A realisability model of linear dependent type theory

Sam Speight, **Niels van der Weide**

This talk

In this talk, we look at an extension of dependent type theory with two features

- **Linearity**: functions use their argument exactly once
- **Impredicativity**: a strong polymorphism principle

Impredicativity in Type Theory

Impredicativity: we have universe $\mathcal{U}$ of types such that

$$\frac{\Gamma \vdash A\ \mathsf{Type} \qquad \Gamma, x : A \vdash B[x] : \mathcal{U}}{\Gamma \vdash \prod(x : A), B[x] : \mathcal{U}}$$

Impredicativity in Type Theory

Impredicativity: we have universe $\mathcal{U}$ of types such that

$$\frac{\Gamma \vdash A\ \mathsf{Type} \qquad \Gamma, x : A \vdash B[x] : \mathcal{U}}{\Gamma \vdash \prod(x : A), B[x] : \mathcal{U}}$$

Consequences:

- $\mathcal{U}$ interprets impredicative polymorphism
- We can encode data types via their recursion principle: **impredicative encodings**

Our goal

- **Goal this talk**: construct a realizability model of linear dependent type theory
- This model supports both linearity and impredicativity
- The material of this talk is based on `https://arxiv.org/abs/2602.08846`
- The model is formalised in Rocq using UniMath [1]
- In the paper, we also demonstrate how to use impredicative encodings to construct initial algebras

[1] `https://github.com/nmvdw/LinearRealizability`

Table of Contents

Table of Contents

Brief Recap: Linear Logic

- Linear logic is a **substructural logic** without weakening and contraction
- This means: every **assumption must be used exactly once**
- Linear logic comes with various connectives, and we only consider $\otimes$, $\multimap$, and ! in this talk

Some Rules in Linear Logic

Rules for $\otimes$

$$\frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi \otimes \psi} \qquad \frac{\Gamma \vdash \phi \otimes \psi \qquad \Delta, \phi, \psi \vdash \chi}{\Gamma, \Delta \vdash \chi}$$

Some Rules in Linear Logic

Rules for $\otimes$

$$\frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi \otimes \psi} \qquad \frac{\Gamma \vdash \phi \otimes \psi \qquad \Delta, \phi, \psi \vdash \chi}{\Gamma, \Delta \vdash \chi}$$

Rules for $\multimap$

$$\frac{\Gamma \vdash \phi \multimap \psi \qquad \Delta \vdash \phi}{\Gamma, \Delta \vdash \psi} \qquad \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \multimap \psi}$$

Some Rules in Linear Logic

Rules for $\otimes$

$$\frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi \otimes \psi} \qquad \frac{\Gamma \vdash \phi \otimes \psi \qquad \Delta, \phi, \psi \vdash \chi}{\Gamma, \Delta \vdash \chi}$$

Rules for $\multimap$

$$\frac{\Gamma \vdash \phi \multimap \psi \qquad \Delta \vdash \phi}{\Gamma, \Delta \vdash \psi} \qquad \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \multimap \psi}$$

Some rules for !

$$\frac{\Gamma \vdash \phi}{\Gamma, !\psi \vdash \phi} \qquad \frac{\Gamma, !\psi, !\psi \vdash \phi}{\Gamma, !\psi \vdash \phi}$$

The Features of Linear Dependent Type Theory

Linear dependent type theory[2] [3] [4] [5] is an extension of dependent type theory with the features of linear logic

- Two notions of types: **Cartesian types** and **Linear types**
- Cartesian types give us good old dependent type theory

[2] *A linear logical framework*, Cervesato, Pfennong
[3] *Integrating linear and dependent types*, Krishnaswami, Pradic, Benton
[4] *Models of linear dependent type theory*, Lundfall
[5] *A Categorical Semantics for Linear Logical Frameworks*, Vakar

The Features of Linear Dependent Type Theory

Linear dependent type theory[2] [3] [4] [5] is an extension of dependent type theory with the features of linear logic

- Two notions of types: **Cartesian types** and **Linear types**
- Cartesian types give us good old dependent type theory
- Linear types can depend **only on Cartesian types**
- Linear types in a fixed context: model of linear logic
- Terms of linear types depend on **both Cartesian variables and linear variables**

[2] *A linear logical framework*, Cervesato, Pfennong
[3] *Integrating linear and dependent types*, Krishnaswami, Pradic, Benton
[4] *Models of linear dependent type theory*, Lundfall
[5] *A Categorical Semantics for Linear Logical Frameworks*, Vakar

The Features of Linear Dependent Type Theory

Linear dependent type theory[2] [3] [4] [5] is an extension of dependent type theory with the features of linear logic

- Two notions of types: **Cartesian types** and **Linear types**
- Cartesian types give us good old dependent type theory
- Linear types can depend **only on Cartesian types**
- Linear types in a fixed context: model of linear logic
- Terms of linear types depend on **both Cartesian variables and linear variables**
- Additional type formers: linear variants of $\sum$ and $\prod$, denoted by $\sqsubset$ and $\sqcap$

[2] *A linear logical framework*, Cervesato, Pfennong
[3] *Integrating linear and dependent types*, Krishnaswami, Pradic, Benton
[4] *Models of linear dependent type theory*, Lundfall
[5] *A Categorical Semantics for Linear Logical Frameworks*, Vakar

The Basic Ingredients

We have:

- Cartesian types $A, B, \ldots$
- Linear types $X, Y, \ldots$
- Cartesian contexts $\Gamma, \Delta, \ldots$ (i.e., lists of Cartesian types)
- Linear contexts $\Xi, \Theta, \ldots$ (i.e. lists of linear types in context Γ)

The Basic Ingredients

We have:

- Cartesian types $A, B, \ldots$
- Linear types $X, Y, \ldots$
- Cartesian contexts $\Gamma, \Delta, \ldots$ (i.e., lists of Cartesian types)
- Linear contexts $\Xi, \Theta, \ldots$ (i.e. lists of linear types in context Γ)

The judgements are as follows

- Cartesian types: $\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}}$
- Linear types: $\Gamma \vdash X\ \mathsf{Type}_{\mathsf{lin}}$
- Cartesian terms: $\Gamma \vdash t : A$
- Linear terms: $\Gamma \mid \Xi \vdash t : X$

The Basic Ingredients

We have:

- Cartesian types $A, B, \ldots$
- Linear types $X, Y, \ldots$
- Cartesian contexts $\Gamma, \Delta, \ldots$ (i.e., lists of Cartesian types)
- Linear contexts $\Xi, \Theta, \ldots$ (i.e. lists of linear types in context Γ)

The judgements are as follows

- Cartesian types: $\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}}$
- Linear types: $\Gamma \vdash X\ \mathsf{Type}_{\mathsf{lin}}$
- Cartesian terms: $\Gamma \vdash t : A$
- Linear terms: $\Gamma \mid \Xi \vdash t : X$

Note the dual context for linear terms

Some Type Formers for Linear Types

Linear conjunction:

$$\frac{\Gamma \vdash X\ \mathsf{Type}_{\mathsf{lin}} \qquad \Gamma \vdash Y\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash X \otimes Y\ \mathsf{Type}_{\mathsf{lin}}}$$

Linear function types:

$$\frac{\Gamma \vdash X\ \mathsf{Type}_{\mathsf{lin}} \qquad \Gamma \vdash Y\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash X \multimap Y\ \mathsf{Type}_{\mathsf{lin}}}$$

Linear Terms

A rule for $\otimes$

$$\frac{\Gamma \mid \Xi \vdash t : X \qquad \Gamma \mid \Theta \vdash t' : Y}{\Gamma \mid \Xi, \Theta \vdash (t, t') : X \otimes Y}$$

Linear Terms

A rule for $\otimes$

$$\frac{\Gamma \mid \Xi \vdash t : X \qquad \Gamma \mid \Theta \vdash t' : Y}{\Gamma \mid \Xi, \Theta \vdash (t, t') : X \otimes Y}$$

A rule for $\multimap$

$$\frac{\Gamma \mid \Xi \vdash f : X \multimap Y \qquad \Gamma \mid \Theta \vdash t : X}{\Gamma \mid \Xi, \Theta \vdash f\ t : Y}$$

Table of Contents

Linear Combinatory Algebras: Idea

Linear combinatory algebras relate to the linear λ-calculus as combinatory algebras do to the λ-calculus.

- Linear combinatory algebras reflect the features of linear logic: linear exponentials and less structural rules
- The combinators in linear combinatory algebras reflect the rules in a Hilbert-style axiomatization of linear logic
- Linear combinatory algebras satisfy a weaker form of combinatory completeness

Combinatory Algebras

Definition

A **combinatory algebra** consists of a set A together with an operation, denoted by $\cdot$, and elements $K, S \in A$ such that

$$K \cdot a \cdot b = a$$

$$S \cdot a \cdot b \cdot c = a \cdot c \cdot (b \cdot c)$$

Linear Combinatory Algebras: Definition

Definition

A **linear combinatory algebra**[6] (LCA) is consists of a set A, a binary operation $\cdot$, and a unary operation $! : A \to A$, such that there are elements $\mathsf{B}, \mathsf{I}, \mathsf{C}, \mathsf{W}, \mathsf{K}, \mathsf{D}, \delta, \mathsf{F} \in A$ satisfying the equations in the next slide.

[6] *Geometry of interaction and linear combinatory algebras*, Abramsky, Haghverdi, Scott

Linear Combinatory Algebras: Definition

Combinator identity	Principal type
$\mathsf{B}abc = a(bc)$	$(\alpha \multimap \beta) \multimap (\gamma \multimap \alpha) \multimap \gamma \multimap \beta$
$\mathsf{I}a = a$	$\alpha \multimap \alpha$
$\mathsf{C}abc = acb$	$(\alpha \multimap \beta \multimap \gamma) \multimap \beta \multimap \alpha \multimap \gamma$
$\mathsf{W}a!b = a!b!b$	$(!\alpha \multimap !\alpha \multimap \beta) \multimap !\alpha \multimap \beta$
$\mathsf{K}a!b = a$	$\alpha \multimap !\beta \multimap \alpha$
$\mathsf{D}!a = a$	$!\alpha \multimap \alpha$
$\delta!a = !!a$	$!\alpha \multimap !!\alpha$
$\mathsf{F}!a!b = !(ab)$	$!(\alpha \multimap \beta) \multimap !\alpha \multimap !\beta$

Linear Combinatory Algebras: Definition

Combinator identity	Logical principle
$\mathsf{B}abc = a(bc)$	cut/composition
$\mathsf{I}a = a$	identity
$\mathsf{C}abc = acb$	exchange
$\mathsf{W}a!b = a!b!b$	contraction
$\mathsf{K}a!b = a$	weakening
$\mathsf{D}!a = a$	dereliction
$\delta!a = !!a$	comultiplication
$\mathsf{F}!a!b = !(ab)$	functoriality

LCAs and CAs

Every LCA gives rise to a CA

Given an LCA A, we define a CA $A_!$:

- Carrier A
- Application $a \cdot_! b$ in $A_!$ is $a \cdot !b$ in A

LCAs and CAs

Every LCA gives rise to a CA

Given an LCA A, we define a CA $A_!$:

- Carrier A
- Application $a \cdot_! b$ in $A_!$ is $a \cdot !b$ in A

Recall: one can embed intuitionistic logic in linear logic by taking $\varphi \to \psi$ to be $!\varphi \multimap \psi$

Linear Combinatory Algebras: Examples[8]

- Every combinatory algebra (take $!x = x$)
- The linear λ-calculus
- Linear graph models
- One can also get examples using the framework of **geometry of interaction**[7]

[7] *Geometry of interaction and linear combinatory algebras*, Abramsky, Haghverdi, Scott

[8] *Linear realizability*, Hoshino

Table of Contents

Overview of the Construction

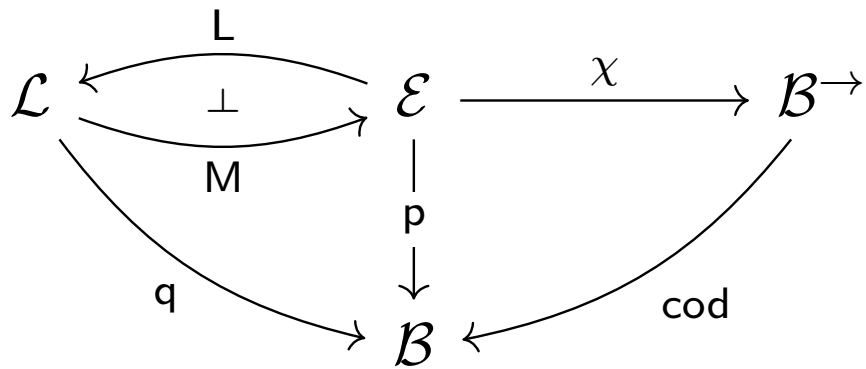
- The linear realizability model is an extension of the traditional realizability model
- Main difference: we use linear combinatory algebras rather than combinatory algebras
- Note: to interpret Cartesian types, we use the realizability model for the CA $A_!$

Overview of the Construction

- The linear realizability model is an extension of the traditional realizability model
- Main difference: we use linear combinatory algebras rather than combinatory algebras
- Note: to interpret Cartesian types, we use the realizability model for the CA $A_!$
- In the semantics, we use the notion of **linear comprehension category**
- This notion gives a compact formulation of models of linear dependent type theory

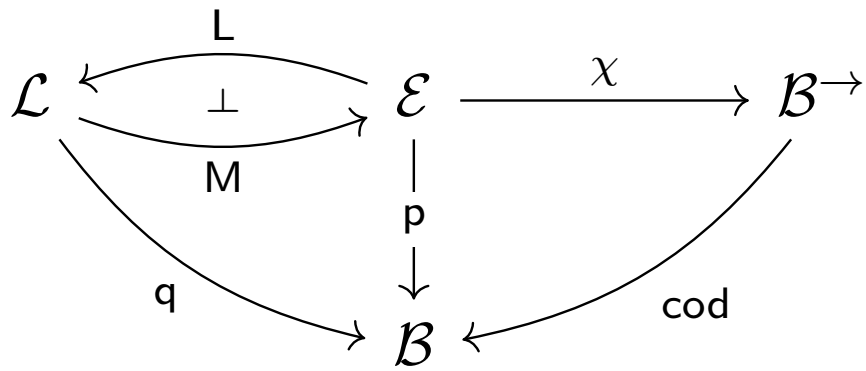
Linear Comprehension Categories

A **linear comprehension category** is a commutative diagram of functors as follows



Linear Comprehension Categories

A **linear comprehension category** is a commutative diagram of functors as follows



Requirements:

- p is a fibration and q is a symmetric monoidal closed fibration
- $\mathcal{E}$ has fibrewise terminal objects and binary products
- χ is fully faithful and preserves cartesian morphisms and terminal objects
- L $\dashv$ M is a fibrewise symmetric monoidal adjunction
- L and M preserve cartesian morphisms.

Interpretation of Contexts: Assemblies

Let A be a combinatory algebra.

Definition

An **assembly** over A consists of a set X together with a relation $\Vdash \subseteq A \times X$ such that for each $x \in X$ there is $a \in A$ with $a \Vdash x$.

Definition

A **morphism** from an assembly X to Y is a map $f : X \to Y$ for which there $e \in A$ such that for all $a \Vdash_X x$ we have $ea \Vdash_Y f(x)$.

We write $\mathrm{Asm}(A)$ for the category of assemblies.

Interpretation of Contexts: Assemblies

Let A be a combinatory algebra.

Definition

An **assembly** over A consists of a set X together with a relation $\Vdash \subseteq A \times X$ such that for each $x \in X$ there is $a \in A$ with $a \Vdash x$.

Definition

A **morphism** from an assembly X to Y is a map $f : X \to Y$ for which there $e \in A$ such that for all $a \Vdash_X x$ we have $ea \Vdash_Y f(x)$.

We write $\mathrm{Asm}(A)$ for the category of assemblies.
Note: we will use this for $A_!$ where A is a LCA

Interpretation of Cartesian Types: Families of Assemblies

Let A be a combinatory algebra.

Definition

A **family Y of assemblies over** $\Gamma : \mathrm{Asm}(A)$ consists of an assembly Y_x for each $x : \Gamma$.

We write $\mathrm{DAsm}_{\mathsf{C}}(A)$ for the category whose objects consists of $\Gamma : \mathrm{Asm}(A)$ together with a family Y of assemblies over Γ.

Interpretation of Cartesian Types: Families of Assemblies

Let A be a combinatory algebra.

Definition

A **family Y of assemblies over** $\Gamma : \mathrm{Asm}(A)$ consists of an assembly Y_x for each $x : \Gamma$.

We write $\mathrm{DAsm}_{\mathsf{C}}(A)$ for the category whose objects consists of $\Gamma : \mathrm{Asm}(A)$ together with a family Y of assemblies over Γ.
Note: we will use this for $A_!$ where A is a LCA

Interpretation of Linear Types and Terms

Let A be an LCA

Interpretation of linear types: assemblies for A (same as Cartesian types)

Interpretation of Linear Types and Terms

Let A be an LCA

Interpretation of linear types: assemblies for A (same as Cartesian types)

Interpretation of linear terms: Suppose we have

- an assembly Γ over $A_!$
- for each $x : \Gamma$ assemblies X_x and Y_x over A

We interpret a linear term as

- maps $f_x : X_x \to Y_x$ for each $x : \Gamma$
- for which there exists $a : A$ such that $a\ !b_1\ b_2 \Vdash f_x(\overline{x})$ whenever $b_1 \Vdash x$ and $b_2 \Vdash \overline{x}$ with $\overline{x} : X_x$

Interpretation of Linear Types and Terms

Let A be an LCA

Interpretation of linear types: assemblies for A (same as Cartesian types)

Interpretation of linear terms: Suppose we have

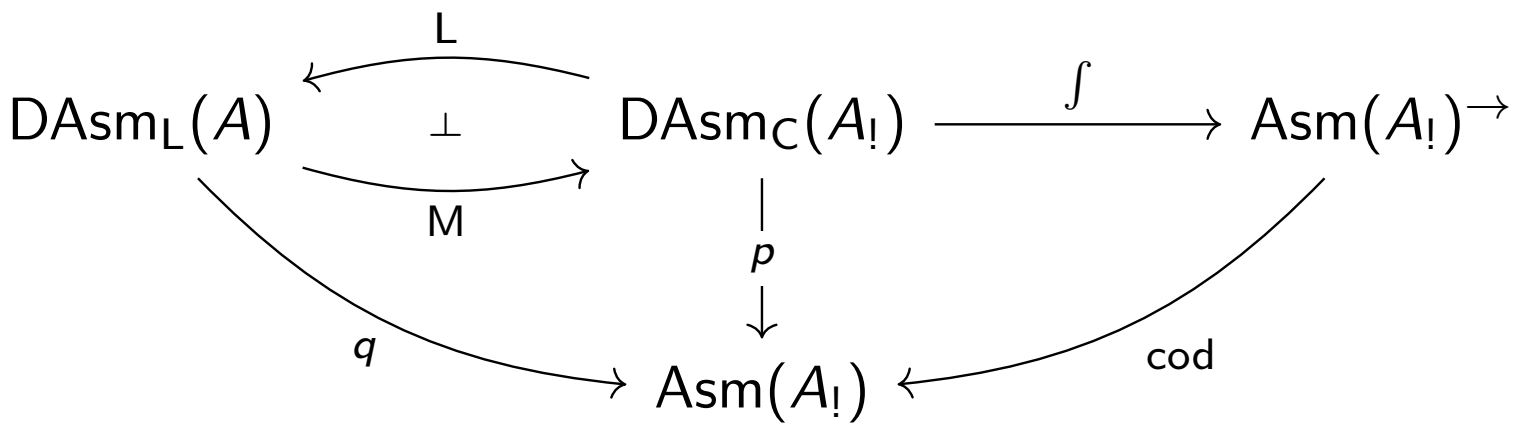
- an assembly Γ over $A_!$
- for each $x : \Gamma$ assemblies X_x and Y_x over A

We interpret a linear term as

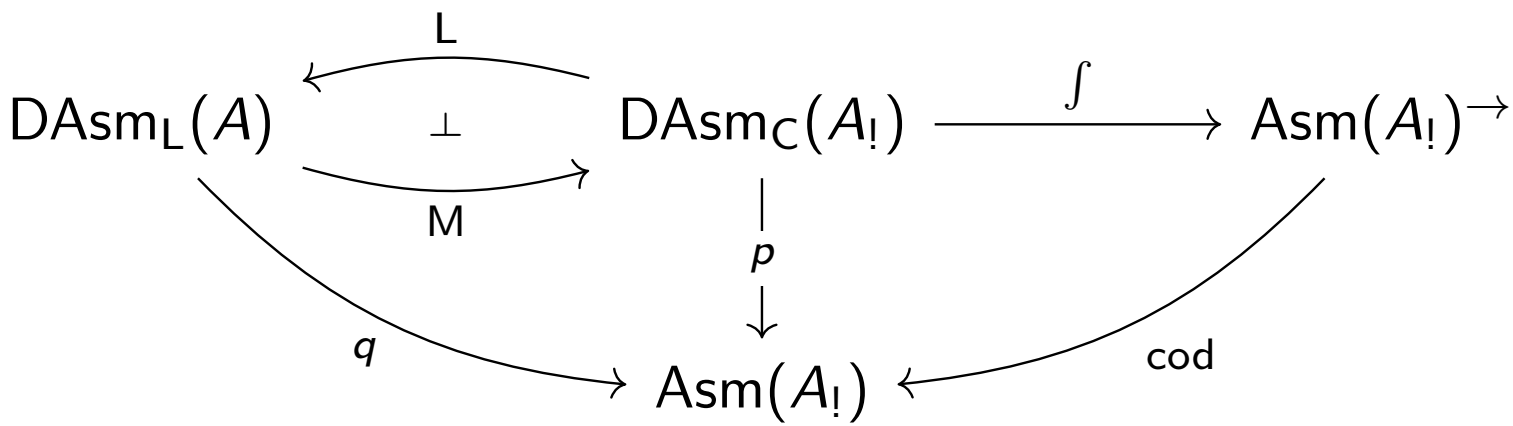
- maps $f_x : X_x \to Y_x$ for each $x : \Gamma$
- for which there exists $a : A$ such that $a\ !b_1\ b_2 \Vdash f_x(\overline{x})$ whenever $b_1 \Vdash x$ and $b_2 \Vdash \overline{x}$ with $\overline{x} : X_x$

Note: this gives a category $\mathsf{DAsm}_\mathsf{L}(A)$

Linear Realizability Model

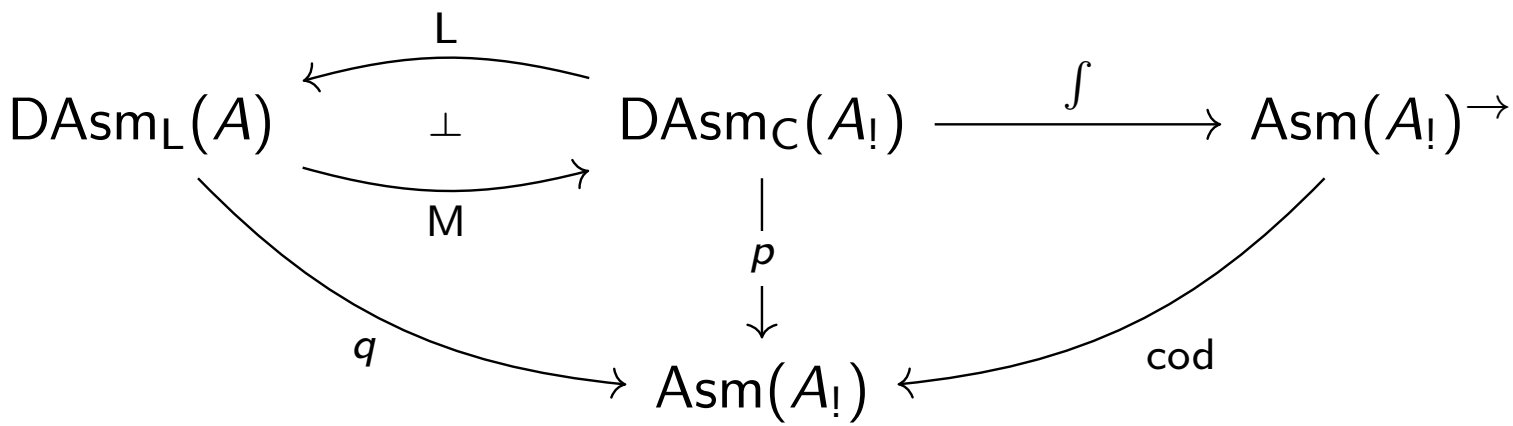


Linear Realizability Model



M is the identity on objects

Linear Realizability Model



M is the identity on objects

Given assemblies Y_x for each $x : \Gamma$, $\mathsf{L}(Y)$ is the assembly

- whose carrier is Y_x
- $a \Vdash \overline{x}$ in $\mathsf{L}(Y)$ if there is b with $a = !b$ and $b \Vdash \overline{x}$

Impredicative Universe

The rules for universes are as follows

$$\Gamma \vdash \mathcal{U}\ \mathsf{Type}_{\mathsf{cart}}$$

$$\frac{\Gamma \vdash a : \mathcal{U}}{\Gamma \vdash \mathsf{El}_{\mathsf{C}}\,(a)\ \ \mathsf{Type}_{\mathsf{cart}}} \qquad \frac{\Gamma \vdash x : \mathcal{U}}{\Gamma \vdash \mathsf{El}_{\mathsf{L}}\,(x)\ \ \mathsf{Type}_{\mathsf{lin}}}$$

Essentially: Tarski-style universe with two decoding function

Impredicative Universe

The rules for universes are as follows

$$\Gamma \vdash \mathcal{U}\ \mathsf{Type}_{\mathsf{cart}}$$

$$\frac{\Gamma \vdash a : \mathcal{U}}{\Gamma \vdash \mathsf{El}_{\mathsf{C}}\,(a)\ \ \mathsf{Type}_{\mathsf{cart}}} \qquad \frac{\Gamma \vdash x : \mathcal{U}}{\Gamma \vdash \mathsf{El}_{\mathsf{L}}\,(x)\ \ \mathsf{Type}_{\mathsf{lin}}}$$

Essentially: Tarski-style universe with two decoding function
Interpretation:

- $\mathcal{U}$: all PERs for A
- El_{C} and El_{L}: modest sets and PERs are equivalent

Table of Contents

Conclusion

- We showed that impredicativity is consistent with linear dependent type theory
- Our model is an extension of traditional realizability models
- Main difference: we use linear combinatory algebras
- Our model interprets all expected type formers from linear dependent type theory

Check our preprint here: `https://arxiv.org/abs/2602.08846`

Extra: Linear Combinatory Algebras: Completeness

In an LCA, we have two ways to do λ-abstraction[9].

Linear λ-abstraction:

- Written: $\lambda^* x.M$
- The variable must occur exactly once and not under a !
- We have β-reduction: $(\lambda^* x.M)N = M[x \mapsto N]$

Nonlinear λ-abstraction:

- Written: $\lambda!^* x.M$
- No restriction on the variable
- β-reduction is restricted: $(\lambda^* x.M)(!N) = M[x \mapsto N]$

[9] *Reduction in a linear lambda-calculus with applications to operational semantics*, Simpson