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

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

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

Propositions-as-Types (à la Curry-Howard)
Proposition = Type

$Prop$	$\bot$	$\top$	$A_1 \supset A_2$	$A_1 \wedge A_2$	$A_1 \vee A_2$
$Type$	N_0	N_1	$A_1 \to A_2$	$A_1 \times A_2$	$A_1 + A_2$

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

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 asigns 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 $Prop$

$Prop$	$\bot$	$\top$	$A_1 \supset A_2$	$A_1 \wedge A_2$	$A_1 \vee A_2$
$Type$	JN_0	JN_1	$A_1 \to A_2$	$J(A_1 \times A_2)$	$J(A_1 + A_2)$

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

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

$$JA : Prop$$

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

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

- Note: For types A, B,

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

where $\begin{array}{l} j1 \equiv (\lambda x : A, p : Prop, y : A \rightarrow p)\, y(x) \\ j2 \equiv (\lambda x : A \rightarrow JB, y : JA)\, y(JB)(x) \end{array}$.

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 $F(t_1, \ldots, t_n)$, where F is a sort constructor and $t_1, \ldots, t_n$ 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 $\sigma : \Delta \to \Gamma$, with a distinguished terminal object $(\,)$,
- a functor $T : Ctxt^{op} \to Fam$ mapping

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

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

$$\begin{array}{ll} A \in Type(\Gamma) & \mapsto A\sigma \in Type(\Delta) \\ a \in Term(\Gamma, A) & \mapsto a\sigma \in Term(\Delta, A\sigma) \end{array}$$

- an assignment, to each context Γ and each $A \in Type(\Gamma)$, of a comprehension $(\Gamma.A, \mathsf{p}_A, \mathsf{v}_A)$ such that

$$\mathsf{p}_A : \Gamma.A \to \Gamma \text{ and } \mathsf{v}_A \in Term(\Gamma.A, A\mathsf{p}_A);$$

  i.e. a terminal object in the category of (Γ', θ, a) such that $\theta : \Gamma' \to \Gamma$ 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 $\sigma : J \to I$ in Set,
- $A\sigma = \{A_{\sigma j}\}_{j \in J}$,
- $a\sigma = \{a_{\sigma j}\}_{j \in J}$, for $a = \{a_i\}_{i \in I}$.
- $I.A = \sum_{i \in I} A_i$,
- $\mathsf{p}_A(i, x) = i$ for $(i, x) \in I.A$,
- $\mathsf{v}_A = \{x\}_{(i,x) \in I.A}$.

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 $\sigma : \Delta \to \Gamma$, between contexts, the contexts and substitutions forming a category $Ctxt$,
- $\iota_\Gamma : \Gamma \to \Gamma$ is the identity on Γ and $\sigma \circ \tau : \Lambda \to \Gamma$ is the composition of $\sigma : \Delta \to \Gamma$ and $\tau : \Lambda \to \Delta$.
- 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 $T : Ctxt^{op} \to Fam$, 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 $\sigma : \Delta \to \Gamma$, $T(\sigma) : T(\Gamma) \to T(\Delta)$ maps

$$\begin{array}{ll} A \in Type(\Gamma) & \mapsto A\sigma \in Type(\Delta) \\ a \in Term(\Gamma, A) & \mapsto a\sigma \in Term(\Delta, A\sigma) \end{array}$$

- such that

$A\iota_\Gamma = A$ and $a\iota_\Gamma = a$

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

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

Type Setups, 3

- Each context Γ is a finite sequence

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

of typed variable declarations.

- The empty sequence $(\ )$ is a context.

- If $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$ then

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

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

Type Setups, 4

- If Γ, Δ are contexts, with

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

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

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

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

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

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

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

Type Setups, 5

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

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

- If $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]_{\Delta \to \Gamma}$ is a substitution then it is the unique substitution $\Delta \to \Gamma$ such that, for $i = 1, \ldots, n$,

$$x_i \sigma = a_i.$$

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 $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$ then

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

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

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

- If $\Gamma' \equiv \Gamma, x : A$ then $(\Gamma', \iota_{\Gamma' \to \Gamma}, x)$ 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

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

where P is a predicate symbol of arity

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

and $b_1, \ldots, b_m$ are the terms of a substitution

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

The Γ-Formulae

• The formulae are inductively generated using the following rules.

- Every atomic Γ-formula $P(b_1, \ldots, b_m)$ 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.
- $\bot$ and $\top$ are Γ-formulae.
- If ψ_1, ψ_2 are Γ-formulae then so is $(\psi_1 \Box \psi_2)$, where $\Box \in \{\wedge, \vee, \supset\}$.
- 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 $\tau : \Lambda \to \Gamma$ a Λ-formula $\phi\tau$ using the following equations.

- If $\phi \equiv P(b_1, \ldots, b_m)$ then $\phi\tau \equiv P(b_1\tau, \ldots, b_m\tau)$.
- If $\phi \equiv (a_1 =_A a_2)$ then $\phi\tau \equiv (a_1\tau =_{A\tau} a_2\tau)$.
- If $\phi \equiv \bot$ or $\top$ then $\phi\tau \equiv \bot$ or $\top$ respectively.
- If $\phi \equiv (\psi_1 \Box \psi_2)$, where $\Box \in \{\wedge, \vee, \supset\}$, then $\phi\tau \equiv (\psi_1\tau \Box \psi_2\tau)$.
- 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 $\tau' \equiv [\tau, x := x]_{(\Lambda, x:A\tau) \to (\Gamma, x:A)}$.

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:

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

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

- 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:

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

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,

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

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

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

- Φ is saturated if it has both properties.
- When is Φ saturated?

The disjunction and existence properties, 2

Define $(\Delta)\ \Phi|\psi$ 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

$$\vdash (\Delta)\ \Phi \Rightarrow \psi \quad \Longleftrightarrow \quad (\Delta)\ \Phi|\psi.$$

- 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 $\quad [(\Delta)\Phi, \phi|\psi$ 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

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

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

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

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

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

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

Types as propositions

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

$$(\Delta)\ A_1, \ldots, A_n \Rrightarrow A$$

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. $\vdash (\Delta)\ !A_1, \ldots, !A_n \Rightarrow !A$.
2. $\vdash (\Delta') \Rightarrow !A$.
3. $(\Delta)\ A_1, \ldots, A_n \Rrightarrow A$

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:

$$B \in Type(\Gamma') \mapsto (\Pi x : A)B \in Type(\Gamma),$$

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

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

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

Π-types, 2

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

$$\begin{aligned} ((\Pi x : A)B)\sigma &= (\Pi x : A\sigma)B\sigma', \\ ((\lambda x)b)\sigma &= (\lambda x)b\sigma', \\ app(f, a)\sigma &= app(f\sigma, a\sigma), \end{aligned}$$

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

- Also, if $y \notin var(\Gamma)$ 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 $A \to B$ and $A \times B$, the forms of type $A_1 + A_2, N_k (k = 0, 1, \ldots)$ 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 $P(b_1, \ldots, b_m)$ then $Pr(\phi)$ is the Γ-type $Pr(P)\sigma$ where
$$\sigma = [y_1 := b_1, \ldots, y_m := b_m]_{\Gamma \to \Delta}.$$
- If ϕ is $(a_1 =_A a_2)$ then $Pr(\phi)$ is the Γ-type $I(A, a_1, a_2)$.
- If ϕ is $\bot$ or $\top$ then $Pr(\phi)$ is N_0 or N_1 respectively.

Propositions as types, 2

- If ϕ is $(\psi_1 \Box \psi_2)$, where $\Box$ is one of $\wedge, \vee, \supset$ then $Pr(\phi)$ is $(Pr(\psi_1)\Box' Pr(\psi_2))$ where $\Box'$ 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 $(\Delta)\Phi \Rrightarrow_{Pr} \phi$, where, if $\Phi \equiv \phi_1, \ldots, \phi_k$ then

$$(\Delta)\Phi \Rrightarrow_{Pr} \phi \quad \text{iff} \quad (\Delta)\ Pr(\phi_1), \ldots, Pr(\phi_k) \Rrightarrow Pr(\phi).$$

- 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

$$(\Gamma)\ (\forall x : A)(\exists y : B)\theta \Longrightarrow_{Pr} (\exists z : (\Pi x : A)B)(\forall x : A)\theta[app(z, x)/y].$$

- 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. $(\Delta)\ \Phi \Rrightarrow_{Pr} \phi$,
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:

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

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.