Constructive domain theory in Univalent Foundations

Tom de Jong

University of Birmingham, United Kingdom

8 April, 2020
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
</tbody>
</table>

- Introduction
- Technical background
  - Univalent Foundations
  - Subsingletons and sets
  - Propositional truncation
  - Univalence
  - Constructivity and predicativity
- Domain theory (classically)
- Our work
  - Predicative dcpos in UF
  - Scott model of PCF
- Conclusion and current work
Outline

1 Introduction

2 Technical background
   - Univalent Foundations
     - Subsingletons and sets
     - Propositional truncation
     - Univalence
   - Constructivity and predicativity
   - Domain theory (classically)
Outline

1 Introduction

2 Technical background
   - Univalent Foundations
   - Subsingletons and sets
   - Propositional truncation
   - Univalence
   - Constructivity and predicativity
   - Domain theory (classically)

3 Our work
   - Predicative dcpos in UF
   - Scott model of PCF
## Outline

1. **Introduction**
2. **Technical background**
   - Univalent Foundations
     - Subsingletons and sets
     - Propositional truncation
     - Univalence
   - Constructivity and predicativity
   - Domain theory (classically)
3. **Our work**
   - Predicative dcpos in UF
   - Scott model of PCF
4. **Conclusion and current work**
Our aim and motivation

Develop domain theory, but constructively and predicatively in Univalent Foundations.
Our aim and motivation

Develop domain theory, but constructively and predicatively in Univalent Foundations.

Why domain theory?
- Classical topic in theoretical computer science
- Applications in:
  - semantics of programming languages
  - topology
Our aim and motivation

Develop domain theory, but constructively and predicatively in Univalent Foundations.

Why domain theory?
- Classical topic in theoretical computer science
- Applications in:
  - semantics of programming languages
  - topology

Why constructively and predicatively?
- More general
- Relevance in:
  - computer science (algorithm extraction)
  - pointfree/formal topology
- No constructive justification of impredicativity axioms (yet)
Our aim and motivation

Develop domain theory, but constructively and predicatively in Univalent Foundations.

Why Univalent Foundations?

- Implemented in proof assistants
- Constructive and predicative by default
- Novel and natural interpretation of mathematical equality

We could also extend our foundations with more higher inductive types, but so far, we haven’t had any need for it.

By developing domain theory constructively in UF, we have also improved our understanding of the foundations themselves.

Further, domain theory serves as a testing ground for (formalisation in) UF.
# Outline

## 1 Introduction

## 2 Technical background
- Univalent Foundations
  - Subsingletons and sets
  - Propositional truncation
  - Univalence
- Constructivity and predicativity
- Domain theory (classically)

## 3 Our work
- Predicative dcpos in UF
- Scott model of PCF

## 4 Conclusion and current work
Univalent Foundations

Intensional Martin-Löf Type Theory with:

- extensionality axioms
- propositional truncation

Vladimir Voevodsky

I will assume some familiarity with dependent type theory, e.g. \(\Pi, \Sigma, \top\)-types.

Specifically, we need function extensionality (pointwise equal functions are equal) and propositional extensionality (logically equivalent propositions are equivalent) (and sometimes, univalence).

I will explain the propositional truncation shortly.
Introduction

Technical background

Our work

Conclusion and current work

Univalent Foundations

Univalent Foundations

Intensional Martin-Löf Type Theory with:
- extensionality axioms
- propositional truncation

Vladimir Voevodsky

Notation:
- For $x, y : X$, write $x = y$ for $\text{Id}_X(x, y)$.
- Use $\equiv$ for judgemental equality.

I will assume some familiarity with dependent type theory, e.g. $\Pi, \Sigma, +$-types.

Specifically, we need function extensionality (pointwise equal functions are equal) and propositional extensionality (logically equivalent propositions are equivalent) (and sometimes, univalence).

I will explain the propositional truncation shortly.
Subsingletons and sets

Definition
A type $X$ is a **subsingleton** (or **proposition**) if we have an element of

$$\text{is-a-prop}(X) \equiv \prod_{x:X} \prod_{y:X} x = y.$$
Subsingletons and sets

**Definition**
A type $X$ is a subsingleton (or proposition) if we have an element of

$$\text{is-a-prop}(X) \equiv \prod_{x:X} \prod_{y:X} x = y.$$ 

**Definition**
A type $X$ is a set if we have an element of

$$\text{is-a-set}(X) \equiv \prod_{x:X} \prod_{y:X} \text{is-a-prop}(x = y).$$

There is a stratification of types in terms of the complexity of their identity types: Voevodsky’s hlevels or truncation levels.

For this talk, we only need to consider two hlevels: the subsingletons and sets.

In a subsingleton, all elements are identified/equal. There is at most one element (up to $=$).

In a set, elements are identified/equal in at most one way.
Propositional truncation

For every type $X$, there is a proposition $\|X\|$ and a map $X \to \|X\|$, such that every map from $X$ to a proposition $P$ factors through it.

$$
\begin{array}{c}
X \\
\|X\|
\end{array} \xrightarrow{\text{dashed map}} P
$$

Borrowing terminology from category theory, we might call propositional truncation subsingleton reflection.

The dashed map is necessarily unique, because of function extensionality and the fact that $P$ is a subsingleton.

The propositional truncation does not erase witnesses. (For instance: if $A$ is a decidable predicate (i.e. proposition-valued family) on $\mathbb{N}$, then we have maps:

$$\sum_{n: \mathbb{N}} A(n) \to \sum_{k: \mathbb{N}} \{k \text{ is the least } n: \mathbb{N} \text{ such that } A(n) \text{ holds} \} \to \sum_{n: \mathbb{N}} A(n),$$

where the first map exists, because the second type may be shown to be a proposition and because $A$ is decidable.)
What about the univalence axiom?

- The **univalence axiom** is an extensionality axiom for type universes.
- It implies function and propositional extensionality.
- Univalent Foundations is about much more than the univalence axiom!

We usually do not need full univalence, because the types under consideration are all propositions and sets (dcpos).

Arguably, univalent type theory is much more about the concept of truncation levels than about the univalence axiom.
Outline

1 Introduction

2 Technical background
   - Univalent Foundations
     - Subsingletons and sets
     - Propositional truncation
     - Univalence
   - Constructivity and predicativity
   - Domain theory (classically)

3 Our work
   - Predicative dcpos in UF
   - Scott model of PCF

4 Conclusion and current work
Constructivity

Definition

*Excluded middle (EM)* in UF: $P + \neg P$ for all *propositions* $P$. 
Constructivity

Definition

Excluded middle (EM) in UF: $P + \neg P$ for all propositions $P$.

Definition

Bishop’s Limited Principle of Omniscience (LPO):

$$\prod_{\alpha: \mathbb{N} \to 2} \left( \left( \prod_{n: \mathbb{N}} \alpha(n) = 0 \right) + \left( \sum_{k: \mathbb{N}} k \text{ is least with } \alpha(k) = 1 \right) \right).$$
Constructivity

Definition

**Excluded middle (EM) in UF:** $P + \neg P$ for all propositions $P$.

Definition

Bishop’s **Limited Principle of Omniscience (LPO):**

$$\prod_{\alpha : \mathbb{N} \to 2} \left( \left( \prod_{n : \mathbb{N}} \alpha(n) = 0 \right) + \left( \sum_{k : \mathbb{N}} k \text{ is least with } \alpha(k) = 1 \right) \right).$$

- EM implies LPO.
- LPO and EM are **constructive taboos:** they cannot be proved or disproved constructively.
Constructivity and predicativity

Predicativity in Univalent Foundations

Impredicativity
The type of propositions in a universe \( \mathcal{U} \)

\[ \Omega_{\mathcal{U}} \equiv \sum_{P : \mathcal{U}} \text{is-a-prop}(P) \]

is (essentially) small, i.e. has an (equivalent) copy in \( \mathcal{U} \).

Here \( \simeq \) refers to Voevodsky’s notion of (type) equivalence.
Predicativity in Univalent Foundations

**Impredicativity**

The type of propositions in a universe $\mathcal{U}$

$$\Omega_\mathcal{U} \equiv \sum_{P : \mathcal{U}} \text{is-a-prop}(P)$$

is (essentially) small, i.e. has an (equivalent) copy in $\mathcal{U}$.

**Theorem**

EM implies Impredicativity.

**Proof.**

With EM, there are only two propositions: 0 and 1, so $\Omega_\mathcal{U} \simeq 2 : \mathcal{U}$. □
# Outline

1. **Introduction**

2. **Technical background**
   - Univalent Foundations
   - Subsingletons and sets
   - Propositional truncation
   - Univalence
   - Constructivity and predicativity
   - Domain theory (classically)

3. **Our work**
   - Predicative dcpos in UF
   - Scott model of PCF

4. **Conclusion and current work**
Domain theory was pioneered by Dana Scott [Sco72; Sco93] and developed further by many others: Plotkin [Plo83], Lawson, Keimel, Abramsky, Jung [AJ94], Simpson and Escardó, just to name a few.

Order theory studies partially ordered sets (posets).
Basic objects in domain theory

Definition

A poset \((P, \leq)\) is direc\(\text{t}ed\) if it is non-empty and for every \(x, y \in P\), there exists some \(z \in P\) such that \(x \leq z\) and \(y \leq z\).

For some (computational) intuition: think of a directed set as a set of approximations (or computations). Given two approximations, we can find a better one.
Basic objects in domain theory

Definition

A poset \((P, \leq)\) is directed if it is non-empty and for every \(x, y \in P\), there exists some \(z \in P\) such that \(x \leq z\) and \(y \leq z\).

For some (computational) intuition: think of a directed set as a set of approximations (or computations). Given two approximations, we can find a better one.
**Basic objects in domain theory**

**Definition**

A poset \((P, \leq)\) is **directed** if it is non-empty and for every \(x, y \in P\), there exists some \(z \in P\) such that \(x \leq z\) and \(y \leq z\).

**Definition**

A **directed complete poset (dcpo)** is a poset \((P, \leq)\) such that every directed subset of \(P\) has a least upper bound in \(P\).

For some (computational) intuition: think of a directed set as a set of approximations (or computations). Given two approximations, we can find a better one.

In a dcpo, we require that all approximations converge to a value (the least upper bound).
Basic objects in domain theory

Definition

A directed complete poset (dcpo) is a poset \((P, \leq)\) such that every directed subset of \(P\) has a least upper bound in \(P\).

For some (computational) intuition: think of a directed set as a set of approximations (or computations). Given two approximations, we can find a better one.

In a dcpo, we require that all approximations converge to a value (the least upper bound).
Outline

1 Introduction

2 Technical background
   - Univalent Foundations
     - Subsingletons and sets
     - Propositional truncation
     - Univalence
   - Constructivity and predicativity
   - Domain theory (classically)

3 Our work
   - Predicative dcpos in UF
   - Scott model of PCF

4 Conclusion and current work
Predicative dcpos in UF

For predicativity reasons, we use families rather than subsets.

Definition
Let $(P, \leq)$ be a poset. A family $u : I \to P$ is directed if $\|I\|$ and
\[ \prod_{i,j : I} \| \sum_{k : I} u_i \leq u_k \times u_j \leq u_k \| . \]

Note the use of the propositional truncation.

We use the propositional truncation here:
- to ensure that being directed is property (rather than structure);
- because for $i, j : I$, there might be many $k : I$ with $u_i \leq u_k \times u_j \leq u_k$ and we don’t mean to specify a choice.

Similarly, asking for an element of $I$ (rather than $\|I\|$) would be asking for a pointed (rather than an inhabited) type.
Predicative dcpos in UF

For predicativity reasons, we use families rather than subsets.

**Definition**

Let $(P, \leq)$ be a poset. A family $u : I \to P$ is **directed** if $\|I\|$ and

$$\Pi_{i,j: I} \sum_{k: I} u_i \leq u_k \times u_j \leq u_k \|.$$

Note the use of the propositional truncation.

Fix a universe $\mathcal{V}$ of “small” types.

**Definition**

A $\mathcal{V}$-dcpo is a poset $(P, \leq)$ such that every directed family $I \to P$ with $I$ small has a least upper bound in $(P, \leq)$.

We use the propositional truncation here:

- to ensure that being directed is **property** (rather than structure);
- because for $i, j : I$, there might be many $k : I$ with $u_i \leq u_k \times u_j \leq u_k$ and we don’t mean to specify a choice.

Similarly, asking for an element of $I$ (rather than $\|I\|$) would be asking for a **pointed** (rather than an inhabited) type.

In a predicative framework, we must be careful about size, which is why we only ask that directed families indexed by types in a fixed universe have least upper bounds.
Scott model of PCF

- **PCF**: typed programming language with a **fixed point combinator** for general recursion. **PCF types**:
  - type \( \iota \) for natural numbers
  - function types
Scott model of PCF

- **PCF**: typed programming language with a **fixed point combinator** for general recursion. PCF types:
  - type \( \iota \) for natural numbers
  - function types
- **Scott model of PCF**: interpret PCF types as dcpos with a least element that represents **non-termination**.

Because of the fixed point combinator, a “standard” set-theoretic interpretation will not work (i.e. one where function types are interpreted as exponentials in Set).

A map between dcpos (with bottom) is continuous if it preserves directed suprema. The point is that such maps have fixed points. The continuous maps between two dcpos with bottom form another dcpo with bottom with the pointwise ordering. This allows us to interpret the function types of PCF.
How to represent the type of natural numbers?

Classically:

```
0  1  2  3  ...
```

But, 

\((N + \{\bot\})\text{ with this order} \Rightarrow \text{LPO.}\)
How to represent the type of natural numbers?

Classically:

\[ 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots \]

But, \((\mathbb{N} + \{\bot\} \text{ with this order})\) is a dcpo \(\Rightarrow\) LPO.

Recall that LPO is:

\[
\prod_{\alpha : \mathbb{N} \rightarrow 2} \left( \left( \prod_{n : \mathbb{N}} \alpha(n) = 0 \right) + \left( \sum_{k : \mathbb{N}} k \text{ is least with } \alpha(k) = 1 \right) \right).
\]

Proof of the implication: given \(\alpha : \mathbb{N} \rightarrow 2\), define \(\beta : \mathbb{N} \rightarrow \mathbb{N} + 1\) by:

\[
\beta(n) \equiv \begin{cases} 
\text{inl}(k) & \text{if } k \text{ is the least number } \leq n \text{ such that } \alpha(k) = 1; \\
\text{inr}(\star) & \text{else}.
\end{cases}
\]

Then \(\beta\) is directed and therefore, if \(\mathbb{N} + 1\) is directed complete, has a least upper bound \(s\).

But we can decide if \(s = \text{inl}(k)\) for some \(k : \mathbb{N}\) or if \(s = \text{inr}(\star)\). But the former implies \(\alpha(k) = 1\), while the latter implies \(\prod_{n : \mathbb{N}} \alpha(n) = 0\).
How to represent the type of natural numbers?

Classically:

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
& & & \ldots \\
\downarrow & & & \\
\bot & & & \\
\end{array}
\]

But,

\((\mathbb{N} + \{\bot\} \text{ with this order}) \text{ is a dcpo} \Rightarrow \text{LPO.}\)

So constructively, this is no good.
Lifting

Definition

The *lifting* of a type $X$ is: $\mathcal{L}(X) \equiv \sum_{P:\Omega} (P \to X)$. 

Theorem (Knapp, Escardó)

$\mathcal{L}$ is monad (on sets) with unit $\eta$ (modulo size).

There is a distinguished element: $\bot_X : \equiv (0, \text{from-empty } X) : \mathcal{L}(X)$.
Lifting

Definition

The lifting of a type $X$ is: $\mathcal{L}(X) \equiv \sum_{P:\Omega}(P \to X)$.

Definition

We can embed a type into its lifting:

$$\eta_X : X \to \mathcal{L}(X)$$

$x \mapsto (1, \lambda(u : 1).x)$
### Lifting

The **lifting** of a type $X$ is: $\mathcal{L}(X) \equiv \sum_{P:Ω}(P \to X)$.

**Definition**

We can embed a type into its lifting:

$$\eta_X : X \to \mathcal{L}(X)$$

$$x \mapsto (1, \lambda (u : 1).x)$$

**Theorem (Knapp, Escardó)**

$\mathcal{L}$ is monad (on sets) with unit $\eta$ (modulo size).

Note that $\mathcal{L}$ (potentially) raises universe levels, so that it is a “monad across universes”. Moreover, for types that are not sets, this would be some kind $\infty$-monad, because it is missing coherence conditions.
Lifting

Definition

The \textit{lifting} of a type \( X \) is: \( \mathcal{L}(X) \equiv \sum_{P:\Omega} (P \to X) \).

Definition

We can embed a type into its lifting:

\[ \eta_X : X \to \mathcal{L}(X) \]
\[ x \mapsto (1, \lambda (u : 1).x) \]

Theorem (Knapp, Escardó)

\( \mathcal{L} \) \textit{is monad (on sets) with unit} \( \eta \) \textit{(modulo size)}. \( \mathcal{L} \)

There is a distinguished element: \( \bot_X \equiv (0, \text{from-empty}_X) : \mathcal{L}(X) \).

Note that \( \mathcal{L} \) (potentially) raises universe levels, so that it is a “monad across universes”. Moreover, for types that are not sets, this would be some kind \( \infty \)-monad, because it is missing coherence conditions.

With Excluded Middle, this is all, i.e. \( \mathcal{L}(X) \simeq X + 1 \).
Definition

The lifting of a type $X$ is: $\mathcal{L}(X) \equiv \sum_{P:Ω}(P \to X)$.

Definition

Let is-defined : $\mathcal{L}(X) \to Ω$ be: $(P, ϕ) = P$. 
The lifting of a type $X$ is: $L(X) \equiv \sum_{P: \Omega} (P \rightarrow X)$.

Let is-defined $: L(X) \rightarrow \Omega$ be: $(P, \varphi) = P$.

Define a partial order $\sqsubseteq$ on $L(X)$ by:

$$l \sqsubseteq m :\equiv \text{is-defined}(l) \rightarrow l = m.$$
**Definition**

The *lifting* of a type $X$ is: $\mathcal{L}(X) \equiv \sum_{P:\Omega} (P \rightarrow X)$.

**Definition**

Let $\text{is-defined} : \mathcal{L}(X) \rightarrow \Omega$ be: $(P, \varphi) = P$.

**Definition**

Define a partial order $\sqsubseteq$ on $\mathcal{L}(X)$ by:

$$l \sqsubseteq m \equiv \text{is-defined}(l) \rightarrow l = m.$$ 

**Theorem (Knapp, Escardó)**

The pair $(\mathcal{L}(X), \sqsubseteq)$ is a dcpo if $X$ is a set.
Soundness and computational adequacy

Using:
- \((L(N), \sqsubseteq)\) to interpret the PCF type of natural numbers
- the monad structure on \(L\)

we can define the Scott model of PCF
Soundness and computational adequacy

Using:

- \((\mathcal{L}(\mathbb{N}), \sqsubseteq)\) to interpret the PCF type of natural numbers
- the monad structure on \(\mathcal{L}\)

we can define the Scott model of PCF and prove:

- **soundness**: if a PCF program \(s\) computes to a term \(t\), then \(s\) and \(t\) are equal in the model;
Soundness and computational adequacy

Using:
- \((\mathcal{L}(\mathbb{N}), \sqsubseteq)\) to interpret the PCF type of natural numbers
- the monad structure on \(\mathcal{L}\)

we can define the Scott model of PCF and prove:
- **soundness**: if a PCF program \(s\) computes to a term \(t\), then \(s\) and \(t\) are equal in the model;
- **computational adequacy**: if a PCF program \(t\) is equal to \(\eta(n)\) with \(n : \mathbb{N}\), then \(t\) computes to the term \(\underline{n}\) (that represents \(n\) in PCF).

What is especially nice about having a constructive proof of computational adequacy is that it allows us run a PCF program once we prove that it is total, cf. [19, end of Section 7].
Conclusion and current work

Conclusion
Constructive and predicative domain theory in Univalent Foundations

- soundness and computational adequacy of Scott model of PCF using lifting monad
- important use of propositional truncation
- formalised in Agda (some in Coq/UniMath)
Conclusion and current work

Conclusion
Constructive and predicative domain theory in Univalent Foundations

- soundness and computational adequacy of Scott model of PCF using lifting monad
- important use of propositional truncation
- formalised in Agda (some in Coq/UniMath)

Current work
✓ bases of dcpos & continuous and algebraic dcpos
✓ formalise Scott's $D_\infty$
- exponentials for continuous dcpos (e.g. SFP domains)
- (predicative version of) Pataraia's fixed point theorem
Conclusion and current work

**Conclusion**

Constructive and predicative domain theory in Univalent Foundations

- **soundness** and **computational adequacy** of Scott model of PCF using lifting monad
- important use of **propositional truncation**
- formalised in Agda (some in Coq/UniMath)

**Current work**

- ✓ bases of dcpos & continuous and algebraic dcpos
- ✓ formalise Scott's $D_\infty$
- exponentials for continuous dcpos (e.g. SFP domains)
- (predicative version of) Pataraia's fixed point theorem

---


Bases for dcpos

- A dcpo is **continuous** if it has a **basis** that “generates” the whole dcpo.
- Predicatively, we need to strengthen the notion of basis.

In our predicative framework, given a dcpo $D$, we say that $\beta : B \to D$ is a **basis** if, in addition to the usual axioms of a basis, $B$ is *small* and the way-below/approximation relation of $D$ is *small* when restricted to elements of the form $\beta(b)$. 
Bases for dcpo’s

- A dcpo is \textit{continuous} if it has a \textit{basis} that “generates” the whole dcpo.
- Predicatively, we need to strengthen the notion of basis.

Examples:
- $\mathcal{L}(X)$ has a very simple basis: $X + 1$.

In our predicative framework, given a dcpo $D$, we say that $\beta : B \to D$ is a \textit{basis} if, in addition to the usual axioms of a basis, $B$ is \textit{small} and the way-below/approximation relation of $D$ is \textit{small} when restricted to elements of the form $\beta(b)$.
Bases for dcpo

A dcpo is *continuous* if it has a *basis* that “generates” the whole dcpo.

Predicatively, we need to strengthen the notion of basis.

Examples:

- \( \mathcal{L}(X) \) has a very simple basis: \( X + 1 \).
- \( \mathcal{P}(X) \) has the Kuratowski finite subsets of \( X \) as a basis.

In our predicative framework, given a dcpo \( D \), we say that \( \beta : B \to D \) is a *basis* if, in addition to the usual axioms of a basis, \( B \) is *small* and the way-below/approximation relation of \( D \) is *small* when restricted to elements of the form \( \beta(b) \).
References I


References II


References III


References IV


References V


