Induction Is Not Derivable in Second Order Dependent Type Theory

Written by Herman (2001)

Presented by Hilde & Sonia (2025)

December 8, 2025

Overview

1. First Section

Before we dive in...

Element		**Set**		**Powerset**
$\llparenthesis - \rrparenthesis$	$\in$	$\llbracket - \rrbracket$	$\in$	$\mathcal{V}(-)$

Our Goal

Theorem: Non-derivability of ind in $\lambda P2$

We need to prove $\mathcal{M} \not\models^{\lambda P2}$ ind

Our Goal

Theorem: Non-derivability of ind in $\lambda P2$

We need to prove $\mathcal{M} \not\models^{\lambda P2}$ ind

Goal of the paper

Construct this counter-model $\mathcal{M}$

Our Goal

Theorem: Non-derivability of ind in $\lambda P2$

$\mathcal{M} \nVdash^{\lambda P2}$ ind

Goal of the presentations

Our Goal

Theorem: Non-derivability of ind in $\lambda P2$

$\mathcal{M} \not\Vdash^{\lambda P2}$ ind

Goal of the presentations

Presentation 1: We will discuss model construction

Our Goal

Theorem: Non-derivability of ind in $\lambda P2$

$\mathcal{M} \not\Vdash^{\lambda P2}$ ind

Goal of the presentations

Presentation 1: We will discuss model construction
Presentation 2: They will discuss the specific counter-model $\mathcal{M}$

Things we have seen before

Second order dependent type theory $\lambda P2$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

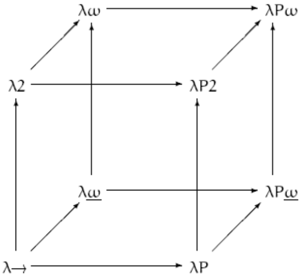

Figure: Pure type systems

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : T \Rightarrow \Gamma \models M : T$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : T \Rightarrow \Gamma \models M : T$

Logic: Syntactic $\Rightarrow$ Semantic

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : T \Rightarrow \Gamma \models M : T$

Logic: Syntactic $\Rightarrow$ Semantic
Type Theory: Derivation of the type $\Rightarrow$ Semantic evaluation is true

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : T \Rightarrow \Gamma \models M : T$

Logic: Syntactic $\Rightarrow$ Semantic
Type Theory: Derivation of the type $\Rightarrow$ Semantic evaluation is true
Using $\vdash$ notation $\Rightarrow$ Using $\in$ notation

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash \mathbf{M} : T \Rightarrow \Gamma \models \mathbf{M} : T$

M is

- An object $t, q, \dots$ (variable version x, y, z)
- A constructor $P, Q, \dots$ (variable version α, β, γ)

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : \mathbf{T} \Rightarrow \Gamma \models M : \mathbf{T}$

M is

- An object $t, q, \dots$ (variable x, y, z)
- A constructor $P, Q, \dots$ (variable α, β, γ)

T is

- A type $\sigma, \tau, \dots$
- A kind $A, B, C, \dots$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : \mathbf{T} \Rightarrow \Gamma \models M : \mathbf{T}$

M is

- An object $t, q, \ldots$ (variable x, y, z)
- A constructor $P, Q, \ldots$ (variable α, β, γ)

T is

- A type $\sigma, \tau, \ldots$
- A kind $A, B, C, \ldots$

$$
\begin{array}{ccccc}
 & & \text{constructor } P & : & \text{kind } A \\
 & & \cup\!\shortmid & & \\
\text{objects } t & : & \text{type } \sigma & : & \star
\end{array}
$$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\Gamma \vdash M : \mathbf{T} \Rightarrow \Gamma \models M : \mathbf{T}$

M is

- An object $t, q, \ldots$ (variable x, y, z)
- A constructor $P, Q, \ldots$ (variable α, β, γ)

T is

- A type $\sigma, \tau, \ldots$
- A kind $A, B, C, \ldots$

$$\begin{array}{ccccc} & & \text{constructor } P & : & \text{kind } A \\ & & \cup\!\text{I} & & \\ \text{objects } t & : & \text{type } \sigma & : & \star \end{array}$$

Element		Set		Powerset
		$[\![P]\!]$	$\in$	$\mathcal{V}(A)$
$(\!|t|\!)$	$\in$	$[\![\sigma]\!]$	$\in$	$\mathcal{V}(*)$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\boldsymbol{\Gamma} \vdash M : T \Rightarrow \boldsymbol{\Gamma} \models M : T$

M is

- An object $t, q, \ldots$
- A constructor $P, Q, \ldots$

T is

- A type $\sigma, \tau, \ldots$
- A kind $A, B, C, \ldots$

Γ is context, assigning variables to types and kinds.
For example $\Gamma = x : \sigma,$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\mathbf{\Gamma} \vdash M : T \Rightarrow \mathbf{\Gamma} \models M : T$

M is

- An object $t, q, \ldots$
- A constructor $P, Q, \ldots$

T is

- A type $\sigma, \tau, \ldots$
- A kind $A, B, C, \ldots$

Γ is context, assigning variables to types and kinds.
For example $\Gamma = x : \sigma,\ \sigma : *,$

Things we have seen before

Second order dependent type theory $\lambda P2$

Extends λP by allowing quantification over type constructors
Extends $\lambda 2$ by adding dependency so types can depend on terms

Soundness

$\boldsymbol{\Gamma} \vdash M : T \Rightarrow \boldsymbol{\Gamma} \models M : T$

M is

- An object $t, q, \ldots$
- A constructor $P, Q, \ldots$

T is

- A type $\sigma, \tau, \ldots$
- A kind $A, B, C, \ldots$

Γ is context, assigning variables to types and kinds.
For example $\Gamma = x : \sigma,\ \sigma : *,\ \alpha : A$

Goal of the following section

Goal

To understand the difference between *variables* and *pseudo-terms*

Goal of the following section

Goal

To understand the difference between *variables* and *pseudo-terms*
and what map we use to map each!

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables

Assigned in our context Γ.

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables

Assigned in our context Γ. We use ρ and ξ to evaluate them.

Variables & Pseudo-Terms

	Objects	Constructors	Kinds
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables

Assigned in our context Γ. We use ρ and ξ to evaluate them.

Pseudo-Terms

Depends on our context Γ.

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables

Assigned in our context Γ. We use ρ and ξ to evaluate them.

Pseudo-Terms

Depends on our context Γ. We use $(\!|-|\!)$, $[\![-]\!]$ and $\mathcal{V}(-)$ to evaluate them.

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables

Assigned in our context Γ. We use ρ and ξ to evaluate them.

Pseudo-Terms

Depends on our context Γ. We use $(\!|-|\!)_\rho$, $[\![-]\!]_{\xi\rho}$ and $\mathcal{V}(-)_{\xi\rho}$ to evaluate them

Variables & Pseudo-Terms

	Objects	**Constructors**	**Kinds**
Variables	$\{x, y, z\}$	$\{\alpha, \beta, \gamma\}$	
$\cap$	$\cap$	$\cap$	
Pseudo-Terms	$\{t, u, v\}$	$\{P, Q, R\}$	$\{*, A, B, C\}$

Variables

Assigned in our context Γ. We use ρ and ξ to evaluate them.

Pseudo-Terms

Depends on our context Γ. We use $(\!|-|\!)_\rho$, $[\![-]\!]_{\xi\rho}$ and $\mathcal{V}(-)_{\xi\rho}$ to evaluate themwhile respecting to ρ and ξ.

Goal of the following section

Goal

To understand truth in $\lambda P2$

Truth in $\lambda 2P$

$\Gamma \models M : T$ if $\Gamma \models^{\mathcal{M}} M : T$ for all $\lambda 2P$-models $\mathcal{M}$

Truth in $\lambda 2P$

$\Gamma \models M : T$ if $\Gamma \models^{\mathcal{M}} M : T$ for all $\lambda 2P$-models $\mathcal{M}$

I.e. a general truth in $\lambda P2$ means this is true in **all** models of $\lambda P2$

Truth in $\lambda 2P$

$\Gamma \models M : T$ if $\Gamma \models^{\mathcal{M}} M : T$ for all $\lambda 2P$-models $\mathcal{M}$

I.e. a general truth in $\lambda P2$ means this is true in **all** models of $\lambda P2$

$$
\begin{array}{c}
\Gamma \models^{\mathcal{M}_1} M : T \\
\Gamma \models^{\mathcal{M}_2} M : T \\
\dots \\
\Gamma \models^{\mathcal{M}_n} M : T
\end{array}
$$

Truth in $\lambda 2P$

$\Gamma \models M : T$ if $\Gamma \models^{\mathcal{M}} M : T$ for all $\lambda 2P$-models $\mathcal{M}$

I.e. a general truth in $\lambda P2$ means this is true in **all** models of $\lambda P2$

$$\begin{array}{c} \Gamma \models^{\mathcal{M}_1} M : T \\ \Gamma \models^{\mathcal{M}_2} M : T \\ \dots \\ \Gamma \models^{\mathcal{M}_n} M : T \end{array} \quad \Rightarrow \quad \Gamma \models M : T \text{ in } \lambda P2$$

Goal of the following section

Goal

To understand the two cases of truth for $\lambda P2$

Truth in $\lambda 2P$

Object case

$\Gamma \models^{\mathcal{M}} t : \sigma$

Truth in $\lambda 2P$

Object case

$\Gamma \models^{\mathcal{M}} t : \sigma$ if $(\!| t |\!)_\rho \in [\![\sigma]\!]_{\xi\rho}$

Constructor case

$\Gamma \models^{\mathcal{M}} P : A$

Truth in $\lambda 2P$

Object case

$\Gamma \models^{\mathcal{M}} t : \sigma$ if $(\!|t|\!)_\rho \in [\![\sigma]\!]_{\xi\rho}$

Constructor case

$\Gamma \models^{\mathcal{M}} P : A$ if $[\![P]\!]_{\xi\rho} \in \mathcal{V}(A)_{\xi\rho}$

Truth in $\lambda 2P$

Object case

$\Gamma \models^{\mathcal{M}} t : \sigma$ if $(\!|t|\!)_\rho \in [\![\sigma]\!]_{\xi\rho}$

Constructor case

$\Gamma \models^{\mathcal{M}} P : A$ if $[\![P]\!]_{\xi\rho} \in \mathcal{V}(A)_{\xi\rho}$

Element		Set		Powerset
		$[\![P]\!]_{\xi\rho}$	$\in$	$\mathcal{V}(A)_{\xi\rho}$
$(\!|t|\!)_\rho$	$\in$	$[\![\sigma]\!]_{\xi\rho}$	$\in$	

Truth in $\lambda 2P$

Object case

$\Gamma \models^{\mathcal{M}} t : \sigma$ if $(\!| t |\!)_\rho \in [\![\sigma]\!]_{\xi\rho}$

Constructor case

$\Gamma \models^{\mathcal{M}} P : A$ if $[\![P]\!]_{\xi\rho} \in \mathcal{V}(A)_{\xi\rho}$

Condition: $\rho, \xi \models \Gamma$

Goal of the following section

Goal

To understand how our maps for variables fulfills our context.

$\rho, \xi \models \Gamma$

$\Gamma = \mathsf{Var}^* \cup \mathsf{Var}^{\mathsf{Kind}}$

$\rho, \xi \models \Gamma$

$\Gamma = \mathsf{Var}^* \cup \mathsf{Var}^{\mathsf{Kind}}$

$\mathsf{Var}^* = \{x, y, z\}$

$\mathsf{Var}^{\mathsf{Kind}} = \{\alpha, \beta, \gamma\}$

$\forall x : \mathsf{Var}^*$

$\rho, \xi \models \Gamma$

$\Gamma = \mathsf{Var}^* \cup \mathsf{Var}^{\mathsf{Kind}}$

$\mathsf{Var}^* = \{x, y, z\}$

$\mathsf{Var}^{\mathsf{Kind}} = \{\alpha, \beta, \gamma\}$

$\forall x : \mathsf{Var}^* \ , \ x : \sigma \in \Gamma$

$\rho, \xi \models \Gamma$

$\Gamma = \mathsf{Var}^* \cup \mathsf{Var}^{\mathsf{Kind}}$

$\mathsf{Var}^* = \{x, y, z\}$

$\mathsf{Var}^{\mathsf{Kind}} = \{\alpha, \beta, \gamma\}$

$\forall x : \mathsf{Var}^*$, $x : \sigma \in \Gamma$ then $\rho(x) \in [\![\sigma]\!]_{\xi\rho}$

$\forall \alpha : \mathsf{Var}^{\mathsf{Kind}}$

$\rho, \xi \models \Gamma$

$\Gamma = \mathsf{Var}^* \cup \mathsf{Var}^{\mathsf{Kind}}$

$\mathsf{Var}^* = \{x, y, z\}$

$\mathsf{Var}^{\mathsf{Kind}} = \{\alpha, \beta, \gamma\}$

$\forall x : \mathsf{Var}^*$, $x : \sigma \in \Gamma$ then $\rho(x) \in [\![\sigma]\!]_{\xi\rho}$

$\forall \alpha : \mathsf{Var}^{\mathsf{Kind}}$, $\alpha : A \in \Gamma$

$\rho, \xi \models \Gamma$

$\Gamma = \mathsf{Var}^* \cup \mathsf{Var}^{\mathsf{Kind}}$

$\mathsf{Var}^* = \{x, y, z\}$

$\mathsf{Var}^{\mathsf{Kind}} = \{\alpha, \beta, \gamma\}$

$\forall x : \mathsf{Var}^*$, $x : \sigma \in \Gamma$ then $\rho(x) \in [\![\sigma]\!]_{\xi\rho}$

$\forall \alpha : \mathsf{Var}^{\mathsf{Kind}}$, $\alpha : A \in \Gamma$ then $\xi(x) \in \mathcal{V}(A)_{\xi\rho}$

Goal of the following section

Goal

To understand the size ordering of a $\lambda 2P$ model

Return to combinatorial algebra $\mathcal{A}$

$\lambda P2$-model

$\langle \mathcal{A}, \mathcal{P}, \mathcal{N} \rangle$ is a $\lambda P2$-model where $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a combinatory algebra

Return to combinatorial algebra $\mathcal{A}$

$\lambda P2$-model

$\langle \mathcal{A}, \mathcal{P}, \mathcal{N} \rangle$ is a $\lambda P2$-model where $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a combinatory algebra

Set	**Powerset**	**Set of Powersets**
$\boldsymbol{A}$		

Return to combinatorial algebra $\mathcal{A}$

$\lambda P2$-model

$\langle \mathcal{A}, \mathcal{P}, \mathcal{N} \rangle$ is a $\lambda P2$-model where $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a combinatory algebra

Set	Powerset		Set of Powersets
$\boldsymbol{A}$			
	$\mathcal{P}$	$\in$	$\mathcal{N}$

Return to combinatorial algebra $\mathcal{A}$

$\lambda P2$-model

$\langle \mathcal{A}, \mathcal{P}, \mathcal{N} \rangle$ is a $\lambda P2$-model where $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a combinatory algebra

Set	**Powerset**		**Set of Powersets**
$\boldsymbol{A}$			
	$\mathcal{P}$	$\in$	$\mathcal{N}$
	$\cup\mathcal{N}$		

where $\cup\mathcal{N} = \bigcup_{X \in \mathcal{N}} X$

Return to combinatorial algebra $\mathcal{A}$

$\lambda P2$-model

$\langle \mathcal{A}, \mathcal{P}, \mathcal{N} \rangle$ is a $\lambda P2$-model where $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a combinatory algebra

Set	Powerset		Set of Powersets
$\boldsymbol{A}$			
	$\mathcal{P}$	$\in$	$\mathcal{N}$
	$\cup\mathcal{N}$		

where $\cup\mathcal{N} = \bigcup_{X \in \mathcal{N}} X$

Thus $\mathcal{N} = \{X, Y, Z\}$

Return to combinatorial algebra $\mathcal{A}$

$\lambda P2$-model

$\langle \mathcal{A}, \mathcal{P}, \mathcal{N} \rangle$ is a $\lambda P2$-model where $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a combinatory algebra

Set	Powerset		Set of Powersets
$\boldsymbol{A}$			
	$\mathcal{P}$	$\in$	$\mathcal{N}$
	$\cup\mathcal{N}$		

where $\cup\mathcal{N} = \bigcup_{X \in \mathcal{N}} X$

Thus $\mathcal{N} = \{X, Y, Z\}$ and $\cup\mathcal{N} = \{x_1, \ldots, x_n, y_1, \ldots, y_m, z_1, \ldots, z_w\}$

Goal of the following section

Goal

To understand how the model respects $\boldsymbol{A}$

Goal of the following section

Goal

To understand how the model respects $\boldsymbol{A}$
and why we need evaluations on a set level

Goal of the following section

Goal

To understand how the model respects $\boldsymbol{A}$
and why we need evaluations on a set level, powerset level

Goal of the following section

Goal

To understand how the model respects $\boldsymbol{A}$
and why we need evaluations on a set level, powerset level and set of powerset level.

Mapping of objects

$$\rho : \quad : \quad \{x, y, z, ...\} \quad \rightarrow \quad \boldsymbol{A}$$

Mapping of objects

$$\left|\begin{array}{lcccc} \rho : & : & \{x, y, z, \ldots\} & \to & \boldsymbol{A} \\ \\ (\!|-|\!)_\rho & : & \{t, q, p, \ldots\} & \to & \boldsymbol{A} \end{array}\right|$$

Mapping of objects

$$\left|\begin{array}{lcccc} \rho : & : & \{x, y, z, ...\} & \to & \boldsymbol{A} \\ \llparenthesis - \rrparenthesis_\rho & : & \{t, q, p, ...\} & \to & \boldsymbol{A} \end{array}\right|$$

Example

Let's say $x : \sigma$ and $y : \sigma$ and $t : \sigma$

Mapping of objects

$$\left|\begin{array}{lcccc} \rho : & : & \{x, y, z, ...\} & \to & \boldsymbol{A} \\ (\!|-|\!)_\rho & : & \{t, q, p, ...\} & \to & \boldsymbol{A} \end{array}\right|$$

Example

Let's say $x : \sigma$ and $y : \sigma$ and $t : \sigma$, then $[\![\sigma]\!]_{\xi\rho} =$

Mapping of objects

$$
\left|\begin{array}{lclcl}
\rho : & : & \{x, y, z, ...\} & \rightarrow & \boldsymbol{A} \\
(\!|-|\!)_\rho & : & \{t, q, p, ...\} & \rightarrow & \boldsymbol{A}
\end{array}\right|
$$

Example

Let's say $x : \sigma$ and $y : \sigma$ and $t : \sigma$, then $[\![\sigma]\!]_{\xi\rho} = \{\rho(x),$

Mapping of objects

$$
\left|\begin{array}{lcccc}
\rho : & : & \{x, y, z, ...\} & \rightarrow & \boldsymbol{A} \\
(\!|-|\!)_\rho & : & \{t, q, p, ...\} & \rightarrow & \boldsymbol{A}
\end{array}\right|
$$

Example

Let's say $x : \sigma$ and $y : \sigma$ and $t : \sigma$, then $[\![\sigma]\!]_{\xi\rho} = \{\rho(x), \rho(y),$

Mapping of objects

$$\left|\begin{array}{lcccc} \rho : & : & \{x, y, z, \ldots\} & \to & \boldsymbol{A} \\ (\!|-|\!)_\rho & : & \{t, q, p, \ldots\} & \to & \boldsymbol{A} \end{array}\right|$$

Example

Let's say $x : \sigma$ and $y : \sigma$ and $t : \sigma$, then $[\![\sigma]\!]_{\xi\rho} = \{\rho(x), \rho(y), (\!|t|\!)_\rho\}$

Mapping of objects

$$
\left|\begin{array}{lcccc}
\rho : & : & \{x, y, z, \ldots\} & \rightarrow & \boldsymbol{A} \\
(\!|-|\!)_\rho & : & \{t, q, p, \ldots\} & \rightarrow & \boldsymbol{A}
\end{array}\right|
$$

Example

Let's say $x : \sigma$ and $y : \sigma$ and $t : \sigma$, then $[\![\sigma]\!]_{\xi\rho} = \{\rho(x), \rho(y), (\!|t|\!)_\rho\} \subseteq \boldsymbol{A}$.

Mapping of constructors

$$\Big|\ \xi : \quad : \quad \{\alpha, \beta, \gamma, ...\} \quad \rightarrow \quad \cup\mathcal{N}$$

Mapping of constructors

$$\left|\begin{array}{lclcl} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \to & \cup\mathcal{N} \\ \\ [\![-]\!]_{\xi\rho} & : & \{P, Q, R, ...\} & \to & \cup\mathcal{N} \end{array}\right|$$

Mapping of constructors

$$\left|\begin{array}{lcccc} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \to & \cup\mathcal{N} \\ \llbracket - \rrbracket_{\xi\rho} & : & \{P, Q, R, ...\} & \to & \cup\mathcal{N} \end{array}\right|$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$

Mapping of constructors

$$\left|\begin{array}{lcccl} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \to & \cup\mathcal{N} \\ \llbracket - \rrbracket_{\xi\rho} & : & \{P, Q, R, ...\} & \to & \cup\mathcal{N} \end{array}\right|$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$, then $\mathcal{V}(A)_{\xi\rho} =$

Mapping of constructors

$$
\left|\begin{array}{lclcl}
\xi : & : & \{\alpha, \beta, \gamma, ...\} & \rightarrow & \cup\mathcal{N} \\
[\![-]\!]_{\xi\rho} & : & \{P, Q, R, ...\} & \rightarrow & \cup\mathcal{N}
\end{array}\right|
$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$, then $\mathcal{V}(A)_{\xi\rho} = \{\xi(\alpha),$

Mapping of constructors

$$\left| \begin{array}{lcccc} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \rightarrow & \cup\mathcal{N} \\ [\![-]\!]_{\xi\rho} & : & \{P, Q, R, ...\} & \rightarrow & \cup\mathcal{N} \end{array} \right|$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$, then $\mathcal{V}(A)_{\xi\rho} = \{\xi(\alpha), \beta(\alpha),$

Mapping of constructors

$$\left| \begin{array}{lllll} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \rightarrow & \cup\mathcal{N} \\ [\![-]\!]_{\xi\rho} & : & \{P, Q, R, ...\} & \rightarrow & \cup\mathcal{N} \end{array} \right|$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$, then $\mathcal{V}(A)_{\xi\rho} = \{\xi(\alpha), \beta(\alpha), [\![P]\!]_{\xi\rho}\}$

Mapping of constructors

$$\left|\begin{array}{lcccc} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \to & \cup\mathcal{N} \\ [\![-]\!]_{\xi\rho} & : & \{P, Q, R, ...\} & \to & \cup\mathcal{N} \end{array}\right|$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$, then $\mathcal{V}(A)_{\xi\rho} = \{\xi(\alpha), \beta(\alpha), [\![P]\!]_{\xi\rho}\} \subseteq \cup\mathcal{N}$.

Mapping of constructors

$$\left|\begin{array}{lllll} \xi : & : & \{\alpha, \beta, \gamma, ...\} & \rightarrow & \cup\mathcal{N} \\ [\![-]\!]_{\xi\rho} & : & \{P, Q, R, ...\} & \rightarrow & \cup\mathcal{N} \end{array}\right|$$

Example

Let's say $\alpha : A$ and $\beta : A$ and $P : A$, then $\mathcal{V}(A)_{\xi\rho} = \{\xi(\alpha), \beta(\alpha), [\![P]\!]_{\xi\rho}\} \subseteq \cup\mathcal{N}$.
Note $\xi(\alpha), \beta(\alpha), [\![P]\!]_{\xi\rho} \subseteq \boldsymbol{A}$.

Goal of the following section

Goal

To understand polyset structure $\mathcal{P}$

Goal of the following section

Goal

To understand polyset structure $\mathcal{P}$

We need a polyset structure $\mathcal{P}$ to evaluate our *constructors*

Defining polyset $\mathcal{P}$

Definition of a Polyset Structure

A polyset structure over the weakly extensional combinatory algebra $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a collection $\mathcal{P} \subseteq \wp(\boldsymbol{A})$ such that

Defining polyset $\mathcal{P}$

Definition of a Polyset Structure

A polyset structure over the weakly extensional combinatory algebra $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a collection $\mathcal{P} \subseteq \wp(\boldsymbol{A})$ such that

- 1. $\boldsymbol{A} \in \mathcal{P}$,

Defining polyset $\mathcal{P}$

Definition of a Polyset Structure

A polyset structure over the weakly extensional combinatory algebra $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a collection $\mathcal{P} \subseteq \wp(\boldsymbol{A})$ such that

- 1. $\boldsymbol{A} \in \mathcal{P}$,
- 2. $\mathcal{P}$ is closed under arbitrary intersection $\cap$,

Defining polyset $\mathcal{P}$

Definition of a Polyset Structure

A polyset structure over the weakly extensional combinatory algebra $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a collection $\mathcal{P} \subseteq \wp(\boldsymbol{A})$ such that

- 1. $\boldsymbol{A} \in \mathcal{P}$,
- 2. $\mathcal{P}$ is closed under arbitrary intersection $\cap$,
- 3. $\mathcal{P}$ is closed under dependent products.

Defining polyset $\mathcal{P}$

Definition of a Polyset Structure

A polyset structure over the weakly extensional combinatory algebra $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a collection $\mathcal{P} \subseteq \wp(\boldsymbol{A})$ such that

- 1. $\boldsymbol{A} \in \mathcal{P}$,
- 2. $\mathcal{P}$ is closed under arbitrary intersection $\cap$,
- 3. $\mathcal{P}$ is closed under dependent products.

Examples

$\mathcal{P} = \wp(A)$ is the full polyset structure.

Defining polyset $\mathcal{P}$

Definition of a Polyset Structure

A polyset structure over the weakly extensional combinatory algebra $\mathcal{A} = \langle \boldsymbol{A}, \cdot, \boldsymbol{k}, \boldsymbol{s} \rangle$ is a collection $\mathcal{P} \subseteq \wp(\boldsymbol{A})$ such that

- 1. $\boldsymbol{A} \in \mathcal{P}$,
- 2. $\mathcal{P}$ is closed under arbitrary intersection $\cap$,
- 3. $\mathcal{P}$ is closed under dependent products.

Examples

$\mathcal{P} = \wp(A)$ is the full polyset structure.
$\mathcal{P} = \{\emptyset, \boldsymbol{A}\}$ is a simple polyset structure.

Interpretations of intersection & depedent product

Dependent product of a polyset structure is used to interpret types of the form $\Pi x : \sigma . \tau$, where both σ and τ are types

Interpretations of intersection & depedent product

Dependent product of a polyset structure is used to interpret types of the form $\Pi x : \sigma.\tau$, where both σ and τ are types

Examples

Types are interpreted as subsets $[\![\sigma]\!], [\![\tau]\!] \subseteq \boldsymbol{A}$.
If $[\![\sigma]\!], [\![\tau]\!] \in \mathcal{P}$ then $[\![\sigma \to \tau]\!] \in \mathcal{P}$ (Using simple notation)

Interpretations of intersection & depedent product

Dependent product of a polyset structure is used to interpret types of the form $\Pi x : \sigma.\tau$, where both σ and τ are types

Examples

Types are interpreted as subsets $[\![\sigma]\!], [\![\tau]\!] \subseteq \boldsymbol{A}$.
If $[\![\sigma]\!], [\![\tau]\!] \in \mathcal{P}$ then $[\![\sigma \to \tau]\!] \in \mathcal{P}$ (Using simple notation)

Intersection of a polyset structure is used to interpret types of the form $\Pi\alpha : A.\sigma$, where σ is a type and A is a kind

Interpretations of intersection & depedent product

Dependent product of a polyset structure is used to interpret types of the form $\Pi x : \sigma.\tau$, where both σ and τ are types

Examples

Types are interpreted as subsets $[\![\sigma]\!], [\![\tau]\!] \subseteq \boldsymbol{A}$.
If $[\![\sigma]\!], [\![\tau]\!] \in \mathcal{P}$ then $[\![\sigma \to \tau]\!] \in \mathcal{P}$ (Using simple notation)

Intersection of a polyset structure is used to interpret types of the form $\Pi\alpha : A.\sigma$, where σ is a type and A is a kind

We need to go one size level larger to accommodate for predicates and their powerset interpretations $\mathcal{V}(A)$

Goal of the following section

Goal

To understand predicative structure $\mathcal{N}$

We need a predicative structure $\mathcal{N}$ to evaluate our *kinds*

Defining predicative structure $\mathcal{N}$

Definition of a Predicative Structure

For a polyset structure $\mathcal{P}$, the predicative structure over $\mathcal{P}$ is the collection of sets $\mathcal{N}$ defined inductively by

- 1. $\mathcal{P} \in \mathcal{N}$
- 2. If $X \in \mathcal{P}$ and $\forall t \in X, F(t) \in \mathcal{N}$ then $\Pi_{t \in X} F(t) \in \mathcal{N}$

Defining predicative structure $\mathcal{N}$

Definition of a Predicative Structure

For a polyset structure $\mathcal{P}$, the predicative structure over $\mathcal{P}$ is the collection of sets $\mathcal{N}$ defined inductively by

- 1. $\mathcal{P} \in \mathcal{N}$
- 2. If $X \in \mathcal{P}$ and $\forall t \in X, F(t) \in \mathcal{N}$ then $\Pi_{t \in X} F(t) \in \mathcal{N}$

Set		**Powerset**		**Set of Powersets**
$\boldsymbol{A}$	$\in$	$\mathcal{P}$	$\in$	$\mathcal{N}$

Defining predicative structure $\mathcal{N}$

Definition of a Predicative Structure

For a polyset structure $\mathcal{P}$, the predicative structure over $\mathcal{P}$ is the collection of sets $\mathcal{N}$ defined inductively by

- 1. $\mathcal{P} \in \mathcal{N}$
- 2. If $X \in \mathcal{P}$ and $\forall t \in X, F(t) \in \mathcal{N}$ then $\Pi_{t \in X} F(t) \in \mathcal{N}$

Set		**Powerset**		**Set of Powersets**
$\boldsymbol{A}$	$\in$	$\mathcal{P}$	$\in$	$\mathcal{N}$

$\mathcal{N} = \{\mathcal{P}, X \to \mathcal{P}, X \to X \to \mathcal{P}, Y \to \mathcal{P}, \}$