
Predicate logic over a type setup

Russell'08: Proof Theory meets Type Theory, Swansea, 15-16 March 2008 .

Peter Aczel

`petera@cs.man.ac.uk`

Manchester University

Type setups and logic enriched type theories

- Dependent type theories often use a fixed interpretation of the logical notions; e.g. **props-as-types** or some variant.
- Logic enriched type theories leave logic uninterpreted.

Plan of Talk

- Some dependent type theories and their logics
- Some category notions for type dependency
- Type setups
- Logic over a type setup
- The disjunction and existence properties
- Propositions as types

Some dependent type theories and their logics, 1

- **Basic Martin-Lof type theory**, with the forms of type

$(\Pi x : A)B(x)$, $(\Sigma x : A)B(x)$, $A_1 + A_2$, $N_k (k = 0, 1, \dots)$, $I(A, a_1, a_2)$

with $A_1 \rightarrow A_2 =_{def} (\Pi _ : A_1)A_2$, $A_1 \times A_2 =_{def} (\Sigma _ : A_1)A_2$.

Propositions-as-Types (à la Curry-Howard)

Proposition = Type

<i>Prop</i>	\perp	\top	$A_1 \supset A_2$	$A_1 \wedge A_2$	$A_1 \vee A_2$
<i>Type</i>	N_0	N_1	$A_1 \rightarrow A_2$	$A_1 \times A_2$	$A_1 + A_2$

<i>Prop</i>	$(\forall x : A)B(x)$	$(\exists x : A)B(x)$	$(a_1 =_A a_2)$
<i>Type</i>	$(\Pi x : A)B(x)$	$(\Sigma x : A)B(x)$	$I(A, a_1, a_2)$

Some dependent type theories and their logics, 2

- **Martin-Lof type theory** in a logical framework

This has a predicative type universe of sets/propositions:

Proposition = Set = Datatype

$(\forall x : A)B(x) : Prop$ can only be formed if $A : Set$.

- **Coq type theory** (a calculus of inductive constructions)

This has a predicative type universe Set of datatypes and an impredicative type $Prop$ where $(\Pi x : A)B(x) : Prop$ can be formed even when we do not have $A : Set$.

- The impredicative Russell-Prawitz representation of logic in $Prop$ is used; This representation can be given in terms of the Russell-Prawitz modality, J , where J assigns to each type A the type $JA : Prop$, where

$$JA \equiv (\Pi p : Prop)((A \rightarrow p) \rightarrow p).$$

Some dependent type theories and their logics, 3

Propositions-as-Types (à la Russell-Prawitz)

Proposition = type in *Prop*

<i>Prop</i>	\perp	\top	$A_1 \supset A_2$	$A_1 \wedge A_2$	$A_1 \vee A_2$
<i>Type</i>	JN_0	JN_1	$A_1 \rightarrow A_2$	$J(A_1 \times A_2)$	$J(A_1 + A_2)$

<i>Prop</i>	$(\forall x : A)B(x)$	$(\exists x : A)B(x)$	$(a_1 =_A a_2)$
<i>Type</i>	$(\Pi x : A)B(x)$	$J(\Sigma x : A)B(x)$	$JI(A, a_1, a_2)$

$$JA \equiv (\Pi p : Prop)((A \rightarrow p) \rightarrow p).$$

$$JA : Prop$$

Some dependent type theories and their logics, 4

- So the propositions-as-types- **à-la-Russell-Prawitz** representation of intuitionistic logic is the result of applying the propositions-as-types-**à-la-Curry-Howard** representation of intuitionistic logic followed by the **j-translation** of intuitionistic logic into itself.
- The **j-translation** generalises the $\neg\neg$ -translation for any unary connective **j** satisfying the laws

$$\phi \supset \mathbf{j}\phi \quad \text{and} \quad (\phi \supset \mathbf{j}\psi) \supset (\mathbf{j}\phi \supset \mathbf{j}\psi).$$

- Note: For types A, B ,

$$j1 : A \rightarrow JA \quad \text{and} \quad j2 : (A \rightarrow JB) \rightarrow (JA \rightarrow JB)$$

where

$$j1 \equiv (\lambda x : A, p : Prop, y : A \rightarrow p) y(x)$$
$$j2 \equiv (\lambda x : A \rightarrow JB, y : JA) y(JB)(x) \quad \blacksquare$$

Logic enriched type theories

These are obtained from type theories by simply adding a logic ‘on top’, using the types of a type theory as the possible ranges of the free and bound variables.

- **Dependently Sorted Logic** is obtained as a logic enrichment of an elementary type theory whose types and typed terms are just the sorts and sorted terms built up using sort and term constructors that may be dependent.
- Each sort has the form $F(t_1, \dots, t_n)$, where F is a sort constructor and t_1, \dots, t_n are terms whose types match the argument types of F .
- Makkai’s FOLDS is dependently sorted logic without function symbols.

Category notions for the semantics of type dependency

- Category with attributes **Cartmell 1978, Moggi 1991**,
Type category **Pitts 1997**
- Contextual category **Cartmell 1978, Streicher 1991**
- Category with families **Dybjer 1996, Hoffman 1997**
- Category with display maps (less general) **Taylor 1986**,
Lamarche 1987, Hyland and Pitts 1989
- Comprehension category (more general) **Jacobs 1991**
- other relevant notions: locally cartesian closed
categories, fibrations, indexed categories
- Type setups (for syntax) **new notion**

Category with families (CwF)

- a category $Ctxt$ of **contexts** Γ and **substitutions** $\sigma : \Delta \rightarrow \Gamma$, with a distinguished terminal object $(\)$,

- a functor $T : Ctxt^{op} \rightarrow Fam$ mapping

$$\Gamma \mapsto \{Term(\Gamma, A)\}_{A \in Type(\Gamma)}$$

and, if $\sigma : \Delta \rightarrow \Gamma$ then

$$A \in Type(\Gamma) \quad \mapsto A\sigma \in Type(\Delta)$$

$$a \in Term(\Gamma, A) \quad \mapsto a\sigma \in Term(\Delta, A\sigma)$$

- an assignment, to each context Γ and each $A \in Type(\Gamma)$, of a **comprehension** $(\Gamma.A, p_A, v_A)$ such that

$$p_A : \Gamma.A \rightarrow \Gamma \text{ and } v_A \in Term(\Gamma.A, Ap_A);$$

i.e. a terminal object in the category of (Γ', θ, a) such that $\theta : \Gamma' \rightarrow \Gamma$ and $a \in Term(\Gamma', A\theta)$.

The large CwF of sets

- $Ctxt = Set$ and, for each set I ,
- $Type(I)$ is the class of families of sets
 $A = \{A_i\}_{i \in I} \in Set^I$,
- $Term(I, A) = \prod_{i \in I} A_i$ and, if $\sigma : J \rightarrow I$ in Set ,
- $A\sigma = \{A_{\sigma j}\}_{j \in J}$,
- $a\sigma = \{a_{\sigma j}\}_{j \in J}$, for $a = \{a_i\}_{i \in I}$.
- $I.A = \sum_{i \in I} A_i$,
- $p_A(i, x) = i$ for $(i, x) \in I.A$,
- $v_A = \{x\}_{(i, x) \in I.A}$.

Type Setups, 1

The notion of a **type setup** abstracts away from the details of how terms and types are formed, but keeps the following notions.

- **contexts** Γ ,
- **substitutions** $\sigma : \Delta \rightarrow \Gamma$, between contexts, the contexts and substitutions forming a category $Ctxt$,
- $\iota_\Gamma : \Gamma \rightarrow \Gamma$ is the identity on Γ and $\sigma \circ \tau : \Lambda \rightarrow \Gamma$ is the composition of $\sigma : \Delta \rightarrow \Gamma$ and $\tau : \Lambda \rightarrow \Delta$.
- For each context Γ , there is the set $Type(\Gamma)$ of **Γ -types** A and the set $Term(\Gamma, A)$ of **Γ -terms** a of type A , for each Γ -type A .
- Substitutions must ‘act’ on types and terms to give a functor $T : Ctxt^{op} \rightarrow Fam$, where Fam is the category of set-indexed families of sets.

Type Setups, 2

- For each context Γ

$$T(\Gamma) = \{Term(\Gamma, A)\}_{A \in Type(\Gamma)}$$

- For each substitution $\sigma : \Delta \rightarrow \Gamma$, $T(\sigma) : T(\Gamma) \rightarrow T(\Delta)$ maps

$$A \in Type(\Gamma) \quad \mapsto \quad A\sigma \in Type(\Delta)$$

$$a \in Term(\Gamma, A) \quad \mapsto \quad a\sigma \in Term(\Delta, A\sigma)$$

- such that

$$A\iota_\Gamma = A \text{ and } a\iota_\Gamma = a$$

and if also $\tau : \Lambda \rightarrow \Delta$ then

$$A(\sigma \circ \tau) = (A\sigma)\tau \text{ and } a(\sigma \circ \tau) = (a\sigma)\tau.$$

Type Setups, 3

- Each context Γ is a finite sequence

$$x_1 : A_1, \dots, x_n : A_n$$

of typed variable declarations.

- The empty sequence $()$ is a context.
- If $\Gamma \equiv x_1 : A_1, \dots, x_n : A_n$ then

$\Gamma' \equiv \Gamma, x : A \equiv x_1 : A_1, \dots, x_n : A_n, x : A$ is a context iff

- Γ is a context,
- x is a variable, not in $\{x_1, \dots, x_n\}$ and
- $A \in Type(\Gamma)$.

Type Setups, 4

- If Γ, Δ are contexts, with

$$\Gamma \equiv x_1 : A_1, \dots, x_n : A_n$$

the each substitution $\Delta \rightarrow \Gamma$ has the form

$$[x_1 := a_1, \dots, x_n := a_n]_{\Delta \rightarrow \Gamma}.$$

- If $\Gamma' \equiv x_1 : A_1, \dots, x_n : A_n, x : A$ is a context then

$$\sigma' \equiv [\sigma, x := a]_{\Delta \rightarrow \Gamma'} \equiv [x_1 := a_1, \dots, x_n := a_n, x := a]_{\Delta \rightarrow \Gamma'}$$

is a substitution $\Delta \rightarrow \Gamma'$ iff

$\sigma \equiv [x_1 := a_1, \dots, x_n := a_n]_{\Delta \rightarrow \Gamma}$ is a substitution, and
 $a \in \text{Term}(\Delta, A\sigma)$.

Type Setups, 5

- If $\Gamma \equiv x_1 : A_1, \dots, x_n : A_n$ is a context then, for $i = 1, \dots, n$,

$$A_i \in \text{Type}(\Gamma) \text{ and } x_i \in \text{Term}(\Gamma, A_i).$$

- If $\sigma \equiv [x_1 := a_1, \dots, x_n := a_n]_{\Delta \rightarrow \Gamma}$ is a substitution then it is

the unique substitution $\Delta \rightarrow \Gamma$ such that, for $i = 1, \dots, n$,

$$x_i \sigma = a_i.$$

Type Setups, 6

- If Γ, Δ are contexts such that $\Gamma \subseteq \Delta$ (i.e. every declaration in Γ is also a declaration in Δ) then

$$Type(\Gamma) \subseteq Type(\Delta) \text{ and } Term(\Gamma, A) \subseteq Term(\Delta, A)$$

for each $A \in Type(\Gamma)$.

- Also, if $\Gamma \equiv x_1 : A_1, \dots, x_n : A_n$ then

$$\iota_{\Delta \rightarrow \Gamma} \equiv [x_1 := x_1, \dots, x_n := x_n]_{\Delta \rightarrow \Gamma}$$

is an inclusion substitution; i.e. for $A \in Type(\Gamma)$ and $a \in Term(\Gamma, A)$,

$$A\iota_{\Delta \rightarrow \Gamma} = A \text{ and } a\iota_{\Delta \rightarrow \Gamma} = a.$$

- If $\Gamma' \equiv \Gamma, x : A$ then $(\Gamma', \iota_{\Gamma' \rightarrow \Gamma}, x)$ is a comprehension.

Logic over a type setup

- We assume given a type setup with a **predicate signature** consisting of a set of predicate symbols, each assigned a context as its **arity**. We define the formulae and inference rules of a formal system of dependently sorted intuitionistic predicate logic with equality, whose sorts are the types of the type setup and whose individual terms are the terms of the setup.
- We use the predicate signature to define the **atomic Γ -formulae** to have the form

$$P(b_1, \dots, b_m)$$

where P is a predicate symbol of arity

$$\Delta \equiv (y_1 : B_1, \dots, y_m : B_m)$$

and b_1, \dots, b_m are the terms of a substitution

$$[y_1 := b_1, \dots, y_m := b_m]_{\Gamma \rightarrow \Delta}.$$

The Γ -Formulae

- The **formulae** are inductively generated using the following rules.
 - Every atomic Γ -formula $P(b_1, \dots, b_m)$ is a Γ -formula.
 - If a_1, a_2 are Γ -terms of a Γ -type A then $(a_1 =_A a_2)$ is a Γ -formula.
 - \perp and \top are Γ -formulae.
 - If ψ_1, ψ_2 are Γ -formulae then so is $(\psi_1 \square \psi_2)$, where $\square \in \{\wedge, \vee, \supset\}$.
 - If ψ_0 is a $(\Gamma, x : A)$ -formula then $(\nabla x : A)\psi_0$ is a Γ -formulae where $\nabla \in \{\forall, \exists\}$.

Substitution

We can define substitution into formulae in more or less the usual way by structural recursion on the formula. So, for each Γ -formula ϕ , we associate with each substitution $\tau : \Lambda \rightarrow \Gamma$ a Λ -formula $\phi\tau$ using the following equations.

- If $\phi \equiv P(b_1, \dots, b_m)$ then $\phi\tau \equiv P(b_1\tau, \dots, b_m\tau)$.
- If $\phi \equiv (a_1 =_A a_2)$ then $\phi\tau \equiv (a_1\tau =_{A\tau} a_2\tau)$.
- If $\phi \equiv \perp$ or \top then $\phi\tau \equiv \perp$ or \top respectively.
- If $\phi \equiv (\psi_1 \square \psi_2)$, where $\square \in \{\wedge, \vee, \supset\}$, then $\phi\tau \equiv (\psi_1\tau \square \psi_2\tau)$.
- If $\phi \equiv (\nabla x : A)\psi_0$, where $\nabla \in \{\forall, \exists\}$, then $\phi\tau \equiv (\nabla x : A\tau)\psi_0\tau'$, where $\tau' \equiv [\tau, x := x]_{(\Lambda, x:A\tau) \rightarrow (\Gamma, x:A)}$.

The rules of Inference, 1

- These are essentially the standard sequent formulation of the natural deduction rules for intuitionistic predicate logic with equality, using sequents of the form $(\Gamma) \Phi \rightarrow \phi$ where Γ is a context, Φ is a finite sequence of Γ -formulae and ϕ is a Γ -formula.
- e.g. here are the quantifier rules:

$(\forall I) \frac{(\Gamma, x : A) \Phi \Rightarrow \psi_0}{(\Gamma) \Phi \Rightarrow (\forall x : A)\psi_0}$	$(\forall E) \frac{(\Gamma) \Phi \Rightarrow (\forall x : A)\psi_0}{(\Gamma, x : A) \Phi \Rightarrow \psi_0[a/x]}$
$(\exists I) \frac{(\Gamma) \Phi \Rightarrow \psi_0[a/x]}{(\Gamma) \Phi \Rightarrow (\exists x : A)\psi_0}$	$(\exists E) \frac{(\Gamma) \Phi \Rightarrow (\exists x : A)\psi_0 \quad (\Gamma, x : A) \Phi, \psi_0 \Rightarrow \phi}{(\Gamma) \Phi \Rightarrow \phi}$

- Here $a \in Term(\Gamma, A)$ and $[a/x] \equiv [\iota_{\Gamma}, x := a]_{\Gamma \rightarrow (\Gamma, x:A)}$.

The rules of Inference, 2

- And here are the equality rules:

$$\boxed{\begin{array}{l} (= I) \frac{}{(\Gamma)\Phi \Rightarrow (a = Aa)} \qquad (= E) \frac{(\Gamma)\Phi \Rightarrow (a_1 =_A a_2) \quad (\Gamma)\Phi \Rightarrow \psi_0[a_1/x]}{(\Gamma)\Phi \Rightarrow \psi_0[a_2/x]} \end{array}}$$

where a, a_1, a_2 are Γ -terms of type A .

The disjunction and existence properties, 1

Let Φ be a finite sequence of Δ -formulae.

- Φ has the **disjunction property** if, for all Δ -formulae ψ_1, ψ_2 ,

$\vdash (\Delta) \Phi \Rightarrow (\psi_1 \vee \psi_2)$ implies $\vdash (\Delta) \Phi \Rightarrow \psi_i$ for some $i \in \{1, 2\}$.

- Φ has the **existence property** if, for all Δ -types A and every $(\Delta, x : A)$ -formula ψ_0 ,

$\vdash (\Delta) \Phi \Rightarrow (\exists x : A)\psi_0$ implies $\vdash (\Delta) \Phi \Rightarrow \psi_0[a/x]$
for some $a \in \text{Term}(\Delta, A)$.

- Φ is **saturated** if it has both properties.

- When is Φ saturated?

The disjunction and existence properties, 2

Define $(\Delta) \Phi \mid \psi$ if $\psi \in \mathcal{X}$, for Δ -formulae ψ , is determined recursively using the following, where

$$\mathcal{X}_0 = \{\psi \mid \vdash (\Delta) \Phi \Rightarrow \psi\}.$$

- If ψ is atomic, an equality or \perp or \top then $\psi \in \mathcal{X}$ iff $\psi \in \mathcal{X}_0$.
- $(\psi_1 \wedge \psi_2) \in \mathcal{X}$ iff $\psi_1 \in \mathcal{X}$ and $\psi_2 \in \mathcal{X}$.
- $(\psi_1 \vee \psi_2) \in \mathcal{X}$ iff $\psi_1 \in \mathcal{X}$ or $\psi_2 \in \mathcal{X}$.
- $(\psi_1 \supset \psi_2) \in \mathcal{X}$ iff $\psi_1 \in \mathcal{X}$ implies $\psi_2 \in \mathcal{X}$ and $\psi \in \mathcal{X}_0$.
- $(\forall x : A)\psi_0 \in \mathcal{X}$ iff $\psi_0[a/x] \in \mathcal{X}$ for all $a \in Term(\Delta, A)$ and $\psi \in \mathcal{X}_0$.
- $(\exists x : A)\psi_0 \in \mathcal{X}$ iff $\psi_0[a/x] \in \mathcal{X}$ for some $a \in Term(\Delta, A)$.

The disjunction and existence properties, 3

Theorem: The following are equivalent:

1. Φ is saturated.
2. $(\Delta) \Phi | \phi$ for all $\phi \in \Phi$.
3. For every Δ -formula ψ

$$\vdash (\Delta) \Phi \Rightarrow \psi \iff (\Delta) \Phi | \psi.$$

- e.g. \emptyset is saturated

Lemma 1:

1. $(\Delta) \Phi | \psi$ implies $\vdash (\Delta) \Phi \Rightarrow \psi$,
2. If $\vdash (\Delta) \Phi \Rightarrow \phi$ then $[(\Delta) \Phi, \phi | \psi$ iff $(\Delta) \Phi | \psi]$.

Proof: By structural induction on ψ .

The disjunction and existence properties, 4

Lemma 2: If $(\Delta, \forall\Gamma)\Phi|\phi$ for all $\phi \in \Phi$ then

$\vdash (\Delta, \Gamma)\Phi \Rightarrow \psi$ implies $(\Delta, \forall\Gamma)\Phi|\psi$.

Definition: $(\Delta, \forall\Gamma)\Phi|\psi$ iff

$\vdash (\Delta, \Gamma)\Phi \Rightarrow \psi$ and $(\Delta)\Phi\tau|\psi\tau$ for all $\tau \in \text{Subst}(\Delta, \Gamma)$,

where $\text{Subst}(\Delta, \Gamma)$ is the set of all substitutions $\tau : \Delta \rightarrow \Delta, \Gamma$ such that $y\tau = y$ for all $y \in \text{var}(\Delta)$.

Corollary: If $(\Delta)\Phi|\phi$ for all $\phi \in \Phi$ then

$\vdash (\Delta)\Phi \Rightarrow \psi$ implies $(\Delta)\Phi|\psi$.

Types as propositions

- Think of a type A as a proposition which is true if there is a term of type A .
- If A_1, \dots, A_n, A are Δ -types and x_1, \dots, x_n are distinct variables not declared in Δ so that $\Delta' \equiv (\Delta, x_1 : A_1, \dots, x_n : A_n)$ is a context we write

$$(\Delta) A_1, \dots, A_n \equiv\Rightarrow A$$

if there is a Δ' -term of type A .

- Also, for each Δ -type A let $!A$ be the Δ -formula $(\exists _ : A) \top$.

Theorem: The following are equivalent:

1. $\vdash (\Delta) !A_1, \dots, !A_n \Rightarrow !A$.
2. $\vdash (\Delta') \Rightarrow !A$.
3. $(\Delta) A_1, \dots, A_n \equiv\Rightarrow A$

Π -types, 1

- We say that a type setup **has Π -types** if the standard formation, introduction, elimination and computation rules for Π -types are correct for the type setup; i.e. if $\Gamma' \equiv \Gamma, x : A$ is a context then there are the following assignments:

$$B \in \text{Type}(\Gamma') \quad \mapsto \quad (\Pi x : A)B \in \text{Type}(\Gamma),$$

$$b \in \text{Term}(\Gamma', B) \quad \mapsto \quad (\lambda x)b \in \text{Term}(\Gamma, (\Pi x : A)B),$$

$$\left. \begin{array}{l} f \in \text{Term}(\Gamma, (\Pi x : A)B) \\ a \in \text{Term}(\Gamma, A) \end{array} \right\} \mapsto \text{app}(f, a) \in \text{Term}(\Gamma, B[a/x])$$

such that if $f = (\lambda x)b$ then $\text{app}(f, a) = b[a/x]$.

Π -types, 2

- These must commute with substitution; i.e. for each $\sigma : \Delta \rightarrow \Gamma$,

$$((\Pi x : A)B)\sigma = (\Pi x : A\sigma)B\sigma',$$

$$((\lambda x)b)\sigma = (\lambda x)b\sigma',$$

$$\text{app}(f, a)\sigma = \text{app}(f\sigma, a\sigma),$$

where $\sigma' \equiv [\sigma, x := x]_{\Delta \rightarrow \Gamma'} : \Delta \rightarrow \Gamma'$.

- Also, if $y \notin \text{var}(\Gamma)$ then

$$(\Pi x : A)B = (\Pi y : A)B[y/x] \text{ and } (\lambda x)b = (\lambda y)b[y/x].$$

- The requirement that the type setup has other forms of type can be explained in a similar way.

Propositions as types, 1

- We assume given a type setup with predicate signature that has the forms of type $(\Pi x : A)B$, $(\Sigma x : A)B$, with the defined forms $A \rightarrow B$ and $A \times B$, the forms of type $A_1 + A_2$, $N_k (k = 0, 1, \dots)$ and also has associated with each predicate symbol P of arity the context Δ a Δ -type $Pr(P)$.
 - Then the **propositions-as-types** interpretation recursively associates with each Γ -formula ϕ a Γ -type $Pr(\phi)$ using the following rules.
 - If ϕ is the atomic Γ -formula $P(b_1, \dots, b_m)$ then $Pr(\phi)$ is the Γ -type $Pr(P)\sigma$ where
$$\sigma = [y_1 := b_1, \dots, y_m := b_m]_{\Gamma \rightarrow \Delta}.$$
 - If ϕ is $(a_1 =_A a_2)$ then $Pr(\phi)$ is the Γ -type $I(A, a_1, a_2)$.
 - If ϕ is \perp or \top then $Pr(\phi)$ is N_0 or N_1 respectively.
-

Propositions as types, 2

- If ϕ is $(\psi_1 \square \psi_2)$, where \square is one of \wedge, \vee, \supset then $Pr(\phi)$ is $(Pr(\psi_1) \square' Pr(\psi_2))$ where \square' is the corresponding one of $\times, +, \rightarrow$.
- If ϕ is $(\nabla x : A)\psi_0$ where ∇ is one of \forall, \exists then $Pr(\phi)$ is $(\nabla' x : A)Pr(\psi_0)$ where ∇' is the corresponding one of Π, Σ .

Proposition: The interpretation is sound; i.e. if $\vdash (\Delta) \Phi \Rightarrow \phi$ then $(\Delta)\Phi \Rightarrow_{Pr} \phi$, where, if $\Phi \equiv \phi_1, \dots, \phi_k$ then

$$(\Delta)\Phi \Rightarrow_{Pr} \phi \quad \text{iff} \quad (\Delta) Pr(\phi_1), \dots, Pr(\phi_k) \Rightarrow Pr(\phi).$$

- But the interpretation is not complete as the type theoretic axiom of choice holds; i.e.

Propositions as types, 3

Proposition: For any context Γ and any distinct variables x, y , not declared in Γ , if A is a Γ -type, B is a $(\Gamma, x : A)$ -type and θ is a $(\Gamma, x : A, y : B)$ -formula then

$$\begin{aligned} & (\Gamma) (\forall x : A)(\exists y : B)\theta \\ & \quad \Rightarrow_{Pr} (\exists z : (\Pi x : A)B)(\forall x : A)\theta[app(z, x)/y]. \end{aligned}$$

- If Σ is a set of sequents we write $\Sigma \vdash (\Gamma) \Phi \Rightarrow \phi$ if the sequent $(\Gamma) \Phi \Rightarrow \phi$ can be derived using the rules of inference for intuitionistic predicate logic and the sequents in Σ as additional axioms.

- Let AC be the set of all sequents

$$(\Gamma) (\forall x : A)(\exists y : B)\theta \Rightarrow (\exists z : (\Pi x : A)B)(\forall x : A)\theta[app(z, x)/y]$$

and let PaT be the set of all sequents having one of the forms $(\Gamma) \phi \Rightarrow !Pr(\phi)$ or $(\Gamma) !Pr(\phi) \Rightarrow \phi$.

Propositions as types, 4

Theorem: The following are equivalent

1. $(\Delta) \Phi \equiv_{Pr} \phi,$
2. $PaT \vdash (\Delta) \Phi \Rightarrow \phi,$
3. $AC \cup \Sigma \vdash (\Delta) \Phi \Rightarrow \phi,$

where Σ is the set of sequents having one of the forms:

- $(\Delta) P \Rightarrow !Pr(P),$
- $(\Delta) !Pr(P) \Rightarrow P,$
- $(\Gamma) \Rightarrow (\forall _ : N_0) \perp,$
- $(\Gamma) \Rightarrow (\forall _ : A + B) (!A \vee !B),$
- $(\Gamma) \Rightarrow (\forall _ : I(A, a_1, a_2)) (a_1 =_A a_2).$

Here P is a predicate symbol of arity Δ , A, B are Γ -types and

a_1, a_2 are Γ -terms of type A .