Titles and Abstracts

Workshop "Directions and Perspectives in the λ-calculus"

08/01/2024, Bologna (Italy)

- *Beniamino Accattoli* (Partout Team, Inria and LIX École Polytechnique)

  It's only lambda calculus (but I like it)

  In this talk, I'll present my view of the lambda calculus as a research topic, discussing historical aspects, what I see as the main research areas, what I think are necessary steps towards a solid and modern theory, how my research fits into the picture, and the directions that I would like to develop in the near future.

- *Elena di Lavore* (Compositional Systems and Methods group, Tallinn University of Technology)

  Effectful transition systems

  Transition systems appear across the computer science literature in different flavours: deterministic, non deterministic or stochastic; and each of these flavours has its own definition of transition, bisimulation and trace equivalence. We propose a definition of transition system in effectful categories. These are premonoidal categories, where morphisms represent effectful computations, with a chosen monoidal centre that selects the pure computations. We define bisimulation and trace equivalence for effectful transition systems and show that they capture the classical and probabilistic notions. We show that bisimulation implies trace equivalence in this general setting.

- *Zeinab Galal* (Dipartimento di Informatica, Università di Bologna)

  Bidimensional fixpoint operators

  Fixpoint operators are tools to reason on recursive programs and data types obtained by induction (e.g. lists, trees) or coinduction (e.g. streams). They were given a categorical treatment with the notion of categories with fixpoints. I will present a categorification of this notion to a 2-dimensional setting where the 2-morphisms allow us to model the execution steps for languages with (co)inductive principles. We recover standard categorical constructions of initial algebras and final coalgebras for endofunctors as well as fixpoints of generalized species and polynomial functors. I will then present ongoing work using this formalism in order to develop the theory of traced monoidal bicategories. In the last part of the talk, I will present joint work in progress with JS Lemay on the interaction of fixpoint theory and differentiation.

- *Francesco Gavazzo* (Dipartimento di Informatica, Università di Pisa)

  Relational Mechanics of the λ-Calculus

  In this talk, I will give a high-level introduction to program relation algebras — a novel family of relation algebras I have recently introduced stemming from the work by Gordon, Lassen, and Pitts on algebras of term relations — by showing some of their applications to the operational theory of the λ-calculus. In particular, I will outline how such algebras allow us to give a unified and syntax-independent operational theory of the lambda-calculus (and related formalisms) encompassing rewriting, dynamic semantics, and program equivalence and approximation.

- *Giulio Manzonetto* (Institut de Recherche en Informatique Fondamentale, Université Paris Cité)

  The lambda-calculus yesterday, today, and tomorrow

  In this talk, I will argue why in my opinion studying the lambda-calculus per se is still meaningful. On the one hand many interesting problems are still open, on the other one the lambda-calculus is still influencing nowadays advances in programming language theory and mathematical logic. In particular, I will discuss the role of denotational semantics in the past, the present, and the future of the discipline.

- *Christina Matache* (Laboratory for Foundations of Computer Science, University of Edinburgh)

  Parameterized algebraic theories and scoped effects

  Algebraic theories provide a way of axiomatizing effects using operations and equations, where operations are basic programming features like reading and updating the state, and equations specify observably equivalent programs. The monads arising from algebraic theories provide a way to model lambda calculus extended with algebraic effects. Effect handlers are a programming construct for implementing algebraic operations and for modularly programming with them.

  I will recall parameterized algebraic theories, a generalization of algebraic theories, and scoped effects, an extension of effect handlers. For example, exception catching can be implemented as a scoped effect, but not an algebraic one. I will argue by example that the framework of parameterized algebraic theories can be used to give equational axiomatizations to scoped effects, and to characterize some existing models of scoped effects.

- *Egbert Rijke* (Faculty of Mathematics and Physics, University of Ljubljana)

  TBA

- *Philip Saville* (Department of Computer Science, University of Oxford)

  Refined syntax and semantics via taking contexts seriously

  Semanticists are increasingly using sophisticated mathematical tools to refine traditional models of lambda calculi. Examples include using higher category theory to capture rewriting or intensional information about derivations, or using enriched category theory to study 'distances' between

terms. These developments are paralleled by new finer-grained calculi reflecting the extra information available in such models. In these situations the proofs for soundness and completeness, or even the question of how to define the syntax in a canonical way, can quickly become far subtler than in the traditional setting. In this talk I aim to show these difficulties are alleviated by embracing Lambek's old idea of using multi-ary structures, in which maps can have multiple inputs. I will argue that the multi-ary perspective clarifies the semantic interpretation of simply-typed lambda calculi, and outline how the same ideas can be used to give both a canonical syntax and clean soundness and completeness proofs for extensions of the lambda calculus, such as with rewriting or effects. Finally I will try to sketch some ideas for other kinds of refined semantics or semantics that might be possible with these methods.

- *Dima Szamozvancev* (Department of Computer Science and Technology, University of Cambridge)

  The trials and tribulations of language formalisation

  Computer formalisation makes enticing promises to type theorists and programming language researchers: easing the burden hand-writing long, tedious, inductive proofs, and letting them focus on the interesting, non-trivial metatheoretic questions surrounding a novel technique or language construct. The harsh reality, however, is that a significant new burden is raised by something that pen-and-paper developments have the liberty of glossing over: binding operators and the treatment of variables. More than just an implementation detail to ignore in order to focus on the big picture, it fundamentally affects the choice of syntax encoding, representation of substitution, equational theory and operational or denotational semantics.

  The λ-calculus is the perfect vehicle for experimenting with formalisation techniques, as even in its minimal form it contains enough constructs to turn formalisation into an exercise in level-headedness. Developing a comprehensive metatheory is a painstakingly stop-and-go process, and the resulting code primarily consists of boilerplate to deal with substitution, rather than anything inherently calculus-specific. Motivated by these frustrations, the talk will outline a language-formalisation framework that addresses the problems in a methodical, mathematically justified way, giving users the tools to focus on the aspects of a calculus they are actually interested in.