ISSN 2186-7437

NII Shonan Meeting Report

No. 151

Higher-order Complexity Theory and its Applications

Bruce M. Kapron
Akitoshi Kawamura
Florian Steinberg

October 7–10, 2019

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Higher-order Complexity Theory and its Applications

Organizers:
Bruce M. Kapron (University of Victoria)
Akitoshi Kawamura (Kyushu University)
Florian Steinberg (INRIA)

October 7–10, 2019

The theory of higher-order complexity is an emerging field which is still in its infancy. More researchers across a broad range of research areas are asking questions that may best be addressed in this framework, or are using established tools and results, but in an isolated way. The objective of this workshop is to both strengthen the foundations and forge stronger connections between the foundations and the various areas in which the theory is now being applied. While there is quite a broad range of application areas, we will focus on three: computable analysis, programming language theory, and NP search complexity. We are also inviting researchers who work in related areas (e.g., type theory, logic, computability, security).

Introduction

The theory of higher-order complexity is an emerging field which is still in its infancy. More researchers across a broad range of research areas are asking questions that may best be addressed in this framework, or are using established tools and results, but in an isolated way. The objective of this workshop is to both strengthen the foundations and forge stronger connections between the foundations and the various areas in which the theory is now being applied. While there is quite a broad range of application areas, we will focus on three: computable analysis, programming language theory, and NP search complexity. We are also inviting researchers who work in related areas (e.g. type theory, logic, computability, security.) Foundations

Traditional computational complexity theory is concerned with computation by Turing machines over objects with finite representations, such as finite graphs, or natural numbers. Higher-order complexity extends this to infinitary domains, for example real numbers, while retaining finitary characteris-tics of the standard theory. This is achieved through the use of oracle Turing machines (OTMs), which allow oracle access to infinitary inputs but with finitary internal state and finite computations. OTMs are a well understood model of computation, and have been widely applied in computability theory, computable anal-ysis, and ordinary complexity. Once we introduce complexity notions into this model the situation becomes more complex.

One particular goal of higher-order complexity has been to provide a notion of feasibility, or poly-time computability, in this setting. While early work in the 1970's by Constable and Mehlhorn made partial progress, there was no simple OTM-based characterization. A breakthrough came with the work of Kapron and Cook in the 1990s, which uses second-order polynomials, and defines an appropriate size measure for function inputs. This work is the basis for much of the subsequent work in type-two complexity, as well as generalizations to higher levels of the type hierarchy. The model has been applied in a range of areas, including computable analysis, theory of programming languages, and ordinary complexity theory, especially the complexity of search problems.

Complexity in Computable Analysis

Computable analysis is the accepted theory of computability for continuous structures. It is well-developed and by nature it is closely related to numerical analysis. Theorems from computable analysis often reflect what is known heuristically in numerical computing, but unprovable. However, numerics is not only interested in providing correctness of algorithms but usually aims to optimize the performance of such algorithms. Computable analysis relies on higher-order computability theory to be able to model computations on uncountable structures. Developing a complexity theory for computable analysis necessitates the use of higher-order complexity theory.

The most popular framework for complexity considerations in computable analysis is the framework of second-order representations introduced by Kawamura and Cook. This framework relies on second-order complexity theory, but to avoid technical complications that arise from the necessity to use partial operators, it only uses a fragment of the theory. New methods are required

for dealing with the partiality that seems unavoidable for applications in computable analysis.

Programming Language Theory

The tools of traditional complexity theory have been quite successful in complexity analyses of small, low-level programs, especially those written in an imperative style. However, extensions to larger-scale programs lead in a natural way to a more abstract and compositional approach, which while making reasoning about correctness more tractable, can obscure information required for com-plexity analysis. Moreover, alternate paradigms such as object-oriented or functional programming require a different approach than for imperative programs, and naturally lead to models inolving higher-type com-putation. There is a rich theory in this area involving techniques from implicit complexity, type theory, rewriting sytems and semantics, among others. Higher-type complexity theory is an essential component in addressing the challenge of providing good complexity models in this setting. Search Complexity

Traditional complexity theory deals with computation over finite structures, but even in this setting there are situations where finite inputs are "too large" to be accessed in an efficient way, in particular in the study of NP-search problems. The class TFNP, introduced by Papadimitriou in the 1990s, consists of search problems for which a small, efficiently verifiable solution is always guaranteed to exist. The question that arises in this setting is the difficulty of efficiently finding a solution. This is a natural model for search algorithms having local access to a large object (e.g. the edges of a graph). One approach is to provide access via a circuit representing the graph. Subsequently, Beame et al. noted these problems may be modeled in a type-two setting, where the large object is presented as a function. This has led to approaches from a diversity of perspectives, including proof theory, the theory of generic oracles, and cryptography.

Outcomes

A major goal of the workshop was to bring together researchers interested in higher-order complexity but with different perspectives in terms of approach an application area. This goal was achieved through talks which addressed a range of topics across the area. The talks were well-attended by all the participants, and generated questions and discussions leading to further interaction. In some cases researchers were able to make potential connections between work arising in different contexts, for example, with respect to notions of continuity arising in feasible analysis (M. Ziegler) and in programming languages (J. Royer.) Another example is in the application of logical relations, appearing explicitly in the work of U. Berger, but also significant in the step algebras as presented by C. Freer.

Overview of Talks

Entropy and Computational Complexity of Continuous Data

Martin Ziegler (KAIST)

The Complexity Theory of real numbers, functions, and subsets provides a rigorous foundation to numerical computation. How to extend the Turing model of computation and cost measure to more general data types, such as elements of Sobolev space or functionals, remains under debate. In the discrete setting, computational cost is measured in dependence of the length of the input – and in "output-sensitive" analysis also that of the output: When either is 'large', it seems only fair to allot more resources to a computation and still consider it efficient. Put differently, algorithmic efficiency should be assessed RELATIVE to 'trivial' / information-theoretic lower bounds. Paradigm 1: For a computationally well-posed problem, the machine model and parameterized cost analysis should be DESIGNED such that, relative to a suitable oracle (containing answers to all instances), the problem admits what is to be considered a polynomial-time solution. In particular for the entire "type" hierarchy of compact metric spaces, $Y := [0,1], Y' := \{1\text{-Lipschitz functions } f : Y \to [0,1]\}, Y'', ...$ we WANT the application functional $X' \times X \ni (f,x) \mapsto f(x) \in [0,1]$ to be polynomial-time computable. With computation over continuous data arise two additional types of information-theoretic lower bounds against which to gauge algorithmic cost: i) Instance-specific ones, such as the 'size' of a string function argument given as oracle, or the modulus of continuity of a function argument. ii) Problem-specific ones, such as Kolmogorov's metric entropy of the input/output spaces under consideration. Weihrauch's TTE/stream theory of computation for example uses (subsets of) Cantor space to encode (subsets of) Euclidean space: both having linear entropy. Kawamura and Cook's second-order Complexity Theory of Operators in Analysis uses (subsets of) Baire space to encode (subsets of) continuous functions: both having exponential entropy. Steinberg's Lemma connects entropy and modulus of continuity; and our quantitative strengthening of "admissibility" and the Kreitz-Weihrauch "Main" Theorem connect both further to computational cost. (Joint work with Akitoshi Kawamura, Donghyun Lim, Svetlana Selivanova)

Uniform limits of conditionally computable real functions

Ivan Georgiev (Sofia University)

We consider two notions for computability of real functions - uniform and conditional computability with respect to some subrecursive complexity class. We give examples of properties which are common for both notions. However, preservation under uniform limits is not such a property, since uniform limits of conditionally computable real functions may fail to be conditionally computable and thus they form a broader class. We discuss some open questions concerning this new class.

Complexity of type-3 sequential functionals

Bruce Kapron (University of Victoria)

One difficulty encountered in formulating higher-order complexity models is that inputs may not be bounded (e.g. in size). This may lead to problematic definitions, such as the Cook-Kapron notion of "function size" needed for type-2 BFF. One way around that particular problem is to restrict our attention to functions with poly-bounded growth (as is done by Townsend.) We show that when moving to type-3 this may not be enough. In particular, in a sequential model where type-2 inputs are treated as decision trees, we again are led to a notion of function size for these inputs, even when type-1 input are poly-bounded.

Feasible corecursion and restricted notions of uniform continuity

Jim Royer (Syracuse University)

We define S, a total functional programming language based on structural recursions and corecursions. S is roughly as computationally powerful as Gödel's System T. We then impose a Bellantoni-Cook-style ramification on S_1, the type-level 1 fragment of S, with the goal of obtaining a polynomial-time version of S_1. With not too much trouble we can achieve this goal if we restrict to structural recursions, but when corecursions allowed, things fail badly. An analysis shows that if recursions and corecursions obey certain moduli of uniform continuity constraints, then the goal is achievable in a way that permits many standard corecursions within the system.

A tier-based typed programming language characterizing feasible functionals

Romain Péchoux (LORIA/Université de Lorraine)

The class of Basic Feasible Functionals BFF_2 is the type-2 counterpart of the class FP of functions computable in polynomial time. Several characterizations have been suggested in the literature, but none of these present a simple programming language guaranteeing this complexity bound. We give a characterization of BFF_2 based on a simple imperative language with oracle calls using a tier-based type system whose inference is decidable. The low complexity (cubic in the size of the program) of the type inference algorithm contrasts with the intractability of the aforementioned methods and does not restrain strongly the expressive power of the language.

Size types for parallel complexity

Alexis Ghyselen (ENS de Lyon)

Type systems as a way to control or analyze programs have been largely studied in the context of functional programming languages. Some of those work

allow to extract from a typing derivation for a program a complexity bound on this program. We present how to adapt this result for parallel complexity in the pi-calculus, as a model of concurrency and parallel communication. We study two notions of time complexity: the total computation time without parallelism (the work) and the computation time under maximal parallelism (the span). We will illustrate those definitions on examples, and then define reduction relations in the pi-calculus to capture those notions. Finally, we will express that we can give type systems, based on input/output size types, giving a complexity bound for those notions of complexity on a process.

A functorial approach to complexity theory?

Damiano Mazza (CNRS, Université Paris 13)

Many-one reductions are a key notion in structural complexity theory. As many basic concepts in complexity, the definition is disarmingly simple and yet gives rise to an wondrously intricate theory. However, if we look at it from the point of view of category theory, we discover that the definition itself, or at least a good categorical definition, becomes a matter for consideration already. I will report on work in progress concerning a possible approach to structural complexity suggested by such considerations, consisting of a blend of descriptive complexity with categorical logic.

Cost recurrence extraction for higher-order languages

Norman Danner (Weslyan University)

In earlier work with Jim Royer, we developed a logical-relations technique for extracting type-2 polynomial bounds from a formalism for feasible type-2 computation. Dan Licata, Ramyaa and I adapted that technique to extract provably correct cost recurrences from programs in a higher-order language with inductive types and structural recursion. The process consists of two stages: extraction of a syntactic recurrence in a call-by-name-like recurrence language that permits typical mathematical reasoning, followed by interpretation in a model of the recurrence language. The composition yields recurrences that are comparable to those that arise from typical informal analyses, but provably give upper bounds on the cost and size of the result. I will describe the basic technique and key ideas and discuss recent results and extensions to handle general recursion and let-polymorphism.

Resource-aware programming ... or what can we learn from Spectre and Meltdown

Georg Moser (University of Innsbruck)

Spectre and Meltdown (among other things) highlighted that a non-functional property of algorithms, their resource usage, can be drastically exploited to breach security. Thus making strong assurances on the security of a computing systems requires (detailed) performance information.

In programming languages, such observations are typically studied in the context of resource analysis, where a static analysis aims to guarantee that a certain (higher-order) program behaves in terms of resource usage.

However, what about a more fundamental change in programming? Would it be too much to ask of the programmer to provide resource annotations in her code? This would treat resources as first-class citizens in programming, eg. as part of the type system. In the talk, I'll briefly review recent work on sophisticated type systems to this effect that could form the logical basis of such a paradigm change in programming.

Type-2 Computability, Generic Oracles, and the Linear Space Hypothesis

Tomoyuki Yamakami (University of Fukui)

This seminal talk begins with a quick explanation of old results on type-2 computability and its close relationship to generic oracles. The second part of the talk covers a recent progress on sub-linear-space computability and short Turing reductions using polynomial time and sub-linear space. This notion helps introduce the linear space hypothesis, which asserts the insolvability of a restricted form of $2SAT$ (called $2SAT_3$) using polynomial time and sub-linear space. In the end, we give a few applications of this working hypothesis.

Complexity of partial differential equations

Donghyun Kim (KAIST)

In the framework of Type-2 Theory of Effectivity, representations of continuous spaces affect computability and computational complexity of problems drastically. We propose "quantitative admissibility" as a criterion for sensible representations. Quantitative admissibility is a refinement of classical admissibility notion by Kreitz and Weihrauch, 1985. Classical setting of second-countable T_0 spaces is concretized to totally bounded metric spaces. We show that there is a close correspondence between modulus of continuity of a function and that of its realizer when the representations are quantitatively admissible.

Logical relations and feasibility in higher types

Ulrich Berger (University of Swansea)

How can the polytime functions on natural numbers be extended to higher-types in a natural way such that the resulting hierarchy is λ-*closed*, that is, closed under definability by typed λ-terms? This question has been answered in various ways by Cobham, Cook, Kapron, Royer and others. Up to type level 2 all of the proposed hierarchies turned out to coincide and are considered to define the 'right' notion of type 2 feasibility. At types beyond level 3 it is unclear what is the best extension.

In this talk we study the possibility of using *logical relations* to define feasibility in higher types. We generalize logical relations and its Fundamental

Theorem in a 'Banach-Mazur style' such that a λ-closed hierarchy can be defined from any starting type level. For a given hierarchy P defined up to some finite type level, the resulting extension P^∞ is lexicographically maximal (w.r.t. set inclusion) among all λ-closed extensions of P. In general P^∞ need not be computable, even if P is. However, for **PTIME**, the polynomial time computable functions of type level 1, $\mathbf{PTIME}^\infty$ *is* computable. We show that $\mathbf{BFF}(\mathbf{2})$, the class of basic feasible functionals of type 2, lies strictly between the minimal and maximal extensions of **PTIME** (the latter being $\mathbf{PTIME}^\infty$ restricted to type level 2). At type levels ≥ 3 the relation between **BFF** and $\mathbf{PTIME}^\infty$ is yet unknown.

Complexity of partial differential equations

Svetlana Selivanova (KAIST)

We study computational complexity of finding solutions to partial differential equations (PDEs) by bridging their existing rich theory and known algorithms with second-order complexity theory on reals. For linear evolutionary systems of PDEs we establish, given natural assumptions, PSPACE upper bounds, measuring the complexity on n, where 2^{-n} is the output precision. This probably can not be improved in general, but can be under additional restrictions: for analytic initial data we get a PTIME algorithm; for some constant coefficient cases our PSPACE algorithm boils down to GAP P even for nonanalytic (but still smooth enough) functions. Joint work with Ivan Koswara, Gleb Pogudin and Martin Ziegler.

A notion of a computational step for Partial Combinatory Algebras

Cameron Freer (MIT)

Working within the general formalism of a partial combinatory algebra (or PCA), we introduce and develop the notion of a step algebra, which enables us to work with individual computational steps, even in very general and abstract computational settings. We show that every partial applicative structure is the closure of a step algebra obtained by repeated application, and identify conditions under which this closure yields a PCA. Joint work with Nathanael Ackerman.

Automating Sized-Type Inference for Complexity Analysis

Martin Avanzini (INRIA Sophia-Antipolis)

One successful approach to automatic verification of termination properties of higher-order functional programs is based on sized-types, and has been shown to be quite robust and amenable to automation. In sized-types, a type carries not only information about the kind of each object, but also about its size, hence the name. This information is then exploited when requiring that recursive calls are done on arguments of strictly smaller size. Estimating the size of intermediate results is also a crucial aspect of complexity analysis. However,

up to now, the only attempt of using sized-types for complexity analysis is due to Vasconcelos, and confined to space complexity. The sized-types system, by itself, does not guarantee any complexity-theoretic property on the typed program, except for the size of the output being bounded by a certain function on the size of the input. Complexity analysis of a program P can however be seen as a size analysis of an instrumented program which computes not only P, but its complexity. In this talk, I will introduce a novel sized-type system aimed towards automated complexity analysis. In particular, I will outline a type inference procedure which reduced sized-type inference to a problem of solving a set of inequalities. These inequalities can in turn be solved by standard techniques employed, e.g., in the synthesis of ranking functions. Despite undecidability of sized-type inference in general, the proposed inference procedure is relative complete in the sense that a given program is typeable if and only if the generated set of equalities can be solved.

List of Participants

- Martin Avanzini, INRIA Sophia-Antipolis
- Arnold Beckman, Swansea University
- Ulrich Berger, Swansea University
- Sam Buss, University of California, San Diego
- Ugo Dal Lago, University of Bologna/INRIA Sophia-Antipolis
- Norman Danner, Wesleyan University
- Hugo Feree, University of Kent
- Cameron Freer, Massachusetts Institute of Technology
- Ivan Dimitrov Georgiev, Sofia University
- Alexis Ghyselen, ENS Lyon
- Emmanuel Hainry, LORIA/Université de Lorraine
- Mathieu Hoyrup, LORIA
- Bruce M. Kapron, University of Victoria
- Akitoshi Kawamura, Kyushu University
- Takayuki Kihara, Nagoya University
- Donghyun Lim, KAIST
- Jean-Yves Marion, LORIA/Université de Lorraine
- Damiano Mazza, CNRS
- Georg Moser, University of Innsbruck
- Norbert Müller, Trier University

- Arno Pauly, Swansea University
- Romain Péchoux, LORIA/Université de Lorraine
- James Royer, Syracuse University
- Thomas Seiller, CNRS
- Svetlana Selivanova, KAIST
- Florian Steinberg, INRIA Sophia Antipolis
- Holger Thies, Kyushu University
- Hideki Tsuiki, Kyoto University
- Yoriyuki Yamagata, National Institute of Advanced Industrial Science and Technology
- Tomoyuki Yamakami, University of Fukui
- Martin Ziegler, KAIST

Meeting Schedule

Day 1: October 7

Morning		
9:00-9:10	Welcome / Introductory film	
9:10-10:10	Martin Ziegler	Entropy and computational complexity of continuous data
10:10-11:00	Break	
11:00-11:30	Ivan Georgiev	Uniform limits of conditionally computable real functions
11:30-12:00	Bruce Kapron	Complexity of type 3 sequential functionals

Afternoon		
14:00-14:30	Jim Royer	Feasible corecursion and restricted notions of uniform continuity
16:00-16:30	Romain Péchoux	A tier-based typed programming language characterizing feasible functionals
15:00-16:00	Break	
16:30-17:00	Alexis Ghyselen	Size types for parallel complexity
17:00-18:00	Open problem session I	

Day 2: October 8

Morning		
9:00-9:30	Hugo Férée	A game semantics approach to higher order complexity
9:30-10:00	Thomas Seiler	
10:00-10:30	Ugo dal Lago	On higher-order cryptography
10:30-11:00	Break	
11:00-11:30	Damiano Mazza	A functorial approach to complexity theory?
11:30-12:00	Arno Pauly	Polynomial-time Weirauch reducibility

Afternoon		
14:00-15:00	Sam Buss	Polynomial Time Computable Set Functions
15:30-16:00	Break	
16:00-16:30	Norman Danner	Cost recurrence extraction for higher-order languages
16:30-17:00	Georg Moser	Resource-aware programming ... or what can we learn from Spectre and Meltdown

Day 3: October 9

Morning		
9:00-10:00	Tomoyuki Yamakami	Type-2 Computability, Generic Oracles, and the Linear Space Hypothesis
10:00-10:30	Donghyun Lim	Complexity of partial differential equations
10:30-11:00	Break	
11:00-11:20	Ulrich Berger	Higher-order complexity via logical relations
11:20-12:00	Open Problem Session II	

Day 4: October 10

Morning		
9:00-9:30	Svetlana Selivanova	Complexity of partial differential equations
9:30-10:00	Cameron Freer	Step algebras
10:00-10:30	Martin Avancini	Automating Sized-Type Inference for Complexity Analysis
10:30-11:00	Break	
11:00-12:00	Closing / Future Directions	