ISSN 2186-7437

NII Shonan Meeting Report

No. 2013-11

Coinduction for Computation Structures and Programming Languages

Ichiro Hasuo
Keiko Nakata
Tarmo Uustalu

October 7–10, 2013

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

Coinduction for Computation Structures and Programming Languages

Organizers:
Ichiro Hasuo (University of Tokyo)
Keiko Nakata (Institute of Cybernetics at TUT, Tallinn)
Tarmo Uustalu (Institute of Cybernetics at TUT, Tallinn)

October 7–10, 2013

This meeting has as its aim to study the applications of coinduction (coinductive datatypes/predicates, bisimilarity, corecursion and coinduction) to reasoning about computation and programs, hence in programming language semantics, program logics. This is motivated by the appropriateness of coinduction for analyzing infinity, so also infinity in computation structures, the prime example being infinitely running computations, e.g., of machines or programs, especially in the context of reactive computation.

Induction and coinduction are, by themselves, dual notions. But in typical non-self-dual settings of actual interest they come out as quite different. The most important outcome of the asymmetries is that induction is about finite construction and infinite use, but coinduction is about infinite construction and finite use of data.

Lately there is a growing interest in coinduction in the areas of programming semantics and formal verification of software systems, in part thanks to advances in type-theoretical programming languages and proof assistants. There are many important and practical examples where one needs to reason about finitely observable infinite computations and replacing this by, e.g., inductive reasoning about finite initial fragments thereof is unnatural or inadequate. For instance, one may want to prove that some program transformations do not change the observational behavior of possibly nonterminating transformational programs or of infinitely running reactive programs.

Although coinduction should be very useful, in reality it is not really well understood. In fact, coinduction is surrounded by quite some popular confusions and has become a standard tool only in concurrency theory and coalgebra. The theory of coinduction is on many occasions subtle and challenging. It is often difficult to translate between the vocabularies developed in different disciplines (e.g., coalgebra, proof theory, type theory). The applications are sometimes not supported well by tools. For example, it is by no means clear how to best support coinductive datatypes and predicates in type-theoretical programming languages and proof assistants. Unfortunately, the mechanisms offered by the current systems are weak and/or cumbersome. As a result, the corresponding applications are underdeveloped.

Of interest for the meeting are at least the following topics:

- coinductive computation structures, in particular for possibly infinite behaviors (of machines, programs etc), interaction, concurrency,
- coinductive program semantics, type systems, program logics,
- computability for coinductive data,
- theory of coinductive types, bisimilarity, corecursion/coinduction in type theory, proof theory, category theory,
- support for coinductive types in dependently typed programming languages, type-theoretic proof assistants.

Coinduction is applied by researchers from different communities, with diverse technical backgrounds. There are multiple approaches to coinduction, and it is used in different applications. The meeting encourages exchanges between researchers representing different areas and communities: programming languages design, implementation, semantics, functional programming, theory of concurrency, category theory (coalgebra), proof theory, and type theory.

List of Participants

Ichiro Hasuo (University of Tokyo, JP)
Tarmo Uustalu (Institute of Cybernetics at TUT, Tallinn, EE)

Ana Sokolova (Universität Salzburg, AT)
Robin Cockett (University of Calgary, CA)
Sergey Goncharov (Friedrich-Alexander-Universität Erlangen-Nürnberg, DE)
Martin Hofmann (Ludwig-Maximilians-Universität München, DE)
Stefan Milius (Friedrich-Alexander-Universität Erlangen-Nürnberg, DE)
Lutz Schröder (Friedrich-Alexander-Universität Erlangen-Nürnberg, DE)
James Chapman (Institute of Cybernetics at TUT, Tallinn, EE)
Wolfgang Jeltsch (Institute of Cybernetics at TUT, Tallinn, EE)
Varmo Vene (University of Tartu, EE)
Davide Ancona (Università di Genova, IT)
Marino Miculan (Università di Udine, IT)
Kazuyuki Asada (University of Tokyo, JP)
Kenichi Asai (Ochanomizu University, JP)
Makoto Hamana (Gunma University, JP)
Zhenjiang Hu (National Institute of Informatics, JP)
Yoshihiko Kakutani (University of Tokyo, JP)
Shin-ya Katsumata (RIMS, Kyoto University, JP)
Kazutaka Matsuda (University of Tokyo, JP)
Eijiro Sumii (Tohoku University, JP)
Jan Rutten (CWI & Radboud Universiteit Nijmegen, NL)
Alexandra Silva (Radboud Universiteit Nijmegen, NL)
Andreas Abel (University of Gothenburg, SE)
Nils Anders Danielsson (Chalmers University of Technology, SE)
Bengt Nordström (Chalmers University of Technology, SE)
Ulrich Berger (Swansea University, UK)
Venanzio Capretta (University of Nottingham, UK)
Ekaterina Komendantskaya (University of Dundee, UK)
Paul Blain Levy (University of Birmingham, UK)
Anton Setzer (Swansea University, UK)

Programme

Monday, October 7, 2013

Morning:

Zhenjiang Hu: can we go from MapReduce to ExpandMap?
Andreas Abel: Wellfounded recursion with copatterns
Venanzio Capretta: The liberation of coinductive objects

Afternoon:

Anton Setzer: Patterns and copatterns - decidability and unfolding
Ichiro Hasuo: Fibered coinduction
Robin Cockett: Datatypes for concurrency: message-passing and protocols
Stefan Milius: Guard your daggers and traces: on the equational properties of guarded (co-)recursion
Alexandra Silva: Coalgebraic up-to techniques

Tuesday, October 8, 2013

Morning:

Shin-ya Katsumata: Categorical $\top\top$-lifting and preorders on monads
Paul Levy: Final coalgebras from modal logic
Kazuyuki Asada: Epsilon-elimination: by categorification or by free-construction
Kazutaka Matsuda: Graph transformation as graph reduction

Afternoon:

Jan Rutten: Syntactic monoids and their dual
Martin Hofmann: Operational semantics and typing for infinite traces
Nils Anders Danielsson: Operational semantics using the partiality monad
Eijiro Sumii: Environmental bisimulations and their open questions
Makoto Hamana: Multiversal polymorphic algebraic theories

Wednesday, October 9, 2013

Morning:

Lutz Schröder: Simulations and bisimulations via predicate liftings
Sergei Goncharov: Weak bisisimulation for monad-type coalgebras
Marino Miculan: Weak bisimulations for LTSs weighted over semirings

Thursday, October 10, 2013

Morning:

Ulrich Berger: Induction and coinduction for program extraction
James Chapman: Restriction categories and the delay monad
Tarmo Uustalu: Finiteness, constructively

Afternoon:

Bengt Nordström: Coinductive structures and Martin-Löf's meaning-theoretic explanations
Ana Sokolova: Determinization and non-determinization for semantics
Ekaterina Komendantskaya: Building a small programming language from scratch, coalgebraically
Davide Ancona: Semantic subtyping between coinductive types in object-oriented languages
Wolfgang Jeltsch: Towards a categorical semantics for functional reactive programming with mutable state

Overview of Talks

Can we go from MapReduce to ExpandMap?

Zhenjiang Hu (National Institute of Informatics, Japan)

MapReduce is known to be very useful for parallel data analysis, and the concept of homomorphism (a composition of map and reduce) plays an important role in programming MapReduce behind. In this talk, I would argue that its dual, ExpandMap, which is little known, is worth more attention. ExpandMap is useful for parallel data generation, and that the dual concept of homomorphism would play a significant role in programming ExpandMap.

Wellfounded recursion with copatterns

Andreas Abel (Gothenburg University)

Inductive datatypes provide mechanisms to define finite data such as finite lists and trees via constructors and allow programmers to analyze and manipulate finite data via pattern matching. In this work, we develop a dual approach for working with infinite data structures such as streams. Infinite data inhabits coinductive datatypes which denote greatest fixpoints. Unlike finite data which is defined by constructors we define infinite data by observations. Dual to pattern matching, a tool for analyzing finite data, we develop the concept of copattern matching, which allows us to synthesize infinite data. This leads to a symmetric language design where pattern matching on finite and infinite data can be mixed.

We present a core language for programming with infinite structures by observations together with its operational semantics based on (co)pattern matching. The duality of pattern and copatterns provide a unifying semantic concept which allows us to elegantly and uniformly support both well-founded induction and coinduction by mere rewriting. Sized inductive types are used to track tree height for guaranteeing termination, and sized coinductive types track observation depth for guaranteeing productivity.

This work is a significant step towards representing observation-centric infinite data in proof assistants such as Coq and Agda.

This is joint work with Brigitte Pientka.

The liberation of coinductive objects

Venanzio Capretta (University of Nottingham)

The talk addresses the foundations of coinductive types, the problem of automatically proving productivity and proposals for more general infinite data structures.

In Martin-Löf's type theory types are defined by giving their constructors; eliminators and computation rules are inferred from them. For coinductive objects the relation is inverted: observations and dynamics define the objects and introduction rules are inferred from them.

We want to define streams, and other coinductive structures, by recursive equations. It is possible that an equation is not guarded by constructors but still

productive. The problem is in general undecidable, but there are some powerful partial algorithms to derive productivity. One consists in creating an inductive type of partially evaluated expressions and defining a coalgebra on them.

Further generalization of coinductive objects leads to the notion of *wander types*. These are the non-well-founded version of inductive-recursive definitions. They allow the specification of a type of potentially infinite structures whose branching behaviour is controlled by a simultaneously defined function on the type. They generalize many of the data structures that have been studied in recent times.

Patterns and copatterns—decidability and unfolding

Anton Setzer (Swansea University)

Continuing the talk by Andreas Abel on nested (co)pattern matching on (co)algebras (POPL'13), we will show how to unfold nested (co)pattern matching, reduce it to (co)recursion operators, and, in the case which should be accepted by a termination checker, to primitive (co)recursion operators.

We will present as well a little theorem, which restricts what can be achieved in intensional type theory regarding decidable type checking. Even assuming that streams are equal if their heads and tails are equal forces an equality on streams to be undecidable.

Fibered coinduction

Ichiro Hasuo (University of Tokyo)

Coinductive predicates express persisting "safety" specifications of transition systems. Previous observations by Hermida and Jacobs identify coinductive predicates as suitable final coalgebras in a fibration–a categorical abstraction of predicate logic. In this paper we follow the spirit of a seminal work by Worrell and study final sequences in a fibration. Our main contribution is to identify some categorical "size restriction" axioms that guarantee stabilization of final sequences after omega steps. In its course we develop a relevant categorical infrastructure that relates fibrations and locally presentable categories, a combination that does not seem to be studied a lot. The genericity of our fibrational framework can be exploited for: binary relations (i.e., the logic of "binary predicates") for which a coinductive predicate is bisimilarity; constructive logics (where interests are growing in coinductive predicates); and logics for name-passing processes.

This is joint work with Kenta Cho, Toshiki Kataoka and Bart Jacobs.

Reference:

I. Hasuo, K. Cho, T. Kataoka and B. Jacobs. Coinductive predicates and final sequences in a fibration. In D. Kozen and M. Mislove, eds., *Proc. of 29th Conf. on Mathematical Foundations of Programming Semantics, MFPS XXIX*, v. 298 of *Electron. Notes in Theor. Comput. Sci.*, pp. 197–214, 2013. doi:10.1016/j.entcs.2013.09.014

Datatypes for concurrency: message-passing and protocols

Robin Cockett (University of Calgary)

Because they can have a hidden or internal state coinductive programs are often associated with concurrent programs. However, a true system for concurrent programming should include primitives for message passing and handling communication channels. This talk describes such a system and its categorical semantics which is built on top of a sequential programming world. One may then ask what datatypes (i.e., initial and final fixed points) are at this concurrent level. It turns out they are protocols for interaction along a channel. The concurrent world is essentially self-dual and, thus, initial and final datatypes do not behave very differently. However, they are also nothing like sequential datatypes. Programs which use protocols almost always involve an internal state.

The talk will give a brief overview of protocols for concurrency.

Guard your daggers and traces: on the equational properties of guarded (co-)recursion

Stefan Milius (Friedrich-Alexander Universität Erlangen-Nürnberg)

Motivated by the recent interest in models of guarded (co-)recursion we study its equational properties. We formulate axioms for guarded fixpoint operators generalizing the axioms of iteration theories of Bloom and Ésik. Models of these axioms include both standard (e.g., cpo-based) models of iteration theories and models of guarded recursion such as complete metric spaces or the topos of trees studied by Birkedal et al. We show that the standard result on the satisfaction of all Conway axioms by a unique dagger operation generalizes to the guarded setting. We also introduce the notion of guarded trace operator on a category, and we prove that guarded trace and guarded fixpoint operators are in one-to-one correspondence. Our results are intended as first steps leading to the description of classifying theories for guarded recursion and hence completeness results involving our axioms of guarded fixpoint operators in future work.

This is joint work with Tadeusz Litak.

Reference:

S. Milius and T. Litak. Guard your daggers and traces: on the equational properties of guarded (co-)recursion. In D. Baelde and A. Carayol, eds., *Proc. of 9th Wksh. on Fixed Points in Computer Science, FICS '13*, v. 126 of *Electron. Proc. in Theor. Comput. Sci.*, pp. 72–86, 2013. doi:10.4204/eptcs.126.6

Coalgebraic up-to techniques

Alexandra Silva (Radboud University Nijmegen)

We present a systematic study of bisimulation-up-to techniques for coalgebras. This enhances the bisimulation proof method for a large class of state based systems, including labelled transition systems but also stream systems and weighted automata.

Our approach allows for compositional reasoning about the soundness of enhancements.

Applications include the soundness of bisimulation up to bisimilarity, up to equivalence and up to congruence. All in all, this gives a powerful and modular framework for simplified coinductive proofs of equivalence.

This is joint work with Filippo Bonchi, Marcello Bonsangue, Damien Pous, Jurriaan Rot and Jan Rutten.

Categorical $\top\top$-lifting and preorders on monads

Shin-ya Katsumata (RIMS, Kyoto University)

In the first part, I will introduce *categorical* $\top\top$*-lifting*, a method to construct liftings of strong monads across fibrations. This construction takes a parameter, and by varying it, the $\top\top$-lifting yields various liftings of strong monads. I will illustrate the $\top\top$-lifting of the finite powerset monad.

I then move to a coalgebraic theory of simulation relations. There, the concept of preordered functors are employed to capture simulation relations between coalgebras. I focus on the concept of preordered monads, and analyse it using the categorical $\top\top$-lifting. The main result is that the class of preorders on a monad is a large limit of an ω^{op}-chain of congruent and substitutive preorders on the monad. Using this, we enumerate the preorders of certain monads.

Final coalgebras from modal logic

Paul Levy (University of Birmingham)

It is well-known that two nodes of a transition system are bisimilar iff they satisfy the same set of Hennessy-Milner formulas. This suggests we can construct a final coalgebra whose elements are sets of Hennessy-Milner formulas. That has been done by Adámek et al in the case of finitary systems, based on the "canonical model" of modal logic, which is known to be a model by a Lindenbaum-style argument that does not work in the infinitary case. This talk gives a different construction that works also in the infinitary case: taking an injectively structured corecursive algebra and cutting it down to obtain a final coalgebra.

Epsilon–elimination: by categorification or by free-construction

Kazuyuki Asada (University of Tokyo)

In this talk, I will explain on-going work about epsilon-elimination for graphs (or automata) having epsilon-edges. Epsilon-elimination is defined using an iteration operator—the dual notion of a fixed-point operator—in the Kleisli category of a monad T by which we define graphs.

We give two ways to construct a monad with iteration operator in the Kleisli category. One is by categorification of the approach using the "trace situation" by (Hasuo, PhD thesis). This approach adds a new example of monads with iteration: viz., the monad of countable lists. The other way is a free construction of monad-with-iter from a given monad. We give such a free construction in some

impredicative way, whose impredicativity can be avoided by usual cardinality technique, for ranked monads. We also (very briefly) compare the two methods.

This is joint work with Soichiro Hidaka, Hiroyuki Kato, Zhenjiang Hu, Keisuke Nakano and Naohiko Hoshino.

Graph transformation as graph reduction: a functional reformulation of UnCAL

Kazutaka Matsuda (University of Tokyo)

In this talk, we will introduce a simple functional programming language and a modal type system so that (1) every possibly-infinite trees in the language is guaranteed to be regular, and (2) the evaluation of a program written in the language can be done in finite time in a lazy way. The language consists of λ-expressions with restricted form recursions, and thus we can reason about the programs written in the language as those in ordinary functional programming languages. Although the language is simple, it can express graph transformations written in UnCAL (Buneman et al., VLDB 00), which has been studied in the database community, as (possibly-infinite) tree transformations in the language. This result enables us to apply program-manipulation techniques, such as optimization and verification, studied in the functional programming community to UnCAL graph transformations studied in the database community.

This is joint work with Kazuyuki Asada.

Syntactic monoids and their dual

Jan Rutten (CWI & Radboud University Nijmegen)

Because of the isomorphism $(X \times A) \to X \cong X \to (A \to X)$, the transition structure of a deterministic automaton with state set X and with inputs from an alphabet A can be viewed both as an algebra and as a coalgebra. This algebra-coalgebra duality goes back to Arbib and Manes, who formulated it as a duality between reachability and observability, and is ultimately based on Kalman's duality in systems theory between controllability and observability. Recently, it was used to give a new proof of Brzozowski's minimization algorithm for deterministic automata.

In this talk, we will discuss deterministic automata as an elementary and typical example for the combined use of algebraic and coalgebraic methods in computer science. The algebra-coalgebra duality of automata will, more specifically, serve as a common perspective for the study of both varieties and covarieties, which are classes of automata and languages defined by equations and coequations, respectively.

Along the way, we will establish a first connection with Eilenberg's definition of varieties of languages, which is based on the classical, algebraic notion of varieties of (transition) monoids.

This is joint work with Adolfo Ballester-Bolinches and Enric Cosme-Llopez (Universitat de València).

Operational semantics and typing for infinite traces

Martin Hofmann (Ludwig-Maximilians-Universität München)

We consider functional programs with recursive procedures and special "event-issuing" commands so that the side-effect of an execution will be a finite (in case of termination or idling) or infinite (in case of "productive" nontermination) trace of events. We use an inductive definition to specify this semantics but would be interested to discuss whether a coinductive version might be more convenient. We then devise a type system capable of certifying infinitary properties of such traces, in particular liveness properties. We also discuss issues of fairness.

This is joint work with Wei Chen.

Operational semantics using the partiality monad

Nils Anders Danielsson (University of Gothenburg)

The operational semantics of a partial, functional language is often given as a relation rather than as a function. The latter approach is arguably more natural: if the language is functional, why not take advantage of this when defining the semantics? One can immediately see that a functional semantics is deterministic and, in a constructive setting, computable.

In the talk I will show how one can use the coinductive partiality monad to define big-step or small-step operational semantics for lambda-calculi and virtual machines as total, computable functions (total definitional interpreters). To demonstrate that the resulting semantics are useful I have proved type soundness and compiler correctness results.

Reference:

N. A. Danielsson. Operational semantics using the partiality monad. In *Proc. of 17th ACM SIGPLAN Int. Conf. on Functional programming, ICFP '12*, pp. 127–138. ACM Press, 2012. doi:10.1145/2364527.2364546

Environmental bisimulations and their open questions

Eijiro Sumii (Tohoku University)

Environmental bisimulations are a method of proving contextual equivalence in a variety of languages, in particular with higher-order functions or processes, and with information hiding such as type abstraction or encryption. I will present a brief historical and technical overview of environmental bisimulations, and discuss some open issues that still need to be addressed, perhaps with the help of ideas from other techniques such as category theory for coinduction.

Multiversal polymorphic algebraic theories

Makoto Hamana (Gunma University)

I will introduce my recent work about polymorphic extension of algebraic theories.

By polymorphic algebraic theory, we mean a theory presented by equations between polymorphically-typed terms with both type and term variable binding. The prototypical example is System F, but our framework applies more widely, such as extensions of System F, existential lambda calculus, and theories of effects, etc.

The work is semantically driven using categorical tools: generalised polynomials, monoids in presheaf categories, and aspects of categorical universal algebra.

This is joint work with Marcelo Fiore.

Simulations and bisimulations via predicate liftings

Lutz Schröder (Friedrich-Alexander-Universität Erlangen-Nürnberg)

Simulations serve as a proof tool to compare the behaviour of reactive systems. We define a notion of Λ-simulation for coalgebraic modal logics, parametric in the choice of a set Λ of monotone predicate liftings for a functor T. That is, we obtain a generic notion of simulation that can be flexibly instantiated to a large variety of systems and logics, in particular in settings that semantically go beyond the classical relational setup, such as probabilistic, game-based, or neighbourhood-based systems. We show that this notion is adequate in several ways: i) Λ-simulations preserve truth of positive formulas, ii) for Λ a separating set of monotone predicate liftings, the associated notion of Λ-bisimulation corresponds to T-behavioural equivalence (moreover, this correspondence extends to the respective finite-lookahead counterparts), and iii) Λ-bisimulations remain sound when taken up to difunctional closure. In essence, we arrive at a modular notion of equivalence that, when used with a separating set of monotone predicate liftings, coincides with T-behavioural equivalence regardless of whether T preserves weak pullbacks. That is, for finitary set-based coalgebras, Λ-bisimulation works under strictly more general assumptions than T-bisimulation in the sense of Aczel and Mendler.

The basic results on simulations have appeared in the proceedings of CALCO 2013.

Simulations moreover form the basis for further work on lightweight coalgebraic logics where logical consequence between formulas can be reduced to simulation between suitable associated models. We briefly explain how this phenomenon can be reduced to a local view of coalgebras, the so-called one-step view, in the usual spirit of coalgebraic logic. Key results obtained in this way include tractability of the conjunctive fragment of K with only boxes, the bounded-depth conjunctive fragments of full KD (with boxes and diamonds), the conjunctive fragment of graded logic with only one graded box, and the conjunctive fragments of several variants of monotone neighbourhood logic.

This is joint work with Daniel Gorín.

Weak bisimulation for monad-type coalgebras

Sergey Goncharov (Friedrich-Alexander-Universität Erlangen-Nürnberg)

Strong bisimulation, in the sense of Park and Milner, is one of the central equivalences of process calculus. Over the years, various definitions of strong

bisimulation for structurally richer systems such as probabilistic transition systems and integer weighted systems have emerged and were found to be instances of the more general notion of coalgebraic bisimulation. The situation for weak bisimulation is much less pleasant. While there are various notions of weak bisimulation for structurally richer classes of systems, we currently lack a satisfactory coalgebraic description of weak bisimulation that would subsume these various existing notions in a more general framework.

We attempt to close this gap: we introduce a generic (coalgebraic) definition of weak bisimulation that instantiates to the concrete notions studied in the literature—most prominently the probabilistic one—study basic structural properties and show how derive logics for weak bisimulation from their strong counterparts. Our key insight lifts Milner's double-arrow construction to the coalgebraic level which allows us to understand weak bisimilarity as strong bisimilarity in a transformed system.

This is joint work with Dirk Pattinson.

Weak bisimulations for LTSs weighted over semirings

Marino Miculan (Università di Udine)

Weighted labelled transition systems are LTSs whose transitions are given weights drawn from a commutative monoid. WLTSs subsume a wide range of LTSs, providing a general notion of strong (weighted) bisimulation. In this talk we extend this framework towards other behavioural equivalences, by considering *semirings* of weights. Taking advantage of this extra structure, we introduce a general notion of *weak weighted bisimulation*. We show that weak weighted bisimulation coincides with the usual weak bisimulations in the cases of non-deterministic and fully-probabilistic systems, among others; moreover, it naturally provides a definition of weak bisimulation also for kinds of LTSs where this notion is currently missing. As an example, we define a weak bisimulation for stochastic systems, using a new semiring of *transition-time random variables*.

As an application of this construction, we give a generic algorithm for deciding weak bisimulations, parametric in the underlying semiring structure. Therefore, the same algorithm can be used for the mechanized analysis and verification of many kinds of systems.

Finally, we provide a categorical account of the coalgebraic construction of weak weighted bisimulation; this construction points out how to port our approach to other equivalences based on different notion of observability.

This is joint work with Marco Peressotti.

Induction and coinduction for program extraction

Ulrich Berger (Swansea University)

We give an overview of program extraction from constructive proofs based on a uniform version of realizability. "Uniform" means that the objects one quantifies over are not assumed to have computational content, hence a realizer of a universally quantified formula must be independent of the quantified variable. This style of realizability supports program extraction in abstract mathematics.

We present a constructive version of Church's simple theory of types as a suitable formal framework for program extraction, and discuss a uniform approach to the realizability of various induction principles.

Restriction categories and the delay monad

James Chapman (Institute of Cybernetics, Tallinn)

Restriction categories are an axiomatic framework for modelling partial functions in category theory, developed by Cockett and Lack. I will explain the basics of restriction categories and our use of them to analyze Caprettas delay monad in type theory. I will also explain our ongoing formalisation in Agda.

This is joint work with Tarmo Uustalu and Niccolò Veltri.

Finiteness, constructively

Tarmo Uustalu (Institute of Cybernetics, Tallinn)

In constructive logic, there is no single right definition of a finite set or predicate. Instead there is a whole spectrum of viable options, notably listability and Noetherianness.

I will discuss these and hint at the implications for rational trees as nonwellfounded trees with finitely many distinct subtrees.

This talk is based on joint work with Marc Bezem and Keiko Nakata.

Coinductive structures and Martin-Löf's meaning-theoretic explanations

Bengt Nordström (Chalmers University of Technology)

The justifications of the rules in type theory are based on the operational semantics of the language. An expression a which has a type A, $a : A$, can always be computed to a value in A, i.e., all welltyped expressions terminate. A set is defined by the typing of its constructors (like data in Haskell).

A way to understand coinductive structures is to try to put them into the framework of type theory. But then we have to allow computations which goes on forever (but produces result in doing so).

The intuition is that an expression e is productive if it computes to canonical form $(c\ e_1\ \ldots\ e_n)$ where c is a constructor and all expressions $e_1, \ldots, e_n$ are productive. We do not want to require that this definition is well-founded (i.e., that the computation finally ends up in canonical expressions with no parts). Instead, we want to express that we can continue to evaluate all subexpressions ad infinitum.

In order to analyze this situation, it is possible to associate an inductively defined set D (the evaluation degree) to each coinductively defined set C. The set D can be automatically deduced from the definition of C. We then index the evaluation relation $a \to b$ (the value of a is b) with an object d in D. The idea is that the object d expresses the degree of evaluation (if d is 0, we do not do any evaluation, else we compute a to a constructor form $(c\ e_1\ \ldots\ e_n)$ and then continue to compute e_1, etc.

The main idea is then that the meaning of $c : C$ for a coinductively defined set C is that c computes to a value for all evaluation degrees d in D.

Determinization and non-determinization for semantics

Ana Sokolova (University of Salzburg)

Determinizations provide a fresh look at the topic of trace semantics for coalgebras. The first development of coalgebraic trace semantics used final coalgebras in Kleisli categories, stemming from an initial algebra in the underlying category. This approach requires some non-trivial assumptions like dcpo enrichment, which do not always hold, even in cases where one can reasonably speak of traces (e.g., weighted automata). More recently, it has been noticed that trace semantics can also arise by first performing a determinization construction. In our work (joint with Bart Jacobs and Alexandra Silva), we develop a systematic approach in which the two approaches correspond to different orders of composing a functor and a monad, and accordingly, to different distributive laws. The relevant final coalgebra that gives rise to trace semantics does not live in a Kleisli category, but more generally, in a category of Eilenberg-Moore algebras. In order to exploit its finality, we identify an extension operation that changes the state space of a coalgebra into a free algebra, which abstractly captures determinization of automata. We show that the two different views on trace semantics are equivalent, in the examples where both approaches are applicable.

Moreover, not only determinizations are of interest (changing the coalgebra into a deterministic automaton over a free algebra) but also non-determinizations, e.g. changing a probabilistic/nondeterministic system into a nondeterministic system over the (free) convex algebra of probability distributions. Such may be valuable for verification purposes. In an ongoing joint work with Filippo Bonchi and Alexandra Silva, we are trying to come up with useful non-determinizations of probabilistic/nondeterministic automata and understand them in a generic categorical setting.

Building a small programming language from scratch, coalgebraically.

Ekaterina Komendantskaya (University of Dundee)

If one decides to build a new programming language based purely upon constructions arising from coalgebraic semantics for predicate logic—how would this programming language look like?

In this talk, I will give a brief description of Coalgebraic Logic Programming (CoALP)—a new language from "Logic Programming" family, built upon Coalgebraic (SOS-style) Semantics recently introduced by myself and J. Power. I will compare its features with other existing parallel and coinductive logic programming dialects and conclude with some remarks on how the language can be used in type inference in other programming languages.

This is joint work with John Power, Martin Schmidt, Jónathan Heras and Vladimir Komendantsky.

Semantic subtyping between coinductive types in object-oriented languages

Davide Ancona (Università di Genova)

Semantic subtyping is an intuitive approach to subtyping where types are interpreted as set of values, and subtyping corresponds to set inclusion between type interpretations. The XDuce and CDuce languages exploit semantic subtyping and support a sound and complete subtyping algorithm; since their main aim is the type safe manipulation of XML documents, types are interpreted inductively. However, in object-oriented languages, like in many other general purposes programming languages, objects can also be cyclic, hence a coinductive interpretation of types is preferable; indeed, coinductive types are more expressive for specifying sets of cyclic values. We will present a system of subtyping rules for coinductive record and union types, prove its soundness wrt. semantic subtyping, and conjecture its completeness.

This is joint work with Andrea Corradi.

Towards a categorical semantics for functional reactive programming with mutable state

Wolfgang Jeltsch (Institute of Cybernetics, Tallinn)

In his paper *A mixed linear and non-linear logic: proofs, terms and models*, Benton developed a categorical semantics that models the interaction between intuitionistic and linear logic by a symmetric lax monoidal adjunction. Since intuitionistic logic corresponds to functional programming and linear logic corresponds to programming with mutable state, Benton's semantics also models functional programming with integrated mutable state.

In this talk, I will show that we can use symmetric lax monoidal adjunctions also to model interactions between non-temporal and temporal logics. This gives us a semantics for the interaction between functional programming and functional reactive programming (FRP). It furthermore leads to a semantics for FRP with integrated mutable state.