Pure Patterns Type Systems

P²T S

Luigi Liquori

joint work with

The Rho-Team

INRIA & LORIA

P^2TS

Motivations and Contributions

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

$$(\lambda \mathbf{X}.\mathbf{X})\ \mathbf{3}$$

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

$$\mathbf{3}$$

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

3

- Lambda-calculus with patterns builds upon pattern abstraction:

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

$$\mathbf{3}$$

- Lambda-calculus with patterns builds upon pattern abstraction:

$$(\lambda\langle \mathbf{f}(\mathbf{X}), \mathbf{g}(\mathbf{Y})\rangle.\langle \mathbf{X}, \mathbf{Y}\rangle)\langle \mathbf{3}, 4\rangle$$

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

$$\mathbf{3}$$

- Lambda-calculus with patterns builds upon pattern abstraction:

$$(\lambda\langle \mathbf{sqr(X)}, \mathbf{trl(X)}\rangle.\mathbf{headof(X)})\ \langle \mathbf{sqr(wood)}, \mathbf{trl(wood)}\rangle$$

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

$$\mathbf{3}$$

- Lambda-calculus with patterns builds upon pattern abstraction:

$$(\lambda\langle \mathbf{sqr(X)}, \mathbf{trl(X)}\rangle.\mathbf{headof(X)})\ \langle \mathbf{sqr(wood)}, \mathbf{trl(wood)}\rangle$$

- Rewriting-calculus builds upon generalised abstraction:

$$(\lambda(\lambda \mathbf{P}.\mathbf{Q}).\mathbf{M})\ \mathbf{N}$$

From Lambda-calculus to Rewriting-calculus

- Lambda-calculus builds upon lambda abstraction:

$$\mathbf{3}$$

- Lambda-calculus with patterns builds upon pattern abstraction:

$$(\lambda\langle \mathbf{sqr(X)}, \mathbf{trl(X)}\rangle.\mathbf{headof(X)})\ \langle \mathbf{sqr(wood)}, \mathbf{trl(wood)}\rangle$$

- Rewriting-calculus builds upon generalised abstraction:

$$(\lambda(\lambda \mathbf{X}.\mathbf{Y}).\mathbf{Y}\ \mathbf{3})(\lambda \mathbf{Z}.\mathbf{Z})$$

PATTERNS

We Want More Patterns!

The Uncle Pat

MATCHING

We Want More Matching Power!

The Lady Match

P^2TS: **Tricky !**

- And by the way . . . the below term can have free variables ?

$$\lambda cons(T\ X\ nil(T)).cons(T\ X\ cons(T\ X\ nil(T)))$$

P^2TS: Tricky !

- And by the way . . . the below term can have free variables ?

$$\lambda cons(\mathbf{T}\ X\ nil(\mathbf{T})).cons(\mathbf{T}\ X\ cons(\mathbf{T}\ X\ nil(\mathbf{T})))$$

P^2TS: Tricky !

- And by the way . . . the below term can have free variables ?

$$\lambda cons(\mathbf{T}\ X\ nil(\mathbf{T})).cons(\mathbf{T}\ X\ cons(\mathbf{T}\ X\ nil(\mathbf{T})))$$

- yes, and we can even abstract the T.

$$\lambda T.\lambda cons(T\ X\ nil(T)).cons(T\ X\ cons(T\ X\ nil(T)))$$

P^2TS: **Tricky !**

- ... we cannot reduce the application of

$$\lambda cons(T\ X\ nil(T)).cons(T\ X\ cons(T\ X\ nil(T)))$$

to $cons(int\ 3\ nil(int))$

P^2TS: Tricky !

- ... we cannot reduce the application of

$$\lambda cons(T\ X\ nil(T)).cons(T\ X\ cons(T\ X\ nil(T)))$$

to $cons(int\ 3\ nil(int))$

- ... but we can reduce the application of

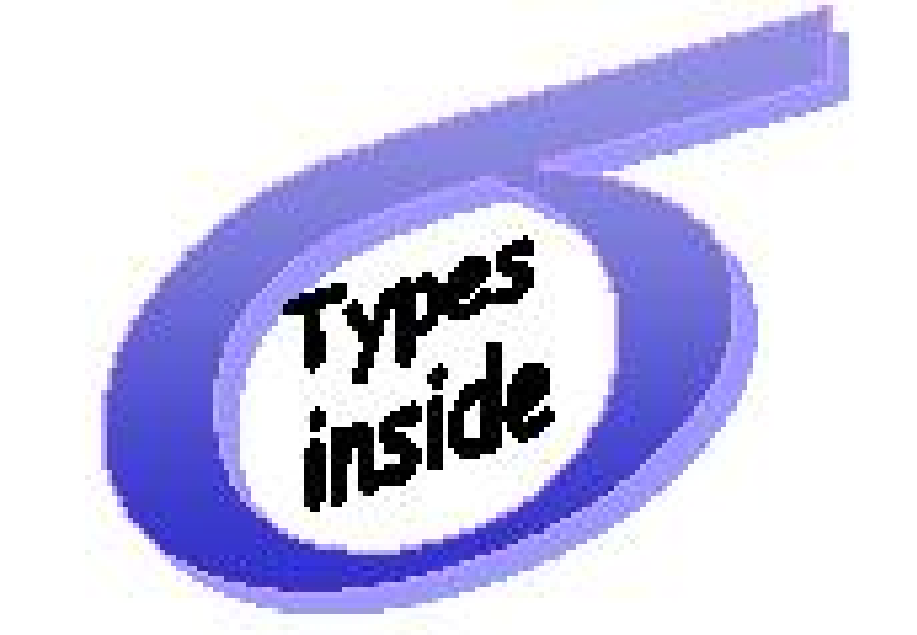
$$\lambda T.\lambda cons(T\ X\ nil(T)).cons(T\ X\ cons(T\ X\ nil(T)))$$

to int and $cons(int\ 3\ nil(int))$

TYPES

We Need to Plug Types!

Thanks to TAL's Group (Cornell)



Typed Rho

- Our goal is to use the Rewriting-calculus as a foundation for proof assistants based on Curry-Howard isomorphism $\grave{a}\ la$ Coq, Twelf, Lego, . . .

Typed Rho

- Our goal is to use the Rewriting-calculus as a foundation for proof assistants based on Curry-Howard isomorphism $\grave{a}\ la$ Coq, Twelf, Lego, . . .
- As an intermediate goal, we develop a dependent type theory for the Rewriting-calculus

Typed Rho

- Our goal is to use the Rewriting-calculus as a foundation for proof assistants based on Curry-Howard isomorphism $\grave{a}\ la$ Coq, Twelf, Lego, . . .
- As an intermediate goal, we develop a dependent type theory for the Rewriting-calculus
- We do so by extending the framework of Pure Type Systems (PTS)

Typed Rho

- Our goal is to use the Rewriting-calculus as a foundation for proof assistants based on Curry-Howard isomorphism $\grave{a}\ la$ Coq, Twelf, Lego, . . .
- As an intermediate goal, we develop a dependent type theory for the Rewriting-calculus
- We do so by extending the framework of Pure Type Systems (PTS)
- We develop the basic theory of the resulting framework which we call

Pure Pattern Type Systems

(P^2TS)

Typed Rho

- Our goal is to use the Rewriting-calculus as a foundation for proof assistants based on Curry-Howard isomorphism $\grave{a}\ la$ Coq, Twelf, Lego, . . .
- As an intermediate goal, we develop a dependent type theory for the Rewriting-calculus
- We do so by extending the framework of Pure Type Systems (PTS)
- We develop the basic theory of the resulting framework which we call

  Pure Pattern Type Systems

  (P^2TS)

- This is not as straightforward as one may imagine

Typed Rho

- Our goal is to use the Rewriting-calculus as a foundation for proof assistants based on Curry-Howard isomorphism $\grave{a}\ la$ Coq, Twelf, Lego, . . .
- As an intermediate goal, we develop a dependent type theory for the Rewriting-calculus
- We do so by extending the framework of Pure Type Systems (PTS)
- We develop the basic theory of the resulting framework which we call

  Pure Pattern Type Systems

  (P^2TS)

- This is not as straightforward as one may imagine :-)

P^{2}T S: Some problems

- Confluence can fails for bad patterns

$$(\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})((\lambda\mathbf{Z}.\mathbf{Z})\mathbf{a}) \mapsto\!\!\!\!\rightarrow$$

P^2TS: Some problems

- Confluence can fails for bad patterns

$$(\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})((\lambda\mathbf{Z}.\mathbf{Z})\mathbf{a}) \mapsto\!\!\!\!\rightarrow (\lambda\mathbf{Z}.\mathbf{Z})$$

P^2TS: Some problems

- Confluence can fails for bad patterns

$$(\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})((\lambda\mathbf{Z}.\mathbf{Z})\mathbf{a}) \mapsto\!\!\!\!\rightarrow (\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})\mathbf{a}$$

P^2TS: Some problems

- Confluence can fails for bad patterns

$$(\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})((\lambda\mathbf{Z}.\mathbf{Z})\mathbf{a}) \mapsto\!\!\!\!\rightarrow (\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})\mathbf{a}$$

- Subject Reduction can fails for bad patterns

$$\vdash (\lambda(\mathbf{X}_\mathbf{1}^{\sigma_1 \rightarrowtriangle \tau}\ \mathbf{Y}_\mathbf{1}^{\sigma_1}).\ \mathbf{Z}^{\sigma_1 \rightarrowtriangle \tau}\ \mathbf{Y}_\mathbf{1}^{\sigma_1})\ (\mathbf{X}_\mathbf{2}^{\sigma_2 \rightarrowtriangle \tau}\ \mathbf{Y}_\mathbf{2}^{\sigma_2}) : \tau$$

P^2TS: Some problems

- Confluence can fails for bad patterns

$$(\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})((\lambda\mathbf{Z}.\mathbf{Z})\mathbf{a}) \mapsto\!\!\!\!\rightarrow (\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})\mathbf{a}$$

- Subject Reduction can fails for bad patterns

$$\not\vdash \mathbf{Z}^{\sigma_1 \rightarrowtriangle \tau}\ \mathbf{Y}_{\mathbf{2}}^{\sigma_\mathbf{2}} : \tau$$

$P^2T\,S$: Some problems

- Confluence can fails for bad patterns

$$(\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})((\lambda\mathbf{Z}.\mathbf{Z})\mathbf{a}) \mapsto\!\!\!\!\rightarrow (\lambda(\mathbf{X}\ \mathbf{Y}).\mathbf{X})\mathbf{a}$$

- Subject Reduction can fails for bad patterns

$$\not\vdash \mathbf{Z}^{\sigma_1 \rightarrowtriangle \tau}\ \mathbf{Y}_{\mathbf{2}}^{\sigma_\mathbf{2}} : \tau$$

- Shapes of good patterns must be "`xunison`-ed" with a sound static type system!

The main contribution of this (ongoing) work are ...

- to provide adequate notions of patterns, substitutions and syntactic matching in a typed setting.
  We introduce delayed matching constraint, and the possibility for patterns in abstractions to evolve (by reduction or substitution) during execution

The main contribution of this (ongoing) work are ...

- to provide adequate notions of patterns, substitutions and syntactic matching in a typed setting.
  We introduce delayed matching constraint, and the possibility for patterns in abstractions to evolve (by reduction or substitution) during execution
- to propose an extension of PTSs supporting abstraction over patterns, and enjoying
  - ⋆ confluence
  - ⋆ subject reduction
  - ⋆ conservativity over PTSs
  - ⋆ consistency for normalizing P^2TSs

The main contribution of this (ongoing) work are ...

- to provide adequate notions of patterns, substitutions and syntactic matching in a typed setting.
  We introduce delayed matching constraint, and the possibility for patterns in abstractions to evolve (by reduction or substitution) during execution
- to propose an extension of PTSs supporting abstraction over patterns, and enjoying
  - ⋆ confluence
  - ⋆ subject reduction
  - ⋆ conservativity over PTSs
  - ⋆ consistency for normalizing P^2TSs
- Strong normalization for all P^2TS is an open problem . . . but it is ok for simple P^2TS-types (see Benjamin Wack SN-paper) and it "seems" ok for simple+dependent P^2TS-types ($\rightsquigarrow$ *Rhogical Framework*)

P^2TS

The Syntax

$\mathsf{P}^2\mathsf{T}\,\mathsf{S}$

its time to be uniform!

$$\lambda A.B \sim A \to B$$

The Typed Syntax

$$\Gamma \ ::= \ \emptyset \mid \Gamma, X{:}A \mid \Gamma, f{:}A$$

$$A \ ::= \ X \mid f \mid A \to_\Delta B \mid A\,A \mid [A \ll_\Delta B]C \mid A; B$$

The Typed Syntax

$$\Gamma \ ::= \ \emptyset \mid \Gamma, X{:}A \mid \Gamma, f{:}A$$

$$A \ ::= \ X \mid f \mid A \to_\Delta B \mid A\,A \mid [A \ll_\Delta B]C \mid A; B$$

1. Term $\mathbf{A \to_\Delta B}$ is an abstraction (resp. product abstraction $i.e.$ $\Pi A{:}\Delta.B$)

The Typed Syntax

$$\Gamma \ ::= \ \emptyset \mid \Gamma, X{:}A \mid \Gamma, f{:}A$$

$$A \ ::= \ X \mid f \mid A \to_\Delta B \mid A\,A \mid [A \ll_\Delta B]C \mid A; B$$

1. Term $\mathbf{A \to_\Delta B}$ is an abstraction (resp. product abstraction $i.e.$ $\Pi A{:}\Delta.B$)

2. Term $\mathbf{[A \ll_\Delta B]C}$ is a delayed matching constraint

The Typed Syntax

$$\Gamma \ ::= \ \emptyset \mid \Gamma, X{:}A \mid \Gamma, f{:}A$$

$$A \ ::= \ X \mid f \mid A \to_\Delta B \mid A\,A \mid [A \ll_\Delta B]C \mid A; B$$

1. Term $\mathbf{A \to_\Delta B}$ is an abstraction (resp. product abstraction $i.e.$ $\Pi A{:}\Delta.B$)
2. Term $\mathbf{[A \ll_\Delta B]C}$ is a delayed matching constraint
3. Term of the form $\mathbf{A; B}$ is called a structure

P^2T S

Galleria & Glance

Galleria I: The Pattern Abstraction $\mathbf{A} \rightarrow_{\Delta} \mathbf{B}$

- Generalisation of the λ-abstraction in PTSs. The rationale is:

$$\mathbf{X} \rightarrow_{\underbrace{(\mathbf{X}:\sigma)}_{\Delta}} \mathbf{A} \sim \lambda \mathbf{X}:\sigma.\mathbf{A}$$

Galleria I: The Pattern Abstraction $\mathbf{A} \rightarrow_{\Delta} \mathbf{B}$

- Generalisation of the λ-abstraction in PTSs. The rationale is:

$$\mathbf{f}(\mathbf{X}\,\mathbf{Y}) \rightarrow \underbrace{(\mathbf{X}{:}\sigma, \mathbf{Y}{:}\tau)}_{\Delta}\ \mathbf{A} \sim \lambda\mathbf{f}(\mathbf{X}\,\mathbf{Y}){:}(\mathbf{X}{:}\sigma, \mathbf{Y}{:}\tau).\mathbf{A}$$

Galleria I: The Pattern Abstraction $\mathbf{A} \to_{\mathbf{\Delta}} \mathbf{B}$

- Generalisation of the λ-abstraction in PTSs. The rationale is:

$$\mathbf{f}(\mathbf{X}\,\mathbf{Y}) \to_{\underbrace{(\mathbf{X}{:}\sigma,\,\mathbf{Y}{:}\tau)}_{\mathbf{\Delta}}} \mathbf{A} \sim \lambda\mathbf{f}(\mathbf{X}\,\mathbf{Y}){:}(\mathbf{X}{:}\sigma,\,\mathbf{Y}{:}\tau).\mathbf{A}$$

- Instead of simple variables we abstract over sophisticated patterns
- The free variables of A (bound in B) are declared in the context Δ, $i.e.$

$$\mathsf{Fv}(\mathbf{A} \to_{\mathbf{\Delta}} \mathbf{B}) \triangleq (\mathsf{Fv}(\mathbf{A}) \cup \mathsf{Fv}(\mathbf{B}) \cup \mathsf{Fv}(\mathbf{\Delta})) \setminus \mathsf{Dom}(\mathbf{\Delta})$$

Galleria I: The Pattern Abstraction $\mathbf{A} \to_{\Delta} \mathbf{B}$

- Generalisation of the λ-abstraction in PTSs. The rationale is:

$$\mathbf{f}(\mathbf{X}\,\mathbf{Y}) \to_{\underbrace{(\mathbf{X}{:}\sigma,\,\mathbf{Y}{:}\tau)}_{\Delta}} \mathbf{A} \sim \lambda\mathbf{f}(\mathbf{X}\,\mathbf{Y}){:}(\mathbf{X}{:}\sigma,\,\mathbf{Y}{:}\tau).\mathbf{A}$$

- Instead of simple variables we abstract over sophisticated patterns
- The free variables of A (bound in B) are declared in the context Δ, $i.e.$

$$\mathsf{Fv}(\mathbf{A} \to_{\mathbf{\Delta}} \mathbf{B}) \triangleq (\mathsf{Fv}(\mathbf{A}) \cup \mathsf{Fv}(\mathbf{B}) \cup \mathsf{Fv}(\mathbf{\Delta})) \setminus \mathsf{Dom}(\mathbf{\Delta})$$

- Δ discriminates on which $\mathsf{Fv}(A)$ will be bound in B and which not

$$\mathbf{cons}(\mathbf{T}\,\mathbf{X}\,\mathbf{nil}(\mathbf{T})) \to_{(\mathbf{X}:\mathbf{T})} \mathbf{cons}(\mathbf{T}\,\mathbf{X}\,\mathbf{cons}(\mathbf{T}\,\mathbf{X}\,\mathbf{nil}(\mathbf{T})))$$

Galleria I: The Pattern Abstraction $\mathbf{A} \to_{\Delta} \mathbf{B}$

- Generalisation of the λ-abstraction in PTSs. The rationale is:

$$\mathbf{f}(\mathbf{X}\,\mathbf{Y}) \to_{\underbrace{(\mathbf{X}{:}\sigma,\,\mathbf{Y}{:}\tau)}_{\Delta}} \mathbf{A} \sim \lambda\mathbf{f}(\mathbf{X}\,\mathbf{Y}){:}(\mathbf{X}{:}\sigma,\,\mathbf{Y}{:}\tau).\mathbf{A}$$

- Instead of simple variables we abstract over sophisticated patterns
- The free variables of A (bound in B) are declared in the context Δ, $i.e.$

$$\mathsf{Fv}(\mathbf{A} \to_{\mathbf{\Delta}} \mathbf{B}) \triangleq (\mathsf{Fv}(\mathbf{A}) \cup \mathsf{Fv}(\mathbf{B}) \cup \mathsf{Fv}(\mathbf{\Delta})) \setminus \mathsf{Dom}(\mathbf{\Delta})$$

- Δ discriminates on which $\mathsf{Fv}(A)$ will be bound in B and which not

$$\mathbf{cons}(\mathbf{T}\;\mathbf{X}\;\mathbf{nil}(\mathbf{T})) \to_{(\mathbf{X}:\mathbf{T})} \mathbf{cons}(\mathbf{T}\;\mathbf{X}\;\mathbf{cons}(\mathbf{T}\;\mathbf{X}\;\mathbf{nil}(\mathbf{T})))$$

Galleria II: The Matching Constraint $[\mathbf{A} \ll_{\mathbf{\Delta}} \mathbf{B}]\mathbf{C}$

- In the term

$$[\mathbf{A} \ll_{\mathbf{\Delta}} \mathbf{B}]\mathbf{C}$$

the matching equation $[A \ll_{\Delta} B]$ is put on the stack, hence constraints and *"de facto"* blocks the evaluation of C

- The body C will be evaluated (in case a matching solution exists) or delayed (in case no solution exists at this stage of the evaluation)
- If a solution exists, the delayed matching constraint self-evaluates to $C\sigma$, otherwise the evaluation is delayed to a later stage
- The free variables of A declared in Δ are bound in B but not in C, *i.e.*

$$\mathsf{Fv}([\mathbf{A} \ll_{\mathbf{\Delta}} \mathbf{B}]\mathbf{C}) \triangleq \mathsf{Fv}((\mathbf{A} \rightarrow_{\mathbf{\Delta}} \mathbf{C})\,\mathbf{B})$$

P^2TS

Matching Algorithm

HARD RUN

EASY RUN

SKIP

Less *Easy Running*

- $X \rightarrow_{(X:i)} X \not\ll^{?}_{\emptyset} X \rightarrow_{(X:i)} X$

Less *Easy Running*

- $X \rightarrow_{(X:i)} X \not\ll^{?}_{\emptyset} X \rightarrow_{(X:i)} X \rightsquigarrow X \not\ll^{?}_{X} X$ OK!!

Less *Easy Running*

- $X \rightarrow_{(X:i)} X \not\ll^{?}_{\emptyset} X \rightarrow_{(X:i)} X \rightsquigarrow X \not\ll^{?}_{X} X$ OK!!
- $X \rightarrow_{(X:i)} X \not\ll^{?}_{\emptyset} X \rightarrow_{(X:i)} Y$

Less *Easy Running*

- $X \rightarrow_{(X:i)} X \not\ll^{?}_{\emptyset} X \rightarrow_{(X:i)} X \rightsquigarrow X \not\ll^{?}_{X} X$ OK!!
- $X \rightarrow_{(X:i)} X \not\ll^{?}_{\emptyset} X \rightarrow_{(X:i)} Y \rightsquigarrow X \not\ll^{?}_{X} X \wedge X \not\ll^{?}_{X} Y$ KO!!

Less *Easy Running*

- $X \rightarrow_{(X:i)} X \lessdot^{?}_{\emptyset} X \rightarrow_{(X:i)} X \rightsquigarrow X \lessdot^{?}_{X} X$ OK!!
- $X \rightarrow_{(X:i)} X \lessdot^{?}_{\emptyset} X \rightarrow_{(X:i)} Y \rightsquigarrow X \lessdot^{?}_{X} X \wedge X \lessdot^{?}_{X} Y$ KO!!
- $X \rightarrow_{(X:i)} f(X\,Y) \lessdot^{Y}_{\emptyset} X \rightarrow_{(X:i)} f(X\,3) \rightsquigarrow\!\!\!\rightarrow$

Less *Easy Running*

- $X \to_{(X:i)} X \lessdot^{?}_{\emptyset} X \to_{(X:i)} X \rightsquigarrow X \lessdot^{?}_{X} X$ OK!!
- $X \to_{(X:i)} X \lessdot^{?}_{\emptyset} X \to_{(X:i)} Y \rightsquigarrow X \lessdot^{?}_{X} X \wedge X \lessdot^{?}_{X} Y$ KO!!
- $X \to_{(X:i)} f(X\,Y) \lessdot^{Y}_{\emptyset} X \to_{(X:i)} f(X\,3) \rightsquigarrow\!\!\!\rightarrow$

  $X \lessdot^{Y}_{X} X \wedge f \lessdot^{Y}_{X} f \wedge X \lessdot^{Y}_{X} X \wedge Y \lessdot^{Y}_{X} 3$ OK!!

Less *Easy Running*

- $X \to_{(X:i)} X \ll^{?}_{\emptyset} X \to_{(X:i)} X \rightsquigarrow X \ll^{?}_{X} X$ OK!!
- $X \to_{(X:i)} X \ll^{?}_{\emptyset} X \to_{(X:i)} Y \rightsquigarrow X \ll^{?}_{X} X \wedge X \ll^{?}_{X} Y$ KO!!
- $X \to_{(X:i)} f(X\,Y) \ll^{Y}_{\emptyset} X \to_{(X:i)} f(X\,3) \rightsquigarrow\!\!\!\rightarrow$

  $X \ll^{Y}_{X} X \wedge f \ll^{Y}_{X} f \wedge X \ll^{Y}_{X} X \wedge Y \ll^{Y}_{X} 3$ OK!!
- $[f(X) \ll_{(X:i)} f(Y)]X \ll^{Y}_{\emptyset} [f(X) \ll_{(X:i)} f(3)].X \rightsquigarrow\!\!\!\rightarrow$

Less *Easy Running*

- $X \to_{(X:i)} X \ll^{?}_{\emptyset} X \to_{(X:i)} X \rightsquigarrow X \ll^{?}_{X} X$ OK!!
- $X \to_{(X:i)} X \ll^{?}_{\emptyset} X \to_{(X:i)} Y \rightsquigarrow X \ll^{?}_{X} X \wedge X \ll^{?}_{X} Y$ KO!!
- $X \to_{(X:i)} f(X\,Y) \ll^{Y}_{\emptyset} X \to_{(X:i)} f(X\,3) \rightsquigarrow\!\!\!\rightarrow$

  $X \ll^{Y}_{X} X \wedge f \ll^{Y}_{X} f \wedge X \ll^{Y}_{X} X \wedge Y \ll^{Y}_{X} 3$ OK!!
- $[f(X) \ll_{(X:i)} f(Y)] X \ll^{Y}_{\emptyset} [f(X) \ll_{(X:i)} f(3)].X \rightsquigarrow\!\!\!\rightarrow$

  $f \ll^{Y}_{X} f \wedge X \ll^{Y}_{X} X \wedge X \ll^{Y}_{X} X \wedge f \ll^{Y}_{\emptyset} f \wedge Y \ll^{Y}_{\emptyset} 3$ OK!!

Two *Easy Running*

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(T\ 3\ nil(T))}$

Two *Easy Running*

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(T\ 3\ nil(T))}$

Solve $\mathbf{cons(T\ X\ nil(T))} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(T\ 3\ nil(T))}$

Two *Easy Running*

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(T\ 3\ nil(T))}$

Solve $\mathbf{cons(T\ X\ nil(T))} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(T\ 3\ nil(T))} \rightsquigarrow$

$\mathbf{X} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{3} \wedge \mathbf{T} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{T}$ **OK!! with** $\sigma = \{\mathbf{3}/\mathbf{X}\}$

Two *Easy Running*

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(T\ 3\ nil(T))}$

  Solve $\mathbf{cons(T\ X\ nil(T))} \prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(T\ 3\ nil(T))} \rightsquigarrow$

  $\mathbf{X} \prec^{\mathbf{X}}_{\emptyset} \mathbf{3} \wedge \mathbf{T} \prec^{\mathbf{X}}_{\emptyset} \mathbf{T}$ **OK!! with** $\sigma = \{\mathbf{3}/\mathbf{X}\}$

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(i\ 3\ nil(i))}$

Two *Easy Running*

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(T\ 3\ nil(T))}$

  Solve $\mathbf{cons(T\ X\ nil(T))} \prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(T\ 3\ nil(T))} \rightsquigarrow$

  $\mathbf{X} \prec^{\mathbf{X}}_{\emptyset} \mathbf{3} \wedge \mathbf{T} \prec^{\mathbf{X}}_{\emptyset} \mathbf{T}$ **OK!! with** $\sigma = \{\mathbf{3/X}\}$

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(i\ 3\ nil(i))}$

  Solve $\mathbf{cons(T\ X\ nil(T))} \prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(i\ 3\ nil(i))}$

Two *Easy Running*

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(T\ 3\ nil(T))}$

  Solve $\mathbf{cons(T\ X\ nil(T))} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(T\ 3\ nil(T))} \rightsquigarrow$

  $\mathbf{X} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{3} \wedge \mathbf{T} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{T}$ **OK!! with** $\sigma = \{\mathbf{3/X}\}$

- $(\mathbf{cons(T\ X\ nil(T))} \rightarrow_{(\mathrm{X:i})} \mathbf{X})\,\mathbf{cons(i\ 3\ nil(i))}$

  Solve $\mathbf{cons(T\ X\ nil(T))} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{cons(i\ 3\ nil(i))} \rightsquigarrow$

  $\mathbf{X} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{3} \wedge \mathbf{T} \prec\!\!\prec^{\mathbf{X}}_{\emptyset} \mathbf{i}$ **KO!!**

P^2T S

Type System

The Type System I

$$\frac{(s_1, s_2) \in \mathcal{A}}{\emptyset \vdash s_1 : s_2}\ (Axioms)$$

The Type System I

$$\frac{(s_1, s_2) \in \mathcal{A}}{\emptyset \vdash s_1 : s_2}\ (Axioms)$$

$$\frac{\Gamma \vdash A : C \qquad \Gamma \vdash B : C}{\Gamma \vdash A; B : C}\ (Struct)$$

The Type System I

$$\frac{(s_1, s_2) \in \mathcal{A}}{\emptyset \vdash s_1 : s_2}\ (Axioms)$$

$$\frac{\Gamma \vdash A : C \qquad \Gamma \vdash B : C}{\Gamma \vdash A; B : C}\ (Struct)$$

$$\frac{\Gamma \vdash A : s \qquad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}A \vdash \alpha : A}\ (Start)$$

The Type System I

$$\frac{(s_1, s_2) \in \mathcal{A}}{\emptyset \vdash s_1 : s_2}\ (Axioms)$$

$$\frac{\Gamma \vdash A : C \qquad \Gamma \vdash B : C}{\Gamma \vdash A; B : C}\ (Struct)$$

$$\frac{\Gamma \vdash A : s \qquad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}A \vdash \alpha : A}\ (Start)$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \qquad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}C \vdash A : B}\ (Weak)$$

The Type System I

$$\frac{(s_1, s_2) \in \mathcal{A}}{\emptyset \vdash s_1 : s_2}\ (Axioms)$$

$$\frac{\Gamma \vdash A : C \qquad \Gamma \vdash B : C}{\Gamma \vdash A; B : C}\ (Struct)$$

$$\frac{\Gamma \vdash A : s \qquad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}A \vdash \alpha : A}\ (Start)$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \qquad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha{:}C \vdash A : B}\ (Weak)$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : D \qquad B =_{\beta\delta} C}{\Gamma \vdash A : C}\ (Conv)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{ll} \Gamma \vdash C : s_1 & (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C & \Gamma, \Delta \vdash B : s_2 \end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}{\Gamma \vdash A\ B : [C \ll_\Delta B]D} \; (Appl)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{c}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}{\Gamma \vdash A\ B : [C \ll_\Delta B]D} \; (Appl)$$

$$\frac{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma,\Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C}\ (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma,\Delta \vdash A : C \qquad \Gamma,\Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3}\ (Prod)$$

$$\frac{\boldsymbol{\Gamma \vdash A : \Pi C{:}\Delta.D} \qquad \Gamma,\Delta \vdash C : E \qquad \Gamma \vdash B : E}{\Gamma \vdash A\ B : [C \ll_\Delta B]D}\ (Appl)$$

$$\frac{\Gamma,\Delta \vdash A : E \qquad \Gamma,\Delta \vdash C : D \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E}\ (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\boldsymbol{\Gamma \vdash A : \Pi C{:}\Delta.D} \qquad \boldsymbol{\Gamma, \Delta \vdash C : E} \qquad \Gamma \vdash B : E}{\Gamma \vdash A\ B : [C \ll_\Delta B]D} \; (Appl)$$

$$\frac{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\mathbf{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}}{\Gamma \vdash A\ B : [C \ll_\Delta B]D} \; (Appl)$$

$$\frac{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\mathbf{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}}{\mathbf{\Gamma \vdash A\ B : [C \ll_\Delta B]D}} \; (Appl)$$

$$\frac{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\mathbf{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}}{\mathbf{\Gamma \vdash A\ B : [C \ll_\Delta B]D}} \; (Appl)$$

$$\frac{\mathbf{\Gamma, \Delta \vdash A : E} \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\mathbf{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}}{\mathbf{\Gamma \vdash A\ B : [C \ll_\Delta B]D}} \; (Appl)$$

$$\frac{\mathbf{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D} \qquad \Gamma \vdash B : D}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{ll}\Gamma \vdash C : s_1 & (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C & \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\mathbf{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}}{\mathbf{\Gamma \vdash A\ B : [C \ll_\Delta B]D}} \; (Appl)$$

$$\frac{\mathbf{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}}{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E} \; (Subst)$$

The Type System II

$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma \vdash \Pi A{:}\Delta.C : s}{\Gamma \vdash A \to_\Delta B : \Pi A{:}\Delta.C} \; (Abs)$$

$$\frac{\begin{array}{l}\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \\ \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2\end{array}}{\Gamma \vdash \Pi A{:}\Delta.B : s_3} \; (Prod)$$

$$\frac{\mathbf{\Gamma \vdash A : \Pi C{:}\Delta.D \qquad \Gamma, \Delta \vdash C : E \qquad \Gamma \vdash B : E}}{\mathbf{\Gamma \vdash A\ B : [C \ll_\Delta B]D}} \; (Appl)$$

$$\frac{\mathbf{\Gamma, \Delta \vdash A : E \qquad \Gamma, \Delta \vdash C : D \qquad \Gamma \vdash B : D}}{\mathbf{\Gamma \vdash [C \ll_\Delta B]A : [C \ll_\Delta B]^\top E}} \; (Subst)$$

Fetch Your System

$$\frac{\Gamma \vdash C : s_1 \qquad (s_1, s_2, s_3) \in \mathcal{R} \qquad \Gamma, \Delta \vdash A : C \qquad \Gamma, \Delta \vdash B : s_2}{\Gamma \vdash \Pi A{:}\Delta.B : s_3}\ (Prod)$$

System	Rules			
$\rho \rightarrow$	$(*,*,*)$			
$\rho 2$	$(*,*,*)$	$(\Box,*,*)$		
$\rho\underline{\omega}$	$(*,*,*)$			$(\Box,\Box,\Box)$
$\rho\omega$	$(*,*,*)$		$(*,\Box,\Box)$	$(\Box,\Box,\Box)$
ρLF	$(*,*,*)$		$(*,\Box,\Box)$	
$\rho P2$	$(*,*,*)$	$(\Box,*,*)$	$(*,\Box,\Box)$	
$\rho P\underline{\omega}$	$(*,*,*)$		$(*,\Box,\Box)$	$(\Box,\Box,\Box)$
$\rho P\omega$	$(*,*,*)$	$(\Box,*,*)$	$(*,\Box,\Box)$	$(\Box,\Box,\Box)$

P^2TS

Typed Examples

Example: Simple Type Derivation

Let $\Gamma \triangleq i{:}*, f{:}\Pi Z{:}\Sigma.i, 3{:}i, 4{:}i$, and $\Delta \triangleq X{:}i$, and $\Sigma \triangleq Z{:}i$,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma,\Delta \vdash (\lambda 3{:}\emptyset.3) : \Pi 3{:}\emptyset.i \quad \boxed{5}\ \boxed{6} \qquad \Gamma,\Delta \vdash X : i}{\Gamma,\Delta \vdash (\lambda 3{:}\emptyset.3)\, X : [3 \ll_\emptyset X].i} \qquad \dfrac{\boxed{3} \quad (*,*,*) \in \mathcal{R}}{\Gamma \vdash [Z \ll_\Sigma X].i : *} \quad \Gamma,\Delta \vdash [3 \ll_\emptyset X].i : *}{\Gamma \vdash \Pi f(X){:}\Delta.[3 \ll_\emptyset X].i : *}}{\Gamma \vdash \lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X : \Pi f(X){:}\Delta.[3 \ll_\emptyset X].i} \quad \boxed{3}\ \boxed{4}}{\Gamma \vdash (\lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X)\, f(3) : [f(X) \ll_\Delta f(3)].[3 \ll_\emptyset X].i} \quad \boxed{1}\ \boxed{2}}{\Gamma \vdash (\lambda \mathbf{f}(\mathbf{X}){:}\mathbf{\Delta}.(\lambda \mathbf{3}{:}\emptyset.\mathbf{3})\mathbf{X})\, \mathbf{f}(\mathbf{3}) : \mathbf{i}}$$

where $\boxed{1} \triangleq [f(X) \ll_\Delta f(3)].[3 \ll_\emptyset X].i =_{\rho\sigma\delta} i$, and $\boxed{2} \triangleq \Gamma \vdash i : *$, and $\boxed{3} \triangleq \Gamma,\Delta \vdash f(X) : [Z \ll_\Sigma X].i$, and $\boxed{4} \triangleq \Gamma \vdash f(3) : [Z \ll_\Sigma 3].i$, and $\boxed{5} \triangleq \Gamma,\Delta \vdash X : i$, and $\boxed{6} \triangleq \Gamma,\Delta \vdash 3 : i$.

Example: Simple Type Derivation

Let $\Gamma \triangleq i{:}*, f{:}\Pi Z{:}\Sigma.i, 3{:}i, 4{:}i$, and $\Delta \triangleq X{:}i$, and $\Sigma \triangleq Z{:}i$,

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \Delta \vdash (\lambda 3{:}\emptyset.3) : \Pi 3{:}\emptyset.i} \quad \boxed{5\ 6}}{\Gamma, \Delta \vdash (\lambda 3{:}\emptyset.3)\ X : [3 \ll_{\emptyset} X].i} \quad \dfrac{\begin{array}{c} \boxed{3} \quad (*, *, *) \in \mathcal{R} \\ \Gamma \vdash [Z \ll_{\Sigma} X].i : * \\ \Gamma, \Delta \vdash [3 \ll_{\emptyset} X].i : * \end{array}}{\Gamma \vdash \Pi f(X){:}\Delta.[3 \ll_{\emptyset} X].i : *}}{\Gamma \vdash \lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X : \Pi f(X){:}\Delta.[3 \ll_{\emptyset} X].i} \quad \boxed{3\ 4}}{\dfrac{\Gamma \vdash (\lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X)\ f(3) : [f(X) \ll_{\Delta} f(3)].[3 \ll_{\emptyset} X].i \quad \boxed{1\ 2}}{\Gamma \vdash \mathbf{(\lambda f(X){:}\Delta.(\lambda 3{:}\emptyset.3)X)\ f(3) : i}}}$$

where $\boxed{1} \triangleq [f(X) \ll_{\Delta} f(3)].[3 \ll_{\emptyset} X].i =_{\rho\sigma\delta} i$, and $\boxed{2} \triangleq \Gamma \vdash i : *$, and $\boxed{3} \triangleq \Gamma, \Delta \vdash f(X) : [Z \ll_{\Sigma} X].i$, and $\boxed{4} \triangleq \Gamma \vdash f(3) : [Z \ll_{\Sigma} 3].i$, and $\boxed{5} \triangleq \Gamma, \Delta \vdash X : i$, and $\boxed{6} \triangleq \Gamma, \Delta \vdash 3 : i$.

Playing with the Rho-cube: ρLF

- Let $\Gamma \stackrel{\triangle}{=} i{:}*, f{:}\Pi X{:}(X{:}i).*, 3{:}i$

$$\dfrac{\dfrac{\dfrac{\Gamma, X{:}i \vdash X : i \quad \Gamma \vdash i : * \quad \Gamma, X{:}i \vdash * : \Box}{\Gamma \vdash \Pi X{:}(X{:}i).* : \Box}}{\Gamma \vdash f : \Pi X{:}(X{:}i).*} \qquad \Gamma, X{:}i \vdash X : i \quad \Gamma \vdash 3 : i}{\Gamma \vdash f(3) : [X \ll_{(X:i)} 3]^{\top} * \equiv *}$$

- $\Gamma \vdash \Pi X{:}(X{:}i).* : \Box$ can be derived thanks to the specific rule $(*, \Box, \Box)$

Playing with the Rho-cube: $\rho 2$

- In this system the following polymorphic identity with pattern f" can be derived (where f denotes $f^{X^* \to X^*}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{\dfrac{\vdots}{\vdash f(Y^{X^*}) : X^*} \quad \dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{}{\vdash * : \Box}}{\vdash Y^{X^*} : X^*} \quad \dfrac{\dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{}{\vdash * : \Box}}{\vdash f(Y^{X^*}) \to Y^{X^*} : f(Y^{X^*}) \to X^*} \quad \dfrac{\dfrac{ok!}{\vdots}}{\vdash f(Y^{X^*}) \to X^* : *}$$

$$\dfrac{X{:}*, Y{:}X \vdash f(Y^{X^*}) \to X : *}{X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to Y^{X^*} : X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to X : *}$$

$(Conv{+}Appl)$

Playing with the Rho-cube: $\rho 2$

- In this system the following polymorphic identity with pattern f" can be derived (where f denotes $f^{X^* \to X^*}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{\dfrac{\vdots}{\vdash f(Y^{X^*}) : X^*} \quad \dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{}{\vdash * : \Box}}{\vdash Y^{X^*} : X^*} \quad \dfrac{\dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{}{\vdash * : \Box} \quad \dfrac{\dfrac{}{ok!}\vdots}{\vdash f(Y^{X^*}) \to X^* : *}}{\dfrac{\vdash f(Y^{X^*}) \to Y^{X^*} : f(Y^{X^*}) \to X^* \qquad X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to X : *}{X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to Y^{X^*} : X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to X : *}}$$

$(Conv{+}Appl)$

- $X{:}*, Y{:}X \vdash f(Y^{X^*}) \to X : *$ can be derived thanks to $(*, *, *)$

Playing with the Rho-cube: $\rho 2$

- In this system the following polymorphic identity with pattern f" can be derived (where f denotes $f^{X^* \to X^*}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{\dfrac{\vdots}{\vdash f(Y^{X^*}) : X^*} \quad \dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \; (Conv{+}Appl)}{\vdash Y^{X^*} : X^*} \quad \dfrac{}{X{:}*, Y{:}X \vdash f(Y^{X^*}) \to X : *}}{\vdash f(Y^{X^*}) \to Y^{X^*} : f(Y^{X^*}) \to X^*} \quad \dfrac{\dfrac{}{\vdash * : \Box}}{\vdash X^* : *} \quad \dfrac{}{\vdash * : \Box} \quad \dfrac{\dfrac{\vdots}{\vdash f(Y^{X^*}) \to X^* : *}\; ok!}{X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to X : *}$$

$$X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to Y^{X^*} : X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to X : *$$

- $X{:}*, Y{:}X \vdash f(Y^{X^*}) \to X : *$ can be derived thanks to $(*, *, *)$

- $X{:}*, Y{:}X \vdash X^* \to f(Y^{X^*}) \to X : *$ can be derived thanks to $(\Box, *, *)$

P^2T S

Metatheory

CONCLUSIONS

P^2TS: Some Results

- **Confluence** The relation $\mapsto_{\rho\partial\delta}$ is confluent

- **Subject Reduction.** If $\Gamma \vdash A : B$, and $A \mapsto_{\rho\partial\delta} C$, then $\Gamma \vdash C : B$

$\mathsf{P}^2\mathsf{T}\,\mathsf{S}$: Some Results

- **Confluence** The relation $\mapsto_{\rho\partial\mathcal{S}}$ is confluent
- **Subject Reduction.** If $\Gamma \vdash A : B$, and $A \mapsto_{\rho\partial\mathcal{S}} C$, then $\Gamma \vdash C : B$
- **Consistency.** Any normalizing $\mathsf{P}^2\mathsf{T}\,\mathsf{S}$ is logically consistent, $i.e.$ for every sort $s \in \mathcal{S}$, $X{:}s \not\vdash A : X$
- **Conservativity.** $\mathsf{P}^2\mathsf{T}\,\mathsf{S}$s are a conservative extension of PTSs:

$$\Gamma \vdash_{\mathsf{PTS}} A : B \quad \Longleftrightarrow \quad \Gamma^\dagger \vdash_{\mathsf{P^2T\,S}} A^\dagger : B^\dagger$$

Open Tracks

- strong normalization, we conjecture that standard model construction techniques can be used to prove strong normalization of the $\lambda^{\ll}$-cube; Benj
- type checking/inference, we conjecture that existing algorithms for PTSs adapt readily to P^2TSs; Luigi
- it would be interesting to study P^2TSs with a limited form of decidable higher-order unification, in the style of λ-Prolog; Claude/Gop?
- encoding dependent case analysis, dependent sum types (records) $\grave{a}$ laCoquand-Pollack-Luo; ?
- explicit substitutions. The extension is not trivial, because of delayed matching constraints, but the resulting formalism could serve as the core $engine$ of a little type-checker underneath of a powerful proof assistant; u:Claude-Germain,t:?

Challenge ... *Extending the Curry-Howard Isomorphism*

- The extension can be considered from the point of view of sequent calculi, deduction modulo, and natural deduction respectively;
- From the point of view of sequent calculi, it remains to investigate how $\mathsf{P}^2\mathsf{TS}$s can be used to extend previous results on term calculi for sequent calculi, and how their extension with matching theories can be used to provide suitable term calculi for deduction modulo;
- From the point of view of natural deduction, $\mathsf{P}^2\mathsf{TS}$s correspond to an extension of natural deduction where parts of proof trees are discharged instead of assumptions;
- To our best knowledge, such an extended form of natural deduction has not been considered previously, but it seems interesting to investigate whether such an extended natural deduction could find some applications in proof assistants, *e.g.* for transforming and optimizing proofs. END

Logics and Rho *à la* Church

The relation with (intuitionistic) logic through the so-called Curry-Howard isomorphism, or '*formulae-as-types*' principle, has been profoundly studied for Lambda. However, for Rho *à la* Church, this relation is less clear, as demonstrated by the authors. The principle could be adapted as follows:

> *Given a typed term A, if we can derive for A a type τ in the typed system Rho, with a derivation $\mathcal{D}er_{\mathsf{T}}$, then the term A can be seen as the coding of a logical proof, proving the formula φ that can be interpreted as the type τ assigned to A.*

Curry from Church

For those systems, if $\mathcal{D}er_{\mathsf{T}}$ is a typed derivation, and $(\!|-|\!)$ is the above meant erasing function, then by applying $(\!|-|\!)$ to the "subject" of every judgment in $\mathcal{D}er_{\mathsf{T}}$, we obtain a valid type assignment derivation $\mathcal{D}er_{\mathsf{U}}$ with the same structure of the typed one. Vice versa, every type assignment derivation can be viewed as the result of an application of $(\!|-|\!)$ to a typed one. In particular, the erasing function $(\!|-|\!)$ induces an *isomorphism* between every typed system and the corresponding type assignment system.

$$
\begin{array}{lcl}
(\!| f |\!) & \triangleq & f \\
(\!| X |\!) & \triangleq & X \\
(\!| A\,\tau |\!) & \triangleq & (\!| A |\!)
\end{array}
\qquad
\begin{array}{lcl}
(\!| [P \ll_{\Delta} A].B |\!) & \triangleq & [P \ll (\!| A |\!)].(\!| B |\!) \\
(\!| P \to_{\Delta} A |\!) & \triangleq & P \to (\!| A |\!) \\
(\!| A\,B |\!) & \triangleq & (\!| A |\!)\,(\!| B |\!) \\
(\!| A; B |\!) & \triangleq & (\!| A |\!); (\!| B |\!)
\end{array}
$$

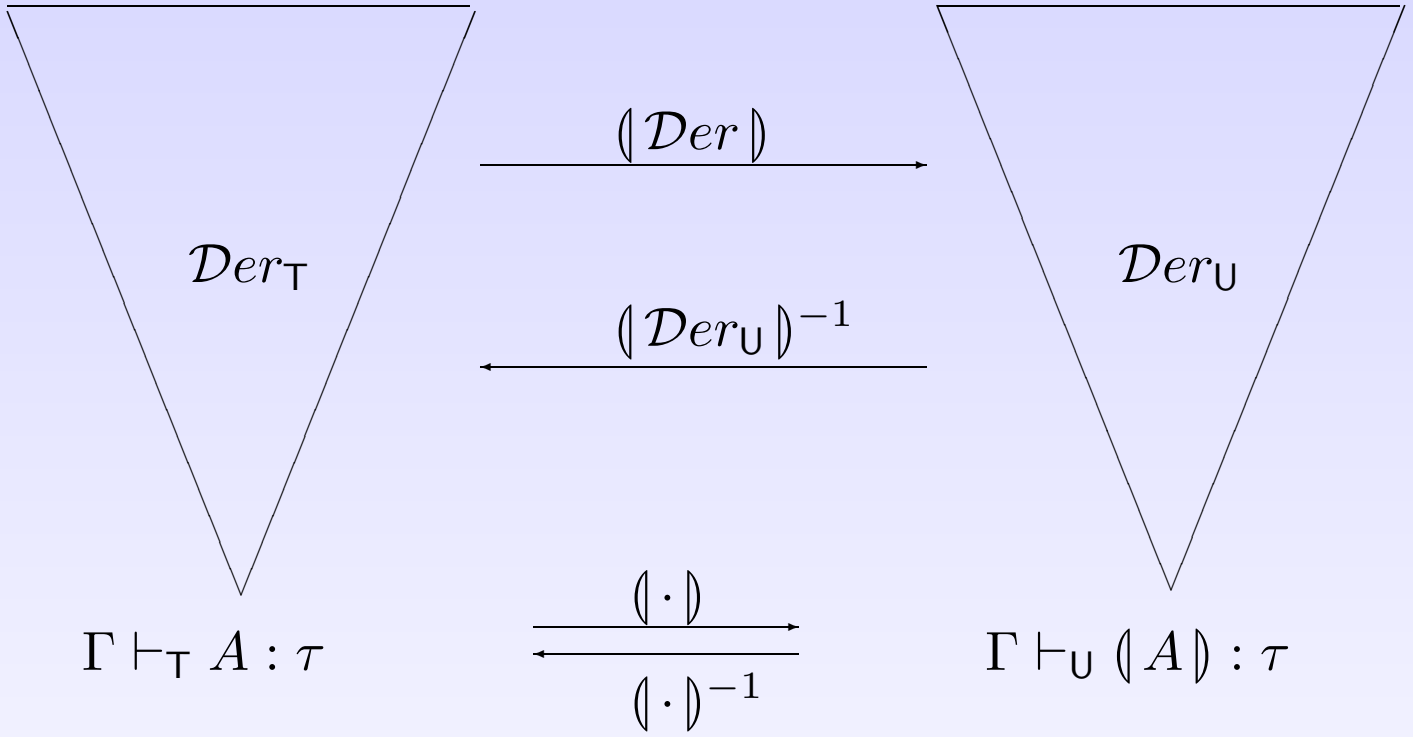
Logics and Rho *à la* Curry

For the type assignment system Rho the relation with logic is less clear even for the corresponding type assignments for the Lambda. The 'formulae-as-types' principle of Curry and Howard could be extended to the above type assignment systems as follows:

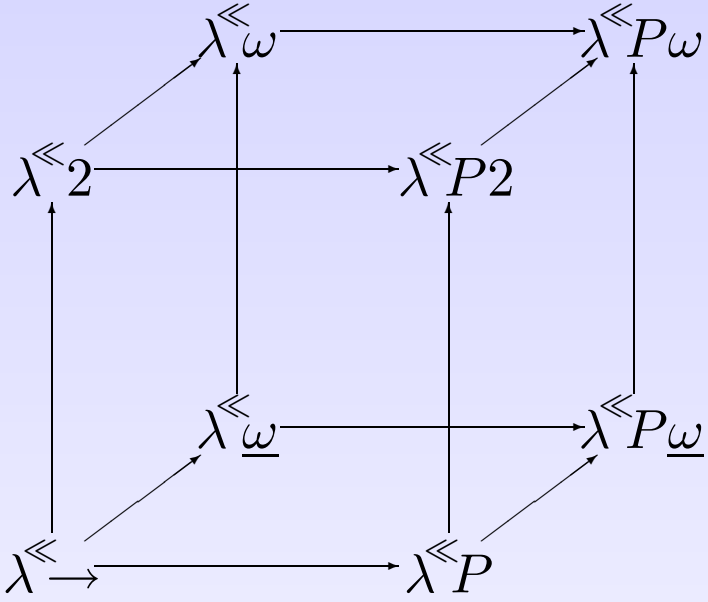
Given an untyped term U, if we can assign a type τ in the type assignment system Rho, with a derivation $\mathcal{D}er_{\mathsf{U}}$, then:

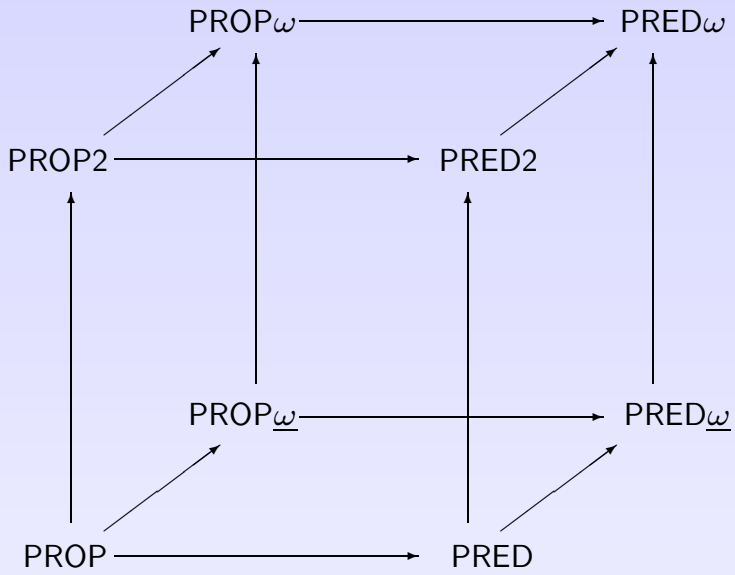
- *$\mathcal{D}er_{\mathsf{U}}$ can be interpreted as the coding of a proof for the logic formulas φ which corresponds to the interpretation of the type τ assigned to U;*
- *U can be interpreted as the coding of a "logical proof schema", whose instances (of the schema) prove, respectively, all the logic formulas φ_i's that can be interpreted as the types τ_i's that can be assigned to U.*

Typed and Untyped Judgments and Derivations



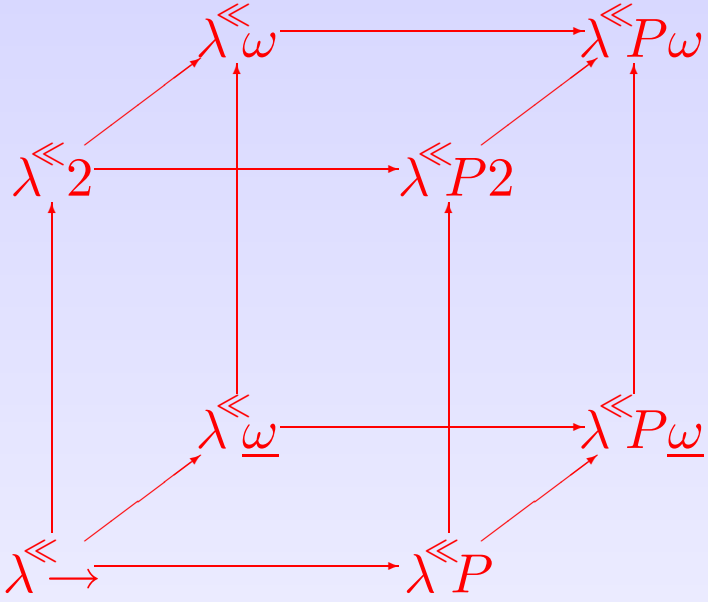
Thanks for youρ attention ...

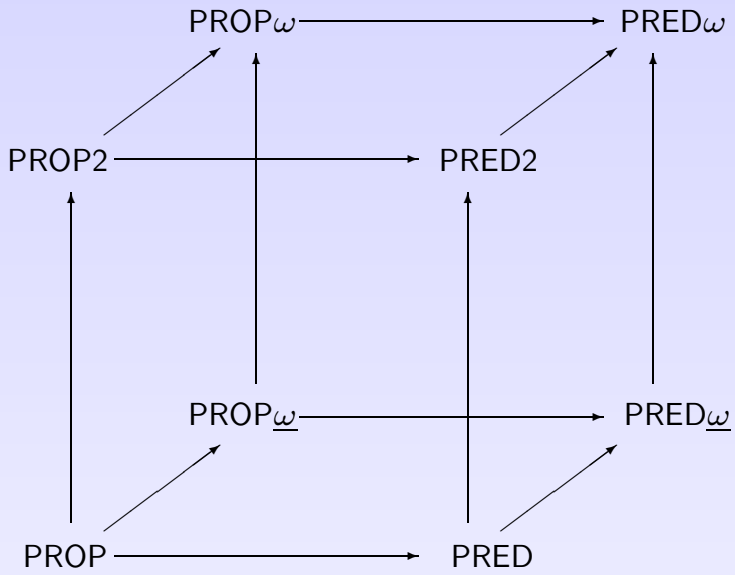




PROP	proposition logic	PRED	predicate logic
PROP2	second-order PROP	PRED2	second-order PRED
PROP$\underline{\omega}$	weakly higher-order PROP	PRED$\underline{\omega}$	weakly higher-order PRED
PROPω	higher-order PROP	PREDω	higher-order PRED

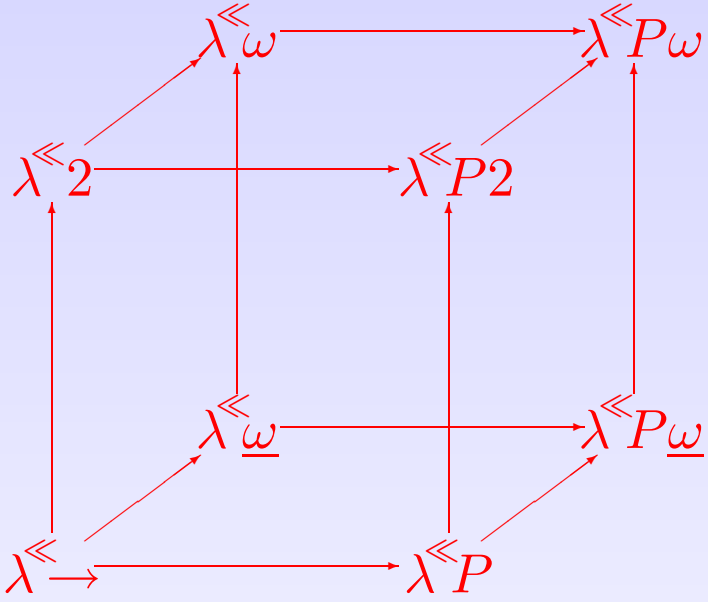
Thanks for youρ attention ...

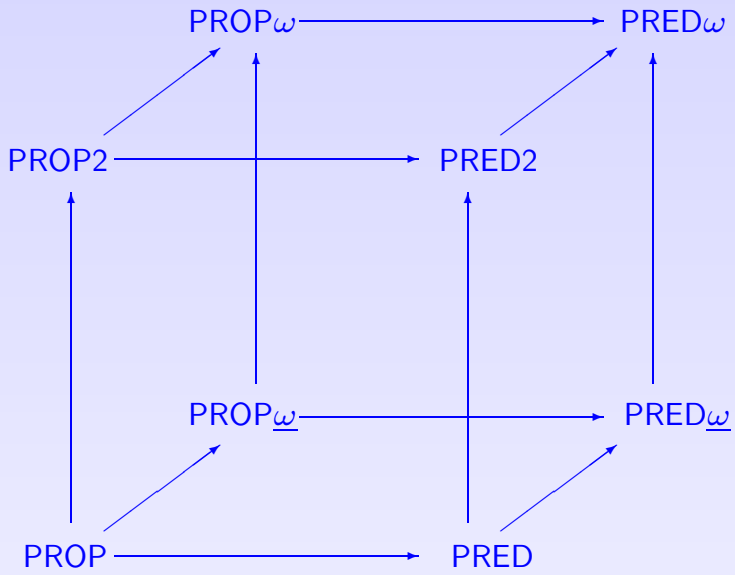




PROP	proposition logic	PRED	predicate logic
PROP2	second-order PROP	PRED2	second-order PRED
PROP$\underline{\omega}$	weakly higher-order PROP	PRED$\underline{\omega}$	weakly higher-order PRED
PROPω	higher-order PROP	PREDω	higher-order PRED

Thanks for youρ attention ...





PROP	proposition logic	PRED	predicate logic
PROP2	second-order PROP	PRED2	second-order PRED
PROP$\underline{\omega}$	weakly higher-order PROP	PRED$\underline{\omega}$	weakly higher-order PRED
PROPω	higher-order PROP	PREDω	higher-order PRED

The Uncle Pat and the Lady Match

+

$= P^2TS$

$P^2T\,S$

ALL THE ZOOMS BELOW!

Lambda Calculi *à la* Church and Logics

- Lambda abstractions are decorated with types, *e.g.* $\lambda x{:}\sigma.M$
- Type Systems λ_i *vs.* Logic Systems $\mathcal{L}_i$ via the well-known Curry-Howard Isomorphism *"proofs-as-(λ)-terms & propositions-as-types"*
- Each logical system $\mathcal{L}_i$ correspond to a type system λ_i, and for every formula ϕ

$$\vdash_{\mathcal{L}_i} \phi \implies \exists M.\ \Gamma \vdash_{\lambda_i} M : [\![\phi]\!]$$

Lambda Calculi *à la* Church and Logics

- Lambda abstractions are decorated with types, *e.g.* $\lambda x{:}\sigma.M$
- Type Systems λ_i *vs.* Logic Systems $\mathcal{L}_i$ via the well-known Curry-Howard Isomorphism *"proofs-as-(λ)-terms & propositions-as-types"*
- Each logical system $\mathcal{L}_i$ correspond to a type system λ_i, and for every formula ϕ

$$\vdash_{\mathcal{L}_i} \phi \implies \exists M.\, \Gamma \vdash_{\lambda_i} M : [\![\phi]\!]$$

- Γ contains the types of the free variables of M, and $[\![\phi]\!]$ is a canonical interpretation of ϕ in λ_i

Lambda Calculi *à la* Church and Logics

- Lambda abstractions are decorated with types, *e.g.* $\lambda x{:}\sigma.M$
- Type Systems λ_i *vs.* Logic Systems $\mathcal{L}_i$ via the well-known Curry-Howard Isomorphism *"proofs-as-(λ)-terms & propositions-as-types"*
- Each logical system $\mathcal{L}_i$ correspond to a type system λ_i, and for every formula ϕ

$$\vdash_{\mathcal{L}_i} \phi \implies \exists M.\, \Gamma \vdash_{\lambda_i} M : [\![\phi]\!]$$

- Γ contains the types of the free variables of M, and $[\![\phi]\!]$ is a canonical interpretation of ϕ in λ_i
- β-reduction $(\lambda x{:}\sigma.M)\, N \to_\beta M\{N/x\}$ in λ_i as cut-elimination in $\mathcal{L}_i$

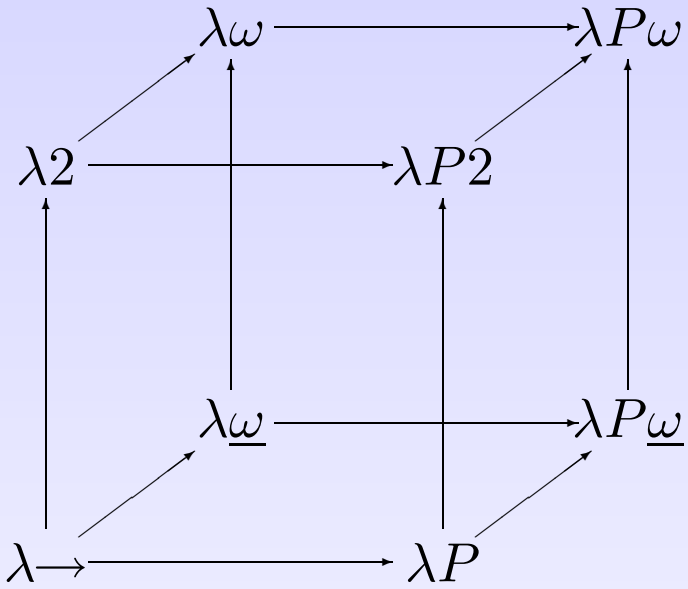
Lambda Calculi *à la* Church and Logics

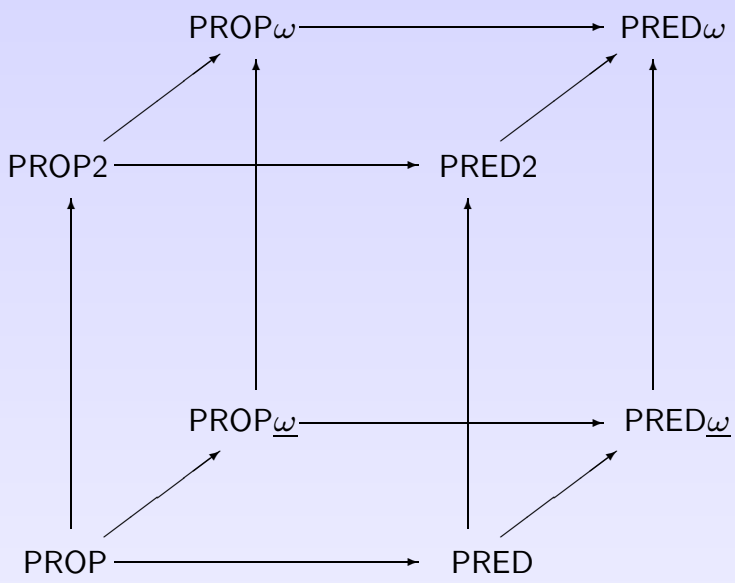
- Lambda abstractions are decorated with types, *e.g.* $\lambda x{:}\sigma.M$
- Type Systems λ_i *vs.* Logic Systems $\mathcal{L}_i$ via the well-known Curry-Howard Isomorphism *"proofs-as-(λ)-terms & propositions-as-types"*
- Each logical system $\mathcal{L}_i$ correspond to a type system λ_i, and for every formula ϕ

$$\vdash_{\mathcal{L}_i} \phi \implies \exists M.\, \Gamma \vdash_{\lambda_i} M : [\![\phi]\!]$$

- Γ contains the types of the free variables of M, and $[\![\phi]\!]$ is a canonical interpretation of ϕ in λ_i
- β-reduction $(\lambda x{:}\sigma.M)\, N \to_\beta M\{N/x\}$ in λ_i as cut-elimination in $\mathcal{L}_i$
- Subject Reduction Theorem as a correction criterion for cut-elimination BACK

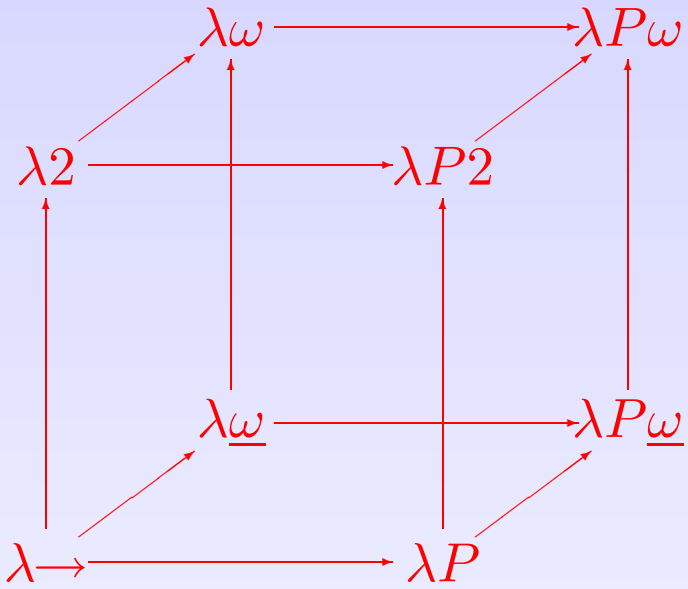
The Famous Barendregt's λ-cube and its 8 logic systems

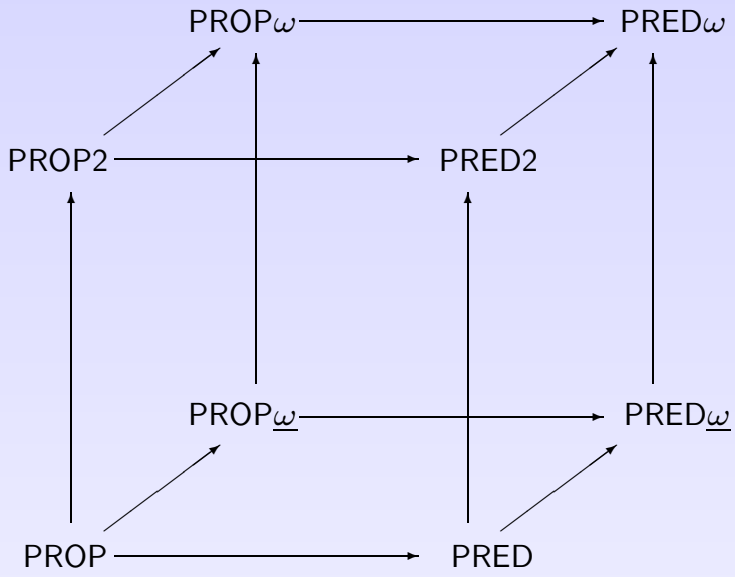




PROP	proposition logic	PRED	predicate logic
PROP2	second-order PROP	PRED2	second-order PRED
PROP$\underline{\omega}$	weakly higher-order PROP	PRED$\underline{\omega}$	weakly higher-order PRED
PROPω	higher-order PROP	PREDω	higher-order PRED

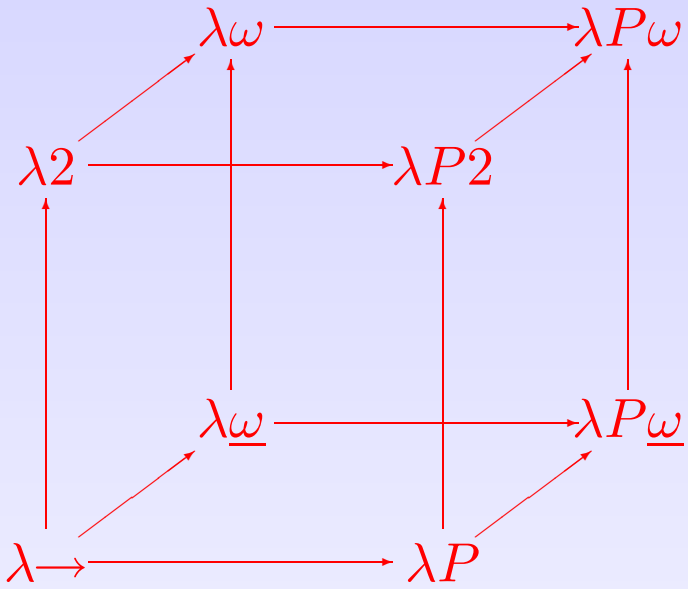
The Famous Barendregt's λ-cube and its 8 logic systems

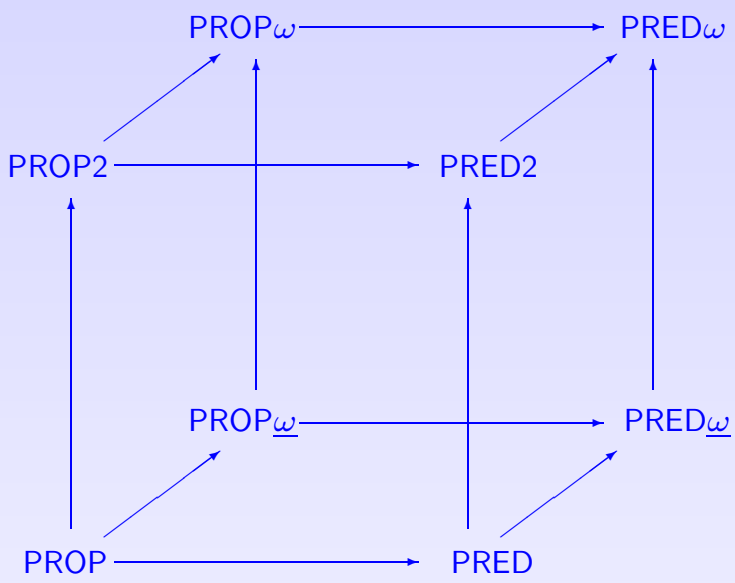




PROP	proposition logic	PRED	predicate logic
PROP2	second-order PROP	PRED2	second-order PRED
PROP$\underline{\omega}$	weakly higher-order PROP	PRED$\underline{\omega}$	weakly higher-order PRED
PROPω	higher-order PROP	PREDω	higher-order PRED

The Famous Barendregt's λ-cube and its 8 logic systems





PROP	proposition logic	PRED	predicate logic
PROP2	second-order PROP	PRED2	second-order PRED
PROP$\underline{\omega}$	weakly higher-order PROP	PRED$\underline{\omega}$	weakly higher-order PRED
PROPω	higher-order PROP	PREDω	higher-order PRED

From Barendregt's λ-cube to Pure Type Systems (PTSs)

- Further generalization of various type systems invented independently by Berardi and Terlouw in '89 BACK
- Many systems of typed λ-calculus $\grave{a}\ la$ Church can be seen as PTSs
- One of the success of PTSs is concerned with logics: the 8 logical systems can be described/generalised as a simple unique PTS

From Barendregt's λ-cube to Pure Type Systems (PTSs)

- Further generalization of various type systems invented independently by Berardi and Terlouw in '89 BACK
- Many systems of typed λ-calculus $\grave{a}\ la$ Church can be seen as PTSs
- One of the success of PTSs is concerned with logics: the 8 logical systems can be described/generalised as a simple unique PTS
- Another one is the compactness of the notation of PTSs which greatly allows to factorise and simplify proofs in metatheory, in the style *"one theorem fits all!"*

From Barendregt's λ-cube to Pure Type Systems (PTSs)

- Further generalization of various type systems invented independently by Berardi and Terlouw in '89 BACK
- Many systems of typed λ-calculus $\grave{a}\ la$ Church can be seen as PTSs
- One of the success of PTSs is concerned with logics: the 8 logical systems can be described/generalised as a simple unique PTS
- Another one is the compactness of the notation of PTSs which greatly allows to factorise and simplify proofs in metatheory, in the style *"one theorem fits all!"*
- Examples of well-known PTSs are λHOL, $\lambda PRED$, λCC ($a.k.a.$ the λ-cube)

More Pragmatically ...

- Almost all proof assistants and relating metalanguages based on the proposition-as-type principle, have a firm theoretical basis in logics represented via PTSs

More Pragmatically ...

- Almost all proof assistants and relating metalanguages based on the proposition-as-type principle, have a firm theoretical basis in logics represented via PTSs
- AUTOMATH, NUPRL, HOL, LEGO, (TW)ELF, AGDA, ISABELLE, COQ, MIZAR, ACL2, PVS ...

More Pragmatically ...

- Almost all proof assistants and relating metalanguages based on the proposition-as-type principle, have a firm theoretical basis in logics represented via PTSs
- AUTOMATH, NUPRL, HOL, LEGO, (TW)ELF, AGDA, ISABELLE, COQ, MIZAR, ACL2, PVS ...
- The degree of automatization of such proof assistants depends also on the capability of simplifying terms

More Pragmatically ...

- Almost all proof assistants and relating metalanguages based on the proposition-as-type principle, have a firm theoretical basis in logics represented via PTSs
- AUTOMATH, NUPRL, HOL, LEGO, (TW)ELF, AGDA, ISABELLE, COQ, MIZAR, ACL2, PVS ...
- The degree of automatization of such proof assistants depends also on the capability of simplifying terms
- The Poincaré principle can be $(\beta\iota\delta)$-reductions, structural well-founded recursion, provable equality, or some arbitrary notion of reduction

More Pragmatically ...

- Almost all proof assistants and relating metalanguages based on the proposition-as-type principle, have a firm theoretical basis in logics represented via PTSs
- AUTOMATH, NUPRL, HOL, LEGO, (TW)ELF, AGDA, ISABELLE, COQ, MIZAR, ACL2, PVS ...
- The degree of automatization of such proof assistants depends also on the capability of simplifying terms
- The Poincaré principle can be $(\beta\iota\delta)$-reductions, structural well-founded recursion, provable equality, or some arbitrary notion of reduction
- The more reductions principles you have in the metalanguage, the more "powerful" the proof assistant is ... BACK

P^2TS: One-step, Many-steps, Congruence

Let $\mathsf{Ctx}[-]$ be any term $\mathcal{T}$ with a "single hole" inside, and let $\mathsf{Ctx}[A]$ be the result of filling the hole with the term A;

1. the one-step evaluation $\mapsto$ is defined by the following inference rule, where $\rightarrow_{\rho\sigma\delta} \equiv \rightarrow_\rho \cup \rightarrow_\sigma \cup \rightarrow_\delta$:

$$\frac{A \rightarrow_{\rho\sigma\delta} B}{\mathsf{Ctx}[A] \mapsto_{\rho\sigma\delta} \mathsf{Ctx}[B]}\ (\mathsf{Ctx}[-])$$

2. the many-step evaluation $\mapsto\!\!\!\!\rightarrow_{\rho\sigma\delta}$ and congruence relation $=_{\rho\sigma\delta}$ are respectively defined as the reflexive-transitive and reflexive-symmetric-transitive closure of $\mapsto_{\rho\sigma\delta}$

BACK

Abbreviations and Priorities

$A(B_1 \cdots B_n)$	$\triangleq$	$A \bullet B_1 \bullet \cdots \bullet B_n$	function-application
$(A_i)^{i=1\ldots n}$	$\triangleq$	$A_1; \cdots ; A_n$	structure/object
$A.B$	$\triangleq$	$A \bullet B \bullet A$	Kamin's self-application

Abbreviations and Priorities

$$A(B_1 \cdots B_n) \triangleq A \bullet B_1 \bullet \cdots \bullet B_n \quad \text{function-application}$$

$$(A_i)^{i=1\ldots n} \triangleq A_1; \cdots; A_n \quad \text{structure/object}$$

$$A.B \triangleq A \bullet B \bullet A \quad \text{Kamin's self-application}$$

Operator	Associate	Priority
_ ; _	Right	>
✓_:_._	Right	>
[_ ≪_ _]._	Right	>
_•_	Left	>

... still substitutions

- We let

$$\mathsf{Dom}(\sigma) = \{X_1, \ldots, X_m\}$$

and

$$\mathsf{CoDom}(\Delta) = \bigcup_{i=1\ldots m} \mathsf{Fv}(A_i)$$

- A substitution σ is independent from Δ, written $\sigma \not\sharp \Delta$ if

$$\mathsf{Dom}(\sigma) \cap \mathsf{Dom}(\Delta) = \emptyset$$

and

$$\mathsf{CoDom}(\sigma) \cap \mathsf{Dom}(\Delta) = \emptyset$$

The Algorithm $\mathcal{A}lg$

$$(Lbd/Prod) \qquad (\checkmark A_1{:}\Delta.B_1) \not\prec^{\Sigma}_{\Gamma} (\checkmark A_2{:}\Delta.B_2)$$

$$\rightsquigarrow \quad A_1 \not\prec^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \not\prec^{\Sigma}_{\Gamma,\Delta} B_2$$

The Algorithm $\mathcal{A}lg$

$$(Lbd/Prod) \qquad (\checkmark A_1{:}\Delta.B_1) \prec\!\!\!\prec^{\Sigma}_{\Gamma} (\checkmark A_2{:}\Delta.B_2)$$

$$\rightsquigarrow \quad A_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} B_2$$

$$(Delay) \qquad [A_1 \ll_{\Delta} C_1].B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma} [A_2 \ll_{\Delta} C_2].B_2$$

$$\rightsquigarrow \quad A_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} B_2 \wedge C_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma} C_2$$

The Algorithm $\mathcal{A}lg$

$$(Lbd/Prod) \qquad (\checkmark A_1{:}\Delta.B_1) \lessdot^{\Sigma}_{\Gamma} (\checkmark A_2{:}\Delta.B_2)$$

$$\rightsquigarrow \quad A_1 \lessdot^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \lessdot^{\Sigma}_{\Gamma,\Delta} B_2$$

$$(Delay) \qquad [A_1 \ll_{\Delta} C_1].B_1 \lessdot^{\Sigma}_{\Gamma} [A_2 \ll_{\Delta} C_2].B_2$$

$$\rightsquigarrow \quad A_1 \lessdot^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \lessdot^{\Sigma}_{\Gamma,\Delta} B_2 \wedge C_1 \lessdot^{\Sigma}_{\Gamma} C_2$$

$$(Appl/Struct) \qquad (A_1; B_1) \lessdot^{\Sigma}_{\Gamma} (A_2; B_2)$$

$$\rightsquigarrow \quad A_1 \lessdot^{\Sigma}_{\Gamma} A_2 \wedge B_1 \lessdot^{\Sigma}_{\Gamma} B_2$$

The Algorithm $\mathcal{A}lg$

$$(Lbd/Prod) \qquad (\checkmark A_1{:}\Delta.B_1) \prec\!\!\!\prec^{\Sigma}_{\Gamma} (\checkmark A_2{:}\Delta.B_2)$$

$$\rightsquigarrow \quad A_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} B_2$$

$$(Delay) \qquad [A_1 \ll_{\Delta} C_1].B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma} [A_2 \ll_{\Delta} C_2].B_2$$

$$\rightsquigarrow \quad A_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} A_2 \wedge B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma,\Delta} B_2 \wedge C_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma} C_2$$

$$(Appl/Struct) \qquad (A_1; B_1) \prec\!\!\!\prec^{\Sigma}_{\Gamma} (A_2; B_2)$$

$$\rightsquigarrow \quad A_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma} A_2 \wedge B_1 \prec\!\!\!\prec^{\Sigma}_{\Gamma} B_2$$

$$\mathbf{[A \ll_{\Theta} C].B \rightarrow_{\sigma} B\sigma_{(A \prec\!\!\!\prec^{\Theta}_{\emptyset} C)}}$$

Termination of $\mathcal{A}lg$

- The relation $\rightsquigarrow\!\!\!\!\rightarrow$ is defined as the reflexive, transitive and compatible closure of $\rightsquigarrow$
- If T $\rightsquigarrow\!\!\!\!\rightarrow$ T′, with T′ a matching system in solved form then, we say that the matching algorithm $\mathcal{A}lg$ (taking as input the system T) succeeds
- The matching algorithm is clearly terminating (since all rules decrease the size of terms) and deterministic (no critical pairs), and of course, it works modulo α-conversion and Barendregt's hygiene-convention
- Starting form a given solved matching system of the form

$$\mathsf{T} \stackrel{\triangle}{=} \bigwedge_{\mathbf{i=0\ldots n}} \mathbf{X_i} \lessdot^{\mathbf{\Sigma}}_{\mathbf{\Delta_i}} \mathbf{A_i} \bigwedge_{\mathbf{j=0\ldots m}} \mathbf{a_j} \lessdot^{\mathbf{\Sigma}}_{\mathbf{\Delta_j}} \mathbf{a_j}$$

the corresponding substitution $\{A_1/X_1 \cdots A_n/X_n\}$ is exhibited.

Functional P^{2}T S

We require all specifications to be *functional*, *i.e.* for every $s_1, s_2, s_2', s_3, s_3' \in \mathcal{S}$, the following holds:

$$\begin{array}{llllll} (s_1, s_2) \in \mathcal{A} & \text{and} & (s_1, s_2') \in \mathcal{A} & \text{implies} & s_2 \equiv s_2' \\ (s_1, s_2, s_3) \in \mathcal{R} & \text{and} & (s_1, s_2, s_3') \in \mathcal{R} & \text{implies} & s_3 \equiv s_3'. \end{array}$$

Furthermore, we let $\mathcal{S}^\top$ denote the set of *topsorts*, *i.e.*

$$\mathcal{S}^\top = \{s \in \mathcal{S} \mid \forall s' \in \mathcal{S}.\ (s, s') \notin \mathcal{A}\}$$

and define a variant of delayed matching constraint as follows:

$$[A \ll_\Delta C]^\top.B = \begin{cases} B & \text{if } B \in \mathcal{S}^\top \\ [A \ll_\Delta C].B & \text{otherwise} \end{cases}$$

BACK