons de $\{0, 1\}^m$. Les coordonnées d'une fonction dans la base canonique, ce sont tout simplement ses valeurs en chaque point.

Les égalités $c_0 = 1 + x$, $c_1 = x$, $1 = c_0 - c_1$, $x = c_1$ montrent que les fonctions 1 et x forment aussi une base de E_1 , et que les fonctions de la forme $x_1^{\varepsilon 1}.x_2^{\varepsilon 2}. \ldots .x_m^{\varepsilon m}$ forment une base de E_m . Autrement dit, toute fonction s'écrit de manière unique comme un polynôme, à coefficient dans A , de degré un ou zéro par rapport à chacune de ses variables ; les coordonnées d'une fonction dans la base des monômes sont les coefficients de son polynôme canonique.

Nous appelons duale f^* de la fonction f la fonction en le même nombre de variables dont f décrit les coefficients du polynôme canonique : $f^*(x_1, \ldots x_m) = \Sigma_{\underline{u} \text{ booléen}} f(u_1,\ldots u_m).x_1^{u1}. \ldots .x_m^{um}$, où l'exponentielle $x^y = 1 - y + xy$ est la fonction qui vaut x si $y = 1$, et qui vaut 1 si $y = 0$. La dualité est un exemple de transformation, notion que nous allons définir.

Nous appelons prétransformation T la donnée pour chaque m d'une application linéaire T_m de E_m dans lui-même satisfaisant aux égalités suivantes, que nous appelons Conditions de Fubini : $T_m(f).T_{m'}(g) = T_{m+m'}(f.g)$, où le m-uplet des variables de f est disjoint du m'-uplet des variables de g . Nous convenons que T_0 vaut l'identité sur l'espace E_0 des fonctions constantes. Puisque la base canonique de $E_{m+m'}$ est formée des produits des éléments de la base canonique de E_m par ceux de la base canonique de $E_{m'}$ (il en est de même pour les bases de monômes !), une prétransformation T est entièrement déterminée par T_1 , qui est une application linéaire quelconque de E_1 dans lui-même. La composée de deux prétransformations en est une. Nous appelons transformation une prétransformation bijective, c'est-à-dire inversible. Le groupe des transformations est isomorphe à $GL_2(A)$.

Nous munissons les fonctions de deux variables du produit suivant, que nous qualifions de produit matriciel : $(f \times g)(x,y) = \Sigma_{t \in \{0,1\}} f(x,t).g(t,y)$; il correspond au produit des matrices à deux lignes et deux colonnes, qui ont respectivement $f(i,j)$ et $g(i,j)$ à l'intersection de leur i-ième ligne et de leur j-ième colonne.

De même, si h est une fonction d'une variable, soit encore un point de E_1 , l'écriture $\Sigma_{t \in \{0,1\}} f(x,t).h(t)$ correspond à la multiplication de la colonne des coordonnées de h par la matrice associée à f .

Comme toute application linéaire se représente par une matrice, à toute prétransformation T est associée univoquement une fonction $\nu(y,x)$, que nous appelons son noyau, telle que $T_1(f)(x) = \Sigma_{t \in \{0,1\}} f(t).\nu(t,x)$; il en résulte que $T_m(f)(x_1, \ldots x_m) = \Sigma_{t \text{ booléen}} f(t_1,t_2,\ldots t_m).\nu(t_1,x_1). \ldots .\nu(t_m,x_m)$ pour chaque entier m, puisque cette égalité est vérifiée, grâce au Théorème de Fubini, lorsque f est dans la base canonique de E_m. La composition des prétransformations correspond à la multiplication matricielle de leurs noyaux.

Par exemple, la transformation identité a pour noyau la fonction $c_y(x) = 1 - (x-y)^2 = 1 - x - y + 2.xy$, qui vaut 1 si $x = y$, et 0 sinon. Si on multiplie ce noyau par une constante λ dans A, on obtient la prétransformation centrale T_λ qui multiplie par λ^m les fonctions de m variables ; c'est une transformation si et seulement si λ est inversible. Plus généralement, on peut multiplier ce noyau par une fonction $\lambda(x) = a.x + b = b.c_0(x) + (a+b).c_1(x)$, et obtenir la prétransformation qui multiplie par $\lambda(x_1). \ldots .\lambda(x_m)$ les fonctions de m variables ; dans la base canonique, $T_{\lambda(x),1}$ est représentée par une matrice diagonale ; elle est inversible si b et $a+b$ le sont.

Ces dernières transformations n'ont qu'un effet bénin sur la complexité des fonctions, qu'elles ne peuvent qu'au plus multiplier par une constante. Il en est de même de T_σ, associée à la permutation σ des deux valeurs 0 et 1, qui à $f(x_1, \ldots x_m)$ associe $f(\sigma(x_1), \ldots \sigma(x_m))$; son noyau s'écrit $(x - y)^2 = x + y - 2.xy$.

En conséquence, nous appellerons transformations bénignes celle qu'on obtient par composition des $T_{\lambda(x)}$ et de T_σ ; dans la base canonique de E_1, elles sont représentées par les matrices diagonales ou antidiagonales inversibles. On voit facilement que ce sont les transformations dont le noyau a la propriété suivante : pour chaque x, il y a un seul y tel que $\nu(y,x) \neq 0$, si bien que la sommation qui les définit se réduit à un seul terme.

Il est par contre probable que la dualité T^*, dont le noyau est l'exponentielle x^y, a un pouvoir explosif ; il devrait en être de même de son inverse $T°$, qui à f associe la fonction $f°$ qui calcule les coefficients de son polynôme canonique. On inverse matriciellement l'exponentielle par la formule $\Sigma_{t \in \{0,1\}} x^t.(a + b.t + c.y + d.ty) = 1 - x - y + 2.xy$, ce qui donne, après identification, $1 - 2.x - y + 3.xy$ comme noyau de $T°$.

On remarque que, dans la base canonique, T^*_1 a pour matrice $(1 \;\; 0 ; 1 \;\; 1)$

(nous ne considérerons que des matrices à deux lignes et deux colonnes, dont nous écrivons les deux lignes séparées par un point-virgule) ; quand $A = (Z/nZ)$, la transformation T^* est donc d'ordre n , et $T°$ s'obtient en l'itérant $n-1$ fois . Quel que soit l'anneau A , T^* et $T°$ sont conjuguées par l'involution bénigne de matrice $(1\ \ 0 ; 0\ \ -1)$, associée à la fonction $\lambda(x) = 1 - 2.x$.

La célèbre prétransformation de Fourier a pour noyau $(-1)^{x.y}$, où le caractère $(-1)^x$ vaut $1 - 2.x$; $(-1)^x.(-1)^y = (-1)^{x \oplus y}$, où $x \oplus y = x + y - 2.xy$ note la somme modulo deux. Le carré matriciel de ce noyau vaut $\Sigma_{t \in \{0,1\}} (-1)^{x.t}.(-1)^{t.y} = 1 + (1 - 2.x)(1 - 2.y) = 2.(1 - x - y + 2.xy)$, qui est le noyau de T_2 . Autrement dit, le carré de la prétransformation de Fourrier multiplie par 2^m les fonctions de m variables ; c'est une transformation si et seulement si 2 est inversible dans A , et dans ce cas toute fonction s'écrit de manière unique comme combinaison linéaire des caractères $(-1)^{\varepsilon_{1.x1}} . (-1)^{\varepsilon_{2.x2}} . \ldots . (-1)^{\varepsilon_{m.xm}}$, la transformée de Fourier inverse donnant justement les coordonnées de la fonction dans cette base des caractères.

2.2. Classes de complexité

Parlons maintenant de complexité. Comme la fonction f s'écrit comme un polynôme, nous pouvons définir sa complexité comme étant le nombre minimal d'additions et de multiplications qu'il faut pour l'obtenir à partir de variables et de constantes. Il y a deux façons d'envisager le rôle des constantes, qui se confondent quand l'anneau A est fini.

La première façon est généralement adoptée dans l'étude des classes de complexité à la Valiant, bien qu'elle ne se prête pas à leur uniformisation. Elle consiste à se donner gratuitement toutes les constantes de A , chaque fonction constante étant de complexité nulle.

Nous adoptons ici une autre convention, qui suppose que l'anneau est finiment engendré. Nous fixons un système générateur G fini de A , et tous nos calculs sont faits à partir de constantes dans G et de variables : quand nous aurons besoin d'autres constantes, il faudra les calculer, et intégrer la complexité de leur calcul dans la complexité de la fonction. Changer de système générateur n'augmente les complexités que d'une constante, ce qui nous permet de supposer que G contient toujours la constante -1 . Cette façon plus stricte de tenir compte du rôle des constantes dans le calcul des polynômes a été introduite par [Malod 2003].

Nous adoptons donc, pour les anneaux Z et Z/nZ, la convention suivante : la complexité d'une fonction est la taille minimale d'un circuit arithmétique qui l'exprime, *et qui n'a à ses entrées que des variables ou la constante* -1. Les seules portes d'opérations de nos circuits sont des additions et des multiplications ; les soustractions se font grâce à des multiplication par -1, et il n'y a pas de divisions. Comme nous l'avons déjà dit, la complexité d'une fonction n'est pas celle de son polynôme canonique : le ou les meilleurs circuits qui calculent f calculent aussi des polynômes à coefficients dans A qui ne sont pas nécessairement égaux au polynôme canonique de f : ils ont seulement même restriction que ce dernier à $\{0, 1\}^m$.

La classe $P(A)$ est formée des suites de fonctions de complexité polynomiale, et la classe $SP(A)$ est ce qu'on obtient en mettant des sommations à la sortie des éléments de $P(A)$. On définit de manière analogue les classes $Pt(A)$ et $SPt(A)$. Grâce au Théorème de parallélisation de Muller et Preparata [MP 1976], qui, lui, repose sur les propriétés algébriques de la somme et du produit, les termes de complexité polynomiale sont équivalents aux circuits de profondeur logarithmique.

Lemme 6. *Quel que soit l'anneau fini* A, $SP(A) = SPt(A)$.

Démonstration. Il nous faut un lemme de réduction parcimonieuse. Nous considérons un circuit arithmétique de complexité c calculant une fonction $f(\underline{x})$ de $\{0, 1\}^m$ dans $A = \{a_1, a_2, ... a_q\}$. Nous remplaçons ce calcul dans A par un calcul booléen, en représentant l'élément a_i par le q-uplet booléen dont les coordonnées sont nulles, sauf la i-ème qui vaut 1 ; la complexité du calcul se multiplie par une constante, puisque les portes de constantes et d'opérations sont remplacées par q portes booléennes, et qu'il faut tenir compte des tables d'addition et de multiplication de A.

On obtient ainsi q fonctions booléennes $f_1(\underline{x})$, ... $f_q(\underline{x})$, puis, par réduction parcimonieuse, q termes booléens $h_1(\underline{x},\underline{u})$, ... , $h_q(\underline{x},\underline{u})$ qui ont la propriété suivante : à $\underline{x}$ fixé, tous les $h_j(\underline{x},\underline{u})$ sont nuls, à l'exception d'une seule valeur de $\underline{u}$ pour laquelle $h_i(\underline{x},\underline{u}) = 1$, où $f(\underline{x}) = a_i$. Nous parallélisons ces termes en profondeur logarithmique, puis nous simulons l'addition et la multiplication de F_2 par celles de A restreintes à $\{0, 1\}$ grâce aux fonctions $x + y - 2.xy$ et xy, ce qui ne fait que multiplier les profondeurs par trois ; nous obtenons ainsi des termes $g_1(\underline{x},\underline{u})$, ... $g_q(\underline{x},\underline{u})$ au sens de A, de complexité polynomiale en c. Le terme $g(\underline{x},\underline{u}) = a_1.g_1(\underline{x},\underline{u}) + ... + a_q.g_q(\underline{x},\underline{u})$, à $\underline{x}$ booléen fixé, est toujours nul, sauf pour un $\underline{u}$ booléen où il calcule $f(\underline{x})$. **Fin**

Remarque 3. Grâce à la simulation de l'addition modulo deux, toute suite appartenant à $P = P(F_2)$ définit aussi un élément de $P(A)$, quel que soit l'anneau A. Par ailleurs, si cet anneau a q éléments, nous avons vu qu'un élément de $P(A)$ se représente par un q-uplet d'éléments de P.

Il ne saurait être question de réduction parcimonieuse dans le cas de Z, puisqu'un circuit peut produire un nombre doublement exponentiel en sa taille, tandis que celui calculé par un terme est simplement exponentiel.

Voici maintenant la version définitive du Théorème 4 :

Théorème 7. *On considère un anneau* A *arbitraire, et une fonction de la forme* $f(\underline{x}) = \Sigma_{\underline{u}}\, g(\underline{x},\underline{u})$ *, où* g *est un terme de profondeur* p *; alors* $f(\underline{x})$ *est projection d'une fonction* $f_1(\underline{y}) = \Sigma_{\underline{u}}\, h(\underline{y},\underline{u})$ *dont la duale inverse se calcule par un terme de profondeur inférieure à* $4.p + 4$.

Démonstration. Il faut modifier la démonstration du Théorème 4 en remplaçant les expressions $1 + v_i$ par $1 - v_i = (-1).(v_i + (-1))$, et les expressions $\theta_i'(\underline{w}') + \theta_i''(\underline{w}'') + \theta_i'(\underline{w}').\theta_i''(\underline{w}'')$ par $\theta_i'(\underline{w}') + \theta_i''(\underline{w}'') - \theta_i'(\underline{w}').\theta_i''(\underline{w}'') = \theta_i'(\underline{w}') + \theta_i''(\underline{w}'') + (-1).\theta_i'(\underline{w}').\theta_i''(\underline{w}'')$; quant au produit final, comme la sommation d'un monôme booléen vaut $2^{n'}$, où n' est le nombre de variables absentes, il devient $\theta_0(\underline{v}).(2 - \theta_1(\underline{v})). \ldots .(2 - \theta_n(\underline{v})) = (-1)^n.\theta_0(\underline{v}).(\theta_1(\underline{v}) + (-1) + (-1)). \ldots .(\theta_n(\underline{v}) + (-1) + (-1))$. On voit facilement que le calcul se fait en la profondeur indiquée. **Fin**

Remarque 4. Dans le calcul de la duale inverse de $f_1(\underline{y})$, nous n'avons utilisé que la constante -1 en plus des constantes qui apparaisssent dans le calcul de $f(\underline{y})$. On ne confondra pas le rôle des constantes figurant aux entrées des circuits, qui sont des éléments de A, avec les constantes de projection, qui sont nécessairement 0 ou 1 ; en effet, nous ne calculons pas des polynômes, mais des fonctions d'arguments booléens, et nous considérons comme équivalents deux circuits qui obtiennent le même résultat quand on donne des valeurs booléennes aux variables attachées à leurs entrées.

2.3. Effet des transformations sur la complexité

Théorème 8. *(i) Quel que soit l'anneau* A *,* $SPt(A) = Pt(A)$ *si et seulement si* $Pt(A)$ *est close par transformation, si et seulement si* $Pt(A)$ *est close par dualité, si et seulement si* $Pt(A)$ *est close par dualité inverse.*

(ii) Si A *est un anneau fini,* $SP(A) = P(A)$ *si et seulement si* $P(A)$ *est*

close par transformation, si et seulement si P(A) *est close par dualité, si et seulement si* P(A) *est close par dualité inverse.*

Démonstration. Le calcul d'une transformée à partir de la formule du noyau montre que SPt(A) est close par transformation : il en sera de même de Pt(A) si elle lui est égale. Réciproquement, si Pt(A) est close par dualité, le Théorème 7 montre que SPt(A) = Pt(A) ; par ailleurs, Pt(A) est close par dualité si et seulement si elle l'est par dualité inverse, puisque ces deux transformations sont conjuguées par une transformation bénigne, qui ne fait guère plus que doubler la profondeur des circuits. On démontre semblablement la deuxième partie, sachant que, dans le cas où A est fini, SP(A) = SPt(A) . **Fin**

Remarque 5. Comme le Théorème 7 concerne la classe SPt(A) , et non la classe SP(A) , la réduction parcimonieuse, c'est-à-dire la finitude de A , est essentielle pour la démonstration de la deuxième partie de ce Théorème 8.

Si A est un corps fini k , il y a une explication simple au rôle prépondérant donné à la dualité par le Théorème 8 : elle engendre avec les transformations bénignes le groupes des transformations tout entier. C'est une conséquence du Théorème 9, et de la phrase qui suit le Théorème 10.

Théorème 9. *Si* k *est un corps fini différent de* $F_3 = Z/3Z$ *et de* $F_5 = Z/5Z$ *, le groupe des transformations bénignes est maximal dans* $GL_2(k)$: *toute transformation non-bénigne engendre avec les transformations bénignes le groupe de toutes les transformation.*

Démonstration. Nous travaillons dans la base canonique, où le groupe B des transformation bénignes se représente par les matrices inversibles diagonales ou antidiagonales : ce sont les matrices inversibles qui ont deux coefficients nuls.

Il est utile de déterminer la décomposition de $GL_2(k)$ en classe doubles BmB , puisque deux matrices m et m' dans la même classe double engendrent avec B le même sous-groupe de $GL_2(k)$. Les opérations qui préservent une classe double sont l'échange des lignes, l'échange des colonnes, la multiplication d'une ligne par un coefficient non nul, et la multiplication d'une colonne par un coefficient non nul ; on en déduit que chaque matrice inversible non-bénigne se trouve dans la classe double d'une matrice $(1\ \ 1 ; 1\ \ a)$, $a \neq 1$.

Si $a = 0$, la classe double est formée des matrices qui ont un et un seul coefficient nul, et contient la dualité $(1\ \ 0 ; 1\ \ 1)$. Comme toute matrice

inversible sans coefficients nuls est produit d'une matrice triangulaire supérieure et d'une matrice triangulaire inférieure, chacun de ses points engendre $GL_2(k)$ avec B .

Si a est différent de 1 , de 0 et de -1 , on peut trouver x et y non nuls tel que (1 1 ; 1 a).(x 0 ; 0 y).(1 1 ; 1 a) ne contienne qu'un seul zéro, et on est ramené au cas précédent.

Reste à traiter le cas où a = - 1 , qui ne peut se produire en caractéristique deux ; les matrices de la classe double sont alors les matrices non-bénignes de permanent nul, comme celle de la transformation de Fourier. Le produit ci-dessus, lorsqu'on fait varier x et y , parcourt l'ensemble des matrices symétriques. Les conditions pour qu'une telle matrice (u v ; v u) ne soit pas dans cette classe double sont $u \neq 0$, $v \neq 0$, $u^2 \neq v^2$, $u^2 \neq -v^2$; elles sont réalisables si on peut trouver un élément non nul qui ne soit pas racine quatrième de l'unité, ce qui est possible sauf dans les deux cas exceptionnels signalés dans l'énoncé. **Fin**

Théorème 10. *Si* k *est un corps fini différent de* F_3 *et de* F_5 *,* SP(k) = P(k) *si et seulement si* P(k) *est close par une quelconque transformation non-bénigne, et* SP(k) = Pt(k) *si et seulement si* Pt(k) *est close par une quelconque transformation non-bénigne.*

Démonstration. Si k est différent de F_3 et de F_5 , et si T est une transformation non-bénigne, la dualité s'exprime comme un produit d'au plus cinq copies de T et de transformations bénignes, si bien que la clôture de P(k) par T implique la clôture de P(k) par dualité ; il en est de même pour la classe Pt(k) . **Fin**

Dans les deux cas exceptionnels, on vérifie aisément que les matrices (1 1 ; 1 -1).(0 x ; y 0).(1 1 ; 1 -1) restent aussi dans la classe double, si bien qu'on obtient un, et un seul, groupe intermédiaire entre B et $GL_2(k)$, celui qui est engendré par les transformations bénignes et la transformation de Fourier. Cela conduit à la question suivante :

Problème. Si $k = F_3$ ou $k = F_5$, est-ce-que la transformée de Fourier a un effet sur la complexité des fonctions vraiment moins désastreux que celui de la dualité ?

Par contraste, dans le cas où l'anneau A est celui des entiers relatifs, dont les seuls inversibles sont 1 et -1 , il n'y a que huit transformations bénignes, et, par exemple, on ne voit pas comment comparer le pouvoir de nuisance de la dualité à celui de son carré, bien qu'elle-même et son carré

soient conjugués par une transformation bénigne dans l'anneau $Z[1/2]$.

2.4. Changement de corps de base

Théorème 11. *Si* k *et* k' *sont deux corps finis de même caractéristique,* $P(k) = SP(k)$ *si et seulement si* $P(k') = SP(k')$ *, et* $Pt(k) = SP(k)$ *si et seulement si* $Pt(k') = SP(k')$.

Démonstration. Nous supposons que k , de caractéristique p , a $q = p^d$ éléments, et nous considérons une suite $f_n(\underline{x}) = \Sigma_{\underline{u}\ \text{booléen}}\ g_n(\underline{x},\underline{u})$ de fonctions dans $SP(Z/pZ)$, où la suite g_n est calculée par une suite de circuits arithmétiques de taille polynomiale à paramètres dans $F_p = Z/pZ$: comme ce dernier est un sous-corps de k , cette suite est dans $P(k)$, et la suite f_n est dans $SP(k)$. Si $SP(k) = P(k)$, cette suite est aussi dans $P(k)$, et calculable par des petits circuits arithmétiques à paramètres dans k . Comme k est un espace vectoriel de dimension d sur F_p , on représente ses éléments par des d-uplets à coordonnées dans F_p pour remplacer ce calcul d'une fonction à valeur dans k par celui de d fonctions à valeur dans F_p , la première de ces fonctions coordonnées étant f_n . Autrement dit, $SP(k) = P(k)$ implique que $SP(F_p) = P(F_p)$.

Pour la réciproque, nous considérons une suite $\varphi_n(\underline{x}) = \Sigma_{\underline{u}\ \text{booléen}}\ \gamma_n(\underline{x},\underline{u})$ dans $SP(k)$, et nous remplaçons comme ci-dessus le calcul de γ_n par celui d'un d-uplet de suites de fonctions dans $P(Z/pZ)$; comme l'addition dans k correspond à l'addition coordonnée par coordonnée des d-uplets représentant ses éléments, le calcul de φ_n est alors simulé par celui de d suites dans $SP(Z/pZ)$; si ces dernières sont dans $P(Z/pZ)$, φ_n est dans $P(k)$.

La même démonstration vaut pour la classe Pt grâce à la parallélisation. **Fin**

Nous sommes donc amenés à considérer les différentes hypothèses $P(Z/pZ) = SP(Z/pZ)$ lorsque le nombre p varie. Il est clair que, par un simple passage au quotient, l'égalité $P(Z) = SP(Z)$ implique $P(Z/pZ) = SP(Z/pZ)$, puisqu'un circuit artithmétique modulo p peut être aussi considéré comme un circuit arithmétique modulo 0 .

3. Intermède : variables parcourant un corps fini

Dans cette courte et schématique section, nous considérons un corps fini k *, et des fonctions dont les variables comme les valeurs parcourent* k *; les sommations sont aussi étendues à* k *tout entier. Nous n'y trouverons pas de classes de complexité vraiment nouvelles, mais seulement des familles plus riches de transformations.*

Si k est un corps fini de caractéristique p , à $q = p^d$ éléments, il est tentant d'introduire des variables à valeurs dans k , et d'étendre les sommations à k tout entier.

Comme toute fonction de k^m dans k s'écrit de manière unique comme un polynôme de degré au plus $q-1$ en chacune de ses variables, on pourra définir leur complexité, et introduire la classe $P'(k)$ des suites de fonctions de complexité polynomiale, et $SP'(k)$ des suites de fonctions de la forme $f_n(\underline{x}) = \Sigma_{\underline{u} \in k} g_n(\underline{x},\underline{u})$, où g_n est dans $P'(k)$; on introduira aussi les classes $P't(k)$ et $SP't(k)$, cette dernière étant égale à $SP'(k)$ par réduction parcimonieuse.

On définira comme auparavant les transformations et leurs noyaux.

Il est facile de voir que les sommations sur $\{0, 1\}$ simulent en douceur celles sur k tout entier, et réciproquement ; par ailleurs, une représentation des éléments de k par des uplets booléens permet de montrer que les improbables égalités $P(k) = SP(k)$ et $P'(k) = SP'(k)$ sont équivalentes. Pour l'équivalence de $Pt(k) = SPt(k)$ et $SP't(k) = P't(k)$, on passera par la parallélisation des termes.

Nous ne gagnons donc rien en matière de classes de complexité, mais nous obtenons un groupe plus large de transformations, isomorphe à $GL_q(k)$, ce qui nous permet d'offrir à la sagacité des chercheurs une famille a priori plus riche de degrés de nocivité.

Les transformations bénignes sont celles dont la matrice, dans la base canonique, a exactement un coefficient non nul dans chaque ligne et dans chaque colonne. Pour définir l'exponentielle, il faut choisir une énumération a_0 , ... a_{q-1} de k , noter $c_i(y)$ la fonction caractéristique de $\{a_i\}$, et poser $x^y = c_0(y) + c_1(y).x + ... + c_i(y).x^i + ... + c_{q-1}(y).x^{q-1}$. Le choix de cette énumération introduit une part d'arbitraire dans la définition de la dualité associée, mais deux de ces dualités sont conjuguées par une transformation bénigne T_σ , où σ est une permutation de k .

Si $k = Z/pZ$, il est naturel de choisir pour a_i le reste de i modulo p , et on

aura $x^i.x^j = x^{i\oplus j}$ si $i + j < p$, et sinon $x^i.x^j = x^{i\oplus j\oplus 1}$, où $x\oplus y$ désigne la somme modulo p : puisque l'ordre du groupe additif de k est premier avec celui de son groupe multiplicatif, il n'est pas question de caractère, ni de transformée de Fourier.

On démontre de même que $SP'(k) = P'(k)$ équivaut à la clôture de $P'(k)$ par dualité, et que $SPt'(k) = Pt'(k)$ équivaut à la clôture de $Pt'(k)$ par dualité.

4. Sommations en chiffres et sommations en entiers

Nous travaillons maintenant en caractéristique nulle, c'est-à-dire avec des entiers ; nous comparons deux manières de faire, celle où les calculs se font directement dans l'anneau Z , et celles où les calculs se font en chiffres. Nous examinons aussi l'incidence des calculs en caractéristique nulle sur les calculs modulo p .

La comparaison des différentes hypothèses $P(Z/pZ) = SP(Z/pZ)$, que l'on espère, bien sûr, toutes fausses, est en fait un problème à la fois désespérant et bien connu, que d'habitude on formule autrement. On introduit la classe $\#pP$, également connue sous le nom de $Mod(p)\ P$, formée des suites $f_n(\underline{x})$ de fonctions booléennes obtenues à partir d'une suite $g_n(\underline{x},\underline{u})$ dans P de la manière suivante : $f_n(\underline{x}) = 0$ si le nombre de solutions $\underline{u}$ de l'équation $g_n(\underline{x},\underline{u}) = 1$ est divisible par p , $f_n(\underline{x}) = 1$ sinon ; on peut d'ailleurs définir cette classe même si p n'est pas un nombre premier. Après avoir observé que savoir si le cardinal d'un ensemble est divisible par p permet en fait de trouver son reste modulo p , en lui ajoutant un point, puis deux, ... puis $p-1$, il est facile de voir que l'égalité $\#pP = P$, qui concerne des classes de fonctions booléennes, équivaut à l'égalité $SP(Z/pZ) = P(Z/pZ)$, qui concerne des classes de fonctions à valeur dans F_p : ce n'est qu'un exercice de codage binaire, que nous abandonnons à notre lecteur.

Comme nous l'avons observé dans la Remarque 3, la simulation de la somme et du produit de $F_2 = Z/2Z$ par $x + y - 2.xy$ et xy donne une grande solidité aux classes $P(Z/pZ)$: elles se réduisent à la classe $P = P(F_2)$ grâce à un codage booléen évident. Par contre, il n'existe aucune méthode de simulation des *sommations* modulo deux par des sommations dans une autre caractéristique, ce qui fragilise les classes SP : il n'y a aucune raison évidente, ni aucune raison connue, pour que les classes $\#pP$ soient égales.

Pour définir la classe $\#P$, introduite dans [Valiant 1979], qui est plus puissante que toutes les $\#pP$, le mieux est d'introduire des multifonctions et

des multicircuits. Une multifonction booléenne est par définition une application de $\{0, 1\}^m$ dans $\{0, 1\}^n$, où m et n sont deux entiers, soit encore la donnée d'un n-uplet de fonctions booléennes en m variables. Nous étendons la définition de la classe P aux multifonctions : ce sont les suites de multifonctions booléennes calculées par des multicircuits - définis comme les circuits, mais pouvant posséder plusieurs sorties - arithmétiques modulo deux de complexité polynomiale. En introduisant n nouvelles variables $y_1, \ldots y_n$, on peut assimiler une multifonction à n sorties à la fonction de complexité voisine $\varphi(\underline{x},\underline{y}) = y_1.f_1(\underline{x}) + \ldots + y_n.f_n(\underline{x})$, si bien que les multifonctions n'apportent rien de vraiment nouveau du point de vue algorithmique

La classe $\#P$ est formée des suites $f_n(\underline{x})$ de multifonctions donnant les chiffres en base deux du nombre de solutions booléennes de l'équation $g_n(\underline{x},\underline{u}) = 1$, où la suite de fonctions g_n est dans P.

La valeur d'une multifonction $g(\underline{x},\underline{u})$ représente si l'on veut un nombre entier donné en chiffres, ce qui permet de considérer la multifonction $\Sigma\, ch_{\underline{u}}\, g(\underline{x},\underline{u})$ de la somme de ces nombres donnée en chiffres. On voit que l'égalité $\#P = P$ signifie que la classe P est close pour la sommation en chiffres : il suffit de compter le nombre de fois où chacun des chiffres vaut 1, puis de faire l'addition.

Cette improbable égalité $\#P = P$ implique bien sûr chacune des $\#pP = P$.

Supposons, en vue d'une tentative réciproque désespérée, que $\#2P = P$; soit $N(\underline{x})$ le nombre de solutions de l'équation $g_n(\underline{x},\underline{u}) = 1$; le premier chiffre de $N(\underline{x})$ s'obtient en additionnant modulo 2 les $g_n(\underline{x},\underline{u})$; pour avoir le deuxième chiffre, on divise par deux la famille des $\underline{u}$ tels que $g_n(\underline{x},\underline{u}) = 1$ en considérant la fonction $(\Sigma_{\underline{v}}\, \varphi(\underline{u},\underline{v}).\, g_n(\underline{x},\underline{v}))\, .\, g_n(\underline{x},\underline{u})$, où $\varphi(\underline{u},\underline{v})$ est la fonction facilement calculable qui vaut 1 si le nombre entier représenté par $\underline{v}$ est inférieur à celui représenté par $\underline{u}$, et 0 sinon ; en sommant modulo deux cette fonction, on obtient le deuxième chiffre de $N(\underline{x})$. Si donc $\#2P = P$, nous obtenons en temps polynomial les deux premiers chiffres de $N(\underline{x})$; et si nous itérons le procédé un nombre a de fois fixé, on obtient les a premiers chiffres de $N(\underline{x})$ au prix de $2.a + 1$ sommations modulo deux, et d'une petite augmentation du nombre d'opérations arithmétiques. Nous voyons donc que $\#2P = P$ implique $\#2^aP = P$. On voit de même, en développant les entiers en base n, que $\#nP = P$ implique $\#n^aP = P$.

Si on ajoute à cela que, si n divise m et si on sait compter modulo m, alors on sait compter modulo n, et aussi que si m et n sont premier entre eux, savoir compter modulo $m.n$ revient à savoir compter modulo m et modulo

n , on parvient au résultat suivant, que nous énonçons comme un fait et non comme un théorème car nous n'osons croire qu'il soit original, bien que nous n'en connaissions pas de référence :

Fait. $\#nP = P$ *si et seulement si* $\#p_iP = P$ *pour chaque diviseur premier* p_i *de* n *; autrement dit* $SP(Z/nZ) = P(Z/nZ)$ *si et seulement si* $SP(Z/p_iZ) = P(Z/p_iZ)$ *pour chaque diviseur premier* p_i *de* n .

Une classe de complexité célèbre entre toutes est la classe NP , formée des suites $f_n(\underline{x})$ de fonctions booléennes obtenues de la manière suivante à partir d'une suite $g_n(\underline{x},\underline{u})$ dans P : $f_n(\underline{x}) = 1$ si l'équation $g_n(\underline{x},\underline{u}) = 1$ a des solutions (booléennes !), $f_n(\underline{x}) = 0$ sinon. Si on sait compter le nombre de solutions d'une équation, on sait encore mieux déterminer s'il en existe ou pas, si bien que $\#P = P$ implique $NP = P$; on ne connait pas de rapports entre les hypothèses $\#pP = P$ et $NP = P$.

Le théorème suivant et dernier établit une équivalence entre une hypothèse ($\#P = P$) qui concerne des classes de complexité booléennes, et une hypothèse portant sur des calculs dans l'anneau Z .

Théorème 12. $\#P = P$ *équivaut à* $SPt(Z) \subseteq P(Z)$.

Démonstration. Supposons que $\#P = P$, et considérons une suite de termes $t_n(\underline{x},\underline{u})$ dans $Pt(Z)$; comme un terme ne produit que des entiers simplement exponentiels, on peut représenter le calcul qu'il fait par un multicircuit booléen de complexité polynomiale qui fait ce calcul en chiffres. Par hypothèse, on peut faire une sommation en chiffres sans exploser, si bien que les chiffres de $\Sigma_{\underline{u}\ \text{booléen}}\ t_n(\underline{x},\underline{u})$ sont donnés par un multicircuit booléen $f_n(\underline{x})$ de complexité polynomiale, lequel se simule par un multicircuit $g_n(\underline{x})$ en caractéristique nulle ; il ne reste plus qu'à multiplier ces chiffres par les puissances de 2 qui leur sont affectées, et à les sommer, ce qui fait un calcul dans $P(Z)$.

Supposons réciproquement que $SPt(Z) \subseteq P(Z)$, et considérons une suite de fonctions booléennes $f_n(\underline{x},\underline{u})$ dans P ; pour ce qui est de compter le nombre de solutions de $f_n(\underline{x},\underline{u}) = 1$, la réduction parcimonieuse nous ramène au cas où elles sont calculées par des termes de profondeur logarithmique. Soit $t_n(\underline{x},\underline{u})$ leur simulation dans $Pt(Z)$; par hypothèse, $\Sigma_{\underline{u}\ \text{booléen}}\ t_n(\underline{x},\underline{u})$ se calcule par une suite de circuits $g_n(\underline{x})$ dans $P(Z)$; ces circuits peuvent produire des nombres doublement exponentiels, mais comme le résultat est simplement exponentiel, il suffit pour l'obtenir de reproduire leurs calculs modulo 2^{c_n} , où c_n est polynomialement borné, ce qui permet un calcul en chiffres, c'est-à-dire booléen, de complexité polynomiale. **Fin**

La seule façon structurelle de borner polynomialement le degré des polynômes et la taille de leur coefficients, c'est d'éviter les cascades incontrolées de multiplications en contraignant les circuits arithmétiques à être multiplicativement disjoints, comme l'a montré [Malod 2003] (voir aussi [MP 2006] et [MP 2008]). Dans un circuit multiplicativement disjoint, chaque porte de multiplication reçoit ses deux flèches de sous-circuits disjoints, alors que dans un terme les portes d'addition ont aussi cette propriété ; on ne sait pas séparer $Pmd(Z)$ de $Pt(Z)$; en présence de sommations, ces circuits multiplicativement disjoints sont équivalents aux termes, ce qui signifie que $SPmd(Z) = SPt(Z)$, si bien que l'hypothèse $SPmd(Z) = Pmd(Z)$ implique l'inclusion de $SPt(Z)$ dans $P(Z)$. On montre, de la même façon que le Théorème 8, que $SPmd(Z) = Pmd(Z)$ équivaut à la clôture de $Pmd(Z)$ par dualité.

Les hypothèses $SP(Z) = P(Z)$ et $SPmd(Z) = Pmd(Z)$ paraissent aussi peu vraisemblables l'une que l'autre, et pourtant leurs rapports sont enveloppés de mystère ; en particulier il n'est pas clair que la première implique la seconde, (voir [Malod 2003], [Poizat 2008]).

Remarque 6. Si une suite de fonctions booléennes est dans $Pmd(Z)$, un simple passage au quotient modulo deux la met dans la classe Pmd des suites de fonctions booléennes calculées par des circuits arithmétiques modulo deux multiplicativement disjoints et de complexité polynomialement bornée ; la réciproque n'a rien de certain, car la simulation de l'addition modulo deux utilise une multiplication, et ne conserve pas la notion de circuit multiplicativement disjoint. On n'a pas ce problème avec les termes, grâce à la parallélisation.

Remarque 7. Malgré la fragilité de la classe Pmd , le Théorème 11 vaut pour l'égalité $SPmd(k) = Pmd(k)$, puisque la simulation de la multiplication des éléments de k' par des d-uplets d'éléments de k préserve la notion de circuit multiplicativement disjoint. Comme $SPmd(A) = SPt(A)$ quel que soit l'anneau A , le Théorème 8 (i) est valable aussi pour la classe $Pmd(A)$; le Théorème 10 s'adapte pareillement, une fois qu'on a remarqué que les transformations bénignes n'ajoutent aux circuits qu'un nombre modéré de multiplications disjointes.

Nous contemplons finalement le paysage suivant : $SP(Z) = P(Z)$ comme $SPmd(Z) = Pmd(Z)$ impliquent $SPt(Z) \subseteq P(Z)$, qui équivaut à $\#P = P$, laquelle implique à son tour chacune des $\#pP = P$, ainsi que $NP = P$.

5. Comparaison avec les résultats de Malod sur les classes de Valiant

En conclusion, nous comparons nos résultats à leur source d'inspiration, les travaux de Guillaume Malod sur les rapports entre la complexité d'un polynôme et celle de la fonction qui exprime ses coefficients.

En fait, $P(Z)$, $SP(Z)$, $Pt(Z)$, $Pmd(Z)$, $SPmd(Z) = SPt(Z)$ sont les parties numériques respectives de $VPdl$, $VNPdl$, VPt , $VPmd$, $VNPmd = VNPt$, selon les notations de [Poizat 2008] ; ce sont des classes de complexité formées de suites de polynômes à coefficients entiers relatifs. En conséquence les égalités $VNPdl = VPdl$, $VNPmd = VPmd$ et $VNPt = VPt$, qui sont au coeur de l'étude des classes de complexité selon Valiant, impliquent leurs restrictions respectives aux fonctions numériques $SP(Z) = P(Z)$, $SPmd(Z) = Pmd(Z)$ et $SPt(Z) = Pt(Z)$.

La clôture des classes $SP(Z/pZ) = SPt(Z/pZ)$ par dualité peut être vue comme une conséquence de la clôture de la classe $VNPmd$ par fonction-coefficient, établie dans [Malod 2003] ; c'est une méthode compliquée pour établir un fait évident, puiqu'il faut faire des regroupements de monômes pour obtenir les coefficients du polynôme canonique, mais je la mentionne ici parce que la considération de cette clôture a été le point de départ du présent article.

Une autre de ses sources d'inspiration, c'est le résultat de Malod affirmant que $VNPmd = VPmd$ équivaut à l'hypothèse suivante : une suite de polynômes est dans $VPmd$ si et seulement si la suite de ses fonctions-coefficient est dans $VPmd$, c'est-à-dire dans $Pmd(Z)$.

Pour le démontrer, Malod utilise un analogue du célèbre résultat de Leslie Valiant sur la complétude du permanent. Rappelons qu'on appelle permanent de taille n le polynôme à coefficients entiers, en n^2 variables, $Per_n(..., x_{i,j}, ...) = \Sigma_\sigma x_{1,\sigma 1}.x_{2,\sigma 2}. ... .x_{n,\sigma n}$, la sommation étant étendue à toutes les permutations σ de l'ensemble $\{1, 2, ... n\}$; sa définition ressemble à celle du déterminant, la différence étant qu'on n'affecte pas ses monômes d'un signe dépendant de la parité des permutations. Il a la propriété de complétude suivante : si $g(\underline{x},\underline{u})$ est un polynôme à coefficient rationnels, calculé par un terme arithmétique de complexité c ayant à ses entrées des variables ou des constantes rationnelles, le polynôme $\Sigma_{\underline{u}} g(\underline{x},\underline{u})$ s'obtient comme projection d'un permanent de taille polynomiale en c . Dans ce résultat, la constante $1/2$ joue un rôle essentiel dans les projections, ce qui n'est pas surprenant puisque le permanent, étant alors égal au déterminant, est

facile à calculer en caractéristique deux.

Pour avoir un résultat semblable avec des polynômes à coefficients entiers, et des projections qui n'utilisent que 0 et 1, ce qui permet de passer au quotient modulo p, et en particulier modulo deux, on utilise un autre polynôme, le hamiltonien, défini comme le permanent, à ceci près que la sommation est restreinte aux permutations circulaires (voir [Bürgisser 2000], [Malod 2003]). Le hamiltonien a une fonction-coefficient très simple, dans VPmd ; si on fait l'hypothèse que cela implique que la suite des hamiltoniens est dans VPmd, et alors tout élément de VNPmd est également dans VPmd. Appelons hamiltonienne la fonction booléenne dont le polynôme canonique est le hamiltonien ; nous aurions pu montrer notre Théorème 5 en utilisant la complétude de cette fonction pour la classe $SPt(F_2)$.

Nous avons préféré une démonstration passant par notre Théorème 4, qui évite de se casser la tête à construire des polynômes universels, et qui fournit directement des fonctions-coefficients calculées par des termes de complexité polynomiale, ce qui permet de l'utiliser pour les classes de complexité termiques. Comme la démonstration du Théorème 7 s'adapte mot-à-mot pour montrer que toute suite dans VNPmd = VNPt est projection d'une suite de polynômes dont la suite de fonctions-coefficients est dans VPt, c'est-à-dire dans Pt(Z), il est ainsi possible de montrer le résultat de Malod, ainsi que sa version termique, sans utiliser le hamiltonien, ce qui, d'après nous, éclaire d'avantage sa démonstration.

Rappelons qu'on ne sait pas si la classe VNPdl est close par fonction-coefficient, mais que [Malod 2003] a démontré qu'il en était ainsi de la classe VNPdl(Z/pZ), sa consoeur en caractéristique p. Il en a déduit l'équivalence des hypothèses VNPdl(Z/pZ) = VNPdl(Z/pZ) et VNPmd(Z/pZ) = VPmd(Z/pZ).

Remarque 8. Il devrait être possible de développer une théorie des transformations pour les suites de polynômes, ou de fonctions-polynômes.

Références

[Bürgisser 2000] Peter Bürgisser, Completeness and reduction in Algebraic Complexity Theory, Springer Verlag

[Malod 2003] Guillaume Malod, Polynômes et coefficients, thèse de doctorat, Université Claude Bernard

[MP 2006] Guillaume Malod et Natacha Portier, Characterizing Valiant's algebraic complexity classes, Lecture Notes in Computer Sciences, vol. 4162, 267-279

[MP 2008] Guillaume Malod et Natacha Portier, Characterizing Valiant's algebraic complexity classes, Journal of Complexity, 24, 16-38

[MP 1976] David E. Muller et Franco P. Preparata, Restructuring of arithmetic expressions for parallel evaluation, Journal of the Association for Computing Machinery, 23, 534-543

[Poizat 1995] Bruno Poizat, Les Petits Cailloux, Aléas, Lyon

[Poizat 2008] Bruno Poizat, A la recherche de la définition de la complexité d'espace pour le calcul des polynômes à la manière de Valiant, the Journal of Symbolic Logic, 73, 1179-1201

[Spira 1971] P.M. Spira, On the time necessary to compute Switching Functions, IEEE Transactions on Comp., 20, 104-105

[Valiant 1979] Leslie G. Valiant, The complexity of computing the permanent, Theoretical Computer Science, 8, 189-201

Efficient Synthesis of Exact Real Number Algorithms

Monika Seisenberger and Ulrich Berger

Swansea University, Wales, UK

1 Introduction

In this paper we present a method for developing correct programs for exact real numbers based on proof theoretic methods. The basic idea in general is to extract from a formal constructive proof of a specification a realisation of it, in the technical sense of a realisability interpretation. The reason why we believe that this method is worth pursuing is the fact that it offers a path to provably correct programs requiring – contrary to common belief – *less* effort than traditional methods based on verification. Our case studies on exact real number computation seem to confirm this. In addition they provide an approach to computability on real numbers which can easily be generalised to other abstract structures.

1.1 Algorithms for exact real numbers

We study real numbers in signed digit representation, see for instance [CDG06,EH02,GNSW07,MRE07]. Let for a real number $r \in [-1,1]$ and an infinite stream $a = a_0, a_1, a_2, \ldots$ of digits $a_i \in \{-1,0,1\}$ the predicate $R(r,a)$ denote that the real number r has stream representation a:

$$R(r,a) \;\leftrightarrow\; r = \sum_{i=0}^{\infty} a_i \cdot \frac{1}{2^{i+1}}\ . \tag{1}$$

For instance, the number $\frac{1}{2}$ has the stream representations: $100000\ldots$, $0111111\ldots$, $101{-}1{-}1{-}1{-}1\ldots$, etc.; hence representations are highly redundant. Our work is concerned with computation on real numbers, in particular with efficiently obtaining algorithms for them. A simple but instructive example is the average of two real numbers which we will consider in section 3. (The reader is encouraged to find such an algorithm on streams herself.)

In general, in order to provide a certified implementation of a, say binary, function f on exact real numbers we need to perform the following two tasks:

1. Specify and implement a corresponding operation f' on streams.
2. Show correctness:

$$\forall r,s,a,b.\ R(r,a) \to R(s,b) \to R(f(r,s), f'(a,b)). \tag{2}$$

Examples for implementations of such operations f' were given in [Plu98]. A formal verification of these algorithms, that is, the completion of task 2 above, was done in [BH08] using coinductive reasoning, where the relation R (see (1)) was characterised coinductively by the rule:

$$R(r,a) \to |r| \leq 1 \ \wedge\ R(2r - a_0, \mathsf{tail}\, a). \tag{3}$$

As indicated by the two tasks, this approach as well as the work of other researchers working in verification of exact real number algorithms have in common that the algorithms are first specified and implemented, and then verified. Moreover, the underlying system has to formalise both, the abstract real numbers together with their representations, as well as the algorithms performing computations on these representations.

1.2 Our approach

Here we deviate from the traditional specify-implement-verify method. We obtain the algorithms (together with correctness certificates) via an application of a well-known method in proof theory, realisability, i.e. extraction from formal proofs. In the example of exact real number computation this is achieved

by changing the binary predicate R (which relates real numbers and streams) to an unary predicate C_0 on real numbers, coinductively defined by:

$$\mathrm{C}_0(r) \;\to\; |r|{\leq}1 \;\wedge\; \exists d_{\in\{-1,0,1\}}.\; \mathrm{C}_0(2r-d). \tag{4}$$

Intuitively, the formula $\mathrm{C}_0(r)$ expresses that the real number r has a signed digit representation. Instead of our two tasks above we now only need to perform one task, i.e. to give a (constructive) proof of the simpler formula

$$\mathrm{C}_0(r) \;\wedge\; \mathrm{C}_0(s) \to \mathrm{C}_0(f(r,s)). \tag{5}$$

On the program level the definition of C_0 yields a stream. (Informal explanation: According to the coinductive definition a realiser of $\mathrm{C}_0(r)$ corresponds to one digit d in front of some realiser of $\mathrm{C}_0(2r-d)$; therefore this process leads to an infinite stream.) Overall, a proof of (5) corresponds to a program taking two streams as input and performing the desired operation on streams. That is, in general, we only have to reason about abstract real numbers such as in (5); their representation and the corresponding algorithms, as well as a formal correctness proof, are obtained automatically via the extraction process.

2 Program extraction from inductive and coinductive proofs

The program extraction method for proofs in a standard system, such as Heyting arithmetic in finite types, is based on Kreisel's modified realisability [Kre59] and well understood, see e.g. [BBS$^+$98] as a reference. The main effort consists now in allowing also coinductive reasoning and extending the program extraction process to coinduction. Program extraction for formal systems including coinduction has been studied in [Tat98] and [MP05]. We present an alternative treatment of realisability for coinductive definitions which has the advantage that it allows for nesting of inductive and coinductive definitions and extracts realisers that directly correspond to programs in a lazy functional programming language. In this extended abstract we only present a brief sketch of the method; for more details we refer to [BS10,BL10].

2.1 Induction and Coinduction

We treat induction and coinduction as least and greatest fixed points ($\mu\Phi$, $\nu\Phi$ respectively) of a strictly positive, and hence monotone, operator $\Phi : \mathcal{P}(U) \to \mathcal{P}(U)$, where U is a set and $\mathcal{P}(U)$ is the powerset of U. We have the axioms:

$$\begin{array}{llll} \text{Closure} & \Phi(\mu\Phi) \subseteq \mu\Phi & \text{Induction} & \Phi(\mathcal{Q}) \subseteq \mathcal{Q} \to \mu\Phi \subseteq \mathcal{Q} \\ \text{Coclosure} & \nu\Phi \subseteq \Phi(\nu\Phi) & \text{Coinduction} & \mathcal{Q} \subseteq \Phi(\mathcal{Q}) \to \mathcal{Q} \subseteq \nu\Phi \end{array}$$

The main task in the following will be to give appropriate realisers for these axioms.

2.2 The term language for the extracted programs

The output of the extraction process will be terms in a λ-calculus with constructors and pattern matching and (ML-)polymorphic recursive types. We let α range over type variables.

$$\text{Type} \ni \rho, \sigma ::= \alpha \mid \mathbf{1} \mid \rho + \sigma \mid \rho \times \sigma \mid \rho \to \sigma \mid \mathrm{fix}\,\alpha\,.\,\rho$$

In the definition of terms we let x range over term variables and C over constructors. It is always assumed that a constructor is applied to the correct number of arguments as determined by its arity. In fact, we will only use the constructors Nil (nullary), Left, Right (unary), Pair (binary), and $\mathrm{In}_{\mathrm{fix}\,\alpha\,.\,\rho}$ (unary) for every fixed point type $\mathrm{fix}\,\alpha\,.\,\rho$.

$$\text{Term} \ni M, N ::= \; x \mid C(\boldsymbol{M}) \mid \mathrm{case}\, M \,\mathrm{of}\{C_1(\boldsymbol{x_1}) \to N_1\,;\,\ldots\,;\,C_n(\boldsymbol{x_n}) \to N_n\} \mid \lambda x.M \mid (M\,N) \mid \mathrm{rec}\,x\,.\,M$$

In a pattern $C_i(\boldsymbol{x_i})$ of a case-term all variables in $\boldsymbol{x_i}$ must be different. The term $\mathrm{rec}\,x\,.\,M$ models recursion (e.g. in Haskell it would be `let {x = M} in x`). One can define a typing relation $\Gamma \vdash M : \rho$ which is ML-polymorphic: if $\vdash M : \rho(\boldsymbol{\alpha})$, then $\vdash M : \rho(\boldsymbol{\sigma})$, for arbitrary types $\boldsymbol{\sigma}$. The types of the constructors are Nil : $\mathbf{1}$, Left : $\rho \to \rho+\sigma$, Right : $\sigma \to \rho+\sigma$, Pair : $\rho \to \sigma \to \rho\times\sigma$, and $\mathrm{In}_{\mathrm{fix}\,\alpha\,.\,\rho} : \rho(\mathrm{fix}\,\alpha\,.\,\rho) \to \mathrm{fix}\,\alpha\,.\,\rho$.

2.3 The extraction process

The first step in program extraction is to assign to every formula A a type $\tau(A)$, the type of potential realisers of A. If the formula A contains neither predicate variables nor the logical connective $\vee$ (disjunction), then we call it "non-computational" and set $\tau(A) = \mathbf{1}$ ($:= \{\mathrm{Nil}\}$). For non-computational A we set $\tau(A \wedge B) = \tau(A \to B) = \tau(B)$. In all other cases we define $\tau(X(\boldsymbol{t})) = \alpha_X$ (a type variable assigned to the predicate variable X), $\tau(A \wedge B) = \tau(A) \times \tau(B)$, $\tau(A \vee B) = \tau(A) + \tau(B)$, $\tau(A \to B) = \tau(A) \to \tau(B)$, $\tau(\forall x\, A) = \tau(\exists x\, A) = \tau(A)$. For inductive and coinductive definitions we set $\tau(\mu\Phi(\boldsymbol{t})) = \tau(\nu\Phi(\boldsymbol{t})) = \mathrm{fix}\,\alpha_X\,.\,\tau(A)$ if $\Phi(X) = \{\boldsymbol{x} \mid A\}$.

The next step is to define for every formula A and every program term M of type $\tau(A)$ what it means for M to *realise* A, formally $M\,\mathbf{r}\,A$. The definition is straightforward (by recursion on the build-up of A) and as expected. We only comment on non-computational and quantified formulas. For a non-computational formula A we set $M\,\mathbf{r}\,A := (M = \mathrm{Nil}) \wedge A$, and for quantifiers we have $M\,\mathbf{r}\,\forall x\, A := \forall x\,(M\,\mathbf{r}\,A)$ and $M\,\mathbf{r}\,\exists x\, A := \exists x\,(M\,\mathbf{r}\,A)$. We see that quantifiers and the quantified variable, although ignored by the program M and its type, of course do play a role in the definition of realisability, i.e. the *specification* of the program.

Finally, we sketch how to extract from a proof of a formula A a program term M realising A. Assuming the proof is given in a natural deduction system the extraction process is straightforward and follows in most cases the usual pattern of the Curry-Howard correspondence. Only quantifier rules and axioms deserve special attention. As to be expected, quantifier rules are largely ignored. For example, in the rules for the universal quantifier the extracted programs of the conclusion are identical to those of the respective premises. Any non-computational axiom has the trivial program Nil as extracted program.

This leads us to realisers for induction and coinduction. The extracted programs of closure, $\Phi(\mu\Phi) \subseteq \mu\Phi$, and induction, $(\Phi(X) \subseteq X) \Rightarrow \Phi(\mu\Phi) \subseteq X$, are $\mathbf{in}_{\mathrm{fix}\,\alpha\,.\,\rho} := \lambda x.\mathrm{In}_{\mathrm{fix}\,\alpha\,.\,\rho}(x)$ and

$$\mathbf{it}_{\mathrm{fix}\,\alpha\,.\,\rho} := \lambda s.\mathrm{rec}\, f\,.\,\lambda x.\mathrm{case}\, x\, \mathrm{of}\{\mathrm{In}_{\mathrm{fix}\,\alpha\,.\,\rho}(y) \to s(\mathbf{map}_{\alpha,\rho} f\, y)\} \;:\; \rho(\mathrm{fix}\,\alpha\,.\,\rho) \to \mathrm{fix}\,\alpha\,.\,\rho$$

where it is assumed that $\tau(\Phi(X)) = \rho(\alpha)$. The term $\mathbf{map}_{\alpha,\rho}$ has type $(\alpha \to \beta) \to \rho(\alpha) \to \rho(\beta)$ and can be defined by induction on $\rho(\alpha)$. For coclosure, $\nu\Phi \subseteq \Phi(\nu\Phi)$, and coinduction, $(X \subseteq \Phi(X)) \Rightarrow X \subseteq \Phi(\nu\Phi)$, the extracted programs are $\mathbf{out}_{\mathrm{fix}\,\alpha\,.\,\rho} := \lambda x.\mathrm{case}\, x\, \mathrm{of}\{\mathrm{In}_{\mathrm{fix}\,\alpha\,.\,\rho}(y) \to y\}$ and

$$\mathbf{coit}_{\mathrm{fix}\,\alpha\,.\,\rho} := \lambda s.\mathrm{rec}\, f\,.\,\lambda x.\mathrm{In}_{\mathrm{fix}\,\alpha\,.\,\rho}(\mathbf{map}_{\alpha,\rho} f\,(s\, x)) \;:\; (\alpha \to \rho(\alpha)) \to \alpha \to \mathrm{fix}\,\alpha\,.\,\rho$$

The Soundness Theorem, stating that the program extracted from a proof does indeed realise the proven formula, is shown in [BS10].

3 Extracting the average function

We apply the theory described in the previous section to obtain a provably correct program that implements the average function on streams. Let $\mathrm{SD} := \{-1, 0, 1\}$ and $\mathbb{I} := [-1, 1] \subseteq \mathbb{R}$. Then $\mathrm{C}_0 := \nu\Phi$ is the greatest fixed point of the operator $\Phi : \mathcal{P}(\mathbb{I}) \to \mathcal{P}(\mathbb{I})$

$$\Phi(X) := \{\frac{x+d}{2} \mid d \in \mathrm{SD}, x \in X\}.$$

(Obviously, the definition of C_0 is equivalent to definition (4) given in section 1.) The coclosure axiom says that C_0 is coclosed: $\mathrm{C}_0 \subseteq \Phi(\mathrm{C}_0)$, and the coinduction principle says that C_0 contains every Φ-coclosed set: If $X \subseteq \Phi(X)$, then $X \subseteq \mathrm{C}_0$.

Lemma 1. *If $x, y \in \mathrm{C}_0$, then $\frac{x+y}{2} \in \mathrm{C}_0$.*

Proof. Set $X := \{\frac{x+y}{2} \mid x, y \in \mathrm{C}_0\}$. We have to show $X \subseteq \mathrm{C}_0$. A first attempt to directly use coinduction and show $X \subseteq \Phi(X)$ turns out to be difficult (try it!). It is easier to define

$$Y := \{\frac{x+y+i}{4} \mid x, y \in \mathrm{C}_0, i \in \mathrm{SD}_2\}$$

where $\mathrm{SD}_2 := \{-2, -1, 0, 1, 2\}$ and prove

i) $X \subseteq Y$.
ii) $Y \subseteq \Phi(Y)$ (hence $Y \subseteq \mathrm{C}_0$, by coinduction).

Proof of i) Let $x, y \in \mathrm{C}_0$. We have to show $z := \frac{x+y}{2} \in Y$. By the coclosure axiom, $x = \frac{x'+d}{2}$, $y = \frac{y'+e}{2}$, for some $d, e \in \mathrm{SD}$ and $x', y' \in \mathrm{C}_0$. Hence $z = \frac{x'+y'+d+e}{4} \in Y$.

Proof of ii) Let $x, y \in \mathrm{C}_0$ and $i \in \mathrm{SD}_2$. We have to show $z := \frac{x+y+i}{4} \in \Phi(Y)$. By coclosure, $x = \frac{x'+d'}{2}$, $y = \frac{y'+e'}{2}$, for some $d', e' \in \mathrm{SD}$ and $x', y' \in \mathrm{C}_0$. Hence $z = \frac{x'+y'+d'+e'+2i}{8}$. We must find $d \in \mathrm{SD}$ and $\tilde{z} \in Y$ such that $z = \frac{\tilde{z}+d}{2}$. By the definition of Y this means that we must find $\tilde{x}, \tilde{y} \in \mathrm{C}_0$ and $j \in \mathrm{SD}_2$ such that $\tilde{z} = \frac{\tilde{x}+\tilde{y}+j}{4}$, i.e. $z = \frac{\tilde{x}+\tilde{y}+j+4d}{8}$. Choosing $\tilde{x} := x'$ and $\tilde{y} := y'$ this leaves us with the equation $d' + e' + 2i = j + 4d$ which is clearly solvable with suitable $d \in \mathrm{SD}$ and $j \in \mathrm{SD}_2$.

From this proof we extract the following program `average` (written in Haskell syntax). `average` takes two infinite streams and first reads the first digits d and e on both of them; this corresponds to the proof of claim i). The functional `aux` recursively calls itself; this corresponds to the use of the coinduction principle with coiteration as realiser. The remainder of the program links to the computational content of claim ii) in an obvious way.

```
type SD = Int   -- -1, 0, 1 only
type SDS = [SD] -- infinite streams only
type SD2 = Int  -- -2, -1, 0, 1, 2 only

average :: SDS -> SDS -> SDS
average (d:ds) (e:es) = aux (d+e) ds es  where

   aux :: SD2 -> SDS -> SDS-> SDS
   aux i (d':ds) (e':es) = d : aux j ds es  where

      k = d'+e'+2*i

      d | abs k <= 2  = 0
        | k > 2       = 1
        | otherwise   = -1

      j = k-4*d
```

As a test we run the extracted program with inputs `[1,0,1]++[0,0..]`$= \frac{5}{8}$ and `[1,1]++[0,0..]` $= \frac{3}{4}$. The Haskell function `take 10` displays 10 digits from the result stream. We obtain as a result: $\frac{1}{2}+\frac{1}{4}-\frac{1}{16} = \frac{11}{16}$.

```
Main> take 10 (average ([1,0,1]++[0,0..]) ([1,1]++[0,0..]))
[1,1,0,-1,0,0,0,0,0,0]
```

4 Conclusion and further work

Using the extraction of the average function as an example, we have demonstrated that program extraction from formal proofs provides an efficient way for obtaining algorithms for real valued data.

For the efficiency of the method it is crucial that the extraction process is fully automated. The standard program extraction process is implemented in the interactive proof assistant Minlog [BBS+98,Sch06]. Minlog (`www.minlog-system.de`) is based on first order natural deduction, extended by inductive data types and inductive predicates, and allows for program extraction from both constructive and classical proofs. The extension of Minlog's extraction mechanism with respect to coinductive definitions is ongoing work together with Munich University.

With similar proofs one can show that C_0 is closed under multiplication and, more generally, under any uniformly continuous function. In recent work, C_0 has been generalised to predicates C_n characterising the uniformly continuous functions of n arguments by an interleaved inductive/coinductive definition [Ber09]. Based on these definitions we are currently developing new algorithms for uniformly continuous real functions, for example integration.

References

[BBS+98] H. Benl, U. Berger, H. Schwichtenberg, M. Seisenberger, and W. Zuber. Proof theory at work: Program development in the Minlog system. In W. Bibel and P.H. Schmitt, editors, *Automated Deduction – A Basis for Applications*, volume II of *Applied Logic Series*, pages 41–71. Kluwer, Dordrecht, 1998.

[Ber09] U. Berger. From coinductive proofs to exact real arithmetic. In E. Grädel and R. Kahle, editors, *Computer Science Logic*, volume 5771 of *LNCS*, pages 132–146. Springer, 2009.

[BH08] U. Berger and T. Hou. Coinduction for exact real number computation. *Theory of Computing Systems*, 43(3-4):394–409, 2008.

[BL10] U. Berger and S. Lloyd. A coinductive approach to verified exact real number computation. In *Proceedings of the Ninth International Workshop on Automated Verification of Critical Systems*, 2010. To appear.

[BS10] U. Berger and M. Seisenberger. Program extraction via typed realisability for induction and coinduction. In Ralf Schindler, editor, *Ways of Proof Theory. Ontos Verlag, Frankfurt*, 2010. To appear.

[CDG06] A. Ciaffaglione and P. Di Gianantonio. A certified, corecursive implementation of exact real numbers. *Theoretical Computer Science*, 351:39–51, 2006.

[EH02] A. Edalat and R. Heckmann. Computing with real numbers: I. The LFT approach to real number computation; II. A domain framework for computational geometry. In G. Barthe, P. Dybjer, L. Pinto, and J. Saraiva, editors, *Applied Semantics - Lecture Notes from the International Summer School, Caminha, Portugal*, pages 193–267. Springer, 2002.

[GNSW07] H. Geuvers, M. Niqui, B. Spitters, and F. Wiedijk. Constructive analysis, types and exact real numbers. *Mathematical Structures in Computer Science*, 17(1):3–36, 2007.

[Kre59] G. Kreisel. Interpretation of analysis by means of constructive functionals of finite types. *Constructivity in Mathematics*, pages 101–128, 1959.

[MP05] F. Miranda-Perea. Realizability for monotone clausular (co)inductive definitions. *Electronic Notes in Theoretical Computer Science*, 123:179–193, 2005.

[MRE07] J. R. Marcial-Romero and M. H. Escardo. Semantics of a sequential language for exact real-number computation. *Theoretical Computer Science*, 379(1-2):120–141, 2007.

[Plu98] D. Plume. A calculator for exact real number computation, 1998.

[Sch06] H. Schwichtenberg. Minlog. In F. Wiedijk, editor, *The Seventeen Provers of the World*, number 3600 in Lecture Notes in Artificial Intelligence, pages 151–157, 2006.

[Tat98] M. Tatsuta. Realizability of monotone coinductive definitions and its application to program synthesis. In R. Parikh, editor, *Mathematics of Program Construction*, pages 338–364. Springer, 1998.

Comparison of Complexity over the Real vs. Complex Numbers

Peter Scheiblechner

Department of Mathematics
Purdue University
West Lafayette, IN 47907-2067, USA,
psceibl@math.purdue.edu

Abstract. We compare complexity questions over the reals with their corresponding versions over the complex numbers. We argue in particular that the problem of deciding membership to a subspace arrangement is strictly harder over $\mathbb{C}$ than over $\mathbb{R}$. There exist decision trees of polynomial depth over $\mathbb{R}$ deciding this problem, whereas certain complex arrangements have no polynomial depth decision trees. This follows from a new lower bound for the decision complexity of a complex algebraic set X in terms of the sum of its (compactly supported) Betti numbers $b_c(X)$, which is for the first time better than logarithmic in $b_c(X)$. On the other hand, the existential theory over $\mathbb{R}$ is strictly harder than that over $\mathbb{C}$. This follows from the observation that the former cannot be solved by a complex BSS-machine.

1 Introduction

1.1 Membership to Subspace Arrangements

This paper consists of two settings, in which we compare the complexity of certain algorithmic problems over the real numbers with that over the complex numbers. In the first setting we study the problem of testing membership of a point to an arrangement of affine linear subspaces of k^n, where k is either $\mathbb{R}$ or $\mathbb{C}$. A famous example of such a problem is the knapsack problem, which is an arrangement of 2^n hyperplanes in k^n. Meyer auf der Heide [MadH84] proved that one can solve the knapsack problem over $\mathbb{R}$ by linear decision trees of polynomial depth. This result is generalized in [Cla87,Mei93] to arbitrary subspace arrangements. More precisely, it is proved that one can decide membership to an arrangement of m subspaces of $\mathbb{R}^n$ by a linear decision tree of depth polynomial in $n \log m$ (see also [BCS97, §3.4]).

It is easy to see by a generic path argument that such a result over $\mathbb{C}$ is impossible [BCSS98,Koi94]. We provide a second proof for this fact by proving a general lower bound for the decision tree complexity of a complex algebraic set X in terms of its (compactly supported) Betti numbers. There is a tradition of lower bounds for the decision complexity in terms of topological invariants [BO83,BLY92,Yao92,BO94,Yao94,LR96], see also the survey [Bür01]. All known lower bounds are *logarithmic*. For instance, the result of [Yao94] bounds the complexity of X by $\Omega(\log b_c(X))$, where $b_c(X)$ denotes the sum of the (compactly supported) Betti numbers of X. Our bound is the first non-logarithmic one. In particular, we prove that the decision tree complexity of an algebraic set X in $\mathbb{C}^n$ is bounded below by $(b_c(X)/m)^{\Omega(\frac{1}{n})}$, where m denotes the number of irreducible components of X. Although this looks like a big improvement, it yields better lower bounds only for quite large Betti numbers. Note that we need $b_c(X)$ to be at least of order $2^{n \log^{1+\varepsilon} n}$ for some $\varepsilon > 0$ to have $b_c(X)^{\frac{1}{n}}$ larger than $\log b_c(X)$. So for instance, our result yields worse lower bounds than Yao's for the element distinctness and k-equality problems. Examples of varieties on which our lower bound performs better are generic hyperplane arrangements or cartesian products of those. We are currently looking for more natural computational problems, where this is the case.

1.2 Existential Theory

The second part of the paper is motivated by the following question. Consider a system

$$f_1 = 0, \dots, f_r = 0, \quad f_i \in \mathbb{R}[X_1, \dots, X_n], \tag{1}$$

of *real* polynomial equations. We compare the following two problems:

$\text{FEAS}_{\mathbb{R}}$: Given $f_1, \ldots, f_r$, decide whether (1) has a *real* solution $x \in \mathbb{R}^n$.

$\text{HN}_{\mathbb{C}}$: Given $f_1, \ldots, f_r$, decide whether (1) has a *complex* solution $x \in \mathbb{C}^n$.

It is proved in [BSS89] that $\text{FEAS}_{\mathbb{R}}$ is complete for the class $\text{NP}_{\mathbb{R}}$ of languages decidable *nondeterministically* in polynomial time in the BSS-model over $\mathbb{R}$. The second problem $\text{HN}_{\mathbb{C}}$ is the restriction to the reals of the $\text{NP}_{\mathbb{C}}$-complete problem called *Hilbert Nullstellensatz*. The reductions used in these completeness-statements are polynomial time reductions in the BSS-model over $\mathbb{R}$ resp. $\mathbb{C}$.

A simple observation is that $\text{HN}_{\mathbb{C}}$ reduces (over the reals) to $\text{FEAS}_{\mathbb{R}}$ by replacing the n complex variables by $2n$ real ones and separating the real and complex parts of the equations. Conversely, one might ask whether one can reduce $\text{FEAS}_{\mathbb{R}}$ to $\text{HN}_{\mathbb{C}}$. We will answer this question negatively. More precisely, there is no (in fact not only polynomial time) reduction from $\text{FEAS}_{\mathbb{R}}$ to $\text{HN}_{\mathbb{C}}$ in the BSS-model over $\mathbb{R}$ *without order*. The question remains open whether there is one using the order.

2 A Lower Bound over $\mathbb{C}$

An *algebraic decision tree* of degree d over $\mathbb{C}$ is a rooted binary tree, whose inner nodes (*branching nodes*) are labeled with polynomials in $\mathbb{C}[X_1, \ldots, X_n]$ of degree at most d, and whose leaves are labeled with either "yes" or "no". The tree encodes a program that on input $(x_1, \ldots, x_n) \in \mathbb{C}^n$ parses the tree from the root to some leaf by testing at each branching node labeled with f, whether $f(x) = 0$, and continuing to the left or right according to the outcome of the test. The program answers the label of the leaf it reaches. Algebraic decision trees over $\mathbb{R}$ are defined analogously by testing $f \leq 0$.

For lower complexity bounds one needs invariants which are subadditive. We use the compactly supported Betti numbers (or equivalently, Borel-Moore Betti numbers). These are defined for a locally closed semialgebraic set X via the cohomology with compact support [Bür01]. We denote by $b_c(X)$ the sum of all compactly supported Betti numbers of X. Our result is based on the fundamental Oleinik-Petrovski/Milnor/Thom-Bound. The following is a version for the compactly supported Betti-numbers which is proved in [Bür01].

Theorem 1. *Let $X = \{f_1 = 0, \ldots, f_r = 0\} \subseteq \mathbb{R}^n$ be an algebraic set defined by the real polynomials f_i of degree at most d. Then $b_c(X) \leq d(2d-1)^n$.*

We need it for locally closed sets over the complex numbers.

Corollary 1. *Let X be the locally closed subset of $\mathbb{C}^n$ defined by $f_1 = 0, \ldots, f_r = 0, g_1 \neq 0, \ldots, g_s \neq 0$, where $s \geq 1$ and $f_i, g_j \in \mathbb{C}[X_1, \ldots, X_n]$ have degree at most d. Then $b_c(X) \leq (sd+1)(2sd+1)^{2n+2}$.*

For a constructible set $X \subseteq \mathbb{C}^n$ the *degree d decision complexity* $C_d(X)$ is the minimal depth of a degree d algebraic decision tree deciding X. Denote by $\#\text{ic}(X)$ the number of irreducible components of X.

Theorem 2. *Let $X \subseteq \mathbb{C}^n$ be a closed algebraic set. Then*

$$C_d(X) \geq \frac{1}{4d^2} \left(\frac{b_c(X)}{\#\text{ic}(X)} \right)^{\frac{1}{3n+1}}.$$

In the following we sketch the proof of this theorem. Its beginning is analogous to the proof of Yao's lower bound. Let T be a degreee d decision tree of depth k deciding X. For a leave ν denote by D_ν its *leaf set* consisting of those $x \in \mathbb{C}^n$ following the path of T to the leaf ν. We clearly have the decomposition $X = \bigsqcup_{\nu \in \Upsilon} D_\nu$, where $\Upsilon = \{\text{yes-leaves of } T\}$. Furthermore, each leaf set can be written as

$$D_\nu = \{f_1 = 0, \ldots, f_r = 0, g_1 \neq 0, \ldots, g_s \neq 0\}, \quad \text{where} \quad \deg f_i,\ \deg g_i \leq d. \tag{2}$$

Note that $s \leq k-1$ if $X \neq \mathbb{C}^n$. Using the subadditivity and Corollary 1 we conclude

$$b_c(X) \leq \sum_{\nu \in \Upsilon} b_c(D_\nu) \leq |\Upsilon| kd(2kd)^{2n+2}. \tag{3}$$

Yao now bounds $|\Upsilon|$ by the number of *all* leaves, which is in our case at most 2^k. The new idea is to improve this to a polynomial bound in k.

First note that each irreducible component Z of X must be contained in a unique $\overline{D}_{\nu(Z)}$. We set $I_1 := \{\nu(Z) \mid Z \text{ component of } X\}$ and $X_1 := \bigcup_{\nu \in \Upsilon \setminus I_1} \overline{D}_\nu$. Then X_1 is a lowerdimensional subvariety of X, and each component Z of X_1 is contained in a unique $\overline{D}_{\nu(Z)}$. We set $I_2 := I_1 \sqcup \{\nu(Z) \mid Z \text{ component of } X_1\}$ and $X_2 := \bigcup_{\nu \in \Upsilon \setminus I_2} \overline{D}_\nu$. Continuing this way we get sequences of subvarieties $X = X_0 \supseteq X_1 \supseteq \cdots \supseteq X_m \supseteq X_{m+1} = \emptyset$ and subsets $\emptyset = I_0 \subseteq I_1 \subseteq \cdots \subseteq I_m \subseteq I_{m+1} = \Upsilon$, where $m = \dim X$. By construction we have $|I_i| \leq \sum_{j<i} \#\mathrm{ic}(X_j)$, hence we are left with the task to bound $\#\mathrm{ic}(X_i)$ or $\deg X_i$. Using (2) and Bézout's Theorem we prove $\deg X_i \leq \deg X (kd)^i$ and conclude $|\Upsilon| \leq \sum_i \#\mathrm{ic}(X_i) \leq \deg X \sum_i (kd)^i \leq 2 \deg X (kd)^m$. Bounding $\deg X \leq \#\mathrm{ic}(X) d^n$, plugging this into (3) and solving for k yields our Theorem.

In order to bound the Betti numbers of subspace arrangements, we collect some well-known facts. We write $b(X)$ for the sum of the singular Betti numbers of a space X. Let $X = \bigcup_i A_i \subseteq \mathbb{R}^n$ be a real subspace arrangement. Its *complexification* is defined to be $X^{\mathbb{C}} := \bigcup_i A_i^{\mathbb{C}}$, where $A_i^{\mathbb{C}} \subseteq \mathbb{C}^n$ is defined by the same equations as A_i. Since the sum of the Betti numbers of the complement of an arrangement depends only on its intersection semilattice, we have $b(\mathbb{R}^n \setminus X) = b(\mathbb{C}^n \setminus X^{\mathbb{C}})$ [Bjö92, §8.3]. Furthermore, Alexander duality in $\mathbb{R}^n$ resp. its Alexandrov compactification $S^n = \mathbb{R}^n \cup \{\infty\}$ implies that for any closed subset $X \subseteq \mathbb{R}^n$ we have $b_c(X) = b(\mathbb{R}^n \setminus X)$.

With these facts it follows from [BCS97, Lemma (11.10)] that $b_c(\mathrm{Dist}_n) \geq n!$, where $\mathrm{Dist}_n = \bigcup_{i<j} \{x_i = x_j\}$ is the element-distinctness problem. Yao's bound shows $C_d(\mathrm{Dist}_n) \geq \Omega(n \log n)$, whereas Theorem 2 only implies $C_d(\mathrm{Dist}_n) \geq \Omega(\sqrt[3]{n})$. For the knapsack problem $\mathrm{KN}_n = \bigcup_{I \subseteq [n]} \left\{\sum_{i \in I} = 1\right\}$, Theorem 2 together with [BCS97, Lemma (11.14)] yields $C_d(\mathrm{KN}_n) \geq 2^{\Omega(n)}$, now better than Yao's quadratic bound. However, a simple generic-path argument implies $C_d(X) \geq \frac{m}{d}$ for a hyperplane arrangement X with m hyperplanes, which also shows the single exponential lower bound for the knapsack problem.

The generic-path arguement does not apply to arrangements of higher codimension. To get such a subspace arrangement with sufficiently large Betti numbers, one can take $\mathrm{KN}_n^2 \subseteq \mathbb{C}^{2n}$, which has codimension 2. Using the Künneth-Theorem one can show $b_c(X \times Y) \geq b_c(X) + b_c(Y)$, hence Theorem 2 implies $C_d(\mathrm{KN}_n^2) \geq 2^{\Omega(2n)}$. However, this lower bound may also be derivable by reducing to the knapsack problem. An interesting problem for our bound would be the following: $\{(X, b) \in \mathbb{C}^{n \times n} \times \mathbb{C}^n \mid \exists e \in \{0,1\}^n \, Xe = b\}$. Unfortunatley, at present we don't know a bound for the Betti numbers of this arrangement.

3 Real Parts

For a complexity class $\mathcal{C}$ of languages in $\mathbb{C}^\infty$ define its *real part* as $\mathbb{R}\mathrm{P}(\mathcal{C}) := \{A \cap \mathbb{R}^\infty \mid A \in \mathcal{C}\}$.

Proposition 1. *We have* $\mathbb{R}\mathrm{P}(\mathrm{P}_{\mathbb{C}}) = \mathrm{P}_{\mathbb{R}}^{=}$ *and* $\mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}}) \subseteq \mathrm{NP}_{\mathbb{R}}^{=} = \mathrm{NP}_{\mathbb{R}}$.

This proposition is proved as follows. One sees $\mathbb{R}\mathrm{P}(\mathrm{P}_{\mathbb{C}}) \subseteq \mathrm{P}_{\mathbb{R}}^{=}$ by simulating a complex machine by a real $=$-machine. For the reverse inclusion, view a real $=$-machine as a complex machine and ensure by clocking a polynomial running time on all complex inputs. The second inclusion of Proposition 1 follows from the first. Finally, $\mathrm{NP}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}^{=}$ follows from the observation that $\textsc{Feas}_{\mathbb{R}}$ is $\mathrm{NP}_{\mathbb{R}}$-complete also for polynomial $=$-reductions, and $\mathrm{NP}_{\mathbb{R}}^{=}$ is closed under those.

We call a subset $S \subseteq \mathbb{R}^n$ *constructible*, iff S can be defined by a quantifier-free first-order formula whose atoms are of the form $f = 0$ or $f \neq 0$ with a polynomial f. We need the following result of [CR93].

Theorem 3. *If $A \subseteq \mathbb{R}^\infty$ is decidable by a $=$-machine, then $A \cap \mathbb{R}^n$ is constructible for all $n \in \mathbb{N}$.*

Observe that $\textsc{Feas}_{\mathbb{R}}$ is not decidable, since $\{(a, b, c) \in \mathbb{R}^3 \mid b^2 - 4ac \geq 0\}$ is not constructible. Theorem 3 implies

Theorem 4. *We have* $\mathbb{R}\mathrm{P}(\mathrm{P}_{\mathbb{C}}) \subseteq \mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}}) \subsetneq \mathrm{NP}_{\mathbb{R}}^{=}$.

Note that for this separation one can also use the language $\{x_1 > 0\}$. It also follows from [MMD83]. To answer the question of the introduction, we remark that there can be no polynomial time $=$-reduction of $\textsc{Feas}_{\mathbb{R}}$ to $\mathrm{HN}_{\mathbb{C}}$, since $\mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}})$ is closed under those reductions.

It is interesting that $\mathrm{NP}_{\mathbb{R}}^{=}$ contains undecidable languages, and also that in the presence of an existential quantifier the destinction between full and $=$-machines disappears. These remarks extend to the higher levels of the polynomial hierarchy. We pose the following questions:

1. We have the following transfer result: $\mathrm{P}_{\mathbb{C}} = \mathrm{NP}_{\mathbb{C}} \Rightarrow \mathrm{P}_{\mathbb{R}}^{=} = \mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}})$. Does the other direction hold?
2. We only have $\mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}}) \neq \mathrm{NP}_{\mathbb{R}}^{=}$ for a silly reason. Is there a natural class of problems $\mathcal{C}$ such that $\mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}}) = \mathrm{NP}_{\mathbb{R}}^{=} \cap \mathcal{C}$? Maybe the class of decidable languages?
3. Does the order help in deciding $\mathrm{HN}_{\mathbb{C}}$? In other words, is $\mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}}) \subseteq \mathrm{P}_{\mathbb{R}}$? Note that the reverse inclusion does not hold: $\{x_1 > 0\} \in \mathrm{P}_{\mathbb{R}} \setminus \mathbb{R}\mathrm{P}(\mathrm{NP}_{\mathbb{C}})$.

Acknowledgements

Peter Scheiblechner is supported by DFG grant SCHE 1639/1-1. The author would like to thank the anonymous referees for valuable comments.

References

[BCS97] P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1997.

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer, 1998.

[Bjö92] A. Björner. Subspace arrangements. In *Proc. of 1st European Congress of Mathematics*, pages 321–370, Paris, 1992, 1992. Birkhäuser.

[BLY92] A. Björner, L. Lovász, and A. Yao. Linear decision trees: volume estimates and topological bounds. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 170–177, New York, NY, USA, 1992. ACM.

[BO83] M. Ben-Or. Lower bounds for algebraic computation trees. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 80–86, New York, NY, USA, 1983. ACM.

[BO94] M. Ben-Or. Algebraic computation trees in characteristic $p > 0$. In *FOCS '94: Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 534–539, Washington, DC, USA, 1994. IEEE Computer Society.

[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers. *Bull. Amer. Math. Soc.*, 21:1–46, 1989.

[Bür01] Peter Bürgisser. Lower bounds and real algebraic geometry. In *Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science*, pages 35–54, 2001.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[CR93] Felipe Cucker and Francesc Rosselló. Recursiveness over the complex numbers is time-bounded. In *Proceedings of the 13th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 260–267, London, UK, 1993. Springer-Verlag.

[Koi94] P. Koiran. Computing over the reals with addition and order. In *Selected papers of the workshop on Continuous algorithms and complexity*, pages 35–47, New York, NY, USA, 1994. Elsevier Science Inc.

[LR96] T. Lickteig and M.M. Roy. Semi-algebraic complexity of quotients and sign determination of remainders. *J. Compl.*, 12(4):545–571, 1996.

[MadH84] F. Meyer auf der Heide. A polynomial linear search algorithm for the n-dimensional knapsack problem. *J. ACM*, 31(3):668–676, 1984.

[Mei93] S. Meiser. Point location in arrangements of hyperplanes. *Inf. Comput.*, 106(2):286–303, 1993.

[MMD83] A. Macintyre, K. McKenna, and L. van den Dries. Elimination of quantifiers in algebraic structures. *Advances in Mathematics*, 47:74–87, 1983.

[Yao92] A. Yao. Algebraic decision trees and Euler characteristics. In *FOCS '92: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 268–277, Washington, DC, USA, 1992. IEEE Computer Society.

[Yao94] A. Yao. Decision tree complexity and Betti numbers. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 615–624, New York, NY, USA, 1994. ACM.

Computable functions of reals

Katrin Tent[1] and Martin Ziegler[2]

[1] Mathematisches Institut, Universität Münster, Einsteinstrasse 62, D-48149 Münster, Germany
tent@math.uni-muenster.de
[2] Mathematisches Institut, Albert-Ludwigs-Universität Freiburg, Eckerstr. 1, D-79104 Freiburg, Germany
ziegler@uni-freiburg.de

We introduce for every suitable class $\mathcal{F}$ of computable functions $\mathbb{N}^n \to \mathbb{N}$ a new notion of $\mathcal{F}$–computable functions from open subsets U of $\mathbb{R}^m$ to R. We show that these functions are closed under various operations, which implies that certain holomorphic functions like the exponentiation, the logarithm, the Gamma function and the Zeta function are low elementary computable on bounded subsets of their domain of definition. Our approach goes back to Grzegorczyk and the hierarchy of elementary functions and real numbers developed by him and Mazur (see [2] footnote p. 201).

The main idea is to use the density of the rationals in the real numbers to approximate functions on the reals by computable functions on the rational numbers. We show that apparently many of the functions appearing naturally in mathematics are computably approximable in this sense. Our results imply that the lower elementary reals and complex numbers form a real closed field or algebraically closed field, respectively.

1 Good classes of functions

A class $\mathcal{F}$ of functions $\mathbb{N}^n \to \mathbb{N}$ is called *good* if it contains the constant functions, the projection functions, the successor function, the modified difference function $x \dot{-} y = \max\{0, x - y\}$, and is closed under composition and *bounded summation*

$$f(\bar{x}, y) = \sum_{i=0}^{y} g(\bar{x}, i).$$

The class of *lower elementary* functions is the smallest good class. The smallest good class which is also closed under *bounded product*

$$f(\bar{x}, y) = \prod_{i=0}^{y} g(\bar{x}, i),$$

or – equivalently – the smallest good class which contains $n \mapsto 2^n$, is the class of *elementary* functions The elementary functions are the third class ϵ^3 of the Grzegorczyk hierarchy. The lower elementary functions belong to ϵ^2. It is not known whether all functions in ϵ^2 are lower elementary.

A function $f : \mathbb{N}^n \to \mathbb{N}^m$ is an $\mathcal{F}$–function if its components $f_i : \mathbb{N}^n \to \mathbb{N}$, $i = 1, \ldots, m$, are in $\mathcal{F}$. A relation $R \subset \mathbb{N}^n$ is called an $\mathcal{F}$–relation if its characteristic function belongs to $\mathcal{F}$. Note that a good class is closed under the bounded μ–operator: if R belongs to $\mathcal{F}$, then so does the function

$$f(\bar{x}, y) = \min\{i \mid R(\bar{x}, i) \vee i = y\}.$$

As a special case we see that $\lfloor \frac{x}{y} \rfloor$ is lower elementary. The $\mathcal{F}$–relations are closed under Boolean combinations and bounded quantification:

$$S(x, y) \;\Leftrightarrow\; \exists\, i \leq y\, R(x, i).$$

It follows for example that for any f in $\mathcal{F}$ the maximum function

$$\max_{j \leq y} f(\bar{x}, j) = \min\{\, i \mid \forall j \leq y\ f(\bar{x}, j) \leq i \,\}$$

is in $\mathcal{F}$ since it is bounded by $\sum_{i=0}^{y} f(\bar{x}, i)$.

We call a set X an $\mathcal{F}$–retract (of $\mathbb{N}^n$) if there are functions $\iota : X \to \mathbb{N}^n$ and $\pi : \mathbb{N}^n \to X$ given with $\pi \circ \iota = \mathrm{id}$ and $\iota \circ \pi \in \mathcal{F}$. Note that the product $X \times X'$ of two $\mathcal{F}$–retracts X and X' is again an $\mathcal{F}$–retract (of $\mathbb{N}^{n+n'}$) in a natural way. We define a function $f : X \to X'$ to be in $\mathcal{F}$ if $\iota' \circ f \circ \pi : \mathbb{N}^n \to \mathbb{N}^{n'}$ is in $\mathcal{F}$. By this definition the two maps $\iota : X \to \mathbb{N}^n$ and $\pi : \mathbb{N}^n \to X$ belong to $\mathcal{F}$. For an $\mathcal{F}$–retract X, a subset of X is in $\mathcal{F}$ if its characteristic function is. It now makes sense to say that a set Y (together with $\iota : Y \to X$ and $\pi : X \to Y$) is a retract of the retract X. Clearly Y is again a retract in a natural way.

Easily, $\mathbb{N}_{>0}$ is a lower elementary retract of $\mathbb{N}$ and $\mathbb{Z}$ is a lower elementary retract of $\mathbb{N}^2$ via $\iota(z) = (\max(z,0), -\min(0,z))$ and $\pi(n,m) = n - m$. We turn $\mathbb{Q}$ into a lower elementary retract of $\mathbb{Z} \times \mathbb{N}$ by setting $\iota(r) = (z,n)$, where $\frac{z}{n}$ is the unique representation of r with $n > 0$ and $(z,n) = 1$. Define $\pi(z,n)$ as $\frac{z}{n}$ if $n > 0$ and as 0 otherwise.

For the remainder of this note we will consider $\mathbb{Z}$ and $\mathbb{Q}$ as *fixed lower elementary* retracts of $\mathbb{N}$, using the maps defined in the last paragraph.

One of the most important observations is the fact that even though sums of a series of rational numbers cannot necessarily be computed accurately in a low way, they can be computed well enough:

Lemma 1. *Let $g : X \times \mathbb{N} \to \mathbb{Q}$ be in $\mathcal{F}$ for some $\mathcal{F}$-retract X. Then there is an $\mathcal{F}$–function $f : X \times \mathbb{N} \times \mathbb{N}_{>0} \to \mathbb{Q}$ such that*

$$\left| f(x,y,k) - \sum_{i=0}^{y} g(x,i) \right| < \frac{1}{k} \text{ for all } x \in X, y \in \mathbb{N} \text{ and } k \in \mathbb{N}_{>0}.$$

Definition 1. *A real number x is an $\mathcal{F}$–real if for some $\mathcal{F}$–function $a : \mathbb{N} \to \mathbb{Q}$*

$$|x - a(k)| < \frac{1}{k}$$

In [5] Skordev has shown among other things that π is lower elementary and that e is in ϵ^2. Weiermann [6] proved that e is lower elementary. He used the representation $\mathrm{e} = \sum \frac{1}{n!}$ and a theorem of d'Aquino [1], which states that the graph of the factorial is Δ_0–definable and therefore lower elementary.

We show that volumes of bounded 0–definable semialgebraic sets are lower elementary, and so is π as the volume of the unit circle. We also show that many special functions, among them the exponential function, map lower elementary reals into lower elementary reals.

2 Computable functions on the reals

Let O be an open subset of $\mathbb{R}^N$ where we allow $N = 0$. For $e \in \mathbb{N}_{>0}$ put

$$O \upharpoonright e = \left\{ x \in O \;\middle|\; |x| \leq e \right\}$$

and

$$O_e = \left\{ x \in O \upharpoonright e \;\middle|\; \mathrm{dist}(x, \mathbb{R}^N \setminus O) \geq \frac{1}{e} \right\}.$$

(We use the maximum norm on $\mathbb{R}^N$.) Notice:

1. O_e is compact.
2. $U \subset O \;\Rightarrow\; U_e \subset O_e$.
3. $e < e' \;\Rightarrow\; O_e \subset O_{e'}$.
4. $O = \bigcup_{e \in \mathbb{N}} O_e$
5. $O_e \subset (O_{2e})^{\circ}_{2e}$

Definition 2. *Let $\mathcal{F}$ be a good class. A function $F : O \to \mathbb{R}$ is in $\mathcal{F}$ if there are $\mathcal{F}$–functions $d : \mathbb{N} \to \mathbb{N}$ and $f : \mathbb{Q}^N \times \mathbb{N} \to \mathbb{Q}$ such that for all $e \in \mathbb{N}_{>0}$ and all $a \in \mathbb{Q}^N$ and $x \in O_e$*

$$|x - a| < \frac{1}{d(e)} \to |F(x) - f(a,e)| < \frac{1}{e}. \tag{1}$$

[3]

[3] As pointed out by the referee even in the case when $\mathcal{F}$ is the class of lower elementary functions, the $\mathcal{F}$–functions on $\mathbb{R}$ are not necessarily computable in the sense of [7].

This definition easily extends to $f : O \to \mathbb{R}^M$ (under the maximum norm): f is in $\mathcal{F}$ if and only if all f_i, $i = 1, \ldots, M$, are in $\mathcal{F}$.

The following properties are easy to see but crucial for everything that follows:

Lemma 2. *1. $\mathcal{F}$–functions map $\mathcal{F}$–reals to $\mathcal{F}$–reals. A constant function on $\mathbb{R}^N$ is uniformly in $\mathcal{F}$ if and only if its value is an $\mathcal{F}$–real.*
2. $\mathcal{F}$–functions $O \to \mathbb{R}$ are continuous.
3. If $F : O \to \mathbb{R}^M$ is in $\mathcal{F}$, $U \subset \mathbb{R}^M$ open and $G : U \to \mathbb{R}$ uniformly in $\mathcal{F}$, then $G \circ F : F^{-1}U \cap O \to \mathbb{R}$ is in $\mathcal{F}$. If F is uniformly in $\mathcal{F}$, then so is $G \circ F$.

3 Semialgebraic functions

In this note, semialgebraic functions (relations) are functions (relations) definable *without parameters* in $\mathbb{R}$. The *trace* of a relation $R \subseteq \mathbb{R}^N$ on $\mathbb{Q}$ is $R \cap \mathbb{Q}^N$.

The following observation is due to Yoshinaga [8] and follows from quantifier elimination.

Lemma 3. *The trace of semialgebraic relations on $\mathbb{Q}$ is lower elementary.*

Note that any semialgebraic function $g : \mathbb{R} \to \mathbb{R}$ is polynomially bounded, i.e. there is some $n \in \mathbb{N}$ with $|g(x)| \leq |x|^n$ for sufficiently large x. This is used to show:

Theorem 31 *Continuous semialgebraic functions $F : O \to \mathbb{R}$ are lower elementary for every open semialgebraic set O.*

Corollary 32 *The set of $\mathcal{F}$-reals $\mathbb{R}_{\mathcal{F}}$ forms a real closed field.*

Corollary 32 was first proved by Skordev (see [4]). For countable good classes $\mathcal{F}$, like the class of lower elementary functions, clearly $\mathbb{R}_{\mathcal{F}}$ is a countable subfield of $\mathbb{R}$.

4 Integration

Theorem 41 *Let $O \subset \mathbb{R}^N$ be open and $G, H : O \to \mathbb{R}$ in $\mathcal{F}$ such that $G < H$ on O. Put*

$$U = \big\{(x, y) \in \mathbb{R}^{N+1} \mid x \in O,\ G(x) < y < H(x)\big\}.$$

Assume further that $F : U \to \mathbb{R}$ is in $\mathcal{F}$ and that $|F(x, y)|$ is bounded by an $\mathcal{F}$–function $K(x)$. Then

$$I(x) = \int_{G(x)}^{H(x)} F(x, y)\,\mathrm{d}y \tag{2}$$

is an $\mathcal{F}$–function $O \to \mathbb{R}$.

This set-up can be used to give a different proof of Yoshinaga's Theoremon periods:

Kontsevich and Zagier [3] define a period as *a complex number whose real and imaginary parts are values of absolutely convergent integrals of rational functions with rational coefficients over domains in $\mathbb{R}^n$ given by polynomial inequalities with rational coefficients.* The periods form a ring containing all algebraic reals. It is an open problem whether e is a period.

Yoshinaga [8] proved that periods are *elementary.* An analysis of his proof shows that he actually showed that periods are lower elementary. We here give a variant of his proof where part of his argument is replaced by an application of Theorem 41.

Corollary 42 *Periods are lower elementary.*

Proof. By Lemma 24 of [8], periods are differences of sums of volumes of bounded open semialgebraic cells.

5 The Inverse Function Theorem

We call a sequence $\mathcal{A}_1, \mathcal{A}_2, \ldots$ of subsets of $\mathbb{Q}^N$ an *$\mathcal{F}$–sequence*, if $\{(e,a) \mid a \in \mathcal{A}_e\}$ is an $\mathcal{F}$–subset of $\mathbb{N}_{>0} \times \mathbb{Q}^N$.

Definition 3. *An open set $O \subset \mathbb{R}^N$ is $\mathcal{F}$–approximable if there is an $\mathcal{F}$–sequence $\mathcal{A}_1, \mathcal{A}_2, \ldots$ of subsets of $\mathbb{Q}^N$ and an $\mathcal{F}$–function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $O_e \cap \mathbb{Q}^N \subset \mathcal{A}_e \subset O_{\alpha(e)}$ for all $e \in \mathbb{N}_{>0}$.*

It follows from Lemma 3 that semialgebraic sets O are lower elementary approximable. We can simply set $\mathcal{A}_e = O_e \cap \mathbb{Q}^N$.

Theorem 51 *Let $F : O \to V$ be a bijection in $\mathcal{F}$ where O is $\mathcal{F}$–approximable and V open in $\mathbb{R}^N$. Assume that the inverse $G : V \to O$ satisfies:*

(i) There is an $\mathcal{F}$–function $d' : \mathbb{N} \to \mathbb{N}$ such that $|G(y) - G(y')| < \frac{1}{e}$ for all $y, y' \in V_e$ with $|y - y'| < \frac{1}{d'(e)}$.
(ii) G is $\mathcal{F}$–compact.

Then G is also in $\mathcal{F}$.

6 Series of functions

In order to investigate holomorphic and analytic functions we need the notion of $\mathcal{F}$-series of functions.

Definition 4. *A sequence $F_1, F_2, \ldots$ of functions $O \to \mathbb{R}^M$ $\mathcal{F}$–converges against F, if there is an $\mathcal{F}$–function $m : \mathbb{N} \to \mathbb{N}$ such that $|F(x) - F_i(x)| < \frac{1}{e}$ for all $x \in O_e$ and $i \geq m(e)$.*

Lemma 4. *The $\mathcal{F}$–limit of an $\mathcal{F}$–sequence of functions is an $\mathcal{F}$–function.*

We now give some examples:
1. **Logarithm**: As a semialgebraic function $\frac{1}{x} : \mathbb{R}_{>0} \to \mathbb{R}$ is lower elementary. So by Theorem 41 $\ln(x) = \int_1^x \frac{1}{y}\,\mathrm{d}y$ is lower elementary, at least on $(0,1)$ and $(1,\infty)$. But writing

$$\ln(x) = \int_0^1 \frac{x-1}{1+t(x-1)}\,\mathrm{d}t,$$

we see that $\ln(x)$ is lower elementary on $(0,\infty)$.
2. **Exponentiation**: As $\exp(x)$ is bounded on every interval $(-\infty, r)$, the Inverses Function Theorem applied to $\ln : (0,1) \to (-\infty, 0)$ allows us to conclude that $\exp(x)$ is lower elementary on $(-\infty, 0)$ and – by translation – on every interval $(-\infty, r)$. $\exp(x)$ cannot be lower elementary on the whole real line since it grows too fast. Nevertheless the following is true.

Lemma 5. *$G(x,y) = \exp(x)$ is lower elementary on $V = \{(x,y) \colon \exp(x) < y\}$.*

It is easy to see that $\exp(z)$ is elementary on $\mathbb{C}$. More is true:
The complex logarithm defines a homeomorphism

$$\ln : \mathbb{C} \to \{z \mid \mathrm{Im}(z) \in (-\pi, \pi)\}$$

and for all $r < s \in \mathbb{R}$ a uniformly lower elementary homeomorphism between $\{z \in \mathbb{C} \setminus \mathbb{R}_{\leq 0} \mid \exp(r) < |z| < \exp(s)\}$ and $\{z \mid \mathrm{Im}(z) \in (-\pi, \pi), r < \mathrm{Re}(z) < s\}$. Again by a variant of the Inverse Function Theorem we see that $\exp(z)$ is lower elementary on $\{z \mid \mathrm{Im}(z) \in (-\pi, \pi), r < \mathrm{Re}(z) < s\}$. Using the periodicity of $\exp(z)$ it is now easy to see that $\exp(z)$ is lower elementary on each strip $\{z \mid r < \mathrm{Re}(z) < s\}$. This implies that $\sin(x) : \mathbb{R} \to \mathbb{R}$ is lower elementary. Now also $\cos(x)$ is lower elementary and since $\exp(x + y\,\mathrm{i}) = \exp(x)(\cos(y) + \sin(y)\,\mathrm{i})$ we see that $\exp(z)$ is lower elementary on every half-space $\{z \mid \mathrm{Re}(z) < s\}$.

3. **Gamma function**:

We have for $\mathrm{Re}(x) > 1$

$$\begin{aligned}\Gamma(x) &= \int_0^\infty t^{-1+x}\exp(-t)\,\mathrm{d}t \\ &= \int_0^1 t^{-1+x}\exp(-t)\,\mathrm{d}t + \int_1^\infty t^{-1+x}\exp(-t)\,\mathrm{d}t \\ &= \int_0^1 t^{-1+x}\exp(-t)\,\mathrm{d}t + \int_0^1 \frac{1}{t^{1+x}}\exp\big(\frac{-1}{t}\big)\,\mathrm{d}t.\end{aligned}$$

Let us check that for every bound $r > 1$ the two integrands are lower elementary on $X_r = \{(x,t) \mid \mathrm{Re}(x) \in (1,r), t \in (0,1)\}$: $\exp(-t)$ is lower elementary on $(0,1)$. And, since $\frac{1}{t} : (0,1) \to (1,\infty)$ is lower elementary *compact*, the function $\exp(\frac{-1}{t})$ is lower elementary on $(0,1)$. t^{-1+x} and $\frac{1}{t^{1+x}} = \big(\frac{1}{t}\big)^{x+1}$ are lower elementary.

If $r > 1$, the absolute values of the integrands are bounded by 1 and $(r+1)^{(r+1)}$, respectively. So by Theorem 41, Γ is lower elementary on every strip $\{z \mid 1 < \mathrm{Re}(z) < r\}$.

4. **Zeta–function**: Since x^y is lower elementary on $(0,1) \times \{z \mid \mathrm{Re}(z) > 0\}$, the function $(\frac{1}{x})^y$ is lower elementary on $(1,\infty) \times \{z \mid \mathrm{Re}(z) > 0\}$. This implies that the sequence $\frac{1}{n^s}$, $(n = 1, 2, \ldots)$ is a lower elementary sequence of functions defined on $\{z \mid \mathrm{Re}(z) > 0\}$. The series

$$\zeta(z) = \sum_{n=1}^{\infty} \frac{1}{n^z}$$

converges whenever $t = \mathrm{Re}(z) > 1$ and we have the estimate

$$\Big|\zeta(z) - \sum_{n=1}^{N} \frac{1}{n^z}\Big| \;\le\; \int_N^\infty \frac{1}{x^t}\,\mathrm{d}x = \frac{1}{(t-1)N^{t-1}}.$$

So, if $\mathrm{Re}(z) \ge 1 + \frac{1}{k}$ and $N \ge (ke)^k$, we have $\Big|\zeta(z) - \sum_{n=1}^{N} \frac{1}{n^z}\Big| < \frac{1}{e}$. This then shows that $\zeta(z)$ is lower elementary on every $\{z \mid \mathrm{Re}(z) > s\}$, $(s > 1)$.

7 Holomorphic functions

Lemma 6. *Let $F(z) = \sum_{i=0}^{\infty} a_i z^i$ be a complex power series with radius of convergence ρ. Let $0 < b < \rho$ be an $\mathcal{F}$–real such that $(a_i b^i)_{i\in\mathbb{N}}$ is an $\mathcal{F}$–sequence of complex numbers. Then F restricted to the open disc $\{z\colon |z| < b\}$ belongs to $\mathcal{F}$.*

In order to develop a local-global principle for holomorphic functions, we use the following lemma.

Lemma 7 (Speed-Up Lemma). *Suppose $(a_n) \in \mathbb{C}$ is a bounded sequence and that $0 < b < 1$ is an $\mathcal{F}$–real. Then $(a_n b^n)$ is an $\mathcal{F}$–sequence if $(a_n b^{2n})$ is.*

Theorem 71 *Let F be a holomorphic function, defined on an open domain $D \subset \mathbb{C}$. Let a be an $\mathcal{F}$–complex number in D and let b be a positive $\mathcal{F}$–real smaller than the radius of convergence of $F(a+z) = \sum_{i=0}^{\infty} a_n z^n$. Then F is locally in $\mathcal{F}$ if and only if $(a_n b^n)$ is an $\mathcal{F}$–sequence.* □

Corollary 72 *Let F be holomorphic on a punctured disk $D_\bullet = \{z \mid 0 < |z| < r\}$. Then the following holds:*

1. *If 0 is a pole of F and F is $\mathcal{F}$ on some non–empty open subset of $D_\bullet$, then F is $\mathcal{F}$ on every proper punctured subdisc $D'_\bullet = \{z \mid 0 < |z| < r'\}$.*
2. *If 0 is an essential singularity of F, F is not lower elementary on $D_\bullet$.*
3. *Let $S = \{-n \mid n \in \mathbb{N}\}$ denote the set of poles of the Gamma function Γ. Γ is lower elementary on every set $\{z : |z| < r\} \setminus S$.*
4. *The Zeta function $\zeta(z)$ is lower elementary on every punctured disk $\{z \mid 0 < |z-1| < r\}$.*

Γ cannot be lower elementary on $\mathbb{C} \setminus S$ since $n!$ grows too fast. Similarly, ζ cannot be lower elementary on $\mathbb{C} \setminus \{1\}$ because ∞ is an essential singularity. However, we believe that Γ is elementary on $\mathbb{C} \setminus S$. and that ζ is elementary on $\mathbb{C} \setminus \{1\}$.

Corollary 73 *The set $\mathbb{C}_{\mathcal{F}} = \mathbb{R}_{\mathcal{F}}[\mathrm{i}]$ of $\mathcal{F}$–complex numbers is algebraically closed and closed under $\ln(z)$, $\exp(z)$, $\Gamma(z)$ and $\zeta(z)$.*

Note that $\mathbb{R}_{\mathcal{F}}[\mathrm{i}]$ is algebraically closed since $\mathbb{R}_{\mathcal{F}}$ is real closed by Corollary 32.

If $a_0, a_1, \ldots$ are $\mathbb{Q}$–linearly independent algebraic numbers, the exponentials $\exp(a_0), \exp(a_1) \ldots$. are lower elementary and algebraically independent by the Lindemann–Weierstra? Theorem. So the field of lower elementary complex numbers has infinite transcendence degree.

References

1. Paola D'Aquino. Local behaviour of the Chebyshev theorem in models of IΔ_0. *J. Symbolic Logic*, 57(1):12–27, 1992.
2. A. Grzegorczyk. Computable functionals. *Fund. Math.*, 42:168–202, 1955.
3. M. Kontsevich and D. Zagier. Periods. In *Mathematics Unlimited - 2001 and Beyond*, pages 771–808. Springer, Berlin, 2001.
4. Dimiter Skordev. Computability of real numbers by using a given class of functions in the set of the natural numbers. *MLQ Math. Log. Q.*, 48(suppl. 1):91–106, 2002. Dagstuhl Seminar on Computability and Complexity in Analysis, 2001.
5. Dimiter Skordev. On the subrecursive computability of several famous constants. *J.UCS*, 14(6):861–875, 2008.
6. Andreas Weiermann. Personal communication. 2008.
7. Klaus Weihrauch. *Computable analysis*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2000. An introduction.
8. Masahiko Yoshinaga. Periods and elementary real numbers. (`http://arxiv.org/math.AG/0805.0349v1`), 2008.

Computational Complexity in Analysis

Klaus Weihrauch

University of Hagen, Germany
Klaus.Weihrauch@FernUni-Hagen.de

Computability of functions on natural numbers or words (and via numberings on countable sets) has been defined in the 1930s. The basics of computational complexity have been developed around 1970. Meanwhile Computational Complexity, essentially for Turing machines, is a rich and deep theory which is still advancing.

For functions on the real numbers and other uncountable sets the development started at the same time but was much slower. Several partly non-equivalent definitions of computability on the real numbers have been introduced some of which allow refinement to computational complexity. Complexity of functions on the real numbers and, more generally, on other uncountable sets from Analysis has been studied to some extent but compared to discrete complexity theory the results are sparse.

While computability of an operator like the solution operator of a differential equation can be defined reasonably, seemingly still there is no satisfactory definition of its computational complexity. The question can be reduced to the definition of computational complexity for type-2 functionals on numbers, that is, for (computable) functions $f : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$.

In the talk existing partial solutions of type-2 complexity are discussed and a more satisfactory concept is suggested.

Ball arithmetic*

Joris van der Hoeven**

LIX, CNRS
École polytechnique
91128 Palaiseau Cedex
France

Abstract. The MATHEMAGIX project aims at the development of a "computer analysis" system, in which numerical computations can be done in a mathematically sound manner. A major challenge for such systems is to conceive algorithms which are both efficient, reliable and available at any working precision. In this paper, we survey several older and newer such algorithms. We mainly concentrate on the automatic and efficient computation of high quality error bounds, based on a variant of interval arithmetic which we like to call "ball arithmetic".
Keywords: Ball arithmetic, interval arithmetic, reliable computing, computable analysis
A.M.S. subject classification: 65G20, 03F60,65F99, 37-04

1 Introduction

Computer algebra systems are widely used today in order to perform mathematically correct computations with objects of algebraic or combinatorial nature. It is tempting to develop a similar system for analysis. On input, the user would specify the problem in a high level language, such as computable analysis [Wei00,BB85,Abe80,Tur36] or interval analysis [Moo66,AH83,Neu90,JKDW01,Kul08]. The system should then solve the problem in a certified manner. In particular, all necessary computations of error bounds must be carried out automatically.

There are several specialized systems and libraries which contain work in this direction. For instance, Taylor models have been used with success for the validated long term integration of dynamical systems [Ber98,MB96,MB04]. A fairly complete MATLAB interval arithmetic library for linear algebra and polynomial computations is INTLAB [Rum99b,Rum99a]. Another historical interval library, which continues to be developed is [ea67]. There exist several libraries for multiple precision arithmetic with correct or at least well specified rounding [HLRZ00,Hai95], as well as a library for multiple precision interval arithmetic [Rev01], and libraries for computable real numbers [M0,Lam07,BCC+06]. There are also libraries for very specific problems, such as the MPSOLVE library [BF00], which allows for the certified computation of the roots of multiple precision polynomials.

The MATHEMAGIX system [vdH+02b] intends to develop a new "computer analysis" system, distributed under a free licence and not depending on proprietary software such as MATLAB. Ultimately, we hope to cover most of the functionality present in the software mentioned above and add several new features, such as computable analytic functions [vdH07b] and transseries [vdH08b]. We also insist on performance and the possibility to easily switch between machine doubles and multiple precision arithmetic (without being penalized too much). In this paper, we will discuss several older and newer ideas for combining efficiency, reliability and multiple precision arithmetic (when necessary). Many algorithms are already present in the MATHEMAGIX system, or will be incorporated soon.

Most of the algorithms in this paper will be based on "ball arithmetic", which provides a systematic tool for the automatic computation of error bounds. In section 3, we provide a short introduction to this topic and describe its relation with computable analysis. Although ball arithmetic is really a variant of interval arithmetic, we will give several arguments in section 4 why we actually prefer balls over intervals for most applications. Roughly speaking, balls should be used for the reliable approximation

* This work has been supported by the ANR-09-JCJC-0098-01 MAGIX project, as well as a Digiteo 2009-36HD grant and Région Ile-de-France.
** vdhoeven@lix.polytechnique.fr *Web:* http://lix.polytechnique.fr/~vdhoeven

of numbers, whereas intervals are mainly useful for certified algorithms which rely on the subdivision of space. Although we will mostly focus on ball arithmetic in this paper, section 5 presents a rough overview of the other kinds of algorithms that are needed in a complete computer analysis system.

After basic ball arithmetic for real and complex numbers, the next challenge is to develop efficient algorithms for reliable linear algebra, polynomials, analytic functions, resolution of differential equations, etc. In sections 6 and 7 we will survey several basic techniques which can be used to implement efficient ball arithmetic for matrices and formal power series. Usually, there is a trade-off between efficiency and the quality of the obtained error bounds. One may often start with a very efficient algorithm which only computes rough bounds, or bounds which are only good in favourable well-conditioned cases. If the obtained bounds are not good enough, then we may switch to a more expensive and higher quality algorithm.

Of course, the main algorithmic challenge in the area of ball arithmetic is to reduce the overhead of the bound computations as much as possible with respect to the principal numeric computation. In favourable cases, this overhead indeed becomes negligible. For instance, for high working precisions p, the centers of real and complex balls are stored with the full precision, but we only need a small precision for the radii. Consequently, the cost of elementary operations ($+$, $-$, $\cdot$, exp, etc.) on balls is dominated by the cost of the corresponding operations on their centers. Similarly, crude error bounds for large matrix products can be obtained quickly with respect to the actual product computation, by using norm bounds for the rows and columns of the multiplicands; see (21).

Another algorithmic challenge is to use fast algorithms for the actual numerical computations on the centers of the balls. In particular, it is important to use existing high performance libraries, such as BLAS, LINPACK, etc., whenever this is possible [Rum99a]. Similarly, we should systematically rely on asymptotically efficient algorithms for basic arithmetic, such as fast integer and polynomial multiplication [KO63,CT65,SS71]. There are several techniques to achieve this goal:

1. The representations of objects should be chosen with care. For instance, should we rather work with ball matrices or matricial balls (see sections 6.4 and 7.1)?
2. If the result of our computation satisfies an equation, then we may first solve the equation numerically and only perform the error analysis at the end. In the case of matrix inversion, this method is known as Hansen's method; see section 6.2.
3. When considering a computation as a tree or dag, then the error tends to increase with the depth of the tree. If possible, algorithms should be designed so as to keep this depth small. Examples will be given in sections 6.4 and 7.5. Notice that this kind of algorithms are usually also more suitable for parallelization.

When combining the above approaches to the series of problems considered in this paper, we are usually able to achieve a constant overhead for sharp error bound computations. In favourable cases, the overhead becomes negligible. For particularly ill conditioned problems, we need a logarithmic overhead. It remains an open question whether we have been lucky or whether this is a general pattern.

In this paper, we will be easygoing on what is meant by "sharp error bound". Regarding algorithms as functions $f : \mathbb{R}^k \to \mathbb{R}^l; x \mapsto y = f(x)$, an error $\delta_x \in \mathbb{R}^k$ in the input automatically gives rise to an error $\delta_y \approx J_f(x)\delta_x$ in the output. When performing our computations with bit precision p, we have to consider that the input error δ_x is as least of the order of $2^{-p}|x| \in (\mathbb{R}^{\geqslant})^k$ (where $|x|_i = |x_i|$ for all i). Now given an error bound $|\delta_x| \leqslant \varepsilon_x$ for the input, an error bound $|\delta_y| \leqslant \varepsilon_y$ is considered to be sharp if $\varepsilon_y \approx \max_{|\delta_x| \leqslant \varepsilon_x} |J_f(x)\delta_x|$. More generally, condition numbers provide a similar device for measuring the quality of error bounds. A detailed investigation of the quality of the algorithms presented in this paper remains to be carried out. Notice also that the computation of optimal error bounds is usually NP-hard [KLRK97].

Of course, the development of asymptotically efficient ball arithmetic presupposes the existence of the corresponding numerical algorithms. This topic is another major challenge, which will not be addressed in this paper. Indeed, asymptotically fast algorithms usually suffer from lower numerical stability. Besides, switching from machine precision to multiple precision involves a huge overhead. Special techniques are required to reduce this overhead when computing with large compound objects, such as matrices or polynomials. It is also recommended to systematically implement single, double, quadruple and multiple precision versions of all algorithms (see [BHL00,BHL01] for a double-double and quadruple-double precision library). The MATHEMAGIX system has been designed so as to ease this task, since most current packages are C++ template libraries.

The aim of our paper is to provide a survey on how to implement an efficient library for ball arithmetic. Even though many of the ideas are classical in the interval analysis community, we do think that the paper sheds a new light on the topic. Indeed, we systematically investigate several issues which have received limited attention until now:

1. Using ball arithmetic instead of interval arithmetic, for a wide variety of center and radius types.
2. The trade-off between algorithmic complexity and quality of the error bounds.
3. The complexity of multiple precision ball arithmetic.

Throughout the text, we have attempted to provide pointers to existing literature, wherever appropriate. We apologize for possible omissions and would be grateful for any suggestions regarding existing literature.

2 Notations and classical facts

2.1 Floating point numbers and the IEEE 754 norm

We will denote by

$$\mathbb{D} = \mathbb{Z}2^{\mathbb{Z}} = \{m2^e : m, e \in \mathbb{Z}\}$$

the set of *dyadic numbers*. Given fixed bit precisions $p \in \mathbb{N}$ and $q \in \mathbb{N}$ for the unsigned mantissa (without the leading 1) and signed exponent, we also denote by

$$\mathbb{D}_{p,q} = \{m2^e : m, e \in \mathbb{Z}, |m| < 2^{p+1}, 1 - 2^{q-1} < e + p < 2^{q-1}\}$$

the corresponding set of floating point numbers. Numbers in $\mathbb{D}_{p,q}$ can be stored in $p + q + 1$ bit space and correspond to "ordinary numbers" in the IEEE 754 norm [ANS08]. For double precision numbers, we have $p = 51$ and $q = 12$. For multiple precision numbers, we usually take q to be the size of a machine word, say $q = 64$, and denote $\mathbb{D}_p = \mathbb{D}_{p,64}$.

The IEEE 754 norm also defines several special numbers. The most important ones are the infinities $\pm\infty$ and "not a number" NaN, which corresponds to the result of an invalid operation, such as $\sqrt{-2}$. We will denote

$$\mathbb{D}^*_{p,q} = \mathbb{D}_{p,q} \cup \{-\infty, +\infty, \mathrm{NaN}\}.$$

Less important details concern the specification of signed zeros ± 0 and several possible types of NaNs. For more details, we refer to [ANS08].

The other main feature of the IEEE 754 is that it specifies how to round results of operations which cannot be performed exactly. There are four basic rounding modes $\uparrow$ (upwards), $\downarrow$ (downwards), $\updownarrow$ (nearest) and 0 (towards zero). Assume that we have an operation $f : U \to \mathbb{R}$ with $U \subseteq \mathbb{R}^k$. Given $x \in U \cap \mathbb{D}^k_{p,q}$ and $y = f(x)$, we define $f^\uparrow(x)$ to be the smallest number $\tilde{y}$ in $\mathbb{D}^*_{p,q}$, such that $\tilde{y} \geqslant y$. More generally, we will use the notation $(f(x)+y)^\uparrow = f^\uparrow(x)+^\uparrow y$ to indicate that all operations are performed by rounding upwards. The other rounding modes are specified in a similar way. One major advantage of IEEE 754 arithmetic is that it completely specifies how basic operations are done, making numerical programs behave exactly in the same way on different architectures. Most current microprocessors implement the IEEE 754 norm for single and double precision numbers. A conforming multiple precision implementation also exists [HLRZ00].

2.2 Classical complexity results

Using Schönhage-Strassen multiplication [SS71], it is classical that the product of two p-bit integers can be computed in time $I(n) = O(n \log n \log\log n)$. A recent algorithm by Fürer [Für07] further improves this bound to $I(n) = O(n \log n 2^{O(\log^* n)})$, where $\log^*$ denotes the iterator of the logarithm (we have $\log^* n = \log^* \log n + 1$). Other algorithms, such as division two $\leqslant n$-bit integers, have a similar complexity $O(I(n))$. Given k prime numbers $p_1 < \cdots < p_k$ and a number $0 \leqslant q < p_1 \cdots p_k$, the computation of $q \bmod p_i$ for $i = 1, \ldots, k$ can be done in time $O(I(n) \log n)$ using binary splitting [GG02, Theorem 10.25], where $n = \log(p_1 \cdots p_k)$. It is also possible to reconstruct q from the remainders $q \bmod p_i$ in time $O(I(n) \log n)$.

Let $\mathbb{K}$ be an effective ring, in the sense that we have algorithms for performing the ring operations in $\mathbb{K}$. If $\mathbb{K}$ admits 2^p-th roots of unity for $2^p \geqslant n$, then it is classical [CT65] that the product of two polynomials $P, Q \in \mathbb{K}[z]$ with $\deg PQ < n$ can be computed using $O(n \log n)$ ring operations in $\mathbb{K}$, using the fast Fourier transform. For general rings, this product can be computed using $M(n) = O(n \log n \log\log n)$ operations [CK91], using a variant of Schönhage-Strassen multiplication.

A formal power series $f \in \mathbb{K}[[z]]$ is said to be *computable*, if there exists an algorithm which takes $n \in \mathbb{N}$ on input and which computes $f_n \in \mathbb{K}$. We denote by $\mathbb{K}[[z]]^{\text{com}}$ the set of computable power series. Many simple operations on formal power series, such as multiplication, division, exponentiation, etc., as well as the resolution of algebraic differential equations can be done up till order n using a similar time complexity $O(M(n))$ as polynomial multiplication [BK78,BCO+06,vdH06b]. Alternative *relaxed* multiplication algorithms of time complexity $R(n) = O(M(n))$ are given in [vdH02a,vdH07a]. In this case, the coefficients $(fg)_n$ are computed gradually and $(fg)_n$ is output as soon as $f_0, \ldots, f_n$ and $g_0, \ldots, g_n$ are known. This strategy is often most efficient for the resolution of implicit equations.

Let $\mathbb{K}^{m \times n}$ denote the set of $m \times n$ matrices with entries in $\mathbb{K}$. Given two $n \times n$ matrices $M, N \in \mathbb{K}^{n \times n}$, the naive algorithm for computing MN requires $O(n^\omega)$ ring operations with $\omega = 3$. The exponent ω has been reduced by Strassen [Str69] to $\omega = \log 7 / \log 2$. Using more sophisticated techniques, ω can be further reduced to $\omega < 2.376$ [Pan84,CW87]. However, this exponent has never been observed in practice.

2.3 Interval and ball notations

Given $x < y$ in a totally ordered set R, we will denote by $[x, y]$ the closed interval

$$[x, y] = \{z \in R : x \leqslant z \leqslant y\}. \tag{1}$$

Assume now that R is a totally ordered field and consider a normed vector space V over R. Given $c \in V$ and $r \in R^{\geqslant} = \{r \in \mathbb{R} : r \geqslant 0\}$, we will denote by

$$\mathcal{B}(c, r) = \{x \in V : \|x - c\| \leqslant r\} \tag{2}$$

the closed ball with center c and radius r. Notice that we will never work with open balls in what follows. We denote the set of all balls of the form (2) by $\mathcal{B}(V, R)$. Given $x \in \mathcal{B}(V, R)$, we will denote by x_c and x_r the center resp. radius of x.

We will use the standard euclidean norm

$$\|x\| = \sqrt{\frac{|x_1|^2 + \cdots + |x_n|^2}{n}}$$

as the default norm on $\mathbb{R}^n$ and $\mathbb{C}^n$. Occasionally, we will also consider the max-norm

$$\|x\|_\infty = \max\{|x_1|, \ldots, |x_n|\}.$$

For matrices $M \in \mathbb{R}^{m \times n}$ or $M \in \mathbb{C}^{m \times n}$, the default operator norm is defined by

$$\|M\| = \max_{\|x\|=1} \|Mx\|. \tag{3}$$

Occasionally, we will also consider the max-norm

$$\|M\|_\infty = \max_{i,j} |M_{i,j}|,$$

which satisfies

$$\|M\|_\infty \leqslant \|M\| \leqslant n \|M\|_\infty.$$

We also define the max-norm for polynomials $P \in \mathbb{R}[x]$ or $P \in \mathbb{C}[x]$ by

$$\|P\|_\infty = \max |P_i|.$$

For power series $f \in \mathbb{R}[[z]]$ or $f \in \mathbb{C}[[z]]$ which converge on the compact disk $\mathcal{B}(0, r)$, we define the norm

$$\|f\|_r = \max_{|z| \leqslant r} |f(z)|. \tag{4}$$

After rescaling $f(z) \mapsto f(rz)$, we will usually work with the norm $\| \cdot \| = \| \cdot \|_1$.

In some cases, it may be useful to generalize the concept of a ball and allow for radii in partially ordered rings. For instance, a ball $X \in \mathcal{B}(\mathbb{R}^n, \mathbb{R}^n)$ stands for the set

$$X = \{x \in \mathbb{R}^n | \forall 1 \leqslant i \leqslant n, |x_i - (X_c)_i| \leqslant (X_r)_i\}.$$

The sets $\mathcal{B}(\mathbb{R}^{m\times n}, \mathbb{R}^{m\times n})$, $\mathcal{B}(\mathbb{R}[x], \mathbb{R}[x])$, $\mathcal{B}(\mathbb{R}[[x]], \mathbb{R}[[x]])$, etc. are defined in a similar component wise way. Given $x \in \mathbb{R}^n$, it will also be convenient to denote by $|x|$ the vector with $|x|_i = |x_i|$. For matrices $M \in \mathbb{R}^{m\times n}$, polynomials $P \in \mathbb{R}[x]$ and power series $f \in \mathbb{R}[[x]]$, we define $|M|$, $|P|$ and $|f|$ similarly.

3 Balls and computable real numbers

3.1 Ball arithmetic

Let $\mathbb{V}$ be a normed vector space over $\mathbb{R}$ and recall that $\mathcal{B}(\mathbb{V}, \mathbb{R})$ stands for the set of all balls with centers in $\mathbb{V}$ and radii in $\mathbb{R}$. Given an operation $\varphi : \mathbb{V}^k \to \mathbb{V}$, the operation is said to lift to an operation

$$f^\circ : \mathcal{B}(\mathbb{V}, \mathbb{R})^k \to \mathcal{B}(\mathbb{V}, \mathbb{R}),$$

if we have

$$f^\circ(X_1, \ldots, X_k) \supseteq \{f(x_1, \ldots, x_k) : x_1 \in X_1, \ldots, x_k \in X_k\},$$

for all $X_1, \ldots, X_k \in \mathcal{B}(\mathbb{V}, \mathbb{R})$. For instance, both the addition $+ : \mathbb{V}^2 \to \mathbb{V}$ and subtraction $- : \mathbb{V}^2 \to \mathbb{V}$ admits lifts

$$\mathcal{B}(x, r) +^\circ \mathcal{B}(y, s) = \mathcal{B}(x + y, r + s) \tag{5}$$
$$\mathcal{B}(x, r) -^\circ \mathcal{B}(y, s) = \mathcal{B}(x - y, r + s). \tag{6}$$

Similarly, if $\mathbb{V}$ is a normed algebra, then the multiplication lifts to

$$\mathcal{B}(x, r) \cdot^\circ \mathcal{B}(y, s) = \mathcal{B}(xy, (|x| + r)s + r|y|) \tag{7}$$

The lifts $+^\circ$, $-^\circ$, $\cdot^\circ$, etc. are also said to be the ball arithmetic counterparts of addition, subtractions, multiplication, etc.

Ball arithmetic is a systematic way for the computation of error bounds when the input of a numerical operation is known only approximately. These bounds are usually not sharp. For instance, consider the mathematical function $f : x \in \mathbb{R} \mapsto x - x$ which evaluates to zero everywhere, even if x is only approximately known. However, taking $x = \mathcal{B}(2, 0.001)$, we have $x -^\circ x = \mathcal{B}(0, 0.002) \neq \mathcal{B}(0, 0)$. This phenomenon is known as *overestimation*. In general, ball algorithms have to be designed carefully so as to limit overestimation.

In the above definitions, $\mathbb{R}$ can be replaced by a subfield $R \subseteq \mathbb{R}$, $\mathbb{V}$ by an R-vector space $V \subseteq \mathbb{V}$, and the domain of f by an open subset of $\mathbb{V}^k$. If V and R are *effective* in the sense that we have algorithms for the additions, subtractions and multiplications in V and R, then basic ball arithmetic in $\mathcal{B}(V, R)$ is again effective. If we are working with finite precision floating point numbers in $\mathbb{D}_{p,q}$ rather than a genuine effective subfield R, we will now show how to adapt the formulas (5), (6) and (7) in order to take into account rounding errors; it may also be necessary to allow for an infinite radius in this case.

3.2 Finite precision ball arithmetic

Let us detail what needs to be changed when using IEEE conform finite precision arithmetic, say $V = R = \mathbb{D}_{p,q}$. We will denote

$$\mathbb{B} = \mathcal{B}(\mathbb{D}, \mathbb{D})$$
$$\mathbb{B}_{p,q} = \mathcal{B}(\mathbb{D}_{p,q}, \mathbb{D}_{p,q})$$
$$\mathbb{B}[\mathrm{i}]_{p,q} = \mathcal{B}(\mathbb{D}_{p,q}[\mathrm{i}], \mathbb{D}_{p,q}).$$

When working with multiple precision numbers, it usually suffices to use low precision numbers for the radius type. Recalling that $\mathbb{D}_p = \mathbb{D}_{p,64}$, we will therefore denote

$$\begin{aligned}\mathbb{B}_p &= \mathcal{B}(\mathbb{D}_p, \mathbb{D}_{64})\\ \mathbb{B}[\mathrm{i}]_p &= \mathcal{B}(\mathbb{D}_p[\mathrm{i}], \mathbb{D}_{64}).\end{aligned}$$

We will write $\epsilon = \epsilon_p = 2^{-p}$ for the machine accuracy and $\eta = \eta_{p,q} = 2^{2-2^q-p}$ for the smallest representable positive number in $\mathbb{D}_{p,q}$.

Given an operation $f : \mathbb{R}^k \to \mathbb{R}$ as in section 3.1, together with balls $X_i = \mathcal{B}(x_i, r_i)$, it is natural to compute the center y of

$$\mathcal{B}(y, s) = f^\circ(X_1, \ldots, X_s)$$

by rounding to the nearest:

$$y = f^\updownarrow(x_1, \ldots, x_k). \tag{8}$$

One interesting point is that the committed error

$$\delta = |y - f(x_1, \ldots, x_k)|$$

does not really depend on the operation f itself: we have the universal upper bound

$$\begin{aligned}\delta &\leqslant \Delta(y)\\ \Delta(y) &:= (|y| +^\uparrow \eta) \cdot^\uparrow \epsilon.\end{aligned} \tag{9}$$

It would be useful if this *adjustment function* Δ were present in the hardware.

For the computation of the radius s, it now suffices to use the sum of $\Delta(y)$ and the theoretical bound formulas for the infinite precision case. For instance,

$$\mathcal{B}(x, r) +^\circ \mathcal{B}(y, s) = \mathcal{B}^\Delta(x +^\updownarrow y, r +^\uparrow s) \tag{10}$$

$$\mathcal{B}(x, r) -^\circ \mathcal{B}(y, s) = \mathcal{B}^\Delta(x -^\updownarrow y, r +^\uparrow s) \tag{11}$$

$$\mathcal{B}(x, r) \cdot^\circ \mathcal{B}(y, s) = \mathcal{B}^\Delta(x \cdot^\updownarrow y, [(|x| + r)s + r|y|]^\uparrow), \tag{12}$$

where $\mathcal{B}^\Delta$ stands for the "adjusted constructor"

$$\mathcal{B}^\Delta(y, s) = \mathcal{B}(y, s +^\uparrow \Delta(y)).$$

The approach readily generalizes to other "normed vector spaces" $\mathbb{V}$ over $\mathbb{D}_{p,q}$, as soon as one has a suitable rounded arithmetic in $\mathbb{V}$ and a suitable adjustment function Δ attached to it.

Notice that $\Delta(y) = \infty$, if $y = \infty$ or y is the largest representable real number in $\mathbb{D}_{p,q}$. Consequently, the finite precision ball computations naturally take place in domains of the form $\mathbb{B}^*_{p,q} = \mathcal{B}(\mathbb{D}^*_{p,q}, \mathbb{D}^*_{p,q})$ rather than $\mathbb{B}_{p,q}$. Of course, balls with infinite radius carry no useful information about the result. In order to ease the reading, we will assume the absence of overflows in what follows, and concentrate on computations with ordinary numbers in $\mathbb{B}_{p,q}$. We will only consider infinities if they are used in an essential way during the computation.

Similarly, if we want ball arithmetic to be a natural extension of the IEEE 754 norm, then we need an equivalent of NaN. One approach consists of introducing a NaB (not a ball) object, which could be represented by $\mathcal{B}(\mathrm{NaN}, \mathrm{NaN})$ or $\mathcal{B}(0, -\infty)$. A ball function f° returns NaB if f returns NaN for one selection of members of the input balls. For instance, $\mathrm{sqrt}^\circ(\mathcal{B}(1, 3)) = \mathrm{NaB}$. An alternative approach consists of the attachment of an additional flag to each ball object, which signals a possible invalid outcome. Following this convention, $\mathrm{sqrt}^\circ(\mathcal{B}(1, 3))$ yields $\mathcal{B}(1, 1)^{\mathrm{NaN}}$.

3.3 Implementation details

Using the formulas from the previous section, it is relatively straightforward to implement ball arithmetic as a C++ template library, as we have done in the Mathemagix system. However, in the case of multiple precision arithmetic this is far from optimal. Let us discuss several possible optimizations:

1. Multiple precision libraries such as MPFR [HLRZ00] suffer from a huge overhead when it comes to moderate (e.g. quadruple) precision computations. Since the radii are always stored in low precision, it is recommended to inline all computations on the radii. In the case of multiplication, this divides the number of function calls by four.
2. When computing with complex numbers $z \in \mathbb{D}_p[\mathrm{i}]$, one may again save several function calls. Moreover, it is possible to regard z as an element of $\mathbb{Z}[\mathrm{i}]2^{\mathbb{Z}}$ rather than $(\mathbb{Z}2^{\mathbb{Z}})[\mathrm{i}]$, i.e. use a single exponent for both the real and imaginary parts of z. This optimization reduces the time spent on exponent computations and mantissa normalizations.
3. Consider a ball $\mathcal{B}(c, r) \in \mathbb{B}_p$ and recall that $c \in \mathbb{D}_p$, $r \in \mathbb{D}_{64}$. If $|c| < 2^{p-64} r$, then the $\lfloor \log_2 r + p - \log_2 |c| - 64 \rfloor$ least significant binary digits of c are of little interest. Hence, we may replace c by its closest approximation in $\mathbb{D}_{p'}$, with $p' = \lceil \log_2 |c| + 64 - \log_2 r \rceil$, and reduce the working precision to p'. Modulo slightly more work, it is also possible to share the exponents of the center and the radius.
4. If we don't need large exponents for our multiple precision numbers, then it is possible to use machine doubles $\mathbb{D}_{51,12}$ as our radius type and further reduce the overhead of bound computations.

When combining the above optimizations, it can be hoped that multiple precision ball arithmetic can be implemented almost as efficiently as standard multiple precision arithmetic. However, this requires a significantly higher implementation effort.

3.4 Computable numbers

Given $x \in \mathbb{R}$, $\tilde{x} \in \mathbb{D}$ and $\varepsilon \in \mathbb{D}^> = \{\varepsilon \in \mathbb{D} : \varepsilon > 0\}$, we say that $\tilde{x}$ is an *ε-approximation* of x if $|\tilde{x} - x| \leqslant \varepsilon$. A real number $x \in \mathbb{R}$ is said to be *computable* [Tur36,Grz57,Wei00] if there exists an *approximation algorithm* which takes an absolute tolerance $\varepsilon \in \mathbb{D}^>$ on input and which returns an ε-approximation of x. We denote by $\mathbb{R}^{\mathrm{com}}$ the set of computable real numbers. We may regard $\mathbb{R}^{\mathrm{com}}$ as a data type, whose instances are represented by approximation algorithms (this is also known as the *Markov representation* [Wei00, Section 9.6]).

In practice, it is more convenient to work with so called ball approximation algorithms: a real number is computable if and only if it admits a *ball approximation algorithm*, which takes a working precision $p \in \mathbb{N}$ on input and returns a *ball approximation* $\mathcal{B}(c_p, r_p) \in \mathbb{B}$ with $x \in \mathcal{B}(c_p, r_p)$ and $\lim_{p \to \infty} r_p = 0$. Indeed, assume that we have a ball approximation algorithm. In order to obtain an ε-approximation, it suffices to compute ball approximations at precisions $p = 32, 64, 128, \ldots$ which double at every step, until $r_p \leqslant \varepsilon$. Conversely, given an approximation algorithm $\check{x} : \mathbb{D}^> \to \mathbb{D}$ of $x \in \mathbb{R}^{\mathrm{com}}$, we obtain a ball approximation algorithm $p \mapsto \mathcal{B}(c_p, r_p)$ by taking $r_p = 2^{-p}$ and $c_p = \check{x}(r_p)$.

Given $x, y \in \mathbb{R}^{\mathrm{com}}$ with ball approximation algorithms $\check{x}, \check{y} : \mathbb{N} \mapsto \mathbb{B}$, we may compute ball approximation algorithms for $x + y, x - y, xy$ simply by taking

$$\begin{aligned} (x \mathbin{\check{+}} y)(p) &= \check{x}(p) +^\circ \check{y}(p) \\ (x \mathbin{\check{-}} y)(p) &= \check{x}(p) -^\circ \check{y}(p) \\ (x \check{y})(p) &= \check{x}(p) \cdot^\circ \check{y}(p). \end{aligned}$$

More generally, assuming a good library for ball arithmetic, it is usually easy to write a wrapper library with the corresponding operations on computable numbers.

From the efficiency point of view, it is also convenient to work with ball approximations. Usually, the radius r_p satisfies

$$\log_2 r_p = -p + o(1),$$

or at least

$$\log_2 r_p = -cp + o(1),$$

for some $c \in \mathbb{Q}^>$. In that case, doubling the working precision p until a sufficiently good approximation is found is quite efficient. An even better strategy is to double the "expected running time" at every step [vdH06a,KR06]. Yet another approach will be described in section 3.6 below.

The concept of computable real numbers readily generalizes to more general normed vector spaces. Let $\mathbb{V}$ be a normed vector space and let $\mathbb{V}^{\mathrm{dig}}$ be an effective subset of *digital points* in $\mathbb{V}$, i.e. the elements

of $\mathbb{V}^{\mathrm{dig}}$ admit a computer representation. For instance, if $\mathbb{V} = \mathbb{R}$, then we take $\mathbb{V}^{\mathrm{dig}} = \mathbb{D}$. Similarly, if $\mathbb{V} = \mathbb{R}^{m\times n}$ is the set of real $m \times n$ matrices, with one of the matrix norms from section 2.3, then it is natural to take $\mathbb{V}^{\mathrm{dig}} = (\mathbb{R}^{\mathrm{dig}})^{m\times n} = \mathbb{D}^{m\times n}$. A point $x \in \mathbb{V}$ is said to be *computable*, if it admits an *approximation algorithm* which takes $\varepsilon \in \mathbb{D}^{>}$ on input, and returns an ε*-approximation* $\tilde{x} \in \mathbb{V}^{\mathrm{dig}}$ of x (satisfying $\|\tilde{x} - x\| \leqslant \varepsilon$, as above).

3.5 Asymmetric computability

A real number x is said to be *left computable* if there exists an algorithm for computing an increasing sequence $\check{x} : \mathbb{N} \mapsto \mathbb{D}; n \mapsto \check{x}_n$ with $\lim_{n\to\infty} \check{x}_n = x$ (and $\check{x}$ is called a left approximation algorithm). Similarly, x is said to be *right computable* if $-x$ is left computable. A real number is computable if and only if it is both left and right computable. Left computable but non computable numbers occur frequently in practice and correspond to "computable lower bounds" (see also [Wei00,vdH07b]).

We will denote by $\mathbb{R}^{\mathrm{lcom}}$ and $\mathbb{R}^{\mathrm{rcom}}$ the data types of left and right computable real numbers. It is convenient to specify and implement algorithms in computable analysis in terms of these data types, whenever appropriate [vdH07b]. For instance, we have computable functions

$$\begin{aligned} + &: \mathbb{R}^{\mathrm{lcom}} \times \mathbb{R}^{\mathrm{lcom}} \to \mathbb{R}^{\mathrm{lcom}} \\ - &: \mathbb{R}^{\mathrm{lcom}} \times \mathbb{R}^{\mathrm{rcom}} \to \mathbb{R}^{\mathrm{lcom}} \\ &\vdots \end{aligned}$$

More generally, given a subset $S \subseteq \mathbb{R}$, we say that $x \in S$ is left computable in S if there exists a left approximation algorithm $\check{x} : \mathbb{N} \to S$ for x. We will denote by S^{lcom} and S^{rcom} the data types of left and computable numbers in S, and define $S^{\mathrm{com}} = S^{\mathrm{lcom}} \cap S^{\mathrm{rcom}}$.

Identifying the type of boolean numbers $\mathbb{T}$ with $\{0, 1\}$, we have $\mathbb{T}^{\mathrm{lcom}} = \mathbb{T}^{\mathrm{rcom}} = \mathbb{T}$ as sets, but *not* as data types. For instance, it is well known that equality is non computable for computable real numbers [Tur36]. Nevertheless, equality *is* "ultimately computable" in the sense that there exists a computable function

$$=: \mathbb{R}^{\mathrm{com}} \times \mathbb{R}^{\mathrm{com}} \to \mathbb{T}^{\mathrm{rcom}}.$$

Indeed, given $x, y \in \mathbb{R}^{\mathrm{com}}$ with ball approximation algorithms $\check{x}$ and $\check{y}$, we may take

$$(x \doteq y)_n = \begin{cases} 1 \text{ if } (\check{x}_0 \cap \check{y}_0 \cap \cdots \cap \check{x}_n \cap \check{y}_n) \neq \varnothing \\ 0 \text{ otherwise} \end{cases}$$

Similarly, the ordering relation $\leqslant$ is ultimately computable.

This asymmetric point of view on equality testing also suggest a semantics for the relations $=$, $\leqslant$, etc. on balls. For instance, given balls $x, y \in \mathbb{B}$, it is natural to take

$$\begin{aligned} x = y &\rightsquigarrow x \cap y \neq \varnothing \\ x \neq y &\rightsquigarrow x \cap y = \varnothing \\ x \leqslant y &\rightsquigarrow \exists a \in x, b \in y, a \leqslant b \\ &\vdots \end{aligned}$$

These definitions are interesting if balls are really used as successive approximations of a real number. An alternative application of ball arithmetic is for modeling non-deterministic computations: the ball models a set of possible values and we are interested in the set of possible outcomes of an algorithm. In that case, the natural return type of a relation on balls becomes a "boolean ball". In the area of interval analysis, this second interpretation is more common [ANS09].

Remark 1. We notice that the notions of computability and asymmetric computability do not say anything about the speed of convergence. In particular, it is usually impossible to give useful complexity bounds for algorithms which are based on these mere concepts. In the case of asymmetric computability, there even do not exist any recursive complexity bounds, in general.

3.6 Lipschitz ball arithmetic

Given a computable function $f : \mathbb{R}^{\mathrm{com}} \to \mathbb{R}^{\mathrm{com}}$, $x \in \mathbb{R}^{\mathrm{com}}$ and $\varepsilon \in \mathbb{D}^{>}$, let us return to the problem of efficiently computing an approximation $\tilde{y} \in \mathbb{D}$ of $y = f(x)$ with $|\tilde{y} - y| < \varepsilon$. In section 3.4, we suggested to compute ball approximations of y at precisions which double at every step, until a sufficiently precise approximation is found. This computation involves an implementation $f^{\circ} : \mathbb{B} \to \mathbb{B}$ of f on the level of balls, which satisfies

$$f^{\circ}(X) \supseteq \{f(X) : x \in X\}, \tag{13}$$

for every ball $X \in \mathbb{B}$. In practice, f is often differentiable, with $f'(x) \neq 0$. In that case, given a ball approximation X of x, the computed ball approximation $Y = f^{\circ}(X)$ of y typically has a radius

$$Y_r \sim |f'(x)| X_r, \tag{14}$$

for $X_r \to 0$. This should make it possible to directly predict a sufficient precision at which $Y_r \leqslant \varepsilon$. The problem is that (14) needs to be replaced by a more reliable relation. This can be done on the level of ball arithmetic itself, by replacing the usual condition (13) by

$$\begin{gathered} f^{\circ}(X) \supseteq \mathcal{B}(f(X_c), MX_r) \\ M = \sup_{x \in X} |f'(x)|. \end{gathered} \tag{15}$$

Similarly, the multiplication of balls is carried out using

$$\mathcal{B}(x, r) \cdot^{\circ} \mathcal{B}(y, s) = \mathcal{B}(xy, (|x| + r)s + (|y| + s)r). \tag{16}$$

instead of (7). A variant of this kind of "Lipschitz ball arithmetic" has been implemented in [M0]. Although a constant factor is gained for high precision computations at regular points x, the efficiency deteriorates near singularities (i.e. the computation of$\sqrt{0}$).

4 Balls versus intervals

In the area of reliable computation, interval arithmetic has for long been privileged with respect to ball arithmetic. Indeed, balls are often regarded as a more or less exotic variant of intervals, based on an alternative midpoint-radius representation. Historically, interval arithmetic is also preferred in computer science because it is easy to implement if floating point operations are performed with correct rounding. Since most modern microprocessors implement the IEEE 754 norm, this point of view is well supported by hardware.

Not less historically, the situation in mathematics is inverse: whereas intervals are the standard in computer science, balls are the standard in mathematics, since they correspond to the traditional ε-δ-calculus. Even in the area of interval analysis, one usually resorts (at least implicitly) to balls for more complex computations, such as the inversion of a matrix [HS67,Moo66]. Indeed, balls are more convenient when computing error bounds using perturbation techniques. Also, we have a great deal of flexibility concerning the choice of a norm. For instance, a vectorial ball is not necessarily a Cartesian product of one dimensional balls.

In this section, we will give a more detailed account on the respective advantages and disadvantages of interval and ball arithmetic.

4.1 Standardization

One advantage of interval arithmetic is that the IEEE 754 norm suggests a natural and standard implementation. Indeed, let f be a real function which is increasing on some interval I. Then the natural interval lift f of f is given by

$$f([l, h]) = [f^{\downarrow}(l), f^{\uparrow}(h)].$$

This implementation has the property that $f([l,h])$ is the smallest interval with end-points in $\mathbb{D}_{p,q} \cup \{\pm\infty\}$, which satisfies

$$f([l,h]) \supseteq \{f(x) : x \in [l,h]\}.$$

For not necessarily increasing functions f this property can still be used as a requirement for the "standard" implementation of f. For instance, this leads to the following implementation of the cosine function on intervals:

$$\cos([l,h]) = \begin{cases} [\cos^{\downarrow} l, \cos^{\uparrow} h] & \text{if } \lfloor l/2\pi \rfloor = \lfloor h/2\pi \rfloor \in 2\mathbb{Z} - 1 \\ [\cos^{\downarrow} h, \cos^{\uparrow} l] & \text{if } \lfloor l/2\pi \rfloor = \lfloor h/2\pi \rfloor \in 2\mathbb{Z} \\ [\min(\cos^{\downarrow} l, \cos^{\downarrow} h), 1] & \text{if } \lfloor l/2\pi \rfloor = \lfloor h/2\pi \rfloor - 1 \in 2\mathbb{Z} - 1 \\ [-1, \max(\cos^{\uparrow} l, \cos^{\uparrow} h)] & \text{if } \lfloor l/2\pi \rfloor = \lfloor h/2\pi \rfloor - 1 \in 2\mathbb{Z} \\ [-1, 1] & \text{if } \lfloor l/2\pi \rfloor < \lfloor h/2\pi \rfloor - 1 \end{cases}$$

Such a standard implementation of interval arithmetic has the convenient property that programs will execute in the same way on any platform which conforms to the IEEE 754 standard.

By analogy with the above approach for standardized interval arithmetic, we may standardize the ball implementation f° of f by taking

$$f^\circ(\mathcal{B}(c,r)) = \mathcal{B}(f^{\updownarrow}(c), s),$$

where the radius s is smallest in $\mathbb{D}_{p,q} \cup \{+\infty\}$ with

$$\mathcal{B}(f^{\updownarrow}(c), s) \supseteq \{f(x) : x \in \mathcal{B}(c,r)\}.$$

Unfortunately, the computation of such an optimal s is not always straightforward. In particular, the formulas (10), (11) and (12) do not necessarily realize this tightest bound. In practice, it might therefore be better to achieve standardization by fixing once and for all the formulas by which ball operations are performed. Of course, more experience with ball arithmetic is required before this can happen.

4.2 Practical efficiency

The respective efficiencies of interval and ball arithmetic depend on the precision at which we are computing. For high precisions and most applications, ball arithmetic has the advantage that we can still perform computations on the radius at single precision. By contrast, interval arithmetic requires full precision for operations on both end-points. This makes ball arithmetic twice as efficient at high precisions.

When working at machine precision, the efficiencies of both approaches essentially depend on the hardware. *A priori*, interval arithmetic is better supported by current computers, since most of them respect the IEEE 754 norm, whereas the function Δ from (9) usually has to be implemented by hand. However, changing the rounding mode is often highly expensive (over hundred cycles). Therefore, additional gymnastics may be required in order to always work with respect to a fixed rounding mode. For instance, if $\uparrow$ is our current rounding mode, then we may take

$$x +^{\downarrow} y = -((-x) +^{\uparrow} (-y)),$$

since the operation $x \mapsto -x$ is always exact (i.e. does not depend on the rounding mode). As a consequence, interval arithmetic becomes slightly more expensive. By contrast, when releasing the condition that centers of balls are computed using rounding to the nearest, we may replace (8) by

$$y = f^{\uparrow}(x_1, \ldots, x_k) \tag{17}$$

and (9) by

$$\Delta(y) := (|y| +^{\uparrow} \eta) \cdot^{\uparrow} (2\epsilon).$$

Hence, ball arithmetic already allows us to work with respect to a fixed rounding mode. Of course, using (17) instead of (8) does require to rethink the way ball arithmetic should be standardized.

An alternative technique for avoiding changes in rounding mode exists when performing operations on compound types, such as vectors or matrices. For instance, when adding two vectors, we may first add all lower bounds while rounding downwards and next add the upper bounds while rounding upwards. Unfortunately, this strategy becomes more problematic in the case of multiplication, because different rounding modes may be needed depending on the signs of the multiplicands. As a consequence, matrix operations tend to require many conditional parts of code when using interval arithmetic, with increased probability of breaking the processor pipeline. On the contrary, ball arithmetic highly benefits from parallel architecture and it is easy to implement ball arithmetic for matrices on top of existing libraries: see [Rum99a] and section 6 below.

4.3 Quality

Besides the efficiency of ball and interval arithmetic for basic operations, it is also important to investigate the quality of the resulting bounds. Indeed, there are usually differences between the sets which are representable by balls and by intervals. For instance, when using the extended IEEE 754 arithmetic with infinities, then it is possible to represent $[1, \infty]$ as an interval, but not as a ball.

These differences get more important when dealing with complex numbers or compound types, such as matrices. For instance, when using interval arithmetic for reliable computations with complex numbers, it is natural to enclose complex numbers by rectangles $X + Y\mathrm{i}$, where X and Y are intervals. For instance, the complex number $z = 1 + \mathrm{i}$ may be enclosed by

$$z \in [1 - \varepsilon, 1 + \varepsilon] + [1 - \varepsilon, 1 + \varepsilon]\mathrm{i},$$

for some small number ε. When using ball arithmetic, we would rather enclose z by

$$z \in \mathcal{B}(1 + \mathrm{i}, \varepsilon).$$

Now consider the computation of $u = z^2$. The computed rectangular and ball enclosures are given by

$$\begin{aligned} u &\in [-2\varepsilon, 2\varepsilon] + [2 - 2\varepsilon, 2 + 2\varepsilon]\mathrm{i} + o(\varepsilon) \\ u &\in \mathcal{B}(2, \sqrt{2}\varepsilon) + o(\varepsilon). \end{aligned}$$

Consequently, ball arithmetic yields a much better bound, which is due to the fact that multiplication by $1 + \mathrm{i}$ turns the rectangular enclosure by 45 degrees, leading to an overestimation by a factor $\sqrt{2}$ when re-enclosing the result by a horizontal rectangle (see figure 1).

This is one of the simplest instances of the *wrapping effect* [Moo66]. For similar reasons, one may prefer to compute the square of the matrix

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{18}$$

in $\mathcal{B}(\mathbb{D}^{2\times 2}, \mathbb{D})$ rather than $\mathcal{B}(\mathbb{D}, \mathbb{D})^{2\times 2}$, while using the operator norm (3) for matrices. This technique highlights another advantage of ball arithmetic: we have a certain amount of flexibility regarding the choice of the radius type. By choosing a simple radius type, we do not only reduce the wrapping effect, but also improve the efficiency: when computing with complex balls in $\mathbb{B}[\mathrm{i}]$, we only need to bound one radius instead of two for every basic operation. More precisely, we replace (8) and (9) by

$$\begin{aligned} y &= f^{\updownarrow}(x_1, \ldots, x_k) \\ &= (\operatorname{Re} f)^{\updownarrow}(x_1, \ldots, x_k) + (\operatorname{Im} f)^{\updownarrow}(x_1, \ldots, x_k)\mathrm{i} \\ \Delta(y) &:= (\operatorname{abs}^{\uparrow}(y) +^{\uparrow} \eta) \cdot^{\uparrow} (\sqrt{2}\epsilon). \end{aligned}$$

On the negative side, generalized norms may be harder to compute, even though a rough bound often suffices (e.g. replacing $\operatorname{abs}^{\uparrow}(y)$ by $|\operatorname{Re} y| +^{\uparrow} |\operatorname{Im} y|$ in the above formula). In the case of matricial balls, a more serious problem concerns overestimation when the matrix contains entries of different orders of magnitude. In such badly conditioned situations, it is better to work in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}^{n\times n})$ rather than $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D})$. Another more algorithmic technique for reducing the wrapping effect will be discussed in sections 6.4 and 7.5 below.

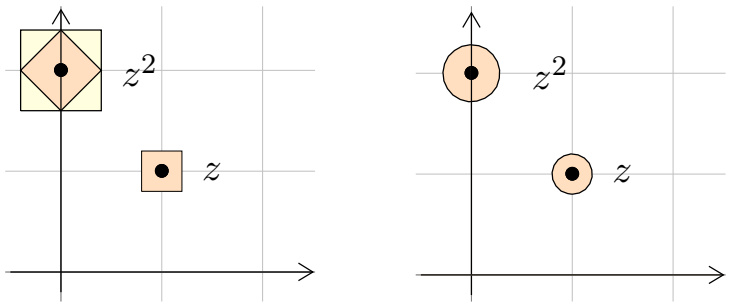


Fig. 1. Illustration of the computation of z^2 using interval and ball arithmetic, for $z = 1 + \mathrm{i}$.

4.4 Mathematical elegance

Even though personal taste in the choice between balls and intervals cannot be discussed, the elegance of the chosen approach for a particular application can partially be measured in terms of the human time which is needed to establish the necessary error bounds.

We have already seen that interval enclosures are particularly easy to obtain for monotonic real functions. Another typical algorithm where interval arithmetic is more convenient is the resolution of a system of equations using dichotomy. Indeed, it is easier to cut an n-dimensional block $[a_1, b_1] \times \cdots \times [a_n, b_n]$ into 2^n smaller blocks than to perform a similar operation on balls.

For most other applications, ball representations are more convenient. Indeed, error bounds are usually obtained by perturbation methods. For any mathematical proof where error bounds are explicitly computed in this way, it is generally easy to derive a certified algorithm based on ball arithmetic. We will see several illustrations of this principle in the sections below.

Implementations of interval arithmetic often rely on floating point arithmetic with correct rounding. One may question how good correct rounding actually is in order to achieve reliability. One major benefit is that it provides a simple and elegant way to specify what a mathematical function precisely does at limited precision. In particular, it allows numerical programs to execute exactly in the same way on many different hardware architectures.

On the other hand, correct rounding does have a certain cost. Although the cost is limited for field operations and elementary functions [Mul06], the cost increases for more complex special functions, especially if one seeks for numerical methods with a constant operation count. For arbitrary computable functions on $\mathbb{R}^{\mathrm{com}}$, correct rounding even becomes impossible. Another disadvantage is that correct rounding is lost as soon we perform more than one operation: in general, $g^{\uparrow} \circ f^{\uparrow}$ and $(g \circ f)^{\uparrow}$ do not coincide.

In the case of ball arithmetic, we only require an upper bound for the error, not necessarily the best possible representable one. In principle, this is just as reliable and usually more economic. Now in an ideal world, the development of numerical codes goes hand in hand with the systematic development of routines which compute the corresponding error bounds. In such a world, correct rounding becomes superfluous, since correctness is no longer ensured at the micro-level of hardware available functions, but rather at the top-level, *via* mathematical proof.

5 The numerical hierarchy

Computable analysis provides a good high-level framework for the automatic and certified resolution of analytic problems. The user states the problem in a formal language and specifies a required absolute or relative precision. The program should return a numerical result which is certified to meet the requirement on the precision. A simple example is to compute an ε-approximation for π, for a given $\varepsilon \in \mathbb{D}^{>}$.

The Mathemagix system [vdH+02b] aims the implementation of efficient algorithms for the certified resolution of numerical problems. Our ultimate goal is that these algorithms become as efficient as more classical numerical methods, which are usually non certified and only operate at limited precision. A naive approach is to systematically work with computable real numbers. Although this approach is convenient

for theoretical purposes in the area of computable analysis, the computation with functions instead of ordinary floating point numbers is highly inefficient.

In order to address this efficiency problem, the MATHEMAGIX libraries for basic arithmetic on analytic objects (real numbers, matrices, polynomials, etc.) are subdivided into four layers of the so called *numerical hierarchy* (see figure 2). We will illustrate this decomposition on the problem of multiplying two $n \times n$ computable real matrices. The numerical hierarchy turns out to be a convenient framework for more complex problems as well, such as the analytic continuation of the solution to a dynamical system. As a matter of fact, the framework incites the developer to restate the original problem at the different levels, which is generally a good starting point for designing an efficient solution.

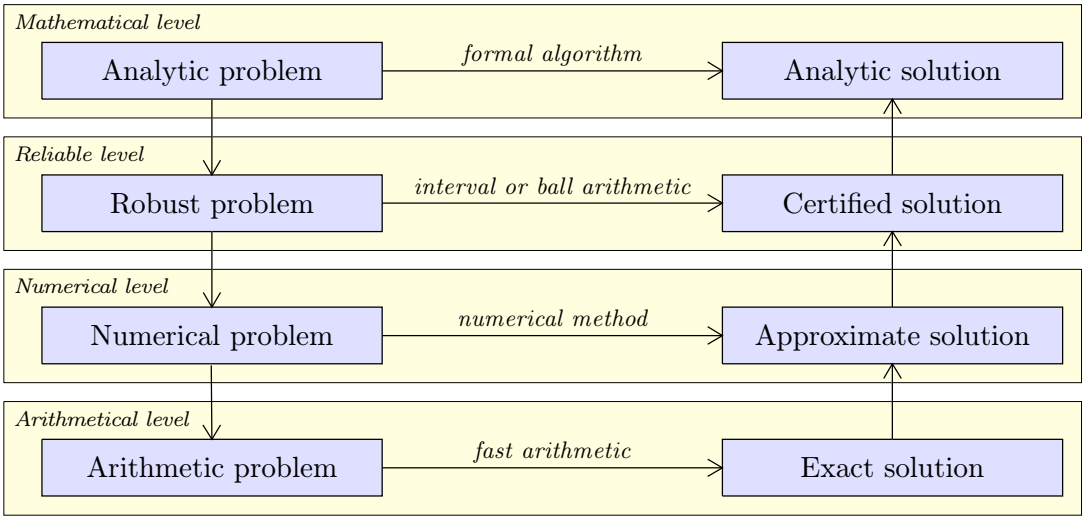


Fig. 2. The numerical hierarchy.

Mathematical level On the mathematical top level, we are given two computable real $n \times n$ matrices $A, B \in (\mathbb{R}^{\mathrm{com}})^{n \times n}$ and an absolute error $\varepsilon \in \mathbb{D}^{>}$. The aim is to compute ε-approximations for all entries of the product $C = AB$.

The simplest approach to this problem is to use a generic formal matrix multiplication algorithm, using the fact that $\mathbb{R}^{\mathrm{com}}$ is an effective ring. However, as stressed above, instances of $\mathbb{R}^{\mathrm{com}}$ are really functions, so that ring operations in $\mathbb{R}^{\mathrm{com}}$ are quite expensive. Instead, when working at precision p, we may first compute ball approximations for all the entries of A and B, after which we form two approximation matrices $\tilde{A}, \tilde{B} \in \mathbb{B}^{n \times n}$. The multiplication problem then reduces to the problem of multiplying two matrices in $\mathbb{B}^{n \times n}$. This approach has the advantage that $O(n^3)$ operations on "functions" in $\mathbb{R}^{\mathrm{com}}$ are replaced by a single multiplication in $\mathbb{B}^{n \times n}$.

Reliable level The aim of this layer is to implement efficient algorithms on balls. Whereas the actual numerical computation is delegated to the numerical level below, the reliable level should be able to perform the corresponding error analysis automatically.

When operating on non-scalar objects, such as matrices of balls, it is often efficient to rewrite the objects first. For instance, when working in fixed point arithmetic (i.e. all entries of A and B admit similar orders of magnitude), a matrix in $\mathbb{B}^{n \times n}$ may also be considered as a *matricial ball* in $\mathcal{B}(\mathbb{D}^{n \times n}, \mathbb{D})$ for the matrix norm $\| \cdot \|_\infty$. We multiply two such balls using the formula

$$\mathcal{B}(x, r) \cdot^{\circ} \mathcal{B}(y, s) = \mathcal{B}(xy, [n((\|x\|_\infty + r)s + r\|y\|_\infty)]^{\uparrow}),$$

which is a corrected version of (7), taking into account that the matrix norm $\| \cdot \|_\infty$ only satisfies $\|xy\|_\infty \leqslant n\|x\|_\infty\|y\|_\infty$. Whereas the naive multiplication in $\mathbb{B}^{n \times n}$ involves $4n^3$ multiplications in $\mathbb{D}$, the

new method reduces this number to $n^3 + 2n^2$: one "expensive" multiplication in $\mathbb{D}^{n\times n}$ and two scalar multiplications of matrices by numbers. This type of tricks will be discussed in more detail in section 6.4 below.

Essentially, the new method is based on the isomorphism

$$\mathcal{B}(\mathbb{D}, \mathbb{D})^{n\times n} \cong \mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}).$$

A similar isomorphism exists on the mathematical level:

$$(\mathbb{R}^{\mathrm{com}})^{n\times n} \cong (\mathbb{R}^{n\times n})^{\mathrm{com}}$$

As a variant, we directly may use the latter isomorphism at the top level, after which ball approximations of elements of $(\mathbb{R}^{n\times n})^{\mathrm{com}}$ are already in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D})$.

Numerical level Being able to perform an automatic error analysis, the actual numerical computations are done at the numerical level. In our example, we should implement an efficient algorithm to multiply two matrices in $\mathbb{D}^{n\times n}$. In single or double precision, we may usually rely on highly efficient numerical libraries (Blas, Lapack, etc.). In higher precisions, new implementations are often necessary: even though there are efficient libraries for multiple precision floating point numbers, these libraries usually give rise to a huge overhead. For instance, when using Mpfr [HLRZ00] at double precision, the overhead with respect to machine doubles is usually comprised between 10 and 100.

When working with matrices with multiple precision floating point entries, this overhead can be greatly reduced by putting the entries under a common exponent using the isomorphism

$$\mathbb{D}^{n\times n} = (\mathbb{Z}2^{\mathbb{Z}})^{n\times n} \cong \mathbb{Z}^{n\times n}2^{\mathbb{Z}}.$$

This reduces the matrix multiplication problem for floating point numbers to a purely arithmetic problem. Of course, this method becomes numerically unstable when the exponents differ wildly; in that case, row preconditioning of the first multiplicand and column preconditioning of the second multiplicand usually helps.

Arithmetic level After the above succession of reductions, we generally end up with an arithmetic problem such as the multiplication of two $n \times n$ matrices in $\mathbb{Z}^{n\times n}$. The efficient resolution of this problem for all possible n and integer bit lengths p is again non-trivial.

Indeed, libraries such as Gmp [Gra91] do implement Schönhage-Strassen's algorithm [SS71] algorithm for integer multiplication. However, the corresponding naive algorithm for the multiplication of matrices has a time complexity $O(I(p)n^3)$, which is far from optimal for large values of n.

Indeed, for large n, it is better to use multi-modular methods. For instance, choosing sufficiently many small primes $q_1, \ldots, q_k < 2^{32}$ (or 2^{64}) with $q_1 \cdots q_k > 2n4^p$, the multiplication of the two integer matrices can be reduced to k multiplications of matrices in $(\mathbb{Z}/q_i\mathbb{Z})^{n\times n}$. Recall that a $2p$-bit number can be reduced modulo all the q_i and reconstructed from these reductions in time $O(I(p)\log p)$. The improved matrix multiplication algorithm therefore admits a time complexity $O(pn^3 + n^2 I(p)\log p)$ and has been implemented in Linbox [DGG+02b,DGG+02a], Mathemagix and several other systems. FFT-based methods achieve similar practical time complexities $O(pn^3 + n^2 I(p))$ when n and p are of the same order of magnitude.

6 Reliable linear algebra

In this section, we will start the study of ball arithmetic for non numeric types, such as matrices. We will examine the complexity of common operations, such as matrix multiplication, linear system solving, and the computation of eigenvectors. Ideally, the certified variants of these operations are only slightly more expensive than the non certified versions. As we will see, this objective can sometimes be met indeed. In general however, there is a trade-off between the efficiency of the certification and its quality, i.e. the sharpness of the obtained bound. As we will see, the overhead of bound computations also tends to diminish for increasing bit precisions p.

6.1 Matrix multiplication

Let us first consider the multiplication of two $n \times n$ double precision matrices

$$M, N \in \mathbb{B}_{51,12}^{n \times n}.$$

Naive strategy The simplest multiplication strategy is to compute MN using the naive symbolic formula

$$(MN)_{i,k} = \sum_{j=1}^{n} M_{i,j} N_{j,k}. \tag{19}$$

Although this strategy is efficient for very small n, it has the disadvantage that we cannot profit from high performance BLAS libraries which might be available on the computer.

Revisited naive strategy Reconsidering M and N as balls with matricial radii

$$M, N \in \mathcal{B}(\mathbb{D}_{51,12}^{n \times n}, \mathbb{D}_{51,12}^{n \times n}),$$

we may compute MN using

$$MN = \mathcal{B}([M_c N_c]^{\updownarrow}, [|M_c| N_r + M_r (|N_c| + N_r) + n\epsilon |M_c||N_c|]^{\uparrow}), \tag{20}$$

where $|M|$ is given by $|M|_{i,j} = |M_{i,j}|$. A similar approach was first proposed in [Rum99a]. Notice that the additional term $n\epsilon |M_c| \cdot^{\uparrow} |N_c|$ replaces $\Delta(MN)$. This extra product is really required: the computation of $(M_c N_c)_{i,j}$ may involve cancellations, which prevent a bound for the rounding errors to be read off from the end-result. The formula (20) does assume that the underlying BLAS library computes $M_c N_c$ using the naive formula (19) and correct rounding, with the possibility to compute the sums in any suitable order. Less naive schemes, such as Strassen multiplication [Str69], may give rise to additional rounding errors.

Fast strategy The above naive strategies admit the disadvantage that they require four non certified $n \times n$ matrix products in $\mathbb{D}_{51,12}^{n \times n}$. If M and N are well-conditioned, then the following formula may be used instead:

$$MN = \mathcal{B}(M_c N_c, R) \tag{21}$$

$$R_{i,k} = [\|(M_{i,\cdot})_c\| \|(N_{\cdot,k})_r\| + \|(M_{i,\cdot})_r\| \|(N_{\cdot,k})_c\| + n\epsilon \|(M_{i,\cdot})_c\| \|(N_{i,\cdot})_c\|]^{\uparrow}, \tag{22}$$

where $M_{i,\cdot}$ and $N_{\cdot,k}$ stand for the i-th row of M and the k-th column of N. Since the $O(n)$ norms can be computed using only $O(n^2)$ operations, the cost of the bound computation is asymptotically negligible with respect to the $O(n^3)$ cost of the multiplication $M_c N_c$.

Hybrid strategy For large $n \times n$ matrices, chances increase that M or N gets badly conditioned, in which case the quality of the error bound (22) decreases. Nevertheless, we may use a compromise between the naive and the fast strategies: fix a not too small constant K, such as $K \approx 16$, and rewrite M and N as $\lceil \frac{n}{K} \rceil \times \lceil \frac{n}{K} \rceil$ matrices whose entries are $K \times K$ matrices. Now multiply M and N using the revisited naive strategy, but use the fast strategy on each of the $K \times K$ block coefficients. Being able to choose K, the user has an explicit control over the trade-off between the efficiency of matrix multiplication and the quality of the computed bounds.

As an additional, but important observation, we notice that the user often has the means to perform an "*a posteri* quality check". Starting with a fast but low quality bound computation, we may then check whether the computed bound is suitable. If not, then we recompute a better bound using a more expensive algorithm.

High precision multiplication Assume now that we are using multiple precision arithmetic, say $M, N \in \mathbb{B}_p^{n \times n}$. Computing MN using (20) requires one expensive multiplication in $\mathbb{D}_p^{n \times n}$ and three cheap multiplications in $\mathbb{D}_{64}^{n \times n}$. For large p, the bound computation therefore induces no noticeable overhead.

6.2 Matrix inversion

Assume that we want to invert an $n \times n$ ball matrix $M \in \mathbb{B}_p^{n\times n}$. This a typical situation where the naive application of a symbolic algorithm (such as gaussian elimination or LR-decomposition) may lead to overestimation. An efficient and high quality method for the inversion of M is called Hansen's method [HS67,Moo66]. The main idea is to first compute the inverse of M_c using a standard numerical algorithm. Only at the end, we estimate the error using a perturbative analysis. The same technique can be used for many other problems.

More precisely, we start by computing an approximation $N_c \in \mathbb{D}_p^{n\times n}$ of $(M_c)^{-1}$. Putting $N = \mathcal{B}(N_c, 0)$, we next compute the product MN using ball arithmetic. This should yield a matrix of the form $1 - E$, where E is small. If E is indeed small, say $\|E\| \leqslant 2^{-p/2}$, then

$$\|(1-E)^{-1} - 1 - E\| \leqslant \frac{\|E\|^2}{1-\|E\|}. \tag{23}$$

Denoting by Ω_n the $n \times n$ matrix whose entries are all $\mathcal{B}(0, 1)$, we may thus take

$$(1-E)^{-1} := 1 + E + \frac{\|E\|^2}{1-\|E\|}\Omega_n.$$

Having inverted $1 - E$, we may finally take $M^{-1} = N(1-E)^{-1}$. Notice that the computation of $\|E\|$ can be quite expensive. It is therefore recommended to replace the check $\|E\| \leqslant 2^{-p/2}$ by a cheaper check, such as $n\|E\|_\infty \leqslant 2^{-p/2}$.

Unfortunately, the matrix E is not always small, even if M is nicely invertible. For instance, starting with a matrix M of the form

$$M = J_{n,K} = \begin{pmatrix} 1 & K & & \\ & 1 & \ddots & \\ & & \ddots & K \\ & & & 1 \end{pmatrix},$$

with K large, we have

$$M^{-1} = \begin{pmatrix} 1 & -K & K^2 & \cdots & (-K)^{n-1} \\ & 1 & -K & \ddots & \vdots \\ & & 1 & \ddots & K^2 \\ & & & \ddots & -K \\ & & & & 1 \end{pmatrix}.$$

Computing $(M_c)^{-1}$ using bit precision p, this typically leads to

$$\|E\| \approx K^{n-1}2^{-p}.$$

In such cases, we rather reduce the problem of inverting $1 - E$ to the problem of inverting $1 - E^2$, using the formula

$$(1-E)^{-1} = (1+E)(1-E^2)^{-1}. \tag{24}$$

More precisely, applying this trick recursively, we compute $E^2, E^4, E^8, \ldots$ until $\|E^{2^k}\|$ becomes small (say $\|E^{2^k}\| \leqslant 2^{-p/2}$) and use the formula

$$\begin{aligned} (1-E)^{-1} &= (1+E)(1+E^2)\cdots(1+E^{2^{k-1}})(1-E^{2^k})^{-1} \\ (1-E^{2^k})^{-1} &:= 1 + E^{2^k} + \frac{\|E^{2^k}\|^2}{1-\|E^{2^k}\|}\Omega_n. \end{aligned} \tag{25}$$

We may always stop the algorithm for $2^k > n$, since $(J_{n,K} - 1)^n = 0$. We may also stop the algorithm whenever $\|E^{2^k}\| \geqslant 1$ and $\|E^{2^k}\| \geqslant \|E^{2^{k-1}}\|$, since M usually fails to be invertible in that case.

In general, the above algorithm requires $O(\log m)$ ball $n \times n$ matrix multiplications, where m is the size of the largest block of the kind $J_{K,m}$ in the Jordan decomposition of M. The improved quality therefore requires an additional $O(\log n)$ overhead in the worst case. Nevertheless, for a fixed matrix M and $p \to \infty$, the norm $\|E\|$ will eventually become sufficiently small (i.e. $2^{-p/2}$) for (23) to apply. Again, the complexity thus tends to improve for high precisions. An interesting question is whether we can avoid the ball matrix multiplication MN altogether, if p gets really large. Theoretically, this can be achieved by using a symbolic algorithm such as Gaussian elimination or LR-decomposition using ball arithmetic. Indeed, even though the overestimation is important, it does not depend on the precision p. Therefore, we have $(M^{-1})_r = O(2^{-p})$ and the cost of the bound computations becomes negligible for large p.

6.3 Eigenproblems

Let us now consider the problem of computing the eigenvectors of a ball matrix $M \in \mathbb{B}_p^{n\times n}$, assuming for simplicity that the corresponding eigenvalues are non zero and pairwise distinct. We adopt a similar strategy as in the case of matrix inversion. Using a standard numerical method, we first compute a diagonal matrix $\Lambda \in \mathbb{D}_p^{n\times n}$ and an invertible transformation matrix $T \in \mathbb{D}_p^{n\times n}$, such that

$$T^{-1} M_c T \approx \Lambda. \tag{26}$$

The main challenge is to find reliable error bounds for this computation. Again, we will use the technique of small perturbations. The equation (26) being a bit more subtle than $M_c N_c \approx 1$, this requires more work than in the case of matrix inversion. In fact, we start by giving a numerical method for the iterative improvement of an approximate solution. A variant of the same method will then provide the required bounds. Again, this idea can often be used for other problems. The results of this section are work in common with B. Mourrain and Ph. Trebuchet; in [vdHMT], an even more general method is given, which also deals with the case of multiple eigenvalues.

Given $\tilde{M}$ close to M_c, we have to find $\tilde{T}$ close to T and $\tilde{\Lambda}$ close to Λ, such that

$$\tilde{T}^{-1} \tilde{M} \tilde{T} = \tilde{\Lambda}. \tag{27}$$

Putting

$$\begin{aligned}
\tilde{T} &= T(1+E)\\
\tilde{\Lambda} &= \Lambda(1+\Delta)\\
N &= T^{-1}\tilde{M}T\\
H &= N - \Lambda,
\end{aligned}$$

this yields the equation

$$(1+E)^{-1} N (1+E) = \Lambda(1+\Delta).$$

Expansion with respect to E yields

$$\begin{aligned}
\frac{1}{1+E} N(1+E) &= N(1+E) - \frac{E}{1+E} N(1+E)\\
&= N + NE - EN + \frac{E^2}{1+E} N - \frac{E}{1+E} NE\\
&= N + [\Lambda, E] + [H, E] + \frac{E^2}{1+E} N - \frac{E}{1+E} NE\\
&= N + [\Lambda, E] + O([H,E]) + O(E^2).
\end{aligned} \tag{28}$$

Forgetting about the non-linear terms, the equation

$$[E, \Lambda] + \Lambda\Delta = H$$

admits a unique solution

$$E_{i,j} = \begin{cases} \frac{H_{i,j}}{\Lambda_{j,j} - \Lambda_{i,i}} \text{ if } j \neq i \\ 0 \quad \text{otherwise} \end{cases}$$
$$\Delta_{i,j} = \begin{cases} \frac{H_{i,i}}{\Lambda_{i,i}} \text{ if } i = j \\ 0 \quad \text{otherwise} \end{cases}$$

Setting

$$\kappa = \kappa(\Lambda) = \max \left\{ \max \left\{ \frac{1}{|\Lambda_{j,j} - \Lambda_{i,i}|} : 1 \leqslant i < j \leqslant n \right\}, \max \left\{ \frac{1}{\Lambda_{i,i}} : 1 \leqslant i \leqslant n \right\} \right\},$$

it follows that

$$\max\{\|E\|, \|\Delta\|\} \leqslant \kappa \sqrt{n} \|H\|. \tag{29}$$

Setting $T' = T(1+E)$, $\Lambda' = \Lambda(1+\Delta)$ and $H' = (T')^{-1} \tilde{M} T' - \Lambda'$, the relation (28) also implies

$$H' = [H, E] + \frac{E^2}{1+E}(\Lambda + H) - \frac{E}{1+E}(\Lambda + H)E.$$

Under the additional condition $\|E\| \leqslant \frac{1}{2}$, it follows that

$$\|H'\| \leqslant 3\|H\|\|E\| + 4\|E\|^2\|\Lambda\|. \tag{30}$$

For sufficiently small H, we claim that iteration of the mapping $\Phi : (T, \Lambda) \mapsto (T', \Lambda')$ converges to a solution of (27).

Let us denote $(T^{(k)}, \Lambda^{(k)}) = \Phi^k(T, \Lambda)$, $H^{(k)} = (T^{(k)})^{-1} \tilde{M} T^{(k)} - \Lambda^{(k)}$ and let $(E^{(k)}, \Delta^{(k)})$ be such that $T^{(k+1)} = T^{(k)}(1 + E^{(k)})$ and $\Lambda^{(k+1)} = \Lambda^{(k)}(1 + \Delta^{(k)})$. Assume that

$$\|H\| \leqslant \frac{1}{80 n \kappa^2 \|\Lambda\|} \tag{31}$$

and let us prove by induction over k that

$$\|H^{(k)}\| \leqslant 2^{-k} \|H\| \tag{32}$$
$$\|\Lambda^{(k)}\| \leqslant 2\|\Lambda\| \tag{33}$$
$$\begin{aligned} \max\{\|E^{(k)}\|, \|\Delta^{(k)}\|\} &\leqslant 2\sqrt{n}\kappa \|H^{(k)}\| \\ &\leqslant \frac{1}{40\sqrt{n}\kappa\|\Lambda\| 2^k} \end{aligned} \tag{34}$$

This is clear for $k = 0$, so assume that $k > 0$. In a similar way as (30), we have

$$\|H^{(k)}\| \leqslant 3\|H^{(k-1)}\|\|E^{(k-1)}\| + 4\|E^{(k-1)}\|^2\|\Lambda^{(k-1)}\|. \tag{35}$$

Using the induction hypotheses and $\kappa\|\Lambda\| \geqslant 1$, it follows that

$$\begin{aligned} \|H^{(k)}\| &\leqslant (3 + 16\sqrt{n}\kappa\|\Lambda\|)\|E^{(k-1)}\|\|H^{(k-1)}\| \\ &\leqslant \frac{1}{2}\|H^{(k-1)}\|, \end{aligned}$$

which proves (32). Now let $\Sigma^{(k)}$ be such that

$$\begin{aligned} \Lambda^{(k)} &= \Lambda(1 + \Delta^{(0)}) \cdots (1 + \Delta^{(k-1)}) \\ &= \Lambda(1 + \Sigma^{(k)}). \end{aligned}$$

From (34), it follows that

$$\|\Sigma^{(k)}\| \leqslant \frac{1}{4\kappa\|\Lambda\|}.$$

On the one hand, this implies (33). On the other hand, it follows that

$$\kappa(\Lambda^{(k)}) \leqslant 2\kappa,$$

whence (29) generalizes to (34). This completes the induction and the linear convergence of $\|H^{(k)}\|$ to zero. In fact, the combination of (34) and (35) show that we even have quadratic convergence.

Let us now return to the original bound computation problem. We start with the computation of $H = T^{-1}MT - \Lambda$ using ball arithmetic. If the condition (31) is met (using the most pessimistic rounding), the preceding discussion shows that for every $\tilde{M} \in M$ (in the sense that $\tilde{M}_{i,j} \in M_{i,j}$ for all i, j), the equation (27) admits a solution of the form

$$\begin{aligned}\tilde{T} &= T(1 + \tilde{\Upsilon}) = T(1 + E^{(0)})(1 + E^{(1)}) \cdots \\ \tilde{\Lambda} &= \Lambda(1 + \tilde{\Sigma}) = \Lambda(1 + \Delta^{(0)})(1 + \Delta^{(1)}) \cdots ,\end{aligned}$$

with

$$\max\{\|E^{(k)}\|, \|\Delta^{(k)}\|\} \leqslant 2^{1-k}\sqrt{n}\kappa\|H\|,$$

for all k. It follows that

$$\max\{\|\tilde{\Upsilon}\|, \|\tilde{\Sigma}\|\} \leqslant \eta := 6\sqrt{n}\kappa\|H\|.$$

We conclude that

$$\begin{aligned}\tilde{T} &\in T(1 + \eta\Omega_n) \\ \tilde{\Lambda} &\in \mathcal{B}(1, \eta)\Lambda.\end{aligned}$$

We may thus return $(T(1 + \eta\Omega_n), \mathcal{B}(1, \eta)\Lambda)$ as the solution to the original eigenproblem associated to the ball matrix M.

The reliable bound computation essentially reduces to the computation of three matrix products and one matrix inversion. At low precisions, the numerical computation of the eigenvectors is far more expensive in practice, so the overhead of the bound computation is essentially negligible. At higher precisions p, the iteration $(T, \Lambda) \mapsto \Phi(T, \Lambda)$ actually provides an efficient way to double the precision of a numerical solution to the eigenproblem at precision $p/2$. In particular, even if the condition (31) is not met initially, then it usually can be enforced after a few iterations and modulo a slight increase of the precision. For fixed M and $p \to \infty$, it also follows that the numerical eigenproblem essentially reduces to a few matrix products. The certification of the end-result requires a few more products, which induces a constant overhead. By performing a more refined error analysis, it is probably possible to make the cost certification negligible, although we did not investigate this issue in detail.

6.4 Matricial balls versus ball matrices

In section 4.3, we have already seen that matricial balls in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D})$ often provide higher quality error bounds than ball matrices in $\mathcal{B}(\mathbb{D}, \mathbb{D})^{n\times n}$ or essentially equivalent variants in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}^{n\times n})$. However, ball arithmetic in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D})$ relies on the possibility to quickly compute a sharp upper bound for the operator norm $\|M\|$ of a matrix $M \in \mathbb{D}^{n\times n}$. Unfortunately, we do not know of any really efficient algorithm for doing this.

One expensive approach is to compute a reliable singular value decomposition of M, since $\|M\|$ coincides with the largest singular value. Unfortunately, this usually boils down to the resolution of the eigenproblem associated to M^*M, with a few possible improvements (for instance, the dependency of the singular values on the coefficients of M is less violent than in the case of a general eigenproblem).

Since we only need the largest singular value, a faster approach is to reduce the computation of $\|M\|$ to the computation of $\|M^*M\|$, using the formula

$$\|M\| = \sqrt{\|M^*M\|}.$$

Applying this formula k times and using a naive bound at the end, we obtain

$$\|M\| \leqslant \sqrt[2^k]{n\|(M^*M)^{2^{k-1}}\|_\infty}.$$

This bound has an accuracy of $\approx k - O(\log_2 n)$ bits. Since M^*M is symmetric, the $k-1$ repeated squarings of M^*M only correspond to about $\frac{k}{2}$ matrix multiplications. Notice also that it is wise to renormalize matrices before squaring them, so as to avoid overflows and underflows.

The approach can be speeded up further by alternating steps of tridiagonalization and squaring. Indeed, for a symmetric tridiagonal matrix D, the computation of D^2 and its tridiagonalization only take $O(n)$ steps instead of $O(n^3)$ for a full matrix product. After a few $k = O(n^2)$ steps of this kind, one obtains a good approximation μ of $\|D\|$. One may complete the algorithm by applying an algorithm with quadratic convergence for finding the smallest eigenvalues of $D - \mu$. In the lucky case when D has an isolated maximal eigenvalue, a certification of this method will provide sharp upper bounds for $\|D\|$ in reasonable time.

Even after the above improvements, the computation of sharp upper bounds for $\|M\|$ remains quite more expensive than ordinary matrix multiplication. For this reason, it is probably wise to avoid ball arithmetic in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D})$ except if there are good reasons to expect that the improved quality is really useful for the application in mind.

Moreover, when using ball arithmetic in $\mathcal{B}(\mathbb{D}^{n\times n}, \mathbb{D}^{n\times n})$, it is often possible to improve algorithms in ways to reduce overestimation. When interpreting a complete computation as a dag, this can be achieved by minimizing the depth of the dag, i.e. by using an algorithm which is better suited for parallelization. Let us illustrate this idea for the computation of the k-th power M^k of a matrix M. When using Horner's method (multiply the identity matrix k times by M), we typically observe an overestimation of $O(k)$ bits (as for the example (18)). If we use binary powering, based on the rule

$$M^k = M^{\lfloor k/2 \rfloor} M^{\lceil k/2 \rceil}, \tag{36}$$

then the precision loss drops down to $O(\log k)$ bits. We will encounter a less obvious application of the same idea in section 7.5 below.

7 Reliable power series arithmetic

7.1 Ball series versus serial balls

There are two typical applications of power series $f \in \mathbb{R}[[z]]$ or $f \in \mathbb{C}[[z]]$ with certified error bounds. When f occurs as a generating function in a counting problem or random object generator, then we are interested in the computation of the coefficients f_n for large n, together with reliable error bounds. A natural solution is to systematically work with computable power series with ball coefficients in $\mathbb{B}_p[[z]]^{\mathrm{com}}$. For many applications, we notice that p is fixed, whereas $n \gg p$ may become very large.

The second typical application is when $f \in \mathbb{C}[[z]]$ is the local expansion of an analytic function on a disk $\mathcal{B}(0, \rho)$ and we wish to evaluate f at a point z with $|z| < \rho$. The geometric decrease of $|f_n z^n|$ implies that we will need only $n = O(p)$ coefficients of the series. In order to bound the remaining error using Cauchy's formula, we do not only need bounds for the individual coefficients f_n, but also for the norm $\|f\|_\rho$ defined in (4). Hence, it is more natural to work with serial balls in $\mathcal{B}(\mathbb{D}_p[\mathrm{i}][[z]]^{\mathrm{com}}, \mathbb{D}_{64})$, while using the $\|\cdot\|_\rho$ norm. Modulo a rescaling $f(z) \mapsto f(\rho z)$, it will be convenient to enforce $\rho = 1$. In order to compute sharp upper bounds $\llbracket f \rrbracket$ for $\|f\| = \|f\|_1$, it will also be convenient to have an algorithm which computes bounds $\llbracket f_{n;} \rrbracket$ for the tails

$$f_{n;} = f_n z^n + f_{n+1} z^{n+1} + \cdots .$$

Compared to the computation of the corresponding head

$$f_{;n} = f_0 + \cdots + f_{n-1} z^{n-1},$$

we will show in section 7.4 that the computation of such a tail bound is quite cheap.

Again the question arises how to represent $f_{;n}$ in a reliable way. We may either store a global upper bound for the error, so that $f_{;n} \in \mathcal{B}(\mathbb{D}_p[\mathrm{i}][z], \mathbb{D}_{64})$, or compute individual bounds for the errors, so that $f_{;n} \in \mathbb{B}[\mathrm{i}]_p[z]$. If our aim is to evaluate f at a point z with $|z| \approx 1$, then both representations $P \in \mathbb{B}[\mathrm{i}]_p[z]$ and $\hat{P} = \mathcal{B}(P_c, \|P_r\|_\infty) \in \mathcal{B}(\mathbb{D}_p[\mathrm{i}][z], \mathbb{D}_{64})$ give rise to evaluations $P(z), \hat{P}(z) \in \mathbb{B}[\mathrm{i}]_p$ with equally precise error bounds. Since the manipulation of global error bounds is more efficient, the corresponding representation should therefore be preferred in this case. In the multivariate case, one has the additional

benefit that "small" coefficients $f_i z^i$ (e.g. $|f_i| \leqslant \epsilon_p \|f\|$) can simply be replaced by a global error $\mathcal{B}(0, |f_i|)$, thereby increasing the sparsity of f. On the other hand, individual error bounds admit the advantage that rescaling $f(z) \mapsto f(\lambda z)$ is cheap. If we suddenly find out that f is actually convergent on a larger disk and want to evaluate f at a point z with $|z| > 1$, then we will not have to recompute the necessary error bounds for $f_{;n}$ from scratch.

Serial ball representations similar to what has been described above are frequently used in the area of Taylor models [MB96,MB04] for the validated long term integration of dynamical systems. In the case of Taylor models, there is an additional twist: given a dynamical system of dimension d, we not only compute a series expansion with respect to the time t, but also with respect to small perturbations $\varepsilon_1, \ldots, \varepsilon_d$ of the initial conditions. In particular, we systematically work with power series in several variables. Although such computations are more expensive, the extra information may be used in order to increase the sharpness of the computed bounds. A possible alternative is to compute the expansions in $\varepsilon_1, \ldots, \varepsilon_d$ only up to the first order and to use binary splitting for the multiplication of the resulting Jacobian matrices on the integration path. This approach will be detailed in section 7.5.

7.2 Reliable multiplication of series and polynomials

In order to study the reliable multiplication of series f and g, let us start with the case when $f, g \in \mathcal{B}(\mathbb{D}_p[[z]]^{\text{com}}, \mathbb{D}_{64})$, using the sup-norm on the unit disk. We may take

$$h \;=\; fg = \mathcal{B}(\widetilde{f_c g_c}, [\|f_c\|\|g_r\| + \|f_r\|(\|g_c\| + \|g_r\|) + \delta]^{\uparrow}),$$

where $h_c = \widetilde{f_c g_c}$ stands for a δ-approximation of $f_c g_c$. Since h_c is really a numeric algorithm for the computation of its coefficients, the difficulty resides in the fact that δ has to be chosen once and for all, in such a way that the bound $\|h_c - f_c g_c\|$ will be respected at the limit. A reasonable choice is $\delta = \epsilon_p \|f_c\|\|g_c\|$. We next distribute this error over the infinity of coefficients: picking some $\alpha < 1$, each coefficient $(h_c)_n$ is taken to be an $[(1-\alpha)\alpha^n \delta]$-approximation of $f_c g_c$. Of course, these computation may require a larger working precision than p. Nevertheless, f and g are actually convergent on a slightly larger disk $\mathcal{B}(0, \rho)$. Picking $\alpha = 1/\rho$, the required increase of the working precision remains modest.

Let us now turn our attention to the multiplication of two computable series $f, g \in \mathbb{B}_p[[z]]^{\text{com}}$ with ball coefficients. Except for naive power series multiplication, based on the formula $(fg)_n = \sum_{i+j=n} f_i g_j$, most other multiplication algorithms (whether relaxed or not) use polynomial multiplication as a subalgorithm. We are thus left with the problem of finding an efficient and high quality algorithm for multiplying two polynomials $P, Q \in \mathbb{B}_p[z]$ of degrees $< n$. In order to simplify the reading, we will assume that $P_0, P_{n-1}, Q_0, Q_{n-1}$ are all non-zero.

As in the case of matrix multiplication, there are various approaches with different qualities, efficiencies and aptitudes to profit from already available fast polynomial arithmetic in $\mathbb{D}_p[z]$. Again, the naive $O(n^2)$ approach provides almost optimal numerical stability and qualities for the error bounds. However, this approach is both slow from an asymptotic point of view and unable to rely on existant multiplication algorithms in $\mathbb{D}_p[z]$.

If the coefficients of P and Q are all of the same order of magnitude, then we may simply convert P and Q into polynomial balls in $\mathcal{B}(\mathbb{D}_p[z], \mathbb{D}_{64})$ for the norm $\|\cdot\|_\infty$ and use the following crude formula for their multiplication:

$$PQ = \mathcal{B}(\widetilde{P_c Q_c}, [n\|P_c\|_\infty \|Q_r\|_\infty + n\|P_r\|_\infty(\|Q_c\|_\infty + \|Q_r\|_\infty) + \delta]^{\uparrow}), \tag{37}$$

where $\widetilde{P_c Q_c}$ stands for a δ-approximation of $P_c Q_c$. In other words, we may use any efficient multiplication algorithm in $\mathbb{D}_p[z]$ for the approximation of $P_c Q_c$, provided that we have a means to compute a certified bound δ for the error.

In our application where P and Q correspond to ranges of coefficients in the series f and g, we usually have $P_i \approx P_0 \rho_f^{-i}$ and $Q_i \approx Q_0 \rho_g^{-i}$ for the convergence radii ρ_f and ρ_g of P and Q. In order to use (37), it is therefore important to scale $P(z) \mapsto P(z/\rho)$ and $Q(z) \mapsto Q(z/\rho)$ for a suitable ρ. If we are really interested in the evaluation of fg at points z in a disk $\mathcal{B}(0, r)$, then we may directly take $\rho = r$. In we are rather interested in the coefficients of fg, then $\rho = \min(\rho_f, \rho_g)$ is the natural choice. However, since ρ_f and ρ_g are usually unknown, we first have to compute suitable approximations for them, based on

the available coefficients of P and Q. A good heuristic approach is to determine indices $i < n/2 \leqslant j$ such that

$$\left|\frac{P_k}{P_i}\right|^{j-i} \leqslant \left|\frac{P_j}{P_i}\right|^{k-i},$$

for all k, and to take

$$\rho_f \approx \left|\frac{P_i}{P_i}\right|^{\frac{1}{j-i}}$$

as the approximation for ρ_f. Recall that the numerical Newton polygon of N_P is the convex hull of all points $(i, \log|P_i| - \lambda) \in \mathbb{R}^2$ with $\lambda \geqslant 0$. Consider the edge of N_P whose projection on the x-axis contains $n/2$. Then i and j are precisely the extremities of this projection, so they can be computed in linear time.

For large precisions $n = O(p)$, the scaling algorithm is both very efficient and almost of an optimal quality. For small p and large n, there may be some precision loss which depends on the nature of the smallest singularities of f and g. Nevertheless, for many singularities, such as algebraic singularities, the precision loss is limited to $O(\log n)$ bits. For a more detailed discussion, we refer to [vdH02a, Section 6.2].

In other applications, where P and Q are not obtained from formal power series, it is usually insufficient to scale using a single factor ρ. This is already the case when multiplying $P = 1 + z$ and $Q = 1 + \delta z$ for small $\delta \ll \epsilon_p$, since the error bound for $(PQ)_2 = \delta$ exceeds ϵ_p. One possible remedy is to "precondition" P and Q according to their numerical Newton polygons and use the fact that N_{PQ} is close to the Minkowski sum $N_P + N_Q$.

More precisely, for each i, let $(N_P)_i > 0$ denote the number such that $(i, \log(N_P)_i)$ lies on one of the edges of N_P. Then $(N_P)_i$ is of the same order of magnitude as P_i, except for indices for which P_i is small "by accident". Now consider the preconditioned relative error

$$\nu_P = \max_i \frac{(P_i)_r}{(N_P)_i}$$

and similarly for Q. Then

$$[(PQ)_i]_r \leqslant n(\nu_P + \nu_Q + \nu_P\nu_Q)(N_P + N_Q)_i, \tag{38}$$

if PQ is computed using infinite precision ball arithmetic. Assuming a numerically stable multiplication algorithm for the centers, as proposed in [vdH08a], and incorporating the corresponding bound for the additional errors into the right hand side of (38), we thus obtain an efficient way to compute an upper bound for $(PQ)_r$.

Notice that the numerical Newton polygon N_P has a close relation to the orders of magnitudes of the roots of P. Even though the error bounds for some "accidentally small" coefficients $(PQ)_i$ may be bad for the above method, the error bounds have a good quality if we require them to remain valid for small perturbations of the roots of PQ.

7.3 Reliable series division and exponentiation

The algorithms from the previous section can in particular be used for the relaxed multiplication of two ball power series $f, g \in \mathbb{B}_p[[z]]^{\mathrm{com}}$. In particular, using the implicit equation

$$g = 1 + zfg, \tag{39}$$

this yields a way to compute $g = (1 - zf)^{-1}$. Unfortunately, the direct application of this method leads to massive overestimation. For instance, the computed error bounds for the inverses of

$$g = \frac{1}{1 - 3z - 2z^2}$$
$$h = \frac{1}{1 - 3z + 2z^2}$$

coincide. Indeed, even when using a naive relaxed multiplication, the coefficients of g and h are computed using the recurrences

$$g_n = 3g_{n-1} + 2g_{n-2}$$
$$h_n = 3h_{n-1} - 2h_{n-2},$$

but the error bound ε_n for g_n and h_n is computed using the same recurrence

$$\varepsilon_n = 3\varepsilon_{n-1} + 2\varepsilon_{n-2},$$

starting with $\varepsilon_0 \approx \epsilon_p$. For $n \to \infty$, it follows that $g_n \sim \lambda\alpha^{-n}$ and $\varepsilon_n \sim \lambda\epsilon_p\alpha^{-n}$ for some λ, where $\alpha \approx 0.281$ is the smallest root of $1 - 3\alpha - 2\alpha^2$. Hence the error bound for g_n is sharp. On the other hand, $h_n \sim \mu\beta^n$ for some μ, where $\beta = \frac{1}{2}$ is the smallest root of $1 - 3\beta + 2\beta^2$. Hence, the error bound ε_n is $n \log_2(\beta/\alpha)$ bits too pessimistic in the case of h. The remedy is similar to what we did in the case of matrix inversion. We first introduce the series

$$\varphi = \frac{1}{1 - zf_c} \in \mathbb{D}_p[[z]]^{\text{com}}$$
$$\psi = \frac{[\varphi(zf - 1)]_{1;}}{z} \in \mathbb{B}_p[[z]]^{\text{com}}$$

and next compute g using

$$g = \frac{\varphi}{1 - z\psi},$$

where $1 - z\psi$ is inverted using the formula (39). This approach has the advantage of being compatible with relaxed power series expansion and it yields high quality error bounds.

Another typical operation on power series is exponentiation. Using relaxed multiplication, we may compute the exponential g of an infinitesimal power series $f \in z\mathbb{B}_p[[z]]^{\text{com}}$ using the implicit equation

$$g = 1 + \int f'g, \tag{40}$$

where $\int$stands for "distinguished integration" in the sense that $(\int h)_0 = 0$ for all h. Again, one might fear that this method leads to massive overestimation. As a matter of fact, it usually does not. Indeed, assume for simplicity that $f_r = 0$, so that $f \in \mathbb{D}_p[[z]]$. Recall that $|f|$ denotes the power series with $|f|_n = |f_n|$. Roughly speaking, the error bound for g_n, when computed using the formula (40) will be the coefficient ε_n of the power series $\varepsilon = \exp(\epsilon_p|f|)$. Since $|f|$ has the same radius of convergence as f, it directly follows that the bit precision loss is sublinear $o(n)$. Actually, the dominant singularity of $|f|$ often has the same nature as the dominant singularity of f. In that case, the computed error bounds usually become very sharp. The observation generalizes to the resolution of linear differential equations, by taking a square matrix of power series for f.

7.4 Automatic tail bounds

In the previous sections, we have seen various reliable algorithms for the computation of the expansion $f_{;n}$ of a power series $f \in \mathbb{C}[[z]]$ up till a given order n. Such expansions are either regarded as ball polynomials in $\mathbb{B}[\mathrm{i}]_p[z]$ or as polynomial balls in $\mathcal{B}(\mathbb{D}_p[\mathrm{i}][z], \mathbb{D}_{64})$. Assuming that f is convergent on the closed unit disk $\mathcal{B}(0, 1)$, it remains to be shown how to compute tail bounds $\llbracket f_{n;} \rrbracket$. We will follow [vdH07b]. Given a ball polynomial P, then we notice that reasonably sharp upper and lower bounds $\llbracket P \rrbracket$ and $\lfloor\!\lfloor P \rfloor\!\rfloor$ for P can be obtained efficiently by evaluating P at $O(\deg P)$ primitive roots of unity using fast Fourier transforms [vdH07b, Section 6.2].

We will assume that the series f is either an explicit series, the result of an operation on other power series or the solution of an implicit equation. Polynomials are the most important examples of explicit series. Assuming that f is a polynomial of degree $< d$, we may simply take

$$\llbracket f_{n;} \rrbracket = \begin{cases} \llbracket z^{-n} f_{n;d} \rrbracket & \text{if } n < d \\ 0 & \text{otherwise} \end{cases}, \tag{41}$$

where

$$f_{n;d} = f_n z^n + \cdots + f_{d-1} z^{d-1}.$$

For simple operations on power series, we may use the following bounds:

$$\llbracket (f+g)_{;n} \rrbracket = \llbracket f_{;n} \rrbracket + \llbracket g_{;n} \rrbracket \tag{42}$$

$$\llbracket (f-g)_{;n} \rrbracket = \llbracket f_{;n} \rrbracket + \llbracket g_{;n} \rrbracket \tag{43}$$

$$\llbracket (fg)_{;n} \rrbracket = \llbracket f_{n;} \rrbracket (\llbracket g_{;n} \rrbracket + \llbracket g_{n;} \rrbracket) + \llbracket f_{;n} \rrbracket \llbracket g_{n;} \rrbracket + \llbracket (f_{;n} g_{;n})_{n;} \rrbracket \tag{44}$$

$$\llbracket (\int f)_{n;} \rrbracket = \frac{1}{n+1} \llbracket f_{n;} \rrbracket, \tag{45}$$

where

$$\llbracket (f_{;n} g_{;n})_{n;} \rrbracket \leqslant \sum_{k=0}^{n-1} |f_k| \left(\sum_{l=n-k}^{n-1} |g_l| \right)$$

can be computed in time $O(n)$. Due to possible overestimation, division has to be treated with care. Given an infinitesimal power series ε and

$$f = \frac{1}{1-\varepsilon},$$

so that

$$f_{n;} = \frac{1 + \varepsilon f_{;n} - f_{;n}}{1-\varepsilon},$$

we take

$$\llbracket f_{n;} \rrbracket = \frac{\llbracket (\varepsilon f_{;n})_{n;} \rrbracket}{\lfloor\!\lfloor 1 - \varepsilon_{;n} + \mathcal{B}(0, \llbracket \varepsilon_{n;} \rrbracket) \rfloor\!\rfloor}, \tag{46}$$

where $\llbracket (\varepsilon f_{;n})_{n;} \rrbracket$ is computed using (44).

Let $\Phi(f)$ be an expression which is constructed from f and polynomials, using the ring operations and distinguished integration (we exclude division in order to keep the discussion simple). Assume that each coefficient $\Phi(f)_n$ only depends on the previous coefficients $f_0, \ldots, f_{n-1}$ of f and not on $f_n, f_{n-1}, \ldots$. Then the sequence $0, \Phi(0), \Phi^2(0), \ldots$ tends to the unique solution f of the implicit equation

$$f = \Phi(f), \tag{47}$$

Moreover, $\Phi^k(0)_n = f_n$ for any $k > n$. In (39) and (40), we have already seen two examples of implicit equations of this kind.

The following technique may be used for the computation of tail bounds $\llbracket f_{n;} \rrbracket$. Given $c \in \mathbb{D}^{\geqslant}$ and assuming that $\llbracket f_{n;} \rrbracket \leqslant c$, we may use the rules (41–46) in order to compute a "conditional tail bound" $\llbracket \Phi(f)_{n;} | c \rrbracket$ for $\| \Phi(f)_{n;} \|$. Mathematically speaking, this bound has the property that for any power series g with $g_{;n} = f_{;n}$ and $\| g_{n;} \| \leqslant c$, we have $\| \Phi(g)_{n;} \| \leqslant \llbracket \Phi(g)_{n;} | c \rrbracket$. If $\llbracket \Phi(g)_{n;} | c \rrbracket \leqslant c$, then it follows that $\| [\Phi^{(k)}(f_{;n})]_{;n} \| \leqslant c$ for all k. Given any disk $\mathcal{B}(0, \rho)$ with $\rho < 1$, it follows that $\| \Phi^{(k)}(f_{;n}) - f \|_\rho \leqslant 2c\rho^k / (1-\rho)$ for any $k \geqslant n$, since $\Phi^{(k)}(f_{;n}) - f = O(z^k)$. In other words, $\Phi^{(k)}(f_{;n})$ uniformly converges to f on $\mathcal{B}(0, \rho)$. Therefore, $\| f_{n;} \|_\rho \leqslant c$ and $\| f_{n;} \| \leqslant c$, by letting $\rho \to 1$.

Let us now consider the mapping $\varphi : c \mapsto \llbracket \Phi(f)_{n;} | c \rrbracket$ and assume that Φ involves no divisions. When computing $\llbracket \Phi(f)_{n;} | c \rrbracket$ using infinite precision and the rules (41–44), we notice that φ is a real analytic function, whose power series expansion contains only positive coefficients. Finding the smallest c with $\varphi(c) \leqslant c$ thus reduces to finding the smallest fixed point c_{fix} of φ, if such a fixed point exists. We may use the secant method:

$$\begin{aligned} c_0 &:= 0 \\ c_1 &:= \varphi(c_0) \\ c_{k+2} &:= c_k + \frac{\varphi(c_k) - c_k}{c_{k+1} - \varphi(c_{k+1}) + \varphi(c_k) - c_k} (c_{k+1} - c_k) \end{aligned}$$

If $c_{k+1} < c_k$ for some k or if k exceeds a given threshold K, then the method fails and we set $[\![f_{n;}]\!] = +\infty$. Otherwise, c_k converges quadratically to c_{fix}. As soon as $|c_{k+1}/c_k - 1| < \varepsilon$, for some given $\varepsilon > 0$, we check whether $\varphi(\tilde{c}_{\text{fix}}) \leqslant \tilde{c}_{\text{fix}}$ for $\tilde{c}_{\text{fix}} = 2c_{k+1} - c_k$, in which case we stop. The resulting $\tilde{c}_{\text{fix}}$ is an approximation of c_{fix} with relative accuracy $\varepsilon > 0$.

Assuming that $\Psi(f)_{;n}$ has been computed for every subexpression of Φ, we notice that the computation of $[\![\Phi(f)_{n;} | c]\!]$ only requires $O(ns)$ floating point operations, where s is the size of Φ as an expression. More generally, the evaluation of $[\![\Phi(f)_{n;} | c_i]\!]$ for k different hypotheses $c_1, \ldots, c_k$ only requires $O(nsk)$ operations, since the heads $\Psi(f)_{;n}$ do not need to be recomputed. Our algorithm for the computation of $\tilde{c}_{\text{fix}}$ therefore requires at most $O(nsK)$ floating point operations. Taking $K = o(R(n)/n)$, it follows that the cost of the tail bound computation remains negligible with respect to the series expansion itself.

The approach generalizes to the case when f is a vector or matrix of power series, modulo a more involved method for the fixed point computation [vdH07b, Section 6.3]. If f is indeed convergent on $\mathcal{B}(0,1)$, then it can also be shown that φ indeed admits a fixed point if n is sufficiently large.

7.5 Long term integration of dynamical systems

Let us now consider a dynamical system

$$f = f(0) + \int \Phi(f), \tag{48}$$

where f is a vector of d unknown complex power series, $f(0) \in \mathbb{B}[\mathrm{i}]_p^d$, and Φ an expression built up from the entries of f and polynomials in $\mathbb{D}_p[\mathrm{i}][z]$ using the ring operations. Given $t \in \mathbb{D}[\mathrm{i}] \setminus \{0\}$, denote by $f_{z \times t}$ the scaled power series $f(tz)$ and by f_{z+t} the shifted power series $f(t+z)$, assuming that f converges at t. Also denote by $\Phi_{z \times t}$ and Φ_{z+t} the expressions which are obtained when replacing each polynomial P in Φ by $P_{z \times t}$ resp. P_{z+t}. Then $f_{z \times t}$ and f_{z+t} satisfy

$$f_{z \times t} = f(0) + t \int \Phi_{z \times t}(f_{z \times t}) \tag{49}$$

$$f_{z+t} = f(t) + \int \Phi_{z+t}(f_{z+t}) \tag{50}$$

Since (48) is a particular case of an implicit equation of the form (47), we have an algorithm for the computation of tail bounds $[\![f_{n;}]\!] \in (\mathbb{D}_{64}^{\geqslant} \cup \{\infty\})^d$ for f on $\mathcal{B}(0,1)$. Modulo rescaling (49), we may also compute tail bounds $[\![f_{n;}]\!]_\rho$ on more general disks $\mathcal{B}(0,\rho)$.

Assume that we want to integrate (48) along the real axis, as far as possible, and performing all computations with an approximate precision of p bits. Our first task is to find a suitable initial step size t and evaluate f at t. Since we require a relative precision of approximately p bits, we roughly want to take $t \approx \rho_f/2$, where ρ_f is the radius of convergence of f, and evaluate the power series f up to order $n \approx p$. We thus start by the numerical computation of $f_{n;}$ and the estimation $\tilde{\rho}_f$ of ρ_f, based on the numerical Newton polygon of $f_{n;}$. Setting $t := \tilde{\rho}_f/2$, we next compute a tail bound $[\![f_{n;}]\!]_t$. This bound is considered to be of an acceptable quality if

$$[\![f_{n;}]\!]_t \leqslant \epsilon_p [\![f_{;n}]\!]_t.$$

In the contrary case, we keep setting $t := t/2$ and recomputing $[\![f_{n;}]\!]_t$, until we find a bound of acceptable quality. It can be shown [vdH07b] that this process ultimately terminates, when p is sufficiently large. We thus obtain an appropriate initial step size t which allows us to compute a δ-approximation of $f(t)$ with $\delta = 2\epsilon_p [\![f_{;n}]\!]_t +^{\uparrow} (f_{;n}(t))_r$.

In principle, we may now perform the translation $z \mapsto t + z$ and repeat the analytic continuation process using the equation (50). Unfortunately, this approach leads to massive overestimation. Indeed, if the initial condition $f(0)$ is given with relative precision η, then the relative precisions of the computed coefficients f_k are typically a non trivial factor times larger than η, as well as the next "initial condition" $f(t)$ at t. Usually, we therefore lose $O(1)$ bits of precision at every step.

One remedy is the following. Let $\Delta = \Delta_{0,t}$ be the analytic continuation operator which takes an initial condition $c \in \mathbb{C}^d$ on input and returns the evaluation $f(t) \in \mathbb{C}^d$ of the unique solution to (48) with $f(0) = c$. Now instead of computing $f_{;n}$ using a ball initial condition $f(0) \in \mathbb{B}[\mathrm{i}]_p^d$, we rather use its

center $f(0)_c$ as our initial condition and compute $\Delta(f(0)_c)$ using ball arithmetic. In order to obtain a reliable error bound for $f(t)$, we also compute the Jacobian J_Δ of Δ at $f(0)_c$ using ball arithmetic, and take

$$f(t) = \Delta(f(0)_c) + \mathcal{B}(0, [J_\Delta(f(0)_c) f(0)_r]^\uparrow).$$

The Jacobian can either be computed using the technique of automatic differentiation [BS83] or using series with coefficients in a jet space of order one.

With this approach $\Delta(f(0)_c)$ is computed with an almost optimal p-bit accuracy, and $J_\Delta(f(0)_c)$ with an accuracy which is slightly worse than the accuracy of the initial condition. In the lucky case when $J_\Delta(f(0)_c)$ is almost diagonal, the accuracy of $f(t)$ will therefore be approximately the same as the accuracy of $f(0)$. However, if $J_\Delta(f(0)_c)$ is not diagonal, such as in (18), then the multiplication $J_\Delta(f(0)_c) f(0)_r$ may lead to overestimation. This case may already occur for simple linear differential equations such as

$$\begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \int \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f \\ g \end{pmatrix} \tag{51}$$

and the risk is again that we lose $O(1)$ bits of precision at every step.

Let us describe a method for limiting the harm of this manifestation of the wrapping effect. Consider the analytic continuations of f at successive points $0 = t_0 < t_1 < \cdots < t_k$ and denote

$$J_i = J_{\Delta_{t_{i-1}, t_i}}(f(t_{i-1})_c), \qquad i = 1, \ldots, k.$$

Instead of computing the error at t_i due to perturbation of the initial condition $f(0)$ using the formula

$$[f(t)_r]^{\mathrm{per}(0)} = J_k(J_{k-1}(\ldots J_1 f(0)_r \ldots)),$$

we may rather use the formula

$$[f(t)_r]^{\mathrm{per}(0)} = (J_k J_{k-1} \cdots J_1) f(0)_r,$$

where matrix chain products are computed using a variant of binary powering (36):

$$J_j \cdots J_i = (J_j \cdots J_{\lfloor (i+j)/2 \rfloor + 1})(J_{\lfloor (i+j)/2 \rfloor} \cdots J_i).$$

In order to be complete, we also have to take into account the additional error δ_i, which occurs during the computation of $\Delta_{t_{i-1}, t_i}(f(t_{i-1})_c)$. Setting $\delta_0 = f(0)_r$, we thus take

$$f(t_i)_r = \delta_i + J_i \delta_{i-1} + \cdots + (J_k \cdots J_1) \delta_0. \tag{52}$$

When using this algorithm, at least in the case of simple systems such as (51), the precision loss at step k will be limited to $O(\log k)$ bits. Notice that we benefit from the fact that the Jacobian matrices remain accurate as long as the initial conditions remain accurate.

Although we have reduced the wrapping effect, the asymptotic complexity of the above algorithm is cubic or at least quadratic in k when evaluating (52) using a binary splitting version of Horner's method. Let us now describe the final method which requires only $O(k \log k)$ matrix multiplications up till step k. For $0 \leqslant i < j$, we define

$$\begin{aligned} \delta_{i,j} &= \delta_{j-1} + J_{j-1} \delta_{j-2} + \cdots + (J_{j-1} \cdots J_{i+1}) \delta_i. \\ J_{i,j} &= J_{j-1} \cdots J_i, \end{aligned}$$

where $J_0 = 1$. At stage $i = 2^{e_1} + \cdots + 2^{e_l}$ with $e_1 > \cdots > e_l$, we assume that

$$\begin{aligned} \delta_{[j]} &:= \delta_{[i;j]} := \delta_{2^{e_1} + \cdots + 2^{e_{j-1}}, 2^{e_1} + \cdots + 2^{e_j}} \\ J_{[j]} &:= J_{[i;j]} := J_{2^{e_1} + \cdots + 2^{e_{j-1}}, 2^{e_1} + \cdots + 2^{e_j}} \end{aligned}$$

are stored in memory for all $1 \leqslant j \leqslant l$. From these values, we may compute

$$f(t_{i-1})_r = \delta_{[l]} + J_{[l]}(\delta_{[l-1]} + \cdots J_{[3]}(\delta_{[2]} + J_{[2]} \delta_{[1]}) \cdots)$$

using only $O(l) = O(\log i)$ matrix multiplications. Now assume that we go to the next stage $i+1$. If m is maximal such that 2^m divides $i+1$, then $i+1 = 2^{e_1} + \cdots + 2^{e_{l-m}} + 2^m$. Consequently, $\delta_{[i+1;j]} = \delta_{[i;j]}$ and $J_{[i+1;j]} = J_{[i;j]}$ for $j < l' := l - m + 1$ and

$$\begin{aligned}\delta_{[i+1;l']} &= \delta_i + J_i(\delta_{[i;l]} + \cdots J_{[i;l'+2]}(\delta_{[i;l'+1]} + J_{[i;l'+1]}\delta_{[i;l']})\cdots)\\ J_{[i+1;l']} &= J_i J_{[i;l]} \cdots J_{[i;l']}.\end{aligned}$$

Hence, the updated lists $\delta_{[i+1;j]}$ and $J_{[i+1;j]}$ can be computed using $O(m) = O(\log i)$ matrix multiplications. Furthermore, we only need to store $O(\log i)$ auxiliary matrices at step i.

7.6 Discussion

There is a vast literature on validated integration of dynamical systems and reduction of the wrapping effect [Moo65,Moo66,Nic85,Loh88,GS88,Neu93,K8,Loh01,Neu02,MB04]. We refer to [Loh01] for a nice review. Let us briefly discuss the different existing approaches.

The wrapping effect was noticed in the early days of interval arithmetic [Moo65] and local coordinate transformations were proposed as a remedy. The idea is to work as much as possible with respect to coordinates in which all errors are parallel to axes. Hence, instead of considering blocks $x \in \mathbb{B}_p^d$, we rather work with parallelepipeds $x = c + T\mathcal{B}(0,r)$, with $c \in \mathbb{D}_p^d$, $T \in \mathbb{D}_p^{d\times d}$ and $r \in \mathbb{D}_p^d$. A natural choice for T after k steps is $T = J_k \cdots J_1$, but more elaborate choices may be preferred [Loh88]. Other geometric shapes for the enclosures have been advanced in the literature, such as ellipsoids [Neu93], which are also invariant under linear transformations, and zonotopes [K8]. However, as long as we integrate (48) using a straightforward iterative method, and even if we achieve a small average loss $\varepsilon \ll 1$ of the bit precision at a single step, the precision loss after k steps will be of the form $k\varepsilon$.

The idea of using dichotomic algorithms in order to reduce the wrapping effect was first described in the case of linear differential equations [GS88]. The previous section shows how to adapt that technique to non linear equations. We notice that the method may very well be combined with other geometric shapes for the enclosures: this will help to reduce the precision loss to $(\log k)\varepsilon$ instead of $k\varepsilon$ in the above discussion.

Notice that there are two common misunderstandings about the dichotomic method. Contrary to what we have shown above (see also [GS88] in the linear case), it is sometimes believed that we need to keep all matrices $J_1, \ldots, J_k$ in memory and that we have to reevaluate products $J_j \cdots J_i$ over and over again. Secondly, one should not confuse *elimination* and *reduction* of the wrapping effect: if M is the 2×2 rotation matrix by a 30° angle, then none of its repeated squarings M^{2^l} will be the identity matrix, so every squaring $(M^{2^l})^2$ will involve a wrapping. Even though we have *not* eliminated the wrapping effect, we *did* achieve to reduce the number of wrappings to l instead of 2^l.

Taylor models are another approach for the validated long term integration of dynamical systems [EKW84,EMOK91,MB96,MB04]. The idea is to rely on higher k-th order expansions of the continuation operator Δ. This allows for real algebraic enclosures which are determined by polynomials of degree k. Such enclosures are *a priori* more precise for non linear problems. However, the method requires us to work in order k jet spaces in d variables; the mere storage of such a jet involves $\binom{d+k}{d}$ coefficients. From a theoretical point of view it is also not established that Taylor methods eliminate the wrapping effect by themselves. Nevertheless, Taylor models can be combined with any of the above methods and the non linear enclosures seem to be more adapted in certain cases. For a detailed critical analysis, we refer to [Neu02].

Let us finally investigate how sharp good error bounds actually may be. If $\rho_{f(0)} = \|f(0)_r\|/\|f(0)\|_c$ denotes the relative precision of the initial condition at the start and $\rho_{f(t)} = \|f(t)_r\|/\|f(t)\|_c$ the relative precision of the final result, then it is classical that

$$\begin{aligned}\rho_{f(t)} &\geqslant \kappa(J_\Delta(f(0)))\rho_{f(0)},\\ \kappa(J_\Delta(f(0))) &= \|J_\Delta(f(0))\|\|J_\Delta(f(0))^{-1}\|,\end{aligned}$$

where $\kappa(J_\Delta(f(0)))$ is the condition number of the matrix $J_\Delta(f(0))$. We propose to define the condition number $\kappa(\Phi, f(0), 0, t)$ for the integration problem (48) on $[0,t]$ by

$$\mathrm{K} = \kappa(\Phi, f(0), 0, t) = \max_{0\leqslant t_1\leqslant t_2\leqslant t} \kappa(J_{\Delta_{t_1,t_2}}(f(t_1))).$$

Indeed, without using any particular mathematical properties of Φ or f, we somehow have to go through the whole interval $[0, t]$. Of course, it may happen that Φ is indeed special. For instance, if $\Phi = Mf$ for a matrix M with a special triangular or diagonal form, then special arguments may be used to improve error bounds and more dedicated condition numbers will have to be introduced.

However, in general, we suspect that a precision loss of $\log \mathrm{K}$ cannot be avoided. If K gets large, the only real way to achieve long term stability is to take $p > 2 \log \mathrm{K}$ (say), whence the interest of efficient multiple precision and high order ball algorithms. It seems to us that the parallelepiped method should manage to keep the precision loss bounded by $\log \mathrm{K}$, for $p > 2 \log \mathrm{K}$ and $\varepsilon \approx \epsilon_p$. The algorithm from section 7.5 (without coordinate transformations) only achieves a $\log k \log \mathrm{K}$ in the worst case, although a $\log \mathrm{K}$ bound is probably obtained in many cases of interest. We plan to carry out a more detailed analysis once we will have finished a first efficient multiple precision implementation.

References

[Abe80] O. Aberth. *Computable analysis*. McGraw-Hill, New York, 1980.

[AH83] G. Alefeld and J. Herzberger. *Introduction to interval analysis*. Academic Press, New York, 1983.

[ANS08] ANSI/IEEE. IEEE standard for binary floating-point arithmetic. Technical report, ANSI/IEEE, New York, 2008. ANSI-IEEE Standard 754-2008. Revision of IEEE 754-1985, approved on June 12, 2008 by IEEE Standards Board.

[ANS09] ANSI/IEEE. IEEE interval standard working group - p1788. `http://grouper.ieee.org/groups/1788/`, 2009.

[BB85] E. Bishop and D. Bridges. *Foundations of constructive analysis*. Die Grundlehren der mathematische Wissenschaften. Springer, Berlin, 1985.

[BCC+06] Andrea Balluchi, Alberto Casagrande, Pieter Collins, Alberto Ferrari, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. Ariadne: a framework for reachability analysis of hybrid automata. In *Proceedings of the 17th International Symposium on the Mathematical Theory of Networks and Systems*, pages 1269âĂŞ–1267, Kyoto, July 2006.

[BCO+06] A. Bostan, F. Chyzak, F. Ollivier, B. Salvy, É. Schost, and A. Sedoglavic. Fast computation of power series solutions of systems of differential equation. preprint, april 2006. submitted, 13 pages.

[Ber98] M. Berz. Cosy infinity version 8 reference manual. Technical Report MSUCL-1088, Michigan State University, East Lansing, 1998.

[BF00] D. A. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, 23:127–173, 2000.

[BHL00] D.H. Bailey, Y. Hida, and X.S. Li. QD, a C/C++/Fortran library for double-double and quad-double arithmetic. `http://crd.lbl.gov/~dhbailey/mpdist/`, 2000.

[BHL01] D.H. Bailey, Y. Hida, and X.S. Li. Algorithms for quad-double precision floating point arithmetic. In *15th IEEE Symposium on Computer Arithmetic*, pages 155–162, June 2001.

[BK78] R.P. Brent and H.T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 22:317–330, 1983.

[CK91] D.G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

[CT65] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

[CW87] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proc. of the* 19^{th} *Annual Symposium on Theory of Computing*, pages 1–6, New York City, may 25–27 1987.

[DGG+02a] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. Linbox: A generic library for exact linear algebra. In *First Internat. Congress Math. Software ICMS*, pages 40–50, Beijing, China, 2002.

[DGG+02b] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. Project linbox: Exact computational linear algebra. `http://www.linalg.org/`, 2002.

[ea67] U.W. Kulisch et al. Scientific computing with validation, arithmetic requirements, hardware solution and language support. `http://www.math.uni-wuppertal.de/~xsc/`, 1967.

[EKW84] J.-P. Eckmann, H. Koch, and P. Wittwer. A computer-assisted proof of universality in area-preserving maps. *Memoirs of the AMS*, 47(289), 1984.

[EMOK91] J.-P. Eckmann, A. Malaspinas, and S. Oliffson-Kamphorst. A software tool for analysis in function spaces. In K. Meyer and D. Schmidt, editors, *Computer aided proofs in analysis*, pages 147–166. Springer, New York, 1991.

[Für07] M. Fürer. Faster integer multiplication. In *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*, pages 57–66, San Diego, California, 2007.

[GG02] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2-nd edition, 2002.

[Gra91] T. Granlund et al. GMP, the GNU multiple precision arithmetic library. `http://www.swox.com/gmp`, 1991.

[Grz57] A. Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.

[GS88] T.N. Gambill and R.D. Skeel. Logarithmic reduction of the wrapping effect with application to ordinary differential equations. *SIAM J. Numer. Anal.*, 25(1):153–162, 1988.

[Hai95] B. Haible. CLN,a Class Library for Numbers. `http://www.ginac.de/CLN/cln.html`, 1995.

[HLRZ00] G. Hanrot, V. Lefèvre, K. Ryde, and P. Zimmermann. MPFR, a C library for multiple-precision floating-point computations with exact rounding. `http://www.mpfr.org`, 2000.

[HS67] E. Hansen and R. Smith. Interval arithmetic in matrix computations, part ii. *Siam J. Numer. Anal.*, 4:1–9, 1967.

[JKDW01] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis*. Springer, London, 2001.

[K8] W. Kühn. Rigourously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61:47–67, 1998.

[KLRK97] V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Springer, 1997.

[KO63] A. Karatsuba and J. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963.

[KR06] V. Kreinovich and S. Rump. Towards optimal use of multi-precision arithmetic: a remark. Technical Report UTEP-CS-06-01, UTEP-CS, 2006.

[Kul08] U.W. Kulisch. *Computer Arithmetic and Validity. Theory, Implementation, and Applications*. Number 33 in Studies in Mathematics. de Gruyter, 2008.

[Lam07] B. Lambov. Reallib: An efficient implementation of exact real arithmetic. *Mathematical Structures in Computer Science*, 17(1):81–98, 2007.

[Loh88] R. Lohner. *Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendugen*. PhD thesis, Universität Karlsruhe, 1988.

[Loh01] R. Lohner. On the ubiquity of the wrapping effect in the computation of error bounds. In U. Kulisch, R. Lohner, and A. Facius, editors, *Perspectives on enclosure methods*, pages 201–217, Wien, New York, 2001. Springer.

[M0] N. Müller. iRRAM, exact arithmetic in C++. `http://www.informatik.uni-trier.de/iRRAM/`, 2000.

[MB96] K. Makino and M. Berz. Remainder differential algebras and their applications. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational differentiation: techniques, applications and tools*, pages 63–74, SIAM, Philadelphia, 1996.

[MB04] K. Makino and M. Berz. Suppression of the wrapping effect by Taylor model-based validated integrators. Technical Report MSU Report MSUHEP 40910, Michigan State University, 2004.

[Moo65] R.E. Moore. Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solutions to ordinary differential equation. In L.B. Rall, editor, *Error in Digital Computation*, volume 2, pages 103–140. John Wiley, 1965.

[Moo66] R.E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, N.J., 1966.

[Mul06] Jean-Michel Muller. *Elementary Functions, Algorithms and Implementation*. Birkhauser Boston, Inc., 2006.

[Neu90] A. Neumaier. *Interval methods for systems of equations*. Cambridge university press, Cambridge, 1990.

[Neu93] A. Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confedence regions. *Computing Supplementum*, 9:175–190, 1993.

[Neu02] A. Neumaier. Taylor forms - use and limits. *Reliable Computing*, 9:43–79, 2002.

[Nic85] K. Nickel. How to fight the wrapping effect. In Springer-Verlag, editor, *Proc. of the Intern. Symp. on interval mathematics*, pages 121–132, 1985.

[Pan84] V. Pan. *How to multiply matrices faster*, volume 179 of *Lect. Notes in Math.* Springer, 1984.

[Rev01] N. Revol. MPFI, a multiple precision interval arithmetic library. `http://perso.ens-lyon.fr/nathalie.revol/software.html`, 2001.

[Rum99a] S.M. Rump. Fast and parallel inteval arithmetic. *BIT*, 39(3):534–554, 1999.

[Rum99b] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. `http://www.ti3.tu-harburg.de/rump/`.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:352–356, 1969.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.
[vdH02a] J. van der Hoeven. Relax, but don't be too lazy. *JSC*, 34:479–542, 2002.
[vdH+02b] J. van der Hoeven et al. Mathemagix, 2002. `http://www.mathemagix.org`.
[vdH06a] J. van der Hoeven. Computations with effective real numbers. *TCS*, 351:52–60, 2006.
[vdH06b] J. van der Hoeven. Newton's method and FFT trading. Technical Report 2006-17, Univ. Paris-Sud, 2006. Submitted to JSC.
[vdH07a] J. van der Hoeven. New algorithms for relaxed multiplication. *JSC*, 42(8):792–802, 2007.
[vdH07b] J. van der Hoeven. On effective analytic continuation. *MCS*, 1(1):111–175, 2007.
[vdH08a] J. van der Hoeven. Making fast multiplication of polynomials numerically stable. Technical Report 2008-02, Université Paris-Sud, Orsay, France, 2008.
[vdH08b] J. van der Hoeven. Meta-expansion of transseries. In Y. Boudabbous and N. Zaguia, editors, *ROGICS '08: Relations Orders and Graphs, Interaction with Computer Science*, pages 390–398, Mahdia, Tunesia, May 2008.
[vdHMT] J. van der Hoeven, B. Mourrain, and Ph. Trébuchet. Efficient and reliable resolution of eigenproblems. In preparation.
[Wei00] K. Weihrauch. *Computable analysis*. Springer-Verlag, Berlin/Heidelberg, 2000.

Recursive Analysis Complexity in Terms of Discrete Complexity

Olivier Bournez[1], Walid Gomaa[2,3], and Emmanuel Hainry[2,4]

[1] Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France Olivier.Bournez@lix.polytechnique.fr
[2] Loria, BP 239 - 54506 Vandœuvre-lès-Nancy Cedex, France
walid.gomaa@loria.fr,Emmanuel.Hainry@loria.fr
[3] Alexandria University, Faculty of Engineering, Alexandria, Egypt
[4] Nancy Université, Université Henri Poincaré, Nancy, France

1 Introduction

Recursive analysis is the most classical approach to model and discuss computations over the reals. It is usually presented using Type 2 or higher order Turing machines: see e.g. monograph [3]. Recently, it has been shown that computability classes of functions computable in recursive analysis can also be defined (or characterized) in an algebraic machine independent way, without resorting to Turing machines. In particular nice connections between the class of computable functions (and some of its sub- and sup-classes) over the reals and algebraically defined (sub- and sup-) classes of $\mathbb{R}$-recursive functions à la Moore 96 have been obtained: see e.g. survey [2].

However, until now, this has been done only at the computability level, and not at the complexity level.

We recently studied a framework that allows us to dive into the complexity level of functions over the reals. In particular we provide the first algebraic characterization of polynomial time computable functions over the reals. This framework opens the field of implicit complexity of functions over the reals, and also provide a new reading of some of the existing characterizations at the computability level.

In this extended abstract, we present some of our results. We divide our discussion into two parts: the first deals with the easier case of Lipschitz functions and the other part deals with the more difficult general case.

2 Lipschitz Functions

The Lipschitz condition provides the continuity information necessary for computability in recursive analysis. In the following we define a notion of *approximation* that will be used to relate integer computability/complexity to the corresponding analog computability/complexity.

Definition 1 (Approximation) *Let $\mathcal{C}$ be a class of functions from $\mathbb{R}^2$ to $\mathbb{R}$. Let $\mathcal{D}$ be a class of functions from $\mathbb{N}^2$ to $\mathbb{N}$. Assume a function $f : [0,1] \to \mathbb{R}$.*

1. *We say that $\mathcal{C}$* approximates *$\mathcal{D}$ if for any function $g \in \mathcal{D}$, there exists some function $\tilde{g} \in \mathcal{C}$ such that for all $x, y \in \mathbb{N}$ we have*
$$|\tilde{g}(x,y) - g(x,y)| \leq 1/4 \tag{1}$$
2. *We say that f is $\mathcal{C}$-*definable *if there exists a function $\tilde{g} \in \mathcal{C}$ such that the following holds*
$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq y \colon |\tilde{g}(x,y) - yf(\frac{x}{y})| \leq 3 \tag{2}$$

We then have the following result which relates analog complexity over $[0,1]$ to approximate complexity over $\mathbb{N}^2$.

Theorem 1 *Consider a class $\mathcal{C}$ of polytime computable real functions that approximates the class of polytime computable discrete functions. Assume that $f : [0,1] \to \mathbb{R}$ is Lipschitz. Then f is polytime computable iff f is $\mathcal{C}$-definable.*

3 The General Case

In order to provide for the continuity condition we need another notion of 'approximation' which is a sort of converse to that given in Definition 1.

Definition 2 (Polytime computable integer approximation) *A function $g : \mathbb{R}^d \to \mathbb{R}$ is said to have a* polytime computable integer approximation *if there exists some polytime computable function $h : \mathbb{N}^d \to \mathbb{N}$ with $|h(\bar{x}) - g(\bar{x})| \leq 1$ for all $\bar{x} \in \mathbb{N}^d$.*

A sufficient condition is that the restriction of g to $\mathbb{N}^2$ is polytime computable. Assume a function $T\colon \mathbb{N} \to \mathbb{N}$ and define $\#_T : \mathbb{R}^{\geq 1} \to \mathbb{R}$ by $\#_T[x] = 2^{T(\lfloor \log_2 x \rfloor)}$. When T is a polynomial function with $T(x) = \Theta(x^k)$ we write $\#_k$ to simplify the notation. As will be seen below the function $\#_T$ is used to quantify the smoothness of real functions (it plays a role similar to that of the modulus of continuity). In the following we define a class of real functions that are well-behaved with respect to their restrictions to the integers. (For ease of notation, we will use $[a, b]$ to denote both $[a, b]$ and $[b, a]$.)

Definition 3 (Peaceful functions) *A function $g : \mathbb{R}^2 \to \mathbb{R}$ is said to be* peaceful *if*

$$\forall x \in \mathbb{R}^{\geq 0}, \forall y \in \mathbb{N}\colon g(x, y) \in [g(\lfloor x \rfloor, y), g(\lceil x \rceil, y)] \tag{3}$$

We say that a class $\mathcal{C}$ of real functions peacefully approximates *some class $\mathcal{D}$ of integer functions, if the subclass of peaceful functions of $\mathcal{C}$ approximates $\mathcal{D}$.*

Definition 4 *Let $\mathcal{C}$ be a class of functions from $\mathbb{R}^2$ to $\mathbb{R}$. Assume a function $f : [0, 1] \to \mathbb{R}$ and a function $T\colon \mathbb{N} \to \mathbb{N}$.*

1. *We say that f is T-$\mathcal{C}$-*definable *if there exists some peaceful function $g \in \mathcal{C}$ such that*

$$\forall x \in \mathbb{N}, \forall y \in \mathbb{N}^{\geq 1}, x \leq \#_T[y]\colon |g(x, y) - yf(\frac{x}{\#_T[y]})| \leq 2, \tag{4}$$

2. *We say that f is T-*smooth *if there exists some integer M such that*

$$\begin{aligned} \forall x, x' \in \mathbb{R}^{\geq 0}, &\forall y \in \mathbb{R}^{\geq 1}, x, x' \leq \#_T[y]\colon \\ &|x - x'| \leq 1 \Rightarrow y|f(\frac{x}{\#_T[y]}) - f(\frac{x'}{\#_T[y]})| \leq M \end{aligned} \tag{5}$$

Now we have the necessary tools to relate analog complexity of arbitrary functions over $[0, 1]$ to approximate complexity over $\mathbb{N}^2$.

Theorem 2 *Consider a class $\mathcal{C}$ of real functions that peacefully approximates polytime computable discrete functions, and whose functions have polytime computable integer approximations.*[5] *Then the following are equivalent:*

1. *a function $f\colon [0, 1] \to \mathbb{R}$ is polytime computable,*
2. *there exists some integer k such that*
   (a) *f is n^k-$\mathcal{C}$-definable,*
   (b) *f is n^k-smooth.*

Remark 1 *Note that all the previous results can be generalized to any complexity and computability class.*

[5] A sufficient condition for that is restrictions to integers of functions from $\mathcal{C}$ are polytime computable.

4 Definability by a Function Algebra

A function algebra is the smallest class of functions that consists of a set of basic functions and their closure under a set of operations. Functions algebra have been used in the discrete computation context to give a machine independent characterization of computability and complexity-theoretic classes. They have also been used in the recursive analysis context to give characterizations of computability classes. However, to the best of our knowledge, no such work has been done at the complexity level. The above results can be used to contribute to that end as follows below.

we define a class of real functions which are essentially extensions to $\mathbb{R}$ of the Bellantoni-Cook class [1]. This latter class was developed to exactly capture discrete polytime computability in an algebraic machine-independent way. In the next definition any function $f(x_1, \ldots, x_m; y_1, \ldots, y_n)$ has two types of arguments (see [1]): *normal* arguments which come first followed by *safe* arguments using ';' for separation. For any $n \in \mathbb{Z}$ we call $[2n, 2n+1]$ an even interval and $[2n+1, 2n+2]$ an odd interval.

Definition 5 *Define the function algebra*

$$\mathcal{W} = [0, 1, +, -, U, p, c, parity; SComp, SI] \tag{6}$$

1. *zero-ary functions for the constants* 0 *and* 1,
2. *a binary addition function:* $+(; x, y) = x + y$,
3. *a binary subtraction function:* $-(; x, y) = x - y$,
4. *a set of projection functions* $U = \{U_i^j : i, j \in \mathbb{N}, i \leq j\}$ *where:*

$$U_i^{m+n}(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+n}) = x_i,$$

5. *a continuous predecessor function* p *defined as follows:*

$$p(; x) = \begin{cases} n & x \in [2n, 2n+1] \text{ for some } n \in \mathbb{Z} \\ x - n - 1 & x \in [2n+1, 2n+2] \end{cases} \tag{7}$$

Note that when x *belongs to an even interval* $p(; x)$ *acts exactly like* $\lfloor \frac{x}{2} \rfloor$. *Otherwise, the function is piecewise linear between even intervals.*

6. *a piecewise linear continuous conditional function* c *defined as follows:*

$$c(; x, y, z) = \begin{cases} y & x \geq 1 \\ z & x \leq 0 \\ xy + (1 - x)z & 0 \leq x \leq 1 \end{cases} \tag{8}$$

7. *a continuous parity function:*

$$parity(; x) = \begin{cases} 4(x - 2n) & x \in [2n, 2n + \frac{1}{2}] \text{ for some } n \in \mathbb{Z} \\ 4(2n + 1 - x) & x \in [2n + \frac{1}{2}, 2n + 1] \\ 0 & ow \end{cases} \tag{9}$$

Hence, $parity(; x)$ *is non-zero if and only if* x *lies inside an even interval. Furthermore, for any* $n \in \mathbb{Z}$ *the following holds:* $\int_{2n}^{2n+1} parity(; x)dx = 1$.

8. *a safe composition operator SComp: assume a vector of functions* $\bar{g}_1(\bar{x};) \in \mathcal{W}$, *a vector of functions* $\bar{g}_2(\bar{x}; \bar{y}) \in \mathcal{W}$, *and a function* $h \in \mathcal{W}$ *of arity* $len(\bar{g}_1) + len(\bar{g}_2)$ *(where len denotes the vector length). Define a new function* f

$$f(\bar{x}; \bar{y}) = h(\bar{g}_1(\bar{x};); \bar{g}_2(\bar{x}; \bar{y})) \tag{10}$$

It is clear from the asymmetry in this definition that normal arguments can be repositioned in safe places whereas the opposite can not happen and that is what controls the computational complexity of the functions generated inside the class.

9. *a safe integration operator SI: assume functions* $g, h_0, h_1 \in \mathcal{W}$. *Let* $p'(;x) = p(;x-1)+1$. *Consider the following differential equation:*

$$\begin{aligned} f(0,\bar{y};\bar{z}) =& g(\bar{y};\bar{z}) \\ \partial_x f(x,\bar{y};\bar{z}) =& \, parity(x;)[h_1(p(x;),\bar{y};\bar{z}, f(p(x;),\bar{y};\bar{z})) \\ & - f(2p(x;),\bar{y};\bar{z})] \\ & + parity(x-1;)[h_0(p'(x;),\bar{y};\bar{z}, f(p'(x;),\bar{y};\bar{z})) \\ & - f(2p'(x;)-1,\bar{y};\bar{z})] \end{aligned} \tag{11}$$

Notice that for simplicity we misused the basic functions (and p'*) so that their arguments are now in normal positions (the alternative is to redefine a new set of basic functions with arguments in normal positions).*

The class $\mathcal{W}$ satisfies nice properties that give rise to the following crucial result.

Theorem 3 *1. A Lipschitz function* $f : [0,1] \to \mathbb{R}$ *is polytime computable iff it is* $\mathcal{W}$*-definable.*
2. Let $f : [0,1] \to \mathbb{R}$ *be some* n^k*-smooth function for some* k. *Then* f *is polytime computable iff it is* n^k*-*$\mathcal{W}$*-definable.*

Notice that $\mathcal{C}$-definability of a function can be seen as a schema that builds a function f from a function $\tilde{g} \in \mathcal{C}$ (see definition of $\mathcal{C}$-definability). Hence, the class of polytime computable functions can be algebraically characterized in a machine-independent way as follows.

Corollary 1 *Let* $Def[\mathcal{C}]$ *stands for* $\mathcal{C}$*-definability. Then a function* $f\colon [0,1] \to \mathbb{R}$ *is polytime computable iff either*

1. f *is Lipschitz and belongs to* $Def[0,1,+,-,U,p,c,parity;SComp,SI]$ *or*
2. f *is* n^k*-smooth and belongs to* n^k*-*$Def[0,1,+,-,U,p,c,parity;SComp,SI]$.

References

1. Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.
2. Olivier Bournez and Manuel L. Campagnolo. *New Computational Paradigms. Changing Conceptions of What is Computable*, chapter A Survey on Continuous Time Computations, pages 383–423. Springer, New York, 2008.
3. Klaus Weihrauch. *Computable Analysis: an Introduction.* Springer, 2000.