Pure Type Systems with Definitions

Paula Severi Erik Poll *
{severi,erik}@win.tue.nl

Eindhoven University of Technology

appeared in Logical Foundations of Computing Science (LFCS'94), Lecture Nores in Computer Science Vol. 813, pp. 316–328, 1994

Abstract. In this paper, an extension of Pure Type Systems (PTS's) with definitions is presented. We prove this extension preserves many of the properties of PTS's. The main result is a proof that for many PTS's, including the Calculus of Constructions, this extension preserves strong normalisation.

1 Introduction

A large class of typed lambda calculi can be described in a uniform way as Pure Type Systems (PTS's). This includes for instance the second-order λ-calculus [Gir72][Rey74], Edinburgh Logical Framework [HHP93], and the Calculus of Constructions [CH88]. For an introduction to PTS's and their most important properties we refer to [Bar92].

One shortcoming of PTS's is that they do not provide the possibility to introduce definitions, i.e. abbreviations for larger expressions that are used several times. A definition mechanism is *essential* for practical use, and indeed implementations of PTS's such as Coq [Dea91], Lego [LP92] or Constructor [Hel91] do provide such a facility, even though the formal definitions of the systems they implement do not.

In this paper, we introduce an extension of PTS's with definitions. Definitions will be of the form $x = a : A$. A definition $x = a : A$ introduces x as an abbreviation of the term a of type A. Definitions are allowed both in contexts, e.g. $\Gamma, x{=}a{:}A, \ldots$, and in terms, e.g. $(x{=}a{:}A \text{ in } b)$. Definitions in contexts are called *global* definitions, definitions in terms are called *local* definitions. A new reduction relation – *δ-reduction* – is defined for the unfolding of definitions.

The extension of a PTS with definitions looks very harmless, and may not seem a topic worthy of investigation. However, the *local* definitions complicate matters, and *it is an open problem whether extending an arbitrary* PTS *with definitions preserves strong normalisation*! Worse still, proving strong normalisation for particular PTS's extended with definitions is already a problem. The strong normalisation proofs for particular type systems given in [Coq85] [Luo89] [GN91] [Bar92] cannot be extended in any obvious way to prove strong normalisation of these systems extended with definitions.

In this paper we show how strong normalisation of a PTS extended with definitions follows from strong normalisation of another (larger) PTS. This enables us to prove that for all strongly normalising PTS's that we know the extensions with definitions are also strongly normalising.

In the systems of the AUTOMATH family (see [dB80]) definitions are considered as part of the formal language. The meta-theory of these systems – including strong normalisation – is treated in detail in [vD80]. However, the proofs of strong normalisation apply only to the particular type systems that are considered, and do not extend to other type systems.

In section 2, we recall the definition of PTS's. Then in section 3, DPTS's – PTS's with definitions – are introduced. In section 4, some properties of δ-reduction are proved, and in section 5 some properties of DPTS's. Finally in section 6 we consider the problem of strong normalisation for DPTS's.

* supported by the Dutch organization for scientific research (NWO).

2 Pure Type Systems

Pure Type Systems (PTS's) are defined as in [Bar92].

Definition 1. A *specification of a* PTS is a triple $\mathcal{S} = (\mathsf{S}, \mathsf{A}, \mathsf{R})$ such that

- S is a set of symbols called the *sorts*.
- $\mathsf{A} \subseteq \mathsf{S} \times \mathsf{S}$ is the set of *axioms*.
- $\mathsf{R} \subseteq \mathsf{S} \times \mathsf{S} \times \mathsf{S}$ is the set of *rules*. We write (s_1, s_2) for a rule $(s_1, s_2, s_3) \in \mathsf{R}$ if $s_2 = s_3$.

The PTS determined by the specification $\mathcal{S}$ is denoted by $\lambda\mathcal{S}$. It consists of sets of pseudoterms and pseudo-contexts, a reduction relation and a typing relation.

Definition 2. The set T of *pseudoterms* and the set C of *contexts* of the PTS $\lambda(\mathsf{S}, \mathsf{A}, \mathsf{R})$ are defined as follows:

$$\mathsf{T} ::= \mathsf{V} \mid \mathsf{S} \mid (\mathsf{T}\ \mathsf{T}) \mid (\lambda \mathsf{V}{:}\mathsf{T}.\ \mathsf{T}) \mid (\Pi \mathsf{V}{:}\mathsf{T}.\ \mathsf{T})$$
$$\mathsf{C} ::= \epsilon \mid \langle \mathsf{C}, \mathsf{V}{:}\mathsf{T} \rangle$$

where V is the set of variables and ϵ is the empty context.

α-equality is defined as usual, and α-equal terms are identified. We write $\mathrm{FV}(A)$ for the set of the free variables of a term A. β-reduction, denoted by $\to_\beta$, is defined as usual by $(\lambda x{:}A.\ b)\ a \to_\beta b[x := a]$. We write $A \to B$ for $(\Pi x{:}A.\ B)$ if $x \notin \mathrm{FV}(B)$.

Definition 3. A term b *has type* B *in context* Γ in the PTS $\lambda\mathcal{S} = \lambda(\mathsf{S}, \mathsf{A}, \mathsf{R})$ – written $\Gamma \vdash_{\lambda\mathcal{S}} b : B$ – if it can be derived using the following rules (the subscript $\lambda\mathcal{S}$ of $\vdash$ is dropped if it is clear which PTS we mean):

$$(axiom) \quad \epsilon \vdash s : s' \qquad \text{for } s : s' \in \mathsf{A}$$

$$(start) \quad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad \text{where } x \text{ is } \Gamma\text{-fresh}$$

$$(weakening) \quad \frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B} \qquad \text{where } x \text{ is } \Gamma\text{-fresh}$$

$$(\Pi - formation) \quad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.B) : s_3} \qquad \text{for } (s_1, s_2, s_3) \in \mathsf{R}$$

$$(\Pi - introduction) \quad \frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x{:}A.\ B) : s}{\Gamma \vdash (\lambda x{:}A.\ b) : (\Pi x{:}A.\ B)}$$

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

$$(\beta - conversion) \quad \frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash b : B'}$$

Here s ranges over sorts, i.e. $s \in \mathsf{S}$, and a variable x is called Γ-*fresh* if $x \notin \{y\} \cup \mathrm{FV}(B)$ for all $y{:}B$ in Γ.

Definition 4. A specification $S = (\mathsf{S}, \mathsf{A}, \mathsf{R})$ is *functional* (or singly sorted) if

1. $(s : s'), (s : s'') \in \mathsf{A} \Rightarrow s' \equiv s''$
2. $(s_1, s_2, s_3), (s_1, s_2, s_3') \in \mathsf{R} \Rightarrow s_3 \equiv s_3'$

In functional PTS's, types are unique up to β-equality:

Theorem 5 Uniqueness of Types. *Let $\lambda\mathcal{S}$ be a functional* PTS.
If $\Gamma \vdash_{\lambda S} a : A$ and $\Gamma \vdash_{\lambda S} a : B$ then $A =_\beta B$.

Definition 6. The Calculus of Constructions (λC) is the PTS specified by

$$\mathsf{S} = \{*, \Box\} \quad \mathsf{A} = \{* : \Box\} \quad \mathsf{R} = \{(*,*), (\Box,*), (*,\Box), (\Box,\Box)\}$$

The Calculus of Constructions extended with an infinite type hierarchy (λC^∞) is the PTS specified by

$$\begin{array}{l} \mathsf{S} = \mathbb{N} \\ \mathsf{A} = \{(n : n+1) \mid n \in \mathbb{N}\} \\ \mathsf{R} = \{(m,0,0) \mid m \in \mathbb{N}\} \cup \{(m,n,max(m,n)) \mid m,n \in \mathbb{N}\} \end{array}$$

We can see that λC^∞ extends λC by reading $*$ for 0 and $\Box$ for 1.

By extending λC^∞ with cumulativity (a subtype relation on the sorts) we obtain the system defined in section 11 of [Coq86]. By extending it with cumulativity and strong Σ-types, we obtain the system ECC introduced in [Luo89].

Theorem 7. *λC^∞ and (hence λC) are strongly normalising, i.e. b and B are β-strongly normalising for all $\Gamma \vdash b : B$.*

Proof. λC^∞ is a subsystem of ECC and ECC is SN_β (see [Luo89]).

3 Pure Type Systems with Definitions

The DPTS determined by the specification $\mathcal{S}$ is denoted by $\lambda\mathcal{S}_\delta$. It consists of sets of pseudoterms and pseudocontexts, reduction relations and a typing relation.

The set of pseudoterms T and the set of contexts C defined in section 2 are extended to include definitions:

Definition 8. The set T_δ of *pseudoterms* of the DPTS $\lambda(\mathsf{S},\mathsf{A},\mathsf{R})$ is given by

$$\mathsf{T}_\delta ::= \mathsf{V} \mid \mathsf{S} \mid (\mathsf{T}_\delta\ \mathsf{T}_\delta) \mid (\lambda \mathsf{V}{:}\mathsf{T}_\delta.\ \mathsf{T}_\delta) \mid (\Pi \mathsf{V}{:}\mathsf{T}_\delta.\ \mathsf{T}_\delta) \mid (\mathsf{V}{=}\mathsf{T}_\delta : \mathsf{T}_\delta \text{ in } \mathsf{T}_\delta)$$

Like λ- and Π-abstractions, definitions introduce bound variables. In $(x{=}a{:}A \text{ in } b)$ the free occurrences of x in b are bound (but not those in a and A). The notions of free variables, substitution, α-equality, and β-reduction are extended to T_δ in the natural way.

Definition 9. The set C_δ of *pseudocontexts* of the DPTS $\lambda(\mathsf{S},\mathsf{A},\mathsf{R})$ is defined as follows

- $\epsilon \in \mathsf{C}_\delta$
- $\langle \Gamma, x{:}A\rangle \in \mathsf{C}_\delta$ if $\Gamma \in \mathsf{C}_\delta$, $x \in \mathsf{V}$,$A \in \mathsf{T}_\delta$ and x is Γ-fresh
- $\langle \Gamma, x{=}a{:}A\rangle \in \mathsf{C}_\delta$ if $\Gamma \in \mathsf{C}_\delta$, $x \in \mathsf{V}$, $a, A \in \mathsf{T}_\delta$, x is Γ-fresh and $x \notin \mathrm{FV}(a) \cup \mathrm{FV}(A)$.

Here a variable x is called *Γ-fresh* if $x \notin \{y\} \cup \mathrm{FV}(B)$ for all $y{:}B$ in Γ and $x \notin \{y\} \cup \mathrm{FV}(b) \cup \mathrm{FV}(B)$ for all $y{=}b{:}B$ in Γ.

Note that the set of contexts is not simply given by $\mathsf{C}_\delta ::= \epsilon \mid \mathsf{C}_\delta, \mathsf{V} : \mathsf{T}_\delta \mid \mathsf{C}_\delta, \mathsf{V}{=}\mathsf{T}_\delta : \mathsf{T}_\delta$, but that we require that all variables introduced in a context are fresh. This is necessary to avoid problems with the capture of free variables in the definition of δ-reduction below.

Definition 10. A term b *δ-reduces to b' in the context Γ* – written $\Gamma \vdash b \to_\delta b'$ – if it can be derived using the following rules:

$$\Gamma_1, x{=}a{:}A, \Gamma_2 \vdash x \to_\delta a$$

$$\Gamma \vdash (x{=}a{:}A \text{ in } b) \to_\delta b \qquad \text{if } x \notin \mathrm{FV}(b)$$

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

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

$$\frac{\Gamma \vdash a \to_\delta a'}{\Gamma \vdash (a\ b) \to_\delta (a'\ b)} \qquad \frac{\Gamma \vdash b \to_\delta b'}{\Gamma \vdash (a\ b) \to_\delta (a\ b')}$$

$$\frac{\Gamma, x{:}A \vdash a \to_\delta a'}{\Gamma \vdash (\lambda x{:}A.\ a) \to_\delta (\lambda x{:}A.\ a')} \qquad \frac{\Gamma \vdash A \to_\delta A'}{\Gamma \vdash (\lambda x{:}A.\ a) \to_\delta (\lambda x{:}A'.\ a)}$$

$$\frac{\Gamma, x{:}A \vdash a \to_\delta a'}{\Gamma \vdash (\Pi x{:}A.\ a) \to_\delta (\Pi x{:}A.\ a')} \qquad \frac{\Gamma \vdash A \to_\delta A'}{\Gamma \vdash (\Pi x{:}A.\ a) \to_\delta (\Pi x{:}A'.\ a)}$$

The first rule allows the unfolding of definitions, the second rule allows the removal of definitions, and the rest are compatibility rules.

We write $\Gamma \vdash a \to_{\beta\delta} a'$ if $\Gamma \vdash a \to_\delta a'$ or $a \to_\beta a'$. For $\rho \in \{\beta, \delta, \beta\delta\}$, the relation $\Gamma \vdash \_ \twoheadrightarrow_\rho \_$ is the transitive, reflexive closure of $\Gamma \vdash \_ \to_\rho \_$, and $\Gamma \vdash \_ =_\rho \_$ is the congruence relation generated by $\Gamma \vdash \_ \twoheadrightarrow_\rho \_$.

Example 1.

$$\begin{array}{ll}
\epsilon \vdash & (id{=}(\lambda y{:}A.\ y) : A \to A \text{ in } (\lambda x{:}A \to A.\quad id\quad)\quad id) \\
\to_\delta & (id{=}(\lambda y{:}A.\ y) : A \to A \text{ in } (\lambda x{:}A \to A.\ (\lambda y{:}A.\ y))\quad id) \\
\to_\delta & (id{=}(\lambda y{:}A.\ y) : A \to A \text{ in } (\lambda x{:}A \to A.\ (\lambda y{:}A.\ y))\ (\lambda y{:}A.\ y)) \\
\to_\delta & \qquad\qquad\qquad\qquad\quad (\lambda x{:}A \to A.\ (\lambda y{:}A.\ y))\ (\lambda y{:}A.\ y))
\end{array}$$

In the first two steps one occurrence of id is unfolded, and in the last step the definition of id is removed.

Definition 11. A term b *has type B in context Γ* in the DPTS $\lambda\mathcal{S}_\delta = \lambda(\mathsf{S}, \mathsf{A}, \mathsf{R})_\delta$ – written $\Gamma \vdash_{\lambda\mathcal{S}_\delta} b : B$ – if it can be derived using the following rules (the subscript $\lambda\mathcal{S}_\delta$ of $\vdash$ is dropped if it is clear which DPTS we mean):

$$(\delta - start) \qquad \frac{\Gamma \vdash a : A}{\Gamma, x{=}a{:}A \vdash x : A} \qquad \text{where } x \text{ is } \Gamma\text{-fresh}$$

$$(\delta - weakening) \qquad \frac{\Gamma \vdash b : B \quad \Gamma \vdash a : A}{\Gamma, x{=}a{:}A \vdash b : B} \qquad \text{where } x \text{ is } \Gamma\text{-fresh}$$

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

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

$$(\delta - conversion) \quad \frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad \Gamma \vdash B =_\delta B'}{\Gamma \vdash b : B'}$$

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

Clearly, a DPTS $\lambda\mathcal{S}_\delta$ is an extension of the PTS $\lambda\mathcal{S}$.

One might consider dropping the premiss "$\Gamma \vdash (x{=}a{:}A \text{ in } B) : s$" in the δ-introduction rule, (and then possibly also omitting the δ-formation rule). However, this results in a badly-behaved type system for which subject reduction fails, as the following example illustrates.

Example 2. Consider the system λC_δ with the rule $(\delta - introduction)$ replaced by the more powerful rule

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

Suppose that $\Gamma \vdash a : A$. Then we can derive $\Gamma \vdash (x{=}a{:}A \text{ in } *) : (x{=}a{:}A \text{ in } \Box)$, but we cannot derive (i) $\Gamma \vdash * : (x{=}a{:}A \text{ in } \Box)$. Since $(x{=}a{:}A \text{ in } *) \to_\delta *$, this means subject reduction fails. The δ-conversion rule cannot be used to derive (i) from $\Gamma \vdash * : \Box$, e.g.

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

because – like $\Box$ – $(x{=}a{:}A \text{ in } \Box)$ does not have any type s.

Remark. **(Definitions vs abstraction and application).**
There are important differences between the terms $(x{=}a : A \text{ in } b)$ and $(\lambda x{:}A.\ b)\ a$, both regarding their typing and their reduction behaviour.

Both terms can be reduced to $b[x := a]$. But the term $(\lambda x{:}A.\ b)\ a$ β-reduces in one step to to $b[x := a]$, whereas $(x{=}a : A \text{ in } b)$ δ-reduces in one step to an expression which has only *one* occurrence of x in b replaced by a. The δ-reduction of $(x{=}a{:}A \text{ in } b)$ to $b[x := a]$ will take several steps.

There are two reasons why $(x{=}a{:}A \text{ in } b)$ may be typable when $(\lambda x{:}A.\ b)\ a$ is not typable:

1. in $(x{=}a{:}A \text{ in } b)$ the fact that x is an abbreviation for a can be used to type b.

   Example 3. It is possible to type the term $(X{=}\alpha \to \alpha{:}* \text{ in } (\lambda y{:}\alpha.\ \lambda f{:}X.\ fy))$. But it is not possible to type the term obtained by replacing the definition with an abstraction and an application: $(\lambda X{:}*.\ \lambda y{:}\alpha.\ \lambda f{:}X.\ fy)\ (\alpha \to \alpha)$. In this term the application fy is not well-typed, because the type α of the argument y does not match the type X of the function f. In the first term this application is well-typed, because we know that X is an abbreviation of $\alpha \to \alpha$.

2. the abstraction $(\lambda x{:}A.\ b)$ may not be allowed in a given type system.

   Example 4. The term $(X{=}\alpha \to \alpha : * \text{ in } (\lambda y{:}X.\ \lambda f{:}X \to X.\ fy))$ is typable in the simply typed lambda calculus $(\lambda_\to)$ extended with definitions. The corresponding term expressed with an application and an abstraction, i.e.
   $(\lambda X{:}*.\ \lambda y{:}X.\ \lambda f{:}X \to X.\ fy)\ (\alpha \to \alpha)$, is not typable in $\lambda_\to$, because in $\lambda_\to$ abstractions over type variables are not allowed.

The fact that the type A of a is recorded in the definition $x = a : A$ is not essential. One can also choose not to record the types of definitions, i.e. have terms and contexts of the form $(x{=}a \text{ in } b)$ and $\Gamma, x{=}a, \Gamma'$. For functional PTS's it makes no difference whether definitions include type information or not, because types are unique (up to β-equality). There are however instances where one may want to record the type of a definition, for example if a is a proof of some proposition A. Also, in any implementation types will have to be recorded for efficient type-checking.

4 Properties of δ-reduction

All proofs of results in this section and the next are given in detail in [SP93]. Many of these proofs are straightforward induction proofs, but they are too long for all of them to be included here.

Definition 12. For $a \in \mathsf{T}_\delta$ and $\Gamma \in \mathsf{C}_\delta$ we define $|a|_\Gamma \in \mathsf{T}$ by induction on the number of symbols occurring in Γ and a, as follows:

$$
\begin{array}{ll}
|x|_\Gamma & = \begin{cases} |a|_{\Gamma_1} & \text{if } \Gamma \equiv \langle \Gamma_1, x{=}a{:}A, \Gamma_2 \rangle \\ x & \text{otherwise} \end{cases} \\
|s|_\Gamma & = s \quad \text{if } s \in \mathsf{S} \\
|a\ b|_\Gamma & = |a|_\Gamma |b|_\Gamma \\
|\lambda x{:}A.\ b|_\Gamma & = (\lambda x{:}|A|_\Gamma.\ |b|_{\Gamma, x:A}) \\
|\Pi x{:}A.\ B|_\Gamma & = (\Pi x{:}|A|_\Gamma.\ |B|_{\Gamma, x:A}) \\
|x{=}a{:}A \text{ in } b|_\Gamma & = |b|_{\Gamma, x=a:A}
\end{array}
$$

So the value $|a|_\Gamma$ is obtained from a by unfolding all the definitions occurring in Γ and in a. The mapping $|\_|_\_$ is used to prove Church-Rosser for δ- and $\beta\delta$-reduction, using the following basic properties, which we give without proof:

Property 13. *1.* $\Gamma \vdash a \twoheadrightarrow_\delta |a|_\Gamma$.
2. if $\Gamma \vdash a \to_\delta a'$ *then* $|a|_\Gamma \equiv |a'|_\Gamma$.
3. if $a \to_\beta a'$ *then* $|a|_\Gamma \twoheadrightarrow_\beta |a'|_\Gamma$.

Theorem 14. *δ-reduction is Church-Rosser.*

Proof. Suppose $\Gamma \vdash a \twoheadrightarrow_\delta a_1$ and $\Gamma \vdash a \twoheadrightarrow_\delta a_2$. By 13.1, $\Gamma \vdash a_1 \twoheadrightarrow_\delta |a_1|_\Gamma$ and $\Gamma \vdash a_2 \twoheadrightarrow_\delta |a_2|_\Gamma$, and by 13.2, $|a|_\Gamma \equiv |a_1|_\Gamma \equiv |a_2|_\Gamma$, so $|a|_\Gamma$ is a common δ-reduct of a_1 and a_2.

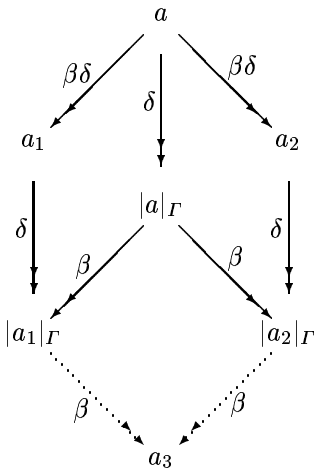
As you probably expected:

Theorem 15. *$|a|_\Gamma$ is the δ-normal form of a in Γ.*

Proof. Given 13.1 and Church-Rosser for δ-reduction, it only remains to be shown that $|a|_\Gamma$ is in δ-normal form in Γ. This can be proved by induction on the number of symbols in Γ and a.

Theorem 16. *$\beta\delta$-reduction is Church-Rosser.*

Proof. Suppose $\Gamma \vdash a \twoheadrightarrow_{\beta\delta} a_1$ and $\Gamma \vdash a \twoheadrightarrow_{\beta\delta} a_2$. By 13.1, $\Gamma \vdash a \to_\delta |a|_\Gamma$, $\Gamma \vdash a_1 \to_\delta |a_1|_\Gamma$, and $\Gamma \vdash a_2 \to_\delta |a_2|_\Gamma$. By 13.2 and 13.3, $\Gamma \vdash |a|_\Gamma \twoheadrightarrow_\beta |a_1|_\Gamma$ and $\Gamma \vdash |a|_\Gamma \twoheadrightarrow_\beta |a_2|_\Gamma$. Now the solid lines of the diagram below are justified:



Then by Church-Rosser for β-reduction there is a common β-reduct a_3 of $|a_1|_\Gamma$ and $|a_2|_\Gamma$, and this is a common $\beta\delta$-reduct of a_1 and a_2.

From the fact that δ-normal forms exist (theorem 15), it follows that δ-reduction is weakly normalising. In fact, δ-reduction is strongly normalising. To prove this, we define δ-reduction for contexts:

Definition 17. For $\rho \in \{\delta, \beta\delta\}$, a context Γ ρ-reduces to Γ' – written $\Gamma \to_\rho \Gamma'$ – if it can be derived using the following rules:

$$\frac{\Gamma \vdash A \to_\rho A'}{\Gamma, y{:}A, \Gamma' \to_\rho \Gamma, y{:}A', \Gamma'} \quad \frac{\Gamma \vdash A \to_\rho A'}{\Gamma, y{=}a{:}A, \Gamma' \to_\rho \Gamma, y{=}a{:}A', \Gamma'} \quad \frac{\Gamma \vdash a \to_\rho a'}{\Gamma, y{=}a{:}A, \Gamma' \to_\rho \Gamma, y{=}a'{:}A, \Gamma'}$$

Theorem 18. SN_δ *δ-reduction is strongly-normalising, i.e. for all $\Gamma \in \mathsf{C}_\delta$ and $b \in \mathsf{T}_\delta$ the pseudoterm b is δ-strongly normalising in Γ.*

Proof. This is proved by showing that the function $nat\_(\_) : \mathsf{C}_\delta \times \mathsf{T}_\delta \to \mathbb{N}$ defined below decreases with δ-reduction.

For $a \in \mathsf{T}_\delta$ and $\Gamma \in \mathsf{C}_\delta$ we define $nat_\Gamma(a) \in \mathbb{N}$, by induction on the number of symbols in Γ and a, as follows:

$$\begin{aligned}
nat_\Gamma(x) &= \begin{cases} nat_{\Gamma_1}(a) + 1 & \text{if } \Gamma \equiv \langle \Gamma_1, x{=}a{:}A, \Gamma_2 \rangle \\ 0 & \text{otherwise} \end{cases} \\
nat_\Gamma(s) &= 0 \quad \text{if } s \in \mathsf{S} \\
nat_\Gamma(x{=}a{:}A \text{ in } b) &= nat_\Gamma(a) + nat_\Gamma(A) + nat_{\Gamma, x=a:A}(b) + 1 \\
nat_\Gamma(a\ b) &= nat_\Gamma(a) + nat_\Gamma(b) \\
nat_\Gamma(\lambda x{:}A.\ b) &= nat_{\Gamma, x:A}(b) + nat_\Gamma(A) \\
nat_\Gamma(\Pi x{:}A.\ B) &= nat_{\Gamma, x:A}(B) + nat_\Gamma(A)
\end{aligned}$$

The following two properties of $nat\_(\_)$ can proved simultaneously by induction on number of symbols in Γ and in a:

(i) if $\Gamma \vdash a \to_\delta a'$ then $nat_\Gamma(a) > nat_\Gamma(a')$

(ii) if $\Gamma \to_\delta \Gamma'$ then $nat_\Gamma(a) \geq nat_{\Gamma'}(a)$

SN_δ follows immediately from (i).

In [vD80] a proof of strong normalisation for the theory of abbreviations LSP, due to N.G. de Bruijn, is given, which corresponds to SN_δ. Our proof is somewhat simpler. By having the context as a parameter, we can give a simpler definition of the measure $nat\_(\_)$.

Strong normalisation for δ implies finiteness of developments for β-reduction (think of a δ-redex ($x{=}a{:}A$ in b) as a 'marked' β-redex $(\lambda x{:}A.\ b)\ a$). In fact, the simple proof of finiteness of developments given in [dV85] was inspired by the proof in [vD80].

5 Properties of DPTS's

To establish a relation between the PTS $\lambda\mathcal{S}$ and the DPTS $\lambda\mathcal{S}_\delta$, the mapping $|\_|\_$ is extended to contexts:

Definition 19. The mapping $|\_| : \mathsf{C}_\delta \to \mathsf{C}$ is defined as follows:

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

The following theorem states that definitions can be eliminated:

Theorem 20. *Elimination of Definitions If $\Gamma \vdash_{\lambda\mathcal{S}_\delta} a : A$ then $|\Gamma| \vdash_{\lambda\mathcal{S}} |a|_\Gamma : |A|_\Gamma$.*

Proof. Induction on the derivation of $\Gamma \vdash a : A$.

It follows from this theorem that extending a PTS with definitions does not increase the strength of the system.

Corollary 21. (Conservativity) *Let $a \in \mathsf{T}$ and $\Gamma \in \mathsf{C}$. Then*

1. $\exists_A\ \Gamma \vdash_{\lambda\mathcal{S}} a : A$ *iff* $\exists_A\ \Gamma \vdash_{\lambda\mathcal{S}_\delta} a : A$
2. $\exists_A\ \Gamma \vdash_{\lambda\mathcal{S}} A : a$ *iff* $\exists_A\ \Gamma \vdash_{\lambda\mathcal{S}_\delta} A : a$

The first part of this corollary states that a $\lambda\mathcal{S}$-term is typable in $\lambda\mathcal{S}$ iff it is typable in $\lambda\mathcal{S}_\delta$. The second part states that a $\lambda\mathcal{S}$-type is inhabited in $\lambda\mathcal{S}$ iff it is inhabited in $\lambda\mathcal{S}_\delta$. The second part is crucial if types are interpreted as propositions and terms as proofs (the Curry-Howard-de Bruijn isomorphism), because it means that a $\lambda\mathcal{S}$-proposition is provable in $\lambda\mathcal{S}$ iff it is provable in $\lambda\mathcal{S}_\delta$.

All the properties proved in [Bar92] for arbitrary PTS's can easily be proved for DPTS's. For example, we have

Theorem 22. (Subject Reduction) *If $\Gamma \vdash a : A$ and $\Gamma \vdash a \to_{\beta\delta} a'$ then $\Gamma \vdash a' : A$.*

Proof. This can be proved in the same way as subject reduction for PTS's. The following properties are proved simultaneously by induction on the derivation of $\Gamma \vdash a : A$

(i) if $\Gamma \vdash a \to_{\beta\delta} a'$ and $\Gamma \vdash a : A$ then $\Gamma \vdash a' : A$

(ii) if $\Gamma \to_{\beta\delta} \Gamma'$ and $\Gamma \vdash a : A$ then $\Gamma' \vdash a : A$

using a substitution lemma.

Theorem 20 can also be used to prove that for functional systems we preserve the property that types are unique up to conversion:

Theorem 23 Uniqueness of Types. *Let $\lambda\mathcal{S}_\delta$ be a functional* DPTS.
If $\Gamma \vdash_{\lambda\mathcal{S}_\delta} a : A$ and $\Gamma \vdash_{\lambda\mathcal{S}_\delta} a : B$ then $\Gamma \vdash A =_{\beta\delta} B$.

Proof. Suppose $\Gamma \vdash_{\lambda\mathcal{S}_\delta} a : A$ and $\Gamma \vdash_{\lambda\mathcal{S}_\delta} a : B$. By theorem 20 $|\Gamma| \vdash_{\lambda\mathcal{S}} |a|_\Gamma : |A|_\Gamma$ and $|\Gamma| \vdash_{\lambda\mathcal{S}} |a|_\Gamma : |B|_\Gamma$. $\lambda\mathcal{S}$ is functional, so by Uniqueness of Types for the PTS $\lambda\mathcal{S}$ $|A|_\Gamma =_\beta |B|_\Gamma$. It follows from 13.1 that $\Gamma \vdash A \twoheadrightarrow_\delta |A|_\Gamma$ and that $\Gamma \vdash B \twoheadrightarrow_\delta |B|_\Gamma$. Hence $\Gamma \vdash A =_{\beta\delta} B$.

6 Strong Normalisation

We now come to the problem of strong normalisation of $\beta\delta$-reduction for DPTS's.

Definition 24. Let $\lambda\mathcal{S}_{(\delta)}$ be a (D)PTS and $\rho \in \{\beta, \beta\delta\}$.

$\lambda\mathcal{S}_{(\delta)}$ is ρ-strongly normalising – written SN_ρ – if b and B are ρ-strongly normalising in Γ for all $\Gamma \vdash_{\lambda\mathcal{S}_{(\delta)}} b : B$.

Similarly, $\lambda\mathcal{S}_{(\delta)}$ is ρ-weakly normalising – WN_ρ – if b and B are ρ-weakly normalising in Γ for all $\Gamma \vdash_{\lambda\mathcal{S}_{(\delta)}} b : B$.

Using theorem 20 it is easy to prove that extending a PTS with definitions preserves *weak* normalisation:

Theorem 25. *$\lambda\mathcal{S}$ is WN_β $\Rightarrow$ $\lambda\mathcal{S}_\delta$ is $\mathsf{WN}_{\beta\delta}$.*

Proof. Suppose $\lambda\mathcal{S}$ is WN_β and $\Gamma \vdash_{\lambda\mathcal{S}_\delta} a : A$. By theorem 20 $|\Gamma| \vdash_{\lambda\mathcal{S}} |a|_\Gamma : |A|_\Gamma$, so by WN_β for $\lambda\mathcal{S}$ the term $|a|_\Gamma$ has a β-normal form b. Then $\Gamma \vdash a \twoheadrightarrow_\delta |a|_\Gamma \twoheadrightarrow_\beta b$, and it is not difficult to show that the β-normal form b of the δ-normal form $|a|_\Gamma$ is also a δ-normal form.

Using theorem 20 it is also easy to show that strong normalisation is preserved for DPTS's *without local definitions*:

Theorem 26. *Let $\lambda\mathcal{S}_\delta^-$ be $\lambda\mathcal{S}$ extended with only global definitions, i.e. the system $\lambda\mathcal{S}_\delta$ without the rules δ-form and δ-intro. (In this system terms cannot be of the form $(x{=}a{:}A \text{ in } b)$, i.e. all terms are in T and not in $\mathsf{T}_\delta \setminus \mathsf{T}$.) Then*

$$\lambda\mathcal{S} \text{ is } \mathsf{SN}_\beta \Rightarrow \lambda\mathcal{S}_\delta^- \text{ is } \mathsf{SN}_{\beta\delta}\ .$$

Proof. Given theorem 20, we only have to show that $|\_|_\_$ maps an infinite $\beta\delta$-sequence to an infinite β-sequence. Any infinite $\beta\delta$-sequence contains an infinite number of β-steps (because of SN_δ), so this follows from property 13.2 and

$$a \to_\beta b \;\Rightarrow\; |a|_\Gamma \to_\beta |b|_\Gamma \quad \text{for all } a, b \in \mathsf{T}.$$

Note that this does not hold for all $a, b \in \mathsf{T}_\delta$!
For example, if $(x{=}a{:}A \text{ in } y) \to_\beta (x{=}a'{:}A \text{ in } y)$ then $|x{=}a{:}A \text{ in } y|_\Gamma = |x{=}a'{:}A \text{ in } y|_\Gamma = |y|_\Gamma$.

The system implemented in Coq [Dea91] does not have local definitions, so for this system strong normalisation follows from strong normalisation of the system without definitions.

To prove strong normalisation for $\beta\delta$-reduction for DPTS's we introduce a mapping $\{\_\}_\_ : \mathsf{T}_\delta \times \mathsf{C}_\delta \to \mathsf{T}$, which maps an infinite $\beta\delta$-reduction sequence in a DPTS to an infinite β-reduction sequence in a slightly "larger" PTS, i.e. a PTS with more sorts, axioms and rules.

Definition 27. The mapping $\{\_\}_\_ : \mathsf{T}_\delta \times \mathsf{C}_\delta \to \mathsf{T}$ is defined as follows:

$$\begin{array}{ll}
\{x\}_\Gamma & = \begin{cases} \{a\}_{\Gamma_1} & \text{if } \Gamma \equiv \langle \Gamma_1, x{=}a{:}A, \Gamma_2 \rangle \\ x & \text{otherwise} \end{cases} \\
\{s\}_\Gamma & = s \quad \text{if } s \in \mathsf{S} \\
\{a\ b\}_\Gamma & = \{a\}_\Gamma \{b\}_\Gamma \\
\{\lambda x{:}A.\ b\}_\Gamma & = \lambda x{:}\{A\}_\Gamma.\ \{b\}_{\Gamma, x:A} \\
\{\Pi x{:}A.\ B\}_\Gamma & = \Pi x{:}\{A\}_\Gamma.\ \{B\}_{\Gamma, x:A} \\
\{x{=}a{:}A \text{ in } b\}_\Gamma & = (\lambda x{:}\{A\}_\Gamma.\ \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma
\end{array}$$

Note that $|\_|_\_$ only differs from $\{\_\}_\_$ in the value given for $(x{=}a{:}A \text{ in } b)$. Like $|\_|_\_$ the mapping $\{\_\}_\_$ unfolds all definitions, but, unlike $|\_|_\_$, the mapping $\{\_\}_\_$ does not remove local definitions. Instead, every local definition is translated to a β-redex, a λ-abstraction with an argument. The redex $(\lambda x{:}\{A\}_\Gamma.\ \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma$ is a special kind of β-redex, called K-redex, because $x \notin \mathrm{FV}(\{b\}_{\Gamma, x=a:A})$.

Example 5. Let $B \equiv (X{=}\alpha \to \alpha{:}* \text{ in } (\lambda y{:}\alpha.\ \lambda f{:}X.\ fy))$. Then $\{B\}_\epsilon$ is

$$(\lambda X{:}*.\ \lambda y{:}\alpha.\ \lambda f{:}\alpha \to \alpha.\ fy)\ (\alpha \to \alpha)$$

Compare this with example 3, where it was shown that replacing the local definition in the term B with an abstraction and application produces an untypable term

$$(\lambda X{:}*.\ \lambda y{:}\alpha.\ \lambda f{:}X.\ fy)\ (\alpha \to \alpha)$$

because in this term the application fy is not well-typed. In the term $\{B\}_\epsilon$ the application fy is well-typed, because all occurrences of X have been unfolded.

Definition 28. The mapping $\{\_\} : \mathsf{C}_\delta \to \mathsf{C}$ is defined by

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

$\{\_\}_\_$ maps $\beta\delta$-reduction sequences to β-reduction sequences:

Lemma 29. *1. If $a \to_\beta b$ then $\{a\}_\Gamma \to_\beta^+ \{b\}_\Gamma$, where $\to_\beta^+$ is the transitive closure of $\to_\beta$.*
2. If $\Gamma \vdash a \to_\delta b$ then $\{a\}_\Gamma \to_\beta^{\equiv} \{b\}_\Gamma$, where $\to_\beta^{\equiv}$ is the reflexive closure of $\to_\beta$.

Proof. Induction on the structure of a.

By this lemma $\{\_\}_\_$ maps a $\beta\delta$-reduction sequence with an infinite number of β-steps to an infinite β-reduction sequence.

It will be proved that $\{\_\}_\_$ maps terms that are typable in a DPTS $\lambda\mathcal{S}_\delta$ to terms that are typable in a PTS $\lambda\mathcal{S}'$ with $\mathcal{S}'$ a *completion* of $\mathcal{S}$:

Definition 30. A specification $\mathcal{S}' = (\mathsf{S}', \mathsf{A}', \mathsf{R}')$ is called a *completion* of a $\mathcal{S} = (\mathsf{S}, \mathsf{A}, \mathsf{R})$ if

(C1) $\mathsf{S} \subseteq \mathsf{S}'$, $\mathsf{A} \subseteq \mathsf{A}'$, $\mathsf{R} \subseteq \mathsf{R}'$, and
(C2) $\mathcal{S}'$ is full, i.e. $\forall s_1, s_2 \in \mathsf{S}'\ \exists s_3 \in \mathsf{S}'\ (s_1, s_2, s_3) \in \mathsf{R}'$, and
(C3) for all $s \in \mathsf{S}$ there is an $s' \in \mathsf{S}'$ such that $(s : s') \in \mathsf{A}'$ (so the sorts of $\mathcal{S}$ are typable in $\lambda\mathcal{S}'$).

Example 6. The system λC^∞ is a completion of λC and of itself.

Remember that $\{\_\}_\_$ translates a local definition to a λ-abstraction with an argument: $\{x{=}a{:}A \text{ in } b\}_\Gamma \equiv (\lambda x{:}\{A\}_\Gamma.\ \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma$. Condition C2 is needed to ensure that all these λ-abstractions introduced by $\{\_\}_\_$ are allowed in $\lambda\mathcal{S}'$. Condition C3 is needed because for example the term $(x{=}* : \Box \text{ in } x)$ is typable in λC_δ but $\{x{=}* : \Box \text{ in } x\}_\epsilon \equiv (\lambda x{:}\Box.\ *)*$ is not typable in λC.

Lemma 31. *Let $\mathcal{S}' = (\mathsf{S}', \mathsf{A}', \mathsf{R}')$ be a completion of $\mathcal{S} = (\mathsf{S}, \mathsf{A}, \mathsf{R})$.*
If $\Gamma \vdash_{\lambda\mathcal{S}\delta} a{:}A$ and $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{a\}_\Gamma : \{A\}_\Gamma$, then $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{A\}_\Gamma : s$.

Proof. Assume (i) $\Gamma \vdash_{\lambda\mathcal{S}\delta} a{:}A$ and (ii) $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{a\}_\Gamma : \{A\}_\Gamma$. By correctness of types [Bar92] it follows from (ii) that $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{A\}_\Gamma : s'$ or $\{A\}_\Gamma \equiv s$.

Suppose $\{A\}_\Gamma \equiv s$. Then $\{A\}_\Gamma \in \mathsf{S}$, so by C3 it is typable in $\lambda\mathcal{S}'$: there is a sort $s' \in \mathsf{S}'$ such that $(\{A\}_\Gamma : s') \in \mathsf{A}'$ and hence $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{A\}_\Gamma : s'$.

Lemma 32. *Let $\mathcal{S}' = (\mathsf{S}', \mathsf{A}', \mathsf{R}')$ be a completion of $\mathcal{S} = (\mathsf{S}, \mathsf{A}, \mathsf{R})$.*
If $\Gamma \vdash_{\lambda\mathcal{S}\delta} a : A$ then $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{a\}_\Gamma : \{A\}_\Gamma$.

Proof. By induction on the derivation of $\Gamma \vdash_{\lambda\mathcal{S}\delta} a{:}A$. We show the case that the last step is the δ-introduction rule, which is the most complicated one.

Suppose the last step in the derivation is $\dfrac{\Gamma, x{=}a{:}A \vdash_{\lambda\mathcal{S}\delta} b : B \quad \Gamma \vdash_{\lambda\mathcal{S}\delta} (x{=}a{:}A \text{ in } B) : s}{\Gamma \vdash_{\lambda\mathcal{S}\delta} (x{=}a{:}A \text{ in } b) : (x{=}a{:}A \text{ in } B)}$

By the IH

$$\{\Gamma, x{=}a{:}A\} \vdash_{\lambda\mathcal{S}'} \{b\}_{\Gamma, x=a:A} : \{B\}_{\Gamma, x=a:A} \tag{i}$$
$$\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{x{=}a{:}A \text{ in } B\}_\Gamma : s \tag{ii}$$

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

$$\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{a\}_\Gamma : \{A\}_\Gamma \tag{iii}$$

By lemma 31 it follows from (i) and (iii) that there are $s_1, s_2 \in \mathsf{S}'$ such that

$$\{\Gamma, x{=}a{:}A\} \vdash_{\lambda\mathcal{S}'} \{B\}_{\Gamma, x=a:A} : s_2 \tag{iv}$$
$$\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{A\}_\Gamma : s_1 \tag{v}$$

By C2 there exist an $s_3 \in \mathsf{S}'$ such that $(s_1, s_2, s_3) \in \mathsf{R}'$, so then

$$\dfrac{\dfrac{\dfrac{}{\{\Gamma\} \vdash_{\lambda\mathcal{S}'} (\Pi x \in \{A\}_\Gamma.\ \{B\}_{\Gamma, x=a:A}) : s_3}\,(\text{iv})\ (\text{v})\ (\Pi\text{-form})}{\{\Gamma\} \vdash_{\lambda\mathcal{S}'} (\lambda x \in \{A\}_\Gamma.\ \{b\}_{\Gamma, x=a:A}) : (\Pi x \in \{A\}_\Gamma.\ \{B\}_{\Gamma, x=a:A})}\,(\text{i})\ (\Pi\text{-intro})}{\{\Gamma\} \vdash_{\lambda\mathcal{S}'} (\lambda x \in \{A\}_\Gamma.\ \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma : \{B\}_{\Gamma, x=a:A}[x := \{a\}_\Gamma]}\,(\text{iii})\ (\Pi\text{-elim})$$

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

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

so using the conversion rule

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

Lemmas 29 and 32 can now be used to prove:

Theorem 33. *Let $\mathcal{S}' = (\mathsf{S}', \mathsf{A}', \mathsf{R}')$ be a completion of $\mathcal{S} = (\mathsf{S}, \mathsf{A}, \mathsf{R})$. Then*

$$\lambda\mathcal{S}' \text{ is } \mathsf{SN}_\beta \Rightarrow \lambda\mathcal{S}_\delta \text{ is } \mathsf{SN}_{\beta\delta}\ .$$

Proof. Suppose that $\lambda\mathcal{S}'$ is SN_β, and suppose towards a contradiction that $\lambda\mathcal{S}_\delta$ is not $\mathsf{SN}_{\beta\delta}$, i.e. $\Gamma \vdash_{\lambda\mathcal{S}\delta} a : A$ and there is an infinite $\beta\delta$-reduction sequence starting at a: $\Gamma \vdash a \to_{\beta\delta} a_1 \to_{\beta\delta} a_2 \ldots$.

Observe that the number of β-reductions in this sequence is infinite. Otherwise the sequence would end with an infinite δ-sequence, and because δ is strongly normalising, this cannot happen. This means it is of the form

$$\Gamma \vdash a \to_\delta^* a_{n_1} \to_\beta a_{n_2} \to_\delta^* a_{n_3} \to_\beta a_{n_4} \to_\delta^* a_{n_5} \to_\beta a_{n_6} \to_\delta^* \ldots$$

By lemma 29 there is an infinite β-reduction sequence starting at $\{a\}_\Gamma$:

$$\{a\}_\Gamma \to_\beta^* \{a_{n_1}\}_\Gamma \to_\beta^+ \{a_{n_2}\}_\Gamma \to_\beta^* \{a_{n_3}\}_\Gamma \to_\beta^+ \{a_{n_4}\}_\Gamma \to_\beta^* \{a_{n_5}\}_\Gamma \to_\beta^+ \{a_{n_6}\}_\Gamma \to_\beta^* \ldots$$

and by theorem 32 $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{a\}_\Gamma : \{A\}_\Gamma$, which contradicts the assumption that $\lambda\mathcal{S}'$ is SN_β.

This theorem can be used to prove strong normalisation of the Calculus of Constructions (with an infinite hierarchy of universes) extended with definitions:

Corollary 34. *λC_δ and λC_δ^∞ are $\mathsf{SN}_{\beta\delta}$.*

Proof. The system λC^∞ is a completion of λC and of itself, so by theorem 33 this corollary follows immediately from the fact that λC^∞ is SN_β (theorem 7).

All systems in Barendregt's lambda cube are subsystems of the Calculus of Constructions, so extended with definitions they are all $\beta\delta$-strongly normalising.

7 Conclusions

Theorem 33 can easily be generalized to include more type constructors and reduction rules than just Π and β, or other features, for instance cumulativity. In particular, we can prove that the system ECC (see [Luo89]) extended with definitions is strongly normalising. This can be proved using the fact that ECC is SN, in the same way that in corollary 34 we prove that λC_δ^∞ is SN using the fact that λC^∞ is SN. So all the type systems implemented in Lego [LP92], which do have local definitions, are strongly normalising.

Theorem 33 is somewhat unsatisfactory. It would be nicer to prove a stronger property, namely

$$\lambda\mathcal{S} \text{ is } \mathsf{SN}_\beta \ \Rightarrow\ \lambda\mathcal{S}_\delta \text{ is } \mathsf{SN}_{\beta\delta}$$

This remains an open problem.

On the other hand, we do not know any strongly normalising PTS $\lambda\mathcal{S}$ for which theorem 33 cannot be used to prove strong normalisation of $\lambda\mathcal{S}_\delta$. In particular, λC^∞ is a completion of all strongly normalising PTS's given in [Bar92].

References

[Bar92] H.P. Barendregt. Lambda calculi with types. In D. M. Gabbai, S. Abramsky, and T. S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1. Oxford University Press, 1992.

[CH88] Thierry Coquand and Gérard Huet. The Calculus of Constructions. *Information and Computation*, 76:95–120, 1988.

[Coq85] Thierry Coquand. *Une Theorie des Constructions.* PhD thesis, Université Paris VII, 1985.

[Coq86] Thierry Coquand. An analysis of Girard's paradox. In *Logic in Computer Science*, pages 227–236. IEEE, 1986.

[dB80] N.G. de Bruijn. A survey of the project AUTOMATH. In J.P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, 1980.

[dV85] Roel de Vrijer. A direct proof of the finite developments theorem. *Journal of Symbolic Logic*, 50(2):339–343, 1985.

[Dea91] G. Dowek et al. The Coq proof assistant version 5.6, users guide. Rapport de Recherche 134, INRIA, 1991.

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, Université Paris VII, 1972.

[GN91] Herman Geuvers and Mark-Jan Nederhof. A modular proof of strong normalisation for the Calculus of Constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

[Hel91] Leen Helmink. Goal directed proof construction in type theory. In *Procs. of the first Workshop on Logical Frameworks.* Cambridge University Press, 1991.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

[LP92] Zhaohui Luo and Robert Pollack. LEGO proof development system: User's manual. Technical Report ECS-LFCS-92-211, LFCS-University of Edinburgh, 1992.

[Luo89] Z. Luo. ECC, the Extended Calculus of Constructions. In *Logic in Computer Science*, pages 386–395. IEEE, 1989.

[Rey74] John C. Reynolds. Towards a theory of type structure. In *Programming Symposium: Colloque sur la Programmation*, volume 19 of *LNCS*, pages 408–425. Springer, 1974.

[SP93] Paula Severi and Erik Poll. Pure type systems with definitions. Computing Science Note (93/24), Eindhoven University of Technology, 1993.

[vD80] D. T. van Daalen. *The Language Theory of Automath.* PhD thesis, Eindhoven University of Technology, 1980.