# A Realisability Model for Impredicative 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 an 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 end, we also study a general notion of model of linear logic

[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 $\Gamma$)

# 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 $\Gamma$)

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 $\Gamma$)

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 Simple 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 $\multimap$

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

# More Type Formers

There are more interesting type formers in linear dependent type theory:

- Linear $\sum$-types: $\sqsubset_{a:A} X[a]$
- Linear $\prod$-types: $\sqcap_{a:A} X[a]$
- The linear exponential: $!X$

# More Type Formers

There are more interesting type formers in linear dependent type theory:

- Linear $\sum$-types: $\sqsubset_{a:A} X[a]$
- Linear $\prod$-types: $\sqcap_{a:A} X[a]$
- The linear exponential: $!X$

**Note**:

- $A$ is **Cartesian** and $X$ is **linear**
- We represent the linear exponential via an adjunction between linear and Cartesian types

# Rules for $\sqsubset$-types

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}} \qquad \Gamma, a : A \vdash X[a]\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \sqsubset_{a:A} X[a]\ \mathsf{Type}_{\mathsf{lin}}}$$

# Rules for $\sqsubset$-types

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}} \qquad \Gamma, a : A \vdash X[a]\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \sqsubset_{a:A} X[a]\ \mathsf{Type}_{\mathsf{lin}}}$$

Introduction:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \mid \Xi \vdash t' : X[a \mapsto t]}{\Gamma \mid \Xi \vdash (t, t') : \sqsubset_{a:A} X[a]}$$

# Rules for $\sqsubset$-types

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}} \qquad \Gamma, a : A \vdash X[a]\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \sqsubset_{a:A} X[a]\ \mathsf{Type}_{\mathsf{lin}}}$$

Introduction:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \mid \Xi \vdash t' : X[a \mapsto t]}{\Gamma \mid \Xi \vdash (t, t') : \sqsubset_{a:A} X[a]}$$

Elimination:

$$\frac{\Gamma \mid \Xi \vdash t : \sqsubset_{a:A} X[a] \qquad \Gamma, a : A \mid \Xi, x : X[a] \vdash t' : Y}{\Gamma \mid \Xi \vdash \mathsf{let}\ t\ \mathsf{be}\ (a, x)\ \mathsf{in}\ t' : Y}$$

# Rules for $\bigsqcap$-types

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}} \qquad \Gamma, a : A \vdash X[a]\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \bigsqcap_{a:A} X[a]\ \mathsf{Type}_{\mathsf{lin}}}$$

# Rules for $\bigsqcap$-types

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}} \qquad \Gamma, a : A \vdash X[a]\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \bigsqcap_{a:A} X[a]\ \mathsf{Type}_{\mathsf{lin}}}$$

Introduction:

$$\frac{\Gamma, a : A \mid \Xi \vdash f : X[a]}{\Gamma \mid \Xi \vdash \lambda(a : A), f : \bigsqcap_{a:A} X[a]}$$

# Rules for $\bigsqcap$-types

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}} \qquad \Gamma, a : A \vdash X[a]\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \bigsqcap_{a:A} X[a]\ \mathsf{Type}_{\mathsf{lin}}}$$

Introduction:

$$\frac{\Gamma, a : A \mid \Xi \vdash f : X[a]}{\Gamma \mid \Xi \vdash \lambda(a : A), f : \bigsqcap_{a:A} X[a]}$$

Elimination:

$$\frac{\Gamma \mid \Xi \vdash f : \bigsqcap_{a:A} X[a] \qquad \Gamma \vdash t : A}{\Gamma \mid \Xi \vdash f\ t : X[a \mapsto t]}$$

# Some Rules for Linear Exponentials

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}}}{\Gamma \vdash \mathsf{L}(A)\ \mathsf{Type}_{\mathsf{lin}}} \qquad \frac{\Gamma \vdash X\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \mathsf{M}(X)\ \mathsf{Type}_{\mathsf{cart}}}$$

We call L **linearisation** and M **multiplication**, and we define $!X$ to be $\mathsf{L}(\mathsf{M}(X))$

# Some Rules for Linear Exponentials

Formation:

$$\frac{\Gamma \vdash A\ \mathsf{Type}_{\mathsf{cart}}}{\Gamma \vdash \mathsf{L}(A)\ \mathsf{Type}_{\mathsf{lin}}} \qquad \frac{\Gamma \vdash X\ \mathsf{Type}_{\mathsf{lin}}}{\Gamma \vdash \mathsf{M}(X)\ \mathsf{Type}_{\mathsf{cart}}}$$

We call L **linearisation** and M **multiplication**, and we define $!X$ to be $\mathsf{L}(\mathsf{M}(X))$

Introduction rules:

$$\frac{\Gamma \mid \cdot \vdash x : X}{\Gamma \vdash \mathsf{M}(x) : \mathsf{M}(X)} \qquad \frac{\Gamma \vdash a : A}{\Gamma \mid \cdot \vdash \mathsf{L}(a) : \mathsf{L}(A)}$$

# Some More Rules for Linear Exponentials

Elimination rules:

$$\frac{\Gamma \vdash x : \mathsf{M}(X)}{\Gamma \mid \cdot \vdash \sigma(x) : X}$$

$$\frac{\Gamma \mid \Xi \vdash t : \mathsf{L}(A) \qquad \Gamma, a : A \mid \Theta \vdash t' : X}{\Gamma \mid \Xi, \Theta \vdash \mathsf{let}\ t\ \mathsf{be}\ a\ \mathsf{in}\ t' : X}$$

# Table of Contents

# Overview: Semantics of Dependent Linear Type Theory

Before we give the realizability model, we define **linear comprehension categories**[6]
This notion combines two ideas

- **First idea**: comprehension categories as models of dependent type theory
- **Second idea**: linear non-linear models of linear logic

[6] *Models of linear dependent type theory*, Lundfall

# Semantics of Multiplicative Intuitionistic Linear Logic

- **Multiplicative Intuitionistic Linear Logic (MILL)**: linear logic restricted to $\multimap$, $\otimes$, and a unit for $\otimes$
- Models for MILL: symmetric monoidal closed categories

# Semantics of Linear Exponentials

There are various notion of categorical models for linear logic:

- Lafont models (based on free comonoids)
- Seely models (based on Kleisli categories)
- Linear categories (based on Eilenberg-Moore categories)
- Linear Non-Linear Models (based on adjunctions)

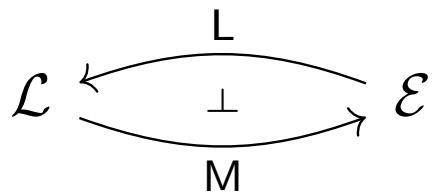
# Semantics of Linear Exponentials

There are various notion of categorical models for linear logic:

- Lafont models (based on free comonoids)
- Seely models (based on Kleisli categories)
- Linear categories (based on Eilenberg-Moore categories)
- **Linear Non-Linear Models** (based on adjunctions)

# Linear Non-Linear Models

A **linear non-linear model** consists of functors as follows

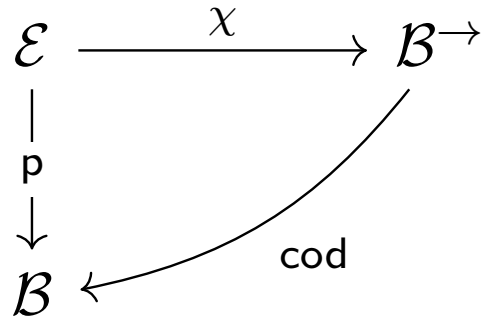



**Requirements**:

- L is a symmetric monoidal closed category
- M is a Cartesian monoidal category
- L $\dashv$ M is a symmetric monoidal adjunction

# Semantics of Dependent Type Theory

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

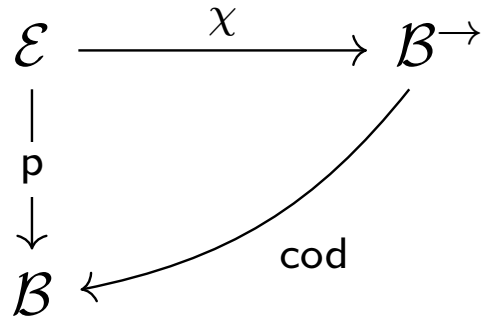



**Requirements**:

- p is a fibration
- $\chi$ preserves Cartesian morphisms

# Semantics of Dependent Type Theory

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




**Requirements**:

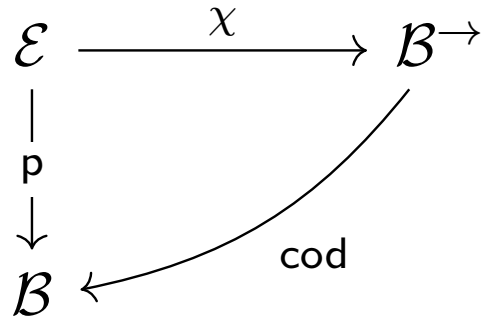
- p is a fibration
- $\chi$ preserves Cartesian morphisms

We also require:

- $\chi$ is fully faithful
- $\chi$ preserves terminal objects

# Semantics of Dependent Type Theory

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




**Requirements**:

- p is a fibration
- $\chi$ preserves Cartesian morphisms

We also require:

- $\chi$ is fully faithful
- $\chi$ preserves terminal objects

To define linear comprehension categories, we "merge" the notations of comprehension category with that of a linear non-linear model.

# Monoidal Fibrations

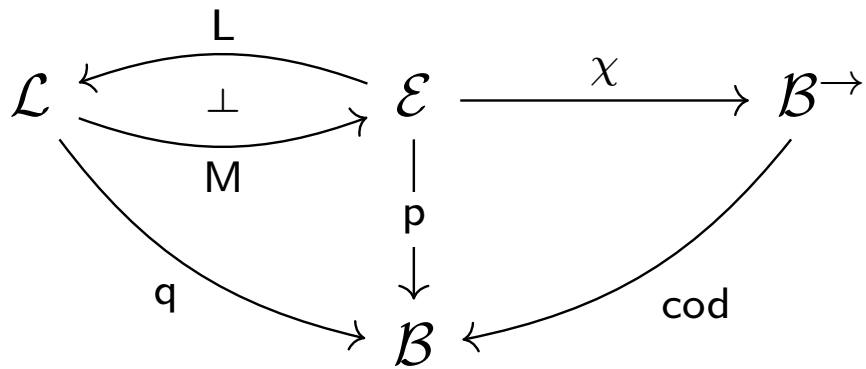
Let $\mathsf{q} : \mathcal{L} \to \mathcal{B}$ be a fibration.
We say that q is a **symmetric monoidal closed fibration** if

- each fibre $\mathsf{q}[\Gamma]$ is symmetric monoidal closed
- the functors $s^* : \mathsf{q}[\Delta] \to \mathsf{q}[\Gamma]$ are symmetric monoidal closed for each $s : \Gamma \to \Delta$

**Note**: such fibrations correspond to pseudofunctors
$\mathcal{B}^{\mathrm{op}} \to \mathsf{SymMonClosedCat}$

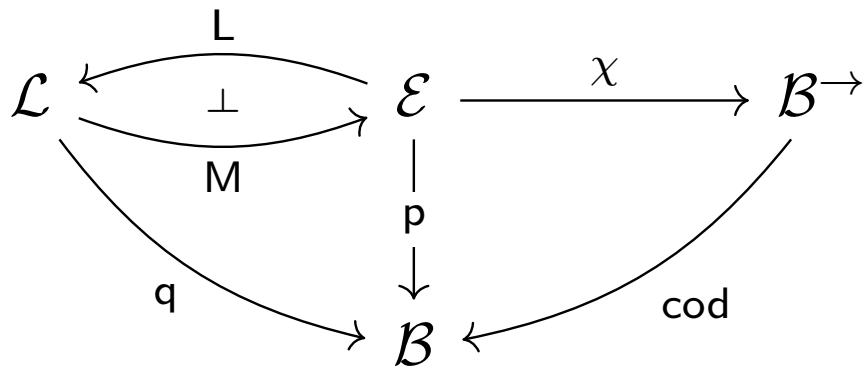
# Linear Comprehension Categories

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
- $\chi$ 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.

# Table of Contents

# Linear Combinatory Algebras: Idea

Linear combinatory algebras relate to the linear $\lambda$-calculus as combinatory algebras do to the $\lambda$-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

## Definition

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

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

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

# Combinatory Algebras: Definition

## Definition

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

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

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

Combinatory algebras are **combinatory complete**: intuitively, this says that we can define elements using $\lambda$-abstraction.

# Combinatory Algebras: Definition

## Definition

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

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

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

Combinatory algebras are **combinatory complete**: intuitively, this says that we can define elements using $\lambda$-abstraction.

## Remark

Usually, the operation $\cdot$ is only required to be partial, but we shall only look at combinatory algebras where $\cdot$ is total

# Combinatory Algebras: Examples

There are various examples of combinatory algebras

- The untyped $\lambda$-calculus (identified up to $\beta\eta$-equivalence)
- Graph models of the $\lambda$-calculus
- Domain models (i.e., every DCPO $D$ such that $D \simeq [D \to D]$)

# Linear Combinatory Algebras: Definition

## Definition

A **linear combinatory algebra**[7] (LCA) is consists of a set $A$, a binary operation $\cdot$, and a unary operation $! : A \to A$, such that there are elements $B, I, C, W, K, D, \delta, F \in A$ satisfying the equations in the next slide.

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

# Linear Combinatory Algebras: Definition

| Combinator identity | Logical principle |
|---|---|
| $Babc = a(bc)$ | cut/composition |
| $Ia = a$ | identity |
| $Cabc = acb$ | exchange |
| $Wa!b = a!b!b$ | contraction |
| $Ka!b = a$ | weakening |
| $D!a = a$ | dereliction |
| $\delta!a =!!a$ | comultiplication |
| $F!a!b =!(ab)$ | functoriality |

# Linear Combinatory Algebras: Completeness

In an LCA, we have two ways to do $\lambda$-abstraction[8].

**Linear $\lambda$-abstraction**:

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

**Nonlinear $\lambda$-abstraction**:

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

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

# 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[10]

- Every combinatory algebra (take $!x = x$)
- The linear $\lambda$-calculus
- Linear graph models
- One can also get examples using the framework of **geometry of interaction**[9]

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

[10] *Linear realizability*, Hoshino

# Table of Contents

# Overview

**Key ideas behind realizability models**[11] [12]:

- We start with a **(partial) combinatory algebra**, and we construct a comprehension category
- To interpret types, we define **assemblies**
- We use **modest sets** to interpret impredicativity

[11] *An extended calculus of constructions*, Luo

[12] *Realizability models for type theories*, Reus

# Overview

**Key ideas behind realizability models**[11] [12]:

- We start with a **(partial) combinatory algebra**, and we construct a comprehension category
- To interpret types, we define **assemblies**
- We use **modest sets** to interpret impredicativity

For linear dependent type theory, we make two changes:

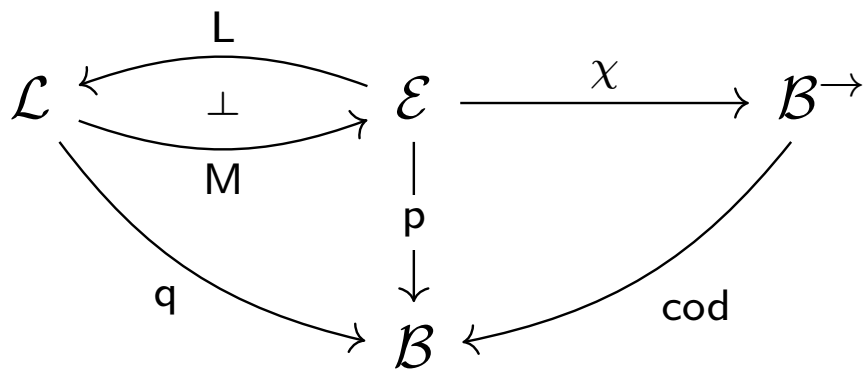
- Our starting point is a **linear combinatory algebra**
- We construct a linear comprehension category

[11] *An extended calculus of constructions*, Luo

[12] *Realizability models for type theories*, Reus

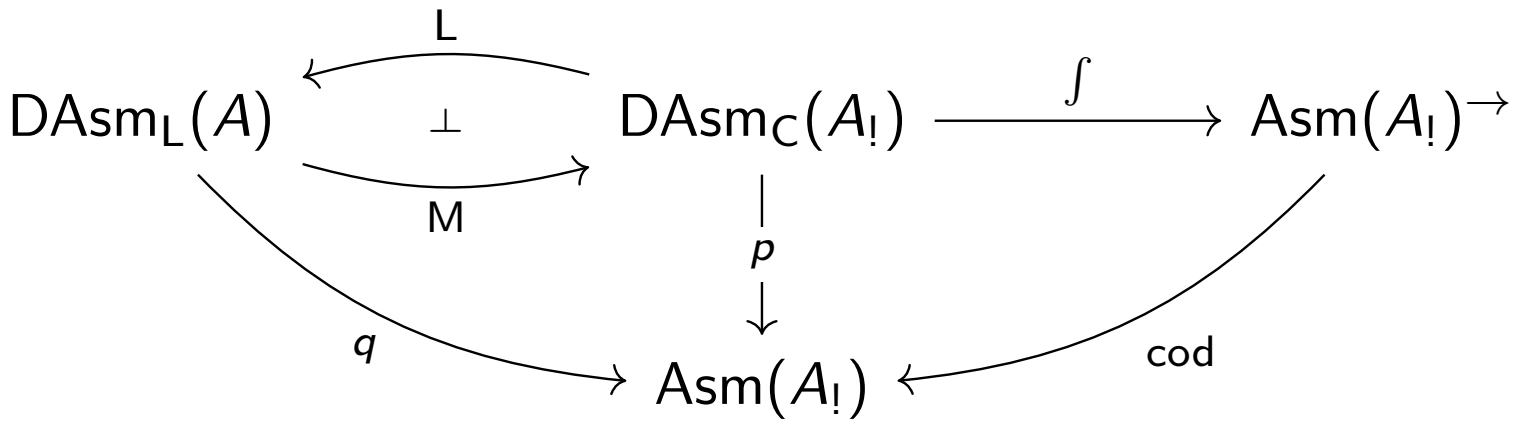
# Recall: Linear Comprehension Categories

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



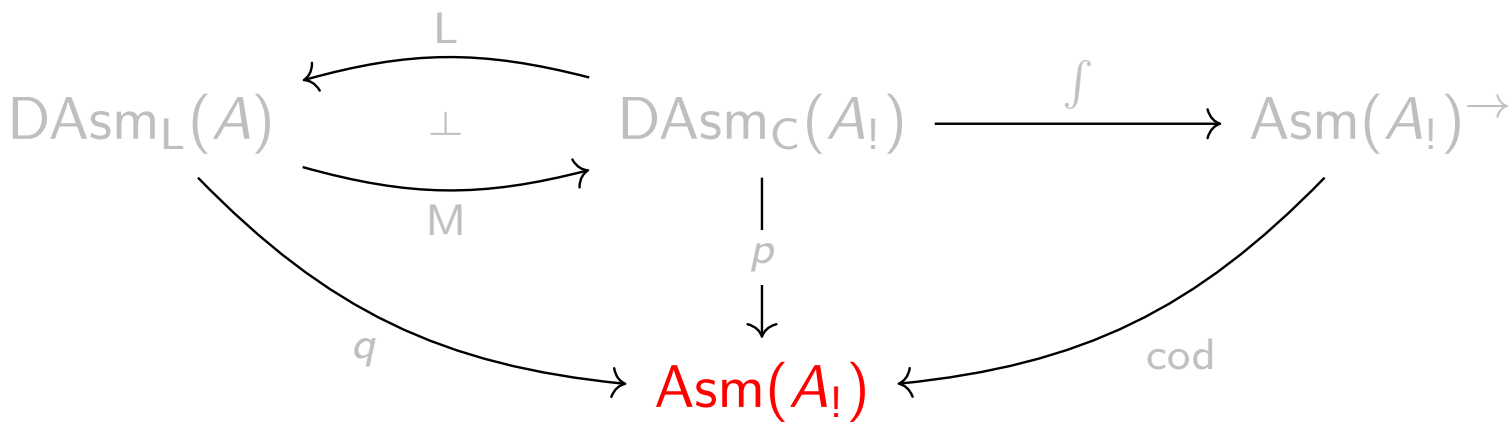
# The Linear Realizability Model

Let $A$ be a linear combinatory algebra



# The Linear Realizability Model: Assemblies

Let $A$ be a linear combinatory algebra



# Assemblies

We look at assemblies for the combinatory algebra $A_!$

## 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 $a \Vdash x$.

# Assemblies

We look at assemblies for the combinatory algebra $A_!$

## 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 $(X, \Vdash_X)$ to an assembly $(Y, \Vdash_Y)$ is a map $f : X \to Y$ such that there is $e : A$ with $e \cdot !a \Vdash f(x)$ whenever $a \Vdash x$.

# Assemblies

We look at assemblies for the combinatory algebra $A_!$

## 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 $a \Vdash x$.

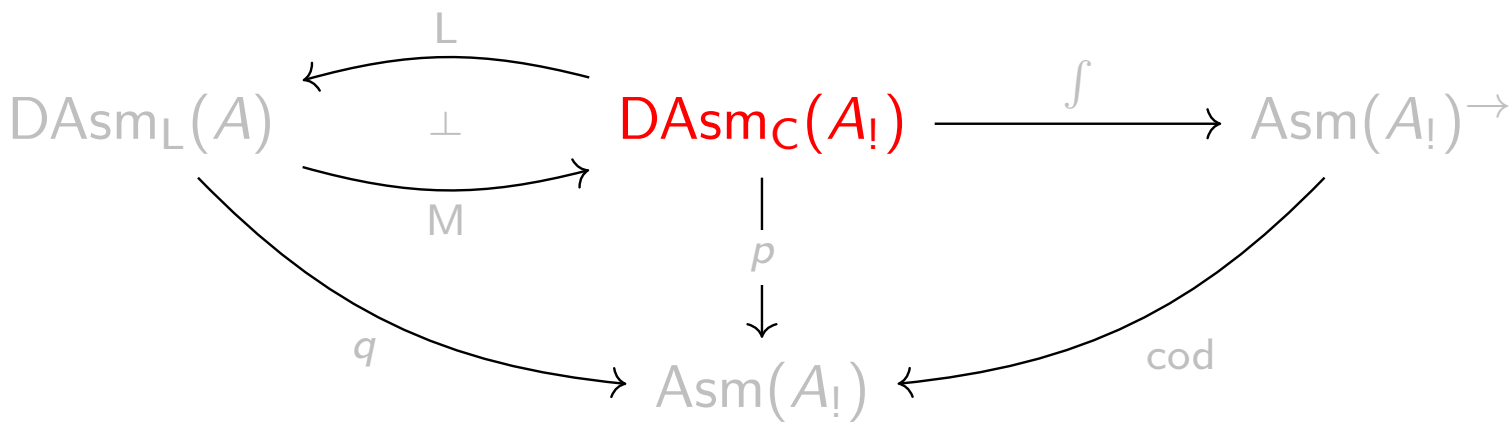
## Definition

A **morphism** from an assembly $(X, \Vdash_X)$ to an assembly $(Y, \Vdash_Y)$ is a map $f : X \to Y$ such that there is $e : A$ with $e \cdot !a \Vdash f(x)$ whenever $a \Vdash x$.

This gives rise to a category $\mathrm{Asm}(A_!)$

# The Linear Realizability Model: Families of Assemblies

Let $A$ be a linear combinatory algebra



# Families of Assemblies

## Definition

Let $\Gamma$ be an assembly for $A_!$. A **family of assemblies** $Y$ over $\Gamma$ consists of an assembly $Y_x$ for each $x : \Gamma$.

# Families of Assemblies

## Definition

Let $\Gamma$ be an assembly for $A_!$. A **family of assemblies** $Y$ over $\Gamma$ consists of an assembly $Y_x$ for each $x : \Gamma$.

## Definition

Let $s : \Gamma \to \Delta$ be a map of assemblies and let $Y$ and $Y'$ be families of assemblies over $\Gamma$ and $\Delta$ respectively. A **morphism over** $s$ consists of maps $g_x : Y_x \to Y'_{s(x)}$ such that there is $e : A$ with $e \cdot !a \cdot !b \Vdash g_x(y)$ whenever $a \Vdash x$ and $b \Vdash y$.

# Families of Assemblies

## Definition

Let $\Gamma$ be an assembly for $A_!$. A **family of assemblies** $Y$ over $\Gamma$ consists of an assembly $Y_x$ for each $x : \Gamma$.

## Definition

Let $s : \Gamma \to \Delta$ be a map of assemblies and let $Y$ and $Y'$ be families of assemblies over $\Gamma$ and $\Delta$ respectively. A **morphism over** $s$ consists of maps $g_x : Y_x \to Y'_{s(x)}$ such that there is $e : A$ with $e \cdot !a \cdot !b \Vdash g_x(y)$ whenever $a \Vdash x$ and $b \Vdash y$.

We get a category $\mathrm{DAsm}_{\mathsf{C}}(A_!)$ whose objects consists of assemblies $\Gamma$ and families of assemblies over $\Gamma$, and we have a functor $p : \mathrm{DAsm}_{\mathsf{C}}(A_!) \to \mathrm{Asm}(A_!)$.

# Type Theory via Assemblies

The interpretation of Martin-Löf type theory via assemblies is as follows:

- Contexts: assemblies $X$ over $A_!$

# Type Theory via Assemblies

The interpretation of Martin-Löf type theory via assemblies is as follows:

- Contexts: assemblies $X$ over $A_!$
- Types in context $X$: for each $x \in X$ an assembly $Y_x$ over $A_!$

# Type Theory via Assemblies

The interpretation of Martin-Löf type theory via assemblies is as follows:

- Contexts: assemblies $X$ over $A_!$
- Types in context $X$: for each $x \in X$ an assembly $Y_x$ over $A_!$
- Terms in context $X$ of type $Y$: a function $f$ assigning to $x \in X$ an element $f(x) \in Y_x$, such there exists $e : A$ with $e \cdot !a \Vdash f(x)$ whenever $a \Vdash x$

# Type Theory via Assemblies

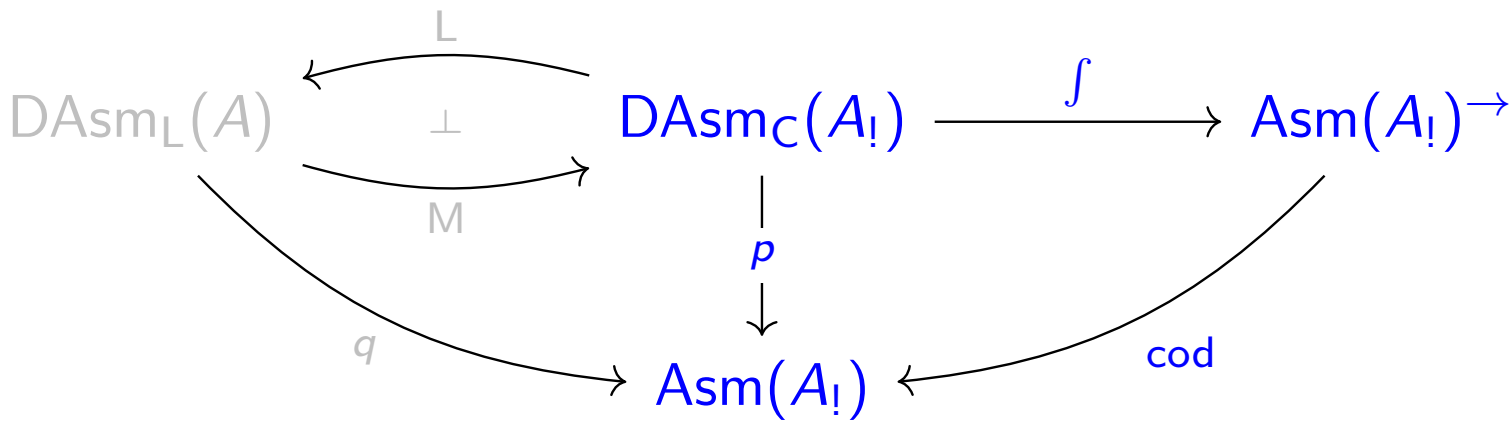
The interpretation of Martin-Löf type theory via assemblies is as follows:

- Contexts: assemblies $X$ over $A_!$
- Types in context $X$: for each $x \in X$ an assembly $Y_x$ over $A_!$
- Terms in context $X$ of type $Y$: a function $f$ assigning to $x \in X$ an element $f(x) \in Y_x$, such there exists $e : A$ with $e \cdot !a \Vdash f(x)$ whenever $a \Vdash x$

This gives us an interpretation of **extensional type theory with $\prod$ and $\sum$-types**.

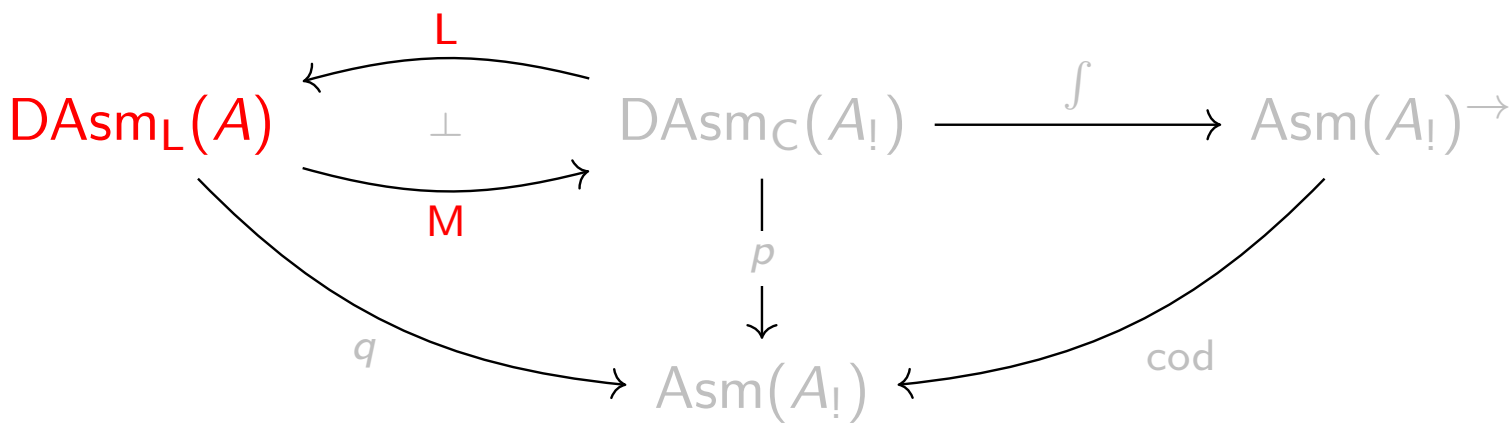
# The Linear Realizability Model: Cartesian Fragment

Let $A$ be a linear combinatory algebra



# The Linear Realizability Model: Linear Fragment

Let $A$ be a linear combinatory algebra



# Interpretation of Linear Types and Terms

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

# Interpretation of Linear Types and Terms

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

**Interpretation of linear terms**: Suppose we have

- an assembly $\Gamma$ 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 $e : A$ such that $e \cdot !b_1 \cdot b_2 \Vdash f_x(\overline{x})$ whenever $b_1 \Vdash x$ and $b_2 \Vdash \overline{x}$ with $\overline{x} : X_x$

# Interpretation of Linear Types and Terms

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

**Interpretation of linear terms**: Suppose we have

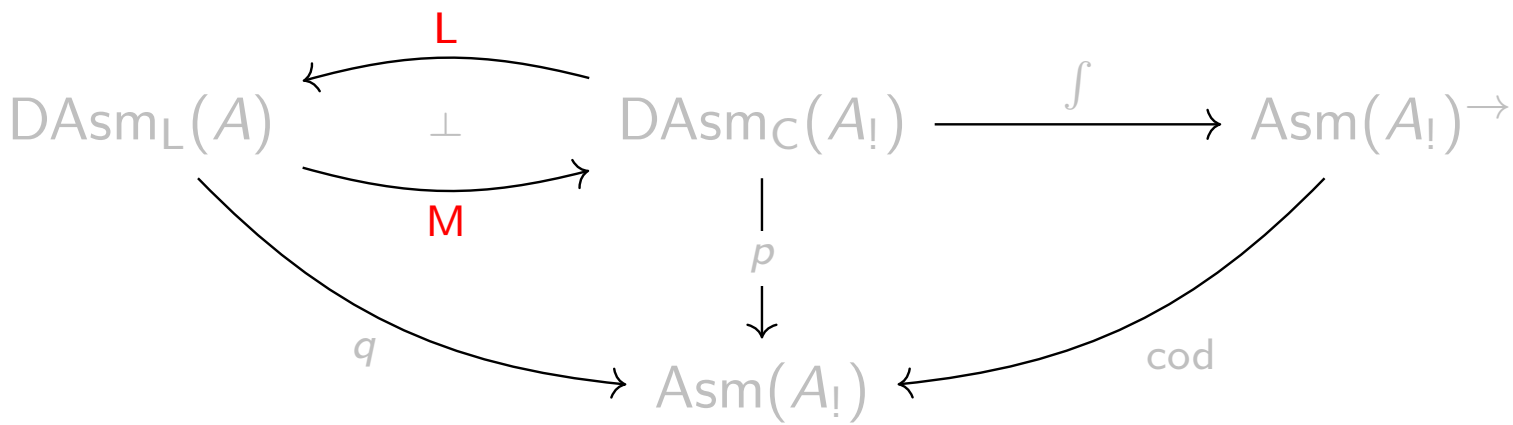
- an assembly $\Gamma$ over $A_!$
- for each $x : \Gamma$ assemblies $X_x$ and $Y_x$ over $A$

We interpret a linear term as

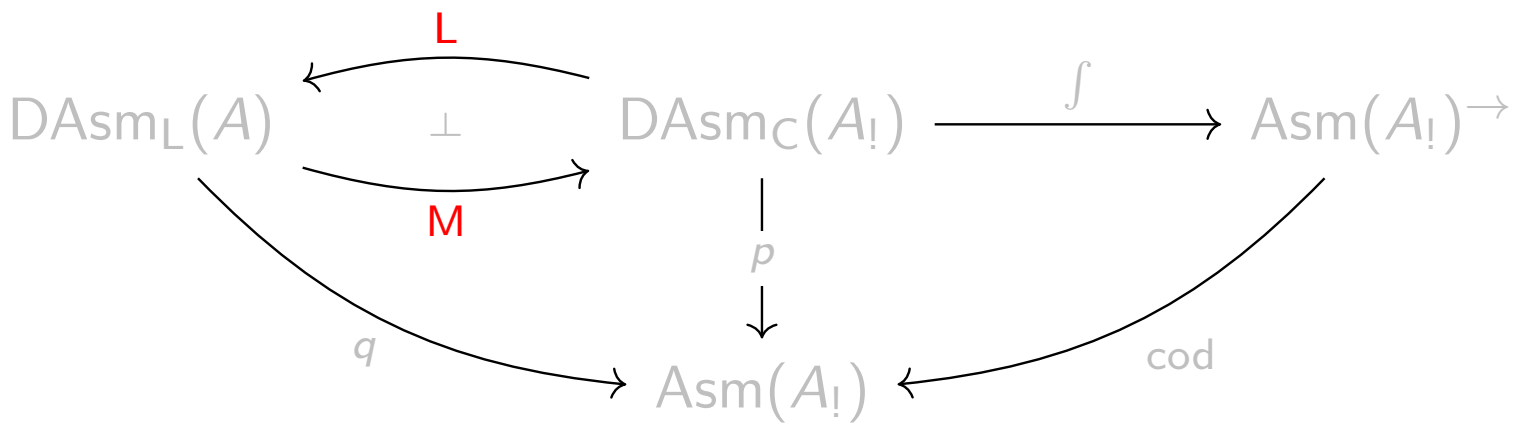
- maps $f_x : X_x \to Y_x$ for each $x : \Gamma$
- for which there exists $e : A$ such that $e \cdot !b_1 \cdot b_2 \Vdash f_x(\overline{x})$ whenever $b_1 \Vdash x$ and $b_2 \Vdash \overline{x}$ with $\overline{x} : X_x$

Note: this gives a category $\mathsf{DAsm}_\mathsf{L}(A)$ and a functor $q : \mathsf{DAsm}_\mathsf{L}(A) \to \mathsf{Asm}(A_!)$

# The Linear Non-Linear Adjunction Model

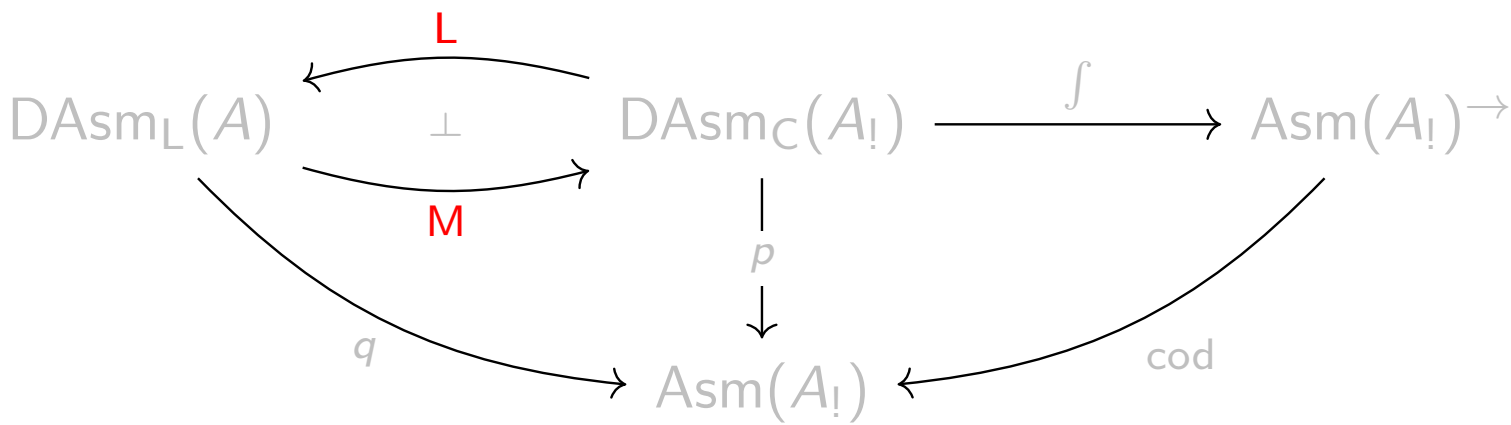


# The Linear Non-Linear Adjunction Model




M is the identity on objects

# The Linear Non-Linear Adjunction 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 $b \Vdash \overline{x}$

# Impredicativity

A key feature of assembly models is that they support an **impredicative universe** of sets. To construct this universe, we take the following steps

- Carve out a **class** of assemblies that will be contained in that universe: **modest sets**
- Find a **set** that represents all modest sets: the set of **partial equivalence relations**
- The impredicative universe is given by the assembly of partial equivalence relations

# Modest Sets and PERs

Recall that we fixed a combinatory algebra $A$

## Definition

An assembly $X$ is said to be **modest** if for all $x, x' \in X$ with $a \Vdash x$ and $a \Vdash x'$, we have $x = x'$.

[13] *A small complete category*, Hyland

# Modest Sets and PERs

Recall that we fixed a combinatory algebra $A$

## Definition

An assembly $X$ is said to be **modest** if for all $x, x' \in X$ with $a \Vdash x$ and $a \Vdash x'$, we have $x = x'$.

## Definition

A **partial equivalence relation (PER)** is a relation on $A$ that is symmetric and transitive.

[13] *A small complete category*, Hyland

# Modest Sets and PERs

Recall that we fixed a combinatory algebra $A$

## Definition

An assembly $X$ is said to be **modest** if for all $x, x' \in X$ with $a \Vdash x$ and $a \Vdash x'$, we have $x = x'$.

## Definition

A **partial equivalence relation (PER)** is a relation on $A$ that is symmetric and transitive.

**Note**:

- We have a **set** of partial equivalence relations
- We have an equivalence between the categories of modest sets and of PERs[13]
- PERs form an impredicative universe

[13] *A small complete category*, Hyland

# 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 $A$
- $\mathsf{El}_{\mathsf{C}}$ and $\mathsf{El}_{\mathsf{L}}$: modest sets and PERs are equivalent

# Table of Contents

# Many Shapes of LNL Models

There are several variations of LNL models in the literature

- LNL models (categories)
- LNL hyperdoctrines[14] (fibrations)

Linear comprehension categories are LNL hyperdoctrine with additional structure
Our study of realisability models motivate internal LNL models (internal categories)

[14] *Linear realizability and full completeness for typed lambda-calculi*, Abramsky, Lenise

# Many Shapes of LNL Models

There are several variations of LNL models in the literature

- LNL models (categories)
- LNL hyperdoctrines[14] (fibrations)

Linear comprehension categories are LNL hyperdoctrine with additional structure

Our study of realisability models motivate internal LNL models (internal categories)

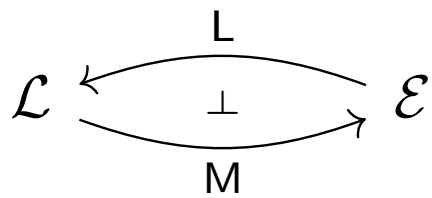
**Goal**: find a single notion that generalises each of these notions of model

[14] *Linear realizability and full completeness for typed lambda-calculi*, Abramsky, Lenise

# Main idea

**Key observation**: LNL models can be defined internal to a 2-category with finite products

A **linear non-linear model** consists of functors as follows




**Requirements**:

- L is a symmetric monoidal closed category
- M is a Cartesian monoidal category
- $\mathsf{L} \dashv \mathsf{M}$ is a symmetric monoidal adjunction

# Cartesian Objects

Let $\mathcal{B}$ be a 2-category with finite products.

## Definition

A **Cartesian object** is an object $x$ such that the 1-cells $\Delta : x \to x \times x$ and $! : x$ have right adjoints.

$$x \times x \underset{m}{\overset{\Delta}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} x \underset{u}{\overset{!}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} 1$$

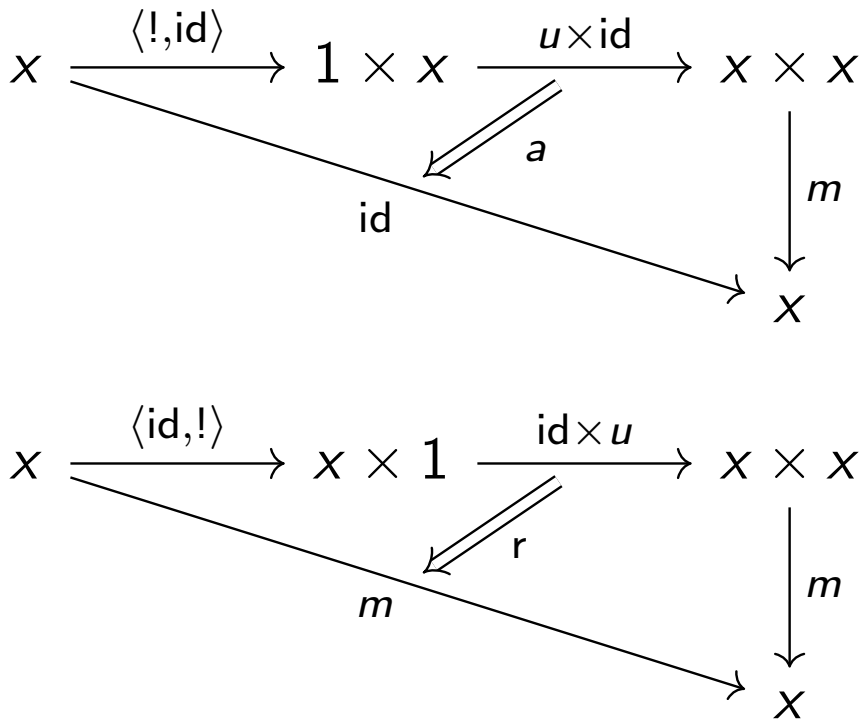
# Pseudomonoids

Let $\mathcal{B}$ be a 2-category with finite products.

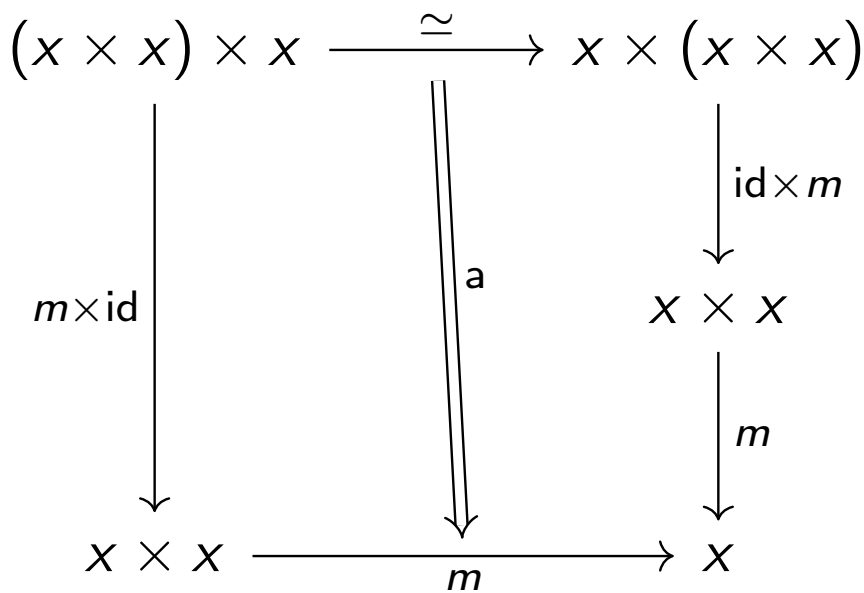
## Definition

A **pseudomonoid** consists an object $x$ of together with 1-cells $u : 1 \to x$ and $m : x \times x \to x$ together with invertible 2-cells as displayed in the next slides.

# Pseudomonoids: Unitors



# Pseudomonoids: Associator

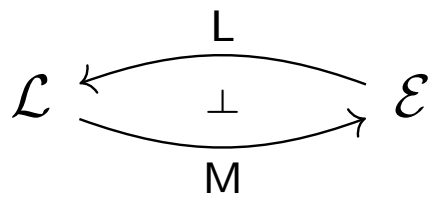


# Pseudomonoids

- We also require various coherence conditions analogous to monoidal categories
- We can also define symmetric pseudomonoids

# Internal LNL Adjunctions

An **internal LNL adjunction** consists of **1-cells** as follows




**Requirements**:

- L is a symmetric pseudomonoid
- M is a Cartesian object
- L $\dashv$ M is a symmetric monoidal adjunction

# Applications of Internal LNL Adjunctions

Instantions of internal LNL adjunctions:

- for categories: LNL models
- for fibered categories: LNL hyperdoctrines
- for internal categories: internal LNL models

# Applications of Internal LNL Adjunctions

Instantions of internal LNL adjunctions:

- for categories: LNL models
- for fibered categories: LNL hyperdoctrines
- for internal categories: internal LNL models

**Observe**:

- If a pseudofunctors preserves finite products, then it also preserves internal LNL adjunctions
- For example: **externalisation of internal categories**

# 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`