TYPES 2010: Abstracts

Warsaw, October 13-16, 2010

INVITED TALKS

Henk Barendregt,
Lambda calculus with types

Coming down from the complexity of the calculus of constructions, the simply typed lambda calculus seems trivial. For example only strongly normalizing terms are typable. In spite of this, there are non-trivial results, like the existence of the five canonical term models. The simple types can be considered modulo some equality between types. Then one can type more terms and other results become valid. Changing the equalities into inequalities again increases the possibility of typing a term. Finally by postulating that every finite set of types has an intersection, all untyped lambda terms can be typed and one obtains filter model versions of classical models. In this world of terms and layers of types we will formulate some open problems.

Yves Bertot,
Formal verification: from computational geometry to real algebraic geometry

Computational geometry is a branch of computer science that deals with large collections of objects (often points) in spaces that are usually affine or euclidean. The algorithms it entails can be very clever and the properties of the objects manipulated can be expressed concisely in the mathematical jargon. Computational geometry also plays a role in many systems where software interacts with the real world. For this reason, it seems to be a good a playground for formal verification. Computational geometry relies on elementary geometry, which in turn can be treated systematically with algebraic methods, like Gröbner bases or cylindrical algebraic decomposition. Several teams have addressed this topic in Edinburgh, Strasbourg, or Sophia Antipolis. However, I will concentrate on the experiments in which I took part, like the formalization of convex hull algorithms, Delaunay triangulation, and the isolation of roots for one-variable polynomials, a basic block to cylindrical algebraic decomposition.

Pierre-Louis Curien,
The duality of computation under focus

(joint work with Guillaume Munch)
We review the relationship between abstract machines for (call-by-name or call-by-value) lambda-calculi and sequent calculus, reintroducing on the way Curien-Herbelin's syntactic kit of the duality of computation. We provide a term language for a presentation of classical sequent calculus. A key slogan is that commutative cut elimination rules are transcribed as explicit substitution propagation rules. We then describe the focalised proof search discipline (in the classical setting), and narrow down our language and the rewriting rules to a confluent calculus. Our analysis highlights a one-to-one correspondence between values and focalisation phases (an observation that was only implicit in Curien-Herbelin's work one decade ago). We then define a game of patterns and counterpatterns, leading us to a fully focalised finitary syntax for a synthetic (and complete) presentation of classical logic, that provides a quotient on (focalised) proofs, abstracting out the order of decomposition of negative connectives, and establishing links with Girard's ludics.

Aarne Ranta,
Machine translation and type theory

Recent advances in systems like Google translate (`translate.google.com`) have revived the interest for machine translation. This system is able to translate any document to and from any

of over 50 languages. There are, however, quality problems, which seem impossible to overcome with the current techniques, statistical in nature. Statistical machine translation is based on ideas originally developed for cryptography using Shannon's notion of a noisy channel. This idea is of course very different from the translation technique used in compilers: converting source code into target code by meaning-preserving transformations via an abstract syntax. The compiler idea is dual to the statistical methods in the sense that it is able to produce high-quality translations, but its scope is limited to fully formalized languages.
Inspired by type theory and logical frameworks, the grammatical framework GF applies a compiler-like translation technique to natural languages. The talk aims to show the strengths and weaknesses of the GF method, and also to identify ways in which statistical techniques can be used for improving the productivity and robustness of GF translation. Some of this work is being carried out in the ongoing FP7 project MOLTO (Multilingual On-Line Translation, www.molto-project.eu).

CONTRIBUTED TALKS

Robin Adams,
Proof reuse in logical frameworks

Suppose we have two systems, say a classical system S and a constructive system T, both declared in the same logical framework. We may have a metatheorem of the form: if A is a theorem of S, then the double negation translation $DN(A)$ is a theorem of T. Can we use such metatheorems in practice?
I shall present a method for taking a proof script that proves A in S, and using that script - immediately, without modification - as a proof of $DN(A)$ in T. The method works by taking each constant in the declaration of S, and defining that constant in T.
This method is very adaptable, and can extend to the double negation translation, the A-translation and the Russell-Prawitz modality, among others. I shall demonstrate its use in practice using the proof assistant Plastic.

Thorsten Altenkirch,
Mixed induction/coinduction

Instead of classifying datatypes into inductive and coinductive definitions, we introduce a type-former ∞ which permits the construction of infinite objects as unevaluated computations. This leads to a more flexible approach allowing datatypes defined via mixed induction/coinduction which turns out to be an extremely useful tool. Using the Agda system, which implements this approach, we look at a number of examples such as stream processors and weak bisimulation.
We also note that ∞ can be used to define recursive datatypes as infinite objects under the additional assumption that datatype constructors preserve guardedness in strictly positive positive definitions. This view unifies inductive/coinductive and recursive definitions and hence reduces induction-recursion to mutual recursion. We discuss termination checking for mixed inductive/coinductive definitions and draw attention to an issue with nested definitions in the current approach.
This talk is based on joint work with Nils Anders Danielsson.

Thomas Braibant,
Rewriting modulo associativity and commutativity in Coq

(Joint work with Damien Pous) We present a Coq plugin that provides tactics for rewriting universally quantified equations, modulo associative (and possibly commutative) operators. We will hint at why these tactics are useful in common proofs, and give an overview of some interesting points of its implementation. We will also sketch how it was improved in the last few months, in particular with respects to AC or A operators with units.

Cyril Cohen,
Formalized foundations of polynomial real analysis

When reasoning formally with polynomials over real numbers, or more generally real closed fields, we need to be able to manipulate easily statements featuring an order relation, either in their

conditions or in their conclusion. For instance, we need to state the intermediate value theorem and the mean value theorem and we need tools to ease both their proof and their further use. For that purpose we propose a Coq library for ordered integral domains and ordered fields with decidable comparison. In this talk we present the design choices of this libraries, and show how it has been used as a basis for developing a fare amount of basic real algebraic geometry.

Hans de Nivelle,
A type system for classical logic with partial functions

We give a natural semantics for classical logic with partial functions. The semantics is based on multivalued logic, so that formulas involving undefined values can have undefined truth values. An unusual aspect of our semantics is that it relies on the order of formulas. In this way, the semantics is able to capture the fact that functions must be declared before they are used.
We will introduce a proper notion of sequent and a corresponding calculus. We prove the calculus sound and complete with respect to the semantics given before.
We think that our approach to partial functions is more natural than existing approaches, because in our approach, formulas involving undefined values are guaranteed to be undefined. This captures the natural behavior of programs containing undefined values. The outcome of such programs should be undefined, even if an outcome could be deduced in principle.

Alejandro Díaz-Caro,
Sums in algebraic lambda-calculi

This is a joint work with Barbara Petit.
In this work we define the confluent additive fragment of the linear-algebraic lambda-calculus. We also define a fine-grained type system which includes sums of types as a reflect of those in the terms. After proving subject reduction and strong normalisation, we study the role of sums within the calculus by interpreting our system into System F with pairs. We shows that this calculus can be interpreted as System F with an associative and commutative pair constructor, which is distributive under application.

Alejandro Díaz-Caro,
A vectorial type system (Work-in-progress)

Joint work with Pablo Arrighi and Benoît Valiron
The linear-algebraic λ-calculus extends the λ-calculus with the possibility of making arbitrary linear combinations of λ-calculus terms. The aim is to combine the Scalar type system with the Additive type system, to provide a fine-grained, System F-like type system for the linear-algebraic λ-calculus that gives rise to an original and general type theory where types, in the same way as terms, have a vector space-like structure. We mention the potential connections with quantum computing.

Silvia Ghilezan,
Resource control in sequent lambda calculus

We present a simply typed sequent lambda calculus with resource control which is in the Curry-Howard correspondence with intuitionistic sequent logic with explicit structural rules of weakening and contraction. For the proposed calculus we prove some operational properties, including subject reduction and strong normalisation. We then relate the proposed calculus to the simply typed intuitionistic calculus of Kesner and Lengrand, which handles explicit operators of weakening and contraction in the natural deduction format.
Joint work with Pierre Lescanne, Jelena Ivetić and Dragiša Žunić.

Stéphane Glondu,
Proving Coq extraction in Coq

The extraction in Coq consists in removing some parts of Coq terms that are irrelevant for computation and turn the remaining parts into programs in a practical programming language such as (purely functional) OCaml. Its principle and several aspects of its current implementation have been described and studied by Pierre Letouzey in his PhD thesis. There is still a gap between the

paper proofs and the implementation, though. How can this gap be reduced? I will present work in progress in this direction.

Katarzyna Grygiel,
Asymptotically almost all lambda terms are strongly normalizing

We present quantitative analysis of various (syntactic and behavioral) properties of random lambda terms. Our main results are that asymptotically all the terms are strongly normalizing and that any fixed closed term almost never appears in a random term. Surprisingly, in combinatory logic (the translation of the lambda calculus into combinators), the result is exactly opposite. We show that almost all terms are not strongly normalizing. This is due to the fact that any fixed combinator almost always appears in a random combinator. Joint work with René David, Jakub Kozik, Christophe Raffalli, Guillaume Theyssier, Marek Zaionc.

Florian Hatat,
A graphical game semantics for response categories

We explore a subset of simply-typed lambda-calculus where arrow types are limited to the form $A \to \bot$, for a constant sort $\bot$, i.e., the typical target language for simply-typed CPS translations. We build a graphical game semantics for this calculus, where positions of the game are "partial" trees and moves between them are a kind of graph morphisms. This extends previous works by Delande-Miller, and Hirschowitz to a non-linear setting.
This game is endowed with a notion of topology on positions, which allows us to describe interaction between two compatible strategies as a strategy resulting from their amalgamation in a sheaf. Hiding the interface at which the two strategies interact is described as a descent operation along particular "cut" morphisms.
Amalgamation followed by hiding define an associative and unital operation on strategies, i.e., make strategies into a category, which we identify as an initial response category in the sense of Selinger.

Hugo Herbelin,
A computational analysis of Gödel's completeness theorem

We carry on Krivine's investigations into the computational content of Henkin's proof of Gödel's completeness theorem and arrive to a presentation of the proof close in style to the reify/reflect-based proofs of completeness for those models (e.g. Kripke, pretopology/phase semantics, ...) that carry more structure than the boolean model.
This is joint work with Danko Ilik.

Clement Houtmann,
Superdeduction in $\overline{\lambda}\mu\tilde{\mu}$

Superdeduction is a method specially designed to ease the use of first-order theories in predicate logic. The theory is used to enrich the deduction system with new deduction rules in a systematic, correct and complete way.
A proof-term language and a cut-elimination reduction already exist for superdeduction, both based on Christian Urban's work on classical sequent calculus. However the computational content of Christian Urban's calculus is not directly related to the (lambda-calculus based) Curry-Howard correspondence. In contrast the $\overline{\lambda}\mu\tilde{\mu}$ calculus is a lambda-calculus for classical sequent calculus.
This work is a first step towards a further exploration of the computational content of superdeduction proofs, for we extend the $\overline{\lambda}\mu\tilde{\mu}$ calculus in order to obtain a proofterm langage together with a cut-elimination reduction for superdeduction. We also prove strong normalisation for this extension of the $\overline{\lambda}\mu\tilde{\mu}$ calculus.

Cezar Ionescu,
Proving versus testing in climate impact research

Higher-order properties arise naturally in some areas of climate impact research. For example, "vulnerability measures", crucial in assessing the vulnerability to climate change of various regions and entities, must fulfill certain conditions which are best expressed by quantification over all

increasing functions of an appropriate type. This kind of property is notoriously difficult to test. However, for the measures used in practice, it is quite easy to encode as a dependent type and prove. Moreover, in scientific programming, one is often interested in correctness "up to implication": the program would work as expected, say, if one would use exact computations instead of floating-point values. Such counterfactuals are impossible to test, but again, they can be easily encoded as types and proven. We show examples of such situations (implemented in Agda), encountered in actual vulnerability assessments.

Agnieszka Kozubek,
Inductive types in Less Naive Type Theory

Less Naive Type Theory is a type system where subsets (predicates) are objects and not, as it is usually the case, kinds. It also has dependent types and higher order logic. However, the system does not have type polymorphism as adding it leads to an inconsistent system.
Less Naive Type Theory on its own is quite powerful but it does not allow to introduce datatypes. This is why we extend the system with inductive types and inductive predicates. It increases the complexity but also the expressive power of the system. We are able to create inductive types which are not allowed in standard frameworks. For example a list of subsets is still an object and not a type. We prove strong normalization property of LNTT with inductive types. This result entails the consistency of the system.

Marc Lasson,
Realizability and parametricty in Pure Type Systems

We describe a systematic method to build a logic from any programming language described as a Pure Type System (PTS). The formulas of this logic express properties about programs. We define a parametricity theory about programs and a realizability theory for the logic. The logic is expressive enough to internalize both theories. Thanks to the PTS setting, we abstract most idiosyncrasies specific to particular type theories. This confers generality to the results, and reveals parallels between parametricity and realizability.

Sergueï Lenglet,
Expansion for universal quantifiers

Expansion is a tranformation on typings (i.e. on pairs typing environment - result types) used in intersection types to deduce a typing from an other one (usually from a principal typing). The definition of this operation has been made easier by the introduction of expansion variables by Carlier and Wells in System E. In this talk, we present System Fs, an extension of system F with expansion variables. We explain how we introduce expansion in a type system with universal quantifiers, present the main differences between the expansion in System E and in System Fs, and give the benefits of this operation in terms of type inference

Assia Mahboubi,
Small scale reflection, a guided tour

The Mathematical Component project, at the Inria Microsoft Research Joint Centre, investigates how to design large and consistent bodies of formalized mathematics. The aim of this talk is to review how the Coq system, and in particular both its computational features and its type inference mechanisms proves a precious and efficient tool in solving compositionality and automation general issues raised by such developments. We will finally propose a brief overview of the available libraries, which now cover a rather wide variety of elementary algebraic structures and theories.

Érik Martin-Dorel,
Formalization of Hensel's lemma in Coq

Suppose we want to find the integral simple roots of a univariate polynomial with integer coefficients. We can use Hensel's lifting, which can be viewed as the p-adic variant of the Newton–Raphson's iteration. It enables us to compute the roots modulo increasing powers of a prime p. As soon as the considered power of p becomes greater than a known bound on the roots, we easily obtain the sought integral roots.

The bivariate version of this root-finding algorithm is extensively used by the Stehlé–Lefèvre–Zimmermann algorithm, designed to solve the so-called Table Maker's Dilemma in an exact way. Consequently, the formal verification of Hensel's lemma (i.e., the correctness lemma of Hensel's lifting method) will contribute to the validation of this kind of algorithm.
In this talk, we describe the various steps we met during the formalization of Hensel's lemma with the Coq proof assistant along with the SSReflect extension.

Keiko Nakata,
Walking through infinite trees with mixed induction and coinduction

There are practically useful properties that involve both induction and coinduction. Weak bisimulation is a typical example. Inspired by discussion with T. Altenkirch, I will present one such example which uses nested induction-into-coinduction as well as coinduction-into-induction: we walk though trees with infinitely deep paths. Two predicates on infinite trees are of particular interest. They are mutually exclusive constructively, and together form tautology classically. Moreover they naturally induce datatype counterparts: particular trees that exactly correspond to the two predicates respectively. I will discuss ongoing work on Coq formalization around this conceptually simple but technically interesting example. (Joint work with T. Uustalu)

Fredrik Nordvall Forsberg,
Interpreting inductive-inductive definitions as indexed inductive definitions

Induction is a powerful and important principle of definition, especially so in dependent type theory and constructive mathematics. Induction-induction (named in reference to Dybjer and Setzer's induction-recursion) is an induction principle which gives the possibility to simultaneously introduce a set A together with an A-indexed set B (i.e. for every $a : A$, $B(a)$ is a set). Both A and $B(a)$ are inductively defined, and the constructors for A can also refer to B and vice versa.
In recent work, we have formalised inductive-inductive definitions and thus made meta-mathematical analysis of the theory possible. In this talk, I will sketch the first result of this kind. I will show that the theory of inductive-inductive definitions can be interpreted in the theory of indexed inductive definitions. In particular, this shows that both theories have the same proof-theoretical strength.

Luís Pinto,
Monadic translation of classical sequent calculus with application to strong normalisation

In this talk we consider monadic translations of the call-by-name (cbn) and the call-by-value(cbv) fragments of the classical sequent calculus $\overline{\lambda}\mu\tilde{\mu}$ by Curien and Herbelin.
The target of the monadic translations is a new meta-language for classical logic, named monadic-lambda-mu. It is a monadic reworking of Parigot's lambda-mu-calculus, where the monadic binding is confined to commands, thus integrating the monad with the classical features, and whose intuitionistic restriction is a variant of Moggi's meta-language.
The new monadic translations strictly simulate reduction in monadic-$\lambda\mu$, and the monad can be instantiated to the continuations monad so as to ensure strict simulation of monadic-$\lambda\mu$ inside simply-typed lambda-calculus.
By composition, we obtain continuation-passing-style translations of cbn and cbv $\overline{\lambda}\mu\tilde{\mu}$ into simply-typed lambda-calculus that strictly simulate reductions, and thus reprove in an elementary syntactical way strong normalisation for these fragments of $\overline{\lambda}\mu\tilde{\mu}$.
The results extend to second-order logic, with polymorphic lambda-calculus as target, giving new strong normalisation results, and constitute an improvement on earlier results for the intuitionistic case.
This is ongoing joint work with José Espírito Santo, Ralph Matthes and Koji Nakazawa.

Andrew Polonsky,
Range property for H

We refute Barendregt's conjecture which concerns the lambda-theory equating all unsolvables.

Christophe Raffalli,
Fancy examples of proofs in PML

The PML implementation has been polished a lot last year. It now allows very "nice" proofs and we will present some of them (featuring category, the use of subtyping to formalise results in logic, . . .).
In the above sentence, "nice" means in particular that proofs are relatively robust both to change in PML and in the proof development itself. We will try to explain why.

Sebastian Reichelt,
Treating sets as types in a proof assistant for ordinary mathematics

I present a proof assistant with a novel graphical user interface, which not only renders user input in a familiar mathematical style, but completely abandons textual input in favor of a menu-driven approach. In order to fill each menu precisely with the set of intuitively meaningful choices, the software employs a custom formal system which appears set-theoretic from the user's point of view but is based on types internally.

Wilmer Ricciotti,
A canonical locally named formalization of an algebraic logical framework

Joint talk with Randy Pollack. We present a formalization, in the Matita Interactive Theorem Prover, of Gordon Plotkin's DMBEL logical framework, along with some basic properties of its metatheory. The formalization, which adopts Sato's canonical locally named representation of binding structures, sheds light on issues that must be considered when dealing with advanced operations like multiple binding and hereditary substitution.

Ondrej Rypacek,
Toposes of polynomial coalgebras

It is a straightforward consequence of known results that the category of coalgebras for a dependent polynomial functor (indexed container) is a topos. After demonstrating this fact we discus its applications to the semantics of datatypes with sharing and cycles, and state transition systems on them. This is a report on a work in progress on type-theoretic formalisation of Model Driven Architecture in object-oriented software engineering.

Claudio Sacerdoti Coen,
Certified Complexity

CerCo (Certified Complexity) is a new European Project about formal verification of a complexity preserving compiler for a large subset of C to an 8-bit microprocessor code. We will give a short overview of the project objectives and methodology.

Jeffrey Sarnat,
Towards a sequent calculus formulation of the calculus of inductive constructions

(Joint work with Hugo Herbelin and Vincent Siles)
The Calculus of Inductive Constructions (CIC) is the core type theory implemented by the proof assistant Coq. Although CIC's precise definition has varied over time, in the past it has always been presented in the style of natural deduction. Here, we work towards a sequent calculus presentation of CIC–based on Herbelin's LJT and Cervesato and Pfenning's spine calculus–by starting with a simple type theory with inductive and coinductive types, pattern matching, and guarded least and greatest fixed-points. This effort is motivated by the following concerns:

1. the metatheoretic properties of the full type theory implemented by Coq has never been undertaken, and sequent calculi tend to be better suited for such investigations than natural deduction
2. we are interested in exploring the symmetry (and asymmetry) between the guard conditions for inductive and coinductive expressions, and
3. we believe that the operational behavior of sequent calculi closely mimics that of the abstract machine currently implemented in Coq.

If time permits, we will discuss the extension of this type theory with dependent types, which requires a dependent version of the cut rule.

Stephan Scheele,
Towards a simply typed CALculus for semantic knowledge bases

This work demonstrates how a constructive version of the description logic ALC can serve as a semantic type system for an extension of the simply typed lambda-calculus to express computations in knowledge bases. This cALculus embodies a functional core language which provides static type checking of semantic information processing of data whose structure is organised under a relational data model as used in description logics. The cALculus arises from a natural interpretation of the tableau rules for constructive ALC following the Curry-Howard-Isomorphism.

Adam Slaski,
Some interesting features of the Polish language in GF

Grammatical Framework (GF) is a formalism for describing natural languages. Currently it provides automatic translation between several languages including Polish and English. Structures of those two languages are very different. In this talk it will be explained what is the difference between Polish and English nominal phrase and how they are translated with GF. No knowledge of Polish is required.

Sergei Soloviev,
Some non-standard reductions in UTT: properties and applications.

In the first part of our work we extend the typed operational semantics for Luo's UTT considered by H. Goguen [1] to UTT with a new rule schema representing the non-standard reduction $|f| \circ |g| \to |f \circ g|$ (f and g being set-theoretical functions on the sets of elements of finite types, $\circ$ - composition, $|f|$, $|g|$, $|f \circ g|$ related maps defined using recursors). We show that all steps of SN and CR go through for this extended UTT. In the second part we consider UTT with stronger equality (induced by new reduction) and its extensions with coercive subtyping. We obtain a complete proof of the conservativity theorem for coercive subtyping in this system: coherence of basic coercions (with respect to ordinary or stronger equality induced by new reduction) implies conservativity of the system with subtyping with respect to the system without subtyping. This part uses the approach to coercive subtyping as an abbreviation mechanism developed by Luo and generalizes and completes the techniques of conservativity proof proposed by Soloviev and Luo in [2]. (The proof in [2] was incomplete.) The non-standard reduction we considered permits to include in intensional framework the extensional equalities on the level of finite types. Thus, coherence of basic coercions between finite types corresponds to coherence of ordinary set-theoretical functions. Hence the system under consideration can be useful for proof-development with applicatons to finite combinatorial problems.

[1] H. Goguen. *A Typed Operational Semantics for Type Theory*. Ph.D. thesis, Univ. of Edinburgh, 1994.
[2] S. Soloviev and Z. Luo. Coercive subtyping: Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic* 113 (1–3), 297–322, 2002.

Matthieu Sozeau,
Equations: a dependent pattern-matching compiler

We present a compiler for definitions made by pattern matching on inductive families in the Coq system. It allows to write structured, recursive dependently-typed functions, automatically find their realization in the core type theory and generate proofs to ease reasoning on them. The high-level interface allows to write dependently-typed functions on inductive families in a style close to Agda or Epigram, while their low-level implementation is accepted by the vanilla core type theory of Coq. This setup uses the smallest trusted code base possible and additional tools are provided to maintain a high-level view of definitions. The compiler makes heavy use of type classes and the high-level tactic language of Coq for greater genericity and extensibility.

Enrico Tassi,
Nonuniform coercions via unification hints

We introduce the notion of *nonuniform* coercion, which is the promotion of a value of one type to an enriched value of a different type via a nonuniform procedure. Nonuniform coercions are a generalization of the (uniform) coercions known in the literature and they arise naturally when formalizing mathematics in an higher order interactive theorem prover using convenient devices like canonical structures, type classes or unification hints. We also show how nonuniform coercions can be naturally implemented at the user level in an interactive theorem prover that allows unification hints.

Paolo Tranquilli,
A calculus of coercions proving the strong normalization of MLF

We provide a strong normalization proof for MLF, a type system generalizing ML with first-class polymorphism as in system F. The proof is achieved by translating MLF into a calculus of coercions, and showing that this calculus is just a decorated version of system F. Simulation results then entail the result.
This is a joint work with Giulio Manzonetto.

Jarosław Tworek,
Reductions between lambda calculus and combinatory logic

This talk will be about reductions between lambda calculus and combinatory logic. There are well known transformations between these systems, but they do not preserve many properties. This work is focused on defining transformations that preserve both, beta/weak equality and typability of terms, easily transfering many proven results from one of them to the other.

Tarmo Uustalu,
Type-theoretic structured general corecursion

Bove and Capretta's popular method for justifying function definitions by general recursive equations is based on the observation that any structured general recursion equation defines an inductive subset of the intended domain (the "inductive domain of definedness") for which the equation has a unique solution. To accept the definition, it is hence enough to prove that this subset contains the whole intended domain.
This approach works very well for "terminating" ("recursive") definitions. But it fails to account for "productive" ("corecursive") definitions, such as typical definitions of stream-valued functions. I will argue that such definitions can be treated in a similar spirit, proceeding from a different unique solvability criterion. Any structured recursion equation defines a coinductive relation between the intended domain and intended codomain (the "coinductive graph"). This relation in turn determines a subset of the intended domain and a quotient of the intended codomain with the property that the equation is uniquely solved for the subset and quotient. The equation is therefore guaranteed to have a unique solution for the intended domain and intended codomain whenever the subset is the full set and the quotient is by equality. In fact, this condition is also necessary. I will demonstrate this principle at work on examples.

Tao Xue,
Contextual coercive subtyping

Coercive subtyping with coercion contexts is studied. Although the framework makes coercive subtyping more flexible in practical applications, it is still a conservative extension of the original type theory. It is shown that this is an improved formulation of coercive subtyping and its relationships with existing coercive subtyping systems are studied.