Initial Algebras of Domains via Quotient Inductive-Inductive Types

Simcha van Collem, Niels van der Weide, Herman Geuvers

20 June 2025

Radboud University Nijmegen

Programming language semantics

- Domain theory is used to give semantics to programming languages
- Example
    - Fixpoints for general recursion
    - Partial functions
    - Lazy evaluation
    - Nondeterminism

What about arbitrary algebraic effects?

- Algebraic effects have
  - Algebraic operations
  - An inequational theory
- How can we construct them in domain theory?

What about arbitrary algebraic effects?

- Algebraic effects have
  - Algebraic operations
  - An inequational theory
- How can we construct them in domain theory?

Solution: construct them as initial domain algebras

These domain algebras should be

- A domain
- Have the operations
- Obey the inequational theory

Contributions

- Framework for
  - describing algebraic effects
  - constructing them as initial algebras via a QIIT
- Examples:
  - Partiality[1]
  - Powerdomain
  - Coalesced sum, Smash products, Coequalizer, ...
- Formalized in Cubical Agda

[1] Altenkirch, Danielsson, and Kraus, "Partiality, Revisited - The Partiality Monad as a Quotient Inductive-Inductive Type".

Directed Complete Partial Order (DCPO)

- DCPO: set D with information ordering $\sqsubseteq$
- Smaller elements contain less information
- Every directed family $\alpha : I \to D$ has a supremum $\bigsqcup_{i:I} \alpha(i)$

Directed Complete Partial Order (DCPO)

- DCPO: set D with information ordering $\sqsubseteq$
- Smaller elements contain less information
- Every directed family $\alpha : I \to D$ has a supremum $\bigsqcup_{i:I} \alpha(i)$
    - Generalizes ω-CPO, where every increasing sequence $\alpha_0 \sqsubseteq \alpha_1 \sqsubseteq \ldots$ has a supremum
    - The elements in the family are consistently moving upward towards more information

Partiality

- Programs may not terminate
- So a program has either no outcome or a specified result
- Given DCPO D, we want a DCPO $D_\perp$, such that
    - For each result d in D, we have the same result in $D_\perp$; written as $\eta(d)$
    - We have a result for non-termination: $\perp$ in $D_\perp$
    - Non-termination gives the least information

Summary:

Two operations:

- $\eta : D \to D_\perp$
- $\perp : D_\perp$

One inequality:

- $\perp \sqsubseteq x$

- Non-termination gives the least information

Nondeterminism

- Nondeterministic programs could have multiple results
- So a program has a set of possible results
- Given a DCPO D, we want a DCPO $\mathcal{P}(D)$ of possible results, such that
  - For each result d in D, we have the result set with a unique result $\{d\}$ in $\mathcal{P}(D)$
  - Given result sets s_1, s_2 in $\mathcal{P}(D)$, we can combine their results as $s_1 \cup s_2$ in $\mathcal{P}(D)$
  - The union operation is
    - commutative and associative: order of combining does not matter
    - idempotent: combining a result set with itself, adds nothing

Nondeterminism

- Nondeterministic programs could have multiple results

Summary:

Two operations:

- $\{-\} : D \to \mathcal{P}(D)$
- $\cup : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{P}(D)$

Four inequalities:

- $x \cup y \sqsubseteq y \cup x$
- $(x \cup y) \cup z \sqsubseteq x \cup (y \cup z)$
- $x \cup x \sqsubseteq x$
- $x \sqsubseteq x \cup x$

lts,

e

ts

ng

Algebras

We have

- A DCPO
- with Scott continuous operations
- operations of the form $X^B \times C \to X$
- respecting certain inequalities

$\Big\}$ Signature Σ

We call this a Σ-algebra

- morphisms are Scott continuous maps commuting with the operations
- this forms a category Σ-Alg

Goal:

- construct the initial Σ-algebra using a QIIT

Algebras

We have

- A DCPO

Scott continuity:

Operations should be monotone and send suprema to suprema, *e.g.*

$$\left\{\bigsqcup_{i:I} \alpha(i)\right\} = \bigsqcup_{i:I} \{\alpha(i)\}$$

We call this a Σ-algebra

- morphisms are Scott continuous maps commuting with the operations
- this forms a category Σ-Alg

Goal:

- construct the initial Σ-algebra using a QIIT

Algebras

We have

- A DCPO
- with Scott continuous operations
- operations of the form $X^B \times C \to X$
- respecting certain inequalities

} Signature Σ

We call this a Σ-algebra

- morphisms are Scott continuous maps commuting with the operations
- this forms a category Σ-Alg

Goal:

- construct the initial Σ-algebra using a QIIT

Quotient Inductive-Inductive Type (QIIT)

- Combination of QIT and IIT:
  - Define a type by specifying constructors and equations
  - Define a type A and a type family on A simultaneously
- Approach used by Altenkirch at al.[2] to construct the partiality effect
- We do it for arbitrary signature Σ

[2] Altenkirch, Danielsson, and Kraus, "Partiality, Revisited - The Partiality Monad as a Quotient Inductive-Inductive Type".

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq: \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT

We define $\mathcal{P}(D) : \mathcal{U}$ and $\sqsubseteq : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{U}$ simultaneously via a QIIT as follows:

- We have constructors expressing that $\sqsubseteq$ is a preorder
- We add a path constructor to make it a partial order
- We add constructors to make it directed complete
- Each operation gives rise to a constructor of $\mathcal{P}(D)$
- We have a path constructor to express the continuity of each operation
- Each inequality gives rise to a constructor for $\sqsubseteq$

Powerdomain QIIT (DCPO constructors)

$$\frac{}{x \sqsubseteq x} \qquad \frac{x \sqsubseteq y \qquad y \sqsubseteq z}{x \sqsubseteq z} \qquad \frac{}{\mathsf{isProp}(x \sqsubseteq y)}$$

$$\frac{x \sqsubseteq y \qquad y \sqsubseteq x}{x = y} \qquad \frac{}{\mathsf{isSet}(\mathcal{P}(D))}$$

Powerdomain QIIT (DCPO constructors)

$$\frac{}{x \sqsubseteq x} \qquad \frac{x \sqsubseteq y \quad y \sqsubseteq z}{x \sqsubseteq z} \qquad \frac{}{\mathsf{isProp}(x \sqsubseteq y)}$$

$$\frac{x \sqsubseteq y \quad y \sqsubseteq x}{x = y} \qquad \frac{}{\mathsf{isSet}(\mathcal{P}(D))}$$

$$\bigsqcup : \prod_{\alpha : I \to \mathcal{P}(D)} \mathsf{isDirected}(\alpha) \to \mathcal{P}(D)$$

Powerdomain QIIT (DCPO constructors)

$$\frac{}{x \sqsubseteq x} \qquad \frac{x \sqsubseteq y \qquad y \sqsubseteq z}{x \sqsubseteq z} \qquad \frac{}{\mathrm{isProp}(x \sqsubseteq y)}$$

$$\frac{x \sqsubseteq y \qquad y \sqsubseteq x}{x = y} \qquad \frac{}{\mathrm{isSet}(\mathcal{P}(D))}$$

$$\bigsqcup : \prod_{\alpha : I \to \mathcal{P}(D)} \mathrm{isDirected}(\alpha) \to \mathcal{P}(D)$$

$$\frac{\alpha : I \to \mathcal{P}(D) \qquad \delta : \mathrm{isDirected}(\alpha)}{\prod_{i:I} \alpha(i) \sqsubseteq \bigsqcup_{i:I} \alpha(i)}$$

$$\frac{\alpha : I \to \mathcal{P}(D) \qquad \delta : \mathrm{isDirected}(\alpha)}{\prod_{v:\mathcal{P}(D)} \mathrm{isUpperbound}(v, \alpha) \to \bigsqcup_{i:I} \alpha(i) \sqsubseteq v}$$

Powerdomain QIIT (Operations)

$$\{-\} : D \to \mathcal{P}(D) \qquad \cup : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{P}(D)$$

Powerdomain QIIT (Operations)

$$\{-\} : D \to \mathcal{P}(D) \qquad\qquad \cup : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{P}(D)$$

$$\frac{\alpha_1, \alpha_2 : I \to \mathcal{P}(D) \qquad \mathsf{isDirected}(\alpha_1) \qquad \mathsf{isDirected}(\alpha_2)}{\bigsqcup_{i:I} \alpha_1(i) \cup \bigsqcup_{i:I} \alpha_2(i) = \bigsqcup_{i:I} \alpha_1(i) \cup \alpha_2(i)}$$

Powerdomain QIIT (Operations)

$$\{-\} : D \to \mathcal{P}(D) \qquad\qquad \cup : \mathcal{P}(D) \to \mathcal{P}(D) \to \mathcal{P}(D)$$

$$\frac{\alpha_1, \alpha_2 : I \to \mathcal{P}(D) \qquad \mathsf{isDirected}(\alpha_1) \qquad \mathsf{isDirected}(\alpha_2)}{\bigsqcup_{i:I} \alpha_1(i) \cup \bigsqcup_{i:I} \alpha_2(i) = \bigsqcup_{i:I} \alpha_1(i) \cup \alpha_2(i)}$$

$$\frac{\alpha : I \to D \qquad \mathsf{isDirected}(\alpha)}{\{\bigsqcup_{i:I} \alpha(i)\} = \bigsqcup_{i:I} \{\alpha(i)\}}$$

Powerdomain QIIT (Inequalities)

$$\frac{}{x \cup y \sqsubseteq y \cup x} \qquad \frac{}{(x \cup y) \cup z \sqsubseteq x \cup (y \cup z)}$$

$$\frac{}{x \cup x \sqsubseteq x} \qquad \frac{}{x \sqsubseteq x \cup x}$$

General QIIT

We define $\mathrm{Initial}_\Sigma : \mathcal{U}$ and $\sqsubseteq : \mathrm{Initial}_\Sigma \to \mathrm{Initial}_\Sigma \to \mathcal{U}$ as a QIIT

- Constructors for DCPO structure stay the same
- For every operation named a in Σ, we add a constructor $\mathsf{app}_a : (\mathrm{Initial}_\Sigma)^B \times C \to \mathrm{Initial}_\Sigma$ and constructors for its continuity
- For every inequality in Σ, we add a constructor

Different notion of continuity

Recall: $f : D \to E$ is continuous if

- f is monotone
- $f(\bigsqcup_{i:I} \alpha(i)) = \bigsqcup_{i:I} f(\alpha(i))$

[3] Jong and Escardó, "Domain Theory in Constructive and Predicative Univalent Foundations".

Different notion of continuity

Recall: $f : D \to E$ is continuous if

- f is monotone
- $f(\bigsqcup_{i:I} \alpha(i)) = \bigsqcup_{i:I} \underbrace{f(\alpha(i))}_{\text{Need monotonicity to show that } f \circ \alpha \text{ is directed}}$

Instead, we define continuity as follows[3]:

- $f(\bigsqcup_{i:I} \alpha(i))$ is a supremum for $f \circ \alpha$

[3] Jong and Escardó, "Domain Theory in Constructive and Predicative Univalent Foundations".

Actual constructors for operations and their continuity

$$\mathsf{app}_a : (\mathrm{Initial}_{\boldsymbol{\Sigma}})^B \times C \to \mathrm{Initial}_{\boldsymbol{\Sigma}}$$

Actual constructors for operations and their continuity

$$\mathsf{app}_a : (\mathrm{Initial}_{\Sigma})^B \times C \to \mathrm{Initial}_{\Sigma}$$

$$\frac{\alpha : I \to (\mathrm{Initial}_{\Sigma})^B \times C \qquad \delta : \mathsf{isDirected}(\alpha)}{\prod_{i:I} \mathsf{app}_a(\alpha(i)) \sqsubseteq \mathsf{app}_a(\bigsqcup_{i:I} \alpha(i))}$$

Actual constructors for operations and their continuity

$$\mathsf{app}_a : (\mathrm{Initial}_\Sigma)^B \times C \to \mathrm{Initial}_\Sigma$$

$$\frac{\alpha : I \to (\mathrm{Initial}_\Sigma)^B \times C \qquad \delta : \mathsf{isDirected}(\alpha)}{\prod_{i:I} \mathsf{app}_a(\alpha(i)) \sqsubseteq \mathsf{app}_a(\bigsqcup_{i:I} \alpha(i))}$$

$$\frac{\alpha : I \to (\mathrm{Initial}_\Sigma)^B \times C \qquad \delta : \mathsf{isDirected}(\alpha)}{\prod_{v:\mathrm{Initial}_\Sigma} \mathsf{isUpperbound}(v, \mathsf{app}_a \circ \alpha) \to \mathsf{app}_a(\bigsqcup_{i:I} \alpha(i)) \sqsubseteq v}$$

Elimination

Elimination principles are as expected:

- For every Σ-algebra X, there exists an algebra morphism $\mathrm{Initial}_{\Sigma} \to X$
- Using the induction principle, we can show uniqueness
- Thus $\mathrm{Initial}_{\Sigma}$ in indeed initial in Σ-Alg

Free algebras

To construct the free Σ-algebra for a DCPO D:

- Define signature $\Sigma + D$, by adding $D \to X$
- Consider $\mathrm{Initial}_{\Sigma+D}$
- This is a Σ-algebra if we forget about the inclusion
- So $F(D) = (\mathrm{Initial}_{\Sigma+D})^{-}$
- $\eta_D : D \to \mathrm{Initial}_{\Sigma+D}$ is given by the inclusion of $\mathrm{Initial}_{\Sigma+D}$

Conclusions

- Framework for describing algebraic effects via
  - Operations
  - Inequalities
- Construct them as initial algebras via a QIIT
- Free algebras
- Partiality, Powerdomain, Smash products, Coequalizers, ...
- Formalized in Cubical Agda

References

Altenkirch, Thorsten, Nils Anders Danielsson, and Nicolai Kraus. "Partiality, Revisited - The Partiality Monad as a Quotient Inductive-Inductive Type". In: *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*. Ed. by Javier Esparza and Andrzej S. Murawski. Vol. 10203. Lecture Notes in Computer Science. 2017, pp. 534–549. DOI: 10.1007/978-3-662-54458-7_31. URL: https://doi.org/10.1007/978-3-662-54458-7%5C_31.

Jong, Tom de and Martín Hötzel Escardó. "Domain Theory in Constructive and Predicative Univalent Foundations". In: *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*. Ed. by Christel Baier and Jean Goubault-Larrecq. Vol. 183. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 28:1–28:18. DOI: 10.4230/LIPICS.CSL.2021.28.