Deriving Hypercubes of Type Systems

Michael Stay L. Gregory Meredith Christian Wells

The question

Barendregt's Lambda Cube starts with untyped lambda calculus and varies how terms and types may depend on one another.

Question

Can other operational calculi generate analogous families of type systems?

Example target: asynchronous communication in the π-calculus.

Main thesis

The dependent product is generated by an operational site.

- In lambda calculus: the function position of the beta redex.
- In other calculi: any declared source occurrence of a redex.
- The result: a hypercube of modal dependent typing disciplines.

Barendregt's dependent product rules

Fix a set of allowed product profiles

$$\mathcal{S} \subseteq \{*, \Box\} \times \{*, \Box\}.$$

Each pair $(s_1, s_2) \in \mathcal{S}$ licenses a dependent product whose domain has sort s_1 and whose codomain has sort s_2.

Formation

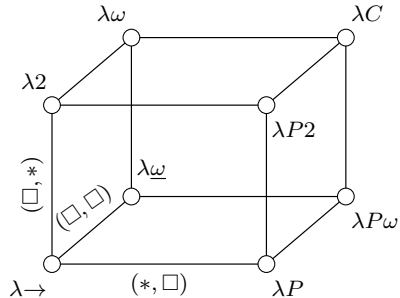
$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi_{x:A} B : s_2}\ \Pi\text{-FORM}$$

Introduction

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad \Gamma, x : A \vdash C : B}{\Gamma \vdash \lambda x : A.C : \Pi_{x:A} B}\ \Pi\text{-INTRO}$$

Different choices of $\mathcal{S}$ give the systems of the Lambda Cube.

The cube



Profile	Reading	Dependency
$(*, *)$	ordinary functions	terms on terms
$(*, \Box)$	dependent types	types on terms
$(\Box, *)$	polymorphism	terms on types
$(\Box, \Box)$	type operators	types on types

Where does Π come from?

A function is useful because application exposes a computation.

$$\mathsf{app}(\mathsf{lam}(t), a) \rightsquigarrow t(a)$$

Selected source subterm

$$U = \mathsf{lam}(t)$$

Surrounding context

$$K[-] = \mathsf{app}([-], a)$$

Why modal?

Ordinary product elimination suggests type preservation.

$$f : \Pi_{x:A} B, \quad a : A \quad \rightsquigarrow \quad f\, a : B[a/x]$$

For general operational theories, we usually only get a one-step may property:

Modal reading

$$P \rightsquigarrow Q \quad \text{and} \quad Q :: C \quad \text{imply} \quad P :: \Diamond C$$

P may step to something of type C.

Input: finite operational presentations

Presentation data

- carriers: $\mathrm{Pr}, \mathrm{R}, \ldots$
- term constructors
- structural equations
- primitive rewrite constructors

Extra design data

- displayed source sites
- profile-matching constraints
- allowed local profiles
- closure support, when needed

Example operational theory: lambda calculus

Wrapped syntax

$$\mathsf{app} : \mathrm{Pr} \times \mathrm{Pr} \to \mathrm{Pr}, \qquad \mathsf{lam} : \mathrm{Pr}^{\mathrm{Pr}} \to \mathrm{Pr}$$

Primitive beta step

$$p : \mathrm{Pr},\ k : \mathrm{Pr} \to \mathrm{Pr} \vdash \mathsf{app}(\mathsf{lam}(k), p) \rightsquigarrow k(p)$$

Chosen evaluation support

A head-context closure rule propagates rewrites through function position.

Example operational theory: asynchronous π

Core syntax

$$\begin{array}{rcl} \mathrm{nil} & : & \mathrm{Pr} \\ p \mid q & : & \mathrm{Pr} \\ n!m & : & \mathrm{Pr} \\ n?k & : & \mathrm{Pr} \\ !p,\ \nu k & : & \mathrm{Pr} \end{array}$$

Communication

$$n!m \mid n?k \rightsquigarrow k(m)$$

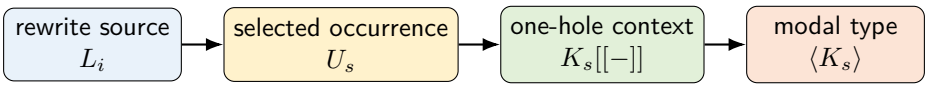
Concurrency changes the typing problem

A process may have many possible futures.

- Parallel composition is commutative and associative.
- A race has inequivalent outcomes.
- Useful properties are often about what may happen next.

$$\text{receive type} \neq \text{invariant type}$$

Declared sites



Key design choice

Sites are selected displayed occurrences, not all subterms modulo equations.

Three classes of site variables

For a site s, factor the source as $L_i = K_s[U_s]$.

Index

Shared by U_s and K_s.

$$\vec{x}_s = \mathrm{FV}(U_s) \cap \mathrm{FV}(K_s)$$

Rely

Supplied by the context only.

$$\vec{y}_s = \mathrm{FV}(K_s) \setminus \mathrm{FV}(U_s)$$

Ambient

Parameters of the selected term or target.

$\vec{z}_s$ stays outside

Comparison I: redex, site, context

	Lambda completion	π-calculus receive completion
Operational redex	$\mathsf{app}(\mathsf{lam}(t), a) \rightsquigarrow t(a)$	$n!m \mid n?k \rightsquigarrow k(m)$
Declared source site	function position $\mathsf{lam}(t)$	input occurrence $n?k$
Displayed context	$K[-] = \mathsf{app}([-], a)$	$K[-] = n!m \mid [-]$

Comparison II: site variables

	Lambda completion	π-calculus receive completion
Index variables	none for the product site	channel name n
Rely variables	argument a	communicated name m
Ambient variables	abstraction body $t : \mathrm{Pr} \to \mathrm{Pr}$	continuation $k : \mathrm{Nm} \to \mathrm{Pr}$

Reading

Index data remain fixed outside the modality; rely data are supplied when the context is supplied.

Comparison III: generated contextual type

Lambda

$$\Pi_{x:A} B$$

is shorthand for

$$\langle \mathsf{app}([-], x) \rangle_{x::A} B$$

Receive

$$\langle n!m \mid [-] \rangle_{m::B}^{n::A} C$$

An input process that can communicate with a matching output and reach C.

Comparison IV: introduction principles

Lambda abstraction

If the body has type B under $x :: A$, then:

$$\frac{x :: A \vdash t(x) :: B}{\mathsf{lam}^\sharp(A, t) :: \Pi_{x:A} B}$$

Input

Let $M = \langle n!m \mid [-] \rangle_{m::B}^{n::A} C$. If the continuation has type C, then:

$$\frac{n :: A,\ m :: B \vdash k(m) :: C}{n?_B k :: M}$$

Comparison V: elimination is effectful

Application

$$\frac{f :: \Pi_{x:A} B \qquad a :: A}{\mathsf{app}(f, a) :: \Diamond(B[a/x])}$$

Communication

Let $M = \langle n!m \mid [-] \rangle_{m::B}^{n::A} C$.

$$\frac{P :: M \qquad a :: B}{n!a \mid P :: \Diamond(C[a/m])}$$

Comparison VI: local hypercubes

Product site

Two profile slots:

$$(u, v) \in \{*, \square\}^2$$

Based choices give $2^{2^2-1} = 2^3$ calculi.

Receive site

Three carrier-sensitive slots:

$$(\alpha, \beta; \gamma) \in \{*, \square\}_n \times \{*, \square\}_m \times \{*, \square\}_C$$

Based choices give $2^{2^3-1} = 2^7$ calculi.

Profile computation

Profiles say which sort levels may appear in a site interface.

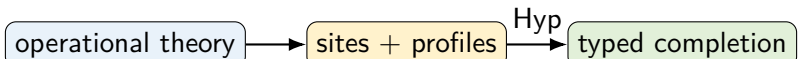
- Raw slots: index variables, rely variables, and target type.
- Matching data may identify slots.
- Quotient profiles are raw profiles constant on identified slots.
- A signature choice selects which quotient profiles are allowed.

Important separation

Structural equations transport rewrites; they do not silently move holes or identify profile slots.

Algorithm skeleton

1. Start with a finite operational presentation.
2. Choose displayed source sites.
3. Factor each selected source as $L_i = K_s[U_s]$.
4. Compute index, rely, and ambient telescopes.
5. Compute quotient profile spaces and choose signatures.
6. Freely add modal type formers and rules.



What the completion adds

New syntax

- sort constants $*^X, \Box^X$
- typing evidence $P :: A$
- possibility $\Diamond C$
- contextual modalities $\langle K_s \rangle$

New rules

- formation and introduction
- modal elimination
- source-subterm introduction
- computation rules for canonical sources

One contextual type former per site; profiles select allowed rule instances.

Lambda completion: nearly the Lambda cube

Product-site completion

$$\Pi(A, \lambda x.B) := \langle \mathsf{app}([-], x) \rangle_{x::A} B$$

- The constructor fragment recovers the product-profile geometry.
- Erasing $\Diamond$ maps it to the corresponding Barendregt system.
- The full modal theory is finer and more operational.

$$(\Diamond^n C)^\circ = C^\circ \qquad \mathsf{app}(f, a) :: \Diamond(B[a/x])$$

Receive completion: a communication type

Receive-site completion

$$n?_B k :: \langle n!m \mid [-] \rangle^{n::A}_{m::B} C$$

- The channel n is indexed.
- The message m is supplied by the context.
- The continuation k remains ambient.

When paired with a matching output, the input may take one communication step to C.

May-properties, not must-properties

Generated modal typing

$$P :: \Diamond C$$

means evidence for some:

$$P \rightsquigarrow Q \qquad Q :: C.$$

Must property

$$\forall Q.\ P \rightsquigarrow Q \Rightarrow Q \in C$$

This is model-theoretic and not generated by the local rules alone.

Correct by construction

The main metatheorems are formal consequences of the generated presentation.

Freeness

The completion adds exactly the chosen generated structure, with no hidden rules.

Functoriality

Morphisms of site-and-signature presentations map to morphisms of completions.

Soundness

Every derivation has proof-relevant evidence in every structured model.

Takeaways

1. Dependent products can be read as contextual may-types generated by the beta site.
2. The same recipe applies to non-lambda operational theories.
3. The π-calculus receive site yields a communication hypercube.
4. The algorithm separates displayed syntax, structural equations, and profile choices.
5. The categorical and semantic theorems certify the construction.

Questions?