AMS-UMI International
Joint Meeting 2024
Palermo, July 23–26, 2024

Proof-Theory and Theoretical Computer Science
Special Session A25

Ugo Dal Lago
Università di Bologna, ITALY

Marco Gaboardi
Boston University, USA

Stéphane Graham-Lengrand
SRI International, USA

Luca Paolini
Università degli Studi di Torino, ITALY

Antonio Piccolomini d'Aragona
Università di Siena, ITALY

Lorenzo Tortora de Falco
Università degli Studi Roma Tre, ITALY

After the unquestionable role played by Logic in the birth of Computer Science at the beginning of last century, a second period of strong interactions between one among the oldest disciplines of human knowledge and one among the most recent ones occurred in the sixties with the discovery of the so-called Curry-Howard correspondence between proofs (of a particular logical system) and programs (of a paradigmatic programming language). Since then a wide interdisciplinary scientific community has grown all over the world.

An example of the fruitfulness of this interaction is type theory: the old idea of Russell to avoid paradoxes of set theory was later used in the framework of Church's Lambda-Calculus to control its computational power. Later on, Martin Löf proposed intuitionistic type theory as a foundation for constructive mathematics and Thierry Coquand introduced the calculus of constructions, at the base of the interactive theorem prover Coq. The vitality of type theory is also witnessed by the recent inception of Homotopy Type Theory (HoTT) in the landscape.

Another striking example is the birth of Linear Logic (at the end of the eighties) that is at the crossroad of traditional Proof Theory and Theoretical Computer Science: on the one hand it reveals a new structure underlying Logic and computational processes in general thus contributing to Proof Theory and Theoretical Computer Science in the most fundamental sense, and on the other hand it brings new concepts and new tools that have been used in various research areas in the last decades (theory of programming languages, implicit computational complexity, concurrency, game semantics, category theory, philosophy, linguistics,...).

In the proposed special session, we aim at bringing together european and american experts in the field. A list of topics of interest of the session include:

- Linear Logic and its applications;
- Type theory, including Homotopy Type Theory;
- Category Theory in Proof Theory and in Programming Languages;
- Extensions of the Curry-Howard Paradigm, Session Types and Concurrency;
- Proof complexity, Implicit computational complexity, Bounded Arithmetics;
- Automated reasoning: Proof compression, decidability and decision procedures, tableaux systems;
- Proof exchange, concept alignment and proof assistant interoperability;
- Realizability, Semantic types, Constructive Semantics, Classical Realizability.

For more information visit `https://sites.google.com/view/ptcs-palermo/home-page`.

Schedule and Abstracts

July 23, 2024

11:00–11:40 Classical System of Martin-Lof's Inductive Definitions is not Equivalent to Cyclic Proofs

Stefano Berardi (University of Torino, Torino, Italy)

Abstract. This is a joint work with Makoto Tatsuta (National Institute of Informatics, Tokyo, Japan).

A cyclic proof system, called CLKID-omega, gives us another way of representing inductive definitions and efficient proof search. The 2005 paper by Brotherston showed that the provability of CLKID-omega includes the provability of LKID, first order classical logic with inductive definitions in Martin-Löf's style, and conjectured the equivalence. The equivalence has been left an open question since 2011. This paper shows that CLKID-omega and LKID are indeed not equivalent. This paper considers a statement called 2-Hydra in these two systems with the first-order language formed by 0, the successor, the natural number predicate, and a binary predicate symbol used to express 2-Hydra. This paper shows that the 2-Hydra statement is provable in CLKID-omega, but the statement is not provable in LKID, by constructing some Henkin model where the statement is false.

All details about this result can be found in paper with the same title published in Logical Method in Computer Science and available at the address: *https://arxiv.org/abs/1712.09603*. In this abstract we only include the statement of the main theorem.

Theorem 1 (Counterexample to the Brotherston-Simpson Conjecture)**.** *Let H be the2-Hydra formula defined in 1. Then H has a proof in the ciclic system* $\texttt{CLKID}(\Sigma_N, \Phi_N)$*, and no proof in the sub-system* $\texttt{LKID}(\Sigma_N, \Phi_N)$*, having inductive definitions +* $(0, succ)$*-axioms.*

Proof. One possible proof of H in the system CLKID is given in 1. The non-provability of H in $\texttt{LKID}(\Sigma_N, \Phi_N) + (0, s)$-axioms is obtained by showing that the structure $\mathcal{M}$ structure defined in 1 verifies all axioms of the system $\texttt{LKID}(\Sigma_N, \Phi_N)$ with inductive definitions, verifies the $(0, succ)$-axioms, but falsifies H. We conclude that there is a theorem in the system CLKID which is not provable in the sub-system $\texttt{LKID}(\Sigma_N, \Phi_N)+(0, succ)$-axioms. □

References

[1] Stefano Berardi, Makoto Tatsuta, *Classical System of Martin-Lof's Inductive Definitions is not Equivalent to Cyclic Proofs*, Logical Methods in Computer Science, Volume 15, Issue 3 (August 1, 2019) lmcs:4173. https://arxiv.org/abs/1712.09603.

11:40–12:20 What it really needs to be conservative

Peter Schuster (Università di Verona, Italy)

Abstract. This is a joint work with Giulio Fellin (Università di Brescia, Italy) and Sara Negri (Università di Genova, Italy).

Kuroda's double negation shift (DNS) is indispensable for the generalisation to first-order logic of Glivenko's theorem for propositional logic: modulo double negation, classical predicate logic is conservative over intuitionistic logic—with DNS added to the latter in the predicate case. Related conservation theorems have turned out to hinge upon variants of DNS. We try to understand this phenomenon in the borderland of classical and constructive reasoning.

Abstracting Glivenko's theorem from double negation and provability to a nucleus over an inductively generated entailment relation requires that all generating rules of the latter remain admissible in the Kleisli extension, i.e. when the nucleus is applied to the succedent of every premiss and the conclusion. This conservation criterion clearly is automatic for every rule which is right invariant, i.e. all premisses and the conclusion of which have in common the very same succedent. Axioms put as premissless rules are trivially right invariant. In the single-succedent calculi for propositional, infinitary and first-order minimal logic, all rules but the right rules for

implication, infinite conjunction and the universal quantifier can be expressed as right invariant rules. The conservation criteria for the critical rules have proved to be counterparts of DNS.

12:20–13:00 Second order well-behaved

Sara Negri (University of Genoa, Italy)

Abstract. This is a joint work with Matteo Tesi (Technical University of Vienna, Austria).

Second-order logic extends the expressive power of first-order logic, allowing for the representation of properties that involve quantification over all subsets or families of subsets within a given structure, addressing a fundamental need in mathematical discourse. However, while offering significant advantages, full second-order logic faces challenges due to its impredicative nature and the absence of crucial metalogical properties, which complicates the development of proof systems. Our research tackles these challenges by introducing G3-style sequent calculi that incorporate a predicative comprehension schema, facilitating constructive cut-elimination proofs.

Expanding upon these calculi, we delve into the proof theory of mathematical theories, adapting methods from first-order calculi and establishing structural results for both classical and intuitionistic versions of the calculi. Additionally, we define extensional equality and apartness within second-order logic, demonstrating the ability to reduce mathematical concepts to pure logical terms. To illustrate, we present the theories of von Neumann–Gödel–Bernays set theory and predicative second-order arithmetic. Moreover, we establish a variant of Herbrand's theorem tailored for predicative second-order intuitionistic logic, demonstrating the conservativity of predicative second-order Heyting arithmetic over its first-order counterpart. Furthermore, we extend the interpolation theorem and modal embedding of intuitionistic logic to predicative second-order logic.

14:30–15:10 Adventures in Subexponential Non-Associative Non-Commutative Linear Logic

Andre Scedrov (University of Pennsylvania, USA)

Abstract. Adding subexponentials to linear logic enhances its power as a logical framework, which has been extensively used in the specification of, e.g., proof systems and programming languages. Initially, subexponentials allowed for classical, linear, affine or relevant behaviors. Recently, this framework was enhanced so to allow for commutativity as well. In [1], we close the cycle by considering associativity. We formulate the resulting intuitionistic, two-sided system and show that it admits the (multi)cut. In the talk we discuss two new undecidability results that strengthen the undecidability results for fragments/variations of the system, given in [1]. If time permits we also discuss a classical, one-sided multi-succedent classical analogue of our intuitionistic system, presented in [2], following the exponential-free calculi of Buszkowski, and of de Groote and Lamarche. As in linear logic, a large fragment of our intuitionistic calculus is shown to embed conservatively into the classical version. It should be noted that such conservativity results are quite unusual, as they do not hold for traditional, richer logics which enjoy more structural rules for arbitrary formulae. This is joint work with Eben Blaisdell, Max Kanovich, Stepan L. Kuznetsov, and Elaine Pimentel.

References

[1] Blaisdell, E., Kanovich, M., Kuznetsov, S.L., Pimentel, E., Scedrov, A. *Non-associative, Non-commutative Multi-modal Linear Logic*, In: Blanchette, J., Kovács, L., Pattinson, D. (eds), Automated Reasoning. IJCAR 2022. Lecture Notes in Computer Science, vol. 13385, pp. 449-467, Springer, Cham. First Online: 01 August 2022. https://doi.org/10.1007/978-3-031-10769-6-27.

[2] Blaisdell, E., Kanovich, M., Kuznetsov, S.L., Pimentel, E., Scedrov, A. *Explorations in Subexponential Non-associative Non-commutative Linear Logic*, In: M.Moortgat and M. Sadrzadeh, eds., Proceedings Modalities in substructural logics: Applications at the interfaces of logic, language and computation. Electronic Proceedings in Theoretical Computer Science EPTCS 381, pp. 4 - 19. Published: 7th August 2023. https://doi.org/10.4204/EPTCS.381.3.

15:10–15:50 Tropical Mathematics and the Lambda-Calculus
Paolo Pistone (Université Lyon 1/ENS Lyon, France)

Abstract. This is a joint work with Davide Barbarossa (University of Bath, United Kingdom).

Tropical geometry has evolved in the last decades into a vast and rich research domain. Tropical methods, based on the tropical semiring $[0, +\infty]$ endowed with the min and + operations, provide a combinatorial counterpart to several algebraic geometry concepts, with important connections with optimisation theory and well-developed applications in convex analysis and machine learning (see [4] for a recent survey). Computationally speaking, working with min and + is generally easier than working with standard addition and multiplication; for instance, the fundamental (and generally intractable) problem of finding the roots of a polynomial admits a *linear time* algorithm in the tropical case.

In this presentation we introduce an interpretation of the λ-calculus based on a variant of the *relational model* of linear logic [3] based on the tropical semiring, and we discuss a few potential applications of tropical methods for the study of non-deterministic and probabilistic programs.

The tropical relational model provides a unifying framework for two well-developed quantitative approaches to program semantics, notably *program metrics*, based on the analysis of program sensitivity via Lipschitz conditions, and *resource analysis*, based on linear logic and higher-order program differentiation. Indeed, in this semantics programs are interpreted as *tropical power series* (tps in short) of the form $f(x)_b = \inf_{n\in\mathbb{N}}\{nx + c_n\}$, where $c_n \in [0, \infty]$. Tps yield at the same time a *Lipschitz approximation* of the program, that is, an approximation by means of *more and more sensitive* linear programs, and a tropical analogue of the usual *Taylor expansion* of λ-terms [1], that is, the decomposition of the program via resource-bounded approximants.

Crucially, in many important situations, tps can be shown equivalent to tropical *polynomials*: this means that the *infinitary* inf above collapses onto a min of *finitely* many elements. This collapse leads then to a purely combinatorial interpretation of probabilistic and non-deterministic higher-order programs, and well-developed tools (e.g. tropical roots, Newton polygons) can be used to develop a *best case analysis* of program behavior.

Finally, we show that this correspondence between metric and differential analysis of higher-order programs can be put at a higher level of generality through an abstract correspondence between tropical algebra and Lawvere's theory of *generalized metric spaces* [5,2]. This generalization suggests then ways to investigate other computational concepts (like e.g. differential privacy) via tropical methods.

References

[1] T. Ehrhard and L. Regnier. *The differential lambda-calculus.* Theoretical Computer Science, 309:1–41, 2003.

[2] S. Fuji. *Enriched categories and tropical mathematics*, https://arxiv.org/abs/1909.07620, 2019.

[3] J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. *Weighted relational models of typed lambda-calculi*, Proceedings LICS 2013, pages 301–310. IEEE Computer Society, 2013.

[4] P. Maragos, V. Charisopoulos, and E. Theodosis. *Tropical geometry and machine learning*, Proceedings of the IEEE, 109(5):728–755, 2021.

[5] I. Stubbe. *An introduction to quantaloid-enriched categories*, Fuzzy Sets and Systems, 256:95–116, 2014.

15:50–16:30 Some remarks on the PAM and its extensions
Stefano Guerrini (Université Sorbonne Paris Nord, France)

Abstract. This is a joint work with Stefano Catozi (Université Sorbonne Paris Nord, France).

The Pointer Abstract Machine (PAM) is an abstract machine implementing the linear head reduction of λ-calculus. Unlike other abstract machines such as the Krivine Abstract Machine (KAM) [5], which explicitly construct environments for evaluating λ-terms, the PAM employs a pointer mechanism to connect the substitution steps applied along the reduction. The PAM has been introduced in an unpublished note by Danos and Regnier [3], and it has successively been studied in relation to game semantics by Danos, Herbelin, and Regnier [2]. A detailed account

of it, with an analysis of its efficiency has been recently published by Accattoli, Dal Lago, and Vanoni in [1].

Despite its significance, the PAM's description is often perceived as complex. This complexity arises partly because it is seen as an ad-hoc technique for the implementation of λ-calculus β-rule unrelated with the more standard techniques used in writing compilers or interpreters. Roughly speaking, the pointers at the core of the machine are seen as some smart but obscure mechanism peculiar to the PAM and to the particular reduction strategy that it implements, whose deeper meaning can be understood primarily within game semantics frameworks.

This talk aims to demystify the PAM by revealing the connection to a well-established concept in programming language compilation: the so-called *static chain* linking the activation record instances in the execution stack. The only peculiarity of the PAM is that the activation records in its stack do not contain the value of any variable: the association arguments/variables, the so-called execution environment, can be instead recovered by traversing the history of the linear head substitutions memorized in the stack; substitutions that in the PAM play the role of function calls, and whose sequence corresponds then to the so-called *dynamic chain*. Remarkably, the purely syntactical notion of nesting behind the construction of the static chain corresponds to the nesting of boxes in the so-called call-by-name translation of λ-calculus into linear logic [4] based on the $!D \multimap D \simeq D$ isomorphism.

As already mentioned, in the PAM the environments for evaluating the λ-terms are not explicitly constructed; instead, they are inferred by following the dynamic chain. The talk will explore how this process can be optimized compared to the original method proposed by Danos and Regnier. Furthermore, it will suggest an extension of the PAM to handle a calculus with continuations, in the style of the $\lambda\mu$-calculus. Finally, an extension of the machine that reduces not just to head normal forms but also to normal forms will be presented.

References

[1] B. Accattoli, U. Dal Lago, and G. Vanoni. The (in)efficiency of interaction. *Proc. ACM Program. Lang.*, 5(POPL), jan 2021.

[2] V. Danos, H. Herbelin, and L. Regnier. Games semantics and abstract machines. In *Proceedings of the eleventh annual symposium on Logic in Computer Science (LICS 96)*, pages 394–405, New Brunswick, July 1996. IEEE, IEEE Press.

[3] V. Danos and L. Regnier. Head linear reduction. http://iml.univ-mrs.fr/ regnier/articles/pam.ps.gz, 2004.

[4] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[5] J.-L. Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.

17:00–17:40 What is an algorithm?
Thomas Seiller (CNRS, France)

Abstract. This project has received financial support from the CNRS through the MITI interdisciplinary exploratory research program, as well as the ANR-22-CE48-0003 project DySCo.

What is a model of computation? What is a program? What is an algorithm?
While theoretical computer science has been founded on computability theory, the latter does not answer these questions. Indeed, it is a mathematical theory of computable functions, and does not account for computation itself: it only focusses on what is computed, and not on how it is computed.

A symptomatic consequence is the notion of Turing-completeness. This standard equivalence between models of computation is purely extensional: it does only care about what is computed and not how, blind to complexity aspects and the question of algorithmic completeness. One may view the multiple *barriers* in computational complexity (negative results showing the limitations of current methods) as a consequence of this failure. More importantly, the theory of computation is growing ever more further from actual computer science practice.

I will present a proposal [1] for alternative foundations more faithful to computer science practice and interests. This mathematisation of computer science is grounded within the theory

of dynamical systems. I will argue that it generalises computability theory while still allowing to recover standard results. This approach underlies a number of results: shedding new light and unifying several known (and new) lower bounds results in algebraic complexity, conceiving static analysis methods which can be implemented to analyse real programming languages, providing a formal definition of the notion of algorithm, and leading to new models of (linear) logic.

Mainly, I define an *abstract model of computation* as follows.

Definition 1. An *abstract model of computation* (AMC) is a monoid action $\alpha : \mathrm{Mon}(I) \curvearrowright \mathbf{X}$:

(1) $\mathbf{X}$ is a space of configurations, together with a notion of morphisms;
(2) I is a set of instructions, generating the monoid action α;

As examples, one can define abstract models of computation accounting for: Turing machines and automata, algebraic models of computation (e.g. Blum Shub Smale models, iterated multiplication, quantum circuits), models based on rewriting (including lambda-calculus), and Instruction Set Architectures (e.g. RISC-V).

Such a monoid action then induces notions of *machines* and *programs*, defined as generalisations of the notion of graphing used in ergodic theory. Intuitively, a program is a graph whose edges represent elements of the monoid. Models of computation can furthermore be endowed with cost models, providing notions of complexity of programs (generalising space and time complexity).

However, in this talk I will focus on one other aspect of this theory. Indeed, as a last step in abstraction, I propose a formal definition of the notion of *algorithm*. An algorithm is defined a labelled graph endowed with an *abstract data structure*: labels then represent specific operations described by the data structure. A program P then implements the algorithm A if P can be defined as a glueing of programs $P_1, \dots, P_k$ along A, where $P_1, \dots, P_k$ are programs interpreting the data structure maps corresponding to the labels. An interesting outcome of the appraoch is that it could be used to define notions of distance between algorithms, or between an algorithm and a program. This proposal differs from those of Y. Gurevich (based on abstract state machines) and Y. Moschovakis (based on recursors), and I will comment on the differences.

References

[1] T. Seiller. Mathematical Informatics: Outline of a mathematical theory of computer science. Habilitation thesis, Université Sorbonne Paris Nord, 2024.

July 24, 2024

11:30–12:10 Proofs and algorithms: some philosophical considerations

Alberto Naibo (University Paris 1 Panthéon-Sorbonne)

Abstract. Since antiquity, the notion of proof has been seen as tightly connected, in mathematical practice, to the notions of construction and algorithm (think for instance of Euclid's way of proving theorems which is based on the possibility of executing certain constructions, and these constructions can be described as algorithmic procedures). Such a view has also been at the core of a certain logic tradition in the 20th century. In particular, according to the Brouwer-Heyting-Kolmogorov (BHK) interpretation of logical connectives, the meaning of a (logical complex) proposition A has to explained in terms of what counts as a proof of it. But, according to such interpretation, a proposition A also expresses the expectation of the solution of a problem, so that the explanation of A is eventually given in terms of what counts as a construction for solving the problem expressed by A. Now, a formal and precise version of the BHK interpretation can be given by appealing to the so-called Curry-Howard correspondance. This is a formal result allowing one to establish a correspondance between proofs (written in some formal deductive system) and programs (written in some abstract programming language), so that the proof of a proposition A corresponds to a (verified) program of type A. Programs thus play the role of the formal correlates of constructions, and additionally they offer a connection with the notion of algorithm: a program can be seen indeed as the implementation of an algorithm.

In this talk, I will try to investigate in some details the connection between proofs and algorithms at the light of the Curry-Howard correspondance. In particular, as the Curry-Howard correspondance provides a formal framework for establishing identity criteria for proofs/programs, I will try to understand whether such a framework can also be used for establishing identity criteria for algorithms. More generally, the aim of the talk is to examine whether it is possible to have a formal definition of algorithm not being *ad hoc* one, but keeping instead a direct connection with the notion of proof. It is in this sense that I will draw a comparison between the positions of A. Kolmogorov & V. Uspensky [3], Y. Gurevich [2] and Y. Moschovakis [4] concerning the possibility of a formal definition of algorithm and the position of J.-Y. Girard, whose project of the *Geometry of Interaction* was not only a continuation of the Curry-Howard correspondance, but also a "general program of mathematisation of algorithmics" ([1, p. 76]).

References

[1] J.-Y. Girard, *Geometry of interaction II: deadlock-free algorithms.* In P. Martin-Löf and G. Mints (eds), COLOG-88, pp. 76–93. Springer, 1990.

[2] Y. Gurevich, *Sequential abstract-state machines capture sequential algorithms*, ACM Transactions on Computational Logic, 1 (2000), 77–111.

[3] A.N. Kolmogorov & V. Uspensky, *On the definition of an algorithm*, ASM Translations, 29 (1963), 217–245.

[4] Y.N. Moschovakis, *On founding the theory of algorithms.* In H.G. Dales and G. Oliveri (eds), *Truth in Mathematics*, pp. 71–104. Oxford University Press, 1998.

12:10–12:50 On semantic perspectives for linear logic
Elaine Pimentel (University College London, United Kingdom)

Abstract. In this talk, we will view linear logic through two different semantical lenses: dialogical games and proof-theoretic semantics.

We first look at substructural calculi from a game semantic point of view, guided by certain intuitions about resource conscious and, more specifically, cost conscious reasoning. To this aim, we start with a game, where player I defends a claim corresponding to a (single-conclusion) sequent, while player II tries to refute that claim. Branching rules for additive connectives are modeled by choices of II, while branching for multiplicative connectives leads to splitting the game into parallel subgames, all of which have to be won by player I to succeed. The game comes into full swing by adding cost labels to assumptions, and a corresponding budget. Different proofs of the same end-sequent are interpreted as more or less expensive strategies for I to defend the corresponding claim. This leads to a labelled calculus, which can be seen as a fragment of SELL (subexponential linear logic). Finally, we generalize the concept of costs in proofs by using a semiring structure, illustrate our interpretation by examples and investigate some proof-theoretical properties.

The second look will view substructural behaviors using proof-theoretic semantics (PTS) lenses. PTS aims not only to elucidate the meaning of a logical proof, but also to provide means for its use as a basic concept of semantic analysis. In this talk, we will propose a "inferences-as-resources" approach to intuitionistic multiplicative linear logic (IMLL), and show how this can be smoothly extended to the full intuitionistic linear logic.

This is a joint work with Timo Lang, Carlos Olarte, Christian G. Fermüller and Victor Nascimento.

References

[1] T. Lang, C. Olarte, E. Pimentel, C. G. Fermüller. *A Game Model for Proofs with Costs*, TABLEAUX 2019, 241–258.

[2] V. Barroso-Nascimento, E. Pimentel. *Linear proof-theoretical semantics*, draft, 2024.

14:30–15:10 Applying Implicit Computational Complexity
Clément Aubert(Augusta University, USA)

Abstract. This abstract benefited from discussions with Neea Rusch, and reflect on work conducted additionally with Thomas Rubiano and Thomas Seiller [3, 4, 5].

This presentation is interested in reasoning about the recent development of static analyzers and optimizers inspired by Implicit Computational Complexity (ICC), which illustrate the challenges and benefits of applying approaches stemming from proof theory to software development.

ICC aims at characterizing complexity classes without referring to models of computations such as Turing machines. Doing so generally requires to define non-Turing-complete programming languages using tools coming from proof theory, model theory or recursion theory, and then to prove that they are *sound* and *complete* w.r.t. a particular complexity class $\mathbb{C}$ (i.e., that their programs compute functions in $\mathbb{C}$, and that for any problem in $\mathbb{C}$, there is a program that solves it). This approach allows to sidesteps Rice's theorem, but cannot be *intentionally* complete, i.e., it will always exclude programs that yet belong to the targeted complexity class.

ICC slightly wandered off this original program to get applied in three fairly different areas:

- A flow-analysis "extend[ing] and refin[ing] work in the Implicit Complexity research community" [1] was leveraged to develop two optimization passes [2, 3], hence facilitating (automatic) optimization of programs.
- Inspired respectively by the flow-analysis previously mentioned and tiering techniques, pymwp [4, 5] and ComplexityParser [6] provide complexity analysis that can help with the development of software products.
- Combining a noninterference-based type system with a stratification (or tiering) discipline was used to provide two types of correctness, related to resource usage and security [7].

Those research lines illustrates a shift from the design of ICC programming languages ("pre-bounding" all programs) to the design of *criteria* that may or may not be met by programs written in a Turing-complete programming language. Adapting the theory to make it implementable [4], finding reasonable metrics to compare to pre-existing tools [3, Section 5] or obtaining termination certificate and tight bounds [6, Section 3] are some of the challenges offered by this shift.

This new direction also allows to materialize some of the theoretical developments of ICC, to illustrate how this approach can sidestep some of the difficulties met by e.g., abstract interpretation-based static analyzers, and to demonstrate that ICC can be applied to semantic properties not related to complexity. It is, however, our hope that the design *and implementation* of complexity-bounded programming languages can emerge from ICC research, as those would offer the considerable advantage of *not* having to run any static analyzer to obtain strong and diverse semantic properties.

References

[1] N. Jones, L. Kristiansen: *A flow calculus of* mwp*-bounds for complexity analysis. ACM Trans. Comput. Log.*, 2009.

[2] J.-Y. Moyen, T. Rubiano, T. Seiller: Loop Quasi-Invariant Chunk Detection. ATVA 2017

[3] C. Aubert, T. Rubiano, N. Rusch, T. Seiller: Distributing and Parallelizing Non-canonical Loops. VMCAI 2023

[4] C. Aubert, T. Rubiano, N. Rusch, T. Seiller: mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity. FSCD 2022

[5] C. Aubert, T. Rubiano, N. Rusch, T. Seiller: pymwp: A Static Analyzer Determining Polynomial Growth Bounds. ATVA 2023

[6] E. Hainry, E. Jeandel, R. Péchoux, O. Zeyen: ComplexityParser: An Automatic Tool for Certifying Poly-Time Complexity of Java Programs. ICTAC 2021

[7] E. Hainry, R. Péchoux: A General Noninterference Policy for Polynomial Time. POPL 2023

15:10–15:50 A Kleene algebra with tests for union bound reasoning about probabilistic programs

Patrick Baillot (CNRS, Université de Lille, France)

Abstract. This is a joint work with Leandro Gomes (Université de Lille) and Marco Gaboardi (Boston University, Boston, USA).

Kleene Algebra with Tests (KAT) [1] provides a framework for algebraic equational reasoning about imperative programs. The recent variant Guarded KAT (GKAT) [4] allows to reason on non-probabilistic properties of probabilistic programs. In this talk we introduce an extension of this framework called approximate GKAT (aGKAT) [3], a variant of GKAT enriched with a partially ordered monoid (real numbers) which enables to express satisfaction of (deterministic) properties *except* with a probability up to a certain bound. This allows to represent in equational reasoning 'à la KAT' proofs of probabilistic programs based on the union bound, a technique from basic probability theory. We show how a propositional variant of approximate Hoare Logic (aHL) [2], a program logic for union bound, can be soundly encoded in our system aGKAT.

References

[1] Dexter Kozen. Kleene algebra with tests. ACM Trans. on Prog. Lang. and Systems, 19(3):427–443, 484 1997. doi:10.1145/256167.256195.

[2] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. A program logic for union bounds. In Proceedings of ICALP 2016, volume 55 of LIPIcs, pages 107:1–107:15. Schloss Dagstuhl - Leibniz-Zentrum fur Informatik, 2016. doi: 10.4230/LIPIcs.ICALP.2016.107.

[3] Leandro Gomes, Patrick Baillot, Marco Gaboardi. A Kleene algebra with tests for union bound reasoning on probabilistic programs. Technical report. 2023. https://hal.science/hal-04196675

[4] Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. 516 Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear 517 time. Proc. ACM Program. Lang., 4(POPL):61:1–61:28, 2020. doi:10.1145/3371129.

15:50–16:30 Towards an induction principle for nested data types
Peter Selinger (Dalhousie University, Canada)

Abstract. This is a joint work with Peng Fu (University of South Carolina, USA).

A well-known problem in the theory of dependent types is how to handle so-called *nested data types*. These data types are difficult to program and to reason about in total dependently typed languages such as Agda and Coq. In particular, it is not easy to derive a canonical induction principle for such types. Working towards a solution to this problem, we introduce *dependently typed folds* for nested data types. Using the nested data type `Bush` as a guiding example, we show how to derive its dependently typed fold and induction principle. We also discuss the relationship between dependently typed folds and the more traditional higher-order folds.

E–mail: `tortora@uniroma3.it`.