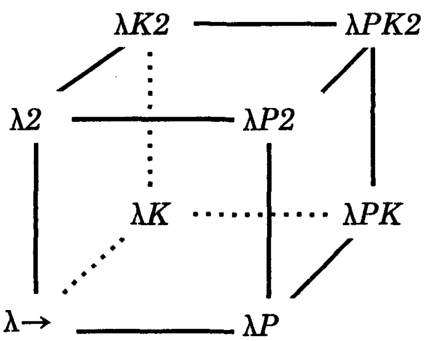
Conservativity of Typed Lambda Calculus over Intuitionistic Logic

1J−2

Ken-etsu FUJITA, Atsushi TOGASHI, Shoichi NOGUCHI
Research Institute of Electrical Communication, Tohoku University

1. Introduction

We investigate the conservativity of typed λ-calculi placed in λ-cube [1] (figure-1) over intuitionistic logical systems:



λ-cube

figure-1

Some of typed λ-calculi in λ-cube are well-known as shown in the following table (table-2):

λL	already existing system
λ→	*typed λ-calculus (Church)*
λ2	*F (Girard)* *2nd-order/polymorphic typed λ-calculus (Reynolds)*
λP	*AUT-QE (De Bruijn)* *LF (Harper, Honsell and Plotkin)*
λK2	*F_ω (Girard)*
λPK2	*Calculus of constructions (Coquand et Huet)*

table-2

It is already proved that using formulae-as-types notion [2], the interpretations from typed λ-calculi in λ-cube to intuitionistic logical systems are sound as depicted in the following table (table-3) [3].

Firstly we summarize already obtained results about the conservativity in 2. Secondly, in order to solve the open problem that λP2 is conservative over PRED2 (denoted 2nd-order int. predicate logic) or not, for a first trial, we will prove the reduced problem that $\lambda P2^-$ is conservative over PRED2' (denoted PRED2 with $(*_t, *_t)$) in 3.

L (logic) ⇨	λL
Minimal propositional logic	λ→
Second. order int. prop. logic	λ2
Min. many sorted pred. logic	λP
Weak higher order minimal propositional logic	λK
Second order intuitionistic manys. predicate logic	λP2
Higher order int. prop. logic	λK2
Weak higher order minimal many sorted predicate	λPK
Higher order intuitionistic many sorted predicate logic	λPK2

table-3

2. Already obtained results about conservativity

2-1. Type systems on the left plane are conservative over the propositional logics

The type systems on the left plane in λ-cube are clearly conservative over the corresponding propositional logics. Using the restriction of [6], the restricted type systems for logics are same as the original type systems in the sense that $* = *_p$.

2-2. λP is conservative over PRED

[4] proved it using the translation from λP to PRED, which consists of the projection from λP to λ→ (eliminating dependent types) and the relativization of quantifiers.

2-3. λPK2 is not conservative over higher order logic

[5] proved it as follows (0 is a sort, φ is a formula in which x does not occur free and P is a predicate): Even though $\forall x{:}0.\phi \rightarrow (\phi \rightarrow \phi)$, we can't find a proof of $P(\forall x{:}0.\phi) \rightarrow P(\phi \rightarrow \phi)$, because of non-extensionality of logic. In λPK2, on the other hand, let $\Gamma = 0{:}*, c{:}0, P{:}* \rightarrow *, \phi{:}*$ (* means the collection of all types), then it has a following proof:

$$\Gamma \vdash \lambda z{:}P(\Pi x{:}0.\phi).\lambda\alpha{:}*.\lambda h{:}(\Pi\psi{:}*.P(\psi\rightarrow\phi)\rightarrow\alpha).\ h0z \quad : P(\Pi x{:}0.\phi)\rightarrow\exists\psi{:}*.P(\psi\rightarrow\phi)$$

The above proof is based on that the logic should be non-extensional and that the translation (based on formulae-as-types) from higher order logic to $\lambda PK2$ allows us to identify formulae *and* sorts as types.

3. $\lambda P2^-$ is conservative over PRED2'

3-1. $\lambda P2^-$

$\lambda P2^-$ has the same rules as $\lambda P2$, i.e., $(*, *)$, $(*, K)$, $(K, *)$. The difference between them is as follows (x, X and Y are variables, c denotes a constant and y is a fresh variable):

Terms

$M ::= x \mid c \mid \lambda x{:}B.M \mid \lambda y{:}A.M \mid MM \mid \lambda X{:}K.M \mid MA$

Types or *families*

$A ::= X \mid \Pi x{:}B.A \mid \Pi y{:}A.A \mid \Pi X{:}K.A \mid \lambda x{:}B.A \mid AM$

$B ::= Y \mid \Pi x{:}B.B \mid \lambda x{:}B.B \mid BM$

Kinds

$K ::= * \mid \Pi x{:}B.K$

That is, in the expressions $\lambda x{:}A_1.A_2$, $\Pi x{:}A_1.A_2$ and $\Pi x{:}A_1.K$, A_1 can't take a universal type $\Pi X{:}K.A$, i.e., only take B which excludes universal types.

3-2. PRED2' (denoted PRED2 with $(*_t, *_t)$)

PRED2' is obtained by adding to PRED2 higher order functions and predicates over function spaces. Sorts are formed by A (ground sort) and Ω (sort for formulae) as follows:

Sorts

$S ::= St \mid Sp$

$St ::= A \mid St \rightarrow St \qquad Sp ::= \Omega \mid St \rightarrow Sp$

SORT is the set of sorts. Terms are constructed by x (variable) and c (constant) as below:

Terms

$t ::= x \mid c \mid t(t) \mid \lambda x.t \mid t \supset t \mid \forall x \in S.t$

For each term t, a sort $[t] \in$ **SORT** is assigned where $[\ ]$:**TERM**→**SORT**. **TERM** denotes the set of terms. Constructors are defined as terms t such that $[t] \in$ **SORT-p**. R denotes x where $[x] \in$ **SORT-p**.

Constructors

$C ::= R \mid Ct \mid \lambda x.C \mid C \supset C \mid \forall x \in S.C$

CONSTR denotes the set of constructors.

3-3. Translations Tr and | | from $\lambda P2^-$ to PRED2'

Tr applying to K or A gives a sort Sp or a constructor, and | | applying to K, B or M gives **SORT-t**, a sort St or a term as follows:

$\lambda P2^-$	$\Rightarrow$ PRED2'
Type A	Constructor
X	R
$\Pi x{:}B.A$	$\forall x \in \lvert B\rvert.Tr(A)$
$\Pi y{:}A_1.A_2$	$Tr(A_1) \supset Tr(A_2)$
$\Pi X{:}K.A$	$\forall x \in Tr(K).Tr(A)$
$\lambda x{:}B.A$	$\lambda x.Tr(A)$ where $[x] = \lvert B\rvert$
AM	$Tr(A)(\lvert M\rvert)$
Kind	Sort Sp
$*$	Ω
$\Pi x{:}B.K$	$\lvert B\rvert \rightarrow Tr(K)$

where | | is defined as follows:

Type B	Sort St
Y	A
$\Pi x{:}B_1.B_2$	$\lvert B_1\rvert \rightarrow \lvert B_2\rvert$
$\lambda x{:}B_1.B_2$	$\lvert B_2\rvert$
BM	$\lvert B\rvert$
Kind	
$*$	**SORT-t**
$\Pi x{:}B.K$	$\lvert K\rvert$
Term M where M:B	
x	x
c	c
$\lambda x{:}B.M$	$\lambda x.\lvert M\rvert$ where $[x] = \lvert B\rvert$
M_1M_2	$\lvert M_1\rvert(\lvert M_2\rvert)$

Proposition 1:

If $\Gamma \vdash_{\lambda P2^-} M{:}B$, then $[\lvert M\rvert] = \lvert B\rvert$.

Proposition 2:

If $\Gamma \vdash_{\lambda P2^-} A{:}K$, then $[Tr(A)] = \dot{T}r(K)$.

An assumption $Tr(\Gamma)$ is defined as $\{\, Tr(A) \mid z{:}A \in \Gamma$ for some z (x or c) and A such that $\Gamma \vdash A{:}* \,\}$.

Proposition 3:

If $\Gamma \vdash_{\lambda P2^-} M{:}A{:}*$ for some M, then $Tr(\Gamma) \vdash_{PRED2'} Tr(A)$.

[PROOF] By induction on a derivation $\Gamma \vdash_{\lambda P2^-} M{:}A$ and on M.

The translation $^+$ from PRED2' to $\lambda P2^-$ is defined by the notion of formulae-as-types [2], i.e., if ϕ is a formula ($[\phi] = \Omega$), then $\phi^+{:}*$.

Proposition 4:

If $C \in$ **CONSTR**, then $Tr(C^+) \equiv C$.

Theorem 1:

If $\Gamma \vdash_{\lambda P2^-} M{:}\phi^+$ for some M, then $Tr(\Gamma) \vdash_{PRED2'} \phi$.

<u>Remark</u> If we had allowed the term MB (we had *only* MA) in $\lambda P2^-$, then the proof of Theorem 1 would fail in Proposition 3.

4. Concluding remarks

In order to solve the open problem that $\lambda P2$ is conservative over PRED2 or not, for a first trial, we have proved Theorem 1 that $\lambda P2^-$ is conservative over PRED2'.

References

[1] Barendregt, H. : *Introduction to Generalised Type Systems*. Proceedings of the 3rd Italian Conference on Theoretical Computer Science, November, 1989.

[2] Howard, W. A. : *The Formulae-as-types Notion of Construction*. Academic Press, 1980 pp.479-490.

[3] Geuvers, H. and Barendregt, H. : *The Interpretation of Logics in Type Systems. Department of Computer Science*, University of Nijmegen, The Netherlands, August 1988.

[4] Barendsen, E. and Geuvers, H. : *Conservativity of λP over PRED*. University of Nijmegen, 1989.

[5] Geuvers, H. : *Theory of con[illegible]uction is not conservative over higher order logic*. University of Nijmegen, 1989.

[6] Fujita, K., Togashi, A. and Noguchi, S. : *Relationship between logic and type system*. 6th Conference Proceedings Japan Society for Software Science and Technology, 1989.

JUNET: fujiken@riec.tohoku.ac.jp