Journal of Functional Programming 1 (2): 155–189, April 1991

Modular proof of strong normalization for the calculus of constructions

HERMAN GEUVERS* AND MARK-JAN NEDERHOF
Faculty of Mathematics and Computer Science, University of Nijmegen, The Netherlands

Abstract

We present a modular proof of strong normalization for the Calculus of Constructions of Coquand and Huet (1985, 1988). This result was first proved by Coquand (1986), but our proof is more perspicious. The method consists of a little juggling with some systems in the cube of Barendregt (1989), which provides a fine structure of the calculus of constructions. It is proved that the strong normalization of the calculus of constructions is equivalent with the strong normalization of $F\omega$.

In order to give the proof, we first establish some properties of various type systems. Therefore, we present a general framework of typed lambda calculi, including many well-known ones.

Capsule review

The calculus of constructions due to Coquand and Huet (1985, 1988) is a very popular subject among those interested in computer science-oriented aspects of intuitionistic type theory. The strong normalization theorem for it, stating that all computation sequences terminate, is one of the most basic results. The authors present a new proof of strong normalization obtained essentially by combining ideas from several previous proofs of similar results for different systems, allowing them to divide up the difficulties and cope with them one at a time. The paper is self-contained and all proofs are given in detail.

1 Introduction

The strong normalization (SN) property for the calculus of constructions (Coquand and Huet, 1985, 1988) was proved by Coquand (1986). This proof is rather 'baroque', although it is reminiscent of other proofs of strong normalization for systems like the simply or polymorphically typed lambda calculus. Therefore, we looked for a conceptually more perspicious proof.

Barendregt (1989) gives a fine structure of the calculus of constructions. He defines a natural cube of eight type systems, ordered along the edges by inclusion, of which the smallest system is the simply typed lambda calculus $\lambda\!\rightarrow$, and the most complicated system is the calculus of constructions λC. Other systems in Barendregt's

* Partially supported by the EEC 'Project Stimulation ST2J/0374/C(EDB): Lambda Calcul Typé'.

cube include the second order (polymorphic) lambda calculus λ2, Girard's Fω system (called λω in the cube) and λP, a system related to the AUTOMATH system AUT-QE and studied by Harper *et al.* (1987) under the name LF

The first step in our proof of strong normalization consists of defining a mapping from λC into λω such that reduction of terms is preserved. (This generalizes a method of Harper *et al.* (1987) mapping λP into λ→). Then, motivated by three kinds of abstractions possible in the various parts of the cube, reductions in λω are divided into three kinds. Two of these reductions turn out to be strongly normalizing. (For one of them we need that λ→ is SN.) To show that the third notion of reduction (even if mixed with the other two) is also SN, we map the terms of λω into the set Λ of typefree lambda terms. Then a higher order version of the argument used for λ2 (by Girard (1972), or Tait (1975)) shows that λω is SN.

The lemmas that are used in the proof of strong normalization are valid for a large class of systems not included in the cube. Therefore, we start this paper by giving a more general notion of type system, following definitions by Terlouw (1989*a*) and Berardi (1988), and prove the basic lemmas for this general notion. This puts the properties of the systems of the cube in a larger framework that might be useful for other arguments about typed lambda calculi. Berardi has shown that various logical systems also fit in the general notion of type system (see Barendregt, 1989, for some examples), which again stresses the relationship between typed lambda calculus and logic, usually expressed by the 'formulae-as-types' notion, discussed by Howard (1980).

2 Framework for type systems

A general notion of type system is presented, following definitions given by Terlouw (1989*a*, *b*) and Berardi (1988), generalizing Barendregt's cube. This notion will serve as a framework for reasoning about type systems and comparing various type systems with each other. One can prove a number of basic properties for this notion, which show how things work. We restrict ourselves to those properties that will be useful in the argument about strong normalization.

Definition 1
A Generalized Type System (*GTS*) is a system consisting of the following objects

(i) $\mathcal{C}$, a set of constants,
(ii) $\mathcal{S} \subseteq \mathcal{C}$, a set of sorts,
(iii) $\mathcal{A}$, a set of pairs (c, c'), with $c, c' \in \mathcal{C}$,
(iv) $\mathcal{R}$, a set of triples (s, s', s''), with $s, s', s'' \in \mathcal{S}$,
(v) for every $s \in \mathcal{S}$, a set $\mathcal{V}^s$ of variables.

The axiom pairs (c, c′) will usually be denoted by c:c′. If $s' \equiv s''$ in a rule (s, s', s''), then the rule (s, s', s'') will be written as (s, s'). A GTS will be denoted by the quadruple $(\mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{R})$. For the set of all variables, we shall write $\mathcal{V}$, so $\mathcal{V} = \bigcup_{s \in \mathcal{S}} \mathcal{V}^s$.

The idea of the GTS definition is that the sorts are the universes of the type system, where the axioms give the hierarchical structure between them. The constants are special objects of the type system, belonging to a universe or another constant. The rules restrict the formation of the Π-type, the collection of (possibly dependent) functions from one type to another. In Chapter 3 it can be seen how the GTS definition works in the case of well-known systems such as simply and polymorphically typed lambda calculus and the calculus of constructions.

Definition 2

(i) A GTS ($\mathcal{C}$, $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$) is *functional* iff

- $\mathcal{A} \subseteq \mathcal{C} \times \mathcal{C}$ is a *function*, i.e., $\forall c, c', c'' \in \mathcal{C}$. $[c:c', c:c'' \in \mathcal{A} \Rightarrow c' \equiv c'']$,
- $\mathcal{R} \subseteq (\mathcal{S} \times \mathcal{S}) \times \mathcal{S}$ is a *function*, i.e., $\forall s_1, s_2, s_3, s_3' \in \mathcal{S}$. $[(s_1, s_2, s_3), (s_1, s_2, s_3') \in \mathcal{R} \Rightarrow s_3 \equiv s_3']$,

(ii) A GTS ($\mathcal{S}$, $\mathcal{C}$, $\mathcal{A}$, $\mathcal{R}$) is *injective* iff

- $\mathcal{A}$ is *injective* on $\mathcal{S} \times \mathcal{S}$, i.e., $\forall s, s' \in \mathcal{S}, c \in \mathcal{C}$. $[s:c, s':c \in \mathcal{A} \Rightarrow s \equiv s']$,
- $\mathcal{R}$ is *injective in its second argument*, i.e., $\forall s_1, s_2, s_2', s_3 \in \mathcal{S}$. $[(s_1, s_2, s_3), (s_1, s_2', s_3) \in \mathcal{R} \Rightarrow s_2 \equiv s_2']$.

A motivation for these two definitions is that type systems which are functional will turn out to have the so called 'uniqueness of assignment' property; the type that can be assigned to a certain object is unique up to β-equality. (Even without knowing the inference rules for GTSs, it will be clear that if $\mathcal{A}$ is not injective, a constant might be typed with two different sorts which are not β-equal.) Further, the properties of functionality and injectivity give rise to nice classifications of objects in a type system. It is not worth explaining the interest of these two definitions at this point. They will only become clear in view of the specific properties for functional and injective GTSs that are given at the end of this chapter.

Let in the following $\zeta = (\mathcal{S}, \mathcal{C}, \mathcal{A}, \mathcal{R})$ be a GTS.

Definition 3

(i) The collection of *pseudoterms of* ζ, $\mathcal{PST}(\zeta)$, is defined by

$$t ::= \mathcal{V} \mid \mathcal{C} \mid (t\,t) \mid (\lambda \mathcal{V}:t.t) \mid (\Pi \mathcal{V}:t.t).$$

(ii) If a term is one of the last three forms, it will be called a *composed* term.

The notions of *bound variable* and *free variable* of a pseudoterm are defined as usual, λ and Π bind variables.

The *substitution operator* works as in the untyped lambda calculus; $t[x := t']$ denotes the substitution of t' for x in the term t. Substitution is only allowed if no free variables become bound.

Just as in untyped lambda calculus, terms that only differ from each other in their bound variables will be identified; we work *modulo* α-*conversion*. If the terms t and

t' are α-convertible, this is denoted by $t \equiv t'$. In general, it will be assumed that the bound variables in a term differ from the free ones.

Also, the notion of *subterm* is directly copied from the untyped case. (For precise definitions, we refer to Barendregt, 1984).

Definition 4

(i) a *redex* is a pseudoterm of the form $(\lambda x{:}t.t')\,t''$.

(ii) *one-step* β-*reduction*, $\to_\beta$ (or just $\to$), is defined by $(\lambda x{:}t.t')\,t'' \to_\beta t'[x := t'']$, and if $t' \to_\beta t''$, then $t\,t' \to_\beta t\,t''$, $t'\,t \to_\beta t''t$, $\lambda x{:}t.t' \to_\beta \lambda x{:}t.t''$, $\lambda x{:}t'.t \to_\beta \lambda x{:}t''.t$, $\Pi x{:}t.t' \to_\beta \Pi x{:}t.t''$ and $\Pi x{:}t'.t \to_\beta \Pi x{:}t''.t$, for all $t, t', t'' \in \mathrm{PST}(\zeta)$.

(iii) β-*reduction*, $\twoheadrightarrow$, is the transitive reflexive closure of $\to$.

(iv) β-*conversion*, $=$, is the least equivalence relation generated by $\twoheadrightarrow$.

Theorem 5

The reduction relation $\twoheadrightarrow$ *on* $PST(\zeta)$ *satisfies the* Church–Rosser property, *i.e.* $\forall M, N \in PST(\zeta).\ [M = N \Rightarrow \exists P \in PST(\zeta).\ M \twoheadrightarrow P\ \&\ N \twoheadrightarrow P]$.

Proof. The proof of the Church–Rosser property for pseudoterms can be given in the same way as the proof of Church–Rosser for the untyped lambda calculus. We do not give it here. (For details see Barendregt and Dekkers, 1990). □

Definition 6

(i) A ζ-*assignment* is an expression of the form $A{:}B$, with $A, B \in \mathrm{PST}(\zeta)$,

(ii) A ζ-*declaration* is an expression of the form $x{:}A$, with $A \in \mathrm{PST}(\zeta)$, $x \in V_{AR}$,

(iii) A ζ-*pseudocontext* is a finite sequence of ζ-declarations,

(iv) A ζ-*statement* is an expression of the form $\Gamma \vdash A{:}B$, with Γ a ζ-pseudocontext, $A{:}B$ a ζ-assignment.

Definition 7

Let Γ' and $\Gamma = x_1{:}A_1, \ldots, x_n{:}A_n$ be pseudocontexts.

(i) The *domain of* Γ, $dom(\Gamma)$, is the set $\{x_1, x_2, \ldots, x_n\}$,

(ii) For $i \leqslant n$, $\Gamma\lceil i$ (Γ *restricted to* i) is the pseudocontext $x_1{:}A_1, \ldots, x_i{:}A_i$,

(iii) For $i \leqslant n$, $\Gamma \setminus (x_i{:}A_i) = x_1{:}A_1, \ldots, x_{i-1}{:}A_{i-1}, x_{i+1}{:}A_{i+1}, \ldots, x_n{:}A_n$,

(iv) $\Gamma' \leqslant \Gamma$ (Γ' *is prefix of* Γ) iff $\Gamma' = \Gamma\lceil i$ for some $i \leqslant n$,

(v) $\Gamma' \subseteq \Gamma$ (Γ' *is subcontext of* Γ) iff $x{:}A$ in $\Gamma' \Rightarrow x{:}A$ in Γ,

(vi) $\Gamma \twoheadrightarrow \Gamma'$ iff $\Gamma' = x_1{:}A_1', \ldots, x_n{:}A_n'$ and $A_i \twoheadrightarrow A_i'$ for all $i \leqslant n$.

We now define for a pseudocontext Γ and pseudoterms A and B, the notion '$\Gamma \vdash A{:}B$ is true'. This definition picks the 'legal' terms out of the set of pseudoterms and the 'legal' contexts out of the set of pseudocontexts. Instead of '$\Gamma \vdash A{:}B$ is true', we shall just write $\Gamma \vdash A{:}B$, in words Γ *proves* $A{:}B$, or Γ *assigns* B *to* A. The notion of $\Gamma \vdash A{:}B$ is generated by the axioms, and inference rules of the system ζ. In case the system ζ in which we are working is not clear from the context, we write $\Gamma \vdash_\zeta A{:}B$.

Definition 8

(i) The *axioms of* ζ are the statements $\vdash c:c'$, with $c:c' \in AXIOM$,

(ii) The *inference rules of* ζ are the rules of the following six forms

$$\text{(Start)}\quad \frac{\Gamma \vdash A:s}{\Gamma, x:A \vdash x:A} \qquad \text{with } s \in SORT,\ x \in VAR^s,\ x \notin \mathrm{dom}(\Gamma),$$

$$\text{(Weakening)}\quad \frac{\Gamma \vdash A:s \quad \Gamma \vdash B:C}{\Gamma, x:A \vdash B:C} \qquad \text{with } s \in SORT,\ x \in VAR^s,\ x \notin \mathrm{dom}(\Gamma),$$

$$\text{(Π-rule)}\quad \frac{\Gamma \vdash B_1:s_1 \quad \Gamma, x:B_1 \vdash B_2:s_2}{\Gamma \vdash \Pi x:B_1.B_2:s_3} \qquad \text{with } (s_1, s_2, s_3) \in RULE,$$

$$\text{(λ-rule)}\quad \frac{\Gamma \vdash B_1:s_1 \quad \Gamma, x:B_1 \vdash B_2:s_2 \quad \Gamma, x:B_1 \vdash C:B_2}{\Gamma \vdash \lambda x:B_1.C:\Pi x:B_1.B_2}$$

$$\text{with } (s_1, s_2, s_3) \in RULE \text{ for some } s_3 \in SORT,$$

$$\text{(Application)}\quad \frac{\Gamma \vdash B_1:\Pi x:C_1.C_2 \quad \Gamma \vdash B_2:C_1}{\Gamma \vdash B_1 B_2:C_2[x:=B_2]},$$

$$\text{(Conversion)}\quad \frac{\Gamma \vdash B:C \quad \Gamma \vdash C':s}{\Gamma \vdash B:C'} \qquad \text{with } C = C',\ s \in SORT.$$

The statements above the line will be called *premises*, the statements below the line the *conclusions* of a rule.

Definition 9

A *derivation* in the system ζ is a finite well-founded tree with

(i) each leaf of the tree is an axiom of ζ,

(ii) each node of the tree which is not a leaf is a conclusion of an inference rule, such that the successors of the node are exactly the premises of the inference rule.

For Δ and Δ' derivations in ζ, $\Delta < \Delta'$ (Δ subderivation of Δ') is defined as usual.

Definition 10

(I) For a derivation Δ, the *length of* Δ, $lh(\Delta)$, is inductively defined by

(i) If Δ consists only of an axiom, then $lh(\Delta) = 0$,

(ii) If the premises of the conclusion of Δ are $F_1, \ldots, F_n$, with derivations $\Delta_1, \ldots, \Delta_n$, then $lh(\Delta) = max\{lh(\Delta_i) \mid 1 \leqslant i \leqslant n\} + 1$.

(II) For a derivation Δ, the *trace of* Δ is the path in Δ that

(i) starts with the root and ends with a leaf,

(ii) takes the left path in case of (application) or (conversion),

(iii) takes the right path in case of (weakening), (Π-rule) or (λ-rule).

Definition 11
Let Γ be a pseudocontext, $A, B \in \mathrm{PST}(\zeta)$, Δ a derivation in ζ.

(i) Δ *is a derivation of* $\Gamma \vdash A:B$ or $\Delta:(\Gamma \vdash A:B)$ iff $\Gamma \vdash A:B$ is the root of the derivation Δ,
(ii) $\Gamma \vdash A:B$ *is true* or $\Gamma \vdash A:B$ iff $\Delta:(\Gamma \vdash A:B)$ for some derivation Δ in ζ.

Notation 12. $\Gamma \vdash A:B:C$ iff $\Gamma \vdash A:B$ and $\Gamma \vdash B:C$.

Definition 13
Let $s \in \mathit{SORT}$.

(i) $\mathit{Context}(\zeta) = \{\Gamma \mid \Gamma \vdash A:B \text{ for some } A, B \in \mathrm{PST}(\zeta)\}$,
(ii) $\Gamma\text{-}\mathit{Term}(\zeta) = \{A \mid \Gamma \vdash A:B \text{ or } \Gamma \vdash B:A \text{ for some } B \in \mathrm{PST}(\zeta)\}$,
(iii) $\Gamma\text{-}s\text{-}\mathit{Term}(\zeta) = \{A \mid \Gamma \vdash A:s\}$,
(iv) $\Gamma\text{-}s\text{-}\mathit{Elt}(\zeta) = \{A \mid \Gamma \vdash A:B:s \text{ for some } B \in \mathrm{PST}(\zeta)\}$,
(v) $\mathit{Term}(\zeta) = \bigcup_{\Gamma \in \mathit{Context}(\zeta)} \Gamma\text{-}\mathit{Term}(\zeta)$,
(vi) $s\text{-}\mathit{Term}(\zeta) = \bigcup_{\Gamma \in \mathit{Context}(\zeta)} \Gamma\text{-}s\text{-}\mathit{Term}(\zeta)$,
(vii) $s\text{-}\mathit{Elt}(\zeta) = \bigcup_{\Gamma \in \mathit{Context}(\zeta)} \Gamma\text{-}s\text{-}\mathit{Elt}(\zeta)$.

Definition 14
Let ζ be a GTS, $M \in \mathit{Term}(\zeta)$, $n \in \mathbb{N}$.

(i) n *is an upperbound to the reductions starting from* M iff

$$\forall M_1, M_2, \ldots, M_m \in \mathit{Term}(\zeta).\, [M \to M_1 \to \ldots \to M_{m-1} \to M_m \Rightarrow m \leqslant n],$$

(ii) M *is strongly normalizable*, or $\mathrm{SN}(M)$, iff $\exists n \in \mathbb{N}.\, [n$ is an upperbound to the reductions starting from $M]$,
(iii) ζ *satisfies the strong normalization property*, or $\zeta \vDash \mathrm{SN}$, iff $\forall M \in \mathit{Term}(\zeta).\, \mathrm{SN}(M)$.

The fact that all terms of a set X are strongly normalizable will be denoted by $\mathrm{SN}(X)$.

In the following, the double negated version of $\zeta \vDash \mathrm{SN}$ will sometimes be used: $\neg\exists M_1 \in \mathit{Term}(\zeta).\ \forall m \in \mathbb{N}.\ \exists M_2, M_3, \ldots, M_m \in \mathit{Term}(\zeta).\ \forall i \leqslant m-1.\ [M_i \to M_{i+1}]$, stating that there are no infinite reduction sequences in ζ. The proofs given below are therefore not all constructive. Analysing the proofs, one can see, however, that the proof of equivalence of $\lambda\omega \vDash \mathrm{SN}$ and $\lambda\mathrm{P}\omega \vDash \mathrm{SN}$ can be done in first order Heyting arithmetic.

From the axioms and inference rules, it will be clear that if a constant does not occur in any of the axioms or on the third place of a rule, then it does not occur in any statement of ζ. In the following, we shall therefore assume that

$$c \in \mathit{CONS} \Rightarrow \exists c'.\, c:c' \in \mathit{AXIOM} \vee c':c \in \mathit{AXIOM}$$
$$\vee\ \exists s, s'.\, (s, s', c) \in \mathit{RULE}.$$

The rest of this chapter will consist of lemmas and proofs for the generalized type systems. For examples we refer to Chapter 3, in which the systems of Barendregt's

cube are defined as GTSs. Some more exotic examples can be found in Barendregt (1989), e.g., the definition of predicate logics as GTSs. Barendregt also shows why, in general, we want the set *RULE* to consist of triples (s_1, s_2, s_3), and not just pairs (s_1, s_2).

Lemma 15 (*Free variables*)
For $\Gamma = x_1:A_1, x_2:A_2, \ldots, x_n:A_n$ *and* $\Gamma \vdash B:C$,

(i) $FV(B:C) \subseteq \{x_1, \ldots, x_n\}$,
(ii) $\forall i,j \leqslant n . x_i \equiv x_j \Rightarrow i = j$.

Proof. Induction on the length of the derivation of $\Gamma \vdash B:C$, distinguishing cases according to the last applied inference rule. □

Lemma 16
For $\Gamma = x_1:A_1, x_2:A_2, \ldots, x_n:A_n \in Context(\zeta)$,

(i) $\Gamma \vdash c:c'$ *for all* $c:c' \in AXIOM$,
(ii) $\Gamma \vdash x_i:A_i$ *for all* $i \leqslant n$.

Proof. The proof of (i) is by easy induction on the length of the tree that proves $\Gamma \in Context(\zeta)$.

For the proof of (ii), let $\Delta:(\Gamma \vdash B:C)$. The proof of $\Gamma \vdash x_i:A_i$ for all $i \leqslant n$ is by induction on the length of Δ, distinguishing cases according to the last applied rule. The two interesting cases are when this is (start) or (weakening). We only treat the case for the last rule being (weakening), as the other case is quite similar.

If the last rule is (weakening), then

$$\frac{x_1:A_1, \ldots, x_{n-1}:A_{n-1} \vdash A_n:s \quad x_1:A_1, \ldots, x_{n-1}:A_{n-1} \vdash B:C}{x_1:A_1, \ldots, x_n:A_n \vdash B:C}.$$

With one application of (start) we find that $\Gamma \vdash x_n:A_n$. By induction hypothesis $\Gamma\lceil n-1 \vdash x_i:A_i$ for all $i \leqslant n-1$, so with one application of (weakening): $\Gamma \vdash x_i:A_i$ for all $i \leqslant n-1$. □

Lemma 17 (*Substitution*)
For Γ_1 *and* $\Gamma_1, y:A, \Gamma_2 \in Context(\zeta)$, $A, B, C, D \in Term(\zeta)$, $y \in VAR$

$$\Gamma_1, y:A, \Gamma_2 \vdash B:C \;\&\; \Gamma_1 \vdash D:A \Rightarrow \Gamma_1, \Gamma_2[y:=D] \vdash B[y:=D]:C[y:=D].$$

Proof. By induction on the length of the derivation of $\Gamma_1, y:A, \Gamma_2 \vdash B:C$, assuming that $\Gamma_1 \vdash D:A$. We distinguish cases according to the last applied inference rule. If this rule is (start) or (weakening), we distinguish subcases $\Gamma_2 = \varnothing$ and $\Gamma_2 \neq \varnothing$.

If the last rule is (Π-rule), (λ-rule) or (conversion), or $\Gamma_2 \neq \varnothing$ and the last rule is (start) or (weakening), then the statement follows immediately from the induction hypothesis and an application of the rule.

If the last rule is (start) and $\Gamma_2 = \varnothing$, then $B \equiv y$ and $C \equiv A$. Now $y[y:=D] \equiv D$ and $A[y:=D] \equiv A$, so we are done by the assumption $\Gamma_1 \vdash D:A$.

If the last rule is (weakening) and $\Gamma_2 = \varnothing$, then

$$\frac{\Gamma_1 \vdash A:s \quad \Gamma_1 \vdash B:C}{\Gamma_1, y:A \vdash B:C}.$$

Now $y \notin FV(B:C)$, so $B[y:=D] \equiv B$ and $C[y:=D] \equiv C$, and we are done.

If the last rule is (application), then $B \equiv B_1 B_2$, $C \equiv C_2[x:=B_2]$ and

$$\frac{\Gamma_1, y:A, \Gamma_2 \vdash B_1:\Pi x:C_1.C_2 \quad \Gamma_1, y:A, \Gamma_2 \vdash B_2:C_1}{\Gamma_1, y:A, \Gamma_2 \vdash B_1 B_2:C_2[x:=B_2]}.$$

Now by induction hypothesis and (application)

$$\Gamma_1, \Gamma_2[y:=D] \vdash B_1 B_2[y:=D]:C_2[y:=D][x:=B_2[y:=D]].$$

We may assume that $x \notin FV(\Gamma_1, y:A, \Gamma_2)$, so $x \not\equiv y$ and $x \notin FV(D)$. It follows that $C_2[y:=D][x:=B_2[y:=D]] \equiv C_2[x:=B_2][y:=D]$, and we are done. □

Lemma 18 (*Thinning*)
For $\Gamma, \Gamma' \in Context(\zeta)$, $B, C \in Term(\zeta)$

$$\Gamma \vdash B:C \;\&\; \Gamma \subseteq \Gamma' \Rightarrow \Gamma' \vdash B:C.$$

Proof. By induction on the length of the derivation of $\Gamma \vdash B:C$, distinguishing cases according to the last applied rule.

If $\Gamma \vdash B:C$ is an axiom or the last rule was (start), we are done by lemma 2.16. If the last rule is (weakening), we are done by the induction hypothesis. If the last rule is (application) or (conversion), the statement follows from the induction hypothesis and an application of the rule.

The argument for the cases when the last rule is (Π-rule) or (λ-rule) is similar. We treat the case for (Π-rule).

Let $\Gamma' \supseteq \Gamma$, $B \equiv \Pi x:B_1.B_2$, $C \equiv s$, and

$$\frac{\Gamma \vdash B_1:s_1 \quad \Gamma, x:B_1 \vdash B_2:s_2}{\Gamma \vdash \Pi x:B_1.B_2:s}.$$

Then we may assume that $x \notin \mathrm{dom}(\Gamma')$. By induction hypothesis $\Gamma' \vdash B_1:s_1$, so $\Gamma', x:B_1 \in Context(\zeta)$. $\Gamma', x:B_1 \supseteq \Gamma, x:B_1$, so by induction hypothesis $\Gamma', x:B_1 \vdash B_2:s_2$. With one application of (Π-rule) $\Gamma' \vdash \Pi x:B_1.B_2:s$. □

Lemma 19 (*Stripping*)
For $\Gamma = x_1:A_1, \ldots, x_n:A_n \in Context(\zeta)$, $M, N, P, R \in Term(\zeta)$

(i) $\Gamma \vdash c:R$, *with* $c \in CONS \Rightarrow \exists c' \in CONS.[R = c' \;\&\; c:c' \in AXIOM]$,

(ii) $\Gamma \vdash x:R$, *with* $x \in VAR \Rightarrow \exists i \leqslant n \exists s \in SORT.[\; x \equiv x_i \in VAR^s \;\&\; R = A_i \;\&\; \Gamma\lceil i-1 \vdash A_i:s]$,

(iii) $\Gamma \vdash \Pi x{:}M.N{:}R \Rightarrow \Gamma \vdash M{:}s_1$ &
$\Gamma, x{:}M \vdash N{:}s_2$ &
$R = s_3$,
for some $(s_1, s_2, s_3) \in \mathit{RULE}$,

(iv) $\Gamma \vdash \lambda x{:}M.N{:}R \Rightarrow \Gamma \vdash M{:}s_1$ &
$\Gamma, x{:}M \vdash B{:}s_2$ &
$\Gamma, x{:}M \vdash N{:}B$ &
$\Gamma \vdash \Pi x{:}M.B{:}s_3$ &
$R = \Pi x{:}M.B$,
for some $(s_1, s_2, s_3) \in \mathit{RULE}$, $B \in \mathit{Term}(\zeta)$,

(v) $\Gamma \vdash MN{:}R \Rightarrow \Gamma \vdash M{:}\Pi x{:}A.B$ &
$\Gamma \vdash N{:}A$ &
$R = B[x := N]$,
for some $A, B \in \mathit{Term}(\zeta)$, $x \in \mathit{VAR}$,

(vi) $\Gamma \vdash P{:}R \Rightarrow \exists c \in \mathit{CONS}.[R \equiv c \vee \Gamma \vdash R{:}c \,\&\, c \in \mathit{SORT}]$.

Proof (*i*)–(*v*). Let $\Delta{:}(\Gamma \vdash P{:}R)$ in one of the first five cases above. When we follow the trace of Δ, we only pass applications of (weakening) and (conversion), which do not change the term P, until we hit upon a rule by which the term P is introduced. In case (i) this is an axiom, in case (ii) this is (start), in case (iii) the (Π-rule), in case (iv) the (λ-rule), and in case (v) the (application). In all cases, the conclusion of the rule is $\Gamma' \vdash P{:}R'$, with $\Gamma' \leqslant \Gamma$ and $R' = R$. The proof of the five cases above now follows immediately by taking a look at the rule by which P is introduced, thinning the context Γ' to Γ and converting R' to R.

Proof (*vi*). By induction on the structure of P. Following the trace up in the tree $\Delta{:}(\Gamma \vdash P{:}R)$, we only pass applications of (weakening) until we hit upon (conversion), or the rule by which P is introduced. In the first case we conclude that $\Gamma \vdash R{:}s$, for some $s \in \mathit{SORT}$, and we are done. In the second case we distinguish subcases according to the structure of P (cases (i)–(v) above).

If $P \equiv c \in \mathit{CONS}$, $P \equiv x \in \mathit{VAR}$, $P \equiv \Pi x{:}M.N$ or $P \equiv \lambda x{:}M.N$, it is immediately clear that $R \equiv c'$ for some $c' \in \mathit{CONS}$ or $\Gamma \vdash R{:}s$ for some $s \in \mathit{SORT}$.

If $P \equiv MN$, then $R \equiv C_2[x := N]$, $\Gamma \vdash M{:}\Pi x{:}C_1.C_2$ and $\Gamma \vdash N{:}C_1$ for certain C_1, C_2. Applying the induction hypothesis to $\Gamma \vdash M{:}\Pi x{:}C_1.C_2$ and case (iii), we find that $\Gamma, x{:}C_1 \vdash C_2{:}s_2$. With the substitution lemma we obtain $\Gamma \vdash C_2[x := N]{:}s_2$. □

Lemma 20 (*Permutation*)
For Γ_1 *and* $\Gamma_1, x{:}A, y{:}D, \Gamma_2 \in \mathit{Context}(\zeta)$, $B, C \in \mathit{Term}(\zeta)$

$$\Gamma_1, x{:}A, y{:}D, \Gamma_2 \vdash B{:}C \,\&\, \Gamma_1 \vdash D{:}s \Rightarrow \Gamma_1, y{:}D, x{:}A, \Gamma_2 \vdash B{:}C.$$

Proof. Remark that $\Gamma_1, y{:}D$ is a legal context and so, by $\Gamma_1 \vdash A{:}s'$ and thinning, $\Gamma_1, y{:}D, x{:}A$ is a legal context too. If $\Gamma_2 = z_1{:}E_1, \ldots, z_n{:}E_n$, we may conclude that $\Gamma_1, y{:}D, x{:}A \vdash E_1{:}s''$ for some sort s'', and so $\Gamma_1, y{:}D, x{:}A, z_1{:}E_1$ is legal. Proceeding in this way for all $i \leqslant n$, we find that $\Gamma_1, y{:}D, x{:}A, \Gamma_2$ is a legal context. Using thinning one concludes that $\Gamma_1, y{:}D, x{:}A, \Gamma_2 \vdash B{:}C$. □

Lemma 21 (*Terms*)

$$M \in Term(\zeta) \Leftrightarrow M \in CONS \vee \exists \Gamma \exists C . \Gamma \vdash M : C.$$

Proof. According to the definition of $Term(\zeta)$,

$$M \in Term(\zeta) \Leftrightarrow \exists \Gamma \exists C . [\Gamma \vdash M : C \vee \Gamma \vdash C : M].$$

If $\Gamma \vdash C : M$, we find with the stripping lemma 19 (vi) that $M \equiv c \in CONS$ or $\Gamma \vdash M : s$ with $s \in SORT \subseteq CONS$. □

Lemma 22 (*Subject reduction*)
For $\Gamma, \Gamma' \in Context(\zeta)$, $B, B', C \in Term(\zeta)$ *and* $\Gamma \vdash B : C$

(i) $B \twoheadrightarrow B' \Rightarrow \Gamma \vdash B' : C$,
(ii) $\Gamma \twoheadrightarrow \Gamma' \Rightarrow \Gamma' \vdash B : C$.

Proof. By simultaneous induction on the length of the derivation of $\Gamma \vdash B : C$, distinguishing cases according to the last applied inference rule. We prove the lemma for one step reductions, so $B \to B'$ or $\Gamma \to \Gamma'$.

Proof of (*i*). If the last rule is (start), there is no redex in B.

If the last rule is (weakening), (conversion), (Π-rule) or (λ-rule), we are done by the induction hypothesis. (For (Π-rule) and (λ-rule), use also the induction hypothesis on (ii).)

If the last rule is (application), we distinguish subcases $B \equiv B_1 B_2 \to B_1' B_2' \equiv B'$ (the reduction taking place inside B_1 or B_2), and $B \equiv (\lambda x : A_1 . A_2) B_2 \to A_2[x := B_2] \equiv B'$. We treat the last case. Let $B \equiv (\lambda x : A_1 . A_2) B_2 \to A_2[x := B_2]$ and

$$\frac{\Gamma \vdash \lambda x : A_1 . A_2 : \Pi x : C_1 . C_2 \quad \Gamma \vdash B_2 : C_1}{\Gamma \vdash (\lambda x : A_1 . A_2) B_2 : C_2[x := B_2]}.$$

Apply the stripping lemma to $\Gamma \vdash \lambda x : A_1 . A_2 : \Pi x : C_1 . C_2 : s$ to find

$$\Gamma \vdash A_1 : s_1 \tag{1}$$

$$\Gamma, x : A_1 \vdash A_2 : C_2' \quad \text{for some } C_2' = C_2 \tag{2}$$

$$\Gamma, x : C_1 \vdash C_2 : s_2 \tag{3}$$

and

$$A_1 = C_1.$$

Further we have

$$\Gamma \vdash B_2 : C_1 \tag{4}$$

Applying (conversion) to (1) and (4), we find that

$$\Gamma \vdash B_2 : A_1. \tag{5}$$

With the substitution lemma we conclude from (2) and (5) that

$$\Gamma \vdash A_2[x := B_2] : C_2'[x := B_2]. \tag{6}$$

Again with the substitution lemma we conclude from (3) and (4) that

$$\Gamma \vdash C_2[x := B_2] : s_2 \tag{7}$$

and to (6) and (7) one can apply (conversion) to obtain $\Gamma \vdash A_2[x:=B_2]:C_2[x:=B_2]$, which was to be proved.

Proof of (ii). If the last rule is (Π-rule), (λ-rule), (application) or (conversion), we are immediately done by induction hypothesis.

If the last rule is (start) or (weakening), we distinguish cases according to whether or not the reduction took place in the last declaration of the context. If this is not so, we are done by induction hypothesis. If the reduction did take place in the last declaration of the context, apply the induction hypothesis on (i) and (conversion) (in case (start) was the last rule) and we are done. □

Corollary 23
For $\Gamma \in Context(\zeta)$, $B, C, C' \in Term(\Gamma)$

$$\Gamma \vdash B:C \;\&\; C \twoheadrightarrow C' \Rightarrow \Gamma \vdash B:C'.$$

Proof. This follows immediately from stripping (vi), applying subject reduction to the term C and (conversion). □

Lemma 24 (*Uniqueness of assignment for functional* GTSs)
Let ζ *be a functional* GTS. *For* $\Gamma \in Context(\zeta)$, $B, C, C' \in Term(\zeta)$

$$\Gamma \vdash B:C \;\&\; \Gamma \vdash B:C' \Rightarrow C = C'.$$

Proof. By induction on the structure of the pseudoterm B. As the proof is easy, we only treat one case, for $B \equiv \lambda x:B_1.B_2$. Then, $\Gamma, x:B_1 \vdash B_2:C_2$ and $\Gamma, x:B_1 \vdash B_2:C_2'$ for some terms C_2 and C_2' and $C = \Pi x:B_1.C_2$, $C' = \Pi x:B_1.C_2'$. By induction hypothesis $C_2' = C_2$. Hence $C = \Pi x:B_1.C_2 = C'$. □

Lemma 25 (*Strengthening modulo reduction for functional* GTSs)
For $\Gamma_1, x:A, \Gamma_2 \in Context(\zeta)$, $B, C \in Term(\zeta)$

$$\Gamma_1, x:A, \Gamma_2 \vdash B:C,\ x \notin FV(\Gamma_2) \cup FV(B)$$
$$\Rightarrow \exists C' \in Term(\zeta).[C \twoheadrightarrow C' \;\&\; \Gamma_1, \Gamma_2 \vdash B:C'].$$

Proof. By induction on the length of the derivation of $\Gamma_1, x:A, \Gamma_2 \vdash B:C$, distinguishing cases according to the last applied rule.

If the last rule is (start) or (Π-rule), we are done by the induction hypothesis and one application of the rule.

If the last rule is (weakening), we are done by the induction hypothesis (distinguish between $\Gamma_2 = \varnothing$ and $\Gamma_2 \neq \varnothing$.)

If the last rule is (conversion), we are done by induction hypothesis, Church–Rosser property and Corollary 23.

If $B \equiv \lambda z:B_1.B_2$, $C \equiv \Pi z:B_1.C_2$ and the last applied rule is (λ-rule), then $\Gamma_1, x:A, \Gamma_2 \vdash B_1:s_1$ and $\Gamma_1, x:A, \Gamma_2, z:B_1 \vdash B_2:C_2:s_2$. By induction hypothesis $\Gamma_1, \Gamma_2 \vdash B_1:s_1$ and $\Gamma_1, \Gamma_2, z:B_1 \vdash B_2:C_2'$ for some C_2' with $C_2 \twoheadrightarrow C_2'$.

Now, $\Gamma_1, \Gamma_2, z:B_1 \vdash C_2':s_2$ (By stripping (vi): $C_2' \equiv c \in \mathcal{C}ons$ and $C_2':s_2$ is an axiom or $\Gamma_1, \Gamma_2, z:B_1 \vdash C_2':s \in \mathcal{S}ort$ and $s \equiv s_2$ by uniqueness of assignment.) With one

application of (λ-rule), we conclude that $\Gamma_1, \Gamma_2 \vdash \lambda z{:}B_1 . B_2 : \Pi z{:}B_1 . C_2'$, where $C \twoheadrightarrow \Pi z{:}B_1 . C_2'$.

If $B \equiv B_1 B_2$, $C \equiv C_2[x := B_2]$, and the last applied rule is (application), then $\Gamma_1, x{:}A, \Gamma_2 \vdash B_1 : \Pi z{:}C_1 . C_2$ and $\Gamma_1, x{:}A, \Gamma_2 \vdash B_2 : C_1$. By induction hypothesis $\Gamma_1, \Gamma_2 \vdash B_1 : \Pi z{:}C_1' . C_2'$ and $\Gamma_1, \Gamma_2 \vdash B_2 : C_1''$, for some C_1', C_1'', C_2' with $C_1 \twoheadrightarrow C_1'$, $C_1 \twoheadrightarrow C_1''$ and $C_2 \twoheadrightarrow C_2'$. Take C_1''' such that $C_1 \twoheadrightarrow C_1'''$ and $C_1 \twoheadrightarrow C_1'''$. With Corollary 23 and (application), we find $\Gamma_1, \Gamma_2 \vdash B_1 B_2 : C_2'[x := B_2]$ with $C \twoheadrightarrow C_2'[x := B_2]$.

Corollary 26 (*Strengthening for functional* GTSs)
Let ζ *be a functional* GTS. *For* $\Gamma_1, x{:}A, \Gamma_2 \in \mathit{Context}(\zeta)$, $B, C \in \mathit{Term}(\zeta)$

$$\Gamma_1, x{:}A, \Gamma_2 \vdash B : C,\ x \notin FV(\Gamma_2) \cup FV(B{:}C)$$
$$\Rightarrow \Gamma_1, \Gamma_2 \vdash B : C.$$

Proof. By Lemma 25, we know that $\Gamma_1, \Gamma_2 \vdash B : C'$, with $C \twoheadrightarrow C'$. By stripping (vi), there are two possibilities, $C \in \mathit{CONS}$ or $\Gamma_1, x{:}A, \Gamma_2 \vdash C : s \in \mathit{SORT}$. In the first case $C' \equiv C$, and we are done. In the second case, $\Gamma_1, \Gamma_2 \vdash C : s$ by Lemma 25 so with one application of (conversion): $\Gamma_1, \Gamma_2 \vdash B : C$. □

The idea of proving the previous corollary using Lemma 25 is due to Luo (1988), who has used it in a proof of strong normalization for an extended calculus of constructions.

Definition 27
Let $\Gamma \in \mathit{Context}(\zeta)$, $\Pi x{:}B_1 . B_2$, $\lambda x{:}B_1 . B_2$, $B_1 B_2 \in \mathit{Term}(\zeta)$.

(i) $\Pi x{:}B_1 . B_2$ *is formed by* (s_1, s_2, s_3) *in* Γ iff
$(s_1, s_2, s_3) \in \mathit{RULE}$ &
$\Gamma \vdash B_1 : s_1$ &
$\Gamma, x{:}B_1 \vdash B_2 : s_2$.

(ii) $\lambda x{:}B_1 . B_2$ *is formed by* (s_1, s_2, s_3) *in* Γ iff
$\exists C_1, C_2 \in \mathit{Term}(\zeta)$. $(s_1, s_2, s_3) \in \mathit{RULE}$ &
$\Gamma \vdash \lambda x{:}B_1 . B_2 : \Pi x{:}C_1 . C_2$ &
$\Pi x{:}C_1 . C_2$ *is formed by* (s_1, s_2, s_3) in Γ.

(iii) $B_1 B_2$ *is formed by* (s_1, s_2, s_3) *in* Γ iff
$\exists C_1, C_2 \in \mathit{Term}(\zeta)$. $(s_1, s_2, s_3) \in \mathit{RULE}$ &
$\Gamma \vdash B_1 : \Pi x{:}C_1 . C_2$ &
$\Pi x{:}C_1 . C_2$ is formed by (s_1, s_2, s_3) in Γ.

Remark 28. By this definition, all composed terms are formed by a certain rule. For the first two cases, this follows immediately from the stripping-lemma. For the third case, this follows from the stripping-lemma 19 (v) and (vi) and case (i).

Lemma 29 (*Uniqueness of formation for functional* GTSs)
Let ζ *be a functional* GTS. *For* $\Gamma \in \mathit{Context}(\zeta)$, $B \in \mathit{Term}(\zeta)$, B *composed*

$$B \text{ formed by } (s_1, s_2, s_3) \text{ in } \Gamma \ \&\ B \text{ formed by } (s_1', s_2', s_3') \text{ in } \Gamma$$
$$\Rightarrow s_1 \equiv s_1', s_2 \equiv s_2', s_3 \equiv s_3'.$$

Proof. We are done if we prove the following property. If $\Pi x{:}B_1.B_2$ formed by (s_1, s_2, s_3) in Γ, $\Pi x{:}B_1'.B_2'$ formed by (s_1', s_2', s_3') in Γ and $\Pi x{:}B_1.B_2 \twoheadrightarrow \Pi x{:}B_1'.B_2'$, then $s_1 \equiv s_1'$, $s_2 \equiv s_2'$ and $s_3 \equiv s_3'$.

For the cases $M \equiv \lambda x{:}B_1.B_2$ and $M \equiv B_1 B_2$, the proof then follows by the uniqueness of assignment. Namely, if $\Gamma \vdash \lambda x{:}B_1.B_2 : \Pi x{:}C_1.C_2$ and $\Gamma \vdash \lambda x{:}B_1.B_2 : \Pi x{:}C_1'.C_2'$, respectively $\Gamma \vdash B_1 : \Pi x{:}C_1.C_2$ and $\Gamma \vdash B_1 : \Pi x{:}C_1'.C_2'$, with $\Pi x{:}C_1.C_2$ formed by (s_1, s_2, s_3) in Γ and $\Pi x{:}C_1'.C_2'$ formed by (s_1', s_2', s_3') in Γ, then $\Pi x{:}C_1.C_2 = \Pi x{:}C_1'.C_2'$. So, take (with Church–Rosser) $\Pi x{:}C_1''.C_2''$, such that $\Pi x{:}C_1'.C_2' \twoheadrightarrow \Pi x{:}C_1''.C_2''$ and $\Pi x{:}C_1.C_2 \twoheadrightarrow \Pi x{:}C_1''.C_2''$. Then $\Pi x{:}C_1''.C_2''$ is formed by (s_1, s_2, s_3) and (s_1', s_2', s_3') in Γ, so $s_1 \equiv s_1'$, $s_2 \equiv s_2'$ and $s_3 \equiv s_3'$.

The proof of the property runs as follows. Let $\Pi x{:}B_1.B_2$ be formed by (s_1, s_2, s_3) in Γ, $\Pi x{:}B_1'.B_2'$ formed by (s_1', s_2', s_3') in Γ and $\Pi x{:}B_1.B_2 \twoheadrightarrow \Pi x{:}B_1'.B_2'$. Then $\Gamma \vdash B_1 : s_1$, $\Gamma, x{:}B_1 \vdash B_2 : s_2$ and $\Gamma \vdash B_1' : s_1'$, $\Gamma, x{:}B_1' \vdash B_2' : s_2'$. By subject reduction and uniqueness of assignment $s_1 \equiv s_1'$ and $s_2 \equiv s_2'$. So $s_3 \equiv s_3'$. □

Lemma 30 (*Classification for injective* GTSs)
Let ζ *be an injective* GTS. *For* $s, s' \in \mathcal{SORT}$, $s \not\equiv s'$,

(i) $s\text{-}Term(\zeta) \cap s'\text{-}Term(\zeta) = \varnothing$
(ii) $s\text{-}Elt(\zeta) \cap s'\text{-}Elt(\zeta) = \varnothing$.

Proof. The proof of (i) and (ii) is simultaneous, by induction on the structure of pseudoterms. We only treat the induction step for variables and for terms B of the form $B_1 B_2$. For the other induction steps, the statement follows immediately from the induction hypothesis using the injectivity properties.

Let $\Gamma \vdash x : B : s$ and $\Gamma' \vdash x : B' : s'$. Then $x{:}A \in \Gamma$ for certain A with $A = B$, and $\Gamma \vdash A : s_0$ and $x{:}A' \in \Gamma'$ for certain A' with $A' = B'$ and $\Gamma' \vdash A' : s_0$. (Where s_0 is the sort for which $x \in \mathcal{VAR}^{s_0}$). By Church–Rosser, subject reduction and uniqueness of assignment $s_0 \equiv s$ and $s_0 \equiv s'$. For $\Gamma \vdash x : s$ and $\Gamma' \vdash x : s'$ the argument is similar.

Let $\Gamma \vdash B_1 B_2 : C$ and $\Gamma' \vdash B_1 B_2 : C'$ Then $\Gamma \vdash B_1 : \Pi x{:}C_1.C_2 : s_3$, $\Gamma \vdash B_2 : C_1 : s_1$, $\Gamma, x{:}C_1 \vdash C_2 : s_2$ and $\Gamma' \vdash B_1 : \Pi x{:}C_1'.C_2' : s_3'$, $\Gamma' \vdash B_2 : C_1' : s_1'$, $\Gamma', x{:}C_1' \vdash C_2' : s_2'$, for certain terms C_1, C_2, C_1', C_2' and (s_1, s_2, s_3), $(s_1', s_2', s_3') \in \mathcal{RULE}$ with $C_2[x := B_2] = C$ and $C_2'[x := B_2] = C'$.

By induction hypothesis $s_1 \equiv s_1'$, $s_3 \equiv s_3'$ and so $s_2 \equiv s_2'$.

By substitution $\Gamma \vdash C_2[x := B_2] : s_2$ and $\Gamma' \vdash C_2'[x := B_2] : s_2'$.

If now $\Gamma \vdash C : s$ and $\Gamma' \vdash C' : s'$, then by Church–Rosser, subject reduction and uniqueness of assignment, $s \equiv s_2$ and $s' \equiv s_2'$, so $s \equiv s'$.

If $C \equiv s$ and $C \equiv s'$, then by subject reduction and uniqueness of assignment $s : s_2$ and $s' : s_2$ are axioms, so $s \equiv s'$. □

The previous lemma motivates terminologies such as 't is a *s-Term*', or 't is a *s-Elt*'. These notions are not ambiguous in an injective GTS. That the lemma does not hold in general for systems which are not injective is shown by the following example.

Example 31
Let ζ be given by

$$\mathcal{SORT} = \{*, *', \nabla, \nabla', \Delta, \Delta', \Box\},$$

$$\mathcal{AXIOM} = \{*:*', \nabla:\nabla', \Delta:\Delta'\},$$

$$\mathcal{RULE} = \{(*, \nabla, \Box), (*, \Delta, \Box)\},$$

$$\alpha \in \mathcal{VAR}^{*'}, \beta \in \mathcal{VAR}^{\Delta'}, \gamma \in \mathcal{VAR}^{\nabla'}, x \in \mathcal{VAR}^{*}, f \in \mathcal{VAR}^{\Box},$$

then ζ is not injective and

$$\alpha:*, \beta:\Delta, f:\alpha \to \beta, x:\alpha \vdash fx:\beta:\Delta,$$

$$\alpha:*, \gamma:\nabla, f:\alpha \to \gamma, x:\alpha \vdash fx:\gamma:\nabla,$$

so $fx \in \Delta\text{-}Elt(\zeta)$ and $fx \in \nabla\text{-}Elt(\zeta)$.

Corollary 32 (*Uniqueness of formation for injective* GTSs)
Let ζ *be an injective* GTS. *For* $\Gamma, \Gamma' \in Context(\zeta)$, $M \in Term(\zeta)$, *M composed*

$$M \text{ formed by } (s_1, s_2, s_3) \text{ in } \Gamma \;\&\; M \text{ formed by } (s_1', s_2', s_3') \text{ in } \Gamma'$$

$$\Rightarrow s_1 \equiv s_1', s_2 \equiv s_2', s_3 \equiv s_3'.$$

Proof. Using the classification lemma for injective ζ, the proof runs just like the proof of the uniqueness of formation lemma for functional ζ. □

This corollary allows us to use the terminology 'formed by' without mentioning the context Γ, in case of an injective GTS. That the corollary is not true for just functional GTS is shown by the following example.

Example 33
Let the system ζ be given by

$$\mathcal{SORT} = \{s_1, s_2, s_2', s_3\},$$

$$\mathcal{AXIOM} = \{s_1:s_3, s_2:s_3, s_2':s_3\},$$

$$\mathcal{RULE} = \{(s_1, s_2, s_3), (s_1, s_2', s_3)\},$$

$$y, z \in \mathcal{VAR}^{s_3}, x \in \mathcal{VAR}^{s_1}.$$

Then $\Pi x:y.z$ is formed by (s_1, s_2, s_3) in the context $y:s_1, z:s_2$,

$\Pi x:y.z$ is formed by (s_1, s_2', s_3) in the context $y:s_1, z:s_2'$.

3 Barendregt's cube of typed lambda calculi

Barendregt's cube consists of a coherent collection of eight type systems, each one corresponding with a vertex of a cube such that there is an inclusion relation along the edges of the cube. The systems of the cube will be defined by giving for each of

them the sets *SORT*, *CONS*, *AXIOM* and *RULE* as in the definition of Generalized Type System in 1–12.

Definition 34

The *Barendregt's cube of typed lambda calculi* is the set of type systems $\lambda\!\rightarrow$, $\lambda 2$, λP, $\lambda\bar{\omega}$, $\lambda P2$, $\lambda\omega$, $\lambda P\bar{\omega}$ and $\lambda P\omega$, defined by

(i) $SORT = \{*, \Box\}$ for all systems,

(ii) $CONS = \{*, \Box\}$ for all systems,

(iii) $AXIOM = \{*:\Box\}$ for all systems,

(iv)
$$\begin{aligned}
RULE(\lambda\!\rightarrow) &= \{(*,*)\},\\
RULE(\lambda 2) &= \{(*,*), (\Box,*)\},\\
RULE(\lambda P) &= \{(*,*), (*,\Box)\},\\
RULE(\lambda\bar{\omega}) &= \{(*,*), (\Box,\Box)\},\\
RULE(\lambda P2) &= \{(*,*), (\Box,*), (*,\Box)\},\\
RULE(\lambda\omega) &= \{(*,*), (\Box,*), (\Box,\Box)\},\\
RULE(\lambda P\bar{\omega}) &= \{(*,*), (*,\Box), (\Box,\Box)\},\\
RULE(\lambda P\omega) &= \{(*,*), (\Box,*), (*,\Box), (\Box,\Box)\}.
\end{aligned}$$

Notice that all eight are injective (Definition 2).

In 37, we give some examples of how the rules work in practice, and which derivations they allow. For this we also refer to Barendregt (1989).

Figure 1 is referred to as *Barendregt's cube*; it is meant to standardize the coordinates of the eight systems, and also notions such as 'upper plane', 'right plane' and 'front plane'.

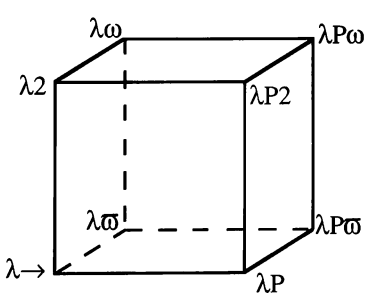

Fig. 1. Barendregt's cube.

Convention 35

To distinguish VAR^* from $VAR^\Box$, the first variables are denoted by Latin characters, so $x, y, z, p, q, \ldots \in VAR^*$, and the second ones by Greek characters or Latin capitals, so $\alpha, \beta, \gamma, \delta, \ldots A, B, C, P, Q, R \ldots \in VAR^\Box$. As metavariable ranging over VAR, we shall use u or v.

Notation 36

Expressions of the form $\Pi x{:}A.B$, will, as usual, be denoted by $A \rightarrow B$ if $x \notin FV(B)$.

Most of the right systems are well known; $\lambda\!\rightarrow$ is a variant of Church's simply typed lambda calculus (Church, 1940), λP and $\lambda P2$ are variants of the AUTOMATH system AUT-QE, defined by de Bruijn (1980), whereas λP is also known under the name LF, studied by Harper *et al.* (1987). $\lambda 2$ and $\lambda\omega$ are the systems F and Fω, defined by Girard (1972), and $\lambda P\omega$ is the calculus of constructions, defined by Coquand and Huet (1985, 1988).

Notice that in the systems of the cube, one does not have 0 as a basic type; $0 \notin \mathit{CONS}$, $0:*$ is not an axiom. The reason for this is that by the axiom $(*:\Box)$ it is possible to introduce any number of basic types in the context by extending the context with a type variable. Inside the context this variable behaves just like a constant. A consequence is that in the lower plane almost nothing is derivable from any empty context, e.g., in $\lambda\!\rightarrow$ only $*:\Box$ is derivable in an empty context. Church's λ^τ in the terminology of GTS is the system with $\mathit{SORT} = \{*\}$, $\mathit{CONS} = \{*, \Box\}$, $\mathit{AXIOM} = \{0:*\}$, $\mathit{RULE} = \{(*,*)\}$. The system $\lambda\!\rightarrow$ is a variant of λ^τ in the sense that $\lambda\!\rightarrow$ is λ^τ with $\mathit{CONS} = \mathit{VAR}^\Box$. (Note that $\Gamma \vdash_{\lambda\rightarrow} t:\sigma \Leftrightarrow \Gamma' \vdash_{\lambda^\tau} t:\sigma$, where Γ' is Γ without all assignments of the form $\alpha:*$.)

Besides the computational aspects of the eight type systems, an important feature is the relation with (intuitionistic) logic. This follows the Curry–de Bruijn–Howard notion of 'formulae-as-types', as discussed by Howard (1980), and leads to an interpretation of various logical systems in typed lambda calculi. With that notion, formulae become types and proofs become typed lambda terms such that a provable formula becomes a nonempty type. The situation for Barendregt's cube is that one can define a cube of logical systems that corresponds with the cube of type systems in the 'formulae-as-types' sense, as studied by Geuvers (1988).

For now, we give some examples of what can be derived in the several type systems of the cube. The examples might best be understood by the logical intuition behind them, i.e., reading '$\forall$' for Π, 'FORM' (the collection of all formulae) for $*$ and interpreting $A \rightarrow *$ as the set of predicates on A (and likewise, $A \rightarrow A \rightarrow *$ as the set of binary relations on A.) After the examples a few lemmas will be proved for systems in the cube.

Examples 37

(i) A derivation in $\lambda\!\rightarrow$, given in full detail, with the numbers of the applied rules

$$\dfrac{\dfrac{\vdash *:\Box}{\alpha:* \vdash \alpha:*}(1) \quad \dfrac{\dfrac{\vdash *:\Box}{\alpha:* \vdash \alpha:*}(1)}{\alpha:*, x:\alpha \vdash \alpha:*}(2) \quad \dfrac{\dfrac{\vdash *:\Box}{\alpha:* \vdash \alpha:*}(1)}{\alpha:*, x:\alpha \vdash x:\alpha}(1)}{\alpha:* \vdash \lambda x:\alpha.x:\alpha\rightarrow\alpha}(4)$$

(ii) In $\lambda 2$ one can derive

$$\vdash \lambda\alpha:*.\lambda\beta:*.\lambda x:\alpha.\lambda y:\beta.x \;:\; \Pi\alpha:*.\Pi\beta:*.\alpha\rightarrow\beta\rightarrow\alpha,$$

$$\vdash \Pi\alpha:*.\alpha \;:\; *.$$

〈The type $\Pi\alpha:*.\alpha$ will also be denoted by $\bot$, a term of type $\Pi\alpha:*.\alpha$ producing an inhabitant of any type (as in logic, where a proof of $\bot$ produces a proof of any

formula). The type $\bot$ can be formed in systems that include the rule $(\Box, *)$, i.e., the systems of the upper plane.〉

Define in $\lambda 2$ $A\&B := \Pi\gamma:*.(A\to B\to\gamma)\to\gamma$, $A\vee B := \Pi\gamma:*.(A\to\gamma)\to(B\to\gamma)\to\gamma$. Then one can derive

$$A:*, B:* \qquad \vdash A\&B : *,$$

$$A:*, B:*, z:A\&B \qquad \vdash z\,A\,(\lambda x:A.\lambda y:B.x) : A,$$

$$A:*, B:*, z:A\&B \qquad \vdash z\,B\,(\lambda x:A.\lambda y:B.y) : B,$$

$$A:*, B:*, x:A, y:B \qquad \vdash \lambda\gamma:*.\lambda f:A\to B\to\gamma.f\,x\,y : A\&B,$$

and

$$A:*, B:* \qquad \vdash A\vee B : *$$

$$A:*, B:*, x:A \qquad \vdash \lambda\gamma:*.\lambda f:A\to\gamma.\lambda g:B\to\gamma.f\,x : A\vee B,$$

$$A:*, B:*, y:B \qquad \vdash \lambda\gamma:*.\lambda f:A\to\gamma.\lambda g:B\to\gamma.g\,y : A\vee B,$$

$$A:*, B:*, \gamma:*, z:\mathrm{A}\vee B, x:A\to\gamma, y:B\to\gamma \vdash z\,\gamma\,x\,y : \gamma,$$

〈Notice the connection with the definitions of & and $\vee$ in second order logic.〉

(iii) In λP one can derive

$$A:*, P:A\to *, Q:A\to *, R:A\to * \vdash$$
$$\lambda x:A.\lambda f:P\,x\to Q\,x.\lambda g:Q\,x\to R\,x.\lambda y:P\,x.g\,(f\,y) :$$
$$\Pi x:A.(P\,x\to Q\,x)\to(Q\,x\to R\,x)\to(P\,x\to R\,x).$$

〈This example expresses the transitivity of inclusion〉

(iv) In $\lambda\bar{\omega}$ one can derive

$$\vdash \lambda\alpha:*.\alpha\to\alpha : *\to *.$$

$$A:*, F:*\to * \vdash \lambda p:(F\,A\to F\,A\to A).\lambda x:F\,A.p\,x\,x : (F\,A\to F\,A\to A)\to F\,A\to A.$$

(v) In λP2 one can derive

$$A:*, R:A\to A\to * \vdash$$
$$\lambda p:(\Pi x:A.\Pi y:A.R\,x\,y\to R\,y\,x\to\bot).\lambda x:A.\lambda q:R\,x\,x.p\,x\,x\,q\,q :$$
$$(\Pi x:A.\Pi y:A.(R\,x\,y\to R\,y\,x\to\bot))\to(\Pi x:A.R\,x\,x\to\bot).$$

〈This example is taken from Barendregt (1989). The typed lambda term proves the fact that an asymmetric relation is irreflexive.〉

Define in λP2 $\exists x:A.B := \Pi\gamma:*.((\Pi x:A.B\to\gamma)\to\gamma)$. Then one can derive

$$A:*, P:A\to * \vdash \exists x:A.P\,x : *$$

and

$$A:*, P:A\to *, a:A \vdash$$
$$\lambda t:\Pi x:A.P\,x.\lambda\gamma:*.\lambda f:\Pi x:A.P\,x\to\gamma.f\,a\,(t\,a) : \Pi x:A.P\,x\to\exists x:A.P\,x,$$

〈which expresses the fact that (in logic) for a nonempty A, $\vdash \forall x\in A.\phi\to\exists x\in A.\phi$.〉

(vi) In λω one can derive

$$\vdash \lambda q{:}(\Pi\delta{:}*\to *.\,\Pi\alpha{:}*.\,\delta\alpha\to\alpha).\,\lambda\beta{:}*.\,q(\lambda\gamma{:}*.\,\gamma\to\gamma)\,\beta(\lambda x{:}\beta.\,x) :$$
$$(\Pi\delta{:}*\to *.\,\Pi\alpha{:}*.\,\delta\alpha\to\alpha)\to\bot.$$

Define in λω

$$\&:=\lambda\alpha{:}*.\,\lambda\beta{:}*.\,\Pi\gamma{:}*.\,(\alpha\to\beta\to\gamma)\to\gamma : *\to *\to *,$$
$$\vee:=\lambda\alpha{:}*.\,\lambda\beta{:}*.\,\Pi\gamma{:}*.\,(\alpha\to\gamma)\to(\beta\to\gamma)\to\gamma : *\to *\to *.$$

⟨$\&AB = A\&B$, $\vee AB = A\vee B$, with & and ∨ as in (ii); in λω one can define the logical connectives & and ∨ in a polymorphic way.⟩

In λPω̄ one can derive

$$\vdash \lambda\alpha{:}*.\,\lambda\delta{:}\alpha\to\alpha\to *.\,\lambda x{:}\alpha.\,\delta\,x\,x : \Pi\alpha{:}*.\,(\alpha\to\alpha\to *)\to\alpha\to *.$$

⟨In this example $\Pi\alpha{:}*.\,(\alpha\to\alpha\to *)\to\alpha\to *$ is not a type, but a kind; the constructor $\lambda\alpha{:}*.\,\lambda\delta{:}\alpha\to\alpha\to *.\,\lambda x{:}\alpha.\ \delta\,x\,x$ produces for each binary predicate its diagonalization.⟩

(viii) In λPω one can derive

$$\vdash \lambda\alpha{:}*.\,\lambda R{:}\alpha\to\alpha\to *.\,\lambda p{:}(\Pi x{:}A.\,\Pi y{:}A.\,Rxy\to Ryx\to\bot).\,\lambda x{:}A.\,\lambda q{:}Rxx.\,pxxqq$$
$$:\Pi\alpha{:}*.\,\Pi R{:}\alpha\to\alpha\to *.\,(\Pi x{:}A.\,\Pi y{:}A.\,(Rxy\to Ryx\to\bot))\to(\Pi x{:}A.\,Rxx\to\bot).$$

⟨This is a higher order version of the example for λP2.⟩

Define in λPω

$$\exists:=\lambda\alpha{:}*.\,\lambda P{:}\alpha\to *.\,\exists x{:}\alpha.\,Px : \Pi\alpha{:}*.\,(\alpha\to *)\to *,$$

with ∃ as in (v).

Then one can derive

$$\alpha{:}*, P{:}\alpha\to *\vdash \exists\,\alpha\,P{:}*$$ and one has $\exists\,\alpha\,P = \exists x{:}\alpha.\,Px$.

Define in λPω

$$\neg:=\lambda P{:}\alpha\to *.\,\lambda x{:}\alpha.\,Px\to\bot : (\alpha\to *)\to\alpha\to *.$$

Then one can derive

$$\alpha{:}*, P{:}\alpha\to *\vdash \neg P : \alpha\to *$$ and

$$\alpha{:}*, P{:}\alpha\to *\vdash t : \Pi x{:}\alpha.\,((Px\to\bot)\leftrightarrow\neg Px)$$ for some term t.

(with ↔ interpreted as $(\ldots\to\ldots)\,\&\,(\ldots\leftarrow\ldots)$).

Definition 38

For λ_ a system of the cube, $\Gamma\in$ *Context* (λ_), the sets Γ-*kind* (λ_), Γ-*constructor*(λ_), Γ-*type*(λ_) *and* Γ-*object*(λ_) are defined by

(i) Γ-*kind*(λ_) = Γ-□-*Term*(λ_),
(ii) Γ-*constructor*(λ_) = Γ-□-*Elt*(λ_),

(iii) $\Gamma\text{-}type(\lambda\_) = \Gamma\text{-}*\text{-}Term(\lambda\_)$,
(vi) $\Gamma\text{-}object(\lambda\_) = \Gamma\text{-}*\text{-}Elt(\lambda\_)$.

The sets $kind(\lambda\_)$, $constructor(\lambda\_)$, $type(\lambda\_)$ *and* $object(\lambda\_)$ are defined by taking the union over all $\Gamma \in Context(\lambda\_)$. (For the definitions of $\ldots\text{-}Term(..)$ and $\ldots\text{-}Elt(\ldots)$ see Definition 13).

Definition 39
For $\lambda\_$ a system of the cube, the *erasure* $||: object\ (\lambda\_) \to \Lambda$ is defined by

(i) $|x| = x$, *if* $x \in V_{AR}{}^*$,
(ii) $|\lambda x{:}A.q| = \lambda x.|q|$, if $\lambda x{:}A.q$ is formed by $(*,*)$,
(iii) $|\lambda \alpha{:}A.q| = |q|$, if $\lambda\alpha{:}A.q$ is formed by $(\Box,*)$,
(iv) $|p\,q| = |p|\,|q|$, if $p\,q$ is formed by $(*,*)$,
(v) $|p\,Q| = |p|$, if $p\,Q$ is formed by $(\Box,*)$.

Remarks 40
(i) By definition $\Gamma\text{-}type(\lambda\_) \subseteq \Gamma\text{-}constructor(\lambda\_)$, and further by the classification lemma for injective GTS (30), $constructor(\lambda\_) \cap object(\lambda\_) = \varnothing$ and $kind(\lambda\_) \cap type(\lambda\_) = \varnothing$. It is also easy to see that the two intersections $kind(\lambda\_) \cap constructor(\lambda\_)$ and $kind(\lambda\_) \cap object(\lambda\_)$ are empty.

(ii) $M \in kind(\lambda\_) \Rightarrow M \equiv * \vee M \equiv \Pi u{:}Q.N$,
$M \in constructor(\lambda\_) \Rightarrow M \notin SORT \;\&\; M \notin V_{AR}{}^*$,
$M \in type(\lambda\_) \Rightarrow M \equiv \alpha \in V_{AR}{}^{\Box} \vee M \equiv QN \vee M \equiv \Pi u{:}Q.N$,
$M \in object(\lambda\_) \Rightarrow M \equiv x \in V_{AR}{}^* \vee M \equiv QN \vee M \equiv \lambda u{:}Q.N$.

This can easily be seen by noticing that in $\lambda\_$, if $M = s \in SORT$, then $M \equiv s$, and examining the inference rules.

Lemma 41 (*Nonemptyness of kinds*)
(i) *For* $\lambda\_$ *a system of the bottom of the cube,* $\Gamma \in Context(\lambda\_)$ *and* $\Gamma \vdash_{\lambda\_} A{:}*$ *for some* A

$$\Gamma \vdash_{\lambda\_} k{:}\Box \Rightarrow \exists t.\,\Gamma \vdash_{\lambda\_} t{:}k.$$

(ii) *For* λ_2 *a system of the top of the cube,* $\Gamma \in Context(\lambda\_2)$,

$$\Gamma \vdash_{\lambda\_2} k{:}\Box \Rightarrow \exists t.\,\Gamma \vdash_{\lambda\_2} t{:}k.$$

Proof. By induction on the structure of the pseudoterm k.

$-\ k \equiv *$ For λ_2, $\Gamma \vdash_{\lambda\_2} \Pi\alpha{:}*.\alpha{:}*$, for instance.
$-\ k \equiv \Pi x{:}M.N$ By the stripping-lemma, $\Gamma \vdash M{:}s$ and $\Gamma, x{:}M \vdash N{:}\Box$, for $s \in \{*, \Box\}$, so by induction hypothesis $\Gamma, x{:}M \vdash t{:}N$ for some t and with rule $(s,\Box)$ $\Gamma \vdash \lambda x{:}M.t : \Pi x{:}M.N$. □

Definition 42
(I) K is the set, inductively defined by

(i) $* \in K$,
(ii) $k_1, k_2 \in K \Rightarrow k_1 \to k_2 \in K$.

(II) T is the set, inductively defined by

(i) $VAR^{\Box} \subseteq T$,
(ii) $\sigma, \tau \in T \Rightarrow \sigma \to \tau \in T$.

Lemma 43
(1) (Structure of kinds in $\lambda\omega$) *For* $\Gamma \in Context(\lambda\omega)$,

$$\Gamma \vdash_{\lambda\omega} k : \Box \Rightarrow k \in K \;\&\; \vdash_{\lambda\omega} k : \Box.$$

(2) (Structure of constructors in $\lambda\to$) *For* $\Gamma \in Context(\lambda\to)$,

$$\Gamma \vdash_{\lambda\to} t : k : \Box \Rightarrow t \in T \;\&\; \Gamma \vdash_{\lambda\to} t : *.$$

Proof. We only prove (1), as the proof of (2) is somewhat the same. The argument runs by induction on the structure of the pseudoterm k.

$-\ k \equiv *$, done.
$-\ k \equiv \Pi x{:}M.N$, then k is formed by rule $(\Box, \Box)$, so $\Gamma \vdash_{\lambda\omega} M : \Box$ and $\Gamma, x{:}M \vdash_{\lambda\omega} N : \Box$.

By induction hypothesis, $M \in K$, $N \in K$, $\vdash_{\lambda\omega} M : \Box$, and $\vdash_{\lambda\omega} N : \Box$. With one application of rule $(\Box, \Box)$ we obtain $\vdash_{\lambda\omega} M \to N : \Box$ and $k \equiv M \to N$. □

Lemma 44
For $A, t \in Term\ (\lambda\omega)$

$$A \in constructor\,(\lambda\omega)\ \&\ t\ subexpression\ of\ A$$
$$\Rightarrow t \in kind\,(\lambda\omega) \vee t \in constructor\,(\lambda\omega).$$

Proof. By induction on the structure of A, using Lemma 43 (1). If $A \in VAR^{\Box}$, we are immediately done. If $A \equiv \Pi x{:}M.N$, $A \equiv \lambda x{:}M.N$ or $A \equiv MN$, then M and N are both constructors or kinds. All subexpressions of M and N are constructors or kinds by induction hypothesis, so we are done. □

4 $\lambda\omega \models SN \Rightarrow \lambda P\omega \models SN$

In Harper *et al.*, (1987) a proof of strong normalization is given for the system LF, which, roughly speaking, corresponds with our system λP. This is done by defining a map from $Term(\lambda P)$ to $Term(\lambda\to)$ that preserves reductions. Then, if there exists an infinite reduction sequence in λP, there is also an infinite reduction sequence in $\lambda\to$. From the strong normalization for $\lambda\to$ immediately follows the strong normalization for λP.

In the case of the systems $\lambda P\omega$ and $\lambda\omega$, the procedure will be the same; we shall define a map from $Term(\lambda P\omega)$ to $Term(\lambda\omega)$, that preserves reduction. The map $[\![\ ]\!]$ that will be used can be seen as a higher order version of the map defined by Harper *et al.*, (1987), although things get quite a bit more complicated here. It is also possible to restrict the map $[\![\ ]\!]$ to $Term(\lambda P2)$, to derive the result $SN(\lambda 2) \Rightarrow SN(\lambda P2)$.

Just as the case for λP, the map [] does not work uniformly on the terms of λPω. That is, we cannot define [] such that for all Γ, M and A

$$\Gamma \vdash_{\lambda P\omega} M:A \Rightarrow [\![\Gamma]\!] \vdash_{\lambda\omega} [\![M]\!]:[\![A]\!].$$

To show that [] really maps terms of λPω on terms of λω, one has to define another map $\tau : \{\Box\} \cup \mathit{kind}(\lambda P\omega) \cup \mathit{constructor}(\lambda P\omega) \to \mathit{type}(\lambda\omega)$ and to prove that

$$\Gamma \vdash_{\lambda P\omega} M:A \Rightarrow \tau(\Gamma) \vdash_{\lambda\omega} [\![M]\!]:\tau(A).$$

As the definitions of [] and τ do not really speak for themselves, we try to give some intuition on how they are defined. Therefore, we present some heuristics below on how the mappings from λP to λ→ should be extended to λPω to obtain a proof of $SN(\lambda\omega) \Rightarrow SN(\lambda P\omega)$.

Analysing the proof of $SN(\lambda\omega) \Rightarrow SN(\lambda P\omega)$, one can see that it does not use any higher order features; the proof can be done totally in Peano Arithmetic. In fact, one can even give a bound to the number of possible reduction steps starting from a certain term M by just taking the bound on the reductions starting from $[\![M]\!]$. The whole argument can therefore be done in Heyting Arithmetic. The equivalence of the two strong normalization properties turns out to be quite elementary, which was already stated by Berardi (1988), but in a much more elaborate way.

Heuristics 45

The idea of the mappings by Harper *et al.*, (1987) is to replace redexes that use type dependency by λ→-redexes. For example, let A be a type and

$$\frac{\Gamma \vdash_{\lambda P\omega} F:A \to * \quad \Gamma \vdash_{\lambda P\omega} t:A}{\Gamma \vdash_{\lambda P\omega} Ft:*},$$

then [] and τ must erase the type dependency such that the following is sound

$$\frac{\tau(\Gamma) \vdash_{\lambda\omega} [\![F]\!]:\tau(A \to *) \quad \tau(\Gamma) \vdash_{\lambda\omega} [\![t]\!]:\tau(A)}{\tau(\Gamma) \vdash_{\lambda\omega} [\![Ft]\!]:\tau(*)}.$$

This is solved by taking $[\![Ft]\!] \equiv [\![F]\!]\,[\![t]\!]$, $\tau(A \to *) \equiv \tau(A) \to 0$ and $\tau(*) \equiv 0$, where 0 is a fixed type variable. Further, if A is a type, M a constructor and t an object, the redex $(\lambda x{:}A.M)\,t$ is replaced by $(\lambda z{:}0.\lambda x{:}\tau(A).[\![M]\!])\,[\![A]\!]\,[\![t]\!]$, where z is a fresh variable not occurring free in A or M. In this way the type dependence redex is replaced by an ordinary one, and the possible redexes in A (which might be erased by τ) are preserved by the abstraction over $z{:}0$.

Now, for polymorphic terms the situation gets quite a bit more complicated. For example let

$$\frac{\Gamma \vdash_{\lambda P\omega} F:\Pi\alpha{:}*.\alpha \to \alpha \quad \Gamma \vdash_{\lambda P\omega} \sigma:*}{\Gamma \vdash_{\lambda P\omega} F\sigma:\sigma \to \sigma},$$

then the following must be sound

$$\frac{\tau(\Gamma) \vdash_{\lambda\omega} [\![F]\!]:\tau(\Pi\alpha{:}*.\alpha \to \alpha) \quad \tau(\Gamma) \vdash_{\lambda\omega} [\![\sigma]\!]:0}{\tau(\Gamma) \vdash_{\lambda\omega} [\![F\sigma]\!]:\tau(\sigma \to \sigma)}.$$

At first sight there are two possibilities:

(1) $$\begin{aligned}\tau(\Pi\alpha{:}*.\,\alpha\to\alpha) &\equiv 0\to\tau(\alpha\to\alpha),\\ [\![F\sigma]\!] &\equiv [\![F]\!]\,[\![\sigma]\!],\end{aligned}$$

but then $\tau(\Gamma)\vdash_{\lambda\omega}[\![F\sigma]\!]:\tau(\sigma\to\sigma)$

(2) $$\begin{aligned}\tau(\Pi\alpha{:}*.\,\alpha\to\alpha) &\equiv \Pi\alpha{:}*.\,\alpha\to\alpha,\\ [\![F\sigma]\!] &\equiv [\![F]\!]\,\tau(\sigma),\end{aligned}$$

but then the possible reductions in σ are not preserved.

The solution given below is to do both (1) and (2) by defining

$$\begin{aligned}\tau(\Pi\alpha{:}*.\,\alpha\to\alpha) &\equiv \Pi\alpha{:}*.\,0\to\alpha\to\alpha,\\ [\![F\sigma]\!] &\equiv [\![F]\!]\,\tau(\sigma)\,[\![\sigma]\!].\end{aligned}$$

Now the application is sound and redexes are preserved.

The definition of τ that follows is not an obviously well-defined mapping from the kinds and constructors of $\lambda P\omega$ to the constructors of $\lambda\omega$. Moreover, in the definition of τ, we shall use something like the 'range' ρ for kinds of $\lambda P\omega$, which is just the $\lambda\omega$-kind obtained by erasing all type dependencies. It will turn out that if M is a constructor of kind N in $\lambda P\omega$, then $\tau(M)$ is a constructor of kind $\rho(N)$ in $\lambda\omega$, which shows that τ is defined in the right way.

Definition 46

The map $\rho:\{\Box\}\cup kind(\lambda P\omega)\to kind(\lambda\omega)$ is inductively defined by

(i) $\rho(*)=\rho(\Box)=*$,

(ii) $\rho(\Pi\alpha{:}M.N)=\rho(M)\to\rho(N)$ if $\Pi\alpha{:}M.N$ is formed by $(\Box,\Box)$,
$\rho(\Pi x{:}M.N)=\rho(N)$ if $\Pi x{:}M.N$ is formed by $(*,\Box)$.

The case distinction in the definition is correct by Corollary 32.

Remark 47

The following properties of the map ρ will be used

(i) $k\in kind(\lambda P\omega), u\in VAR, A\in Term(\lambda P\omega)\Rightarrow \rho(k[u:=A])\equiv\rho(k)\equiv\rho(k)\,[u:=A]$,

(ii) $k_1,k_2\in kind(\lambda P\omega), k_1=k_2\Rightarrow \rho(k_1)\equiv\rho(k_2)$.

Now we choose one of the variables of $VAR^{\Box}$ to act as a fixed constant, i.e., it will not be used as bound variable in an abstraction. This variable will be denoted by 0.

Definition 48

The map $\tau:\{\Box\}\cup kind(\lambda P\omega)\cup constructor(\lambda P\omega)\to Term(\lambda\omega)$ is inductively defined by

(i) $\tau(*)=\tau(\Box)=0$,

(ii) $\tau(\alpha)=\alpha$,

(iii) $\tau(\Pi\alpha{:}M.N)=\Pi\alpha{:}\rho(M).\tau(M)\to\tau(N)$ if $\Pi\alpha{:}M.N$ is formed by $(\Box,s)$,
$\tau(\Pi x{:}M.N)=\Pi x{:}\tau(M).\tau(N)$ if $\Pi x{:}M.N$ is formed by $(*,s)$,

(iv) $\tau(\lambda\alpha{:}M.N) = \lambda\alpha{:}\rho(M).\tau(N)$ if $\lambda\alpha{:}M.N$ is formed by $(\Box,\Box)$,
$\tau(\lambda x{:}M.N) = \tau(N)$ if $\lambda x{:}M.N$ is formed by $(*,\Box)$,
(v) $\tau(MN) = \tau(M)\tau(N)$ if MN is formed by $(\Box,\Box)$,
$\tau(MN) = \tau(M)$ if MN is formed by $(*,\Box)$.

The definition by cases is again correct by Corollary 32. That the range of τ is a subset of *Term*($\lambda\omega$) is stated in Lemma 52.

Lemma 49
For $B, B' \in kind(\lambda P\omega) \cup constructor(\lambda P\omega)$

(i) $x \in VAR^*, A \in object(\lambda P\omega) \Rightarrow \tau(B[x:=A]) \equiv \tau(B) \equiv \tau(B)[x:=A]$,
(ii) $\alpha \in VAR^\Box, A \in constructor(\lambda P\omega) \Rightarrow \tau(B[\alpha:=A]) \equiv \tau(B)[\alpha:=\tau(A)]$,
(iii) $B \to_\beta B' \Rightarrow \tau(B) \to_\beta \tau(B')$ or $\tau(B) \equiv \tau(B')$.

Proof. By induction on the structure of B. After applying the stripping lemma and the induction hypothesis, (i) and (ii) follow immediately from Remark 47(ii).

The proof of (iii) goes by induction on the structure of B. The interesting situation occurs when $B \equiv (\lambda u{:}M.N)P$, $B' \equiv N[u:=P]$. If $M \in type(\lambda P\omega)$, then $\tau((\lambda u{:}M.N)P) \equiv \tau(N) \equiv \tau(N[u:=P])$ by (i). If $M \in kind(\lambda P\omega)$, then $\tau((\lambda\alpha{:}M.N)P) \equiv (\lambda\alpha{:}\rho(M).\tau(N))\tau(P) \to_\beta \tau(N)[\alpha:=\tau(P)] \equiv \tau(N[\alpha:=P])$ by (ii). □

In order to map *Context*($\lambda P\omega$) into *Context*($\lambda\omega$), we choose for every variable α of $VAR^\Box$ a connected variable x^α in VAR^*, such that no two variables of $VAR^\Box$ are connected with the same variable of VAR^*.

Definition 50
Let $A \in Term(\lambda P\omega)$, $\Gamma = u_1{:}A_1, u_2{:}A_2, \ldots, u_n{:}A_n \in Context(\lambda P\omega)$

(i) $\tau(x{:}A) := x{:}\tau(A)$ if $x \in VAR^*$,
$\tau(\alpha{:}A) := \alpha{:}\rho(A)$, $x^\alpha{:}\tau(A)$ if $\alpha \in VAR^\Box$.
(ii) $\tau(\Gamma) := 0{:}*, d{:}\bot, \tau(u_1{:}A_1), \tau(u_2{:}A_2), \ldots, \tau(u_n{:}A_n)$.

The reason for putting $0{:}*$ and $d{:}\bot \equiv \Pi\alpha{:}*.\alpha$ in the context is that in the definition of $[\![\ ]\!]$ on terms of $\lambda P\omega$ it will be necessary to have a canonical inhabitant for every type and kind. If $\tau(\Gamma) \vdash_{\lambda\omega} B{:}*/\Box$, we want $\tau(\Gamma) \vdash_{\lambda\omega} c^B{:}B$ for a c^B which does not depend on the structure of Γ.

Now, if $\tau(\Gamma) \vdash_{\lambda\omega} B{:}*$ we shall put $c^B \equiv dB$ and if $\tau(\Gamma) \vdash_{\lambda\omega} B{:}\Box$ a canonical inhabitant of B is defined inductively by

(i) $B \equiv *$, then $c^* \equiv 0$,
(ii) $B \equiv k_1 \to k_2$, then $c^{k_1 \to k_2} \equiv \lambda\alpha{:}k_1.c^{k_2}$, which gives in every context an inhabitant of B by Lemma 41.

Note that $c^B[u:=N] \equiv c^{B[u:=N]}$ for all kinds and types B, variables u and terms N, in $\lambda\omega$.

Before proving the lemma stating $\Gamma \vdash_{\lambda P\omega} M:N \Rightarrow \tau(\Gamma) \vdash_{\lambda\omega} \tau(M):\rho(N)$ for Γ a context, N a kind and M a constructor, we shall give some examples. In these examples the declarations $x^\alpha:\tau(A)$ and $x:\sigma$, with σ a type, do not play a role in the context $\tau(\Gamma)$. That is because in constructors and kinds of $\lambda\omega$ no object variables occur. The role of the object variables will become clear in the definition of $[\![\,]\!]$.

Examples 51

(i) $\vdash_{\lambda P\omega} \Pi\alpha:*.\alpha\to\alpha:*$ and $0:*, d:\bot \vdash_{\lambda\omega} \Pi\alpha:*.0\to\alpha\to\alpha:*$.

(ii) $\alpha:* \vdash_{\lambda P\omega} \lambda x:\alpha.\lambda P:\alpha\to *.Px:\alpha\to(\alpha\to *)\to *$ and
$0:*, d:\bot, \alpha:*, x^\alpha:0 \vdash_{\lambda\omega} \lambda P:*.P : *\to *$.

Lemma 52
For $\Gamma \in Context(\lambda P\omega)$, $M, N \in Term(\lambda P\omega)$

$$\Gamma \vdash_{\lambda P\omega} M:N:\Box \vee \Gamma \vdash_{\lambda P\omega} M:N \equiv \Box \Rightarrow \tau(\Gamma) \vdash_{\lambda\omega} \tau(M):\rho(N).$$

Proof. By induction on the length of the derivation of $\Gamma \vdash_{\lambda P\omega} M:N$, distinguishing cases according to the last applied rule

– If $\vdash_{\lambda P\omega} M:N$ is an axiom, we are done.

– If the last rule is (start), then $\Gamma', \alpha:* \vdash_{\lambda P\omega} \alpha:*$ is the conclusion, and so $\tau(\Gamma', \alpha:*) \vdash_{\lambda P\omega} \tau(\alpha):\rho(*)$. ($\tau(\Gamma')$ is a legal context by induction hypothesis.)

– If the last rule is (weakening), we are immediately done by the induction hypothesis and once or twice (weakening).

– If the last rule is (conversion), we are done by the induction hypothesis.

– $M \equiv \Pi u:B_1.B_2$, $N \equiv s \in \{*, \Box\}$ and the last rule is (Π-rule), then by induction hypothesis $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1):*$ and $\tau(\Gamma, u:B_1) \vdash_{\lambda\omega} \tau(B_2):*$.

If $B_1 \in type(\lambda P\omega)$, then $\tau(\Gamma, u:B_1) = \tau(\Gamma), u:\tau(B_1)$, and so $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1)\to\tau(B_2):*$ (by rule $(*,*)$).

If $B_1 \in kind(\lambda P\omega)$, then $\tau(\Gamma, u:B_1) = \tau(\Gamma), u:\rho(B_1), x^u:\tau(B_1)$, and so, by rules $(*,*)$ and $(\Box,*)$, $\tau(\Gamma) \vdash_{\lambda\omega} \Pi u:\rho(B_1).\tau(B_1)\to\tau(B_2):*$.

– $M \equiv \lambda u:B_1.B_2$, $N \equiv \Pi u:B_1.C_2 \in kind(\lambda P\omega)$ and the last rule is (λ-rule), then by induction hypothesis $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1):*$ and $\tau(\Gamma, u:B_1) \vdash_{\lambda\omega} \tau(B_2):\rho(C_2)$.

If $B_1 \in type(\lambda P\omega)$, then $\tau(\lambda u:B_1.B_2) \equiv \tau(B_2)$, $\rho(\Pi u:B_1.C_2) \equiv \rho(C_2)$ and $\tau(u:B_1) = u:\tau(B_1)$. By Lemma 43 $u \notin FV(\tau(B_2):\rho(C_2))$, so by substituting $c^{\tau(B_1)}$ for u we find $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_2):\rho(C_2)$.

If $B_1 \in kind(\lambda P\omega)$, then $\tau(\lambda u:B_1.B_2) \equiv \lambda u:\rho(B_1).\tau(B_2)$, $\rho(\Pi u:B_1.C_2) \equiv \rho(B_1)\to\rho(C_2)$ and $\tau(\Gamma, u:B_1) = \tau(\Gamma), u:\rho(B_1), x^u:\tau(B_1)$. By 3.10 $x^u \notin FV(\tau(B_2):\rho(C_2))$, so by substituting $c^{\tau(B_1)}$ for x^u we find $\tau(\Gamma), u:\rho(B_1) \vdash_{\lambda\omega} \tau(B_2):\rho(C_2)$, and by one application of (λ-rule) (rule $(\Box,\Box)$) $\tau(\Gamma) \vdash_{\lambda\omega} \lambda u:\rho(B_1).\tau(B_2):\rho(B_1)\to\rho(C_2)$.

– $M \equiv B_1\,B_2$, $N = C_2[x:=B_2] \in kind(\lambda P\omega)$ and the rule is (application), then let $\Gamma \vdash_{\lambda P\omega} B_1:\Pi x:C_1.C_2$, and we find by stripping and the induction hypothesis:

If $B_2 \in object(\lambda P\omega)$, then $\tau(B_1\,B_2) \equiv \tau(B_1)$, $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1):\rho(\Pi x:C_1.C_2)$ and $\rho(\Pi x:C_1.C_2) \equiv \rho(C_2) \equiv \rho(C_2[x:=B_2])$.

If $B_2 \in constructor(\lambda P\omega)$, then $\tau(B_1\,B_2) \equiv \tau(B_1)\,\tau(B_2)$, $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1):\rho(\Pi x:C_1.C_2)$ and $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_2):\rho(C_1)$. Furthermore, $\rho(\Pi x:C_1.C_2) \equiv \rho(C_1)\to\rho(C_2)$, so $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1\,B_2):\rho(C_2) \equiv \rho(C_2[x:=B_2])$. $\Box$

Definition 53
The map $[\![\,]\!] : \mathit{kind}(\lambda P\omega) \cup \mathit{constructor}(\lambda P\omega) \cup \mathit{object}(\lambda P\omega) \to \mathit{Term}(\lambda\omega)$ is inductively defined by

(i) $[\![*]\!] = c^0$,
(ii) $[\![x]\!] = x$, if $x \in \mathit{VAR}^*$,
(iii) $[\![\alpha]\!] = x^\alpha$, if $\alpha \in \mathit{VAR}^\Box$,
(iv) $[\![\Pi x{:}M.N]\!] = c^{0\to 0\to 0}[\![M]\!][\![N]\!][x := c^{\tau(M)}]$ if $\Pi x{:}M.N$ is formed by $(*,*)$ or $(*,\Box)$,
$[\![\Pi\alpha{:}M.N]\!] = c^{0\to 0\to 0}[\![M]\!][\![N]\!][\alpha := c^{\rho(M)}, x^\alpha := c^{\tau(M)}]$ if $\Pi\alpha{:}M.N$ is formed by $(\Box,*)$ or $(\Box,\Box)$,
(v) $[\![\lambda x{:}M.N]\!] = (\lambda z{:}0.\lambda x{:}\tau(M).[\![N]\!])[\![M]\!]$ with $z \in \mathit{VAR}^*$ fresh, if $\lambda x{:}M.N$ is formed by $(*,*)$ or $(*,\Box)$,
$[\![\lambda\alpha{:}M.N]\!] = (\lambda z{:}0.\lambda\alpha{:}\rho(M).\lambda x^\alpha{:}\tau(M).[\![N]\!])[\![M]\!]$ with $z \in \mathit{VAR}^*$ fresh, if $\lambda\alpha{:}M.N$ is formed by $(\Box,*)$ or $(\Box,\Box)$,
(vi) $[\![MN]\!] = [\![M]\!][\![N]\!]$ if MN is formed by $(*,*)$ or $(*,\Box)$,
$[\![MN]\!] = [\![M]\!]\tau(N)[\![N]\!]$ if MN is formed by $(\Box,*)$ or $(\Box,\Box)$.

The definition by cases is all right by Corollary 32. The idea of the mapping $[\![\,]\!]$ is that it maps terms of $\lambda P\omega$ on terms of $\lambda\omega$ (as stated in the definition), and preserves all possible reduction sequences. The first fact is stated in Theorem 55, the second in Theorem 57. To illustrate these theorems we first give some examples of $\lambda P\omega$-terms and their image in $\lambda\omega$ under the mapping $[\![\,]\!]$.

Examples 54
(i) $\vdash_{\lambda P\omega} \lambda\alpha{:}*.\lambda x{:}\alpha.x : \Pi\alpha{:}*.\alpha\to\alpha$ and
$0{:}*, d{:}\bot \vdash_{\lambda\omega} (\lambda z_2{:}0.\lambda\alpha{:}*.\lambda x^\alpha{:}0.(\lambda z_1{:}0.\lambda x{:}\alpha.x)x^\alpha)c^0 : \Pi\alpha{:}*.0\to\alpha\to\alpha$.
(ii) $\beta{:}*, y{:}\beta \vdash_{\lambda P\omega} (\lambda\alpha{:}*.\lambda x{:}\alpha.x)\beta y : \beta$ and
$0{:}*, d{:}\bot, \beta{:}*, x^\beta{:}0, y{:}\beta \vdash_{\lambda\omega} (\lambda z_2{:}0.\lambda\alpha{:}*.\lambda x^\alpha{:}0.(\lambda z_1{:}0.\lambda x{:}\alpha.x)x^\alpha)c^0\beta x^\beta y : \beta$.
(iii) $\alpha{:}* \vdash_{\lambda P\omega} \lambda P{:}\alpha\to*.\lambda x{:}\alpha.\lambda y{:}Px.y : \Pi P{:}\alpha\to*.\Pi\alpha{:}*.Px\to Px$ and
$\alpha{:}*, x^\alpha{:}0 \vdash_{\lambda\omega} (\lambda z{:}0.\lambda P{:}*.\lambda x^P{:}\alpha\to 0.(\lambda z{:}0.\lambda x{:}\alpha.(\lambda z{:}0.\lambda y{:}P.y)(x^P x))x^\alpha)$
$(c^{0\to 0\to 0}c^0c^0) : \Pi P{:}*.(\alpha\to 0)\to(\Pi x{:}\alpha.P\to P)$, which term reduces to
$\lambda P{:}*.\lambda x^P{:}\alpha\to 0.\lambda x{:}\alpha.\lambda y{:}P.y$.

Theorem 55
For $\Gamma \in \mathit{Context}(\lambda P\omega)$, $M, N \in \mathit{Term}(\lambda P\omega)$

$$\Gamma \vdash_{\lambda P\omega} M{:}N \Rightarrow \tau(\Gamma) \vdash_{\lambda\omega} [\![M]\!]{:}\tau(N).$$

Proof. By induction on the structure of M. By Lemma 52 we know that $\tau(\Gamma)$ is a legal context in $\lambda\omega$ and that from $\tau(\Gamma) \vdash_{\lambda\omega} [\![M]\!]{:}\tau(N)$, $N = N'$ and $\Gamma \vdash_{\lambda P\omega} N' : */\Box$, we may conclude $\tau(\Gamma) \vdash_{\lambda\omega} [\![M]\!]{:}\tau(N')$.
– $M \equiv *$, $N \equiv \Box$, then $[\![*]\!] \equiv c^0 : 0 \equiv \tau(\Box)$ in $\tau(\Gamma)$.
– $M \equiv u \in \mathit{VAR}^\Box$, then $u{:}A \in \Gamma$, with $A = N$.
If $u \equiv \alpha \in \mathit{VAR}^\Box$, then $\tau(\alpha{:}A) \equiv \alpha{:}\rho(A), x^\alpha{:}\tau(A) \in \tau(\Gamma)$, if $u \equiv x \in \mathit{VAR}^*$, then $\tau(x{:}A) \equiv x{:}\tau(A) \in \tau(\Gamma)$. In both cases $\tau(\Gamma) \vdash_{\lambda\omega} [\![u]\!]{:}\tau(A) = \tau(N)$ and we are done.
– $M \equiv \Pi u{:}B_1.B_2$, then $\Gamma \vdash_{\lambda P\omega} B_1 : s_1$ and $\Gamma, u{:}B_1 \vdash_{\lambda P\omega} B_2 : s_2$ for some $s_1, s_2 \in \{*, \Box\}$.

By induction hypothesis $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_1]\!] : 0$ and $\tau(\Gamma, u:B_1) \vdash_{\lambda\omega} [\![B_2]\!] : 0$.

By Lemma 52 $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1) : *$ and so $\tau(\Gamma) \vdash_{\lambda\omega} c^{\tau(B_1)} : \tau(B_1)$.

If $s_1 \equiv *$, $u \equiv x \in VAR^*$, then $\tau(\Gamma, x:B_1) = \tau(\Gamma), x:\tau(B_1)$, so with the substitution lemma $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_2]\!][x := c^{\tau(B_1)}] : 0$.

With twice (application) $\tau(\Gamma) \vdash_{\lambda\omega} c^{0\to 0\to 0} [\![B_1]\!] [\![B_2]\!][x := c^{\tau(B_1)}] : 0$.

If $s_1 \equiv \Box$, $u \equiv \alpha \in VAR^{\Box}$, then $\tau(\Gamma, \alpha:B_1) = \tau(\Gamma), \alpha:\rho(B_1), x^\alpha:\tau(B_1)$, so with the substitution lemma $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_2]\!][x^\alpha := c^{\tau(B_1)}, \alpha := c^{\rho(B_1)}] : 0$.

With twice (application) $\tau(\Gamma) \vdash_{\lambda\omega} c^{0\to 0\to 0} [\![B_1]\!] [\![B_2]\!][x^\alpha := c^{\tau(B_1)}, \alpha := c^{\rho(B_1)}] : 0$. In both cases, $\tau(\Gamma) \vdash_{\lambda\omega} [\![\Pi u:B_1 . B_2]\!] : 0$.

– $M \equiv \lambda u:B_1 . B_2$, then $\Gamma \vdash_{\lambda P\omega} B_1 : s_1$, $\Gamma, u:B_1 \vdash_{\lambda P\omega} B_2 : C_2 : s_2$ for some $C_2 \in Term(\lambda P\omega)$ with $N = \Pi u:B_1 . C_2$ and $s_1, s_2 \in \{*, \Box\}$.

By induction hypothesis $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_1]\!] : 0$ and $\tau(\Gamma, u:B_1) \vdash_{\lambda\omega} [\![B_2]\!] : \tau(C_2)$.

By Lemma 52 $\tau(\Gamma) \vdash_{\lambda\omega} \tau(B_1) : *$ and $\tau(\Gamma, u:B_1) \vdash_{\lambda\omega} \tau(C_2) : *$.

If $s_1 \equiv *$, $u \equiv x \in VAR^*$, then $\tau(\Gamma, x:B_1) = \tau(\Gamma), x:\tau(B_1)$. Applying twice ($\lambda$-rule) and once (application) we find $\tau(\Gamma) \vdash_{\lambda\omega} (\lambda z:0 . \lambda x:\tau(B_1) . [\![B_2]\!])\, [\![B_1]\!] : \tau(B_1) \to \tau(C_2)$.

If $s_1 \equiv \Box$, $u \equiv \alpha \in VAR^{\Box}$, then $\tau(\Gamma, \alpha:B_1) = \tau(\Gamma), \alpha:\rho(B_1), x^\alpha:\tau(B_1)$. With three times ($\lambda$-rule) and once (application) we find
$\tau(\Gamma) \vdash_{\lambda\omega} (\lambda z:0 . \lambda\alpha:\rho(B_1) . \lambda x^\alpha:\tau(B_1) . [\![B_2]\!])\, [\![B_1]\!] : \Pi\alpha:\rho(B_1) . \tau(B_1) \to \tau(C_2)$.
In both cases, $\tau(\Gamma) \vdash_{\lambda\omega} [\![\lambda u:B_1 . B_2]\!] : \tau(\Pi u:B_1 . C_2) = \tau(N)$ and we are done.

– $M \equiv B_1\, B_2$, then $\Gamma \vdash_{\lambda P\omega} B_1 : \Pi u:C_1 . C_2$ and $\Gamma \vdash_{\lambda P\omega} B_2 : C_1$ for some $C_1, C_2 \in Term(\lambda P\omega)$ with $N = C_2[u := B_2]$.

By induction hypothesis $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_1]\!] : \tau(\Pi u:C_1 . C_2)$ and $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_2]\!] : \tau(C_1)$. By Lemma 52 $\tau(\Gamma) \vdash_{\lambda\omega} \tau(C_1) : *$.

If $C_1 \in type(\lambda P\omega)$, $u \equiv x \in VAR^*$, then $\tau(\Pi x:C_1 . C_2) \equiv \tau(C_1) \to \tau(C_2)$, so $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_1]\!]\, [\![B_2]\!] : \tau(C_2)[x := [\![B_2]\!]] \equiv \tau(C_2[x := B_2])$ (by Lemma 49(i)).

If $C_1 \in kind(\lambda P\omega)$, $u \equiv \alpha \in VAR^{\Box}$, then $\tau(\Pi\alpha:C_1 . C_2) \equiv \Pi\alpha:\rho(C_1) . \tau(C_1) \to \tau(C_2)$, so $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_1]\!]\, \tau(B_2)\, [\![B_2]\!] : \tau(C_2)[\alpha := \tau(B_2), x^\alpha := [\![B_2]\!]] \equiv \tau(C_2[\alpha := B_2])$ (by Lemma 49).
In both cases, $\tau(\Gamma) \vdash_{\lambda\omega} [\![B_1\, B_2]\!] : \tau(C_2[u := B_2]) = \tau(N)$ and we are done. □

Lemma 56
For $M \in Term(\lambda P\omega)$

(i) $x \in VAR^*$, $N \in object(\lambda P\omega)$ $\Rightarrow [\![M]\!][x := [\![N]\!]] \equiv [\![M[x := N]]\!]$,
(ii) $\alpha \in VAR^{\Box}$, $N \in constructor(\lambda P\omega)$ $\Rightarrow [\![M]\!][\alpha := \tau(N), x^\alpha := [\![N]\!]] \equiv [\![M[\alpha := N]]\!]$.

Proof (*i*). By induction on the structure of M. If $M \in VAR^*$ or $M \in VAR^{\Box}$, we are immediately done. If M is a composed term, we are done by the induction hypothesis, and the fact that $\tau(Q[x := N]) \equiv \tau(Q)[x := [\![N]\!]]$ and $\rho(Q[x := N]) \equiv \rho(Q)[x := [\![N]\!]]$ (by 47 and 49), and so $c^{\tau(B)}[u := [\![N]\!]] \equiv c^{\tau(B[u := N])}$ and $c^{\rho(B)}[u := [\![N]\!]] \equiv c^{\rho(B[u := N])}$.

Proof (*ii*). By induction on the structure of M. If $M \in VAR^*$ or $M \in VAR^{\Box}$, we are immediately done. If M is a composed term, we are done by the induction hypothesis and the following facts

$$\tau(Q)[\alpha := \tau(N), x^\alpha := [\![N]\!]] \equiv \tau(Q[\alpha := N]),$$

$$\rho(Q)[\alpha := \tau(N), x^\alpha := [\![N]\!]] \equiv \rho(Q[\alpha := N]),$$

$$c^{\tau(Q)}[\alpha := \tau(N), x^{\alpha} := [\![N]\!]] \equiv c^{\tau(Q[\alpha := N])},$$

$$c^{\rho(Q)}[\alpha := \tau(N), x^{\alpha} := [\![N]\!]] \equiv c^{\rho(Q[\alpha := N])},$$

which are true by 47 and 49, and the fact that $x^{\alpha} \notin FV(\rho(Q))$. □

Theorem 57
For $B, B' \in Term(\lambda P\omega)$

$$M \to_{\beta} M' \Rightarrow [\![M]\!] \twoheadrightarrow^{+0} [\![M']\!],$$

where $\twoheadrightarrow^{+0}$ denotes a non zero step reduction.

Proof. By induction on the structure of M.

If $M \equiv \Pi u{:}B_1 . B_2$, $M \equiv \lambda u{:}B_1 . B_2$ or $M \equiv B_1 B_2$ with $M' \equiv B_1 B_2'$ or $B_1' B_2$, then $[\![M]\!] \twoheadrightarrow^{+0} [\![M']\!]$ follows immediately from the induction hypothesis.

If $M \equiv (\lambda u{:}B_1 . B_2) C$, $M' \equiv B_2[u := C]$, then we distinguish two cases, according to the rule by which $M \equiv (\lambda u{:}B_1 . B_2) C$ is formed and apply Lemma 56.

– $(\lambda u{:}B_1 . B_2) C$ is formed by $(*, *)$ or $(*, \Box)$. Then $[\![M]\!] \equiv [\![(\lambda u{:}B_1 . B_2) C]\!] \equiv (\lambda z{:}0 . \lambda u{:}\tau(B_1) . [\![B_2]\!])\, [\![B_1]\!]\, [\![C]\!] \twoheadrightarrow^{+0} [\![B_2]\!][u := [\![C]\!]] \equiv [\![B_2[u := C]]\!] \equiv [\![M']\!]$.

– $(\lambda u{:}B_1 . B_2) C$ is formed by $(\Box, *)$ or $(\Box, \Box)$. Then $[\![M]\!] \equiv [\![(\lambda u{:}B_1 . B_2) C]\!] \equiv (\lambda z{:}0 . \lambda u{:}\rho(B_1) . \lambda x^{u}{:}\tau(B_1) . [\![B_2]\!])\, [\![B_1]\!]\, \tau(C)\, [\![C]\!] \twoheadrightarrow^{+0} [\![B_2]\!][u := \tau(C), \quad x^{u} := [\![C]\!]] \equiv [\![B_2[u := C]]\!] \equiv [\![M']\!]$. □

Theorem 58

$$\lambda\omega \models SN \Rightarrow \lambda P\omega \models SN.$$

Proof. Suppose $\lambda\omega \models SN$.

Now let $M_1, M_2, M_3, \ldots \in Term(\lambda P\omega)$, and $M_1 \to M_2 \to M_3 \to \ldots$ an infinite reduction sequence. Then $[\![M_1]\!], [\![M_2]\!], [\![M_3]\!], \ldots \in Term(\lambda\omega)$ and $[\![M_1]\!] \twoheadrightarrow^{+0} [\![M_2]\!] \twoheadrightarrow^{+0} [\![M_3]\!] \twoheadrightarrow^{+0} \ldots$, so there is an infinite reduction sequence in $\lambda\omega$. This is not possible, so there is no infinite reduction sequence in $\lambda P\omega$.

One can even give a bound to the number of possible reduction steps starting from a term M by taking the bound on the reductions starting from $[\![M]\!]$ in $\lambda\omega$. □

5 $\lambda{\to} \models SN$

In this chapter the strong normalization for the system $\lambda{\to}$ will be proved. It gives a good idea of the proof of strong normalization for $\lambda\omega$. As a matter of fact, the proof of strong normalization for the untyped terms of $\lambda\omega$ is a straightforward higher order extension of the proof given below for $\lambda{\to}$.

Another reason for first proving $\lambda{\to} \models SN$ is, that as a corollary we will find $SN(constructor(\lambda\omega))$. This reduces the question of SN for $\lambda\omega$ to the question of SN for $object(\lambda\omega)$. In Chapter 6 this question will again be reduced to the question of SN for untyped objects of $\lambda\omega$.

The strong normalization proof below is a variant of Tait's proof for the system $\lambda 2$ (Tait, 1975).

Facts 59

$$\forall M \in object(\lambda\!\rightarrow).[SN(|M|)] \Rightarrow \forall M \in Term(\lambda\!\rightarrow).[SN(M)].$$

Definition 60

Let Λ be the set of untyped lambda terms.

$X \subseteq \Lambda$ is *saturated* iff

(i) $\forall M \in X.[SN(M)]$,

(ii) $\forall \vec{Q} \in SN, x \in VAR.[x\vec{Q} \in X]$,

(iii) $\forall \vec{Q}, M, P \in SN.[M[x:=P]\vec{Q} \in X \Rightarrow (\lambda x.M)P\vec{Q} \in X]$.

$E := \{X \subseteq \Lambda \mid X \text{ is saturated}\}$.

Definition 61

Let X and Y be saturated sets

$$X \rightarrow Y := \{M \in \Lambda \mid \forall N \in X.[MN \in Y]\}.$$

Lemma 62

(i) $SN \in E$.

(ii) $\forall X, Y \in E.[X \rightarrow Y \in E]$.

Proof. Easy checking of properties (i), (ii) and (iii) of definition 60.

Definition 63

A *type valuation* is a map $\xi : VAR^{\square} \rightarrow E$.

Definition 64

Let ξ a type evaluation. The map $[\![\,]\!]^{\xi} : type(\lambda\!\rightarrow) \rightarrow E$ is defined by

(i) $[\![\alpha]\!]^{\xi} = \xi(\alpha)$, if $\alpha \in VAR^{\square}$,

(ii) $[\![A \rightarrow B]\!]^{\xi} = [\![A]\!]^{\xi} \rightarrow [\![B]\!]^{\xi}$.

Notice that all constructors of $\lambda\!\rightarrow$ are types, and that all types of $\lambda\!\rightarrow$ are variables or of the form $A \rightarrow B$ (see Lemma 41).

Definition 65

Let ξ be a type valuation, $\Gamma \in Context(\lambda\!\rightarrow)$. An *object valuation of* Γ *with respect to* ξ is a map $\rho : VAR^{*} \rightarrow \Lambda$ such that $x : A \in \Gamma \Rightarrow \rho(x) \in [\![A]\!]^{\xi}$.

Definition 66

Let $\Gamma \in Context(\lambda\!\rightarrow)$, ξ a type valuation, ρ an object valuation of Γ w.r.t. ξ. The map $[\![\,]\!]_{\rho} : \Gamma\text{-}object(\lambda\!\rightarrow) \rightarrow \Lambda$ is defined by

(i) $[\![x]\!]_{\rho} = \rho(x)$,

(ii) $[\![\lambda x : A.b]\!]_{\rho} = \lambda x.[\![b]\!]_{\rho(x:=x)}$,

(iii) $[\![c\,a]\!]_{\rho} = [\![c]\!]_{\rho}[\![a]\!]_{\rho}$.

Notice that this definition is correct; the three cases cover all possible structures of objects. As one easily checks, if $c\,a \in object(\lambda\!\rightarrow)$, then $c, a \in object(\lambda\!\rightarrow)$ and if $\lambda x{:}A\,.\,b \in object(\lambda\!\rightarrow)$, then $b \in \text{object}(\lambda\!\rightarrow)$.

Definition 67
Let $\Gamma \in Context(\lambda\!\rightarrow)$, $M \in object(\lambda\!\rightarrow)$, $C \in type(\lambda\!\rightarrow)$. $\Gamma \models_{\lambda\rightarrow} M{:}C$ iff $\forall \xi$ type valuation. $\forall \rho$ object valuation of Γ w.r.t. ξ. $[[\![M]\!]_\rho \in [\![C]\!]^\xi]$.

Theorem 68
For $\Gamma \in Context(\lambda\!\rightarrow)$, $M \in object(\lambda\!\rightarrow)$, $C \in type(\lambda\!\rightarrow)$

$$\Gamma \vdash_{\lambda\rightarrow} M{:}C \Rightarrow \Gamma \models_{\lambda\rightarrow} M{:}C.$$

Proof. By induction on the structure of M. The interesting case occurs when $M \equiv \lambda x{:}A\,.\,b$, $C \equiv A \rightarrow B$. Then, by stripping and the induction hypothesis, $\Gamma, x{:}A \models_{\lambda\rightarrow} b{:}B$, i.e., $\forall a \in [\![A]\!]^\xi\,.\,[[\![b]\!]_{\rho(x:=a)} \in [\![B]\!]^\xi]$. But then $\lambda x\,.\,[\![b]\!]_{\rho(x:=x)} \in [\![A]\!]^\xi \rightarrow [\![B]\!]^\xi$, and we are done. □

Theorem 69

$$\lambda\!\rightarrow \models SN.$$

Proof. Let $\Gamma \in Context(\lambda\!\rightarrow)$, $M \in object(\lambda\!\rightarrow)$, $C \in type(\lambda\!\rightarrow)$ and $\Gamma \vdash_{\lambda\rightarrow} M{:}C$. Take as type valuation ξ the function $\xi{:}\alpha \mapsto SN$, and as term valuation ρ of Γ w.r.t. ξ, the map $\rho{:}x \mapsto x$. Then $[\![M]\!]_\rho \equiv |M|$ and $[\![C]\!]^\xi \subseteq SN$. Moreover, $[\![M]\!]_\rho \in [\![C]\!]^\xi$, so $|M| \in SN$ and $M \in SN$. □

Corollary 70

$$\forall M \in constructor(\lambda\omega)\,.\,SN(M).$$

Proof. We define a map $[\![\,]\!]$ from $kind(\lambda\omega)$ to $type(\lambda\!\rightarrow)$, and from $constructor(\lambda\omega)$ to $object(\lambda\!\rightarrow)$ that preserves reduction. Just as in Chapter 4, we assume to have a variable $x^\alpha \in VAR^*$ for every variable $\alpha \in VAR^\square$ Further, 0 is a fixed element of $VAR^\square$ and $c^{[k_1]}$ is a canonical constant of type $[\![k_1]\!]$, defined by taking c^0 as a fixed variable in VAR^*, and defining $c^{[k_1]\rightarrow[k_2]} = \lambda x{:}[\![k_1]\!]\,.\,c^{[k_2]}$.

Define

$$\begin{aligned}
[\![*]\!] &= 0,\\
[\![k_1 \rightarrow k_2]\!] &= [\![k_1]\!] \rightarrow [\![k_2]\!] \quad \text{if } k_1 \text{ and } k_2 \text{ are kinds},\\
[\![\alpha]\!] &= x^\alpha,\\
[\![\lambda\alpha{:}k\,.\,P]\!] &= \lambda x^\alpha{:}\,[\![k]\!]\,.\,[\![P]\!] \quad \text{if } P \text{ is a constructor},\\
[\![P\,Q]\!] &= [\![P]\!]\,[\![Q]\!] \quad \text{if } P \text{ and } Q \text{ are constructors},\\
[\![\sigma \rightarrow \tau]\!] &= c^{0\rightarrow 0\rightarrow 0}\,[\![\sigma]\!]\,[\![\tau]\!],\\
[\![\Pi\alpha{:}k\,.\,\sigma]\!] &= [\![\sigma]\!]\,[x^\alpha := c^{[k]}] \quad \text{if } k \text{ is a kind}.
\end{aligned}$$

Define further for $\Gamma \in Context(\lambda\omega)$ the context $[\![\Gamma]\!]$ by removing declarations of the form $x{:}A$, replacing declarations of the form $\alpha{:}A$ by $x^\alpha{:}[\![A]\!]$, and adding the two declarations $0{:}*$ and $c^0{:}0$.

As one easily checks (using strengthening to verify that the removal of object variable declarations is all right), we then have $\Gamma \vdash_{\lambda\omega} M:k(:\Box) \Rightarrow [\![\Gamma]\!] \vdash_{\lambda\to} [\![M]\!]:[\![k]\!]$. Moreover, if $M \to_\beta M'$ then $[\![M]\!] \twoheadrightarrow^{+0} [\![M']\!]$, and so we can conclude that the constructors of $\lambda\omega$ are *SN*. (One can even give a bound to the number of reduction steps starting from a constructor M of $\lambda\omega$ by taking the bound on the number of reduction steps in $\lambda\to$, starting from $[\![M]\!]$.) $\Box$

6 $\lambda\omega \models |SN| \Rightarrow \lambda\omega \models SN$

In this chapter it will be proved that for all $M \in object(\lambda\omega)$ $SN(|M|) \Rightarrow SN(M)$. This is done by analysing the different types of abstractions that can occur in an object of $\lambda\omega$, and showing that if there is an infinite reduction path starting from $M \in object(\lambda\omega)$, then there is an infinite reduction path starting from $|M| \in \Lambda$.

The advantage of this approach is that to prove strong normalization, we only have to check the erased terms. This makes the definition of the so called 'computability predicates' easier, because we do not have to take the type information into account.

Definition 71
Let $M \in Term(\lambda\omega)$. The *λ-decoration of* M, M^+, is defined by

(i) $u^+ = u$, for $u \in V_{AR}$,
(ii) $(\lambda u:A.t)^+ = \lambda_0 u:A.t$ if it is formed by rule $(*,*)$,
$(\lambda u:A.t)^+ = \lambda_2 u:A.t$ if it is formed by rule $(\Box,*)$,
$(\lambda u:A.t)^+ = \lambda_\omega u:A.t$ if it is formed by rule $(\Box,\Box)$,
(iii) $(PQ)^+ = P^+Q^+$.

Definition 72
The redexes in a term of $\lambda\omega$ are divided in three kinds, *0-redexes*, *2-redexes and ω-redexes*, by defining for $(\lambda u:A.t)q \in Term(\lambda\omega)$

(i) $(\lambda u:A.t)q$ is a *0-redex* iff $(\lambda u:A.t)^+ = \lambda_0 u:A.t$,
(ii) $(\lambda u:A.t)q$ is a *2-redex* iff $(\lambda u:A.t)^+ = \lambda_2 u:A.t$,
(iii) $(\lambda u:A.t)q$ is a *ω-redex* iff $(\lambda u:A.t)^+ = \lambda_\omega u:A.t$.

A reduction of one 0-redex will be called a *0-reduction-step*, a reduction of one 2-redex a *2-reduction-step*, and a reduction of one ω-redex a *ω-reduction-step*.

Note that a 0-redex is of the form $(\lambda x:A.t)q$, with A a type, q and t objects, a 2-redex is of the form $(\lambda\alpha:A.t)q$, with A a kind, q a constructor and t an object, and a ω-redex is of the form $(\lambda\alpha:A.t)q$, with A a kind, q and t constructors.

Lemma 73
For $M, M' \in object(\lambda\omega)$

(i) $M \to_2 M' \Rightarrow \#(\lambda_2 s \text{ in } M'^+) = \#(\lambda_2 s \text{ in } M^+) - 1$,
(ii) $M \to_\omega M' \Rightarrow \#(\lambda_2 s \text{ in } M'^+) = \#(\lambda_2 s \text{ in } M^+)$,
(iii) $M \to_2 M'$ *or* $M \to_\omega M' \Rightarrow |M| \equiv |M'|$.

Proof. Note that the only case in which a reduction does not decrease the number of λs by at least 1 is the following: $(\lambda u{:}A\,.\,t)\,q \to A[u := q]$, with u occurring in t more than once and a λ in the term q.

Further, we only have to compare the numbers of λs in the redex and its reduct; in the rest of the term everything remains the same.

(i) If $(\lambda\alpha{:}k\,.\,b)\,P \to_2 b[\alpha := P]$, then $P \in constructor(\lambda\omega)$, so in P no object-expressions occur and $\#(\lambda_2\text{s in } P^+) = 0$. We conclude that $\#(\lambda_2\text{s in } (b[\alpha := P])^+) = \#(\lambda_2\text{s in } b^+) = \#(\lambda_2\text{s in } ((\lambda\alpha{:}k\,.\,b)\,P)^+) - 1$.

(ii) If $(\lambda\alpha{:}k\,.\,Q)\,P \to_\omega Q[\alpha := P]$, then both $(\lambda\alpha{:}k\,.\,Q)\,P$ and $Q[\alpha := P]$ are constructors, so $\#(\lambda_2\text{s in } (Q[\alpha := P])^+) = 0 = \#(\lambda_2\text{s in } ((\lambda\alpha{:}k\,.\,Q)\,P)^+)$.

(iii) It is easy to see that an ω- or 2-reduction does not change the structure of the erased term, so if M' is obtained from M by a ω- or 2-reduction, then $|M| \equiv |M'|$. □

Note that (as Barendregt pointed out to us) Lemma 73(i) does not hold if we count the number of 2-redexes, e.g., in $(\lambda\alpha{:}*\,.\,\lambda\beta{:}*\,.\,t)\,\sigma\tau \to (\lambda\beta{:}*\,.\,t[\alpha := \sigma])\,\tau$ the number of 2-redexes is the same in both terms. In Lemma 73 the number of λs is used as an upperbound for the number of redexes.

Lemma 74
There is no infinite 2/ω reduction path in λω.

Proof. Suppose $M_1 \to M_2 \to M_3 \ldots$ is an infinite reduction path in which only 2- and ω-redexes are being reduced. With Corollary 70, we know that there is no tail $M_n, M_{n+1}, M_{n+2,\ldots}$ of this sequence such that $M_n \to_\omega M_{n+1} \to_\omega M_{n+2} \to_{\omega\ldots}$. So, after finitely many ω-reductions, a 2-reduction is performed, and there occur infinitely many 2-reduction-steps in the path. However, when we look at the decorated reduction path $M_1^+ \to M_2^+ \to M_3^+ \to \ldots$, a 2-reduction reduces the number of λ_2s by one, whereas an ω-reduction does not create any new λ_2s (Lemma 73). We conclude that there is no infinite 2/ω reduction path in λω. □

Theorem 75

$$\forall M \in object(\lambda\omega)\,.\,[SN(|M|) \Rightarrow SN(M)].$$

Proof. Suppose $M_1 \to M_2 \to M_3 \to \ldots$ is an infinite reduction path in object(λω). Then only finitely many 2/ω reduction steps are being performed after one another. The situation is $M_1 \twoheadrightarrow_{2/\omega} N_1 \twoheadrightarrow_0^{\neq 0} N_2 \twoheadrightarrow_{2/\omega} N_3 \twoheadrightarrow_0^{\neq 0} N_4 \twoheadrightarrow_{2/\omega\ldots}$, where all N_i are in the sequence $M_1, M_2, M_3, \ldots$ and $\twoheadrightarrow_0^{\neq 0}$ denotes a non-zero step reduction. We find that $|M_1| \equiv |N_1| \twoheadrightarrow_0^{\neq 0} |N_2| \equiv |N_3| \twoheadrightarrow_0^{\neq 0} |N_4| \equiv \ldots$, so there is an infinite reduction path in Λ, starting from $|M_1|$. We conclude that $SN(|M|) \Rightarrow SN(M)$ for all $M \in object(\lambda\omega)$. □

7 $\lambda\omega \models |SN|$

In the previous chapters it has been proved that the constructors of λω are strongly normalizing, and that the objects are strongly normalizing if the erased objects are.

Now, to finish the proof of $\lambda\omega \models SN$, the only thing left to do is to prove $SN(|M|)$ for all $M \in object(\lambda\omega)$. With the result of Chapter 4, we then have $\lambda P\omega \models SN$.

Strong normalization for $\lambda\omega$ was first proved by Girard (1972), using the notion of 'candidat de réductibilité'. For every type he defines a predicate on the terms of that type such that all terms satisfying the predicate are *SN*, and all terms of a type satisfy the corresponding predicate. Our task is a bit easier, as we only have to prove the strong normalization for the erased terms. The proof given below is a higher order version of the proof given for the system $\lambda\!\rightarrow$ in Chapter 5.

Definition 76
Let $k \in kind(\lambda\omega)$. The set of *computability predicates for k*, $CP(k)$, is defined by

(i) $CP(*) = \{X \subseteq \Lambda \mid X \text{ is saturated}\}$,
(ii) $CP(k_1 \rightarrow k_2) = \{f \mid f : CP(k_1) \rightarrow CP(k_2)\}$.

For the definition of saturated, see Definition 60. Notice that $CP(*) = E$ of Definition 60.

Lemma 77
For I a set and X_i saturated for all $i \in I$,

$$\bigcap_{i \in I} X_i \text{ is a saturated set.}$$

Proof. Straightforward, checking the properties of saturated set of Definition 60. □

Definition 78
Let $\Gamma \in Context(\lambda\omega)$. A *constructor valuation of* Γ is a map

$$\xi : VAR^{\Box} \rightarrow \bigcup_{k \in K} CP(k) \text{ such that } \forall \alpha \in VAR^{\Box}.[\alpha : k \in \Gamma \Rightarrow \xi(\alpha) \in CP(k)].$$

Notation. $\xi \models \Gamma$ if ξ is a constructor valuation of Γ.

Definition 79
Let $\Gamma \in Context(\lambda\omega)$, $\xi \models_{\lambda\omega} \Gamma$.
$[\![\,]\!]^{\xi} : \Gamma\text{-}constructor(\lambda\omega) \rightarrow \bigcup_{k \in K} CP(k)$ is defined inductively by

(i) $[\![\alpha]\!]^{\xi} = \xi(\alpha)$ if $\alpha \in VAR^{\Box}$,
(ii) $[\![PQ]\!]^{\xi} = [\![P]\!]^{\xi}[\![Q]\!]^{\xi}$ if PQ formed by $(\Box, \Box)$,
(iii) $[\![\lambda\alpha : k.Q]\!]^{\xi} = \lambda f \in CP(k).[\![Q]\!]^{\xi(\alpha := f)}$, if $\lambda\alpha : k.Q$ is formed by $(\Box, \Box)$,
(iv) $[\![\sigma \rightarrow \tau]\!]^{\xi} = [\![\sigma]\!]^{\xi} \rightarrow [\![\tau]\!]^{\xi}$ if $\sigma \rightarrow \tau$ is formed by $(*, *)$,
(v) $[\![\Pi\alpha : k.\sigma]\!]^{\xi} = \bigcap_{f \in CP(k)} [\![\sigma]\!]^{\xi(\alpha := f)}$ if $\Pi\alpha : k.\sigma$ is formed by $(\Box, *)$.

The definition is justified by the stripping lemma, $\forall f \in CP(k).\ \xi \models_{\lambda\omega} \Gamma \Rightarrow \xi(\alpha := f) \models_{\lambda\omega} \Gamma, \alpha : k$ and the following lemma, which shows that in the case of $\sigma \rightarrow \tau : *$, both $[\![\sigma]\!]^{\xi}$ and $[\![\tau]\!]^{\xi}$ are saturated sets, and so $[\![\sigma \rightarrow \tau]\!]^{\xi}$ is well defined.

Lemma 80
For $\Gamma \in Context(\lambda\omega)$, $\xi \models_{\lambda\omega} \Gamma$

(i) $t \in \Gamma\text{-}constructor(\lambda\omega), \Gamma \vdash_{\lambda\omega} t:k \Rightarrow [\![t]\!]^\xi \in CP(k)$,
(ii) $t, t' \in \Gamma\text{-}constructor(\lambda\omega), \alpha \in Var^\Box \Rightarrow [\![t[\alpha:=t']]\!]^\xi = [\![t]\!]^{\xi(\alpha:=[\![t']\!]^\xi)}$,
(iii) $t', t \in \Gamma\text{-}constructor(\lambda\omega), t = t' \Rightarrow [\![t]\!]^\xi = [\![t']\!]^\xi$.

Proof. Both (i) and (ii) by induction on the derivation of $\Gamma \vdash_{\lambda\omega} t:k$ for certain $k \in K$; (iii) immediately by the definition of $[\![\,]\!]^\xi$ and (ii). □

Definition 81
Let $\Gamma \in Context(\lambda\omega)$, $\xi \models \Gamma$. An *object valuation of* Γ *with respect to* ξ is a map $\rho: VAR^* \to \Lambda$ such that $\forall x \in VAR^*.\, x:\sigma \in \Gamma \Rightarrow \rho(x) \in [\![\sigma]\!]^\xi$.

Notation. $\rho, \xi \models_{\lambda\omega} \Gamma$ if $\xi \in \Gamma$ and ρ is an object valuation of Γ w.r.t. ξ.

Definition 82
Let $\Gamma \in Context(\lambda\omega)$, $\rho, \xi \models \Gamma$. $[\![\,]\!]_\rho : \Gamma\text{-}object(\lambda\omega) \to \Lambda$ is defined by

(i) $[\![x]\!]_\rho = \rho(x)$ if $x \in \mathrm{dom}(\Gamma)$,
(ii) $[\![\lambda x:A.t]\!]_\rho = \lambda x.[\![t]\!]_{\rho(x:=x)}$ if $\lambda x:A.t$ is formed by $(*,*)$,
(iii) $[\![\lambda\alpha:k.t]\!]_\rho = [\![t]\!]_\rho$ if $\lambda\alpha:k.t$ is formed by $(\Box,*)$,
(iv) $[\![t\,q]\!]_\rho = [\![t]\!]_\rho [\![q]\!]_\rho$ if $t\,q$ is formed by $(*,*)$,
(v) $[\![t\,Q]\!]_\rho = [\![t]\!]_\rho$ if $t\,Q$ is formed by $(\Box,*)$.

Note that for $M \in \Gamma\text{-}object(\lambda\omega)$, if $\vec{x}$ is the vector of free object variables, then $[\![M]\!]_\rho \equiv |M[\vec{x}:=\rho(\vec{x})]|$.

Definition 83
Let $\Gamma \in Context(\lambda\omega)$, $M \in object(\lambda\omega)$, $C \in type(\lambda\omega)$.

$$\Gamma \models_{\lambda\omega} M:C \quad \text{iff} \quad \forall \rho, \xi.[\rho, \xi \models_{\lambda\omega} \Gamma \Rightarrow [\![M]\!]_\rho \in [\![C]\!]^\xi].$$

Theorem 89
For $\Gamma \in Context(\lambda\omega)$, $M \in object(\lambda\omega)$, $C \in type(\lambda\omega)$

$$\Gamma \vdash_{\lambda\omega} M:C \Rightarrow \Gamma \models_{\lambda\omega} M:C.$$

Proof. By induction on the derivation of $\Gamma \vdash_{\lambda\omega} M:C$, distinguishing cases according to the structure of M. Let $\rho, \xi \models_{\lambda\omega} \Gamma$.

– $M \equiv x \in VAR^*$, then $[\![x]\!]_\rho \in [\![C']\!]^\xi$, with $x:C' \in \Gamma$ and $C' = C$. By Lemma 80 $[\![C']\!]^\xi = [\![C]\!]^\xi$ so $[\![x]\!]_\rho \in [\![C]\!]^\xi$.

– $M \equiv \lambda x:A.t$ formed by $(*,*)$, then $\Gamma, x:A \vdash_{\lambda\omega} t:B$, for certain B, with $\Pi x:A.B = C$. By induction hypothesis $[\![t]\!]_{\rho(x:=q)} \in [\![B]\!]^\xi$, for all $q \in [\![A]\!]^\xi$ and so $\lambda x.[\![t]\!]_\rho \in [\![A]\!]^\xi \to [\![B]\!]^\xi = [\![C]\!]^\xi$.

– $M \equiv \lambda a:k.t$, formed by $(\Box,*)$, then $\Gamma, \alpha:k \vdash_{\lambda\omega} t:B$, for certain B, with $\Pi\alpha:k.B = C$. By induction hypothesis $[\![t]\!]_\rho \in [\![B]\!]^{\xi(\alpha:=f)}$, for all $f \in CP(k)$, and so $[\![t]\!]_\rho \in \bigcap_{f \in CP(k)} [\![B]\!]^{\xi(\alpha:=f)}$

– $M \equiv t\,q$, formed by $(*,*)$, then $\Gamma \vdash_{\lambda\omega} t : A \to B$ and $\Gamma \vdash_{\lambda\omega} q : A$, for certain A, B with $B = C$. By induction hypothesis $[\![t]\!]_\rho \in [\![A \to B]\!]^\xi$ and $[\![q]\!]_\rho \in [\![A]\!]^\xi$, and so $[\![t\,q]\!]_\rho \in [\![B]\!]^\xi$.

– $M \equiv t\,Q$, formed by $(\Box,*)$, then $\Gamma \vdash_{\lambda\omega} t : \Pi\alpha{:}k.B$ and $\Gamma \vdash_{\lambda\omega} Q : k$, for some k, B with $B[\alpha := Q] = C$. By induction hypothesis, $[\![t]\!]_\rho \in \bigcap_{f \in CP(k)} [\![B]\!]^{\xi(\alpha:=f)}$ and $[\![Q]\!]^\xi \in CP(k)$.

We conclude that $[\![t]\!]_\rho \in [\![B]\!]^{\xi(\alpha:=[\![Q]\!]_\xi)} \equiv [\![B[\alpha := Q]]\!]^\xi$, and we are done. □

Theorem 85

$$\forall M \in object(\lambda\omega).\, SN(|M|).$$

Proof. Let $M \in object(\lambda\omega)$, $\Gamma \in Context(\lambda\omega)$, $C \in type(\lambda\omega)$ and $\Gamma \vdash_{\lambda\omega} M : C$. With Theorem 84, $\Gamma \models_{\lambda\omega} M : C$. Define canonical elements of the sets $CP(k)$ for every $k \in kind(\lambda\omega)$ by

$$c^* = SN,$$
$$c^{k_1 \to k_2} = \lambda f \in CP(k_1).\, c^{k_2}.$$

Then take a constructor valuation ξ of Γ, and an object valuation ρ of Γ w.r.t. ξ, such that, if $\alpha : k \in \Gamma$, then $\xi(\alpha) = c^k$, and if $x : A \in \Gamma$, then $\rho(x) = x$.

Now, $\rho, \xi \models_{\lambda\omega} \Gamma$, because all object variables are elements of all saturated sets. Further, $|M| = [\![M]\!]_\rho \in [\![C]\!]^\xi \subseteq SN$, so $SN(|M|)$. □

This is the final result to be proved in order to establish the strong normalization of the calculus of constructions.

References

Barendregt, H. P. 1984. *The Lambda calculus, its syntax and semantics*, North Holland.

Barendregt, H. P. 1989. Introduction to generalised type systems. *Proc. 3rd Italian Conference on Theoretical Computer Science*, World Scientific Publishing Co., Singapore.

Barendregt, H. P. 1990. Lambda calculi with types. In S. Abramsky, D. M. Gabbai and T. S. E. Maibaum (editors), *Handbook of Logic in Computer Science*. Oxford University Press.

Barendregt, H. P. and Dekkers, W. 1990. Typed Lambda Calculi.

Barendsen, E. 1989. *Representation of logic, data types and recursive functions in typed lambda calculi.* Masters Thesis, Faculty of Mathematics and Computer Science, University of Nijmegen, The Netherlands.

Berardi, S. 1988. *Towards a mathematical analysis of type dependence in Coquand-Huet calculus of constructions and the other systems in Barendregt's cube*. Department of Computer Science, CMU, and Dipartimento di Matematica, Torino.

de Bruijn, N. G. 1980. A survey of the project AUTOMATH. In J. R. Hindley and J. P. Seldin (editors), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 579–606. Academic Press.

Church, A. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic*, *5*: 56–68.

Coquand, T. 1986. *Metamathematical Investigations of a Calculus of Constructions*, INRIA, France and Cambridge University, UK.

Coquand, T. and Huet, G. 1985. Constructions: a higher order proof system for mechanizing mathematics. In B. Buchberger (editor), *EUROCAL 85*. Volume 203 of Lecture Notes in Computer Science, pp. 151–184.

Coquand, T. and Huet, G. 1988. The calculus of constructions. In A. R. Meyer (editors), *Information and Computation*, pp. 95–120.

Geuvers, H. 1988. *The interpretation of logics in type systems*. Masters Thesis, Faculty of Mathematics and Computer Science, Catholic University Nijmegen, The Netherlands.
Girard, J.-Y. 1972. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*, Thèse de Doctorat d'Etat, Université de Paris VIII, France.
Girard, J.-Y. 1971. Une extension de l'interprétation de Gödel à l'analyse, et son application a l'élimination des coupures dans l'analyse et la theorie des types. In J. E. Fenstad (editor), *Proceedings of the second Scandinavian Logic Symposium*. North Holland.
Harper, R., Honsell, F. and Plotkin, G. 1987. A framework for defining logics. *Proc. Symposium on Logic in Computer Science, Ithaca, New York*, IEEE, Washington.
Howard, W. 1980. The formulae-as-types notion of construction. In J. R. Hindley and J. P. Seldin (editors), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 479–490.
Luo, Z. 1988. $CC^{\infty}_{\subseteq}$ *and its strong normalization*. Notes of a talk given at the Jumelage meeting on typed Lambda Calculus, Nijmegen, The Netherlands.
Tait, W. W. 1967. Intensional interpretation of functions of finite type. *Journal of Symbolic Logic*, *32*: 198–212.
Tait, W. W. 1975. A realizability interpretation of the theory of species. In R. Parikh (editor), *Logic Colloquium*. Volume 453 of *Lecture Notes in Mathematics*, pp. 240–51. Springer-Verlag.
Terlouw, J. 1989*a*. *Een nadere bewijstheoretische analyse van GSTT's*. Internal report, Faculty of Mathematics and Computer Science, University of Nijmegen, The Netherlands.
Terlouw, J. 1989*b*. *Sterke normalisatie in C à la Tait*. Notes of a lecture held at the *Intercity Seminar Typed Lambda Calculus*, Nijmegen, The Netherlands.

Herman Geuvers and Mark-Jan Nederhof, Faculty of Mathematics and Computer Science University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands.