

Strictly Associative Sigmas (Work In Progress)

Emmanuel (Emma) Suárez Acevedo

MURI Meeting (2024)

Martin-Löf Type Theory (MLTT)

There are the following judgements:

- ▶ **Contexts:** $\vdash \Gamma \text{ cx}$
- ▶ **Types:** $\Gamma \vdash A \text{ type}$
- ▶ **Substitutions:** $\Delta \vdash \gamma : \Gamma$
- ▶ **Terms:** $\Gamma \vdash a : A$

Of particular interest to this talk are the **Unit** type and Σ -types:

$$\frac{\vdash \Gamma \text{ cx}}{\Gamma \vdash \mathbf{Unit} \text{ type}} \qquad \frac{\vdash \Gamma \text{ cx}}{\Gamma \vdash \mathbf{tt} : \mathbf{Unit}} \qquad \frac{\Gamma \vdash A \text{ type} \quad \Gamma.A \vdash B \text{ type}}{\Gamma \vdash \Sigma(A, B) \text{ type}}$$
$$\frac{\Gamma \vdash a : A \quad \Gamma.A \vdash B \text{ type} \quad \Gamma \vdash b : B[\mathbf{id}.a]}{\Gamma \vdash \mathbf{pair}(a, b) : \Sigma(A, B)}$$

Idea

Suggested by Favonia, Carlo Angiuli, and Jon Sterling: **What if we make Σ -types unital and associative?**

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(\mathbf{Unit}, A) = A \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(A, \mathbf{Unit}) = A \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A.B \vdash C \text{ type}}{\Gamma \vdash \Sigma(A, \Sigma(B, C)) = \Sigma(\Sigma(A, B), C) \text{ type}}$$

Consequences?

- ▶ Consistency?
- ▶ Normalization?
- ▶ Elaboration? (i.e. to develop a proof assistant)

Motivation

1. Usability of proof assistants
2. Curiosity?

Motivation

1. **Usability of proof assistants**
2. Curiosity?

Example

$\text{Poset} : \text{Set} \rightarrow \text{Set}_1$

$\text{Poset } X = \Sigma[_ \leq _ \in (X \rightarrow X \rightarrow \text{Set})]$

- reflexivity

$(\forall x \rightarrow x \leq x) \times$

- antisymmetry

$(\forall x y \rightarrow x \leq y \rightarrow y \leq x \rightarrow x \equiv y) \times$

- transitivity

$(\forall x y z \rightarrow x \leq y \rightarrow y \leq z \rightarrow x \leq z)$

Example

Semigroup : Set \rightarrow Set

Semigroup S = $\Sigma[_+_ \in (S \rightarrow S \rightarrow S)]$

- $_+_$ is associative

$$(\forall s_1 s_2 s_3 \rightarrow s_1 + (s_2 + s_3) \equiv (s_1 + s_2) + s_3)$$

Monoid : Set \rightarrow Set

Monoid M = $\Sigma[(_+_ , _) \in \text{Semigroup } M]$

$\Sigma[e \in M]$

- e is a left and right identity

$$(\forall m \rightarrow e + m \equiv m) \times$$

$$(\forall m \rightarrow m + e \equiv m)$$

Example

- Poset X : $_ \leq _ \times \leq / \text{refl} \times \leq / \text{antisym} \times \leq / \text{trans}$
- Semigroup S : $_ + _ \times + / \text{assoc}$
- Monoid M : $\text{semigrp} \times e \times + / \text{identity}^l \times + / \text{identity}^r$

PoMonoid $M = \Sigma[((_ \cdot _, _), _, _, _) \in \text{Monoid } M]$
 $\Sigma[(_ \leq _, _) \in \text{Poset } M]$
- compatibility of $_ \cdot _$ with $_ \leq _$
 $(\forall x y z \rightarrow x \leq y \rightarrow (x \cdot z) \leq (y \cdot z)) \times$
 $(\forall x y z \rightarrow x \leq y \rightarrow (z \cdot x) \leq (z \cdot y))$

Example

- Poset $X : _ \leq _ \times \leq / \text{refl} \times \leq / \text{antisym} \times \leq / \text{trans}$
- Semigroup $S : _ + _ \times + / \text{assoc}$
- Monoid $M : \text{semigrp} \times e \times + / \text{identity}^1 \times + / \text{identity}^r$
- PoMonoid $M : \text{monoid} \times \text{poset} \times + / \text{compat}^r \times + / \text{compat}^1$

prop : $\forall \{M\}$
 $(((((_ + _, _) , e , _, _) , (_ \leq _, _, _, _)) , _, _)) : \text{PoMonoid } M)$
 $\rightarrow \forall m \rightarrow (e + m) \leq m$

prop ...

Example

- Mental overhead to use the components of the pomonoid

- Poset $X : _ \leq _ \times \leq / \text{refl} \times \leq / \text{antisym} \times \leq / \text{trans}$
- Semigroup $S : _ + _ \times + / \text{assoc}$
- Monoid $M : \text{semigrp} \times e \times + / \text{identity}^l \times + / \text{identity}^r$
- PoMonoid $M : \text{monoid} \times \text{poset} \times + / \text{compat}^r \times + / \text{compat}^l$

`prop` : $\forall \{M\}$
 $((((_ + _, _) , e , _, _) , (_ \leq _, _, _, _) , _, _) : \text{PoMonoid } M)$
 $\rightarrow \forall m \rightarrow (e + m) \leq m$
`prop` ...

Example

- What if we had strictly associative sigmas?

- Poset $X : _ \leq _ \times \leq / \text{refl} \times \leq / \text{antisym} \times \leq / \text{trans}$
- Semigroup $S : _ + _ \times + / \text{assoc}$
- Monoid $M : \text{semigrp} \times e \times + / \text{identity}^l \times + / \text{identity}^r$
- PoMonoid $M : \text{monoid} \times \text{poset} \times + / \text{compat}^r \times + / \text{compat}^l$

`prop` : $\forall \{M\}$
 $((_ + _, _, e, _, _, _ \leq _, _, _, _, _)) : \text{PoMonoid } M)$
 $\rightarrow \forall m \rightarrow (e + m) \leq m$
`prop` ...

Motivation

1. Usability of proof assistants ✓
 - ▶ e.g. reduces mental overhead when dealing with nested sums
2. Curiosity?

Motivation

1. Usability of proof assistants ✓
 - ▶ e.g. reduces mental overhead when dealing with nested sums
2. **Curiosity?**

Related work

Type theory is a polynomial pseudomonad and polynomial pseudoalgebra (Awodey and Newstead 2018)

Review: polynomials

Let \mathcal{E} be a locally cartesian closed category (lccc).

A **polynomial** $p : I \rightarrowtail J = (s, f, t)$ in \mathcal{E} is a diagram of the form:

$$\begin{array}{ccc} & B & \xrightarrow{f} A \\ & \swarrow s & \searrow t \\ I & & J \end{array}$$

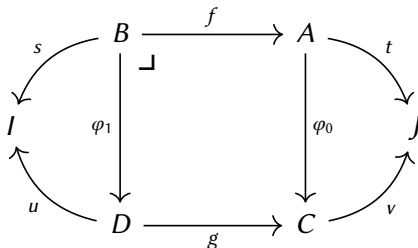
Every morphism $f : B \rightarrow A$ in \mathcal{E} is a polynomial $\mathbf{1} \rightarrowtail \mathbf{1}$ (taking s and t to be the unique morphisms to the terminal object $\mathbf{1}$ of \mathcal{E})

For any object I , the **identity polynomial** $i_I : I \rightarrowtail I$ is $(\text{id}_I, \text{id}_I, \text{id}_I)$

Review: polynomials

A **morphism of polynomials** $\varphi : p \rightrightarrows q$ is an object D_φ and a triplet of morphisms $(\varphi_0, \varphi_1, \varphi_2)$

φ is **cartesian** if φ_2 is invertible, in which case it is uniquely represented by the following diagram:



with $p = (s, f, t)$ and $g = (u, g, v)$

Review: polynomials

Recall any morphism in \mathcal{E} can be considered as a polynomial $\mathbf{1} \rightarrow \mathbf{1}$.

For two morphisms $p : B \rightarrow A$ and $q : D \rightarrow C$, a cartesian morphism $\varphi : p \Rightarrow q$ can be further simplified to the following pullback square:

$$\begin{array}{ccc} B & \xrightarrow{f} & A \\ \varphi_1 \downarrow \lrcorner & & \downarrow \varphi_0 \\ D & \xrightarrow{g} & C \end{array}$$

Review: natural models

A **natural model** of type theory is a category \mathbb{C} along with:

- ▶ a terminal object \diamond
- ▶ a *representable* map of presheaves $p : \dot{U} \rightarrow U$ on \mathbb{C}

Review: natural model and polynomials

- ▶ $p : \dot{U} \rightarrow U$ is a morphism in the lccc $\mathbf{Set}^{\mathbf{C}^{\text{op}}}$
- ▶ p can be considered a polynomial $\mathbf{1} \rightarrow \mathbf{1}$ in $\mathbf{Set}^{\mathbf{C}^{\text{op}}}$
- ▶ The conditions for the natural model to support unit and dependent sum types can be phrased in terms of morphisms of polynomials

Review: natural model and **Unit** type

The model supports unit types iff there exists a cartesian morphism $\eta : i_1 \rightrightarrows p$. Diagrammatically:

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{\eta_1} & \dot{U} \\ \parallel & \lrcorner & \downarrow p \\ \mathbf{1} & \xrightarrow{\eta_0} & U \end{array}$$

Review: natural model and Σ -types

The model supports dependent sum types iff there exists a cartesian morphism $\mu : p \cdot p \Rightarrow p$. Diagrammatically:

$$\begin{array}{ccc} \sum_{A:U} \sum_{B:U^A} \sum_{a:A} B(a) & \xrightarrow{\mu_1} & \dot{U} \\ \downarrow p \cdot p & \lrcorner & \downarrow p \\ \sum_{A:U} U^A & \xrightarrow{\mu_0} & U \end{array}$$

Review: polynomial monad

A **polynomial monad** is a quadruple (I, p, η, μ) consisting of:

- ▶ an object I of \mathcal{E}
- ▶ a polynomial $p : I \nrightarrow I$ in \mathcal{E}
- ▶ cartesian morphisms $\eta : i_I \rightrightarrows p$ and $\mu : p \cdot p \rightrightarrows p$ satisfying the usual monad axioms (e.g. $\mu \circ (p \cdot \eta) = \text{id}_p$)

Review: natural model and polynomial monads

Is $(\mathbf{1}, p, \eta, \mu)$ a polynomial monad? In particular, does it satisfy the usual monad laws?

For example:

$$\mu \circ (p \cdot \eta) = \text{id}_p$$

$$\mu \circ (\eta \cdot p) = \text{id}_p$$

Review: natural model and polynomial monads

Is $(\mathbf{1}, p, \eta, \mu)$ a polynomial monad? In particular, does it satisfy the usual monad laws?

For example:

$$\mu \circ (p \cdot \eta) = \text{id}_p$$

$$\mu \circ (\eta \cdot p) = \text{id}_p$$

No — this would correspond to $\Sigma(\mathbf{Unit}, A)$ being equal to A and $\Sigma(A, \mathbf{Unit})$ being equal to A , which is not the case in MLTT.

Review: natural model and polynomial monads

Is $(\mathbf{1}, p, \eta, \mu)$ a polynomial monad? In particular, does it satisfy the usual monad laws?

For example:

$$\mu \circ (p \cdot \mu) = \mu \circ (\mu \cdot p)$$

Review: natural model and polynomial monads

Is $(1, p, \eta, \mu)$ a polynomial monad? In particular, does it satisfy the usual monad laws?

For example:

$$\mu \circ (p \cdot \mu) = \mu \circ (\mu \cdot p)$$

No — this would correspond to $\Sigma(A, \Sigma(B, C))$ being equal to $\Sigma(\Sigma(A, B), C)$, which is not the case in MLTT.

Review: natural model and polynomial monads

Dependent type theories admitting a unit type and dependent sum types give rise to a polynomial *pseudomonad*. (Awodey and Newstead 2018)

- ▶ On the other hand, if $(\mathbf{1}, p, \eta, \mu)$ were a polynomial monad — this model would seem to have a correspondence with MLTT with unital and associative Σ -types.

Motivation

1. Usability of proof assistants ✓
 - ▶ e.g. reduces mental overhead when dealing with nested sums
2. Curiosity? ✓
 - ▶ e.g. learning more about type theory as a polynomial monad

Σ -types are unital

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(\mathbf{Unit}, A) = A \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(A, \mathbf{Unit}) = A \text{ type}}$$

Σ -types are unital

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(\mathbf{Unit}, \mathbf{A}) = A \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(A, \mathbf{Unit}) = A \text{ type}}$$

Σ -types are unital

$$\frac{\vdash \Gamma \text{ cx}}{\vdash \Gamma.\mathbf{Unit} = \Gamma \text{ cx}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(\mathbf{Unit}, A) = A \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \Sigma(A, \mathbf{Unit}) = A \text{ type}}$$

Σ -types are associative

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A.B \vdash C \text{ type}}{\Gamma \vdash \Sigma(A, \Sigma(B, C)) = \Sigma(\Sigma(A, B), C) \text{ type}}$$

Σ -types are associative

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A.B \vdash C \text{ type}}{\Gamma \vdash \Sigma(A, \Sigma(B, C)) = \Sigma(\Sigma(A, B), \mathbf{C}) \text{ type}}$$

Σ -types are associative

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma.A \vdash B \text{ type}}{\vdash \Gamma.\Sigma(A, B) = \Gamma.A.B \text{ cx}}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma.A \vdash B \text{ type} \quad \Gamma.A.B \vdash C \text{ type}}{\Gamma \vdash \Sigma(A, \Sigma(B, C)) = \Sigma(\Sigma(A, B), C) \text{ type}}$$

Context equations?

What does this mean for elaboration?

- ▶ e.g. synthesizing a type for a variable preterm (with the usual de Bruijn index representation)

$$\frac{}{\mathbf{1.Nat.Nat} \vdash (\text{var } 0) \Rightarrow \textcolor{red}{??} \rightsquigarrow \mathbf{q}}$$

Context equations?

What does this mean for elaboration?

- ▶ Synthesizing a type for a variable (with the usual de Bruijn index representation)

$$\frac{}{\mathbf{1.Nat.Nat} \vdash (\text{var } 0) \Rightarrow \mathbf{Nat} \rightsquigarrow \mathbf{q}}$$

$$\frac{}{\mathbf{1.Nat.Nat} \vdash (\text{var } 0) \Rightarrow \Sigma(\mathbf{Nat}, \mathbf{Nat}) \rightsquigarrow \mathbf{q}}$$

- ▶ No longer deterministic! This is an issue even if we change variables to be checked

Context equations?

What does this mean for normalization?

- ▶ Contexts have normal forms!
- ▶ An algorithm for normalization (e.g. NbE) now must first normalize the context

Simply-typed lambda calculus

What about in the simpler setting of the simply-typed lambda calculus (STLC)?

Context equations:

$$\frac{\vdash \Gamma \text{ cx}}{\vdash \Gamma.\mathbf{Unit} = \Gamma \text{ cx}}$$

$$\frac{\vdash \Gamma \text{ cx} \quad A \text{ type} \quad B \text{ type}}{\vdash \Gamma.(A * B) = \Gamma.A.B \text{ cx}}$$

Simply-typed lambda calculus

What about in the simpler setting of the simply-typed lambda calculus (STLC)?

Context equations:

$$\frac{\vdash \Gamma \text{ cx}}{\vdash \Gamma.\mathbf{Unit} = \Gamma \text{ cx}} \qquad \frac{\vdash \Gamma \text{ cx} \quad A \text{ type} \quad B \text{ type}}{\vdash \Gamma.(A * B) = \Gamma.A.B \text{ cx}}$$

- ▶ The context equations have the same effect on normalization and elaboration!

Current and future work

- ▶ **NbE for STLC with unital and associative product types (including context equations)**
- ▶ Elaboration for STLC with unital and associative product types (including context equations)
- ▶ Adapt both for MLTT
- ▶ Learn more about relationship of the type theory with the natural model with $(\mathbf{1}, p, \eta, \mu)$ as a polynomial monad

Thank you!

- ▶ Questions?

References:

- ▶ Awodey, S. (2018). Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28(2), 241-286.
- ▶ Awodey, S. and Newstead, C. (2018). Polynomial pseudomonads and dependent type theory. arXiv preprint [arXiv:1802.00997](https://arxiv.org/abs/1802.00997).