Domain Theory in Constructive and Predicative Univalent Foundations

Tom de Jong

Supervisor: Prof. Martín Hötzel Escardó
Co-supervisor: Dr. Benedikt Ahrens

External examiner: Prof. Andrej Bauer
Internal examiner: Dr. Vincent Rahli
Chair: Dr. Paul Blain Levy

School of Computer Science

UNIVERSITY OF BIRMINGHAM

PhD viva
20 December 2022

Introduction

- We develop domain theory in constructive and predicative univalent foundations.
- Our development is accompanied by a formalisation and illustrated by applications in the semantics of programming languages.

Domain theory

- Classic topic in theoretical computer science.
- Study of directed complete posets.
- Applications in programming language semantics, higher-type computability and topology.

Univalent foundations a.k.a. homotopy type theory

- Augments intensional Martin-Löf Type Theory with (some) higher inductive types and extensionality axioms, like function extensionality and univalence.
- Complementary uses:
  - internal language for $(\infty, 1)$-toposes,
  - synthetic homotopy type theory,
  - synthetic group theory,
  - sophisticated foundation for mathematics with implementations in proof assistants and an alternative to set theory.
- Constructive and predicative by default.

The univalent point of view

- The univalent point of view: types can be stratified according to the complexity of their identity types into (sub)singletons (propositions, truth values), sets, 1-groupoids, 2-groupoids, ...

  E.g. if a type X is equipped with a subsingleton-valued reflexive and antisymmetric relation, then X is a set, meaning its elements are identified in at most one way.

- The mathematical distinction between a property and (additional) data/structure is also internalised.

- The propositional truncation turns structure into property by reflecting a type to a subsingleton and plays a fundamental role in our development.

Constructivity in our work

- We do not assume excluded middle (or weaker variants such as Bishop's LPO) or the axiom of choice (or weaker variants like the axiom of countable choice).

  (Unique choice is a *theorem* of univalent foundations.)

- Hence, our development is valid in *every* $(\infty, 1)$-topos and not just those which validate classical logic.

- Working constructively means that we, for example,
  - work with a type of partial elements to capture nontermination in the Scott model of PCF, and
  - carefully distinguish between structure and property in our discussion of (structurally) continuous dcpos.

Constructivity in a wider context

- Martin-Löf invented his type theory to serve as a constructive foundation of mathematics.
- Univalent foundations has been given a computational interpretation through cubical type theory and this has been implemented in practice as Cubical Agda.
- Constructive proofs give rise to algorithms.
  E.g. the constructive proof that the Scott model of PCF is computationally adequate yields an *interpreter:* if we can prove that a program of base type is total, then computational adequacy computes its numerical outcome.
- A constructive treatment of domain theory could highlight and inform its effective/computational aspects.

Predicativity

Logic

- We do *not* asssume Voevodsky's resizing rules or axioms.
- Hence, powersets of small types are large.
- Propositions need not be small, and the type of small propositions is itself large.

Motivation

- It is an open problem whether propositional resizing axioms have a computational interpretation. Univalence was given a computational interpretation in cubical type theory.
- Resizing axioms fail in some interesting models of univalent type theory, such as Uemura's cubical assemblies model.
- Resizing axioms are expected to significantly increase the proof theoretic strength of the type theory.
- One may object to impredicativity on philosophical grounds.

Predicativity in domain theory and our work

In related work

Avoid size issues in predicative foundations by working with

- information systems, abstract bases or formal topologies rather than directed complete posets (dcpos), and
- approximable relations rather than Scott continuous functions.

Our approach

- We work *directly* with dcpos and Scott continuous functions.
- In dealing with size issues, we draw inspiration from category theory and make crucial use of type universes and equivalences to capture smallness.

An example of predicativity in our work

- Seeing a poset as a category: our dcpos are large, but locally small, and have small filtered colimits.
- E.g. in the Scott model of PCF, the dcpos
  - have carriers in the second universe $\mathcal{U}_1$,
  - least upper bounds for directed families indexed by types in the first universe $\mathcal{U}_0$, and
  - up to equivalence of types, the partial orders have values in $\mathcal{U}_0$.

Summarising our key contributions (1/6)

Distinguishing features of our approach

- Adopt homotopy type theory as our foundation and avoid setoids thanks to the univalent perspective.
- Commit to predicatively and constructively valid reasoning.
- Use type universes to avoid size issues involving large posets.
- Accompanied by an extensive formalisation of domain theory.

Summarising our key contributions (2/6)

Domain theory

- Allow for general universe parameters:
  $\mathcal{V}$-DCPO$_{\mathcal{U},\mathcal{T}}$ is the type of dcpos with
  - carriers in $\mathcal{U}$,
  - partial orders taking values in $\mathcal{T}$, and
  - directed suprema for families indexed by types in $\mathcal{V}$.
- Constructions of dcpos: products, exponentials and bilimits, carefully keeping track of universe parameters.
- Use the Escardó–Knapp lifting monad to give a constructive treatment of the free dcpo with a least element on a set as the type of partial elements.

Summarising our key contributions (3/6)

Applications in the semantics of programming languages

- Perhaps surprisingly, many complex constructions of dcpos are predicatively possible without needing ever-increasing universes.

- The Scott model of the programming language PCF

$$[\![-]\!] : \text{PCF types} \to \mathcal{U}_0\text{-DCPO}_{\mathcal{U}_1,\mathcal{U}_1}$$

  is sound and computationally adequate:
  For every PCF program t of the base type and $n : \mathbb{N}$,

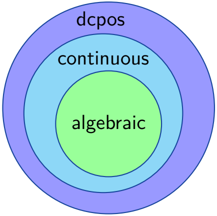
$$[\![t]\!] = [\![\underline{n}]\!] \iff t \text{ reduces to } \underline{n}.$$

- Scott's famous D_∞ model of the untyped λ-calculus can be constructed predicatively as an element of $\mathcal{U}_0\text{-DCPO}_{\mathcal{U}_1,\mathcal{U}_1}$.

Summarising our key contributions (4/6)

Continuous and algebraic dcpos

- Classically, a dcpo is continuous if every element is the *directed* supremum of the elements way below it, and algebraic if we can further restrict to compact elements.



- We give a predicatively adequate account of continuous and algebraic dcpos following an approach by Johnstone and Joyal to continuous categories.

- We investigate issues related to the axiom of choice and the distinction between property and structure.

Summarising our key contributions (5/6)

Dcpos with small (compact) bases

- We give predicative size-aware adaptations of the notions of a basis and the rounded ideal completion.
- If a dcpo has a small basis, then it is continuous.
  The converse is false in general, but our main examples of continuous dcpos *do* have small bases.
- For example,
  - the *small* type of Booleans is a small compact basis for the large type of propositions, and
  - the *small* type of lists on a small set X is a small compact basis for the large powerset of X.
- Dcpos with small bases are well-behaved predicatively:
  - They are locally small and so are their exponentials.
  - They can be presented by round ideals.

Summarising our key contributions (6/6)

No-go theorems in our predicative and constructive setting

- The fact that nontrivial dcpos have large carriers is in fact unavoidable and characteristic of our predicative setting.

  So the carriers of the dcpos of the Scott model of PCF can live only in the lowest universe $\mathcal{U}_0$ if we work impredicatively.

- Moreover, nontrivial dcpos necessarily lack decidable equality.

Conclusion

- Domain theory is *necessarily* more involved in a constructive and predicative setting.
- We completed a comprehensive and formalised account of it in univalent foundations, including applications in the semantics of programming languages.
- This helps to cement the status of homotopy type theory as a modern and suitable foundation for mathematics.

Publications

[1] Tom de Jong. "The Scott model of PCF in univalent type theory". In: *Mathematical Structures in Computer Science* 31.10 (2019): *Homotopy Type Theory 2019*, pp. 1270–1300. DOI: 10.1017/S0960129521000153.

[2] Tom de Jong and Martín Hötzel Escardó. "Domain Theory in Constructive and Predicative Univalent Foundations". In: *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*. Ed. by Christel Baier and Jean Goubault-Larrecq. Vol. 183. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, 28:1–28:18. DOI: 10.4230/LIPIcs.CSL.2021.28. Expanded version with full proofs available on arXiv: 2008.01422 [math.LO].

[3] Tom de Jong and Martín Hötzel Escardó. "On Small Types in Univalent Foundations". Sept. 2022. arXiv: 2111.00482 [cs.LO]. Revised and expanded version of [4]. Accepted pending minor revision to a special issue of *Logical Methods in Computer Science* on selected papers from *FSCD 2021*.

[4] Tom de Jong and Martín Hötzel Escardó. "Predicative Aspects of Order Theory in Univalent Foundations". In: *6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021)*. Ed. by Naoki Kobayashi. Vol. 195. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, 8:1–8:18. DOI: 10.4230/LIPIcs.FSCD.2021.8.

The last publication won the *Best Paper by a Junior Researcher* award.