Canonical typing and Π-conversion in the Barendregt Cube

Journal of Functional Programming 6(2), 1995*

Fairouz Kamareddine †
Department of Computing Science
17 Lilybank Gardens
University of Glasgow
Glasgow G12 8QQ, Scotland
email: `fairouz@dcs.glasgow.ac.uk`
and

Rob Nederpelt
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O.Box 513
5600 MB Eindhoven, the Netherlands
email: `wsinrpn@win.tue.nl`

June 5, 1996

*First of all, we are very grateful to our colleague Bert van Benthem Jutting who has read draft versions of the manuscript, and who has made very useful suggestions. Furthermore, we are grateful for the discussions with Henk Barendregt, Roel Bloo, Tijn Borghuis, Herman Geuvers, Kevin Hammond, Bart-Jan de Leuw, Simon Peyton-Jones, Erik Poll and Phil Wadler, and for the helpful remarks received from them. Last but not least, we are grateful to the anonymous referees for their constructive comments and criticisms.

†Kamareddine is grateful to the Department of Mathematics and Computing Science, Eindhoven University of Technology, for their financial support and hospitality from October 1991 to September 1992, and during various short visits in 1993 and 1994. Furthermore, Kamareddine is grateful to the Department of Mathematics and Computer Science, University of Amsterdam, and in particular to Jan Bergstra and Inge Bethke for their hospitality during the preparation of this article, and to the Dutch organisation of research (NWO) for its financial support. Last but not least, Kamareddine is grateful to the ESPRIT Basic Action for Research project "Types for Proofs and Programming" for its financial support.

Abstract

In this article, we extend the Barendregt Cube with Π-conversion (which is the analogue of β-conversion, on product type level) and study its properties. We use this extension to separate the problem of whether a term is typable from the problem of what is the type of a term.

Keywords: *Barendregt Cube,* Π*-conversion, Canonical Typing, Typable Terms, Subject Reduction, Church Rosser, Strong Normalisation.*

Contents

1 Introduction **3**

2 The formal machinery of the Cube **6**

3 The ordinary typing relation $\vdash_\beta$ and its properties **8**

4 The extended typing relation $\vdash_{\beta\Pi}$ and its properties **10**

5 The canonical typing operator τ and its properties **14**

6 The typability relation $\vdash$ and its properties **16**

7 Conclusion **19**

1 Introduction

At the end of the nineteenth century, types did not play a role in mathematics or logic, unless at the meta-level, in order to distinguish between different 'classes' of objects. Frege's formalization of logical reasoning, as explained in the *Begriffsschrift* ([Frege 1879]), was untyped. Only after the discovery of Russell's paradox, undermining Frege's work, one may observe various formulations of typed theories. Types could explain away the paradoxical instances. The first theory which aimed at doing so, was that of Russell and Whitehead, as exposed in their famous *Principia Mathematica* ([Whitehead and Russell 1910]). Their 'ramified theory of types' has later been adapted and simplified by Hilbert and Ackermann ([Hilbert and Ackermann 1928]).

Church was the first to define a type theory 'as such', almost a decade after he developed a theory of functionals which is nowadays called λ-*calculus* ([Church 1932]). This calculus was used for defining a notion of computability that turned out to be of the same power as Turing-computability or general recursiveness. However, the original, untyped version did not work as a foundation for mathematics. In order to come round the inconsistencies in his proposal for logic, Church developed the 'simple theory of types' ([Church 1940]).

From then till the present day, research on the area has grown and one can find various reformulations of type theories. A taxonomy of type systems has recently been given by Barendregt ([Barendregt 92]). A version of Church's simple theory of types is found in this taxonomy under the name $\lambda_{\rightarrow}$. This $\lambda_{\rightarrow}$ has, apart from *type variables*, so-called *arrow-types* of the form $A \rightarrow B$. In higher type theories, arrow-types are replaced by dependent products $\Pi_{x:A}.B$, where B may contain x as a free variable, and thus may *depend on* x. This means that abstraction can be over types, similarly to the abstraction over terms: $\lambda_{x:A}.b$.

But, once we allow abstraction over types, it would be nice to discuss the reduction rules which govern these types. We propose reduction rules which treat alike types and terms. That is, not only we have $(\lambda_{x:A}.b)C \rightarrow_\beta b[x := C]$, but also $(\Pi_{x:A}.B)C \rightarrow_\beta B[x := C]$.

This strategy of permitting Π-*application* $(\Pi_{x:A}.B)C$ in term construction is not commonly used, yet is desirable for the following reasons:

1. Π-reduction behaves like β-reduction. One may say that β-reduction has been invented as an expedient in order to forebode a possible substitution. So why does one use a direct substitution as in equation 1 below, (which is used almost everywhere) if β-reduction can be used to do the job, as shown in equation 2? (We omit the contexts, for the sake of simplicity):

$$\text{If } f : \Pi_{x:A}.B \text{ and } a : A, \text{ then } fa : B[x := a] \tag{1}$$

$$\text{If } f : \Pi_{x:A}.B \text{ and } a : A, \text{ then } fa : (\Pi_{x:A}.B)a \text{ (which } \beta\text{−reduces to } fa : B[x := a]). \tag{2}$$

In fact, it is more elegant and uniform to use the second notation instead of the first one.

2. Compatibility. With Π-reduction, one introduces a *compatibility property* for the typing of applications:

$$M : N \Rightarrow MP : NP.$$

This is in line with the compatibility property for the typing of abstractions, which *does* hold in general:

$$M : N \Rightarrow \lambda_{y:P}M : \Pi_{y:P}N.$$

As an example, we give a simple derivation with the above-described *compatible* application rule and with conversion on Π-application:

$$\begin{array}{lll} A:*, b:A, a:A & \vdash & a:A \qquad\qquad\qquad\qquad \text{(start)} \\ A:*, b:A & \vdash & (\lambda_{a:A}.a):(\Pi_{a:A}.A) \qquad \text{(abstraction)} \\ A:*, b:A & \vdash & (\lambda_{a:A}.a)b:(\Pi_{a:A}.A)b \quad \text{(application)} \\ A:*, b:A & \vdash & (\lambda_{a:A}.a)b:A \qquad\qquad \text{(conversion)} \end{array}$$

3. Unified treatment of terms and types. It is our belief that with Π-reduction it is simpler to treat terms and types in a unified manner. Such a treatment provides a step towards the generalisation of type systems which is an important topic of research at the present time. For example, Barendregt's taxonomy of type systems in [Barendregt 92], but also Pure Type Systems (PTSs) introduced by Terlouw and Berardi (see [Ter 89]), and our generalised system in [NK 94] are attempts at combining all the important results of type systems in a compact and elegant way. As a step towards this goal, we believe that conversion should apply to both types and terms. In fact, Π is indeed a kind of λ, hence eligible for an application. This is a quite natural approach and one may interpret $(\Pi_{x:A}.B)a$ as the wish to select the "axis" $B(a)$ in the Cartesian product $\Pi_{x:A}.B$. One might argue that *implicit* Π-reduction (as is the case of the ordinary Cube) is closer to the intuition in the most usual applications. However, experiences with the Automath-languages ([de Bruijn 74]), containing *explicit* Π-reduction, demonstrated that there exists no formal or informal objection against the use of this explicit Π-reduction in natural applications of type systems.

4. The ability to divide two important questions of typing. Introducing explicit Π-reduction gives an elegant way to divide two important questions which are usually answered together via the judgement $? \vdash A : B$. These questions are:

1. Is A typable in $?$? (Below we use the simplified judgement $? \vdash A$ for this question.)
2. Is B the type of A in $?$? (Below we use a canonical type $\tau(?, A)$ for A and compare this canonical type with B, for this question.)

Π-reduction is needed in order to split elegantly these two questions. In particular, we require for an applcation $\tau(?, Fa) \equiv \tau(?, F)a$ on the condition that $\tau(?, F) = \Pi_{x:A}.B$, hence we obtain $(\Pi_{x:A}.B)a$, a Π-redex.

There are reasons why separating the questions "what is the type of a term" (via τ) and "is the term typable" (via $\vdash$), is advantageous. Here are some:

1. The canonical type of A is easy to calculate. The canonical type of A, $\tau(?, A)$ is defined by just scanning through A, removing all so called main Π-items $\Pi_{x:B}$, replacing all main λ-items $\lambda_{x:B}$ by $\Pi_{x:B}$ and replacing the heart of A by its obvious type in A. For example: if $A \equiv \Pi_{z:*}.(\lambda_{y:*}.(\lambda_{x:*}.x^\circ)y)(\Pi_{w:*}.(\lambda_{x:*}.x)y)$, then $\Pi_{z:*}$ is the main Π-item of A, $\lambda_{y:*}$ and $\lambda_{x:*}$ are the main λ-items and x° is the heart of A. Hence, $\tau(?, A) \equiv (\Pi_{y:*}.(\Pi_{x:*}.*)y)(\Pi_{w:*}.(\lambda_{x:*}.x)y)$.

A consequence is that the mapping algorithm (in order to find a type for a term) is extremely simple. This contrasts with the mapping algorithm in the usual setting, which needs intermediate applications of the conversion rule. This is caused by the fact that Fa is only typable if F has an appropriate Π-type. If F has not (yet) a Π-type, then the conversion rule must be used to find one. Of course we will need a conversion rule in order to check whether A has type B in context $?$ (by establishing that $\tau(?, A) = B$). Note, however, that we use only typing for the calculation of the canonical type, and only conversion for the second part ("$\tau(?, A) = B$?"). This is clearly a separation of concerns.

2. $\tau(A)$ plays the role of a preference type for A. To define the type of a term, in the traditional Cube, one starts with the types of variables, and subsequently deduces other statements of the form $? \vdash A : B$, by regarding more complex terms and their types. Finally, a conversion rule expresses that the types of terms are given *modulo* conversion; i.e., if $A : B$ and $B =_\beta C$, then $A : C$. The typing relation is the smallest relation satisfying these rules.

In our opinion, the approach in the traditional frameworks is, in a sense, ambiguous. Note that with *each variable* x and pseudo-context $?$, there is associated a preference type, which is B for $x : B \in ?$. For *terms in general* no preference type has been given, but a whole collection of types, which are typeable by themselves and linked by means of β-reduction.

We define however, the canonical type of A, $\tau(A)$, which plays the role of a preference type. For example, the preference type of $A \equiv \lambda_{x:*}.(\lambda_{y:*}.y)x$ is $\tau(<>, A) \equiv \Pi_{x:*}.(\Pi_{y:*}.*)x$. This type indeed reduces with the relation $\twoheadrightarrow_{\beta\Pi}$ to $\Pi_{y:*}.*$, the type traditionally given to A.

3. The conversion rule is no longer needed as a separate rule in the definition of $\vdash$. In our approach, β-conversion finds its place in the application condition of the rules of $\vdash$, where it naturally belongs. The conversion rule of the cube is redundant in our system. It is accommodated in our application rule:

$$\frac{? \vdash A \qquad ? \vdash B}{? \vdash AB} \text{ if } \tau(?, A) =_{\beta\Pi} \Pi_{x:C}.D \text{ and } \tau(?, B) =_{\beta\Pi} C$$

It will be the case that $\tau(?, AB) \equiv \tau(?, A)B =_{\beta\Pi} (\Pi_{x:C}.D)B \to_{\beta\Pi} D[x := B]$ and so indeed $\tau(?, AB) =_{\beta\Pi} D[x := C]$.

4. Higher degrees If we use λ^1 for Π and λ^2 for λ then we can aim for a possible generalization. In fact, we can extend our system by incorporating more different λ's. For example, with an infinity of λ's, viz. λ^0, λ^1, λ^2, λ^3 ..., we replace $\tau(?, \lambda_{x:A}.B) \equiv \Pi_{x:A}.\tau(?.\lambda_{x:A}, B)$ and $\tau(?, \Pi_{x:A}.B) \equiv \tau(?.\lambda_{x:A}, B)$ by the following:

$$\tau(?, \lambda^{i+1}_{x:A}.B) \equiv \lambda^i_{x:A}.\tau(?.\lambda_{x:A}, B), \text{for } i = 0, 1, 2, \ldots \text{ where } \lambda^0_{x:A}.B \equiv B$$

There is no reason why one cannot use as many λ^i as possible in a type system. In fact, even though in the Cube there are only two, there are other systems with more. There may be circumstances in which one desires to have more "layers" of λ's. As an example we refer to [de Bruijn 74].

Following the above observations, we introduce and study three typing relations ($\vdash_\beta$, $\vdash_{\beta\Pi}$ and $\vdash$) and a canonical typing operator τ. $\vdash_\beta$ is the typing relation of [Barendregt 92] and $\vdash_{\beta\Pi}$ is what we propose as its extension with Π-conversion. $\vdash$ and τ are what we use to divide the two important questions of typing as mentioned above. We divide the paper as follows:

- In Section 2, we introduce the formal machinery needed for $\vdash_\beta$, $\vdash_{\beta\Pi}$, $\vdash$ and τ.
- In Section 3, we introduce the usual properties of the Cube for $\vdash_\beta$ and $\twoheadrightarrow_\beta$ which will be studied for our extensions.
- In Section 4, we study in detail the properties of the Barendregt Cube extended with Π-conversion and show that $\vdash_{\beta\Pi}$ satisfies all the essential properties of $\vdash_\beta$ except for Subject Reduction. That is: $? \vdash_{\beta\Pi} A : B \wedge A \twoheadrightarrow_{\beta\Pi} A' \not\Rightarrow ? \vdash_{\beta\Pi} A' : B$. Subject Reduction however holds for the case $B \equiv \Box$ or $? \vdash_{\beta\Pi} B : S$. This Weak Subject Reduction is sufficient to obtain the desirable typing properties such as unicity of typing.

The explanation for this is that, this B which is not $\Box$ or of type S, reduces via $\twoheadrightarrow_{\beta\Pi}$ to B' which is itself either $\Box$ or of type S, and hence $? \vdash_{\beta\Pi} A : B$ implies $? \vdash_{\beta} B'$ where $B \twoheadrightarrow_{\beta\Pi} B'$ and B' has no Π-redexes.

- In Sections 5 and 6 we study the properties of the two separate typing questions regarding τ and $\vdash$.

2 The formal machinery of the Cube

The systems of the Cube (see [Barendregt 92]), are based on a set of *pseudo-expressions* or *terms* $\mathcal{T}$ defined by the following abstract syntax (let π range over both Π and λ):

$$\mathcal{T} = * \mid \Box \mid V \mid \mathcal{T}\mathcal{T} \mid \pi_{V:\mathcal{T}}.\mathcal{T}$$

where V is an infinite collection of variables over which $x, y, z, \ldots$ range. $*$ and $\Box$ are called sorts over which $S, S_1, S_2, \ldots$ are used to range. We take $A, B, C, a, b \ldots$ to range over $\mathcal{T}$.

Bound and free variables and substitution are defined as usual. We write $BV(A)$ and $FV(A)$ to represent the bound and free variables of A respectively. We write $A[x := B]$ to denote the term where all the free occurrences of x in A have been replaced by B. Furthermore, we take terms to be equivalent up to variable renaming. For example, we take $\lambda_{x:A}.x \equiv \lambda_{y:A}.y$ where $\equiv$ is used to denote syntactical equality of terms. We assume moreover, the Barendregt variable convention which is formally stated as follows:

Convention 2.1 *(BC: Barendregt's Convention)*
Names of bound variables will always be chosen such that they differ from the free ones in a term. Moreover, different λ's have different variables as subscript. Hence, we will not have $(\lambda_{x:A}.x)x$, but $(\lambda_{y:A}.y)x$ instead.

Terms can be related via a reduction relation. An example is β-reduction (see Section 3). We say that a reduction relation $\rightarrow$ on terms is compatible iff the following holds:

$$\frac{A_1 \rightarrow A_2}{A_1 B \rightarrow A_2 B} \qquad \frac{B_1 \rightarrow B_2}{A B_1 \rightarrow A B_2}$$

$$\frac{A_1 \rightarrow A_2}{\pi_{x:A_1}.B \rightarrow \pi_{x:A_2}.B} \qquad \frac{B_1 \rightarrow B_2}{\pi_{x:A}.B_1 \rightarrow \pi_{x:A}.B_2}$$

A *statement* is of the form $A : B$ with $A, B \in \mathcal{T}$. A is the *subject* and B is the *predicate* of $A : B$. A *declaration* is of the form $\lambda_{x:A}$ with $A \in \mathcal{T}$ and $x \in V$. A *pseudo-context* is a finite ordered sequence of declarations, all with distinct subjects. The *empty* context is denoted by $<>$. If $? = \lambda_{x_1:A_1}.\ldots.\lambda_{x_n:A_n}$ then $?.\lambda_{x:B} = \lambda_{x_1:A_1}.\ldots.\lambda_{x_n:A_n}.\lambda_{x:B}$ and $dom(?) = \{x_1, \ldots, x_n\}$. We use $?, \Delta, ?', ?_1, ?_2, \ldots$ to range over pseudo-contexts.

A *typability* relation $\vdash$ is a relation between pseudo-contexts and pseudo-expressions written as $? \vdash A$. The *rules of typability* establish which *judgements* $? \vdash A$ can be derived. A judgement $? \vdash A$ states that A is typable in the pseudo-context $?$.

A *type assignment* relation is a relation between a pseudo-context and two pseudo-expressions written as $? \vdash A : B$. The *rules of type assignment* establish which *judgements* $? \vdash A : B$ can be derived. A judgement $? \vdash A : B$ states that $A : B$ can be derived from the pseudo-context $?$.

When $? \vdash A$ or $? \vdash A : B$ then A and B are called (legal) *expressions* and $?$ is a (legal) *context*.

We write $? \vdash A : B : C$ for $? \vdash A : B \wedge ? \vdash B : C$. If $\Delta \equiv \lambda_{x_1:A_1}.\ldots.\lambda_{x_n:A_n}$ with $n \geq 0$ is a pseudo-context, then $? \vdash \Delta$, for $?$ a type assignment, means $? \vdash x_i : A_i$ for $1 \leq i \leq n$. If $A \to B$ then we also say $?_1.\lambda_{x:A}.?_2 \to ?_1.\lambda_{x:B}.?_2$ and define $\twoheadrightarrow$ on pseudo-contexts to be the reflexive transitive closure of $\to$.

Remark 2.2 Note that we differ from [Barendregt 92] in that we take a declaration to be $\lambda_{x:A}$ rather than $x : A$. The reason for this is that we want pseudo-contexts to be as close as possible to terms. In fact the context $?$ can be mapped to the term $?.*$ for example, and definitions of boundness/freeness of variables in a term and the Barendregt convention are thus easily extended to pseudo-contexts.

Definition 2.3 *(Type of Bound Variables, $\heartsuit$)*

- *If x occurs free in B, then all its occurrences are bound with type A in $\pi_{x:A}.B$.*
- *If an occurrence of x is bound with type A in B, then it is also bound with type A in $\pi_{y:C}.B$ for $y \not\equiv x$, in BC, and in CB.*
- *Define $\heartsuit(x) = x$, $\heartsuit(\pi_{x:A}.B) = \heartsuit(B)$ and $\heartsuit(AB) = \heartsuit(A)$.*

In this paper (Section 6) we introduce a system where the type information B of a judgement $? \vdash A : B$ is no longer needed. Hence, judgements obtain the form $? \vdash A$ (a simple judgement). In the following definition, we include these simple judgements.

Definition 2.4 *Let $?$ be a pseudo-context, A be a pseudo-expression and $\vdash$ be a typability or a type assignment relation.*

1. *$?$ is called legal if $\exists P, Q \in \mathcal{T}$ such that $? \vdash P(: Q)$.*
2. *$A \in \mathcal{T}$ is called a $?$-`term` if $? \vdash A(\exists B \in \mathcal{T}[? \vdash A : B \vee ? \vdash B : A])$.*
   We take $?$-terms $= \{A \in \mathcal{T} \mid ? \vdash A\ (\exists B \in \mathcal{T}[? \vdash A : B \vee ? \vdash B : A])\}$.
3. *$A \in \mathcal{T}$ is called legal if $\exists ?\,[A \in ?\text{-terms}]$.*
4. *We say that A is strongly normalising with respect to a reduction relation $\twoheadrightarrow$ (written $SN_{\twoheadrightarrow}(A)$) iff every $\twoheadrightarrow$-reduction path starting at A terminates.*

Definition 2.5 *Let $? \equiv \lambda_{x_1:A_1}.\ldots.\lambda_{x_n:A_n}$ and $\Delta \equiv \lambda_{y_1:B_1}.\ldots.\lambda_{y_m:B_m}$ be pseudo-contexts.*

1. *We write $\lambda_{x:A} \in ?$ if $x \equiv x_i$ and $A \equiv A_i$ for some i.*
2. *$?$ is part of Δ, notation $? \subseteq \Delta$, if every $\lambda_{x:A}$ in $?$ is also in Δ.*
3. *Let X be a set of variables. Then $? \restriction X$ is $?$ where $\lambda_{x_i:A_i}$ is removed for every $x_i \notin X$.*

3 The ordinary typing relation $\vdash_\beta$ and its properties

Definition 3.1 *(β-reduction $\rightarrow_\beta$ for the Cube)*
β-reduction $\rightarrow_\beta$, is the least compatible relation generated out of the following axiom:

$$(\beta) \qquad (\lambda_{x:B}.A)C \rightarrow_\beta A[x := C]$$

We take $\twoheadrightarrow_\beta$ to be the reflexive transitive closure of $\rightarrow_\beta$ and we take $=_\beta$ to be the least equivalence relation generated by $\twoheadrightarrow_\beta$.

Definition 3.2 *($\vdash_\beta$) The type assignement relation $\vdash_\beta$ is defined by the following inference rules:*

(axiom) $\qquad <> \vdash_\beta * : \Box$

(start rule) $\qquad \dfrac{? \vdash_\beta A : S}{?.\lambda_{x:A} \vdash_\beta x : A}\ x \notin ?$

(weakening rule) $\qquad \dfrac{? \vdash_\beta A : S \qquad ? \vdash_\beta D : E}{?.\lambda_{x:A} \vdash_\beta D : E}\ x \notin ?$

(application rule) $\qquad \dfrac{? \vdash_\beta F : \Pi_{x:A}.B \qquad ? \vdash_\beta a : A}{? \vdash_\beta Fa : B[x := a]}$

(abstraction rule) $\qquad \dfrac{?.\lambda_{x:A} \vdash_\beta b : B \qquad ? \vdash_\beta \Pi_{x:A}.B : S}{? \vdash_\beta \lambda_{x:A}.b : \Pi_{x:A}.B}$

(conversion rule) $\qquad \dfrac{? \vdash_\beta A : B \qquad ? \vdash_\beta B' : S \qquad B =_\beta B'}{? \vdash_\beta A : B'}$

(formation rule) $\qquad \dfrac{? \vdash_\beta A : S_1 \qquad ?.\lambda_{x:A} \vdash_\beta B : S_2}{? \vdash_\beta \Pi_{x:A}.B : S_2}$ *if* (S_1, S_2) *is a rule*

Each of the eight systems of the Cube is obtained by taking its set of (S_1, S_2) rules allowed in the formation rule out of $\{(*,*), (*,\Box), (\Box,*), (\Box,\Box)\}$. The basic system is the one where $(S_1, S_2) = (*,*)$ is the only possible choice. All other systems have this version of the formation rules, plus one or more other combinations of $(*,\Box)$, $(\Box,*)$ and $(\Box,\Box)$ for (S_1, S_2). Here is the table which presents the eight systems of the Cube (see also Figure 1):

System	*Allowed* (S_1, S_2) *rules*			
$\lambda_\rightarrow$	$(*,*)$			
$\lambda 2$	$(*,*)$	$(\Box,*)$		
λP	$(*,*)$		$(*,\Box)$	
$\lambda P2$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$
$\lambda P\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
$\lambda P\omega = \lambda C$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

Now, we list the properties of the Cube without proofs (see [Barendregt 92]). These properties will be studied in Section 4 for the Cube extended with Π-conversion and will be discussed for the two different subjects of canonical typing and typability in Sections 5 and 6 respectively.

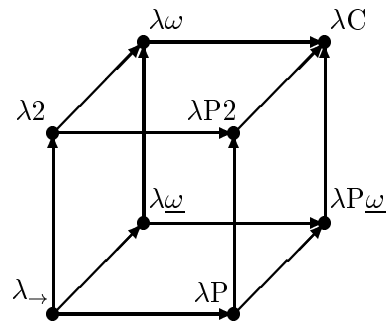

Figure 1: The Cube

Theorem 3.3 *(The Church Rosser Theorem CR, for $\twoheadrightarrow_\beta$)*
If $A \twoheadrightarrow_\beta B$ and $A \twoheadrightarrow_\beta C$ then there exists D such that $B \twoheadrightarrow_\beta D$ and $C \twoheadrightarrow_\beta D$ □

Lemma 3.4 *(Free variable lemma for $\vdash_\beta$)*
Let $? \equiv \lambda_{x_1:A_1}.\ldots.\lambda_{x_n:A_n}$ be a $\vdash_\beta$-legal context such that $? \vdash_\beta B : C$. Then we have:

1. *The $x_1 \ldots x_n$ are all distinct.*
2. *$FV(B), FV(C) \subseteq \{x_1, \ldots, x_n\}$.*
3. *$FV(A_i) \subseteq \{x_1, \ldots x_{i-1}\}$ for $1 \leq i \leq n$.* □

Lemma 3.5 *(Start Lemma for $\vdash_\beta$)*
Let $?$ be a $\vdash_\beta$-legal context. Then $? \vdash_\beta * : \Box$ and $\forall \lambda_{x:C} \in ?[? \vdash_\beta x : C]$. □

Lemma 3.6 *(Transitivity Lemma for $\vdash_\beta$)*
Let $?$ and Δ be $\vdash_\beta$-legal contexts. Then: $[? \vdash_\beta \Delta \wedge \Delta \vdash_\beta A : B] \Rightarrow ? \vdash_\beta A : B$. □

Lemma 3.7 *(Substitution Lemma for $\vdash_\beta$)*
Assume $?.\lambda_{x:A}.\Delta \vdash_\beta B : C$ and $? \vdash_\beta D : A$ then $?.(\Delta[x := D]) \vdash_\beta B[x := D] : C[x := D]$. □

Lemma 3.8 *(Thinning Lemma for $\vdash_\beta$)*
Let $?$ and Δ be $\vdash_\beta$-legal contexts such that $? \subseteq \Delta$. Then $? \vdash_\beta A : B \Rightarrow \Delta \vdash_\beta A : B$ □

Lemma 3.9 *(Generation Lemma for $\vdash_\beta$)*

1. *$? \vdash_\beta S : C \Rightarrow S \equiv *, C =_\beta \Box$, and if $C \not\equiv \Box$ then $? \vdash_\beta C : S'$ for some sort S'.*
2. *$? \vdash_\beta x : C \Rightarrow \exists B =_\beta C[\lambda_{x:B} \in ? \wedge$ if $C \not\equiv B$ then $? \vdash_\beta C : S$ for some sort S].*
3. *$? \vdash_\beta \Pi_{x:A}.B : C \Rightarrow \exists (S_1, S_2)[? \vdash_\beta A : S_1 \wedge ?.\lambda_{x:A} \vdash_\beta B : S_2 \wedge (S_1, S_2)$ is a rule $\wedge\, C =_\beta S_2 \wedge [C \not\equiv S_2 \Rightarrow \exists S[? \vdash_\beta C : S]]]$*
4. *$? \vdash_\beta \lambda_{x:A}.b : C \Rightarrow \exists (S, B)[? \vdash_\beta \Pi_{x:A}.B : S \wedge ?.\lambda_{x:A} \vdash_\beta b : B \wedge C =_\beta \Pi_{x:A}.B \wedge [C \not\equiv \Pi_{x:A}.B \Rightarrow \exists S[? \vdash_\beta C : S]]]$.*

5. $? \vdash_\beta Fa : C \Rightarrow \exists A, B, x[? \vdash_\beta F : \Pi_{x:A}.B \wedge ? \vdash_\beta a : A \wedge C =_\beta B[x := a] \wedge (B[x := a] \not\equiv C \Rightarrow \exists S[? \vdash_\beta C : S])]$. □

Corollary 3.10 *(Correctness of types for $\vdash_\beta$)*
If $? \vdash_\beta A : B$ then ($B \equiv \Box$ or $? \vdash_\beta B : S$ for some sort S). □

Lemma 3.11 *(Legal terms and contexts for $\vdash_\beta$ and $\twoheadrightarrow_\beta$)*
$\vdash_\beta$-legal terms and contexts contain no Π-redexes. □

Theorem 3.12 *(Subject Reduction SR, for $\vdash_\beta$ and $\twoheadrightarrow_\beta$)*
$? \vdash_\beta A : B \wedge A \twoheadrightarrow_\beta A' \Rightarrow ? \vdash_\beta A' : B$ □

Corollary 3.13 *(SR Corollary for $\vdash_\beta$ and $\twoheadrightarrow_\beta$)*

1. *If $? \vdash_\beta A : B$ and $B \twoheadrightarrow_\beta B'$ then $? \vdash_\beta A : B'$.*
2. *If A is a $?^{\vdash_\beta}$-term and $A \twoheadrightarrow_\beta A'$ then A' is a $?^{\vdash_\beta}$-term.* □

Lemma 3.14 *(Unicity of Types for $\vdash_\beta$ and $\twoheadrightarrow_\beta$)*

1. $? \vdash_\beta A : B_1 \wedge ? \vdash_\beta A : B_2 \Rightarrow B_1 =_\beta B_2$
2. $? \vdash_\beta A : B \wedge ? \vdash_\beta A' : B' \wedge A =_\beta A' \Rightarrow B =_\beta B'$
3. $? \vdash_\beta B : S, B =_\beta B', ? \vdash_\beta A' : B'$ *then* $? \vdash_\beta B' : S$. □

Theorem 3.15 *(Strong Normalisation with respect to $\vdash_\beta$ and $\twoheadrightarrow_\beta$)*
For all $\vdash_\beta$-legal terms M, $SN_{\twoheadrightarrow_\beta}(M)$; i.e. M is strongly normalising with respect to $\twoheadrightarrow_\beta$. □

4 The extended typing relation $\vdash_{\beta\Pi}$ and its properties

Definition 4.1 *($\beta\Pi$-reduction $\rightarrow_{\beta\Pi}$ for the Cube)*
$\beta\Pi$-reduction $\rightarrow_{\beta\Pi}$, is the least compatible relation generated out of the following axiom:

$$(\beta\Pi) \qquad (\pi_{x:B}.A)C \rightarrow_{\beta\Pi} A[x := C]$$

We take $\twoheadrightarrow_{\beta\Pi}$ to be the reflexive transitive closure of $\rightarrow_{\beta\Pi}$ and we take $=_{\beta\Pi}$ to be the least equivalence relation generated by $\twoheadrightarrow_{\beta\Pi}$.

Definition 4.2 *($\vdash_{\beta\Pi}$) We define $\vdash_{\beta\Pi}$ as $\vdash_\beta$ of Section 3 with the difference that the application and conversion rules change as follows:*

$$\textit{(new application rule)} \qquad \frac{? \vdash_{\beta\Pi} F : \Pi_{x:A}.B \qquad ? \vdash_{\beta\Pi} a : A}{? \vdash_{\beta\Pi} Fa : (\Pi_{x:A}.B)a}$$

$$\textit{(new conversion rule)} \qquad \frac{? \vdash_{\beta\Pi} A : B \qquad ? \vdash_{\beta\Pi} B' : S \qquad B =_{\beta\Pi} B'}{? \vdash_{\beta\Pi} A : B'}$$

The following lemmas hold for $\vdash_{\beta\Pi}$ and $\twoheadrightarrow_{\beta\Pi}$ and have the same formulation (only change β to $\beta\Pi$ everywhere) and proofs as for the case of $\vdash_\beta$ and $\twoheadrightarrow_\beta$:

- The Church Rosser Theorem for $\twoheadrightarrow_{\beta\Pi}$

- Free variable lemma for $\vdash_{\beta\Pi}$
- Start lemma for $\vdash_{\beta\Pi}$
- Transitivity lemma for $\vdash_{\beta\Pi}$
- Thinning lemma for $\vdash_{\beta\Pi}$
- Substitution lemma for $\vdash_{\beta\Pi}$
- Generation lemma for $\vdash_{\beta\Pi}$ where in clause 5, we replace $B[x := a]$ by $(\Pi_{x:A}.B)a$

Remark 4.3 *(Correctness of types does not hold for $\vdash_{\beta\Pi}$)*
The new legal terms of the form $(\Pi_{x:B}.C)A$ imply the failure of Corollary 3.10 for $\vdash_{\beta\Pi}$. That is, even in $\lambda_{\rightarrow}$, $? \vdash_{\beta\Pi} A : B \not\Rightarrow (B \equiv \Box$ or $? \vdash_{\beta\Pi} B : S$ for some sort $S)$. For example, if $? \equiv \lambda_{z:*}.\lambda_{x:z}$ then $? \vdash_{\beta\Pi} (\lambda_{y:z}.y)x : (\Pi_{y:z}.z)x$, but $? \not\vdash_{\beta\Pi} (\Pi_{y:z}.z)x : S$ from Lemma 4.5.

Failure of correctness of types implies failure of Subject Reduction even in $\lambda_{\rightarrow}$:

Example 4.4 In $\lambda_{\rightarrow}$, $\lambda_{z:*}.\lambda_{x:z} \not\vdash_{\beta\Pi} x : (\Pi_{y:z}.z)x$. Otherwise, by generation: $\lambda_{z:*}.\lambda_{x:z} \vdash_{\beta\Pi} (\Pi_{y:z}.z)x : S$, which is absurd by Lemma 4.5. Yet in $\lambda_{\rightarrow}$, $\lambda_{z:*}.\lambda_{x:z} \vdash_{\beta\Pi} (\lambda_{y:z}.y)x : (\Pi_{y:z}.z)x$.

We do have however, a weak subject reduction which we will prove after we show the relationship between $\vdash_{\beta\Pi}$ and $\vdash_{\beta}$.

Lemma 4.5 *For any A, B, C, S, $? \not\vdash_{\beta\Pi} (\Pi_{x:A}.B)C : S$.*
Proof: *If $? \vdash_{\beta\Pi} (\Pi_{x:A}.B)C : S$ then by generation, $? \vdash_{\beta\Pi} \Pi_{x:A}.B : \Pi_{x:A'}.B'$ and again by generation, $?.\lambda_{x:A} \vdash_{\beta\Pi} B : S' \wedge S' =_{\beta\Pi} \Pi_{x:A'}.B'$ which is absurd.* $\Box$

We do have the following lemma which is a sort of weak generation corollary:

Lemma 4.6 *$? \vdash_{\beta\Pi} A : B \wedge B$ is not a Π-redex $\Rightarrow (B \equiv \Box$ or $? \vdash_{\beta\Pi} B : S$ for some sort $S)$.*
Proof: *By a trivial induction on the derivation of $? \vdash_{\beta\Pi} A : B$ noting that the application rule does not apply as $(\Pi_{x:A}.B)a$ is not a Π-redex.* $\Box$

Lemma 4.7 *(Legal terms and contexts for $\vdash_{\beta\Pi}$ and $\twoheadrightarrow_{\beta\Pi}$)*

1. *If $? \vdash_{\beta\Pi} A : B$ then A and $?$ are free of Π-redexes, and either B contains no Π-redexes or B is the only Π-redex in B.*
2. *If $A \equiv (\pi_{x:D}.E)B$ is $\vdash_{\beta\Pi}$-legal, then $E[x := B]$ contains no Π-redexes.*

Proof: *1. is by induction on the derivation of $? \vdash_{\beta\Pi} A : B$. 2. By 1, we only need to show that if $B \equiv \Pi_{y:G}.H$, then E does not contain a subterm xF. Now, suppose $B \equiv \Pi_{y:G}.H$ and $E \equiv C[xF]$, then it is easy to see that $D \equiv \Pi_{z:I}.J$ for some I, J, and $? \vdash_{\beta\Pi} B : D$ for some context $?$. But $? \vdash_{\beta\Pi} \Pi_{y:G}.H : \Pi_{z:I}.J$ is impossible.* $\Box$

To relate $\vdash_{\beta}$ and $\vdash_{\beta\Pi}$, we introduce a notation which removes the unique Π-redex in a $\vdash_{\beta\Pi}$-legal term (if it exists):

Definition 4.8 *For A $\vdash_{\beta\Pi}$-legal, let $\hat{A}$ be $C[x := D]$ if $A \equiv (\Pi_{x:B}.C)D$ and A otherwise.*

Lemma 4.9

1. *If* $? \vdash_{\beta\Pi} A : B$ *then* $? \vdash_{\beta} A : \hat{B}$.

2. *If* $? \vdash_{\beta} A : B$ *then* $? \vdash_{\beta\Pi} A : B$.

Proof: *1. By a trivial induction on the derivations* $? \vdash_{\beta\Pi} A : B$. *2. By induction on the derivation* $? \vdash_{\beta} A : B$. *The only interesting cases come from conversion and application. The conversion case is easy as if* $B =_{\beta} B'$ *then* $B =_{\beta\Pi} B'$. *The application case is shown as follows: If* $? \vdash_{\beta} Fa : B[x := a]$ *comes from* $? \vdash_{\beta} F : \Pi_{x:A}.B$ *and* $? \vdash_{\beta} a : A$, *then by IH,* $? \vdash_{\beta\Pi} F : \Pi_{x:A}.B$ *and* $? \vdash_{\beta\Pi} a : A$. *Hence, by application,* $? \vdash_{\beta\Pi} Fa : (\Pi_{x:A}.B)a$. *But* $(\Pi_{x:A}.B)a =_{\beta\Pi} B[x := a]$. *If* $? \vdash_{\beta\Pi} B[x := a] : S$ *for some* S, *then by conversion* $? \vdash_{\beta\Pi} Fa : B[x := a]$. *But* $? \vdash_{\beta\Pi} B[x := a] : S$ *is shown as follows:*
$? \vdash_{\beta\Pi} F : \Pi_{x:A}.B$ *then* $\Pi_{x:A}.B$ *is* $\vdash_{\beta}$*-legal and* $? \vdash_{\beta\Pi} \Pi_{x:A}.B : S'$ *for some* S' *by Lemma 4.6. Now, by the generation lemma* $?.\lambda_{x:A} \vdash_{\beta\Pi} B : S$ *for some* S. *But* $? \vdash_{\beta\Pi} a : A$. *Hence by the substitution lemma:* $? \vdash_{\beta\Pi} B[x := a] : S$. □

Remark 4.10 Note that we may have $? \vdash_{\beta} A : \hat{B}$ without having $? \vdash_{\beta\Pi} A : B$, even if B is $\vdash_{\beta\Pi}$-legal. Take for example $? \equiv \lambda_{x:*}.\lambda_{y:*}$, $A \equiv x$ and $B \equiv (\Pi_{y:*}.*)x$. We have $? \vdash_{\beta\Pi} (\lambda_{y:*}.y)x : B$ hence B is $\vdash_{\beta\Pi}$-legal. We also have $? \vdash_{\beta} x : \hat{B}$. Yet $? \not\vdash_{\beta\Pi} x : B$.

Lemma 4.11 *If* $? \vdash_{\beta\Pi} A : B$ *and* $A \twoheadrightarrow_{\beta\Pi} A'$ *then* A' *has no* Π*-redexes.*

Proof: *We only show this for* $A \to_{\beta\Pi} A'$. *Note that* A *has no* Π*-redexes and so* $A \to_{\beta} A'$. *Now, from* $? \vdash_{\beta\Pi} A : B$ *we get by Lemma 4.9, 1,* $? \vdash_{\beta} A : \hat{B}$ *and so by Subject reduction for* $\twoheadrightarrow_{\beta}$ *we get* $? \vdash_{\beta} A' : \hat{B}$. *Hence* A' *has no* Π*-redexes by Lemma 4.7.* □

Lemma 4.12 *(Weak Subject Reduction for* $\vdash_{\beta\Pi}$ *and* $\twoheadrightarrow_{\beta\Pi}$*)*

1. $? \vdash_{\beta\Pi} A : B \wedge A \twoheadrightarrow_{\beta\Pi} A' \Rightarrow ? \vdash_{\beta\Pi} A' : \hat{B}$

2. $? \vdash_{\beta\Pi} A : B \wedge A \twoheadrightarrow_{\beta\Pi} A' \wedge B$ *is* $\vdash_{\beta}$*-legal* $\Rightarrow ? \vdash_{\beta\Pi} A' : B$

Proof: *1. From* $? \vdash_{\beta\Pi} A : B$, *and Lemma 4.9, 1,* $? \vdash_{\beta} A : \hat{B}$. *also, from* $A \twoheadrightarrow_{\beta\Pi} A'$, *and* A *and* A' *have no* Π*-redexes (Lemmas 4.7 and 4.11),* $A \twoheadrightarrow_{\beta} A'$. *Now, from SR for* $\twoheadrightarrow_{\beta}$ *we get* $? \vdash_{\beta} A' : \hat{B}$. *Hence, by Lemma 4.9, 2, we get* $? \vdash_{\beta\Pi} A' : \hat{B}$. *2. is a corollary of 1.* □

Corollary 4.13 *(WSR Corollary for* $\vdash_{\beta\Pi}$ *and* $\twoheadrightarrow_{\beta\Pi}$*)*

1. *If* $? \vdash_{\beta\Pi} A : B$ *and* $? \twoheadrightarrow_{\beta\Pi} ?'$ *then* $?' \vdash_{\beta\Pi} A : B$.

2. *If* $? \vdash_{\beta\Pi} A : B_1$ *and* $B_1 \twoheadrightarrow_{\beta\Pi} B_2$ *then* $? \vdash_{\beta\Pi} A : \hat{B}_2$.

3. *If* $? \vdash_{\beta\Pi} A : B$ *and* $B =_{\beta\Pi} S$ *then* $? \vdash_{\beta\Pi} A : S$.

4. *If* A *is* $?^{\vdash_{\beta\Pi}}$*-term and* $A \twoheadrightarrow_{\beta\Pi} A'$ *then* A' *is a* $?^{\vdash_{\beta\Pi}}$*-term.*

Proof: *1. By an easy induction on* $? \vdash_{\beta\Pi} A : B$ *using Lemma 4.12. 2. Use* $? \vdash_{\beta} A : \hat{B}$, $\hat{B} \to_{\beta} \hat{B}'$ *and SR for* $\to_{\beta}$. *3. is a corollary of 2. 4. Case* $? \vdash_{\beta\Pi} A : B$ *and* $A \twoheadrightarrow_{\beta\Pi} A'$ *then it is easy to show* A' *is* $\vdash_{\beta\Pi}$*-legal using Lemma 4.12. Here we show that if* $? \vdash_{\beta\Pi} B : A$ *and* $A \twoheadrightarrow_{\beta\Pi} A'$ *then* A' *is* $\vdash_{\beta\Pi}$*-legal. We will only consider the case where* $A \to_{\beta\Pi} A'$ *as the reflexivity and transitivity of* $\twoheadrightarrow_{\beta\Pi}$ *are easy. There are only three cases to consider:*

- *Case* $A \equiv \hat{A}$ *then* $A \to_{\beta} A'$ *and by Lemma 4.9, 1,* $? \vdash_{\beta} B : A$. *Hence,* $? \vdash_{\beta} B : A'$ *by SR for* $\twoheadrightarrow_{\beta}$ *and so* $? \vdash_{\beta\Pi} B : A'$ *by Lemma 4.9, 2.*

- *Case* $A \equiv (\Pi_{x:D}.E)A$, $A' \equiv E[x := C]$ *then by Lemma 4.9, 1,* $? \vdash_\beta B : \hat{A} \equiv A'$. *hence,* $? \vdash_{\beta\Pi} B : A'$ *by Lemma 4.9, 2.*

- *Case* $A \equiv (\Pi_{x:D}.E)C$, $A' \equiv (\Pi_{x:D'}.E')C'$, *then* C, D, E *are* $\vdash_\beta$*-legal,* $B \equiv FC$, $? \vdash_{\beta\Pi} F : \Pi_{x:D}.E$, $? \vdash_{\beta\Pi} C : D$ *and hence* $? \vdash_\beta F : \Pi_{x:D}.E$, $? \vdash_\beta C : D$. *So* $? \vdash_\beta F : \Pi_{x:D'}.E'$, $? \vdash_\beta C' : D'$. *Therefore,* $? \vdash_{\beta\Pi} F : \Pi_{x:D'}.E'$, $? \vdash_{\beta\Pi} C' : D'$ *and so* $? \vdash_{\beta\Pi} FC' : (\Pi_{x:D'}.E')C'$. □

Remark 4.14 We cannot replace 2 of Corollary 4.13 by: If $? \vdash_{\beta\Pi} A : B$ and $B \twoheadrightarrow_{\beta\Pi} B'$ then $? \vdash_{\beta\Pi} A : B'$. For example, take $? \equiv \lambda_{\alpha:*}.\lambda_{y:\alpha}$, $A \equiv (\lambda_{z:\alpha}.z)((\lambda_{x:\alpha}.x)y)$, $B \equiv (\Pi_{z:\alpha}.\alpha)((\lambda_{x:\alpha}.x)y)$ and $B' \equiv (\Pi_{z:\alpha}.\alpha)y$. Then, $? \vdash_{\beta\Pi} A : B$ but $? \not\vdash_{\beta\Pi} A : B'$ because if otherwise, we get by generation, $? \vdash_{\beta\Pi} (\Pi_{z:\alpha}.\alpha)y : S$, absurd by Lemma 4.5.

The result concerning WSR might look a bit disappointing. It is however discussed in detail in Section 7 which explains how the legal terms for $\vdash_{\beta\Pi}$ are not rich enough even though they are richer than the legal terms for $\vdash_\beta$. Furthermore, in Section 7, we also explain how WSR can be pushed back to full SR if the system is extended further.

Lemma 4.15 *(Unicity of Types for* $\vdash_{\beta\Pi}$ *and* $\twoheadrightarrow_{\beta\Pi}$*)*

1. $? \vdash_{\beta\Pi} A : B_1 \wedge ? \vdash_{\beta\Pi} A : B_2 \Rightarrow B_1 =_{\beta\Pi} B_2$

2. $? \vdash_{\beta\Pi} A : B \wedge ? \vdash_{\beta\Pi} A' : B' \wedge A =_{\beta\Pi} A' \Rightarrow B =_{\beta\Pi} B'$

3. $? \vdash_{\beta\Pi} B : S, B =_\beta B', ? \vdash_{\beta\Pi} A' : B'$ *then* $? \vdash_{\beta\Pi} B' : S$.

Proof: *1. by induction on the structure of A using the generation lemma. 2. by Church Rosser, Weak Subject Reduction, 1, and Lemma 4.7. 3. This is the same as* $? \vdash_\beta B : S, B =_\beta B', ? \vdash_\beta A' : B'$ *then* $? \vdash_\beta B' : S$ *which is 3 of lemma 3.14 and hence has the same proof. It is to be noted here that 3 fails for the case* $B =_{\beta\Pi} B'$. *Take for example* $? \vdash_{\beta\Pi} * : \Box, * =_{\beta\Pi} (\Pi_{\beta:*}.*)\alpha, \lambda_{\alpha:*} \vdash_{\beta\Pi} (\lambda_{\beta:*}.\beta)\alpha : (\Pi_{\beta:*}.*)\alpha, ? \not\vdash_{\beta\Pi} (\Pi_{\beta:*}.*)\alpha : \Box$ □

Lemma 4.16
If $SN_{\rightarrow_{\beta\Pi}}(B[x := C])$, $SN_{\rightarrow_{\beta\Pi}}(A)$, $SN_{\rightarrow_{\beta\Pi}}(B)$ *and* $SN_{\rightarrow_{\beta\Pi}}(C)$ *then* $SN_{\rightarrow_{\beta\Pi}}((\Pi_{x:A}.B)C)$.
Proof: *This is standard.* □

Theorem 4.17 *(Strong Normalisation with respect to* $\vdash_{\beta\Pi}$ *and* $\twoheadrightarrow_{\beta\Pi}$*)*
For all $\vdash_{\beta\Pi}$*-legal terms A,* $SN_{\rightarrow_{\beta\Pi}}(A)$*; i.e. A is strongly normalising with respect to* $\twoheadrightarrow_{\beta\Pi}$.
Proof: *Note that if A is* Π*-redex free and* $SN_{\rightarrow_\beta}(A)$ *then* $SN_{\rightarrow_{\beta\Pi}}(A)$. *We show that if* $? \vdash_{\beta\Pi} A : B$ *then* $SN_{\rightarrow_{\beta\Pi}}(A)$ *and* $SN_{\rightarrow_{\beta\Pi}}(B)$. *By Lemma 4.9, 1,* $? \vdash_\beta A : \hat{B}$. *Hence, by Theorem 3.15,* $SN_{\rightarrow_\beta}(A)$ *and* $SN_{\rightarrow_\beta}(\hat{B})$. *Hence,* $SN_{\rightarrow_{\beta\Pi}}(A)$ *and we only have to show that* $SN_{\rightarrow_{\beta\Pi}}(B)$.

- *Case* $B \equiv \hat{B}$ *then* $SN_{\rightarrow_{\beta\Pi}}(B)$.

- *Case* $B \equiv (\Pi_{x:B_1}.B_2)B_3$ *then* $\hat{B} \equiv B_2[x := B_3]$, B_1, B_2, B_3 *are* $\vdash_\beta$*-legal. By Lemma 4.16,* $SN_{\rightarrow_{\beta\Pi}}(B_i)$ *for* $1 \leq i \leq 3$ *and* $SN_{\rightarrow_{\beta\Pi}}(\hat{B})$, *we get* $SN_{\rightarrow_{\beta\Pi}}((\Pi_{x:B_1}.B_2)B_3)$. □

5 The canonical typing operator τ and its properties

Definition 5.1 *(Canonical Type Operator) For any pseudo-context ? and pseudo-expression A, we define the canonical type of A in ?, $\tau(?, A)$ as follows:*

$$\begin{array}{lcll}
\tau(?, *) & \equiv & \Box & \\
\tau(?, x) & \equiv & A \textit{ if } \lambda_{x:A} \in ? & \\
\tau(?, Fa) & \equiv & \tau(?, F)a & \\
\tau(?, \lambda_{x:A}.B) & \equiv & \Pi_{x:A}.\tau(?.\lambda_{x:A}, B) & \textit{if } x \notin dom(?) \\
\tau(?, \Pi_{x:A}.B) & \equiv & \tau(?.\lambda_{x:A}, B) & \textit{if } x \notin dom(?)
\end{array}$$

Example 5.2 In usual type theory, the type of $\lambda_{x:*}.\lambda_{y:x}.y$ is $\Pi_{x:*}.\Pi_{y:x}.x$ and the type of $\Pi_{x:*}.\Pi_{y:x}.x$ is $*$. Now, with our τ, we get the same result:
$\tau(<>, \lambda_{x:*}.\lambda_{y:x}.y) \equiv \Pi_{x:*}.\tau(\lambda_{x:*}, \lambda_{y:x}.y) \equiv \Pi_{x:*}.\Pi_{y:x}.\tau(\lambda_{x:*}.\lambda_{y:x}, y) \equiv \Pi_{x:*}.\Pi_{y:x}.x$
$\tau(<>, \Pi_{x:*}.\Pi_{y:x}.x) \equiv \tau(\lambda_{x:*}, \Pi_{y:x}.x) \equiv \tau(\lambda_{x:*}.\lambda_{y:x}, x) \equiv *$

Remark 5.3 Note that $\tau(?, \Box)$ is undefined. We write $\downarrow \tau(?, A)$ for $\tau(?, A)$ defined. Note also that $FV(\tau(?, A)) \neq FV(?.A)$. For example, if $? \equiv \lambda_{x:*}\lambda_{y:x}.\lambda_{z:p}$, then $\tau(?, y) \equiv x$, $x \in FV(\tau(?, y)) \setminus FV(?.y)$, and $p \in FV(?.y) \setminus FV(\tau(?, y))$.

In what follows, we study the properties of τ.

Lemma 5.4 *(τ-weakening)*
Let $?, ?'$ be pseudo-contexts. $? \subseteq ?' \wedge \downarrow \tau(?, A) \Rightarrow [\downarrow \tau(?', A)$ and $\tau(?, A) \equiv \tau(?', A)]$.
Proof: *By induction on A, noting that bound variables in A can always be renamed so that they don't occur in $dom(?')$.* $\Box$

Lemma 5.5 *(Context-reduction for τ)*
For $?, ?'$ be pseudo-contexts, $? \twoheadrightarrow_\beta ?' \wedge \downarrow \tau(?, A) \Rightarrow [\downarrow \tau(?', A) \wedge \tau(?, A) \twoheadrightarrow_\beta \tau(?', A)]$.
Proof: *By induction on $\tau(?, A)$.* $\Box$

Lemma 5.6 *(τ-restriction)*
If $\downarrow \tau(?, A)$ then $\tau(? \restriction FV(A), A) \equiv \tau(?, A)$.
Proof: *By induction on A.* $\Box$

Lemma 5.7 *(τ-Substitution Lemma) Let $\sim$ be $\twoheadrightarrow_{\beta\Pi}, =_{\beta\Pi}$ or $\equiv$.*
If $\tau(?.\lambda_{x:A}.\Delta, B) \equiv C$ and $\tau(?, D) \sim A$ then $\tau(?.(\Delta[x := D]), B[x := D]) \sim C[x := D]$.
Proof: *By induction on the structure of A.* $\Box$

Note that when $?, A$ contain no Π-redexes, $\tau(?, A)$ is exactly as A except that:

1. An occurrence of $\pi_{x:B}$ in A which is not an occurrence in some C where $\pi_{y:C}.D$ or DC is a subterm of A, disappears in the case $\pi = \Pi$ and becomes $\Pi_{x:B}$ in the case $\pi = \lambda$.

2. $\heartsuit(A)$ is replaced by $\tau(?', \heartsuit(A))$ where $?' = ?.\lambda_{x_1:A_1} \ldots .\lambda_{x_n:A_n}$ and $x_i : A_i$ are those of $\pi_{y:B}$ which have either disappeared or been replaced by $\Pi_{y:B}$, taken in the same order in which they appeared in A.

Example 5.8

$$\begin{array}{lllllll} \tau(<>, \Pi_{z:*}. & (\lambda_{y:*}. & (\lambda_{x:*}. & x &)y)(\Pi_{w:*}.(\lambda_{x:*}.x)y) & \equiv \\ & (\Pi_{y:*}. & (\Pi_{x:*}. & * &)y)(\Pi_{w:*}.(\lambda_{x:*}.x)y) & \end{array}$$

$$\begin{array}{lllllll} \tau(<>, & ((\lambda_{x:*}. & (\lambda_{y:*}. & \lambda_{z:*}. & z &)x)C)D & \equiv \\ & ((\Pi_{x:*}. & (\Pi_{y:*}. & \Pi_{z:*}. & * &)x)C)D & \end{array}$$

This can be made clearer by using the *item notation* via a translation function $\mathcal{I}$ where $\mathcal{I}(\pi_{x:A}.B) \equiv (\mathcal{I}(A)\pi_x)\mathcal{I}(B)$ and $\mathcal{I}(AB) \equiv (\mathcal{I}(B)\delta)\mathcal{I}(A)$. Note that for each A, $\mathcal{I}(A) \equiv I_1 I_2 \ldots I_n x$ where each *main item* I_i is of the form $(A_i\omega)$ for $\omega \in \{\delta\} \cup \{\pi_y; y \in V\}$ and $x \equiv \heartsuit(A)$. Moreover, any π-redex $(\pi_{y:B}.C)D$ in A will be $(\mathcal{I}(D)\delta)(\mathcal{I}(B)\pi_y)\mathcal{I}(C)$. Hence, π-redexes start by a δ-item just before a π-item.

With this item notation, it is clearer to evaluate τ. In fact, we go through $\mathcal{I}(A)$ from left to right and for every I_i we reach, we keep it unchanged if it is a δ-item, we remove it if it is a Π-item and we change the λ to Π if it is a λ-item. Finally, we replace $\heartsuit(A)$ which is x by $\tau(?', x)$ where $?' \equiv \mathcal{I}(?).I'_{i_1}.\ldots.I'_{i_k}$ and I'_{i_j} are all the π-items of A where Π is changed to λ. Of course $\mathcal{I}(\tau(?, A)) \equiv \tau(\mathcal{I}(?), \mathcal{I}(A))$. For example, for $A \equiv \Pi_{z:*}.(\lambda_{y:*}.(\lambda_{x:*}.x)y)(\Pi_{w:*}.(\lambda_{x:*}.x)y)$,

$$\begin{array}{llllllll} \mathcal{I}(A) & \equiv (*\Pi_z) & ((*\Pi_w)(y\delta)(*\lambda_x)x\delta) & (*\lambda_y) & (y\delta) & (*\lambda_x) & x & \\ \tau(<>, \mathcal{I}(A)) & \equiv & ((*\Pi_w)(y\delta)(*\lambda_x)x\delta) & (*\Pi_y) & (y\delta) & (*\Pi_x) & \tau((*\lambda_z)(*\lambda_y)(*\lambda_x), x) & \\ & \equiv & ((*\Pi_w)(y\delta)(*\lambda_x)x\delta) & (*\Pi_y) & (y\delta) & (*\Pi_x) & * & \end{array}$$

Note that I_1 has disappeared, I_2 and I_4 remained unchanged whereas the λ in I_3 and I_5 changed to Π. Note also that $\mathcal{I}(\tau(<>, A)) \equiv \tau(<>, \mathcal{I}(A))$. In item notation, every term is of the form $\overline{S}x$ or $\overline{S}*$ where $\overline{S}$ is a segment, i.e. a sequence of items. For a segment $\overline{S}$, we define $\overline{S}^{\lambda}$ as $\overline{S}$ where all the main π-items are written as λ-items and where all the main δ-items are removed. We define $\overline{S}^{\Pi}$ as $\overline{S}$ where all the main λ-items are replaced by Π-items, all the main δ-items remain unchanged and all the main Π-items are removed. For example, if $\overline{S} \equiv (x\delta)(y\lambda_z)(z\Pi_r)$ then $\overline{S}^{\lambda} \equiv (y\lambda_z)(z\lambda_r)$ and $\overline{S}^{\Pi} \equiv (x\delta)(y\Pi_z)$. With these notations, $\tau(?, \overline{S}x) \equiv \overline{S}^{\Pi}\tau(?\,\overline{S}^{\lambda}, x)$.

This item notation has been used to study, extend and clarify many notions of the λ-calculus (see [KN 93] and [KN 9y]).

Remark 5.9 Note that typability of subterms fails for τ. That is, τ can be defined for some A without being defined for all its subterms. For example, $\tau(<>, (\lambda_{x:*}.x)y) \equiv (\Pi_{x:*}.*)y$, but $\tau(<>, y)$ is not defined. Note also that unicity of types fails for τ. That is, we can have $A \twoheadrightarrow_{\beta\Pi} A'$ without having $\tau(?, A) =_{\beta\Pi} \tau(?, A')$. For example, $A \equiv (\lambda_{x:*}.x)(\lambda_{y:*}.y) \rightarrow_{\beta\Pi} \lambda_{y:*}.y \equiv A'$ yet $\tau(<>, A) \equiv (\Pi_{x:*}.*)(\lambda_{y:*}.y) \neq_{\beta\Pi} \tau(<>, \lambda_{y:*}.y) \equiv \Pi_{y:*}.*$. Moreover, $\mathrm{SN}_{\twoheadrightarrow_{\beta\Pi}}(A) \not\Rightarrow \mathrm{SN}_{\twoheadrightarrow_{\beta\Pi}}(\tau(?, A))$. For example, take $? \equiv \lambda_{x:(\Pi_{x:*}.xx)(\Pi_{x:*}.xx)}$ and $A \equiv x$. In Lemmas 6.7 and 6.17, we show that typability of subterms and unicity of types hold for τ when $? \vdash A$. We conjecture moreover, that if $? \vdash A$ then $\tau(?, A)$ is strongly normalising.

6 The typability relation $\vdash$ and its properties

Definition 6.1 *($\vdash$) The Typability relation $\vdash$ is defined by the following rules:*

($\vdash$-axiom) $\quad <> \vdash *$

($\vdash$-start rule) $\quad \dfrac{? \vdash A}{?.\lambda_{x:A} \vdash x}$ *if vc*

($\vdash$-weakening rule) $\quad \dfrac{? \vdash A \qquad ? \vdash D}{?.\lambda_{x:A} \vdash D}$ *if vc*

($\vdash$-application rule) $\quad \dfrac{? \vdash F \qquad ? \vdash a}{? \vdash Fa}$ *if ap*

($\vdash$-abstraction rule) $\quad \dfrac{?.\lambda_{x:A} \vdash b \qquad ? \vdash \Pi_{x:A}.B}{? \vdash \lambda_{x:A}.b}$ *if ab*

($\vdash$-formation) $\quad \dfrac{? \vdash A \qquad ?.\lambda_{x:A} \vdash B}{? \vdash \Pi_{x:A}.B}$ *if fc*

vc (variable condition): $x \notin ?$ and $\tau(?, A) \twoheadrightarrow_{\beta\Pi} S$ for some S
ap (application condition): $\tau(?, F) =_{\beta\Pi} \Pi_{x:A}.B$ and $\tau(?, a) =_{\beta\Pi} A$ for some A, B.
ab (abstraction condition): $\tau(?.\lambda_{x:A}, b) =_{\beta\Pi} B$ and $\tau(?, \Pi_{x:A}.B) \twoheadrightarrow_{\beta\Pi} S$ for some S.
fc (formation condition): $\tau(?, A) \twoheadrightarrow_{\beta\Pi} S_1$ and $\tau(?.\lambda_{x:A}, B) \twoheadrightarrow_{\beta\Pi} S_2$ for some (S_1, S_2) rule.

When $? \vdash A$, we say that A is typable in $?$.

Lemma 6.2 *(Free variable lemma and type-definability for $\vdash$ and τ)*
Let $? \equiv \lambda_{x_1:A_1}. \ldots .\lambda_{x_n:A_n}$. If $? \vdash A$. Then we have:

1. *The $x_1 \ldots x_n$ are all distinct.*
2. *$FV(A) \subseteq \{x_1, \ldots, x_n\}$.*
3. *$FV(A_i) \subseteq \{x_1, \ldots x_{i-1}\}$ for $1 \leq i \leq n$.*
4. *$\downarrow \tau(?, A)$ and $FV(\tau(?, A)) \subseteq \{x_1, \ldots, x_n\}$.*

Proof: *By induction on $? \vdash A$.* □

Lemma 6.3 *(Start Lemma for $\vdash$ and τ)*
If $?$ is $\vdash$-legal, then $? \vdash *$ and $\forall \lambda_{x:C} \in ?[? \vdash x \wedge \tau(?, x) \equiv C]$.
Proof: *By induction on the derivation $? \vdash A$.* □

Lemma 6.4 *(Substitution Lemma for $\vdash$ and τ)*
If $?.\lambda_{x:A}.\Delta \vdash B$ and $? \vdash D$ and $\tau(?, D) =_{\beta\Pi} A$, then $?.(\Delta[x := D]) \vdash B[x := D]$ and $\tau(?.(\Delta[x := D]), B[x := D]) =_{\beta\Pi} \tau(?.\lambda_{x:A}.\Delta, B)[x := D]$.
Proof: *By induction on the derivations of $?.\lambda_{x:A}.\Delta \vdash B$.* □

Lemma 6.5 *(Thinning Lemma for $\vdash$ and τ)*
If $?$ and Δ be $\vdash$-legal and $? \subseteq \Delta$, then $? \vdash A \Rightarrow \Delta \vdash A$ (note that $\tau(?, A) \equiv \tau(\Delta, A)$).
Proof: *By induction on the length of the derivations $? \vdash A$.* □

Lemma 6.6 *(Generation Lemma for $\vdash$ and τ)*

1. $? \vdash S \Rightarrow S \equiv *$.
2. $? \vdash x \Rightarrow \exists A[\lambda_{x:A} \in ? \wedge \tau(?, x) \equiv A]$.
3. $? \vdash \Pi_{x:A}.B \Rightarrow \exists S_1, S_2[? \vdash A \wedge ?.\lambda_{x:A} \vdash B \wedge \tau(?, A) =_{\beta\Pi} S_1 \wedge \tau(?.\lambda_{x:A}, B) =_{\beta\Pi} S_2 \wedge (S_1, S_2)$ *is a rule*$]$.
4. $? \vdash \lambda_{x:A}.b \Rightarrow \exists S, B[? \vdash \Pi_{x:A}.B \wedge ?.\lambda_{x:A} \vdash b \wedge \tau(?.\lambda_{x:A}, b) =_{\beta\Pi} B \wedge \tau(?, \Pi_{x:A}.B) =_{\beta\Pi} S]$.
5. $? \vdash Fa \Rightarrow \exists A, B, x[? \vdash F \wedge ? \vdash a \wedge \tau(?, F) =_{\beta\Pi} \Pi_{x:A}.B \wedge \tau(?, a) =_{\beta\Pi} A]$.

Proof: *By induction on the derivations $? \vdash A$.* □

Lemma 6.7 *(Typability of subterms)*
If $? \vdash A$ and A' is a subexpression of A then $(\exists ?')[?.?' \vdash A']$.
Proof: *By induction on $? \vdash A$.* □

Lemma 6.8 *(Legal terms and contexts for $\vdash$)*
$\vdash$-legal terms and contexts are free of Π-redexes,
Proof: *By induction on $? \vdash A$. The only interesting case is application. Assume $? \vdash F$, $? \vdash a$, $\tau(?, F) =_{\beta\Pi} \Pi_{x:A}.B$ and $\tau(?, a) =_{\beta\Pi} A$. By IH, $?, F, a$ are Π-redexes free. Also, $F \not\equiv \Pi_{x:C}.D$, otherwise, $\tau(?.\lambda_{x:C}, D) \equiv \tau(?, F) =_{\beta\Pi} S_2 =_{\beta\Pi} \Pi_{x:A}.B$, absurd.* □

Note that $? \vdash A \not\Rightarrow (\tau(?, A) \equiv \Box \vee ? \vdash \tau(?, A))$. For example, $\lambda_{x:*} \vdash (\lambda_{y:*}.y)x$ and $\lambda_{x:*} \nvdash (\Pi_{x:*}.*)x$, by Lemma 6.8. The property however holds when $\tau(?, A)$ is Π-redex free. We need first the following lemma:

Lemma 6.9
If $? \vdash A$, $? \vdash B$ and $A =_\beta B$ then $\tau(?, A) =_{\beta\Pi} \tau(?, B)$.
Proof: *By induction on $A =_\beta B$ using Lemmas 5.5 and 5.7.* □

Lemma 6.10 *If $? \vdash A$ and $\tau(?, A)$ is Π-redex free, then $\tau(?, A) \equiv \Box$ or $? \vdash \tau(?, A)$.*
Proof: *By induction on $? \vdash A$ using Lemma 6.9 (application cannot apply otherwise, $\tau(?, Fa) \equiv \tau(?, F)a =_{\beta\Pi} (\Pi_{x:A}.B)a \Rightarrow \tau(?, F) \equiv \Pi_{x:A'}.B'$ and $\tau(?, Fa)$ is a Π-redex).* □

Now, let us study the relationship between $\vdash_{\beta\Pi}$ and $\vdash$.

Lemma 6.11 *If $? \vdash_{\beta\Pi} A : B$ then $? \vdash A$ and $\tau(?, A) =_{\beta\Pi} B$.*
Proof: *By induction on the derivations $? \vdash_{\beta\Pi} A : B$.* □

Definition 6.12 *For A a pseudo-term, we take $\overline{A}$ to be the $\beta\Pi$-normal form of A.*

Lemma 6.13 *If* $? \vdash A$ *then* $\downarrow \overline{\tau(?, A)}$ *and* $? \vdash_\beta A : \overline{\tau(?, A)}$

Proof: *By induction on* $? \vdash A$. *We only treat three cases:*

application: *Assume* $? \vdash F$ *and* $? \vdash a$ *give* $? \vdash Fa$ *where the application condition (ap) holds and IH holds for the first two derivations.* $\tau(?, F) =_{\beta\Pi} \Pi_{x:A}.B \wedge \tau(?, a) =_{\beta\Pi} A \Rightarrow \exists C, D$ *where* $A \twoheadrightarrow_{\beta\Pi} C$, $B \twoheadrightarrow_{\beta\Pi} D$, $\overline{\tau(?, F)} \equiv \Pi_{x:C}.D$ *and* $\overline{\tau(?, a)} \equiv C$.
Moreover, by IH $? \vdash_\beta \overline{\tau(?, F)} : S$ *(otherwise by Corollary 3.10,* $\Pi_{x:C}.D \equiv \Box$ *absurd).*
Now, use application on $? \vdash_\beta a : C$, $? \vdash_\beta F : \Pi_{x:C}.D$ *to get* $? \vdash_\beta Fa : D[x := a]$.
Hence by Strong Normalisation of $\vdash_\beta$, $\downarrow \overline{D[x := a]}$.
But, $\overline{\tau(?, Fa)} \equiv \overline{\tau(?, F)a} \equiv \overline{(\Pi_{x:C}.D)a} \equiv \overline{D[x := a]}$ *and so* $\downarrow \overline{\tau(?, Fa)}$.
Now, by Corollary 3.10, $? \vdash_\beta Fa : D[x := a] \Rightarrow D[x := a] \equiv \Box \vee \exists S[? \vdash_\beta D[x := a] : S]$.

- *Case* $D[x := a] \equiv \Box$ *then* $\overline{\tau(?, Fa)} \equiv \overline{D[x := a]} \equiv D[x := a]$ *and* $? \vdash_\beta Fa : \overline{\tau(?, Fa)}$.
- *Case* $? \vdash_\beta D[x := a] : S$, *then by SR for* $\vdash_\beta$, *as* $D[x := a] \twoheadrightarrow_\beta \overline{D[x := a]}$,
  $? \vdash_\beta \overline{D[x := a]} : S$.
  Now, use $? \vdash_\beta Fa : D[x := a]$, $? \vdash_\beta \overline{D[x := a]} : S$ *and* $D[x := a] =_\beta \overline{D[x := a]}$ *and conversion for* $\vdash_\beta$ *to get* $? \vdash_\beta Fa : \overline{D[x := a]}$. *Hence,* $? \vdash_\beta Fa : \overline{\tau(?, Fa)}$.

abstraction: *assume* $? \vdash \Pi_{x:A}.B$ *and* $?.\lambda_{x:A} \vdash b$ *imply* $? \vdash \lambda_{x:A}.b$ *where* $\tau(?.\lambda_{x:A}, b) =_{\beta\Pi} B$, *and* $\tau(?, \Pi_{x:A}.B) \twoheadrightarrow_{\beta\Pi} S$. *Hence,* $\overline{\tau(?, \Pi_{x:A}.B)} \equiv S$.
By IH, $? \vdash_\beta \Pi_{x:A}.B : \overline{\tau(?, \Pi_{x:A}.B)} \equiv S$. *Moreover, by ab as* $\tau(?.\lambda_{x:A}, b) =_{\beta\Pi} B$, *we get* $B \twoheadrightarrow_{\beta\Pi} \overline{\tau(?.\lambda_{x:A}, b)}$. *Hence,* $\Pi_{x:A}.B \twoheadrightarrow_{\beta\Pi} \Pi_{x:A}.\overline{\tau(?.\lambda_{x:A}, b)}$ *and* $? \vdash_\beta \Pi_{x:A}.\overline{\tau(?.\lambda_{x:A}, b)} : S$ *by SR for* $\vdash_\beta$.
Furthermore, by IH, $?.\lambda_{x:A} \vdash_\beta b : \overline{\tau(?.\lambda_{x:A}, b)}$.
Now, use $?.\lambda_{x:A} \vdash_\beta b : \overline{\tau(?.\lambda_{x:A}, b)}$, $? \vdash_\beta \Pi_{x:A}.\overline{\tau(?.\lambda_{x:A}, b)} : S$ *and abstraction to get*
$? \vdash_\beta \lambda_{x:A}.b : \Pi_{x:A}.\overline{\tau(?.\lambda_{x:A}, b)}$.
But $\Pi_{x:A}.\overline{\tau(?.\lambda_{x:A}, b)} \twoheadrightarrow_\beta \Pi_{x:A}.\overline{\tau(?.\lambda_{x:A}, b)} \equiv \overline{\tau(?, \lambda_{x:A}, b)}$.
Hence by Corollary 3.13, $? \vdash_\beta \lambda_{x:A}.b : \overline{\tau(?, \lambda_{x:A}.b)}$.
formation: *Assume* $? \vdash A$ *and* $?.\lambda_{x:A} \vdash B$ *give* $? \vdash \Pi_{x:A}.B$ *and IH holds for the first two derivations. Hence,* $\downarrow \overline{\tau(?, A)}$, $\overline{\tau(?.\lambda_{x:A}, B)}$, $? \vdash_\beta A : \overline{\tau(?, A)}$ *and* $?.\lambda_{x:A} \vdash_\beta B : \overline{\tau(?.\lambda_{x:A}, B)}$.
Hence, as $\overline{\tau(?, \Pi_{x:A}.B)} \equiv \overline{\tau(?.\lambda_{x:A}, B)}$, *we get* $\downarrow \overline{\tau(?, \Pi_{x:A}.B)}$.
Furthermore, as by fc, $\tau(?, A) =_{\beta\Pi} S_1$ *and* $\tau(?.\lambda_{x:A}, B) =_{\beta\Pi} S_2$, *for some* (S_1, S_2) *rule, we get* $\overline{\tau(?, A)} \equiv S_1$ *and* $\overline{\tau(?.\lambda_{x:A}, B)} \equiv S_2$.
Now, we use formation to get $? \vdash_\beta \Pi_{x:A}.B : \overline{\tau(?, \Pi_{x:A}.B)}$. $\Box$

Lemma 6.14 *(Subject Reduction for* $\vdash$ *and* τ*)*
$? \vdash A \wedge A \twoheadrightarrow_{\beta\Pi} A' \Rightarrow [? \vdash A' \wedge \tau(?, A) =_{\beta\Pi} \tau(?, A')]$

Proof: *Use Lemmas 6.11, 6.13 and SR for* $\vdash_\beta$. $\Box$

Corollary 6.15 *(SR corollary for* $\vdash$ *and* τ*)*

1. *If* $? \vdash A$ *and* $? \twoheadrightarrow_\beta ?'$ *then* $?' \vdash A$ *and* $\tau(?, A) =_{\beta\Pi} \tau(?'A)$.
2. *If* A *is* $?^\vdash$*-term and* $A \twoheadrightarrow_\beta A'$ *then* A' *is a* $?^\vdash$*-term.*

Proof: *1.* $? \vdash_\beta A : \overline{\tau(?, A)} \Rightarrow ?' \vdash_\beta A : \overline{\tau(?, A)}$. *Hence, by Lemma 6.11* $?' \vdash A$ *and* $\tau(?', A) =_{\beta\Pi} \overline{\tau(?, A)} =_{\beta\Pi} \tau(?, A)$. $\Box$

Remark 6.16 Note that $? \vdash A$ and $A \twoheadrightarrow_\beta A' \not\Rightarrow \tau(?, A) \twoheadrightarrow_\beta \tau(?, A')$. For example, If $A \equiv (\lambda_{z:w}.z)y$ and $? \equiv \lambda_{w:*}.\lambda_{y:(\lambda_{x:*}.x)w}$, then $A \twoheadrightarrow_\beta y$, $\tau(?, A) \equiv (\Pi_{z:w}.w)y \not\twoheadrightarrow_\beta \tau(?, y)$.

Lemma 6.17 *(Unicity of Types for $\vdash$ and τ)*

1. $? \vdash A \wedge ? \vdash B \wedge A =_\beta B \Rightarrow \tau(?, A) =_{\beta\Pi} \tau(?, B)$

   Proof: *Use CR and SR to show $? \vdash C$, $\tau(?, A) =_{\beta\Pi} \tau(?, C) =_{\beta\Pi} \tau(?, B)$.* □

Theorem 6.18 *(Strong Normalisation for $\vdash$)*
If A is $?^{\vdash}$-legal, then $SN_{\twoheadrightarrow_\beta}(A)$.
Proof: *By Lemma 6.13, $? \vdash_\beta A : \overline{\tau(?, A)}$. Hence, by Theorem 3.15, $SN_{\twoheadrightarrow_\beta}(A)$.* □

We believe that if $? \vdash A$ then $\mathrm{SN}_{\twoheadrightarrow_{\beta\Pi}}(\tau(?, A))$. We leave this as an open problem for the moment.

Remark 6.19 *Note that from Lemmas 6.11, 6.13 and 4.9 , Π-reduction is necessary for splitting $? \vdash A : B$ into $? \vdash A$ and $\tau(?, A) =_{\beta\Pi} B$, yet $\vdash_{\beta\Pi}$ is not necessary. This is shown by the following proposition (call B $\vdash_\beta$-legal type iff $B \equiv \Box$ or $? \vdash_\beta B : S$ for some $?, S$).*

Proposition 6.20
$? \vdash_\beta A : B \Leftrightarrow ? \vdash A \wedge \tau(?, A) =_{\beta\Pi} B \wedge B$ *is $\vdash_\beta$-legal type.*
Proof:
$\Rightarrow$) By Lemma 4.9, $? \vdash_{\beta\Pi} A : B$. Hence, by Lemma 6.11, $? \vdash A$ and $\tau(?, A) =_{\beta\Pi} B$. Moreover, by Corollary 3.10, as $? \vdash_\beta A : B$, B is $\vdash_\beta$-legal type.
$\Leftarrow$) By Lemma 6.13, $\downarrow \overline{\tau(?, A)}$ and $? \vdash_\beta A : \overline{\tau(?, A)}$. Moreover, $B \twoheadrightarrow_\beta \overline{\tau(?, A)}$.

- *Case $B \equiv \Box$ then $\overline{\tau(?, A)} \equiv \Box$ and $? \vdash_\beta A : B$.*
- *Case $? \vdash_\beta B : S$ then by $? \vdash_\beta A : \overline{\tau(?, A)}$, $B =_\beta \overline{\tau(?, A)}$ and conversion, we get $? \vdash_\beta A : B$.* □

Note in this proposition that B is $\vdash_\beta$-legal type is needed. The reason is obvious of course. We may have $\tau(?, A) =_{\beta\Pi} B$ and $? \vdash A$, yet B contains Π-redexes, hence making it impossible to have $? \vdash_\beta A : B$. For example, if $? \equiv \lambda_{p:*}.\lambda_{z:*}.\lambda_{u:z}$, $A \equiv (\lambda_{x:z}.(\lambda_{y:z}.p)u)u$ and $B \equiv (\Pi_{y:z}.*)u$ then obviously $? \vdash A$ and $\tau(?, A) \equiv (\Pi_{x:z}.(\Pi_{y:z}.*)u)u =_{\beta\Pi} B$ but $? \not\vdash_\beta A : B$. In fact, B is not a legal term nor type for $\vdash_\beta$ according to Lemma 3.11. We do however have the following:

Lemma 6.21
If B is in $\beta\Pi$-normal form, then $? \vdash_\beta A : B \Leftrightarrow ? \vdash A \wedge \tau(?, A) =_{\beta\Pi} B$.
Proof:
$\Rightarrow$) is a corollary of Proposition 6.20. $\Leftarrow$) As B is in $\beta\Pi$-normal form and $\tau(?, A) =_{\beta\Pi} B$, we get $\overline{\tau(?, A)} \equiv B$. Now, use Lemma 6.13 to get $? \vdash_\beta A : B$. □

7 Conclusion

In Section 1 we introduced various desirable properties for type theory. In this section we remark how these properties have been treated in our paper discussing any limitations or future work.

1. Π-reduction behaves like β-reduction. This has of course been a fundamental point to our paper. In fact, recall Remark 6.19 which explained that Π-reduction is necessary for splitting the question *does A have B as a type* into the two questions about whether A is typable and whether its preference type is equal to B.

2. Compatibility. This has certainly been achieved in $\vdash_{\beta\Pi}$ via the new application rule.

3. Unified treatment of terms and types. This is achieved slightly in the Barendregt Cube. With our Π-reduction we go a step further allowing types to have similar reduction rights as terms.

4. The ability to divide two important questions of typing. This has been achieved in our paper by replacing $\vdash_\beta$ or $\vdash_{\beta\Pi}$ by $\vdash$ and τ. The important relation between the standard way of typing terms and our two separate questions is given in Proposition 6.20.

As for the other points, it has been made clear in the paper that $\tau(A)$ plays the role of a preference type for A and that it is very easy to calculate. Furthermore, we have eliminated the conversion rule from the typing rules for $\vdash$.

Now, let us reflect on the legal terms obtained via $\vdash_{\beta\Pi}$ comparing them to those legal terms of $\vdash_\beta$. Lemma 3.11 informs us that $\vdash_{\beta\Pi}$-legal terms and contexts have no Π-redexes. Lemma 4.7 tells us that if $? \vdash_{\beta\Pi} A : B$ then we can only have Π-redexes in B and if this is the case than B is itself the unique Π-redex. So really, we have not increased our terms or types much via $\vdash_{\beta\Pi}$. Still this tiny increase is what led to the loss of SR (even though we get WSR). It is however easy to get back full SR in two different ways which have been ignored in this article because they emphasize different issues than those we emphasize in this paper. We will here just briefly discuss how these two methods work.

The first method (which is being investigated) adds definitions to $\vdash_{\beta\Pi}$ via the following extra typing rule (note $\pi = \lambda$ or Π):

$$\text{(def rule)} \qquad \frac{?.(\pi_{x:A}.-)B \vdash_{\beta\Pi} C : D}{? \vdash_{\beta\Pi} (\pi_{x:A}.C)B : D[x := B]}$$

The intuition behind this rule is obvious. It says that if $C : D$ can be typed using the definition that x of type A is B, then $(\pi_{x:A}.C)B : D[x := B]$ can be typed without this definition. With definitions, terms, types and contexts contain as many Π-redexes as they like.

A second method to retrieve back full SR would be to add the following rule to $\vdash_{\beta\Pi}$:

$$\frac{? \vdash_{\beta\Pi} \pi_{x:A}.C : S \qquad ? \vdash_{\beta\Pi} B : A}{? \vdash_{\beta\Pi} (\pi_{x:A}.C)B : S}$$

The intuition behind this rule is obvious. In fact, think of the formation rule. For $\Pi_{x:A}.B : S$ we needed $B : S$. Now, if $B : S$ then $B[x : a] : S$ and hence $(\Pi_{x:A}.B)a : S$. With this extension, terms would contain as many Π-redexes as they like. Contexts however would still not contain any Π-redex.

References

[Barendregt 92] Barendregt, H., Lambda calculi with types, *Handbook of Logic in Computer Science*, volume II, eds. Abramsky S., Gabbay D.M., Maibaum T.S.E., Oxford University Press, 118-414, 1992.

[de Bruijn 74] Bruijn, N.G. de, Some extensions of AUTOMATH: the AUT-4 family, Dept. of Mathematics, Eindhoven University of Technology, 1974.

[Church 1932] Church, A., A set of postulates for the foundation of logic, *Annals of Math.* 33 (1932), 346–366 and 34 (1933), 839–864.

[Church 1940] Church, A., A formulation of the simple theory of types, *Journal of Symbolic Logic* 5 (1940), 56–68.

[Frege 1879] Frege, G., *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens* (Halle, Verlag von Louis Nebert, 1879). Reprint 1964 (Hildesheim, Georg Olms Verlagsbuchhaltung).

[Hilbert and Ackermann 1928] Hilbert, D. and Ackermann, W., *Grundzüge der theoretischen Logik* (Berlin, Springer Verlag, 1928).

[How 80] Howard, W.A., The formulae-as-types notion of constructions, in *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, eds. Hindley J.R., and Seldin, J.P. Academic Press, 1980.

[KN 93] Kamareddine, F., and Nederpelt, R.P., On stepwise explicit substitution, *International Journal of Foundations of Computer Science 4 (3)*, 197-240, 1993.

[NK 94] Kamareddine, F., and Nederpelt, R.P., A unified approach to type theory through a refined λ-calculus, *Theoretical Computer Science 136*, 183-216, 1994.

[KN 9y] Kamareddine, F., and Nederpelt, R.P., *The Beauty of the λ-Calculus*, in preparation.

[KN 94] Kamareddine, F., and Nederpelt, R.P., Canonical Typing and Π-Conversion, Research report, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.

[Ter 89] Terlouw, J., Een nadere bewijstheoretische analyse van GSTT's. Technical report, Department of Computer Science, University of Nijmegen, 1989.

[Whitehead and Russell 1910] Whitehead, A.N. and Russell, B., *Principia Mathematica* (Cambridge, Cambridge University Press, 1910/1913). Reprint 1960, same editor.