Pure type systems with definitions

Citation for published version (APA):

Severi, P. G., & Poll, E. (1993). *Pure type systems with definitions*. (Computing science notes; Vol. 9324). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/1993

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Eindhoven University of Technology
Department of Mathematics and Computing Science

Pure Type Systems with definitions

by

Paula Severi and Erik Poll

93/24

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

All rights reserved
editors: prof.dr.M.Rem
 prof.dr.K.M.van Hee.

Abstract

In this paper, an extension of Pure Type Systems (PTS's) to include definitions is presented and the meta-theory of these PTS's with definitions is treated in detail. We prove that all the properties of a PTS are preserved by the extension.

Chapter 1

Introduction

In this paper, the definitions and notations of [Bar92] will be used. In particular, we assume that the reader is familiar with the concepts of λ -cube and Pure Type Systems (PTS's) as defined in [Bar92]. The λ -cube forms a natural framework in which several systems *à la* Church are given in an uniform way. The description method of the systems in the λ -cube is generalised in the definition of a PTS. A PTS does not provide the possibility to introduce a definition, i.e. an abbreviation (name) for a larger term which can be used several times in a program or proof. In this paper, an extension of the PTS that includes definitions will be introduced.

— o —

Definitions are names abbreviating expressions. In $(x=a:A \text{ in } b)$ the identifier x abbreviates the expression a of type A in the expression b . We assume that x does not occur either in a or in A . In other words, it is not a recursive definition.

Example 1.1. $id = (\lambda x:\alpha. x):\alpha \rightarrow \alpha \text{ in } (\lambda y:\alpha \rightarrow \alpha. id) \text{ id}.$

The name id is an abbreviation for a larger term, i.e. $\lambda x:\alpha. x$ and is used several times in the expression $(\lambda y:\alpha \rightarrow \alpha. id) \text{ id}.$

Example 1.2.

1. Given $\alpha, \beta : *$

$$\wedge = (\lambda \alpha : *. \lambda \beta : *. (\Pi \gamma : *. (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow \gamma)) : (* \rightarrow * \rightarrow *) \text{ in } (\alpha \wedge \beta) \rightarrow \alpha$$

2. Given $\alpha : *$

$$\perp = (\Pi \alpha : *. \alpha) : * \text{ in } (\neg = (\lambda \alpha : *. \alpha \rightarrow \perp) : * \rightarrow * \text{ in } (\alpha \rightarrow \neg \alpha))$$

The intended meaning of a definition is that the *definiendum* x can be substituted by the *definiens* a in the expression b . A definition $(x=a:A \text{ in } b)$ can be considered as having a similar behaviour than $(\lambda x:A. b)a$, i.e. the substitution of the variable x by a in the expression b . In the β -reduction where $(\lambda x:A. b)a$ reduces to $b[x := a]$, the operation $b[x := a]$ is the substitution of **all the occurrences** of x by a in the expression b . In contrast to β -reduction, the expression $(x=a:A \text{ in } b)$ reduces to an expression which has at most **one occurrence** of x unfolded by a . In order to unfold definitions, a new relation called δ -reduction is introduced. We will show that a definition $(x=a:A \text{ in } b)$ has the same behaviour as $(\lambda x:A. b)a$ in the sense that the definition $(x=a:A \text{ in } b)$ δ -reduces in *several steps* to $b[x := a]$. Then the unfolding of definitions that is made via the δ -reduction corresponds to the operation of substitution $b[x := a]$.

A definition $(x=a:A \text{ in } b)$ is not another way of writing $(\lambda x:A. b)a$. There are important differences between $(x=a:A \text{ in } b)$ and $(\lambda x:A. b)a$, both regarding their reduction behaviour and their typing. One of the reasons for considering $(x=a:A \text{ in } b)$ and not $(\lambda x:A. b)a$ is that in some cases it is convenient to have the freedom of substituting only in **some of the occurrences** of

an expression in a given formula.

Another reason for considering $(x=a:A \text{ in } b)$ and not $(\lambda x:A. b)a$ is that in $(x=a:A \text{ in } b)$ the fact that x is an abbreviation for a can be used to type b . This is shown in the following example.

Example 1.3. The term $\lambda\alpha:*. (x=\alpha:*\text{ in } \lambda y:x. \lambda f:\alpha \rightarrow \alpha. fy)$ is typable in the system $\lambda 2$ extended with definitions. But it is not possible to type the corresponding term expressed with an application and an abstraction in any system of the λ -cube,

$$\lambda\alpha:*. (\lambda x:*. \lambda y:x. \lambda f:\alpha \rightarrow \alpha. fy)\alpha$$

A third reason for considering $(x=a:A \text{ in } b)$ instead of $(\lambda x:A. b)a$ is that the abstraction $(\lambda x:A. b)$ may not be allowed when $(x=a:A \text{ in } b)$ is.

Example 1.4. The term $(x=\alpha:*\text{ in } \lambda y:x. \lambda f:x \rightarrow x. fy)$ is typable in the system $\lambda_{\rightarrow}$ extended with definitions. The corresponding term expressed with an application and an abstraction, i.e. $(\lambda x:*. \lambda y:x. \lambda f:x \rightarrow x. fy)\alpha$, is not typable in $\lambda_{\rightarrow}$ but it can be typable in $\lambda 2$.

The fact that the type A of a is explicitly written in $(x=a:A \text{ in } b)$ is not essential, at least for PTS's where types are unique (up to β -equality). We want to record this type A in the case that we consider A as a proposition.

— o —

We present a short description of the paper.

In **Chapter 2**, we recall the definition of Pure Type Systems (PTS's) and their main properties from [Bar92].

In **Chapter 3**, Pure Type Systems with definitions (DPTS's) are introduced. The set of terms and contexts are extended to include definitions. A new relation $\rightarrow_{\delta}$ is introduced that corresponds to the unfolding of a definition.

In **Chapter 4**, Church-Rosser property for $\rightarrow_{\beta\delta}$ and strong normalization property for $\rightarrow_{\delta}$ are proved. All the properties in this chapter, in particular strong normalization for $\rightarrow_{\delta}$, hold for all the pseudoterms, not only for well-typed terms.

In **Chapter 5**, the meta-theory of DPTS's is treated. All properties of PTS's with the exception of strong normalization are preserved by the extension. We prove that strong normalization is preserved for some PTS's such as the examples of PTS's given in [Bar92].

In **Appendix A**, a summary of the definitions of DPTS's is presented.

In **Appendix B**, the main features of the proofs of Church-Rosser for δ and $\beta\delta$ -reductions, weak normalization for $\rightarrow_{\delta}$ and strong normalization for $\rightarrow_{\beta\delta}$ are abstracted and presented for Abstract Reduction Systems.

Chapter 2

Pure Type Systems

In this chapter the concept of Pure Type System (PTS) is defined as in [Bar92]. Only the notion of topsort and the system λC_∞ (see definitions 2.11 and 2.14) are not defined in [Bar92].

Definition 2.1. *The set T of pseudoterms and the set C of contexts are defined as follows:*

$$\begin{aligned} T &::= V \mid C \mid (T \ T) \mid (\lambda V:T. T) \mid (\Pi V:T. T) \\ C &::= \epsilon \mid \langle C, V:T \rangle \end{aligned}$$

where V is the set of variables and C is the set of constants.

Convention 2.2. The following convention will be used in this paper:

1. Variables will be denoted as $x, y, z, \dots, \alpha, \beta, \gamma, \dots$
 Contexts will be denoted as $\Gamma, \Gamma', \dots, \Delta, \Delta', \dots$
 Terms will be denoted as $a, b, d, \dots, A, B, C, \dots$
 Constants will be denoted as $c, c', \dots$
2. The usual parenthesis conventions for abstraction, application and product will be used (see [Bar92]).

Definition 2.3. *Let be $a, a' \in T$. The concept "the term a β -reduces to the term a' " is written as $a \rightarrow_\beta a'$ and is defined by $(\lambda x:A. a)b \rightarrow_\beta a[x := b]$ and the compatibility rules.*

The α -equality is defined as usual. α -equal terms are identified.

Definition 2.4. *The specification of Pure Type System (PTS) is a triple $S = (S, A, R)$ such that*

- $S \subseteq C$ is the set of sorts.
- $A \subseteq C \times S$ is the set of axioms
- $R \subseteq S \times S \times S$ is the set of rules

Definition 2.5. *The PTS determined by the specification $S = (S, A, R)$ is denoted as $\lambda S = \lambda(S, A, R)$ and defined by the notion of type derivation $\Gamma \vdash_{\lambda S} b : B$ (or $\Gamma \vdash b : B$) given by the following axioms and rules:*

$$\begin{array}{lll} \text{(axiom)} & \epsilon \vdash c : s & \text{for } c : s \in A \\ \text{(start)} & \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} & \text{where } x \text{ is } \Gamma\text{-fresh} \end{array}$$

$$\begin{array}{ll}
\text{(weakening)} & \frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B} \quad \text{where } x \text{ is } \Gamma\text{-fresh} \\
\text{(formation)} & \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x:A. B) : s_3} \quad \text{for } (s_1, s_2, s_3) \in R \\
\text{(abstraction)} & \frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x:A. B) : s}{\Gamma \vdash (\lambda x:A. b) : (\Pi x:A. B)} \\
\text{(application)} & \frac{\Gamma \vdash b : (\Pi x:A. B) \quad \Gamma \vdash a : A}{\Gamma \vdash (b \ a) : B[x := a]} \\
\text{(conversion)} & \frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash b : B'}
\end{array}$$

where s ranges over sorts, i.e. $s \in S$.

Theorem 2.6. (Church Rosser for β -reduction) Let $\Gamma \in \mathcal{C}$ and $a \in \mathcal{T}$ be such that $a \rightarrow_\beta b$ and $a \rightarrow_\beta c$. Then there exists a term $d \in \mathcal{T}$ such that $b \rightarrow_\beta d$ and $c \rightarrow_\beta d$.

The following properties are in [Bar92].

Theorem 2.7. (Correctness of Types) Let $\Gamma \in \mathcal{C}$ and $d, d', D \in \mathcal{T}$ be such that $\Gamma \vdash d : D$. Then $\Gamma \vdash D : s$ or $D = s$.

Theorem 2.8. (Subject Reduction Theorem) Let $\Gamma \in \mathcal{C}$ and $d, d', D \in \mathcal{T}$ be such that $\Gamma \vdash d : D$. If $d \rightarrow_\beta d'$ then $\Gamma \vdash d' : D$.

Definition 2.9. The specification $S = (S, A, R)$ is called singly sorted if

1. $(c : s_1), (c : s_2) \in A$ implies $s_1 \equiv s_2$
2. $(s_1, s_2, s_3), (s_1, s_2, s'_3) \in R$ implies $s_3 \equiv s'_3$

Theorem 2.10. (Uniqueness of Types) Let S be a singly sorted specification, $\Gamma \in \mathcal{C}$ and $a, A, B \in \mathcal{T}$ such that $\Gamma \vdash a : A$ and $\Gamma \vdash a : B$. Then $A =_\beta B$.

Definition 2.11. Let λS be a PTS. A sort s in S is called a topsort if there is no $s_0 \in S$ such that $\Gamma \vdash s : s_0$.

Definition 2.12. The λ -cube is a cube of eight systems defined by the same set of sorts $S = \{*, \square\}$ and the same set of axioms $A = \{* : \square\}$. They differ in the set of rules R .

System	R
$\lambda_{\rightarrow}$	$(*, *)$
λ_2	$(*, *) \quad (\square, *)$
λ_P	$(*, *) \quad (*, \square)$
λ_{P2}	$(*, *) \quad (\square, *) \quad (*, \square)$
λ_{ω}	$(*, *) \quad (\square, \square)$
λ_{ω}	$(*, *) \quad (\square, *) \quad (\square, \square)$
$\lambda_{P\omega}$	$(*, *) \quad (*, \square) \quad (\square, \square)$
$\lambda_{P\omega} = \lambda_C$	$(*, *) \quad (\square, *) \quad (*, \square) \quad (\square, \square)$

The rule (s_1, s_2) is an abbreviation for (s_1, s_2, s_3) .

The system λ_C is the Calculus of Constructions.
There is only one topsort in the λ -cube and that is $\square$.

Definition 2.13. The system of higher order logic can be described by the following PTS:

$$\lambda\mathbf{HOL} \quad \boxed{\begin{array}{ll} \mathbf{S} & *, \square, \Delta \\ \mathbf{A} & * : \square, \square : \Delta \\ \mathbf{R} & (*, *), (\square, *), (\square, \square) \end{array}}$$

There is only one topsort in $\lambda\mathbf{HOL}$ and that is Δ .

Definition 2.14. The Calculus of Constructions extended with an infinite type hierarchy can be described by the following PTS:

$$\lambda C_{\infty} \quad \boxed{\begin{array}{ll} \mathbf{S} & \mathbb{N} \\ \mathbf{A} & \forall n \in \mathbb{N} \quad n : n + 1 \\ \mathbf{R} & \forall m \in \mathbb{N} \quad (m, 0, 0) \text{ \& } \\ & \forall m, n \in \mathbb{N} \quad \max(m, n) \leq r \quad (m, n, r) \end{array}}$$

The system λC_{∞} extended with strong Σ -types and cumulativity is the system **ECC**(see [Luo]). We can see that λC_{∞} is an extension of λC and of $\lambda\mathbf{HOL}$ writing $*$ instead of 0 , $\square$ instead of 1 and Δ instead of 3 .

Note that there is no topsort in λC_{∞} .

Definition 2.15. Let λS be a PTS. Then λS is β -strongly normalizing if a and A β -strongly normalize for all $a, A \in T$ and $\Gamma \in C$ such that $\Gamma \vdash_{\lambda S} a : A$.

Theorem 2.16.

1. The system λC_{∞} is β -strongly normalizing.
2. The systems of the λ -cube are β -strongly normalizing.
3. The system $\lambda\mathbf{HOL}$ is β -strongly normalizing.

Proof: The system **ECC** is strongly normalizing(see [Luo]) and contains λC_{∞} , the systems of the λ -cube and $\lambda\mathbf{HOL}$. □

Not all PTS's are strongly normalizing.

Example 2.17. The PTS $\lambda*$ determined by the specification (S, A, R) where $S = \{*\}$, $A = \{* : *\}$ and $R = \{(*, *)\}$ is not β -strongly normalizing.

Chapter 3

Pure Type Systems with definitions

In Section 3.1, the set of pseudoterms T and the set of contexts C defined in chapter 2 are extended to include definitions.

In Section 3.2, a new relation $\rightarrow_\delta$ between pseudoterms that corresponds to the unfolding of a definition is introduced. Also, the relation $\rightarrow_\beta$ is extended to the new pseudoterms.

In Section 3.3, Pure Type Systems with definitions (DPTS's) are introduced by extending PTS's with the new pseudoterms and contexts.

3.1 Pseudoterms

Definition 3.1.1. *The set T_δ of pseudoterms is given by*

$$T_\delta ::= V \mid C \mid (T_\delta \ T_\delta) \mid (\lambda V:T_\delta. T_\delta) \mid (\Pi V:T_\delta. T_\delta) \mid (V=T_\delta:T_\delta \text{ in } T_\delta)$$

where V is the set of variables and C is the set of constants.

The mapping $FV : T \rightarrow P(V)$ defined in [Bar92] is extended to T_δ .

Definition 3.1.2. *The mapping $FV : T_\delta \rightarrow P(V)$ is defined as follows:*

$$FV(x) = \{x\}$$

$$FV(c) = \emptyset$$

$$FV(a \ b) = FV(a) \cup FV(b)$$

$$FV(\lambda x:A. a) = FV(A) \cup (FV(a) - \{x\})$$

$$FV(\Pi x:A. a) = FV(A) \cup (FV(a) - \{x\})$$

$$FV(x=a:A \text{ in } b) = FV(A) \cup FV(a) \cup (FV(b) - \{x\})$$

Definition 3.1.3. *The set C_δ of contexts is given by*

$$i) \epsilon \in C_\delta$$

$$ii) \langle \Gamma, x:A \rangle \in C_\delta \text{ if } \Gamma \in C_\delta, x \in V, A \in T_\delta \text{ and } x \text{ is } \Gamma\text{-fresh}$$

$$iii) \langle \Gamma, x=a:A \rangle \in C_\delta \text{ if } \Gamma \in C_\delta, x \in V, a \in T_\delta, A \in T_\delta, x \text{ is } \Gamma\text{-fresh and } x \notin FV(a) \cup FV(A)$$

Note that the set of contexts is not given by $C_\delta ::= \epsilon \mid C_\delta, V : T_\delta \mid C_\delta, V = T_\delta : T_\delta$. In order to avoid capture of bound variables in the definition of δ -reduction given below a new variable is introduced to a context Γ only if it is Γ -fresh, i.e. it does not occur in Γ .

It is said that the variable x *abbreviates* the expression a in $(x=a:A \text{ in } b)$ and in $\Gamma, x=a:A$.

Convention 3.1.4. The following convention about contexts will be used in this paper:

1. The expression $\Gamma, x:A$ stands for the context $\langle \Gamma, x:A \rangle$.
2. The expression $\Gamma, x=a:A$ stands for the context $\langle \Gamma, x=a:A \rangle$.

Definition 3.1.5. A mapping $Dom : \mathcal{C}_\delta \rightarrow P(V)$ is defined as follows:

$$Dom(\epsilon) = \emptyset$$

$$Dom(\Gamma, x:A) = Dom(\Gamma) \cup \{x\}$$

$$Dom(\Gamma, x=a:A) = Dom(\Gamma) \cup \{x\}$$

Definition 3.1.6. A mapping $Def : \mathcal{C}_\delta \rightarrow P(V)$ is defined as follows:

$$Def(\epsilon) = \emptyset$$

$$Def(\Gamma, x:A) = Def(\Gamma)$$

$$Def(\Gamma, x=a:A) = Def(\Gamma) \cup \{x\}$$

Substitution is extended to pseudoterms $\mathcal{T}_\delta$ and to contexts $\mathcal{C}_\delta$.

Definition 3.1.7. Let $M, N \in \mathcal{T}_\delta$, $x \in V$. The result of substituting N for (the free occurrences of) x in M is denoted as $M[x := N]$ and is defined as follows:¹

$$\begin{array}{ll}
c[x := N] & \equiv c & c \in \mathcal{C} \\
x[x := N] & \equiv N \\
y[x := N] & \equiv y \\
(\lambda x:A. a)[x := N] & \equiv (\lambda x:A. a) \\
(\lambda y:A. a)[x := N] & \equiv (\lambda y:A[x := N]. a[x := N]) & y \notin FV(N) \\
(\lambda y:A. a)[x := N] & \equiv (\lambda z:A[y := z][x := N]. a[y := z][x := N]) & y \in FV(N) \\
(a \ b)[x := N] & \equiv (a[x := N] \ b[x := N]) \\
(\Pi x:A. a)[x := N] & \equiv (\Pi x:A. a) \\
(\Pi y:A. a)[x := N] & \equiv (\Pi y:A[x := N]. a[x := N]) & y \notin FV(N) \\
(\Pi y:A. a)[x := N] & \equiv (\Pi z:A[y := z][x := N]. a[y := z][x := N]) & y \in FV(N) \\
(x=a:A \text{ in } b)[x := N] & \equiv (x=a:A \text{ in } b) \\
(y=a:A \text{ in } b)[x := N] & \equiv (y=a[x := N]:A[x := N] \text{ in } b[x := N]) & y \notin FV(N) \\
(y=a:A \text{ in } b)[x := N] & \equiv (z=a[x := N]:A[x := N] \text{ in } b[y := z][x := N]) & y \notin FV(N)
\end{array}$$

Definition 3.1.8. The result of substituting N for (the free occurrences of) a variable x in Γ such that $x \notin Dom(\Gamma)$ is denoted as $\Gamma[x := N]$ and is defined as follows:

$$\begin{array}{ll}
\epsilon[x := N] & \equiv \epsilon \\
\langle \Gamma, y:A \rangle [x := N] & \equiv \langle y:A[x := N], \Gamma[x := N] \rangle \\
\langle \Gamma, y=a:A \rangle [x := N] & \equiv \langle y=a[x := N]:A[x := N], \Gamma[x := N] \rangle
\end{array}$$

The definitions of change of bound variables and α -congruence in λ -terms are extended to pseudoterms in $\mathcal{T}_\delta$.

Definition 3.1.9. Let $t \in \mathcal{T}_\delta$ be such that t contains the term $(x=a:A \text{ in } b)$. A change of a bound variable in the term t is the replacement of a subterm $(x=a:A \text{ in } b)$, $(\lambda x:A. b)$ or $(\Pi y:A. b)$ by $(y=a:A \text{ in } b[x := y])$, $(\lambda x:A. b[x := y])$ or $(\Pi y:A. b[x := y])$, respectively, where $y \notin FV(b)$.

The relation of α -equality between λ -terms is extended to pseudoterms $\mathcal{T}_\delta$.

Definition 3.1.10. Let $\Gamma \in \mathcal{C}_\delta$ and $t, t' \in \mathcal{T}_\delta$, the term t is α -equal to t' (written $t =_\alpha t'$) if t' is the result of applying to t a series of changes of variables or viceversa.

Convention 3.1.11. Two terms are identified if they are α -equal.

¹ In this definition the variable y is different from x and z is fresh.

3.2 Reductions

The relation $\rightarrow_\beta$ between terms in $\mathcal{T}$ will be extended to terms in $\mathcal{T}_\delta$.

Definition 3.2.1. Let $a, a' \in \mathcal{T}_\delta$. The concept "the term a β -reduces to the term a' " is written as $a \rightarrow_\beta a'$ and is defined as usual by the following rules

$$\begin{array}{c}
 (\lambda x:A. a)b \rightarrow_\beta a[x := b] \\
 \hline
 \frac{b \rightarrow_\beta b'}{(x=a:A \text{ in } b) \rightarrow_\beta (x=a:A \text{ in } b')} \\
 \hline
 \frac{a \rightarrow_\beta a'}{(x=a:A \text{ in } b) \rightarrow_\beta (x=a':A \text{ in } b)} \\
 \hline
 \frac{A \rightarrow_\beta A'}{(x=a:A \text{ in } b) \rightarrow_\beta (x=a:A' \text{ in } b)} \\
 \hline
 \frac{a \rightarrow_\beta a'}{(a \ b) \rightarrow_\beta (a' \ b)} \qquad \frac{b \rightarrow_\beta b'}{(a \ b) \rightarrow_\beta (a \ b')} \\
 \hline
 \frac{a \rightarrow_\beta a'}{(\lambda x:A. a) \rightarrow_\beta (\lambda x:A. a')} \qquad \frac{A \rightarrow_\beta A'}{(\lambda x:A. a) \rightarrow_\beta (\lambda x:A'. a)} \\
 \hline
 \frac{a \rightarrow_\beta a'}{(\Pi x:A. a) \rightarrow_\beta (\Pi x:A. a')} \qquad \frac{A \rightarrow_\beta A'}{(\Pi x:A. a) \rightarrow_\beta (\Pi x:A'. a)}
 \end{array}$$

A new relation $\rightarrow_\delta \subseteq \mathcal{C}_\delta \times \mathcal{T}_\delta \times \mathcal{T}_\delta$ is introduced.

Definition 3.2.2. Let $\Gamma \in \mathcal{C}_\delta$, and $a, a' \in \mathcal{T}_\delta$. The concept "the term a δ -reduces to the term a' in the context Γ " is written as $\Gamma \vdash a \rightarrow_\delta a'$ and is defined by the following rules:

$$\begin{array}{c}
 \Gamma_1, x=a:A, \Gamma_2 \vdash x \rightarrow_\delta a \\
 \Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta b \qquad \text{if } x \notin FV(b) \\
 \hline
 \frac{\Gamma, x=a:A \vdash b \rightarrow_\delta b'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A \text{ in } b')} \\
 \hline
 \frac{\Gamma \vdash a \rightarrow_\delta a'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a':A \text{ in } b)} \qquad \frac{\Gamma \vdash A \rightarrow_\delta A'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A' \text{ in } b)} \\
 \hline
 \frac{\Gamma \vdash a \rightarrow_\delta a'}{\Gamma \vdash (a \ b) \rightarrow_\delta (a' \ b)} \qquad \frac{\Gamma \vdash b \rightarrow_\delta b'}{\Gamma \vdash (a \ b) \rightarrow_\delta (a \ b')} \\
 \hline
 \frac{\Gamma, x:A \vdash a \rightarrow_\delta a'}{\Gamma \vdash (\lambda x:A. a) \rightarrow_\delta (\lambda x:A. a')} \qquad \frac{\Gamma \vdash A \rightarrow_\delta A'}{\Gamma \vdash (\lambda x:A. a) \rightarrow_\delta (\lambda x:A'. a)} \\
 \hline
 \frac{\Gamma, x:A \vdash a \rightarrow_\delta a'}{\Gamma \vdash (\Pi x:A. a) \rightarrow_\delta (\Pi x:A. a')} \qquad \frac{\Gamma \vdash A \rightarrow_\delta A'}{\Gamma \vdash (\Pi x:A. a) \rightarrow_\delta (\Pi x:A'. a)}
 \end{array}$$

Note that the first rule allows to unfold definitions, the second rule allows to remove definitions and the remaining are compatibility rules. In a context $\Gamma, x=a:A$ a term d can be δ -reduced to a term d' where d' is obtained replacing one occurrence of x by a in d .

When Γ is the empty context, $a \rightarrow_\delta a'$ is written instead of $\Gamma \vdash a \rightarrow_\delta a'$.

Example 3.2.3.

1. Assume $\Gamma \equiv A : *$.

$$\begin{array}{lcl} \Gamma \vdash & (id = (\lambda y:A. y) : (\Pi y:A. A)) \text{ in } & (\lambda x:(\Pi y:A. A). \text{ id }) \quad id \quad \rightarrow_\delta \\ & (id = (\lambda y:A. y) : (\Pi y:A. A)) \text{ in } & (\lambda x:(\Pi y:A. A). (\lambda y:A. y)) \quad id \quad \rightarrow_\delta \\ & (id = (\lambda y:A. y) : (\Pi y:A. A)) \text{ in } & (\lambda x:(\Pi y:A. A). (\lambda y:A. y)) \quad (\lambda y:A. y) \quad \rightarrow_\delta \\ & & (\lambda x:(\Pi y:A. A). (\lambda y:A. y)) \quad (\lambda y:A. y) \end{array}$$

2. Assume $\Gamma \equiv < A : *, B : *, C : *, \rightarrow = \lambda\alpha:*. \lambda\beta:*. (\Pi x:\alpha. \beta) : \Pi\alpha:*. (\Pi\beta:*. *) ,$
 $K = (\lambda x:A. \lambda y:B. x) : (A \rightarrow B \rightarrow A),$
 $S = (\lambda x:(A \rightarrow B \rightarrow C). \lambda y:(A \rightarrow B). \lambda z:A. x \ z \ (y \ z)):$
 $((A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C)) >$

$$\begin{array}{lcl} \Gamma \vdash & (S \quad K \quad K) & \rightarrow_\delta \\ & (S \quad K \quad (\lambda x:A. \lambda y:B. x)) & \rightarrow_\delta \\ & (S \quad (\lambda x:A. \lambda y:B. x) \quad (\lambda x:A. \lambda y:B. x)) & \end{array}$$

$A \rightarrow B$ is written instead of $((\rightarrow A)B)$

The examples above show that in each step of a δ -reduction, exactly one occurrence of a variable is unfolded. The first example also shows that the definition is removed in the last step of δ -reductions .

The next example shows that α -conversion is necessary when rewriting terms.

Example 3.2.4.

$$\begin{array}{l} x=y:A \vdash \quad (y=z:u \text{ in } x) \\ \quad =_\alpha (y'=z:u \text{ in } x) \\ \quad \rightarrow_\delta (y'=z:u \text{ in } y) \end{array}$$

The variable y occurs in the term $(y=z:u \text{ in } x)$ and in the context $x=y:A$.

Convention 3.2.5. From now on, the bound variables in the term will be assumed different from the variables in the context.

Definition 3.2.6. Let $\Gamma \in \mathcal{C}_\delta$ and $a, a' \in \mathcal{T}_\delta$. Then

$$\Gamma \vdash a \rightarrow_{\beta\delta} a' \text{ if } \Gamma \vdash a \rightarrow_\delta a' \text{ or } a \rightarrow_\beta a'.$$

Definition 3.2.7. The relation $\rightarrow_\rho^\equiv$ is the reflexive closure of $\rightarrow_\rho$. The relation $\rightarrow_\rho^+$ is the transitive closure of the reduction relation $\rightarrow_\rho$. The relation $\twoheadrightarrow_\rho$ is the transitive, reflexive closure of the reduction relation $\rightarrow_\rho$. The relation $=_\rho$ is the equivalence relation generated by $\twoheadrightarrow_\rho$.

Note that for $R \in \{\twoheadrightarrow_\delta, =_\delta, \twoheadrightarrow_{\beta\delta}, =_{\beta\delta}\}$ a context has to be specified, i.e. $\Gamma \vdash a \ R \ b$ for $a, b \in \mathcal{T}_\delta$ and $\Gamma \in \mathcal{C}_\delta$.

3.3 Types

We extend the notion of Pure Type System to include definitions.

Definition 3.3.1. The DPTS determined by the specification $S = (S, A, R)$ is denoted as $\lambda S_\delta = \lambda(S, A, R)_\delta$ and defined by the notion of type derivation $\Gamma \vdash_{\lambda S_\delta} b : B$ (or just $\Gamma \vdash b : B$) with $\Gamma \in \mathcal{C}_\delta$ and $b, B \in \mathcal{T}_\delta$ given by the following axioms and rules:

(axiom)	$\frac{}{c \vdash c : s}$	for $c : s \in A$
(start)	$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$	where x is Γ -fresh
(weakening)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B}$	where x is Γ -fresh
(formation)	$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x : A. B) : s_3}$	for $(s_1, s_2, s_3) \in R$
(abstraction)	$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A. B) : s}{\Gamma \vdash (\lambda x : A. b) : (\Pi x : A. B)}$	
(application)	$\frac{\Gamma \vdash b : (\Pi x : A. B) \quad \Gamma \vdash a : A}{\Gamma \vdash (b \ a) : B[x := a]}$	
(conversion)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash b : B'}$	
(δ -start)	$\frac{\Gamma \vdash a : A}{\Gamma, x = a : A \vdash x : A}$	where x is Γ -fresh
(δ -weakening)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash a : A}{\Gamma, x = a : A \vdash b : B}$	where x is Γ -fresh
(δ -formation)	$\frac{\Gamma, x = a : A \vdash B : s}{\Gamma \vdash (x = a : A \text{ in } B) : s}$	
(δ -introduction)	$\frac{\Gamma, x = a : A \vdash b : B \quad \Gamma \vdash (x = a : A \text{ in } B) : s}{\Gamma \vdash (x = a : A \text{ in } b) : (x = a : A \text{ in } B)}$	
(δ -conversion)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad \Gamma \vdash B =_\delta B'}{\Gamma \vdash b : B'}$	

where s ranges over sorts, i.e. $s \in S$.

The condition " x is Γ -fresh" in δ -start and δ -weakening rules could be omitted because of convention 3.1.11.

Observe that the new rules are: δ -start, δ -weakening, δ -formation, δ -introduction and δ -conversion. The system λS_δ is an extension of λS .

Observe that in a DPTS, only the typable terms can be abbreviated. This is because the δ -start and δ -weakening rules allow the abbreviation of the term a in the context $\Gamma, x = a : A$ only if $\Gamma \vdash a : A$. Hence it is not possible to abbreviate any topsort. For example in the λ -cube it is not possible to abbreviate $\square$.

Observe that if B is a topsort then $(x = a : A \text{ in } B)$ is neither typable nor the type of another definition $(x = a : A \text{ in } b)$. This is because the δ -formation rule requires that $\Gamma, x = a : A \vdash B : s$ and because the δ introduction rule requires that $\Gamma \vdash (x = a : A \text{ in } B) : s$. For example $(x = a : A \text{ in } \square)$ is not typable in λC_δ and it is not a type of $(x = a : A \text{ in } *)$. The condition " $\Gamma \vdash (x = a : A \text{ in } B) : s$ " in the premises of the δ -introduction rule is necessary to ensure Subject Reduction Theorem. Since we add this condition we must also add δ -formation rule. We call $(\lambda S)_\delta^+$ the system obtained from λS_δ by removing the δ -formation rule and the condition " $\Gamma \vdash (x = a : A \text{ in } B) : s$ " from the premises of the δ -introduction rule. Subject Reduction Theorem does not hold in $(\lambda S)_\delta^+$ as the following example shows.

Example 3.3.2. Suppose $\Gamma \vdash a : A$. The term $(x = a : A \text{ in } *)$ has type $(x = a : A \text{ in } \square)$ in $(\lambda C)_\delta^+$. This term δ -reduces to $*$ but $*$ does not have type $(x = a : A \text{ in } \square)$. In order to derive $\Gamma \vdash * : (x = a : A \text{ in } \square)$ we apply δ -conversion rule

$$\frac{\Gamma \vdash * : \square \quad \Gamma \vdash (x = a : A \text{ in } \square) : s \quad \Gamma \vdash (x = a : A \text{ in } \square) =_\delta \square}{\Gamma \vdash * : (x = a : A \text{ in } \square)}$$

But $(x=a:A \text{ in } \Box)$ is not typable because $\Box$ is not.

Example 3.3.3.

- Let $\Gamma_1 \equiv < \alpha : *, ID = (\Pi x:\alpha. \alpha) : *, id = (\lambda x:\alpha. x) : ID >$. In $(\lambda \rightarrow)_\delta$, the following can be derived.

$$\Gamma_1 \vdash id : ID$$

- Let $\Gamma_2 \equiv < \gamma : *, \rightarrow = \lambda \alpha : *. \lambda \beta : *. (\Pi x:\alpha. \beta) : (\Pi \alpha : *. \Pi \beta : *. *) >$. In $\lambda \underline{w}_\delta$, the following can be derived.

$$\Gamma_2 \vdash \lambda x:\gamma. x : (\gamma \rightarrow \gamma)$$

- Let $\Gamma_3 \equiv < \Gamma_2, \perp = (\Pi \alpha : *. \alpha) : *, \neg = (\lambda \alpha : *. \alpha \rightarrow \perp) : (\Pi \alpha : *. *) >$. In λw_δ , the following can be derived.

$$\Gamma_3 \vdash (\lambda x:\gamma. \lambda y:\neg\gamma. (y \ x)) : (\gamma \rightarrow \neg\neg\gamma)$$

We can not abbreviate the expression $\lambda \alpha : \Box. \lambda \beta : \Box. (\Pi x:\alpha. \beta)$ because it is not typable in the λ -cube.

- Let $\Gamma_4 \equiv < \Gamma_2, \alpha_1 : *, \alpha_2 : *, \wedge = (\lambda \alpha : *. \lambda \beta : *. (\Pi \gamma : *. (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow \gamma)) : (\Pi \alpha : *. \Pi \beta : *. *) >, K = (\lambda \alpha : *. \lambda \beta : *. \lambda x:\alpha. \lambda y:\beta. x) : (\Pi \alpha : *. \Pi \beta : *. \alpha \rightarrow \beta \rightarrow \alpha) >$. In λw_δ , the following can be derived.

$$\Gamma_4 \vdash (\lambda x:(\wedge \alpha_1 \alpha_2). x \ \alpha_1 \ (K \alpha_1 \alpha_2)) : (\wedge \alpha_1 \alpha_2) \rightarrow \alpha_1$$

We will prove the following properties:

- Church Rosser Theorem for $\rightarrow_\delta$ and for $\rightarrow_{\beta\delta}$.
- Strong Normalization Theorem for $\rightarrow_\delta$.
- Subject Reduction Theorem for DPTS's, i.e. if $\Gamma \vdash d : D$ and $\Gamma \vdash d \rightarrow_{\beta\delta} d'$ then $\Gamma \vdash d' : D$
- Uniqueness of Types for singly sorted DPTS's, i.e. if $\Gamma \vdash d : D$ and $\Gamma \vdash d : D'$ then $D =_{\beta\delta} D'$.
- Conservativity, i.e. for $A \in \mathcal{T}$, $\Gamma \in \mathcal{C}$ $\exists a \Gamma \vdash_{\lambda S} a : A$ iff $\exists a \Gamma \vdash_{\lambda S_\delta} a : A$
- If a PTS is $\beta\delta$ -weakly normalizing then the corresponding DPTS is $\beta\delta$ -weakly normalizing too.
- An extension of a PTS is $\beta\delta$ -strongly normalizing if a "slightly" larger PTS is β -strongly normalizing. In particular, the Calculus of Constructions extended with definitions is strongly normalizing.

Chapter 4

Properties of General Pseudoterms

In this chapter, basic properties of β and δ -reductions for all pseudoterms are proved.

In Section 4.2, Church Rosser property is proved for $\rightarrow_\delta$ and for $\rightarrow_{\beta\delta}$.

The idea of the proof of Church Rosser for δ -reduction is as follows:

- A "projection" map, $|-|_\Gamma : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow \mathcal{T}$, is defined. The "projection" $|a|_\Gamma$ is a term that is obtained from a by unfolding all the definitions occurring in Γ and in a .
- A term a δ -reduces to its projection, i.e. $\Gamma \vdash a \rightarrow_\delta |a|_\Gamma$.
- The projections of two terms that are in $\rightarrow_\delta$ are equal, i.e. if $\Gamma \vdash c \rightarrow_\delta d$ then $|c|_\Gamma = |d|_\Gamma$.

Finally, the proof of Church Rosser for $\beta\delta$ -reduction follows from the previous considerations and the following one:

- The projection preserves β -reduction, i.e. if $\Gamma \vdash c \rightarrow_\beta d$ then $|c|_\Gamma \rightarrow_\beta |d|_\Gamma$.

In Section 4.3, Strong Normalization property is proved for $\rightarrow_\delta$.

Also, an illustrative and intuitive proof of Weak Normalization for δ -reduction is presented. The δ -normal form of a term a in a context Γ is its projection, i.e. the result from a by unfolding all the definitions occurring in Γ and in a .

In order to prove Strong Normalization for $\rightarrow_\delta$, the well-known method of defining a function $nat_{-}(-) : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow \mathbb{N}$ which decreases with δ -reduction is used.

4.1 Basic Properties

Lemma 4.1.1. *Let $\langle \Gamma_1, \Gamma_2, \Gamma_3 \rangle \in \mathcal{C}_\delta$ and $b, b' \in \mathcal{T}_\delta$ be such that $\Gamma_1, \Gamma_3 \vdash b \rightarrow_\delta b'$. Then*

$$\Gamma_1, \Gamma_2, \Gamma_3 \vdash b \rightarrow_\delta b'.$$

Proof: It is proved by induction on the definition of $\rightarrow_\delta$. □

Lemma 4.1.2. *Let be $\Gamma \in \mathcal{C}_\delta$, $x \in V$ and $a, A, b, b' \in \mathcal{T}_\delta$. The following rule is derived from the ones in the definition of $\rightarrow_\delta$.*

$$\frac{\Gamma \vdash b \rightarrow_\delta b'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A \text{ in } b')}$$

Proof: By lemma 4.1.1, it follows that $\Gamma, x=a:A \vdash b \rightarrow_\delta b'$. By definition of $\rightarrow_\delta$, it follows that $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A \text{ in } b')$. □

Lemma 4.1.3.

1. Let $\langle \Gamma, x:A, \Gamma' \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$.
 $\Gamma, x:A, \Gamma' \vdash b \rightarrow_{\beta\delta} b'$ if and only if $\Gamma, \Gamma' \vdash b \rightarrow_{\beta\delta} b'$.
2. Let $\langle \Gamma, x=a:A, \Gamma' \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$ be such that $x \notin FV(b)$.
 $\Gamma, x=a:A, \Gamma' \vdash b \rightarrow_{\beta\delta} b'$ if and only if $\Gamma, \Gamma' \vdash b \rightarrow_{\beta\delta} b'$.

Proof:

1. The implication from left to right is proved by induction on the definition of $\rightarrow_{\beta\delta}$. The converse implication follows by lemma 4.1.1.
2. The implication from left to right is proved by induction on the definition of $\rightarrow_{\beta\delta}$. The converse implication follows by lemma 4.1.1.

□

Example 4.1.4. In example 3.2.3 we show that $id = (\lambda y:A. y) \cdot (\Pi y:A. A)$ in $(\lambda x: (\Pi y:A. A). id) id$ δ reduces in several steps to the term

$$(\lambda x: (\Pi y:A. A). (\lambda y:A. y)) (\lambda y:A. y)$$

Note that $(\lambda x: (\Pi y:A. A). (\lambda y:A. y)) (\lambda y:A. y)$ is the substitution of all the occurrences of id by $(\lambda y:A. y)$ in the expression $(\lambda x: (\Pi y:A. A). id) id$, i.e.

$$(\lambda x: (\Pi y:A. A). id) id [id := (\lambda y:A. y)] \equiv (\lambda x: (\Pi y:A. A). (\lambda y:A. y)) (\lambda y:A. y)$$

This is a particular case of the following theorem:

Theorem 4.1.5. Let $\Gamma \equiv \langle \Gamma_1, x=a:A, \Gamma_2 \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$. Then

$$\Gamma \vdash b \rightarrow_\delta b[x := a].$$

Proof: It follows easily by induction on the structure of b .

□

Corollary 4.1.6. Let $\Gamma \in \mathcal{C}_\delta$ and $(x=a:A \text{ in } b) \in \mathcal{T}_\delta$. Then

$$\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta b[x := a]$$

Proof: By theorem 4.1.5, it follows that $\Gamma, x=a:A \vdash b \rightarrow_\delta b[x := a]$.

Then $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A \text{ in } b[x := a])$. Since $x \notin FV(b[x := a])$, it follows that $(x=a:A \text{ in } b[x := a]) \rightarrow_\delta b[x := a]$.

□

Lemma 4.1.7. [Substitution Lemma] Let $a, b, d \in \mathcal{T}_\delta$. Suppose $x \neq y$ and $x \notin FV(d)$. Then

$$a[x := b][y := d] \equiv a[y := d][x := b[y := d]]$$

Proof: It is proved by induction on the structure of a .

□

Lemma 4.1.8. Given $a, b \in \mathcal{T}_\delta$ such that $a \rightarrow_\beta a'$. Then

$$a[x := b] \rightarrow_\beta a'[x := b].$$

Proof: It is proved by induction on the generation of $\rightarrow_\beta$.

□

The previous lemma means that β is substitutive. The following example shows that $\rightarrow_\delta$ is not substitutive.

Example 4.1.9. Let Γ be the context $\langle A : *, id = (\lambda y:A. y) : (\Pi y:A. A) \rangle$.

$$\Gamma \vdash id \rightarrow_\delta (\lambda y:A. y)$$

But it is not true that $\Gamma \vdash id[id := b] \rightarrow_\delta (\lambda y:A. y)[id := b]$ for all $b \in \mathcal{T}_\delta$. This is because $id \in Def(\Gamma)$.

Lemma 4.1.10. Given $a, b \in \mathcal{T}_\delta$ such that $\Gamma \vdash b \rightarrow_{\beta\delta} b'$. Then

$$\Gamma \vdash a[x := b] \rightarrow_{\beta\delta} a[x := b'].$$

Proof: It is proved by induction on the structure of a .

- Assume a is a variable. There are two possible cases.
 - Assume $a \equiv x$. By hypothesis, $\Gamma \vdash x[x := b] \rightarrow_{\beta\delta} x[x := b']$.
 - Assume $a \equiv y$ and that y is different from x . Hence $\Gamma \vdash y[x := b] \rightarrow_{\beta\delta} y[x := b']$.
- Assume $a \equiv (y=c:D \text{ in } e)$. There are two possible cases.
 - Assume $y \equiv x$. Hence $\Gamma \vdash (y=c:D \text{ in } e)[x := b] \rightarrow_{\beta\delta} (y=c:D \text{ in } e)[x := b']$.
 - Assume y is different from x . It follows from IH that $\Gamma \vdash c[x := b] \rightarrow_{\beta\delta} c[x := b']$, $\Gamma \vdash D[x := b] \rightarrow_{\beta\delta} D[x := b']$ and $\Gamma \vdash e[x := b] \rightarrow_{\beta\delta} e[x := b']$. It follows from lemma 4.1.2, and the definition of $\rightarrow_{\beta\delta}$ that $\Gamma \vdash (y=c[x := b]:D[x := b] \text{ in } e[x := b]) \rightarrow_{\beta\delta} (y=c[x := b']:D[x := b'] \text{ in } e[x := b'])$. Then by definition of substitution $\Gamma \vdash (y=c:D \text{ in } e)[x := b] \rightarrow_{\beta\delta} (y=c:D \text{ in } e)[x := b']$.

□

Theorem 4.1.11.

1. If $\Gamma \vdash e \rightarrow_{\beta\delta} e'$ and $\Gamma, x=e:E, \Gamma' \vdash B \rightarrow_\delta B'$ then $\Gamma, x=e':E, \Gamma' \vdash B \rightarrow_{\beta\delta} B'$.
2. If $\Gamma, x=e:E, \Gamma' \vdash B \rightarrow_{\beta\delta} B'$ then $\Gamma, \Gamma'[x := e] \vdash B[x := e] \rightarrow_{\beta\delta} B'[x := e]$.

Proof:

1. It is proved by induction on the definition of $\Gamma, x=e:E, \Gamma' \vdash B \rightarrow_\delta B'$.
2. It is proved that if $\Gamma, x=e:E, \Gamma' \vdash b \rightarrow_\delta b'$ then $\Gamma, \Gamma'[x := e] \vdash b[x := e] \rightarrow_\delta b'[x := e]$ (by induction on the definition of $\rightarrow_\delta$).
By lemma 4.1.8, it follows that if $\Gamma, x=e:E, \Gamma' \vdash b \rightarrow_\beta b'$ then $\Gamma, \Gamma'[x := e] \vdash b[x := e] \rightarrow_\beta b'[x := e]$.

□

Corollary 4.1.12.

1. If $\Gamma \vdash e \rightarrow_{\beta\delta} e'$ and $\Gamma, x=e:E, \Gamma' \vdash B =_{\beta\delta} B'$ then $\Gamma, x=e':E, \Gamma' \vdash B =_{\beta\delta} B'$.
2. If $\Gamma, x=e:E, \Gamma' \vdash B =_{\beta\delta} B'$ then $\Gamma, \Gamma'[x := e] \vdash B[x := e] =_{\beta\delta} B'[x := e]$.

4.2 Church Rosser for β , δ and $\beta\delta$ -reductions

Theorem 4.2.1 (Church Rosser for β -reduction). *Let $\Gamma \in \mathcal{C}_\delta$ and $a, a_1, a_2 \in \mathcal{T}_\delta$ such that $\Gamma \vdash a \rightarrow_\beta a_1$ and $\Gamma \vdash a \rightarrow_\beta a_2$. Then there exists a term a_3 such that $\Gamma \vdash a_1 \rightarrow_\beta a_3$ and $\Gamma \vdash a_2 \rightarrow_\beta a_3$.*

Proof: It follows from the fact that the Combinatorial Reduction System $\langle \mathcal{T}_\delta, \rightarrow_\beta \rangle$ is orthogonal (See [Klo90]). $\square$

Definition 4.2.2. A mapping $|_ : \mathcal{T}_\delta \times \mathcal{C}_\delta \rightarrow \mathcal{T}$ is defined as follows.

$$\begin{aligned} |x|_\Gamma &= \begin{cases} |a|_{\Gamma_1} & \text{if } \Gamma \equiv \langle \Gamma_1, x = a : A, \Gamma_2 \rangle \\ x & \text{otherwise} \end{cases} \\ |c|_\Gamma &= c \quad \text{for } c \in \mathcal{C} \\ |a \ b|_\Gamma &= |a|_\Gamma |b|_\Gamma \\ |\lambda x:A. a|_\Gamma &= (\lambda x:|A|_\Gamma. |a|_{\Gamma, xA}) \\ |\Pi x:A. B|_\Gamma &= (\Pi x:|A|_\Gamma. |B|_{\Gamma, xA}) \\ |x=a:A \text{ in } b|_\Gamma &= |b|_\Gamma[x := |a|_\Gamma] \quad \text{where } x \text{ is } \Gamma\text{-fresh.} \end{aligned}$$

The value $|d|_\Gamma$ is obtained from d by unfolding all the definitions occurring in Γ and in d . Note that $|d|_\Gamma \in \mathcal{T}$ and $\text{Def}(\Gamma) \cap FV(|d|_\Gamma) = \emptyset$. Conversely, if $d \in \mathcal{T}$ is such that $\text{Def}(\Gamma) \cap FV(d) = \emptyset$ then $|d|_\Gamma = d$. Hence $|_$ is a "projection" from $\mathcal{T}_\delta$ and $\mathcal{C}_\delta$ to $\mathcal{T}$. Later, it will be proved that $|d|_\Gamma$ is the δ -normal form of the term d .

Example 4.2.3. The value $|d|_\Gamma$ is computed for one of the examples 3.2.3.

$$|(id = (\lambda y:A. y) : (\Pi y:A. A) \text{ in } (\lambda x:(\Pi y:A. A). id) \ id)|_\Gamma = (\lambda x:(\Pi y:A. A). \lambda y:A. y)(\lambda y:A. y)$$

The function $FV : \mathcal{T}_\delta \rightarrow P(V)$ is extended to $\mathcal{C}_\delta \times \mathcal{T}_\delta$.

Definition 4.2.4. The mapping $FV : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow P(V)$ is defined as follows:

$$\begin{aligned} FV_\epsilon(b) &= FV(b) \\ FV_{x:A, \Gamma}(b) &= FV_\Gamma(A) \cup (FV_\Gamma(b) - \{x\}) \\ FV_{x=a:A, \Gamma}(b) &= FV_\Gamma(A) \cup FV_\Gamma(a) \cup (FV_\Gamma(b) - \{x\}) \end{aligned}$$

Lemma 4.2.5.

1. If $x \notin FV(b)$ and x is Γ -fresh then $x \notin FV(|b|_\Gamma)$.
2. Let $\langle \Gamma_1, \Gamma_2, \Gamma_3 \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$ be such that $(FV_{\Gamma_3}(b)) \cap \text{Def}(\Gamma_2) = \emptyset$. Then

$$|b|_{\Gamma_1, \Gamma_2, \Gamma_3} = |b|_{\Gamma_1, \Gamma_3}.$$

3. Let $\langle \Gamma_1, x=a:A, \Gamma_2 \rangle \in \mathcal{C}_\delta$ and $a \in \mathcal{T}_\delta$. Then

$$|a|_{\Gamma_1, x=a:A, \Gamma_2} = |a|_{\Gamma_1}$$

Proof:

1. It is proved by induction on the number of symbols occurring in Γ and b .
2. It is proved by induction on the structure of b .
3. None of the variables in $\text{Def}(x=a:A, \Gamma_2)$ can occur in a . Hence the result follows immediately from the previous part.

□

Lemma 4.2.6. *Let $\langle \Gamma, x=a:A \rangle \in C_\delta$ and $b \in T_\delta$. Then*

$$|b|_\Gamma [x := |a|_\Gamma] = |b[x := a]|_\Gamma = |b|_{\Gamma, x=a:A}$$

Proof: It is proved by induction on the structure of b . Only some cases are considered.

- Assume that b is a variable.
 - Assume $b \equiv x$.
 - $|x|_\Gamma [x := |a|_\Gamma] = x[x := |a|_\Gamma] = |a|_\Gamma$ (because x is Γ -fresh)
 - $|x[x := a]|_\Gamma = |a|_\Gamma$ (By definition of substitution)
 - $|x|_{\Gamma, x=a:A} = |a|_\Gamma$ (By definition 4.2.2 part 1).
 - Assume $b \equiv y \neq x$.
 - $|y|_\Gamma [x := |a|_\Gamma] = |y|_\Gamma$ (By lemma 4.2.5)
 - $|y[x := a]|_\Gamma = |y|_\Gamma$ (By definition of substitution)
 - $|y|_{\Gamma, x=a:A} = |y|_\Gamma$ (By lemma 4.2.5 part 2).
- Assume $b \equiv (y=d:D \text{ in } e)$. Then y is $\langle \Gamma, x=a:A \rangle$ -fresh by convention 3.1.11. First the equality $|b|_\Gamma [x := |a|_\Gamma] = |b|_{\Gamma, x=a:A}$ will be proved.
 - $|b|_\Gamma [x := |a|_\Gamma] =$
 - $|y=d:D \text{ in } e|_\Gamma [x := |a|_\Gamma] =$ (By definition 4.2.2)
 - $|e|_\Gamma [y := |d|_\Gamma] [x := |a|_\Gamma] =$ (By Lemma 4.1.7)
 - $|e|_\Gamma [x := |a|_\Gamma] [y := |d|_\Gamma [x := |a|_\Gamma]] =$ (By IH)
 - $|e|_{\Gamma, x=a:A} [y := |d|_{\Gamma, x=a:A}] =$ (By definition 4.2.2)
 - $|y=d:D \text{ in } e|_{\Gamma, x=a:A}$

Next the equality $|b|_\Gamma [x := |a|_\Gamma] = |b[x := a]|_\Gamma$ will be proved.

$$\begin{aligned}
 &|b|_\Gamma [x := |a|_\Gamma] = \\
 &|y=d:D \text{ in } e|_\Gamma [x := |a|_\Gamma] = \text{(By definition 4.2.2)} \\
 &|e|_\Gamma [y := |d|_\Gamma] [x := |a|_\Gamma] = \text{(By Lemma 4.1.7)} \\
 &|e|_\Gamma [x := |a|_\Gamma] [y := |d|_\Gamma [x := |a|_\Gamma]] = \text{(By IH)} \\
 &|e[x := a]|_\Gamma [y := |d[x := a]|_\Gamma] = \text{(By definition 4.2.2)} \\
 &|y=d[x := a]:D[x := a] \text{ in } e[x := a]|_\Gamma = \text{(By definition of substitution)} \\
 &|y=d:D \text{ in } e[x := a]|_\Gamma
 \end{aligned}$$

□

The following theorem states that a term reduces to its projection.

Lemma 4.2.7. *Let $\Gamma \in C_\delta$ and $d \in T_\delta$. Then*

$$\Gamma \vdash d \rightarrow_\delta |d|_\Gamma.$$

Proof: It is proved by induction on the number of symbols occurring in Γ and in d . Only two cases are considered.

- Assume $d \equiv x$.
 - If $\Gamma \equiv \langle \Gamma_1, x = b : B, \Gamma_2 \rangle$ then $|x|_\Gamma = |b|_{\Gamma_1}$. By IH, it follows that $\Gamma_1 \vdash b \rightarrow_\delta |b|_{\Gamma_1}$. By lemma 4.1.1, it follows that $\Gamma \vdash b \rightarrow_\delta |b|_{\Gamma_1}$. Hence $\Gamma \vdash x \rightarrow_\delta b \rightarrow_\delta |x|_\Gamma = |b|_{\Gamma_1}$.
 - If $x \notin \text{Def}(\Gamma)$ then $\Gamma \vdash x \rightarrow_\delta x = |x|_\Gamma$.
- Assume $d \equiv (x=a:A \text{ in } b)$.
 - By IH, it follows that $\Gamma \vdash a \rightarrow_\delta |a|_\Gamma$, $\Gamma \vdash A \rightarrow_\delta |A|_\Gamma$ and that $\Gamma \vdash b \rightarrow_\delta |b|_\Gamma$. By definition of $\rightarrow_\delta$ and lemma 4.1.2, it follows that
 - $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=|a|_\Gamma:|A|_\Gamma \text{ in } |b|_\Gamma)$.
 - By corollary 4.1.6, it follows that
 - $\Gamma \vdash (x=|a|_\Gamma:|A|_\Gamma \text{ in } |b|_\Gamma) \rightarrow_\delta |b|_\Gamma [x := |a|_\Gamma]$.
 - As $|b|_\Gamma [x := |a|_\Gamma] = |x=a:A \text{ in } b|_\Gamma$, it follows that
 - $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta |x=a:A \text{ in } b|_\Gamma$.

□

The following theorem states that the projection of two terms that are in $\rightarrow_\delta$ are equal.

Lemma 4.2.8. *Let $\Gamma \in \mathcal{C}_\delta$ and $c, d \in \mathcal{T}_\delta$ be such that $\Gamma \vdash c \rightarrow_\delta d$. Then*

$$|c|_\Gamma = |d|_\Gamma.$$

Proof: It is proved by induction on the definition of $\rightarrow_\delta$. Only some cases are considered.

- Suppose that $\Gamma \vdash c \rightarrow_\delta d$ is $\Gamma_1, x=a:A, \Gamma_2 \vdash x \rightarrow_\delta a$
By definition 4.2.2, it follows that $|x|_{\Gamma_1, x=a:A, \Gamma_2} = |a|_{\Gamma_1}$. By lemma 4.2.5, it follows that $|a|_{\Gamma_1, x=a:A, \Gamma_2} = |a|_{\Gamma_1}$. Then $|x|_{\Gamma_1, x=a:A, \Gamma_2} = |a|_{\Gamma_1, x=a:A, \Gamma_2}$.
- Suppose that $\Gamma \vdash c \rightarrow_\delta d$ is $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta b$ if $x \notin FV(b)$.
By definition 4.2.2, it follows that $|x=a:A \text{ in } b|_\Gamma = |b|_\Gamma[x := |a|_\Gamma]$. By convention 3.1.11 the variable x is Γ -fresh. Then by lemma 4.2.5 part 1 $|b|_\Gamma[x := |a|_\Gamma] = |b|_\Gamma$. Hence $|x=a:A \text{ in } b|_\Gamma = |b|_\Gamma$.
- Suppose that $\Gamma \vdash c \rightarrow_\delta d$ is $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A \text{ in } b')$ under the hypothesis $\Gamma, x=a:A \vdash b \rightarrow_\delta b'$.
By definition 4.2.2, it follows that
 $|x=a:A \text{ in } b|_\Gamma = |b|_\Gamma[x := |a|_\Gamma]$
 $|x=a:A \text{ in } b'|_\Gamma = |b'|_\Gamma[x := |a|_\Gamma]$.
By lemma 4.2.6, the following equalities hold
 $|b|_{\Gamma, x=a:A} = |b|_\Gamma[x := |a|_\Gamma]$
 $|b'|_{\Gamma, x=a:A} = |b'|_\Gamma[x := |a|_\Gamma]$.
By IH, the equality $|b|_{\Gamma, x=a:A} = |b'|_{\Gamma, x=a:A}$ holds.
Hence $|x=a:A \text{ in } b|_\Gamma = |x=a:A \text{ in } b'|_\Gamma$.
- Suppose that $\Gamma \vdash c \rightarrow_\delta d$ is $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a':A \text{ in } b)$ under the hypothesis $\Gamma \vdash a \rightarrow_\delta a'$.
By definition 4.2.2, it follows that
 $|x=a:A \text{ in } b|_\Gamma = |b|_\Gamma[x := |a|_\Gamma]$
 $|x=a':A \text{ in } b|_\Gamma = |b|_\Gamma[x := |a'|_\Gamma]$.
By IH, it follows that $|a|_\Gamma = |a'|_\Gamma$. Hence $|b|_\Gamma[x := |a|_\Gamma] = |b|_\Gamma[x := |a'|_\Gamma]$.

□

Theorem 4.2.9 (Church Rosser for δ -reduction). *Let $\Gamma \in \mathcal{C}_\delta$ and $a, a_1, a_2 \in \mathcal{T}_\delta$ such that $\Gamma \vdash a \rightarrow_\delta a_1$ and $\Gamma \vdash a \rightarrow_\delta a_2$. Then there exists a term a_3 such that $\Gamma \vdash a_1 \rightarrow_\delta a_3$ and $\Gamma \vdash a_2 \rightarrow_\delta a_3$.*

Proof: By theorem 4.2.7, $\Gamma \vdash a \rightarrow_\delta |a|_\Gamma$, $\Gamma \vdash a_1 \rightarrow_\delta |a_1|_\Gamma$ and $\Gamma \vdash a_2 \rightarrow_\delta |a_2|_\Gamma$.

By Lemma 4.2.8, it follows that $|a|_\Gamma = |a_1|_\Gamma = |a_2|_\Gamma$.

The scheme of the proof is illustrated in the following diagram.

The term $|a|_\Gamma$ is the δ -nf in Γ of a , a_1 and a_2 . This will be proved in corollary 4.3.3

□

The following lemma states that the projection preserves β -reduction.

Lemma 4.2.10. *Given $a, a' \in \mathcal{T}_\delta$ and $\Gamma \in \mathcal{C}_\delta$ such that $a \rightarrow_\beta a'$. Then*

$$|a|_\Gamma \rightarrow_\beta |a'|_\Gamma.$$

Proof: It is proved by induction on the definition of $a \rightarrow_\beta a'$.

- Assume $a \equiv (c \ d)$.

There are 3 possibilities for the reduction $a \rightarrow_\beta a'$.

1. Assume $a \rightarrow_\beta a'$ is $(\lambda x:B. b)d \rightarrow_\beta b[x := d]$.
By definition 4.2.2, it follows that
 $|a|_\Gamma = |(\lambda x:B. b)d|_\Gamma = (\lambda x:|B|_\Gamma. |b|_\Gamma)|d|_\Gamma$.
By Lemma 4.2.6, it follows that $|b[x := d]|_\Gamma = |b|_\Gamma[x := |d|_\Gamma]$.
Hence $|(\lambda x:B. b)d|_\Gamma \rightarrow_\beta |b[x := d]|_\Gamma$.
2. Assume $a \rightarrow_\beta a'$ is $(c \ d) \rightarrow_\beta (c' \ d)$ under the hypothesis $c \rightarrow_\beta c'$.
By IH, it follows that $|c|_\Gamma \rightarrow_\beta |c'|_\Gamma$. Hence $|c \ d|_\Gamma \rightarrow_\beta |c' \ d|_\Gamma$.
3. Assume $a \rightarrow_\beta a'$ is $(c \ d) \rightarrow_\beta (c \ d')$.
By IH, it follows that $|d|_\Gamma \rightarrow_\beta |d'|_\Gamma$. Hence $|c \ d|_\Gamma \rightarrow_\beta |c \ d'|_\Gamma$.

- Assume $a \equiv (x=b:B \text{ in } c)$.

There are 3 possibilities for the reduction $a \rightarrow_\beta a'$.

1. Assume $a \rightarrow_\beta a'$ is $(x=b:B \text{ in } c) \rightarrow_\beta (x=b:B \text{ in } c')$ under the hypothesis $c \rightarrow_\beta c'$.
By the definition 4.2.2, it follows that
 $|a|_\Gamma = |x=b:B \text{ in } c|_\Gamma = |c|_\Gamma[x := |b|_\Gamma]$
 $|a'|_\Gamma = |x=b:B \text{ in } c'|_\Gamma = |c'|_\Gamma[x := |b|_\Gamma]$.
By IH, it follows that $|c|_\Gamma \rightarrow_\beta |c'|_\Gamma$.
By lemma 4.1.8, it follows that $|c|_\Gamma[x := |b|_\Gamma] \rightarrow_\beta |c'|_\Gamma[x := |b|_\Gamma]$.
2. Assume $a \rightarrow_\beta a'$ is $(x=b:B \text{ in } c) \rightarrow_\beta (x=b:B' \text{ in } c)$ with $B \rightarrow_\beta B'$.
It follows that $|a|_\Gamma = |c|_\Gamma[x := |b|_\Gamma] = |a'|_\Gamma$.
3. Assume $a \rightarrow_\beta a'$ is $(x=b:B \text{ in } c) \rightarrow_\beta (x=b':B \text{ in } c)$ under the hypothesis $b \rightarrow_\beta b'$.
By the definition 4.2.2, the values of $|a|_\Gamma$ and $|a'|_\Gamma$ are
 $|a|_\Gamma = |x=b:B \text{ in } c|_\Gamma = |c|_\Gamma[x := |b|_\Gamma]$
 $|a'|_\Gamma = |x=b':B \text{ in } c|_\Gamma = |c|_\Gamma[x := |b'|_\Gamma]$.
By IH, it follows that $|b|_\Gamma \rightarrow_\beta |b'|_\Gamma$.
By lemma 4.1.10, it follows that $|c|_\Gamma[x := |b|_\Gamma] \rightarrow_\beta |c|_\Gamma[x := |b'|_\Gamma]$.

□

Theorem 4.2.11 (Church Rosser for $\beta\delta$ -reduction). *Let $\Gamma \in \mathcal{C}_\delta$ and $a \in \mathcal{T}_\delta$ be such that $\Gamma \vdash a \rightarrow_{\beta\delta} b$ and $\Gamma \vdash a \rightarrow_{\beta\delta} c$. Then there exists a term $d \in \mathcal{T}_\delta$ such that $\Gamma \vdash b \rightarrow_{\beta\delta} d$ and $\Gamma \vdash c \rightarrow_{\beta\delta} d$.*

Proof: By Lemma 4.2.7, it is deduced that $\Gamma \vdash a \rightarrow_\delta |a|_\Gamma$, $\Gamma \vdash b \rightarrow_\delta |b|_\Gamma$ and that $\Gamma \vdash c \rightarrow_\delta |c|_\Gamma$.
By lemma 4.2.10 and lemma 4.2.8, it follows that $\Gamma \vdash |a|_\Gamma \rightarrow_\beta |b|_\Gamma$ and that $\Gamma \vdash |a|_\Gamma \rightarrow_\beta |c|_\Gamma$.
Hence the solid lines of the diagram are justified.

By Church Rosser theorem for β -reduction between pseudoterms in $\mathcal{T}$, the dotted arrows of the diagram are justified.

By composition along the borders, it follows that $\Gamma \vdash b \twoheadrightarrow_{\beta\delta} d$ and $\Gamma \vdash c \twoheadrightarrow_{\beta\delta} d$.

□

4.3 Weak and Strong Normalization for $\rightarrow_\delta$

Definition 4.3.1. Let $a \in \mathcal{T}_\delta$ and $\Gamma \in \mathcal{C}_\delta$. The term a is in δ -normal form (or δ -nf) in the context Γ if there is no term b such that $\Gamma \vdash a \rightarrow_\delta b$.

A term can be in δ -normal form but not in β -normal form, for example $(\lambda y:*. y)(\lambda y:A. y)$.

According to the following theorem, a term a is in δ -nf in a context Γ if and only if a does not contain definitions and its free variables are not definitions in the context Γ . As a corollary $|a|_\Gamma$ is the δ -nf of a in Γ .

Theorem 4.3.2. Let $a \in \mathcal{T}_\delta$. Then

a is in δ -nf in Γ if and only if $a \in \mathcal{T}$ and $FV(a) \cap Def(\Gamma) = \emptyset$.

Proof: It follows easily by induction on the structure of a .

□

Corollary 4.3.3. (Weak Normalization for δ -reduction) Let $\Gamma \in \mathcal{C}_\delta$ and $a \in \mathcal{T}_\delta$, the term $|a|_\Gamma$ is the δ -nf of a in Γ .

Proof: By the previous theorem, it follows from $|a|_\Gamma \in \mathcal{T}$ and $FV(|a|_\Gamma) \cap Def(\Gamma) = \emptyset$ that $|a|_\Gamma$ is in δ -nf in Γ . By Church Rosser for δ -reduction, the δ -normal form is unique.

□

Definition 4.3.4. Let $\Gamma \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$.

- The term b δ -strongly normalizes in Γ if there is no infinite δ -reduction starting with b in the context Γ .
- The reduction δ is strongly normalizing if for all pairs $(\Gamma, b) \in \mathcal{C}_\delta \times \mathcal{T}_\delta$, the term b δ -strongly normalizes in Γ .

Note that by theorem 4.3.3 the normal form for an arbitrary δ -term exists, but it is not guaranteed that all δ -paths starting at the term are finite.

Next a function $nat_{-}(-) : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow \mathbb{N}$ that decreases with δ -reduction is defined. It will be used to prove Strong Normalization for $\rightarrow_\delta$.

Definition 4.3.5. The mapping $\text{nat}_-(-) : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow \mathbb{N}$ is defined as follows.

$$\begin{aligned}
\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(x) &= \text{nat}_{\Gamma_1}(a) + 1 \\
\text{nat}_\Gamma(x) &= 0 \text{ if } x \notin \text{Def}(\Gamma) \\
\text{nat}_\Gamma(c) &= 0 \text{ if } c \in \mathcal{C} \\
\text{nat}_\Gamma(x=a:A \text{ in } b) &= \text{nat}_\Gamma(a) + \text{nat}_\Gamma(A) + \text{nat}_{\Gamma, x=a:A}(b) + 1 \\
\text{nat}_\Gamma(a \ b) &= \text{nat}_\Gamma(a) + \text{nat}_\Gamma(b) \\
\text{nat}_\Gamma(\Pi x:A. a) &= \text{nat}_{\Gamma, x:A}(a) + \text{nat}_\Gamma(A) \\
\text{nat}_\Gamma(\lambda x:A. a) &= \text{nat}_{\Gamma, x:A}(a) + \text{nat}_\Gamma(A)
\end{aligned}$$

Lemma 4.3.6. Let $\langle \Gamma_1, \Gamma_2, \Gamma_3 \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$ be such that $FV_{\Gamma_3}(b) \cap \text{Def}(\Gamma_2) = \emptyset$. Then

$$\text{nat}_{\Gamma_1, \Gamma_2, \Gamma_3}(b) = \text{nat}_{\Gamma_1, \Gamma_3}(b).$$

Proof: It is proved by induction on the number of symbols in $\langle \Gamma_1, \Gamma_2, \Gamma_3 \rangle$ and in b . □

Lemma 4.3.7. Let $\langle \Gamma_1, x=a:A, \Gamma_2 \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$

$$\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(b) \geq \text{nat}_{\Gamma_1, \Gamma_2}(b).$$

Proof: It is proved by induction on the number of symbols in $\Gamma_1, x=a:A, \Gamma_2$ and in b . Only the case that b is a variable is considered.

- Assume $b \equiv x$. Since $\text{nat}_{\Gamma_1, \Gamma_2}(x) = 0$ and $\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(x) = \text{nat}_{\Gamma_1}(a) + 1$, it follows that $\text{nat}_{\Gamma_1, \Gamma_2}(x) < \text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(x)$.
- Assume $b \equiv y$ and $y \in \text{Def}(\Gamma_1)$. Let $\Gamma_1 \equiv \Gamma_{11}, y=c:C, \Gamma_{12}$. Since $\text{nat}_{\Gamma_1, \Gamma_2}(y) = \text{nat}_{\Gamma_{11}}(c) + 1$ and $\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(y) = \text{nat}_{\Gamma_{11}}(c) + 1$, it follows that $\text{nat}_{\Gamma_1, \Gamma_2}(y) = \text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(y)$.
- Assume $b \equiv y$ and $y \in \text{Def}(\Gamma_2)$. Let $\Gamma_2 \equiv \Gamma_{21}, y=c:C, \Gamma_{22}$. Then $\text{nat}_{\Gamma_1, \Gamma_2}(y) = \text{nat}_{\Gamma_1, \Gamma_{21}}(c) + 1$ and $\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(y) = \text{nat}_{\Gamma_1, x=a:A, \Gamma_{21}}(c) + 1$. By IH, we have that $\text{nat}_{\Gamma_1, \Gamma_{21}}(c) \leq \text{nat}_{\Gamma_1, x=a:A, \Gamma_{21}}(c)$. Hence, it follows that $\text{nat}_{\Gamma_1, \Gamma_2}(y) \leq \text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(y)$.
- Assume $b \equiv y$ and $y \notin \text{Def}(\Gamma_1, x=a:A, \Gamma_2)$. Since $\text{nat}_{\Gamma_1, \Gamma_2}(y) = 0$ and $\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(y) = 0$, it follows that $\text{nat}_{\Gamma_1, \Gamma_2}(y) = \text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(y)$.

□

Definition 4.3.8. The concept "the context Γ δ -reduces to the context Γ' " for $\Gamma, \Gamma' \in \mathcal{C}_\delta$ is written as $\Gamma \rightarrow_\delta \Gamma'$ and is defined as follows.

$$\begin{array}{c}
\frac{\Gamma \vdash E \rightarrow_\delta E'}{\Gamma, y:E, \Gamma' \rightarrow_\delta \Gamma, y:E', \Gamma'} \\
\frac{\Gamma \vdash E \rightarrow_\delta E'}{\Gamma, y=e:E, \Gamma' \rightarrow_\delta \Gamma, y=e:E', \Gamma'} \\
\frac{\Gamma \vdash e \rightarrow_\delta e'}{\Gamma, y=e:E, \Gamma' \rightarrow_\delta \Gamma, y=e':E, \Gamma'}
\end{array}$$

Lemma 4.3.9. Let $\Gamma \in \mathcal{C}_\delta$ and terms $d, d' \in \mathcal{T}_\delta$ be such that $\Gamma \vdash d \rightarrow_\delta d'$. Then

$$\text{nat}_\Gamma(d) > \text{nat}_\Gamma(d').$$

Proof: The following two properties are proved simultaneously by induction on number of symbols in Γ and in d .

1. If $\Gamma \vdash d \rightarrow_\delta d'$ then $\text{nat}_\Gamma(d) > \text{nat}_\Gamma(d')$.
2. If $\Gamma \rightarrow_\delta \Gamma'$ then $\text{nat}_\Gamma(d) \geq \text{nat}_{\Gamma'}(d)$.

We only give the case that $d \equiv x$.

1. Then $\Gamma \vdash d \rightarrow_\delta d'$ is $\Gamma_1, x = a : A, \Gamma_2 \vdash x \rightarrow_\delta a$
 Since $\text{nat}_{\Gamma_1, x=a:A, \Gamma_2}(x) = \text{nat}_{\Gamma_1}(a) + 1$ and by lemma 4.3.6, $\text{nat}_{\Gamma_1}(a) = \text{nat}_\Gamma(a)$, it follows that $\text{nat}_\Gamma(x) > \text{nat}_\Gamma(a)$.
2. There are three possible cases. Suppose $\Gamma \equiv \Gamma_1, y = e : E, \Gamma_2$ with $\Gamma_1 \vdash e \rightarrow_\delta e'$ and $\Gamma_2 \equiv \Gamma_1, y = e' : E, \Gamma_2$.
 If $x \in \Gamma_1$ then $\text{nat}_\Gamma(x) = \text{nat}_{\Gamma'}(x)$. If $x \equiv y$ then it follows from IH(1) that $\text{nat}_\Gamma(x) = \text{nat}_{\Gamma_1}(e) > \text{nat}_{\Gamma_1}(e') = \text{nat}_{\Gamma_1, y=e':E, \Gamma_2}(x)$. If $x \in \Gamma_2$ then it follows from IH(2) that $\text{nat}_\Gamma(x) \geq \text{nat}_{\Gamma'}(x)$.
 If $x \notin \Gamma$ then $\text{nat}_\Gamma(x) = x = \text{nat}_{\Gamma'}(x)$.
 The rest of the cases are easy to prove.

□

Theorem 4.3.10. (Strong Normalization for δ) *The reduction δ is strongly normalizing.*

Proof: This follows directly from lemma 4.3.9.

□

Chapter 5

Properties of Well-Typed Terms

The properties in this chapter are proved for all well-typed terms, i.e. for terms a such that $\exists A, \Gamma[\Gamma \vdash a : A]$.

In Section 5.1, basic properties are proved for DPTS's. In Section 5.2 we prove that λS is Weakly Normalizing if and only if λS_δ is Weakly Normalizing.

Finally Strong Normalization for $\beta\delta$ reduction is proved for certain DPTS's.

5.1 Basic Properties

Lemma 5.1.1. (Generation Lemma)

1. $\Gamma \vdash c : D \Rightarrow \exists s \in \mathbf{S}[\Gamma \vdash D =_{\beta\delta} s \ \& \ c : s \in A]$
2. $\Gamma \vdash x : D \Rightarrow \exists s \in \mathbf{S} \exists b \exists B =_{\beta\delta} D[\Gamma \vdash B : s \ \& \ x : B \in \Gamma \vee \Gamma \vdash b : B \ \& \ x = b : B \in \Gamma]$
3. $\Gamma \vdash (\Pi x:A. B) : D \Rightarrow \exists (s_1, s_2, s_3) \in \mathbf{R}[\Gamma \vdash A : s_1 \ \& \ \Gamma, x:A \vdash B : s_2 \ \& \ D =_{\beta\delta} s_3]$
4. $\Gamma \vdash (\lambda x:A. b) : D \Rightarrow \exists s \in \mathbf{S} \exists B[\Gamma \vdash (\Pi x:A. B) : s \ \& \ \Gamma, x:A \vdash b : B \ \& \ D =_{\beta\delta} (\Pi x:A. B)]$
5. $\Gamma \vdash (b \ a) : D \Rightarrow \exists A, B[\Gamma \vdash b : (\Pi x:A. B) \ \& \ \Gamma \vdash a : A \ \& \ D =_{\beta\delta} B[x := a]]$
6. $\Gamma \vdash (x=a:A \text{ in } b) : D \Rightarrow \exists B[\Gamma, x=a:A \vdash b : B \ \& \ \Gamma \vdash (x=a:A \text{ in } B) : s \ \& \ D =_{\beta\delta} (x=a:A \text{ in } B) \vee \Gamma, x=a:A \vdash b : s \ \& \ D =_{\beta\delta} s]$

Proof: The different cases are all proved by induction on the derivation. □

Observe that in 6, the type of a term $(x=a:A \text{ in } b)$ can be a sort s or an expression of the form $(x=a:A \text{ in } B)$.

Lemma 5.1.2. (Correctness of types)

1. $\Gamma \vdash A : B \Rightarrow \exists s \in \mathbf{S}[B \equiv s \vee \Gamma \vdash B : s]$
2. $\Gamma \vdash b : (\Pi x:A. B) \Rightarrow \exists (s_1, s_2, s_3) \in \mathbf{R}[\Gamma \vdash A : s_1 \ \& \ \Gamma, x:A \vdash B : s_2]$

Proof:

1. It is proved by induction on the derivation of $\Gamma \vdash A : B$.
2. It follows from the previous part that $\Gamma \vdash (\Pi x:A. B) : s$. By the Generation Lemma part 3, it follows that $\exists (s_1, s_2, s_3) \in \mathbf{R}[\Gamma \vdash A : s_1 \ \& \ \Gamma, x:A \vdash B : s_2,]$

□

Lemma 5.1.3. *If there exists $b, B \in \mathcal{T}_\delta$ such that $\Gamma, x=a:A, \Gamma' \vdash b : B$ then $\Gamma \vdash a : A$.*

Proof: It is proved by induction on the derivation of $\Gamma, x=a:A, \Gamma' \vdash b : B$. $\square$

Lemma 5.1.4. (Thinning Lemma) Assume $\Gamma' \vdash b : B$. If $\Gamma \vdash a : A$ and $\Gamma \subseteq \Gamma'$ then $\Gamma' \vdash a : A$

Proof: It is proved by induction on the derivation $\Gamma \vdash a : A$. $\square$

Lemma 5.1.5. If $\Gamma \vdash a : A$ and $\Gamma, x:A, \Gamma' \vdash b : B$ then $\Gamma, x=a:A, \Gamma' \vdash b : B$

Proof: It is proved by induction on the derivation of $\Gamma, x:A, \Gamma' \vdash b : B$. $\square$

Example 5.1.6. It is not true that $\Gamma, x=a:A, \Gamma' \vdash b : B \Rightarrow \Gamma, x:A, \Gamma' \vdash b : B$. For instance, we can derive $\alpha:*, x=\alpha:*, y:x, f:(\Pi y:\alpha. \alpha) \vdash (f \ y) : \alpha$.

But there is no term B such that $\alpha:*, x:*, y:x, f:(\Pi y:\alpha. \alpha) \vdash (f \ y) : B$.

Lemma 5.1.7. (Substitution Lemma)

1. If $\Gamma, x=a:A, \Gamma' \vdash b : B$ then $\Gamma, \Gamma'[x := a] \vdash b[x := a] : B[x := a]$
2. If $\Gamma \vdash a : A$ and $\Gamma, x:A, \Gamma' \vdash b : B$ then $\Gamma, \Gamma'[x := a] \vdash b[x := a] : B[x := a]$.

Proof:

1. It is proved by induction on the derivation of $\Gamma, x=a:A, \Gamma' \vdash b : B$.
2. It follows by lemma 5.1.5 and the previous part.

$\square$

Definition 5.1.8. The concept "the context Γ $\beta\delta$ -reduces to the context Γ' " is written as $\Gamma \rightarrow_{\beta\delta} \Gamma'$ and is defined as follows.

$$\frac{\Gamma \vdash E \rightarrow_{\beta\delta} E'}{\Gamma, y:E, \Gamma' \rightarrow_{\beta\delta} \Gamma, y:E', \Gamma'} \quad \frac{\Gamma \vdash E \rightarrow_{\beta\delta} E'}{\Gamma, y=e:E, \Gamma' \rightarrow_{\beta\delta} \Gamma, y=e:E', \Gamma'} \quad \frac{\Gamma \vdash e \rightarrow_{\beta\delta} e'}{\Gamma, y=e:E, \Gamma' \rightarrow_{\beta\delta} \Gamma, y=e':E, \Gamma'}$$

Theorem 5.1.9. (Subject Reduction Theorem) If $\Gamma \vdash d : D$ and $\Gamma \rightarrow_{\beta\delta} \Gamma'$ then $\Gamma' \vdash d : D$

Proof: The following properties are proved simultaneously by induction on the derivation of $\Gamma \vdash d : D$.

1. If $\Gamma \vdash d \rightarrow_{\beta\delta} d'$ and $\Gamma \vdash d : D$ then $\Gamma \vdash d' : D$
2. If $\Gamma \rightarrow_{\beta\delta} \Gamma'$ and $\Gamma \vdash d : D$ then $\Gamma' \vdash d : D$

We only give the proof for some cases of the first property. Suppose that the last rule in the derivation of $\Gamma \vdash d : D$ is:

$$\bullet \text{ (application) } \frac{\Gamma \vdash b : (\Pi x:A. B) \quad \Gamma \vdash a : A}{\Gamma \vdash (b \ a) : B[x := a]}$$

There are 3 possibilities:

- Assume $\Gamma \vdash (b \ a) \rightarrow_{\beta\delta} (b' \ a)$ under the hypothesis $\Gamma \vdash b \rightarrow_{\beta\delta} b'$. By IH(1), it follows that $\Gamma \vdash b' : (\Pi x:A. B)$. By Application Rule, it follows that $\Gamma \vdash (b' \ a) : B[x := a]$.

- Assume $\Gamma \vdash (b \ a) \rightarrow_{\beta\delta} (b \ a')$ under the hypothesis $\Gamma \vdash a \rightarrow_{\beta\delta} a'$.
By IH(1), it follows that $\Gamma \vdash a' : A$. By Application Rule, it follows that $\Gamma \vdash (b \ a') : B[x := a']$. By lemma 4.1.10, it follows that $\Gamma \vdash B[x := a] \rightarrow_{\beta\delta} B[x := a']$. By Correctness of Types, lemma 5.1.2 part 2, it follows that $\Gamma, x:A \vdash B : s_2$. By Substitution Lemma 5.1.7 part 2, it follows that $\Gamma \vdash B[x := a] : s_2$. By Conversion Rule, it follows that $\Gamma \vdash (b \ a') : B[x := a]$.
- Assume $b \equiv (\lambda x:A_1. b_1)$ and $\Gamma \vdash (\lambda x:A_1. b_1)a \rightarrow_{\beta} b_1[x := a]$. By Generation Lemma part 4, it follows that $A =_{\beta\delta} A_1$ and that $\Gamma, x : A_1 \vdash b_1 : B$. By Generation Lemma part 3, it follows that $\Gamma \vdash A_1 : s$. By Conversion Rule, it follows that $\Gamma \vdash a : A_1$. By Substitution Lemma 5.1.7 part 2, it follows that $\Gamma \vdash b_1[x := a] : B[x := a]$.

$$\bullet (\delta - \text{introduction}) \frac{\Gamma, x=a:A \vdash b : B \quad \Gamma \vdash (x=a:A \text{ in } B) : s}{\Gamma \vdash (x=a:A \text{ in } b) : (x=a:A \text{ in } B)}$$

The following 5 cases are considered:

- Assume $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_{\beta\delta} (x=a:A' \text{ in } b)$ under the hypothesis $\Gamma \vdash A \rightarrow_{\beta\delta} A'$.
By IH(2), it follows that $\Gamma, x=a:A' \vdash b : B$. By IH(1), it follows that $\Gamma \vdash (x=a:A' \text{ in } B) : s$. By the δ -introduction Rule, it follows that $\Gamma \vdash (x=a:A' \text{ in } b) : (x=a:A' \text{ in } B)$. By the Conversion Rule, it follows that $\Gamma \vdash (x=a:A' \text{ in } b) : (x=a:A \text{ in } B)$.
- Assume $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_{\beta\delta} (x=a':A \text{ in } b)$ under the hypothesis $\Gamma \vdash a \rightarrow_{\beta\delta} a'$.
By IH(1), it follows that $\Gamma \vdash (x=a':A \text{ in } B) : s$. By IH(2), it follows that $\Gamma, x=a':A \vdash b : B$. By δ -introduction Rule, it follows that $\Gamma \vdash (x=a':A \text{ in } b) : (x=a':A \text{ in } B)$. By Conversion Rule, it follows that $\Gamma \vdash (x=a':A \text{ in } b) : (x=a:A \text{ in } B)$.
- Assume $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_{\beta} (x=a:A \text{ in } b')$ under the hypothesis $\Gamma \vdash b \rightarrow_{\beta} b'$.
By IH(1), it follows that $\Gamma, x=a:A \vdash b' : B$. By δ -introduction Rule, it follows that $\Gamma \vdash (x=a:A \text{ in } b') : (x=a:A \text{ in } B)$.
- Assume $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_{\delta} b$ under the hypothesis $x \notin FV(b)$. By lemma 5.1.3, it follows that $\Gamma \vdash a : A$. By the Substitution Lemma 5.1.7 part 1, it follows that $\Gamma \vdash b[x := a] : B[x := a]$. Since $x \notin FV(b)$, the equality $b[x := a] \equiv b$ holds. By corollary 4.1.6, it follows that $\Gamma \vdash (x=a:A \text{ in } b) =_{\delta} B[x := a]$. By Conversion Rule, $\Gamma \vdash b : (x=a:A \text{ in } B)$.
- Assume $\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_{\delta} (x=a:A \text{ in } b')$ under the hypothesis $\Gamma, x=a:A \vdash b \rightarrow_{\delta} b'$.
By IH(1), it follows that $\Gamma, x=a:A \vdash b' : B$. By δ -introduction Rule, it follows that $\Gamma \vdash (x=a:A \text{ in } b') : (x=a:A \text{ in } B)$.

□

Definition 5.1.10. The mapping $|-| : \mathcal{C}_{\delta} \rightarrow \mathcal{C}$ is defined as follows:

$$\begin{aligned} |\epsilon| &= \epsilon \\ |\Gamma, x:A| &= |\Gamma|, x:|A|_{\Gamma} \\ |\Gamma, x=a:A| &= |\Gamma| \end{aligned}$$

Note that if $\Gamma \in \mathcal{C}$ then $|\Gamma| = \Gamma$. This mapping $|-|$ is the projection from $\mathcal{C}_{\delta}$ to $\mathcal{C}$.

Theorem 5.1.11. Let $\Gamma \in \mathcal{C}_{\delta}$ and $a, A \in \mathcal{T}_{\delta}$.

$$\text{If } \Gamma \vdash_{\lambda S_{\delta}} a : A \text{ then } |\Gamma| \vdash_{\lambda S} |a|_{\Gamma} : |A|_{\Gamma}.$$

Proof: Suppose that the last rule in the derivation of $\Gamma \vdash_{\lambda S_{\delta}} a : A$ is:

$$\bullet (\delta - \text{start}) \frac{\Gamma \vdash_{\lambda S_{\delta}} a : A}{\Gamma, x=a:A \vdash_{\lambda S_{\delta}} x : A} \text{ where } x \text{ is } \Gamma\text{-fresh.}$$

By IH $|\Gamma| \vdash_{\lambda S} |a|_{\Gamma} : |A|_{\Gamma}$. By definition 4.2.2 we have that $|x|_{\Gamma, x=a:A} = |a|_{\Gamma}$. By lemma 4.2.5 part 2, it follows that $|A|_{\Gamma, x=a:A} = |A|_{\Gamma}$. Hence $|\Gamma, x=a:A| \vdash_{\lambda S} |a|_{\Gamma, x=a:A} : |A|_{\Gamma, x=a:A}$.

- (δ -introduction) $\frac{\Gamma, x = a : A \vdash_{\lambda S \delta} b : B \quad \Gamma \vdash_{\lambda S \delta} (x = a : A \text{ in } B) : s}{\Gamma \vdash_{\lambda S \delta} (x = a : A \text{ in } b) : (x = a : A \text{ in } B)}$.

By IH $|\Gamma, x = a : A| \vdash_{\lambda S} |b|_{\Gamma, x=a:A} : |B|_{\Gamma, x=a:A}$. By lemma 4.2.6 and definition 4.2.2 we have that $|b|_{\Gamma, x=a:A} = |b|_{\Gamma}[x := |a|_{\Gamma}] = |x = a : A \text{ in } b|_{\Gamma}$ and that $|B|_{\Gamma, x=a:A} = |B|_{\Gamma}[x := |a|_{\Gamma}] = |x = a : A \text{ in } B|_{\Gamma}$.
Hence $|\Gamma| \vdash_{\lambda S \delta} |x = a : A \text{ in } b|_{\Gamma} : |x = a : A \text{ in } B|_{\Gamma}$.

□

Example 5.1.12. In $(\lambda_{\rightarrow})_{\delta}$ the following can be derived.

$$< \alpha : *, ID = (\Pi x : \alpha. \alpha) : *, id = (\lambda x : \alpha. x) : ID > \vdash_{(\lambda_{\rightarrow})_{\delta}} id : ID$$

It follows from the previous theorem that

$$\alpha : * \vdash_{\lambda_{\rightarrow}} (\lambda x : \alpha. x) : (\Pi x : \alpha. \alpha)$$

All the definitions are unfolded and a derivation in $\lambda_{\rightarrow}$ is obtained.

In the sense that there are not more typable terms with abbreviations, the inclusion of definitions to a PTS does not increase the strength of the system.

Corollary 5.1.13. (Conservativity) *Let $a \in T$ and $\Gamma \in \mathcal{C}$. Then*

1. $\exists A \quad \Gamma \vdash_{\lambda S} a : A \text{ iff } \exists A \quad \Gamma \vdash_{\lambda S \delta} a : A$
2. $\exists A \quad \Gamma \vdash_{\lambda S} A : a \text{ iff } \exists A \quad \Gamma \vdash_{\lambda S \delta} A : a$
3. *Let $A \in T$. Then $\Gamma \vdash_{\lambda S} a : A \text{ iff } \Gamma \vdash_{\lambda S \delta} a : A$*

The first part of the previous corollary states that a term is typable in λS iff it is typable in λS_{δ} . The second part states that a type is inhabited in λS iff it is inhabited in λS_{δ} . If we interpret types as propositions and terms as proofs, this means that a proposition is provable in λS iff it is provable in λS_{δ} .

Theorem 5.1.14 (Uniqueness of Types). *Let S be a singly sorted specification, $\Gamma \in \mathcal{C}_{\delta}$ and $a, A, B \in \mathcal{T}_{\delta}$ such that $\Gamma \vdash_{\lambda S \delta} a : A$ and $\Gamma \vdash_{\lambda S \delta} a : B$. Then $\Gamma \vdash A =_{\beta \delta} B$.*

Proof: It follows from lemma 5.1.11 that $|\Gamma| \vdash_{\lambda S} |a|_{\Gamma} : |A|_{\Gamma}$ and $|\Gamma| \vdash_{\lambda S} |a|_{\Gamma} : |B|_{\Gamma}$. As λS is singly sorted then Uniqueness of Types holds. Hence $|A|_{\Gamma} =_{\beta} |B|_{\Gamma}$. It follows from lemma 4.2.7 that $\Gamma \vdash A \rightarrow_{\delta} |A|_{\Gamma}$ and that $\Gamma \vdash B \rightarrow_{\delta} |B|_{\Gamma}$. Hence $A =_{\beta \delta} B$. □

5.2 Weak and Strong Normalization for $\beta\delta$ -reduction

In this section we will prove that if a PTS is weakly normalizing then the corresponding DPTS is weakly normalizing too.

Also we will prove that a DPTS is strongly normalizing if a slightly larger PTS is strongly normalizing. The idea of the proof of Strong Normalization is as follows:

- We define a mapping $\{ _ \}_- : \mathcal{T}_{\delta} \times \mathcal{C}_{\delta} \rightarrow T$ similar to the projection $| _ |_-$. The value $\{a\}_{\Gamma}$ is a term that is obtained from a by unfolding all the definitions occurring in Γ and in a . However $\{ _ \}_-$ differs from $| _ |_-$ in the value given to $\{x=a:A \text{ in } b\}_{\Gamma}$. Instead of removing the local definition it is translated to a β -redex, i.e. an application and an abstraction.
- This function $\{ _ \}_-$ maps an infinite $\beta\delta$ reduction sequence to an infinite β reduction sequence.
- The function $\{ _ \}_-$ maps terms that are typable in a DPTS λS_{δ} to terms that are typable in a PTS slightly larger than λS .

Definition 5.2.1. Let $\Gamma \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$. The term b $\beta\delta$ -weakly normalizes in Γ if there is a term d in $\beta\delta$ -normal form in Γ such that $\Gamma \vdash b \rightarrow_{\beta\delta} d$.

Definition 5.2.2. Let λS_δ be a DPTS and $\rightarrow_\rho$ a reduction relation. Then λS_δ is ρ -weakly normalizing if a ρ -weakly normalize in Γ for all $a \in \mathcal{T}_\delta$ and $\Gamma \in \mathcal{C}_\delta$ such that $\Gamma \vdash_{\lambda S_\delta} a : A$.

Theorem 5.2.3. Let λS be a PTS. λS is β -weakly normalizing if and only if λS_δ is $\beta\delta$ -weakly normalizing.

Proof: Suppose $\Gamma \vdash_{\lambda S_\delta} a : A$. Then $|\Gamma|_{\lambda S} \vdash |a|_\Gamma : |A|_\Gamma$. There exists $b \in \mathcal{T}$ such that $|a|_\Gamma \rightarrow_\beta b$. Hence $\Gamma \vdash a \rightarrow_\delta |a|_\Gamma \rightarrow_\beta b$.

The converse is obvious. □

Definition 5.2.4. Let $\Gamma \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$. The term b $\beta\delta$ -strongly normalizes in Γ if there is no infinite $\beta\delta$ -reduction starting at b in Γ .

Definition 5.2.5. Let λS_δ be a DPTS and $\rightarrow_\rho$ a reduction relation. Then λS_δ is ρ -strongly normalizing if a ρ -strongly normalize in Γ for all $a \in \mathcal{T}_\delta$ and $\Gamma \in \mathcal{C}_\delta$ such that $\Gamma \vdash_{\lambda S_\delta} a : A$.

The function $|\cdot|_\Gamma$ may not map infinite β reduction sequences to infinite β reduction sequences as the following example shows:

Example 5.2.6. Suppose there is an infinite β -reduction sequence starting at a and hence at $(x = a : A \text{ in } b)$. If $x \notin FV(b)$ and b is a β normal form then there is no β -reduction sequence starting at $|x=a:A \text{ in } b|_\Gamma \equiv b$.

This means that the function $|\cdot|_\Gamma$ can not be used to establish a relation between PTS's and DPTS's that are strongly normalizing. A new function $\{-\}_\Gamma$ will be defined.

Definition 5.2.7. The mapping $\{-\}_\Gamma : \mathcal{T}_\delta \times \mathcal{C}_\delta \rightarrow \mathcal{T}$ is defined as follows:

$$\begin{aligned} \{x\}_\Gamma &= \begin{cases} \{a\}_{\Gamma_1} & \text{if } \Gamma \equiv \Gamma_1, x = a : A, \Gamma_2 \\ x & \text{otherwise} \end{cases} \\ \{c\}_\Gamma &= c \quad \text{if } c \in \mathcal{C} \\ \{a \ b\}_\Gamma &= \{a\}_\Gamma \{b\}_\Gamma \\ \{\lambda x:A. b\}_\Gamma &= \lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x:A} \\ \{\Pi x:A. b\}_\Gamma &= \Pi x:\{A\}_\Gamma. \{b\}_{\Gamma, x:A} \\ \{x = a : A \text{ in } b\}_\Gamma &= (\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma \end{aligned}$$

Like $|\cdot|_\Gamma$, the value $\{a\}_\Gamma$ is the unfolding of all the definitions occurring in Γ and in a . However $\{-\}_\Gamma$ differs from $|\cdot|_\Gamma$ in the value given for $(x=a:A \text{ in } b)$. Instead of removing the local definition it is translated to a β -redex, an application and an abstraction.

Example 5.2.8. Recall that in example 1.3 we show that $(\lambda x:A. b)a$ may not be typable when $(x=a:A \text{ in } b)$ is. Let $e \equiv \lambda\alpha:*. (x=\alpha:*. \lambda y:\alpha. \lambda f:\alpha \rightarrow \alpha. fy)$ be the term used in example 1.3. The corresponding term expressed as an application and an abstraction is not typable in any system of the λ -cube. But

$$\{e\}_\Gamma \equiv \lambda\alpha:*. (\lambda x:*. \lambda y:\alpha. \lambda f:\alpha \rightarrow \alpha. fy)\alpha$$

is typable in $\lambda 2$. This is because the definition of x is unfolded by α and then x does not occur in the expression $\lambda y:\alpha. \lambda f:\alpha \rightarrow \alpha. fy$.

The mapping $\{-\}_\Gamma$ is extended to contexts.

Definition 5.2.9. The mapping $\{-\} : \mathcal{C}_\delta \rightarrow \mathcal{C}$ is defined as follows:

$$\begin{aligned} \{\epsilon\} &= \epsilon \\ \{\Gamma, x : A\} &= \{\Gamma\}, x : \{A\}_\Gamma \\ \{\Gamma, x = a : A\} &= \{\Gamma\}, x : \{A\}_\Gamma \end{aligned}$$

Similar properties proved for the projection $|-|_-$ will be proved for the function $\{-\}_-$.

Lemma 5.2.10.

1. If x is Γ -fresh and $x \notin FV(b)$ then $x \notin FV(\{b\}_\Gamma)$.
2. Let $\langle \Gamma_1, \Gamma_2, \Gamma_3 \rangle \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$ be such that $(FV_{\Gamma_3}(b)) \cap Def(\Gamma_2) = \emptyset$. Then

$$\{b\}_{\Gamma_1, \Gamma_2, \Gamma_3} \equiv \{b\}_{\Gamma_1, \Gamma_3}.$$

3. Let $\langle \Gamma_1, y = a : A, \Gamma_2 \rangle \in \mathcal{C}_\delta$. Then $\{a\}_{\Gamma_1, y=a:A, \Gamma_2} \equiv \{a\}_{\Gamma_1}$.

Proof:

It follows from the previous part. □

Lemma 5.2.11. $\{b\}_\Gamma[x := \{a\}_\Gamma] \equiv \{b[x := a]\}_\Gamma \equiv \{b\}_{\Gamma, x=a:A}$

Proof: Induction on the structure of b . □

By the following lemma, $\{-\}_-$ maps an infinite $\beta\delta$ reduction sequence to an infinite β reduction sequence.

Lemma 5.2.12.

1. If $c \rightarrow_\beta d$ then $\{c\}_\Gamma \rightarrow_\beta^+ \{d\}_\Gamma$.
2. If $\Gamma \vdash c \rightarrow_\delta d$ then $\{c\}_\Gamma \rightarrow_\beta^{\equiv} \{d\}_\Gamma$.

Proof: Both parts are proved by induction on the structure of c .

1. Suppose c is an application, $c \equiv (a \ b)$. There are two possibilities for the reduction $c \rightarrow_\beta d$:
 - $d \equiv (a' \ b')$ with $(a \rightarrow_\beta a' \wedge b \equiv b')$ or $(a \equiv a' \wedge b \rightarrow_\beta b')$. Then it follows immediately from the IH that $\{a\}_\Gamma \{b\}_\Gamma \rightarrow_\beta^+ \{a'\}_\Gamma \{b'\}_\Gamma$.
 - $a \equiv (\lambda x:A. B)$ and $d \equiv B[x := b]$. Then

$$\begin{aligned} \{c\}_\Gamma &\equiv \{a\}_\Gamma \{b\}_\Gamma \\ &\equiv (\lambda x:\{A\}_\Gamma. \{B\}_\Gamma) \{b\}_\Gamma \\ &\rightarrow_\beta \{B\}_\Gamma[x := \{b\}_\Gamma] \\ &\equiv \{B[x := b]\}_\Gamma && \text{by lemma 5.2.11} \\ &\equiv \{d\}_\Gamma \end{aligned}$$

2. Only two cases are considered.

- (a) Suppose $c \equiv (x=a:A \text{ in } b)$. There are 4 possibilities for the reduction $c \rightarrow_\delta d$:
 - $d \equiv (x=a:A' \text{ in } b)$ with $\Gamma \vdash A \rightarrow_\delta A'$. Then it follows from the IH that $(\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x=a:A}) \{a\}_\Gamma \rightarrow_\beta^{\equiv} (\lambda x:\{A'\}_\Gamma. \{b\}_{\Gamma, x=a:A}) \{a\}_\Gamma$.

- $d \equiv (x=a':A \text{ in } b)$ with $\Gamma \vdash a \rightarrow_\delta a'$. Then it follows from the IH and lemma 4.1.10 that

$$\begin{aligned} (\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma &= (\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x:=\{a\}_\Gamma})\{a\}_\Gamma \\ &\rightarrow_{\beta}^{\equiv} (\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x:=\{a'\}_\Gamma})\{a'\}_\Gamma \end{aligned}$$

- $d \equiv (x=a:A \text{ in } b')$ with $\Gamma, x=a:A \vdash b \rightarrow_\delta b'$. Then it follows from the IH that $(\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma \rightarrow_{\beta}^{\equiv} (\lambda x:\{A\}_\Gamma. \{b'\}_{\Gamma, x=a:A})\{a\}_\Gamma$.
- Suppose $x \notin FV(b)$ and $d \equiv b$.

$$\begin{aligned} \{x = a : A \text{ in } b\} &\equiv (\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma \\ &\rightarrow_{\beta} \{b\}_{\Gamma, x=a:A}[\{a\}_\Gamma] \\ &\equiv \{b\}_{\Gamma, x=a:A} && \text{by lemma 5.2.10} \\ &\equiv \{b\}_\Gamma \end{aligned}$$

(b) Suppose $c \equiv x$. This means that $\Gamma \equiv \Gamma_1, x = d : D, \Gamma_2$ and $\Gamma \vdash x \rightarrow_\delta d$.

By the definition of $\{-\}_-$ and lemma 5.2.10 part 3 we have that $\{x\}_{\Gamma_1, x=d:D, \Gamma_2} \equiv \{d\}_{\Gamma_1}$, and $\{d\}_{\Gamma_1} \equiv \{d\}_{\Gamma_1, x=d:D, \Gamma_2}$.

□

Corollary 5.2.13.

1. If $\Gamma \vdash B \rightarrow_\delta B'$, then $\{B\}_\Gamma \rightarrow_\beta \{B'\}_\Gamma$.
2. If $\Gamma \vdash B =_{\beta\delta} B'$, then $\{B\}_\Gamma =_\beta \{B'\}_\Gamma$.

Proof: These follow from part 2 of the previous lemma. □

Definition 5.2.14. The specification $S = (\mathbf{S}, \mathbf{A}, \mathbf{R})$ is called *quasi full* if for all $s_1, s_2 \in \mathbf{S}$ there exists $s_3 \in \mathbf{S}$ such that $(s_1, s_2, s_3) \in \mathbf{R}$.¹

Definition 5.2.15. Let $S = (\mathbf{S}, \mathbf{A}, \mathbf{R})$ and $S' = (\mathbf{S}', \mathbf{A}', \mathbf{R}')$ be such that

1. $\mathbf{S} \subseteq \mathbf{S}'$, $\mathbf{A} \subseteq \mathbf{A}'$, and $\mathbf{R} \subseteq \mathbf{R}'$
2. S' is quasi-full
3. for all $s \in \mathbf{S}$ there is a sort $s' \in \mathbf{S}'$ such that $(s : s') \in \mathbf{A}'$ (i.e. the sorts of S are not topsorts in S').

Then the specification S' is called a *completion* of S .

Example 5.2.16. The system λC_∞ is a completion of λC , $\lambda \mathbf{HOL}$ and itself.

This definition is necessary in order to prove that $\{-\}_-$ maps terms that are typable in a DPTS λS_δ to terms that are typable in a slightly larger PTS, i.e. a PTS $\lambda S'$ with S' a completion of S . Remember that $\{-\}_-$ translates a local definition to a λ -abstraction with an argument:

$$\{x=a:A \text{ in } b\}_\Gamma = (\lambda x:\{A\}_\Gamma. \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma$$

Condition 2 is necessary to ensure that all these λ -abstractions introduced by $\{-\}_-$ are allowed in S' . The typing of the abstraction is restricted by the set $\mathbf{R}$ of rules whereas the typing of $(x=a:A \text{ in } b)$ is not. Let $e \equiv (x=\alpha : * \text{ in } \lambda y:x. \lambda f:x \rightarrow x. fy)$ be the term in the example 1.4. This term is typable in the system $(\lambda_-)_\delta$ but the term $\{e\}_\epsilon \equiv (\lambda x : *. \lambda y : \alpha. \lambda f : \alpha \rightarrow \alpha. fy)\alpha$ is not typable in λ_- .

¹ Note that if a specification is full then it is quasi-full. But the converse is not true. A specification is full if for all $s_1, s_2 \in \mathbf{S}$ $(s_1, s_2, s_3) \in \mathbf{R}$.

Condition 3 is necessary because we can not type these abstractions introduced by $\{-\}_-$ if A is a topsort. For example the term $(x = * : \square \text{ in } x)$ is typable in λC_δ but

$$\{x = * : \square \text{ in } x\}_\epsilon \equiv (\lambda x : \square. *)^*$$

is not typable in λC .

Lemma 5.2.17. *Let $c \in C$, $a, A \in T_\delta$ and $\Gamma \in T_\delta$. If c occurs in A or a or Γ and $\Gamma \vdash_{\lambda S_\delta} a : A$ then $c \in S$.*

Proof: By induction on the derivation of $\Gamma \vdash_{\lambda S_\delta} a : A$. □

The next lemma states that any $\lambda S'$ type that is in the range of $\{-\}_-$ cannot be one of the topsorts of $\lambda S'$:

Lemma 5.2.18. *Let $S = (S, A, R)$ and $S' = (S', A', R')$ be such that S' is a completion of S . Then If $\Delta \vdash_{\lambda S_\delta} a : A$ and $\Delta \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma$ then $\Delta \vdash_{\lambda S'} \{A\}_\Gamma : s$*

Proof: Assume $\Delta \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma$. By correctness of types $\Delta \vdash_{\lambda S'} \{A\}_\Gamma : s'$ or $\{A\}_\Gamma \equiv s$. Suppose $\{A\}_\Gamma \equiv s$. Since $\Delta \vdash_{\lambda S_\delta} a : A$ and by lemma 5.2.17 we have that $s \in S$. By (3) there is a sort $s' \in S'$ such that $(s : s') \in A'$ and hence $\Delta \vdash_{\lambda S'} s : s'$, i.e. $\Delta \vdash_{\lambda S'} \{A\}_\Gamma : s'$. □

The next theorem states that $\{-\}_-$ maps terms that are typable in a DPTS λS_δ to terms that are typable in a slightly larger PTS, i.e. $\lambda S'$ with S' a completion of S .

Theorem 5.2.19. *Let $S = (S, A, R)$ and $S' = (S', A', R')$ be such that S' is a completion of S . Then $\Gamma \vdash_{\lambda S_\delta} a : A \Rightarrow \{\Gamma\} \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma$.*

Proof: By induction on the derivation of $\Gamma \vdash_{\lambda S_\delta} a : A$. Suppose the last step in the derivation is

- (δ -start)
$$\frac{\Gamma \vdash_{\lambda S_\delta} a : A}{\Gamma, x = a : A \vdash_{\lambda S_\delta} x : A}$$

By IH $\{\Gamma\} \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma$. By definition we have that $\{x\}_{\Gamma, x=a:A} = \{a\}_\Gamma$. By lemma 5.2.10 part 2 we have that $\{A\}_{\Gamma, x=a:A} = \{A\}_\Gamma$. It follows from lemma 5.2.18 that $\{\Gamma\} \vdash_{\lambda S'} \{A\}_\Gamma : s$. By weakening rule $\{\Gamma\}, x : \{A\}_\Gamma \vdash_{\lambda S'} \{x\}_{\Gamma, x=a:A} : \{A\}_\Gamma$.

- (δ -weakening)
$$\frac{\Gamma \vdash_{\lambda S_\delta} b : B \quad \Gamma \vdash_{\lambda S_\delta} a : A}{\Gamma, x = a : A \vdash_{\lambda S_\delta} b : B}$$

By IH $\{\Gamma\} \vdash_{\lambda S'} \{b\}_\Gamma : \{B\}_\Gamma$ and $\{\Gamma\} \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma$. Then by lemma 5.2.18 $\{\Gamma\} \vdash_{\lambda S'} \{A\}_\Gamma : s$. By weakening rule $\{\Gamma\}, x : \{A\}_\Gamma \vdash_{\lambda S'} \{b\}_\Gamma : \{B\}_\Gamma$.

We have that $x \notin FV(b)$ and $x \notin FV(B)$. By lemma 5.2.10 $\{b\}_{\Gamma, x=a:A} \equiv \{b\}_\Gamma$ and $\{B\}_{\Gamma, x=a:A} \equiv \{B\}_\Gamma$.

- δ -conversion
$$\frac{\Gamma \vdash_{\lambda S_\delta} b : B \quad \Gamma \vdash_{\lambda S_\delta} B' : s \quad \Gamma \vdash_{\lambda S_\delta} B =_\delta B'}{\Gamma \vdash b : B}$$

It follows from IH that $\{\Gamma\} \vdash_{\lambda S'} \{b\}_\Gamma : \{B\}_\Gamma$ and $\{\Gamma\} \vdash_{\lambda S'} \{B'\}_\Gamma : s$. By corollary 5.2.13 part 2 it follows from $\Gamma \vdash B =_\delta B'$ that $\{B\}_\Gamma =_\beta \{B'\}_\Gamma$. Then using β -conversion rule $\{\Gamma\} \vdash_{\lambda S'} \{b\}_\Gamma : \{B'\}_\Gamma$.

- δ -formation
$$\frac{\Gamma, x = a : A \vdash_{\lambda S_\delta} B : s}{\Gamma \vdash_{\lambda S_\delta} (x = a : A \text{ in } B) : s}$$

By IH

$$\{\Gamma, x = a : A\} \vdash_{\lambda S'} \{B\}_{\Gamma, x=a:A} : s \tag{5.1}$$

The derivation of $\Gamma, x = a : A \vdash_{\lambda S \delta} B : s$ contains a (shorter) derivation of $\Gamma \vdash_{\lambda S \delta} a : A$, so also by IH

$$\{\Gamma\} \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma \quad (5.2)$$

By lemma 5.2.18 it follows from (5.1) and (5.2) that there are $s_1, s_2 \in S'$ such that

$$\{\Gamma, x = a : A\} \vdash_{\lambda S'} s : s_2 \quad (5.3)$$

$$\{\Gamma\} \vdash_{\lambda S'} \{A\}_\Gamma : s_1 \quad (5.4)$$

The following is a derivation of $\{\Gamma\} \vdash_{\lambda S'} \{x = a : A \text{ in } B\}_\Gamma : s$.

$$\frac{\frac{\frac{(5.3) \quad (5.4)}{\{\Gamma\} \vdash_{\lambda S'} (\prod x \in \{A\}_\Gamma. s) : s_3} (prod) \quad (5.1)}{\{\Gamma\} \vdash_{\lambda S'} (\lambda x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A}) : (\prod x \in \{A\}_\Gamma. s)} (abs) \quad (5.2)}{\{\Gamma\} \vdash_{\lambda S'} (\lambda x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A}) \{a\}_\Gamma : s[x := \{a\}_\Gamma]} (app)$$

and $(\lambda x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A}) \{a\}_\Gamma \equiv \{x = a : A \text{ in } B\}_\Gamma$.

• δ -introduction
$$\frac{\Gamma, x = a : A \vdash_{\lambda S \delta} b : B \quad \Gamma \vdash_{\lambda S \delta} (x = a : A \text{ in } B) : s}{\Gamma \vdash_{\lambda S \delta} (x = a : A \text{ in } b) : (x = a : A \text{ in } B)}$$

By IH

$$\{\Gamma, x = a : A\} \vdash_{\lambda S'} \{b\}_{\Gamma, x=a:A} : \{B\}_{\Gamma, x=a:A} \quad (5.1)$$

$$\{\Gamma\} \vdash_{\lambda S'} \{x = a : A \text{ in } B\}_\Gamma : s \quad (5.2)$$

The derivation of $\Gamma, x = a : A \vdash_{\lambda S \delta} b : B$ contains a (shorter) derivation of $\Gamma \vdash_{\lambda S \delta} a : A$, so also by the IH

$$\{\Gamma\} \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma \quad (5.3)$$

By lemma 5.2.18 it follows from (5.1) and (5.3) that there are $s_1, s_2 \in S'$ such that

$$\{\Gamma, x = a : A\} \vdash_{\lambda S'} \{B\}_{\Gamma, x=a:A} : s_2 \quad (5.4)$$

$$\{\Gamma\} \vdash_{\lambda S'} \{A\}_\Gamma : s_1 \quad (5.5)$$

Then

$$\frac{\frac{\frac{(5.4) \quad (5.5)}{\{\Gamma\} \vdash_{\lambda S'} (\prod x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A}) : s_3} (prod) \quad (5.1)}{\{\Gamma\} \vdash_{\lambda S'} (\lambda x \in \{A\}_\Gamma. \{b\}_{\Gamma, x=a:A}) : (\prod x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A})} (abs) \quad (5.3)}{\{\Gamma\} \vdash_{\lambda S'} (\lambda x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A}) \{a\}_\Gamma : \{B\}_{\Gamma, x=a:A} [x := \{a\}_\Gamma]} (app)$$

$\{x = a : A \text{ in } b\}_\Gamma \equiv (\lambda x \in \{A\}_\Gamma. \{b\}_{\Gamma, x=a:A}) \{a\}_\Gamma$ and

$$\begin{aligned} \{x = a : A \text{ in } B\}_\Gamma &\equiv (\lambda x \in \{A\}_\Gamma. \{B\}_{\Gamma, x=a:A}) \{a\}_\Gamma \\ &=_{\beta} \{B\}_{\Gamma, x=a:A} [x := \{a\}_\Gamma] \end{aligned} \quad (5.6)$$

so using the conversion rule

$$\frac{\{\Gamma\} \vdash_{\lambda S'} \{x = a : A \text{ in } b\}_\Gamma : \{B\}_{\Gamma, x=a:A} [x := \{a\}_\Gamma] \quad (5.2) \quad (5.6)}{\{\Gamma\} \vdash_{\lambda S'} \{x = a : A \text{ in } b\}_\Gamma : \{x = a : A \text{ in } B\}_\Gamma} (\beta - conv)$$

The rest of the cases are easy to prove. $\square$

Theorem 5.2.20. *Let $S = (S, A, R)$ and $S' = (S', A', R')$ be such that S' is a completion of S . If the PTS $\lambda S'$ is β -strongly normalizing, then the DPTS $\lambda S \delta$ is $\beta\delta$ -strongly normalizing.*

Proof: Suppose that $\lambda S'$ is β -strongly normalizing, and suppose towards a contradiction that λS_δ is not $\beta\delta$ -strongly normalizing, i.e. there is an infinite $\beta\delta$ -reduction sequence starting at a and $\Gamma \vdash_{\lambda S_\delta} a : A$.

Observe that the number of β -reductions in this sequence is infinite, i.e. $\forall n \in \mathbb{N} \exists m > n : \Gamma \vdash a_m \rightarrow_\beta a_{m+1}$. Otherwise it would follow that there is $n_0 \in \mathbb{N}$ such that $\forall m > n_0 \Gamma \vdash a_m \rightarrow_\delta a_{m+1}$. Hence the sequence $a_{n_0+1} \rightarrow_\delta a_{n_0+2} \rightarrow_\delta \dots$ would be infinite. As δ is strongly normalizing, this can not happen. Hence the number of β -reduction steps in the sequence $\Gamma \vdash a \rightarrow_{\beta\delta} a_1 \rightarrow_{\beta\delta} a_2 \rightarrow_{\beta\delta} \dots$ is infinite. Then this sequence is of the form

$$\Gamma \vdash a \rightarrow_\delta a_{n_1} \rightarrow_\beta a_{n_2} \rightarrow_\delta a_{n_3} \rightarrow_\beta a_{n_4} \rightarrow_\delta a_{n_5} \rightarrow_\beta a_{n_6} \rightarrow_\delta \dots$$

By lemma 5.2.12 and corollary 5.2.13 part 1 there is an infinite β -reduction sequence starting at $\{a\}_\Gamma$:

$$\{a\}_\Gamma \rightarrow_\beta \{a_{n_1}\}_\Gamma \rightarrow_\beta^+ \{a_{n_2}\}_\Gamma \rightarrow_\beta \{a_{n_3}\}_\Gamma \rightarrow_\beta^+ \{a_{n_4}\}_\Gamma \rightarrow_\beta \{a_{n_5}\}_\Gamma \rightarrow_\beta^+ \{a_{n_6}\}_\Gamma \rightarrow_\beta \dots$$

and by lemma 5.2.19 $\{\Gamma\} \vdash_{\lambda S'} \{a\}_\Gamma : \{A\}_\Gamma$, which contradicts the assumption that $\lambda S'$ is β -strongly normalizing. $\square$

Corollary 5.2.21. *The following systems are strongly normalizing:*

1. *The system λC_∞ extended with definitions, i.e. $(\lambda C_\infty)_\delta$.*
2. *The calculus of constructions extended with definitions, i.e. λC_δ .*
3. *The system of higher order logic extended with definitions, i.e. $\lambda \text{HOL}_\delta$ is strongly normalizing.*

Proof: The system ECC (see [Luo]) is strongly normalizing and contains λC_∞ . Then λC_∞ is strongly normalizing too. This system λC_∞ is a completion of itself. Hence it follows from the previous theorem that λC_∞ extended with definitions is strongly normalizing.

Since $(\lambda C_\infty)_\delta$ contains λC_δ and $\lambda \text{HOL}_\delta$, part 2 and 3 follow from part 1. $\square$

Theorem 5.2.20 is somewhat unsatisfactory. It would be nicer to prove a stronger property, namely that a DPTS λS_δ is $\beta\delta$ -strongly normalizing if the PTS λS is β -strongly normalizing. On the other hand, we do not know any strongly normalizing PTS λS for which theorem 5.2.20 cannot be used to prove strong normalization of λS_δ . In particular, all strongly normalizing PTS's given in [Bar92] have a completion that is λC_∞ .

Appendix A

Pure Type Systems with definitions

Definition A.1.

The set T_δ of pseudoterms and the set C_δ of contexts is given by

$$\begin{aligned} T_\delta &::= V \mid \mathbf{C} \mid (T_\delta \ T_\delta) \mid (\lambda V:T_\delta. T_\delta) \mid (\Pi V:T_\delta. T_\delta) \mid (V=T_\delta:T_\delta \text{ in } T_\delta) \\ C_\delta &::= \epsilon \mid \langle C_\delta, V:T_\delta \rangle \mid \langle C_\delta, V=T_\delta:T_\delta \rangle \end{aligned}$$

where V is the set of variables and $\mathbf{C}$ is the set of constants.

Definition A.2.

Let be $\Gamma \in C_\delta$, and $a, a' \in T_\delta$. The concept "the term a δ -reduces to the term a' in the context Γ " is written as $\Gamma \vdash a \rightarrow_\delta a'$ and is defined by the following rules:

$$\begin{array}{c} \Gamma_1, x=a:A, \Gamma_2 \vdash x \rightarrow_\delta a \quad (\delta) \\ \Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta b \quad \text{if } x \notin FV(b) \quad (\delta) \\ \frac{\Gamma, x=a:A \vdash b \rightarrow_\delta b'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A \text{ in } b')} \quad (\delta) \\ \frac{\Gamma \vdash a \rightarrow_\delta a'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a':A \text{ in } b)} \quad \vdots \\ \frac{\Gamma \vdash A \rightarrow_\delta A'}{\Gamma \vdash (x=a:A \text{ in } b) \rightarrow_\delta (x=a:A' \text{ in } b)} \\ \frac{\Gamma \vdash a \rightarrow_\delta a'}{\Gamma \vdash (ab) \rightarrow_\delta (a'b)} \quad \frac{\Gamma \vdash b \rightarrow_\delta b'}{\Gamma \vdash (ab) \rightarrow_\delta (ab')} \\ \frac{\Gamma, x:A \vdash a \rightarrow_\delta a'}{\Gamma \vdash (\lambda x:A. a) \rightarrow_\delta (\lambda x:A. a')} \quad \frac{\Gamma \vdash A \rightarrow_\delta A'}{\Gamma \vdash (\lambda x:A. a) \rightarrow_\delta (\lambda x:A'. a)} \\ \frac{\Gamma, x:A \vdash a \rightarrow_\delta a'}{\Gamma \vdash (\Pi x:A. a) \rightarrow_\delta (\Pi x:A. a')} \quad \frac{\Gamma \vdash A \rightarrow_\delta A'}{\Gamma \vdash (\Pi x:A. a) \rightarrow_\delta (\Pi x:A'. a)} \end{array}$$

When Γ is the empty context, it is written $a \rightarrow_\delta a'$ instead of $\Gamma \vdash a \rightarrow_\delta a'$.

The relation $\rightarrow_\beta$ between terms in $\mathcal{T}$ will be extended to terms in T_δ .

Definition A.3.

Let $\Gamma \in C_\delta$, $b, B \in T_\delta$. The DPTS determined by the specification $S = (S, \mathbf{A}, \mathbf{R})$ is denoted as $\lambda S_\delta = \lambda(S, \mathbf{A}, \mathbf{R})_\delta$ and defined by the notion of type derivation $\Gamma \vdash b : B$ given by the following axioms and rules:

(axiom)

$c \vdash c : s$

for $c : s \in \mathbf{A}$

(start)	$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$	where x is Γ -fresh
(weakening)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B}$	where x is Γ -fresh
(formation)	$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x:A. B) : s_3}$	for $(s_1, s_2, s_3) \in \mathcal{R}$
(abstraction)	$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x:A. B) : s}{\Gamma \vdash (\lambda x:A. b) : (\Pi x:A. B)}$	
(application)	$\frac{\Gamma \vdash b : (\Pi x:A. B) \quad \Gamma \vdash a : A}{\Gamma \vdash (b \ a) : B[x := a]}$	
(conversion)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad B =_{\beta} B'}{\Gamma \vdash b : B'}$	
(δ - start)	$\frac{\Gamma \vdash a : A}{\Gamma, x=a:A \vdash x : A}$	where x is Γ -fresh
(δ - weakening)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash a : A}{\Gamma, x=a:A \vdash b : B}$	where x is Γ -fresh
(δ - formation)	$\frac{\Gamma, x=a:A \vdash B : s}{\Gamma \vdash x=a:A \text{ in } B : s}$	
(δ - introduction)	$\frac{\Gamma, x=a:A \vdash b : B \quad \Gamma \vdash x=a:A \text{ in } B : s}{\Gamma \vdash x=a:A \text{ in } b : x=a:A \text{ in } B}$	
(δ - conversion)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad \Gamma \vdash B =_{\delta} B'}{\Gamma \vdash b : B'}$	

where s ranges over sorts, i.e. $s \in S$.

Appendix B

Generalization for Abstract Reduction Systems

The notions of *Abstract Reduction Systems (ARS)* and *sub-ARS* are presented as in [Klo90].

Definition B.1.

An abstract reduction system (ARS) is a structure $\mathcal{A} = \langle A, (\rightarrow_\alpha)_{\alpha \in I} \rangle$ such that

- A is a set
- For all $\alpha \in I$, $\rightarrow_\alpha \subseteq A \times A$. These binary relations are called reduction or rewrite relations.

Definition B.2. The relation $\twoheadrightarrow_\alpha$ is the transitive, reflexive closure of $\rightarrow_\alpha$.

Definition B.3.

Let $\mathcal{A} = \langle A, \rightarrow_\alpha \rangle$ and $\mathcal{B} = \langle B, \rightarrow_\beta \rangle$ be two ARS's. The concept " $\mathcal{A}$ is a sub-ARS of $\mathcal{B}$ " is written as $\mathcal{A} \subseteq \mathcal{B}$ and defined as follows:

- $A \subseteq B$
- $\rightarrow_\alpha$ is the restriction of $\rightarrow_\beta$, i.e. $\forall a, a' \in A (a \rightarrow_\alpha a' \Leftrightarrow a \rightarrow_\beta a')$

$\mathcal{B}$ is also called an extension of $\mathcal{A}$.

Theorem B.4.

Let $\mathcal{A} = \langle A, \rightarrow_\beta \rangle$ and $\mathcal{A}' = \langle A', \rightarrow_{\beta'} \rangle$ be ARS's be such that $\mathcal{A}'$ is an extension of $\mathcal{A}$. Let $\mathcal{F} = \langle A', \rightarrow_\delta \rangle$ be an ARS and suppose there is a mapping $\pi : A' \rightarrow A$ such that:

- a) $\pi|_A = id$
- b) $\forall a \in A' (a \rightarrow_\delta \pi(a))$
- c) If $a \rightarrow_\delta b$ then $\pi(a) = \pi(b)$ for all $a, b \in A'$.
- d) if $a \rightarrow_{\beta'} b$ then $\pi(a) \rightarrow_\beta \pi(b)$, for all $a, b \in A'$.

1. From b) and c) the following is deduced:

δ is Church-Rosser

2. From b), c) and d) the following conclusion is deduced:

If β is Church-Rosser so is $\beta'\delta$.

3. From a), b) and c) the following conclusion is deduced:

Suppose that A is closed with respect to δ (i.e. $a \rightarrow_\delta b$ and $a \in A$ then $b \in A$) and that $\Delta = \{(d, d), d \rightarrow_\delta d\} = \emptyset$.

Then δ is weakly normalizing.

Proof:

1. The proof is shown in the following diagram.

2. The proof is shown in the following diagram.

3. Let be $a \in A'$. It follows from hypothesis b that $a \rightarrow_\delta \pi(a)$. Suppose $\pi(a)$ is not in δ -nf then $\pi(a) \rightarrow_\delta d$. Since A is closed with respect to δ , it follows that $d \in A$. It follows from hypothesis a) and c) that $\pi(a) = \pi(\pi(a)) = \pi(d) = d$. Then $d \rightarrow_\delta d$. Since $\Delta = \emptyset$, it follows that $\pi(a)$ is in δ -nf.

□

Example B.5.

In the case of Chapter 4,

- $\mathcal{A}_\Gamma \equiv \langle \{a \in T, FV(a) \cap Def(\Gamma) = \emptyset\}, \rightarrow_\beta \rangle$
- $\mathcal{A}'_\Gamma \equiv \langle T_\delta, \rightarrow_\beta \rangle$
- $\mathcal{F}_\Gamma \equiv \langle T_\delta, \rightarrow_{\delta_\Gamma} \rangle$
- $a \rightarrow_{\delta_\Gamma} b$ if and only if $\Gamma \vdash a \rightarrow_\delta b$

- $\pi_\Gamma \equiv | - |_\Gamma$

Parts 1 and 2 of theorem B.4 are the generalization of the proofs in Section 4.2 for Church-Rosser for $\rightarrow_\delta$ and for $\rightarrow_{\beta\delta}$. Part 3 of this theorem is somewhat different from the proof of weak normalization for $\rightarrow_\delta$ presented in Section 4.3. In that section in order to prove that $\pi(a)$ is in δ -nf, it is proved that all elements in A are in δ -nf.

Theorem B.6. *Let $\mathcal{A} = \langle A, \rightarrow_\beta \rangle$, $\mathcal{A}' = \langle A', \rightarrow_{\beta'} \rangle$ and $\mathcal{F} = \langle A', \rightarrow_\delta \rangle$ be ARS's. Suppose there is a mapping $\pi : A' \rightarrow A$ such that:*

- a) *If $a \rightarrow_\delta b$ then $\pi(a) \rightarrow_\beta \pi(b)$ for all $a, b \in A'$.*
- b) *if $a \rightarrow_{\beta'} b$ then $\pi(a) \rightarrow_\beta^+ \pi(b)$, for all $a, b \in A'$.*

If δ and β are strongly normalizing. Then $\beta'\delta$ is strongly normalizing.

Proof: Suppose towards a contradiction that $\beta'\delta$ is not strongly normalizing, i.e. there is an infinite $\beta'\delta$ -reduction sequence starting at $a \in A'$.

Observe that the number of β' -reductions in this sequence is infinite, i.e. $\forall n \in \mathbb{N} \exists m > n : a_m \rightarrow_{\beta'} a_{m+1}$. Otherwise it would follow that there is $n_0 \in \mathbb{N}$ such that $\forall m > n_0 a_m \rightarrow_\delta a_{m+1}$. Hence the sequence $a_{n_0+1} \rightarrow_\delta a_{n_0+2} \rightarrow_\delta \dots$ would be infinite. As δ is strongly normalizing, this can not happen. Hence the number of β' -reduction steps in the sequence $a \rightarrow_{\beta'\delta} a_1 \rightarrow_{\beta'\delta} a_2 \dots$ is infinite. Then this sequence is of the form

$$a \rightarrow_\delta a_{n_1} \rightarrow_{\beta'} a_{n_2} \rightarrow_\delta a_{n_3} \rightarrow_{\beta'} a_{n_4} \rightarrow_\delta a_{n_5} \rightarrow_{\beta'} a_{n_6} \rightarrow_\delta \dots$$

By hypothesis a) and b) there is an infinite β -reduction sequence starting at $\pi(a)$:

$$\pi(a) \rightarrow_\beta \pi(a_{n_1}) \rightarrow_\beta^+ \pi(a_{n_2}) \rightarrow_\beta \pi(a_{n_3}) \rightarrow_\beta^+ \pi(a_{n_4}) \rightarrow_\beta \pi(a_{n_5}) \rightarrow_\beta^+ \pi(a_{n_6}) \rightarrow_\beta \dots$$

which contradicts the assumption that β is strongly normalizing. □

Example B.7.

In the case of Chapter 5,

- $A_\Gamma \equiv \{d \in T, \exists D (\{\Gamma\} \vdash_{\lambda S'} d : D)\}$ where S' is a completion of S .
- $A'_\Gamma = \{d \in T_\delta, \exists D (\Gamma \vdash_{\lambda S_\delta} d : D)\}$
- $\mathcal{A}_\Gamma \equiv \langle A_\Gamma, \rightarrow_\beta \rangle$
- $\mathcal{A}'_\Gamma \equiv \langle A'_\Gamma, \rightarrow_\beta \rangle$
- $\mathcal{F}_\Gamma \equiv \langle A'_\Gamma, \rightarrow_{\delta_\Gamma} \rangle$
- $a \rightarrow_{\delta_\Gamma} b$ if and only if $\Gamma \vdash a \rightarrow_\delta b$
- $\pi_\Gamma \equiv \{-\}_\Gamma$

Bibliography

- [Bar92] H. Barendregt. *Lambda Calculi with Types*. Handbook of logic in Computer Science, volume II S. Abramsky, D.M. Gabbai and T.S.E. Maibaum (eds.) Oxford University Press, Oxford, 1992.
- [Klo90] J.W. Klop. *Term Rewriting Systems*. Centre for Mathematics and Computer Science Amsterdam, 1990.
- [Luo] Z. Luo. *ECC and extended Calculus of Constructions*. Department of Computer Science, University of Edinburgh.

Contents

1	Introduction	1
2	Pure Type Systems	3
3	Pure Type Systems with definitions	6
3.1	Pseudoterms	6
3.2	Reductions	8
3.3	Types	9
4	Properties of General Pseudoterms	12
4.1	Basic Properties	12
4.2	Church Rosser for β , δ and $\beta\delta$ -reductions	15
4.3	Weak and Strong Normalization for $\rightarrow_\delta$	19
5	Properties of Well-Typed Terms	22
5.1	Basic Properties	22
5.2	Weak and Strong Normalization for $\beta\delta$ -reduction	25
A	Pure Type Systems with definitions	32
B	Generalization for Abstract Reduction Systems	34

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt
H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen
L.A.M. Schoenmakers	Parallel Programs for the Recognition of <i>P</i> -invariant Segments, p. 16.
91/04	E. v.d. Sluis
A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse
P.J. de Bruin
P. Hoogendijk
G. Malcolm
E. Voermans
J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse
P.J. de Bruin
G.Malcolm
E.Voermans
J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts
K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts
P.M.E. de Bra
K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.

91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic process creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Bacten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.
91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.

92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Bacten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.
92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottom-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for F ω , p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Velkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottom-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for $F\omega$, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Velkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.

- 93/15 J.C.M. Baeten
J.A. Bergstra
R.N. Bol A Real-Time Process Logic, p. 31.
- 93/16 H. Schepers
J. Hooman A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
- 93/17 D. Alstein
P. van der Stok Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
- 93/18 C. Verhoef A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
- 93/19 G-J. Houben The Design of an Online Help Facility for ExSpec, p.21.
- 93/20 F.S. de Boer A Process Algebra of Concurrent Constraint Programming, p. 15.
- 93/21 M. Codish
D. Dams
G. Filé
M. Bruynooghe Freeness Analysis for Logic Programs - And Correctness?, p. 24.
- 93/22 E. Poll A Typechecker for Bijective Pure Type Systems, p. 28.
- 93/23 E. de Kogel Relational Algebra and Equational Proofs, p. 23.
- 93/24 E. Poll and Paula Severi Pure Type Systems with Definitions, p. 38.
- 93/25 H. Schepers and R. Gerth A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
- 93/26 W.M.P. van der Aalst Multi-dimensional Petri nets, p. 25.
- 93/27 T. Kloks and D. Kratsch Finding all minimal separators of a graph, p. 11.
- 93/28 F. Kamareddine and
R. Nederpelt A Semantics for a fine λ -calculus with de Bruijn indices, p. 49.
- 93/29 R. Post and P. De Bra GOLD, a Graph Oriented Language for Databases, p. 42.
- 93/30 J. Deogun
T. Kloks
D. Kratsch
H. Müller On Vertex Ranking for Permutation and Other Graphs, p. 11.
- 93/31 W. Körver Derivation of Delay Insensitive and Speed Independent CMOS Circuits, using Directed Commands and Production Rule Sets, p. 39.
- 93/32 H. ten Eikelder and
H. van Geldrop On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
- 93/33 L. Loyens and J. Moonen ILIAS, a sequential language for parallel matrix computations, p. 20.

93/34 J.C.M. Baeten and
J.A. Bergstra

Real Time Process Algebra with Infinitesimals, p.39.