DAYS IN λOGIC 2014

23-25 January | Universidade do Minho | Braga | Portugal

Booklet of abstracts

(*courses and contributed talks*)

DAYS IN λOGIC 2014

23-25 January | Universidade do Minho | Braga | Portugal

Courses by Invited Speakers

Alex Simpson (University of Edinburgh, U.K.)
Real numbers in type theory

Luís Antunes (Universidade do Porto, Portugal)
A crash course on computational complexity

Mário Edmundo (Universidade Aberta, Portugal)
Model theory (analytic part): from Grothendieck to André-Oort

Michael Rathjen (University of Leeds, U.K.)
Proof Theory: From arithmetic to set theory

Real numbers in type theory

Alex Simpson
Alex.Simpson@ed.ac.uk

University of Edinburgh, U.K.

Abstract. The course will look at two different approaches to implementing real-number computation in type theory. One approach is to encode real numbers using any one of a range of standard encodings (Cauchy sequences, signed binary, etc.). An alternative, which will be the main focus of the course, is to add real numbers as an abstract datatype, thereby providing an implementation-independent interface for real-number computation. One substantial issue concerning the second approach is how to realise it at all: what is a suitable interface for real-number computation? A second concern is to relate the two approaches.

To keep matters simple, much of the course will look at implementing the closed interval [-1,1] of real numbers in the basic setting of simply-typed lamda-caclculus with natural numbers (Goedel's System T). But the course will end by extending to the full real line, on the one hand, and dependent type theory, on the other.

Lecture 1: Simply-typed lambda-calculus and System T. Encoding real-number computation in System T. Adding an abstract datatype for [-1,1]. Programming using the abstract datatype. Equational reasoning with the abstract datatype.

Lecture 2: Implementing the abstract datatype via compilation back to system T. Completeness of the abstract datatype relative to T-computability. Necessity of the double function.

Lecture 3: Implementing the full real line. Extending the type theory to dependent type theory. Directions for further research.

The course is based longstanding but still ongoing joint work with Martin Escardo (University of Birmingham).

A crash course on computational complexity

Luís Antunes
lfa@dcc.fc.up.pt

Universidade do Porto, Portugal

Abstract. Computational Complexity had its origin in the mid 60s. It aims to investigate the intrinsic complexity of computational problems, and is based on the amount of resources (such as time and/or space and/or randomness and/or parallelism) necessary to solve a computational task (or class of tasks). During these three lectures we will briefly describe the motivation for the study of computational complexity and cover some historical marks. Then we will survey the major classical results and open problems. During the last lecture we will present the notion of Kolmogorov complexity and show that it can be an useful tool in computational complexity.

Model theory (analytic part): from Grothendieck to André-Oort

Mário Edmundo
edmundo@cii.fc.ul.pt

Universidade Aberta and CMAF, Universidade de Lisboa, Portugal

Abstract. O-minimality is the analytic part of model theory and deals with theories of ordered, hence topological, structures satisfying certain tameness properties. O-minimality was isolated by van den Dries as the requisite condition to prove the basic structural results of semialgebraic geometry and then by Steinhorn and Pillay in its logical form. The definition of o-minimality is rather simple and innocent looking, surprisingly however this notion turned out to be very deep with somewhat unexpected applications: it generalizes semi-algebraic geometry and globally sub-analytic geometry and it is claimed to be the formalization of Grothendieck's notion of tame topology (topologie modérée). In this short course we will: (i) introduce the audience to the subject; (ii) mention the recent development of the formalism of the Grothendieck six operations on o-minimal sheaves (a generalization and a new approach to similar formalisms for semi-algebraic sheaves (Delfs) and sub-analytic sheaves (Kashiwara-Schapira)); (iii) mention the application of o-minimality to a recent unconditional proof of the Andre-Oort conjecture for arbitrary products of modular curves by J. Pila (previous proofs were known only in some special cases and some under the Generalized Riemann Hypothesis).

Proof Theory: From arithmetic to set theory

Michael Rathjen
rathjen@maths.leeds.ac.uk

University of Leeds, U.K.

Abstract. Ordinal analysis of theories is a core area of proof theory. The origins of proof theory can be traced back to the second problem on Hilbert's famous list of problems. Proof theory was invented as the main tool for carrying out Hilbert's programme. In the main, ordinal-theoretic proof theory came into existence in 1936, springing forth from Gentzen's consistency proof of arithmetic. The intent of the talks is to elucidate the underlying notions, the ubiquitous tool of cut elimination and the rationale of ordinal-theoretic proof theory by relating the developments from Gentzen up to more recent advances in ordinal analysis of set theories.

Contributed talks

Diana Costa, Dept. Mathematics, Univ. Aveiro, Portugal
(with: Manuel A. Martins)
Paraconsistency in hybrid logic

Eduardo Hermo Reyes, Dept. Logic, History Phil. Science, Univ. Barcelona, Spain
(with: Joost J. Joosten, and Laia Jornet Somoza)
Worms, Ordinal Worms and Ordinal Analysis

Fernando Ferreira, Univ. Lisboa, Portugal
New proof-theoretic facts about $KP\omega$

Filipe Casal, SQIG, Instituto de Telecomunicações, Lisboa, Portugal
(with: João Rasga)
SMT and Theory Combination Techniques

Jaime Gaspar, Dept. Comp. Eng. and Math., Univ. Rovira i Virgili, Catalonia, Spain, and CMA, Univ. Nova Lisboa, Portugal
Completeness of Peano arithmetic with the ω*-rule*

João Enes, Univ. Lisboa, Portugal
A new interpretation of ID_1 *in* $\mathsf{ID}_1(\mathsf{W})$

Luís Barbosa, INESC TEC - HASLab, Univ. Minho, Portugal
(with: Alexandre Madeira, Manuel A. Martins, and Renato Neves)
Hybridize to Specify: institution-independent foundations for reconfigurable systems specification

Mário Pereira, LIACC and Dept. Computer Science, Univ. Porto, Portugal
(with: Sandra Alves, and Mário Florido)
Liquid Types Revisited

Pridon Alshibaia, Tbilisi State Univ., Georgia
On Finitely Valued Bimodal Symmetric Gödel Logics

Tarmo Uustalu, Institute of Cybernetics and Tallinn Univ. of Technology, Estonia
Finiteness and rational datatypes, constructively

Paraconsistency in hybrid logic

Diana Costa
Department of Mathematics, University of Aveiro
Manuel A. Martins
Center for R&D in Mathematics and Applications, Dep. of Mathematics, University of Aveiro

When collecting information one can come across with inconsistencies in various forms and there must be a way to deal with them. In fact, databases, knowledgebases and software specifications very often carry inconsistencies and we should be able to reason about this kind of data having, at the same time, assertions of the form q and $\neg q$ (local inconsistencies) without producing global inconsistency. Paraconsistent reasoning is the natural way to deal with inconsistencies. There are several paraconsistent logics studied in the literature [4]. One of them is the quasi-classical (QC) logic proposed in [1, 3]. This logic turns out to be powerful since it provides a measure for the inconsistency of data represented as a set of first-order formulas. Measuring inconsistency is crucial to an effective management of the information. It is worth to be able to compare different knowledgebases in what concerns to inconsistencies and choose the one with less conflicts.

In [3], the notion of Tarski's satisfaction is decoupled by considering two interpretations for the propositional symbols: one for positive literals and the other for negative literals. As in the standard case, models (called bistructures) can be represented by a set of ground literals. The definition of minimal QC models is introduced and it is proved that no useful information is lost when using only these models. The main result of the paper is the way we can determine the inconsistency measure of a model; it is given by the quotient between the number of inconsistencies on the bistructure and the total possible number of inconsistencies on it.

In this talk we present an introduction to the study of paraconsistency in hybrid logic ([2]) following the work by Grant and Hunter ([3]). One important result that allows this generalization is the existence of Robinson diagrams in (global) hybrid logical. Moreover, we can represent hybrid models by a set of hybrid formulas in an extended language with new nominals such that all worlds are named. In order to avoid double negation, we assume that all formulas are in negation normal form. The generalization is not straightforward as we will explain, but the analogue of many notions can also be formulated in this context, for example: bistructure, conflict base, minimal model, etc.

Acknowledgements. This work was supported by the FCT project FCOMP-01-0124-FEDER-028923 (Nasoni).

References

[1] P. Besnard and A. Hunter. Quasi-classical logic: Non-trivializable classical reasoning from inconsistent information. In C. Froidevaux and J. Kohlas, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 946 of *Lecture Notes in Computer Science*, pages 44–51. Springer Berlin Heidelberg, 1995.

[2] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

[3] J. Grant and A. Hunter. Measuring inconsistency in knowledgebases. *Journal of Intelligent Information Systems*, 27(2):159–184, 2006.

[4] C. A. Middelburg. A survey of paraconsistent logics. *CoRR*, abs/1103.4324, 2011.

Worms, Ordinal Worms and Ordinal Analysis

Eduardo Hermo Reyes, Joost J. Joosten, Laia Jornet Somoza

Department of Logic, History and Philosophy of Science - Universitat de Barcelona
Barcelona, Spain
ehermo.reyes@gmail.com,jjoosten@ub.edu,laiajornet@gmail.com

Gödel-Löb Polymodal logic **GLP** is a provability logic that has for each ordinal α a modality $[\alpha]$, whose intended interpretation is a provability predicate in a hierarchy of theories of increasing strength. The logic $\mathbf{GLP}_\omega$ -that only has modalities $[n]$ for $n < \omega$- was first introduced by Japaridze, and recently applied by Beklemishev to give a Π^0_1-ordinal analysis of Peano Arithmetic and related systems. This ordinal analysis was carried out within the *closed fragment* of $\mathbf{GLP}_\omega$. Within this fragment, we find some particular terms; formulas of the form $\langle n_0\rangle \ldots \langle n_j\rangle\top$ -so called *worms*- that constitute an alternative ordinal notation system for ordinals below ϵ_0.

Another interesting property of $\mathbf{GLP}_\omega$ is that we have arithmetical completeness for a wide range of interpretations of $[n]$. In particular, $\mathbf{GLP}_\omega$ is sound and complete when reading $[n]$ as "provable in EA together with all true Π^0_n sentences". This reading is closely related to Turing progressions, that are hierarchies of theories such that given an initial theory T, we can construct a transfinite sequence of extensions of T by iteratedly adding n-consistency statements. Nevertheless, this link between **GLP** and Turing progressions is only by approximating the progression, so that it requires many technical results.

A weaker system, called *Reflection Calculus* **RC**, was introduced by Beklemishev and Dashkov. It is much simplier than **GLP** but yet expressive enough to maintain its main proof theoretic applications as the ones mentioned above. From the point of view of modal logic, **RC** can be seen as the positive fragment of **GLP**. An advantage of going to a positive language is that we gain a more general arithmetical interpretation. Since we discard some elements as the negation, modal formulas can be interpreted as arithmetical theories rather than arithmetical sentences.

In order to get a logic which can be used to directly denote Turing progressions, positive language together with some special worms, called *ordinal worms*, seems to be appropriate. These ordinals worms are built up from a new modality $\langle\alpha, A\rangle$, where α is an ordinal and A is a worm. The intended interpretation of this new modality would be:

$$\langle\alpha, A\rangle\varphi \equiv \langle\alpha\rangle^{o(A)}\varphi \equiv \underbrace{\langle\alpha\rangle\langle\alpha\rangle\ldots}_{o(A)-times}\varphi.$$

where $o(A)$ is the ordinal corresponding to A. This way, since worms gives us a nice ordinal notation system for ordinals below ϵ_0, and positive language allows us to interpret modal formulas as theories, we can easily use them to denote transfinite levels in a progression.

New proof-theoretic facts about $\mathsf{KP}\omega$

Fernando Ferreira
Universidade de Lisboa

Abstract

The Σ_1-ordinal of $\mathsf{KP}\omega$ (Kripke-Platek set theory with infinity) is, by definition,

$$\min\{\alpha : L_\alpha \models \psi, \text{ for all } \Sigma_1\text{-sentences } \psi \text{ such that } \mathsf{KP}\omega \vdash \psi\}.$$

It is well-known that this is the Bachmann-Howard ordinal. We introduce a finite-order term language T_Ω with two ground types: N for the natural numbers and Ω for the countable construtive tree ordinals.

Let W the smallest set which contains 0 and is such that, whenever f is a function that maps ω into W, then $(1, f) \in W$. Each element a of W has a (set-theoretical) ordinal height $|a|$. Each closed term of T_Ω of type Ω denotes an element of W. Each closed term of type $\Omega \to \Omega$ denotes a function from W to W.

(a) The supremum of the ordinal heights of the (denotations of the) closed terms of T_Ω is the Σ_1-ordinal of $\mathsf{KP}\omega$. This is proved using a (bounded) functional interpretation.

(b) If $\mathsf{KP}\omega \vdash \forall x \exists y\, \phi(x, y)$, where ϕ is a bounded formula, then there is a closed term t of type $\Omega \to \Omega$ such that $\forall a \in W \forall x \in L_{|a|} \exists y \in L_{|t(a)|}\, \phi(x, y)$.

The above two results also hold for a second-order version $\mathsf{KP}\omega^2$ of $\mathsf{KP}\omega$ together with the schema of Δ_1-comprehension and of strict-Π^1_1 reflection. Moreover, this second-order theory is Σ_1-conservative over $\mathsf{KP}\omega$. It is an open question whether this conservativity result extends to Π_2-sentences.

References

[1] J. Avigad and H. Towsner. Functional interpretation and inductive definitions. *The Journal of Symbolic Logic*, 74:1100–1120, 2009.

[2] F. Ferreira. A new computation of the Σ-ordinal of $\mathsf{KP}\omega$. To appear in *The Journal of Symbolic Logic*.

[3] W. A. Howard. A system of abstract constructive ordinals. *The Journal of Symbolic Logic*, 37:355–374, 1972.

SMT and Theory Combination Techniques

Filipe Casal

SQIG, Instituto de Telecomunicações, Lisboa

Recently, the Automated Reasoning community has turned its attention to the area of Satisfiability Modulo Theories (SMT). Shortly, SMT deals with problems that are a generalization of boolean SAT problems, in the sense that we are dealing with first-order formulas and not just propositional formulas. This generalization is natural: SAT solving techniques are extensively studied and meaningful improvements to SAT solvers are extremely hard to develop, and SMT problems arise ever more frequently in fields such as Automated Theorem Proving and Software Verification.

Concretely, an SMT solver for a generic first-order theory generally consists of a Boolean Reasoner that breaks down the formula and finds high level inconsistencies (the formula $\varphi \wedge \psi \wedge \neg\varphi$ would be automatically ruled out, independently of whether φ or ψ are satisfiable) and a Theory Solver that verifies whether the formula is in fact satisfiable in the underlying theory. Essentially, the Boolean Reasoners are formula simplification mechanisms with a SAT solver, and the Theory Solvers are the decision procedures for decidable theories (usually Presburger arithmetic, arrays or bitvectors).

Suppose now we would like to formally verify an assertion that deals with both bitvectors as well as with arithmetic. This formula contains symbols from both theories, so the respective Theory Solvers would not be able to parse this formula. Here, we would like to modularly combine the decision procedures for these theories into a decision procedure for the union of these theories. This method of combination, the Nelson-Oppen method, requires the theories to satisfy many properties.

Since Nelson and Oppen introduced this combination procedure in 1979 [3], the study of the classes of theories which decision procedures can be combined has been actively studied. In 2005, it was shown that shiny [5] and polite [4] theories could be combined with an arbitrary theory. Later, a stronger notion of polite theory was proposed, see [2], in order to overcome a subtle issue with a proof in [4]. In [1], we analyse the relationship between shiny and strongly polite theories in the one-sorted case. We show that a shiny theory with a decidable quantifier-free satisfiability problem is strongly polite and provide two different sufficient conditions for a strongly polite theory to be shiny. Based on these results, we derive a combination method for the union of a polite theory with an arbitrary theory. Joint work with João Rasga, SQIG, Instituto de Telecomunicações and Departamento de Matemática, Instituto Superior Técnico, Universidade de Lisboa.

References

1. F. Casal and J. Rasga Revisiting the Equivalence of Shininess and Politeness. In *Proceedings of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'2013)*, volume 8312 of *LNCS*, pages 198–212, 2013.
2. D. Jovanović and C. Barrett. Polite theories revisited. In *Proceedings of the Seventeenth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'2010)*, volume 6397 of *LNCS*, pages 402–416, 2010.
3. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.
4. S. Ranise, C. Ringeissen, and C. G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In *Proceedings of the Fifth International Workshop on Frontiers of Combining Systems (FroCoS'2005)*, volume 3717 of *LNAI*, pages 48–64, 2005.
5. C. Tinelli and C. G. Zarba. Combining nonstably infinite theories. *Journal of Automated Reasoning*, 34(3):209–238, 2005.

Completeness of Peano arithmetic with the ω-rule

Jaime Gaspar[*]

12 January 2014

Gödel's first incompleteness theorem implies that Peano arithmetic is incomplete (there is a sentence that cannot be proved nor refuted). We take a fresh look at the following folklore result: if we add to Peano arithmetic the ω-rule

$$\frac{F(0) \quad F(1) \quad F(2) \quad \ldots}{\forall n\, F(n)}$$

(allowing to combine infinitely many proofs into a single infinite proof), then Peano arithmetic becomes complete (every sentence can be proved or refuted). Wee keep this talk short, simple and sweet.

[*]Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, `jaime.gaspar@urv.cat`. Centro de Matemática e Aplicações (CMA), FCT, UNL. Financially supported by the Martí Franquès Research Fellowship Programme grant number 2013PMF-PIPF-24 of the Universitat Rovira i Virgili.

A new interpretation of ID_1 in $\mathsf{ID}_1(\mathsf{W})$

João Enes
Universidade de Lisboa

Abstract

We consider the theories of (non-iterated) monotone inductive definitions ID_1 and $\mathsf{ID}_1(\mathsf{W})$. The first includes inductive definitions for every positive arithmetical operator, whereas the second is restricted to the inductive definition of the codes of well-founded recursive trees. We present a novel interpretation of the theory ID_1 in $\mathsf{ID}_1(\mathsf{W})$.

It is known from the work of Kreisel that these theories are proof-theoretic equivalent. However, unlike the interpretation of Kreisel, which is done via the intuitionistic conterparts $\mathsf{ID}_1^{\mathsf{i}}$ and $\mathsf{ID}_1^{\mathsf{i}}(\mathsf{W})$, our interpretation is direct from ID_1 to $\mathsf{ID}_1(\mathsf{W})$.

We would like to thank Fernando Ferreira for suggesting the possibility of the direct interpretation.

References

[1] S. Feferman and W. Sieg. Inductive definitions and subsystems of analysis. In *Iterated inductive definitions and subsystems of analysis: recent proof-theoretical studies, Lecture Notes in Mathematics,* 897: 16-77. Springer-Verlag, Berlin 1981.

[2] J. Enes. *Um estudo sobre teorias de definições indutivas e admissibilidade.* Master's thesis, Universidade de Lisboa, 2013.

Hybridize to Specify: institution-independent foundations for reconfigurable systems specification

Alexandre Madeira
INESC TEC (HASLab), UMinho

Renato Neves
INESC TEC (HASLab), UMinho

Manuel A. Martins
CIDMA, University of Aveiro

Luis S. Barbosa
INESC TEC (HASLab),UMinho

This talks aims to overview our recent works on the study and development of formal logics and semantics to specify *reconfigurable systems*, i.e., systems which behave differently in different modes of operation (often called *configurations*) and commute between them along their lifetime. In the suggested approach, models for reconfigurable software are structured transition systems described within appropriate logical systems. Their states corresponds to the individual configurations with whatever structure they have to bear in concrete applications. Transitions correspond to the admissible reconfigurations. This constitutes what we called the 'reconfigurations as transitions, configurations as local models' specification paradigm [4, 3]. Once chosen the semantics, the next issue concerns the definition of a suitable specification logic(s). Modal languages are the natural choice to talk about transition systems. Modal logic, however, is not expressive enough to deal with properties holding in specific states, a limitation which is overcome in hybrid logics [1] by considering a special set of symbols for naming states. Additionally, we need to specify the local configurations, at each state, as a model of a given (base) logic. The recent method for the hybridisation of institutions [6, 3] offer the source of logics for this specification. Concretely, it consists in a systematic method to extend arbitrary logics (formalized as institutions [2]) with hybrid logic features. Concretely, they are extended with Kripke semantics, for multi-modalities with arbitrary arities, as well as nominals and local satisfaction operators. The relevance of this generalisation step is in line with a basic engineering concern which recommends that the choice of a specification framework should depend on the nature of the requirements one has to deal with. For example, it may happen that, in a specific context, one would prefer to equip each local state with a partial algebra, a hidden algebra, a propositional valuation or even a hybrid logic model (since the method recurs).

On this talk we make an overview in the method, and we discuss a general construction of first-order encodings of hybridized institutions [6] as well as a suitable bisimulation notion for hybrid models [5].

References

[1] T. Braüner. *Hybrid Logic and its Proof-Theory*. Applied Logic Series. Springer, 2010.

[2] R. Diaconescu. *Institution-independent Model Theory*. Studies in Universal Logic. Birkhäuser Basel, 2008.

[3] A. Madeira. *Foundations and techniques for software reconfigurability*. PhD thesis, Universidades do Minho, Aveiro and Porto (Joint MAP-i Doctoral Programme), July 2013.

[4] A. Madeira, J. M. Faria, M. A. Martins, and L. S. Barbosa. Hybrid specification of reactive systems: An institutional approach. In G. Barthe, A. Pardo, and G. Schneider, editors, *Software Engineering and Formal Methods (SEFM 2011, Montevideo, Uruguay, November 14-18, 2011)*, volume 7041 of *Lecture Notes in Computer Science*, pages 269–285. Springer, 2011.

[5] A. Madeira, M. A. Martins, and L. S. Barbosa. Bisimilarity and refinement for hybrid(ised) logics. In J. Derrick, E. A. Boiten, and S. Reeves, editors, *Refine - Proceedings 16th International Refinement Workshop*, volume 115 of *Electronic Proceedings in Theoretical Computer Science*, pages 84–98, 2013.

[6] M. A. Martins, A. Madeira, R. Diaconescu, and L. S. Barbosa. Hybridization of institutions. In A. Corradini, B. Klin, and C. Cîrstea, editors, *Algebra and Coalgebra in Computer Science (CALCO 2011, Winchester, UK, August 30 - September 2, 2011)*, volume 6859 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2011.

Liquid Types Revisited

Mário Pereira, Sandra Alves, and Mário Florido

University of Porto, Department of Computer Science & LIACC,
R. do Campo Alegre 823, 4150-180, Porto, Portugal

Refinement types [2] state complex program's invariants, by simplying augmenting type systems with logical predicates. A refinement type of the form $\{\nu : B \mid \phi\}$ stands for the set of values from basic type B restricted to the filtering predicate (refinement) ϕ. However, the use of arbitrary boolean terms as refinement expressions leads to undecidable type systems, both for type checking and inference.

Liquid Types [4] (*Logically Qualified Data Types*) present a system capable of automatically infer refinement types, by means of two main restrictions to a general refinement type system: every refinement predicate is a conjunction of expressions exclusively taken from a global, user-supplied set of logical qualifiers (simple predicates over program variables, the value variable ν and the variable placeholder $\star$); and a conservative (ence decidable) notion of subtyping.

The Liquid Types system is defined as an extension to the Damas-Milner type system, with the term language extended with an `if-then-else` constructor and constants. A key idea behind this system is that the refinement type of every term is a refinement of the corresponding ML type.

We propose a refinement type system based on Liquid Types, with the addition of intersection types [1]. Our intersections are at the refinement expressions level only, i.e. for the type $\sigma \cap \tau$ both σ and τ are of the same form, solely differing in the refinement predicates. As an example, the identity function $id = \lambda x.x$ (considered to act only over integer values) could be typed within our system as $(x : \{\nu : int \mid \nu \geq 0\} \rightarrow \{\nu : int \mid \nu \geq 0\}) \cap (x : \{\nu : int \mid \nu \leq 0\} \rightarrow \{\nu : int \mid \nu \leq 0\})$. Our use of intersections for refinement types draws some inspiration from [3].

With our type system we are able to derive more precise types than in the original system, leading to a detailed description of programs' behaviour.

References

1. Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The journal of symbolic logic*, 48(4):931–940, 1983.
2. E. Denney. Refinement types for specification. In David Gries and Willem-Paul Roever, editors, *Programming Concepts and Methods PROCOMET '98*, IFIP — The International Federation for Information Processing. Springer US, 1998.
3. Tim Freeman and Frank Pfenning. Refinement types for ML. In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, PLDI '91, pages 268–277, New York, NY, USA, 1991. ACM.
4. Patrick M. Rondon, Ming Kawaguci, and Ranjit Jhala. Liquid types. In *Proceedings of the 2008 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '08, pages 159–169, New York, NY, USA, 2008. ACM.

Phridoni Alshibaia
Tbilisi State University-Georgia
palshibaia@yahoo.com
On Finitely Valued Bimodal Symmetric Gödel Logics

A "symmetric" formulation of intuitionistic propositional calculus Int^2, suggested by various authors (G. Moisil, A. Kuznetsov, C. Rauszer), presupposes that each of the connectives $\&, \vee, \rightharpoonup, \top, \bot$ has its dual $\vee, \&, \rightharpoondown, \bot, \top$, and the duality principle of the classical logic is restored. Gödel logic is the extension of intuitionistic logic by linearity axiom: $(p \to q) \vee (q \to p)$. Denote by G_n the n valued Gödel logic.

We investigate symmetric Gödel logic G_n^2, the language of which is enriched by two modalities $\Box_1$,$\Box_2$. The resulting system is named bimodal symmetric Gödel logic and is denoted by MG_n^2. MG_n^2-algebras represent algebraic models of the logic MG_n^2. The variety $\mathbf{MG_n^2}$ of all MG_n^2-algebras is generated by finite linearly ordered MG^2-algebras of finite height m, where $1 \leq m \leq n$. We focus on MG_n^2 algebras, which correspond to n valued MG_n^2 logic.

A description and characterization of m-generated free and projective MG^2-algebras in the variety $\mathbf{MG_n^2}$ is given.

FINITENESS AND RATIONAL DATATYPES, CONSTRUCTIVELY

TARMO UUSTALU

In constructive logic, finiteness of a set [or of a subset of a given set] can be defined in several inequivalent ways, and there is no obvious “right” definition. The situation is especially subtle, if equality on the set [or equality on the encompassing set] is not decidable [or the predicate defining the subset is not decidable]. Two of the most fundamental notions of finiteness are listability and Noetherianness, listability being generally stronger.

For a given a branching type, rational trees are by definition those non-wellfounded trees that have a finite number of distinct subtrees, extensional equality between non-wellfounded trees being given by bisimilarity. Since this definition refers to finiteness, different notions of finiteness could a priori lead to different notions of rationality.

In this talk, I explain the relationship between different notions of finiteness generally and in the special case of subsequences of a given sequence. I demonstrate that, for subsequences of a sequence, listability and Noetherianness are equivalent and exactly this equivalence leads to useful function definition and reasoning principles for rational sequences, including an inductive representation. Similar considerations apply to rational datatypes generally.

We are formalizing this development in the dependently typed programming language Agda.

This is ongoing joint work with James Chapman and Niccolò Veltri.

INSTITUTE OF CYBERNETICS AT TUT
AKADEEMIA TEE 21
12618 TALLINN, ESTONIA
E-mail: tarmo@cs.ioc.ee
URL: http://www.cs.ioc.ee/~tarmo