Pure Type Systems with definitions

Paula Severi Erik Poll
severi@win.tue.nl erik@win.tue.nl
Eindhoven University of Technology
Postbus 513 5600 MB Eindhoven

Abstract

In this paper, an extension of Pure Type Systems (PTS's) to include definitions is presented. The theorems of Church Rosser and Subject Reduction are proved for this extension. It is also proved for certain PTS's, including the Calculus of Constructions, that *Strong Normalisation* is preserved by the extension. We have proved not only this property for the Calculus of Constructions but also for all the PTS's that are known to be strongly normalizing.

Topics : Logic in Computer Science, Type Theory.

1 Introduction

A large class of typed lambda calculi can be described in a uniform way as Pure Type Systems. This includes for instance the second-order λ-calculus [?] [?], Edinburgh Logical Framework [?] and the Calculus of Constructions [?]. For an introduction to Pure Type Systems(PTS's) we refer to [?]. 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. This possiblity is essential por pratical use and implementatios of Pure Type Systems such as Coq [?], Lego [?] or Constructor [?] do provide this possibility.
In this paper, we introduce an extension of the PTS's to include definitions.
Definitions are names abbreviating expressions. They can occur in contexts and in terms. These two kinds of definitions are usually called *global* and *local* definitions. In the term $(x\!=\!a\!:\!A\ in\ b)$ the identifier x abbreviates the expression a of type A in the expression b. In the context $\Gamma, x\!=\!a\!:\!A, \Gamma'$ the identifier x abbreviates the expression a in Γ' and in any expression associated to this context. 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. $id{=}(\lambda x{:}\alpha.\ x){:}\alpha \to \alpha\ in\ (\lambda y{:}\alpha \to \alpha.\ id)\ \ id.$
The name id is an abbreviation for a larger term, i.e. $\lambda x{:}\alpha.\ x$ and is used several times in the expression $(\lambda y{:}\alpha \to \alpha.\ id)\ \ id.$

The intended meaning of a definition is that the *definiendum* x can be substituted by the *definiens* a in the expression b. In order to unfold definitions, we introduce a new relation called δ-reduction. The term $(x{=}a{:}A\ in\ b)$ δ-reduces to an expression which has at most **one occurrence** of x unfolded by a. The δ-reduction for a term b under the context $\Gamma, x{=}a{:}A, \Gamma'$ is defined in a similar way. This reduction relation is similar to the δ-reduction of the systems of the Automath family like LSP, AUT-68, AUT-QE [**?**].
The fact that the type A of a is explicitly written in $(x{=}a{:}A\ in\ b)$ 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.
We list the main properties that we prove:

- Church Rosser for $\rightarrow_\delta$ and for $\rightarrow_{\beta\delta}$.
- Strong Normalization for $\rightarrow_\delta$.
- Subject reduction theorem for all PTS's.
- Uniqueness of Types is preserved by the extension.
- Conservativity.
- Weak normalization is preserved by the extension, i.e. if a PTS is β-weakly normalizing then its extension is $\beta\delta$-strongly normalizing.
- Strong normalization for $\beta\delta$-reduction is preserved for certain PTS's, including the Calculus of Constructions.

In [**?**] an extension with definitions of the Calculus of Construction is presented. These definitions can occur only in the context and the metatheory of this extended system is not treated in detail.
Although the systems of AUTOMATH family like LSP, AUT-68 and AUT-QE [**?**] provide definitions and their metatheory has been treated in detail, they are only particular type systems.
In this paper, we extend a large class of type systems, i.e the PTS's, to include definitions. These definitions can occur either in the context or in the terms. Also we treat the metatheory of this extension in detail.
The proof of Strong Normalization for $\rightarrow_\delta$ is similar to the proof of Finiteness of Developments given in [**?**] and the one given for LSP [**?**]. We define a function that decreases with the reduction that is a little simpler than the ones presented in [**?**] and in [**?**].
For the systems of the AUTOMATH family the proof of $\beta\delta$-Strong Normalization is not complicated. If we have definitions only in the context like in [**?**], the proof of strong normalization is not complicated either.
It is difficult to prove strong normalization for the extended PTS's where definitions may occurr either in contexts or in terms. We prove that $\beta\delta$-strong normalization is preserved for those PTS's that have a completion which is β-strongly normalizing. All the PTS's that

are known to be β-strongly normalizing, including the Calculus of Constructions, have a completion that is β-strongly normalizing, and then our result applies.
All proofs of results in this paper are given in detail in [?].

2 Pure Type Systems with definitions

The concept of Pure Type System (PTS) is defined as in [?].
The set $\mathcal{T}$ of *pseudoterms* and the set $\mathcal{C}$ of *contexts* are defined as follows:

$$\begin{array}{lcl} \mathcal{T} & ::= & \mathbf{V} \mid \mathbf{C} \mid (\mathcal{T}\ \mathcal{T}) \mid (\lambda \mathbf{V}{:}\mathcal{T}.\ \mathcal{T}) \mid (\Pi \mathbf{V}{:}\mathcal{T}.\ \mathcal{T}) \\ \mathcal{C} & ::= & \epsilon \mid < \mathcal{C}, \mathbf{V}{:}\mathcal{T} > \end{array}$$

where $\mathbf{V}$ is the set of variables and $\mathbf{C}$ is the set of constants.
We define α-equality as usual and we identify α-equal terms. We write $FV(A)$ for the set containing the free variables of a term A. We denote β-reduction by $\to_\beta$ and it is defined as usual: $(\lambda x{:}A.\ b)a \to_\beta b[x := a]$. We write $A \to B$ instead of $(\Pi x{:}A.\ B)$ when $x \notin FV(B)$.
The specification of Pure Type System (PTS) is a triple $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ such that

- $\boldsymbol{S} \subseteq \mathbf{C}$ is the set of *sorts.*
- $\boldsymbol{A} \subseteq \mathbf{C} \times \boldsymbol{S}$ is the set of *axioms*
- $\boldsymbol{R} \subseteq \boldsymbol{S} \times \boldsymbol{S} \times \boldsymbol{S}$ is the set of *rules*

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

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

where s ranges over sorts, i.e. $s \in \boldsymbol{S}$.
A sort s in $\boldsymbol{S}$ is called a topsort if there is no $s_0 \in \boldsymbol{S}$ such that $\Gamma \vdash s : s_0$.
The Calculus of Constructions (λC) is the PTS specified by $\boldsymbol{S} = \{*, \Box\}$, $\boldsymbol{A} = \{(*, \Box)\}$ and $\boldsymbol{R} = \{(*, *), (\Box, *), (*, \Box), (\Box, \Box)\}$.
The Calculus of Constructions extended with an infinite type hierarchy is denoted by λC_∞ and it can be described by $\boldsymbol{S} = \mathbb{N}$, $\boldsymbol{A} = \{(n, n+1) \mid n \in \mathbb{N}\}$, $\boldsymbol{R} = \{(m, 0, 0) \mid m \in \mathbb{N}\} \cup \{(m, n, r) \mid m, n \in \mathbb{N}\ max(m, n) \leq r\}$. The system λC_∞ extended with strong Σ-types and cumulativity is the system **ECC**(see [**?**]). We can see that λC_∞ is an extension of λC writing $*$ instead of 0 and $\Box$ instead of 1.
The set of pseudoterms $\mathcal{T}$ and the set of contexts $\mathcal{C}$ defined are extended to include definitions. The set $\mathcal{T}_\delta$ of *pseudoterms* is given by

$$\mathcal{T}_\delta \quad ::= \quad \mathbf{V} \mid \mathbf{C} \mid (\mathcal{T}_\delta\ \mathcal{T}_\delta) \mid (\lambda \mathbf{V}{:}\mathcal{T}_\delta.\ \mathcal{T}_\delta) \mid (\Pi \mathbf{V}{:}\mathcal{T}_\delta.\ \mathcal{T}_\delta) \mid (\mathbf{V}{=}\mathcal{T}_\delta{:}\mathcal{T}_\delta\ in\ \mathcal{T}_\delta)$$

where $\mathbf{V}$ is the set of variables and $\mathbf{C}$ is the set of constants.
Like λ- and Π-abstractions, definitions introduce bound variables. In $(x = a : A\ in\ b)$ the variable x is bound in b. The notions of free variables, substitution, α-equality and β-reduction are extended to $\mathcal{T}_\delta$.
The set of contexts is not simply given by $\mathcal{C}_\delta ::= \epsilon \mid \mathcal{C}_\delta, \mathbf{V} : \mathcal{T}_\delta \mid \mathcal{C}_\delta, \mathbf{V} = \mathcal{T}_\delta : \mathcal{T}_\delta$. 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.
We write $Def(\Gamma)$ for the set containing the definitions occuring in the context Γ.

Definition 2.1. *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 $\Gamma \vdash a \to_\delta a'$ and is defined by the following rules:*

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

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

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

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

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

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

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

Note that the first rule allows to unfold definitions, the second rule allows to remove definitions and the remaining are compatibility rules.

Example 2.2. Assume $\Gamma \equiv A : *$.

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

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

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

Definition 2.3. *The DPTS determined by the specification $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ is denoted by $\lambda S_\delta = \lambda(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})_\delta$. It is defined by the notion of type derivation $\Gamma \vdash_{\lambda S_\delta} b : B$ with $\Gamma \in \mathcal{C}_\delta$ and $b, B \in \mathcal{T}_\delta$ given by the PTS type derivation rules and the following rules:*

$$\begin{array}{rcl}
(\delta - start) & \dfrac{\Gamma \vdash a : A}{\Gamma, x{=}a{:}A \vdash x : A} & where\ x\ is\ \Gamma\text{-}fresh \\[2ex]
(\delta - weakening) & \dfrac{\Gamma \vdash b : B \quad \Gamma \vdash a : A}{\Gamma, x{=}a{:}A \vdash b : B} & where\ x\ is\ \Gamma\text{-}fresh \\[2ex]
(\delta - formation) & \dfrac{\Gamma, x{=}a{:}A \vdash B : s}{\Gamma \vdash (x{=}a{:}A\ in\ B) : s} & \\[2ex]
(\delta - introduction) & \dfrac{\Gamma, x{=}a{:}A \vdash b : B \quad \Gamma \vdash (x{=}a{:}A\ in\ B) : s}{\Gamma \vdash (x{=}a{:}A\ in\ b) : (x{=}a{:}A\ in\ B)} & \\[2ex]
(\delta - conversion) & \dfrac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad \Gamma \vdash B =_\delta B'}{\Gamma \vdash b : B'} &
\end{array}$$

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

A DPTS λS_δ is an extension of the PTS 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 $\Gamma \vdash a : A$.
The condition "$\Gamma \vdash (x{=}a{:}A\ in\ B) : s$" 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 "$\Gamma \vdash (x{=}a{:}A\ in\ B) : s$" from the premises of the δ-introduction rule. Subject Reduction Theorem does not hold in $(\lambda S)_{\delta+}$ as the following example shows.

Example 2.4. Suppose $\Gamma \vdash a : A$. The term $(x{=}a{:}A \ in \ *)$ has type $(x{=}a{:}A \ in \ \Box)$ in $(\lambda C)_{\delta+}$. This term δ-reduces to $*$ but $*$ does not have type $(x{=}a{:}A \ in \ \Box)$. In order to derive $\Gamma \vdash * : (x{=}a{:}A \ in \ \Box)$ we apply δ-conversion rule

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

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

Note that the system $\lambda C_{\delta+}$ does not have more typable terms than λC_δ. There are only more types in $\lambda C_{\delta+}$ than in λC_δ and these are of the form $(x{=}a{:}A \ in \ \Box)$.

3 Properties of Pseudoterms

All the properties in this section, in particular Strong Normalisation for $\rightarrow_\delta$, hold for all the pseudoterms, not only for well-typed terms. The Church Rosser property is proved for $\rightarrow_\delta$ and for $\rightarrow_{\beta\delta}$.

Theorem 3.1 (Church Rosser for δ-reduction). *Let $\Gamma \in \mathcal{C}_\delta$ and $a \in \mathcal{T}_\delta$ such that $\Gamma \vdash a \twoheadrightarrow_\delta a_1$ and $\Gamma \vdash a \twoheadrightarrow_\delta a_2$. Then there exists a term a_3 such that $\Gamma \vdash a_1 \twoheadrightarrow_\delta a_3$ and $\Gamma \vdash a_2 \twoheadrightarrow_\delta a_3$.*

We give an idea of the proof of Church Rosser for δ-reduction:

a) A "projection" map, $|\_|_\_ : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow \mathcal{T}$, 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.

$$\begin{array}{lcl}
|x|_\Gamma & = & \begin{cases} x & \text{if } x \in \mathbf{V} \text{ and } x \notin Def(\Gamma) \\ |a|_{\Gamma_1} & \text{if } x \in \mathbf{V} \text{ and } \Gamma \equiv <\Gamma_1, x = a : A, \Gamma_2> \end{cases} \\
|c|_\Gamma & = & c \quad \text{if } c \in \mathbf{C} \\
|a\ b|_\Gamma & = & |a|_\Gamma |b|_\Gamma \\
|\lambda x{:}A.\ b|_\Gamma & = & (\lambda x{:}|A|_\Gamma.\ |b|_{\Gamma,x:A}) \\
|\Pi x{:}A.\ B|_\Gamma & = & (\Pi x{:}|A|_\Gamma.\ |B|_{\Gamma,x:A}) \\
|x{=}a{:}A \ in \ b|_\Gamma & = & |b|_{\Gamma,x=a:A}
\end{array}$$

Note that $|a|_\Gamma \in \mathcal{T}$ and $Def(\Gamma) \cap FV(|a|_\Gamma) = \emptyset$. Conversely, if $a \in \mathcal{T}$ is such that $Def(\Gamma) \cap FV(a) = \emptyset$ then $|a|_\Gamma = a$. Hence $|\_|\_$ is a "projection" from $\mathcal{T}_\delta$ and $\mathcal{C}_\delta$ to $\mathcal{T}$. Later, it will be proved that $|a|_\Gamma$ is the δ-normal form of a.

b) A term a δ-reduces to its projection, i.e. $\Gamma \vdash a \twoheadrightarrow_\delta |a|_\Gamma$.

c) The projections of two terms that are in $\rightarrow_\delta$ are equal, i.e. if $\Gamma \vdash a \rightarrow_\delta b$ then $|a|_\Gamma = |b|_\Gamma$.

It follows from part **b)** that $\Gamma \vdash a \twoheadrightarrow_\delta |a|_\Gamma$, $\Gamma \vdash a_1 \twoheadrightarrow_\delta |a_1|_\Gamma$ and $\Gamma \vdash a_2 \twoheadrightarrow_\delta |a_2|_\Gamma$.
It follows from part **c)** that $|a|_\Gamma = |a_1|_\Gamma = |a_2|_\Gamma$.
The scheme of the proof is illustrated in the following diagram.

$$
\begin{tikzcd}
 & a \arrow[dl, ->>, "\delta"'] \arrow[dd, ->>, "\delta"] \arrow[dr, ->>, "\delta"] & \\
a_1 \arrow[d, ->>, "\delta"'] & & a_2 \arrow[d, ->>, "\delta"] \\
|a_1|_\Gamma \arrow[r, equal] & |a|_\Gamma \arrow[r, equal] & |a_2|_\Gamma
\end{tikzcd}
$$

The term $|a|_\Gamma$ is the δ-nf in Γ of a, a_1 and a_2.
The proof of Church Rosser for $\beta\delta$-reduction follows from the previous considerations given for the proof of theorem **??** and the following:

d) The projection preserves β-reduction, i.e. if $\Gamma \vdash a \twoheadrightarrow_\beta b$ then $|a|_\Gamma \twoheadrightarrow_\beta |b|_\Gamma$.

Theorem 3.2 (Church Rosser for $\beta\delta$-reduction). *Let $\Gamma \in \mathcal{C}_\delta$ and $a \in \mathcal{T}_\delta$ be such that $\Gamma \vdash a \twoheadrightarrow_{\beta\delta} a_1$ and $\Gamma \vdash a \twoheadrightarrow_{\beta\delta} a_2$. Then there exists a term $d \in \mathcal{T}_\delta$ such that $\Gamma \vdash a_1 \twoheadrightarrow_{\beta\delta} a_3$ and $\Gamma \vdash a_2 \twoheadrightarrow_{\beta\delta} a_3$.*

$$
\begin{tikzcd}
 & a \arrow[dl, ->>, "\beta\delta"'] \arrow[dd, ->>, "\delta"] \arrow[dr, ->>, "\beta\delta"] & \\
b \arrow[dd, ->>, "\delta"'] & & c \arrow[dd, ->>, "\delta"] \\
 & |a|_\Gamma \arrow[dl, ->>, "\beta"'] \arrow[dr, ->>, "\beta"] & \\
|b|_\Gamma \arrow[dr, ->>, dotted, "\beta"] & & |c|_\Gamma \arrow[dl, ->>, dotted, "\beta"'] \\
 & d &
\end{tikzcd}
$$

If follows from part **b)** that $\Gamma \vdash a \rightarrow_\delta |a|_\Gamma$, $\Gamma \vdash b \rightarrow_\delta |b|_\Gamma$ and that $\Gamma \vdash c \rightarrow_\delta |c|_\Gamma$. It follows from parts **c)** and **d)** that $\Gamma \vdash |a|_\Gamma \twoheadrightarrow_\beta |b|_\Gamma$ and that $\Gamma \vdash |a|_\Gamma \twoheadrightarrow_\beta |c|_\Gamma$. Hence the solid lines of the diagram above are justified.
By Church Rosser theorem for β-reduction between pseudoterms $\mathcal{T}$, the dotted arrows of the diagram are justified.
By composition along the borders, it follows that $\Gamma \vdash b \twoheadrightarrow_{\beta\delta} d$ and $\Gamma \vdash c \twoheadrightarrow_{\beta\delta} d$.

Definition 3.3. *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 simple proof of Weak Normalisation for δ-reduction is given. A term a is in δ-nf in a context Γ if and only if it does not contain definitions and its free variables are not in the context Γ. Hence $|a|_\Gamma$ is the δ-nf of a in Γ.

Theorem 3.4. (Weak Normalisation for δ-reduction) *For all $\Gamma \in \mathcal{C}_\delta$ and $a \in \mathcal{T}_\delta$, the term $|a|_\Gamma$ is the δ-nf of a in Γ.*

Definition 3.5. *Let $\Gamma \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$. For $\rho \in \{\delta, \beta\delta\}$, the term b ρ-strongly normalises in Γ if there is no infinite ρ-reduction starting with b in the context Γ.*

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

Theorem 3.6 (Strong Normalisation for δ). *The reduction δ is strongly-normalising, i.e. for all $\Gamma \in \mathcal{C}_\delta$ and $b \in \mathcal{T}_\delta$, the term b δ-strongly normalises in Γ.*

Strong Normalisation is proved for $\rightarrow_\delta$ by defining a function $nat_{\_}(\_) : \mathcal{C}_\delta \times \mathcal{T}_\delta \rightarrow \mathbb{N}$ which decreases with δ-reduction.

$$
\begin{array}{lcl}
nat_\Gamma(x) & = & \begin{cases} 0 & \text{if } x \in \mathbf{V} \text{ and } x \notin Def(\Gamma) \\ nat_{\Gamma_1}(a) + 1 & \text{if } x \in \mathbf{V} \text{ and } \Gamma \equiv < \Gamma_1, x = a : A, \Gamma_2 > \end{cases} \\
nat_\Gamma(c) & = & 0 \quad \text{if } c \in \mathbf{C} \\
nat_\Gamma(x{=}a{:}A \text{ } in \text{ } b) & = & nat_\Gamma(a) + nat_\Gamma(A) + nat_{\Gamma, x=a:A}(b) + 1 \\
nat_\Gamma(a \ b) & = & nat_\Gamma(a) + nat_\Gamma(b) \\
nat_\Gamma(\lambda x{:}A.\ b) & = & nat_{\Gamma, x:A}(b) + nat_\Gamma(A) \\
nat_\Gamma(\Pi x{:}A.\ B) & = & nat_{\Gamma, x:A}(B) + nat_\Gamma(A)
\end{array}
$$

4 Properties of Well-typed Terms

The properties in this section are proved for all well-typed terms, i.e. for terms a such that $\exists A, \Gamma[\Gamma \vdash a : A]$.
First the Subject Reduction theorem is proved for DPTS's. Then it is proved that λS is Weakly Normalising if and only if λS_δ is Weakly Normalising. Uniqueness of Types is proved for singly sorted DPTS's.

Theorem 4.1. **(Subject Reduction Theorem)** *If* $\Gamma \vdash a : A$ *and* $\Gamma \vdash a \rightarrow_{\beta\delta} a'$ *then* $\Gamma \vdash a' : A$

A map $|\_| : \mathcal{C}_\delta \rightarrow \mathcal{C}$ is defined as follows:

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

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

Theorem 4.2. *If* $\Gamma \vdash_{\lambda S_\delta} a : A$ *then* $|\Gamma| \vdash_{\lambda S} |a|_\Gamma : |A|_\Gamma$.

It follows from this theorem that the inclusion of definitions to a PTS does not increase the strength of the system, in the sense that there are not more typable terms with abbreviations.

Corollary 4.3. **(Conservativity)** *Let* $a \in \mathcal{T}$ *and* $\Gamma \in \mathcal{C}$. *Then*

1. $\exists A \;\; \Gamma \vdash_{\lambda S} a : A$ *iff* $\exists A \;\; \Gamma \vdash_{\lambda S_\delta} a : A$
2. $\exists A \;\; \Gamma \vdash_{\lambda S} A : a$ *iff* $\exists A \;\; \Gamma \vdash_{\lambda S_\delta} A : a$

The first part of this corollary states that a λS-term is typable in λS iff it is typable in λS_δ. The second part states that a λS-type is inhabited in λS iff it is inhabited in λS_δ. If types are interpreted as propositions and terms as proofs then the second part means that a proposition is provable in λS iff it is provable in λS_δ.

Theorem 4.4. *Let* λS *be a PTS.* λS *is* β*-weakly normalising if and only if* λS_δ *is* $\beta\delta$*-weakly normalising.*

A specification $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ is called *singly sorted* if

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

In all *singly sorted* PTS's, types are unique up to β-equality, i.e. if $\Gamma \vdash_{\lambda S} a : A$ and $\Gamma \vdash_{\lambda S} a : B$ then $\Gamma \vdash A =_\beta B$ (see [?]). This property is preserved when definitions are added:

Theorem 4.5 (Uniqueness of Types). *Let* S *be a singly sorted specification. If* $\Gamma \vdash_{\lambda S_\delta} a : A$ *and* $\Gamma \vdash_{\lambda S_\delta} a : B$ *then* $\Gamma \vdash A =_{\beta\delta} B$.

5 Strong Normalization

Strong normalization for $\beta\delta$-reduction is proved for certain DPTS's.

Definition 5.1. *Let λS_δ be a DPTS and $\rightarrow_\rho$ a reduction relation for $\rho \in \{\delta, \beta\delta\}$. Then λS_δ is ρ-strongly normalising if b ρ-strongly normalises in Γ for all $b \in \mathcal{T}_\delta$ and $\Gamma \in \mathcal{C}_\delta$ such that $\Gamma \vdash_{\lambda S_\delta} b : B$.*

The abbreviation $SN_\rho(\lambda S_\delta)$ denotes that λS_δ is ρ-strongly normalising.
The idea of the proof of Strong Normalization is as follows:

- We need a function from $\mathcal{T}_\delta \times \mathcal{C}_\delta$ to $\mathcal{T}$ that preserves the typing and that maps an infinite $\beta\delta$-reduction in an infinite β-reduction sequence. We cannot take the projection $\mid \_ \mid\_$ as this function because it may map infinite $\beta\delta$-reduction in a finite β-reduction sequence. We cannot translate a definition into an abstraction and an application because this translation does not preserve the typing. We define a mapping $\{\_\}\_$ that is a combination of $\mid \_ \mid\_$ and the translation of a definition into an abstraction and an application.

$$
\begin{array}{lcl}
\{x\}_\Gamma & = & \left\{ \begin{array}{ll} x & \text{if } x \in \mathbf{V} \text{ and } x \notin \mathit{Def}(\Gamma) \\ \{a\}_{\Gamma_1} & \text{if } x \in \mathbf{V} \text{ and } \Gamma \equiv < \Gamma_1, x = a : A, \Gamma_2 > \end{array} \right. \\
\{c\}_\Gamma & = & c \quad \text{if } c \in \mathbf{C} \\
\{a\ b\}_\Gamma & = & \{a\}_\Gamma \{b\}_\Gamma \\
\{\lambda x{:}A.\ b\}_\Gamma & = & \lambda x{:}\{A\}_\Gamma.\ \{b\}_{\Gamma, x:A} \\
\{\Pi x{:}A.\ B\}_\Gamma & = & \Pi x{:}\{A\}_\Gamma.\ \{B\}_{\Gamma, x:A} \\
\{x = a : A\ \mathit{in}\ b\}_\Gamma & = & (\lambda x{:}\{A\}_\Gamma.\ \{b\}_{\Gamma, x=a:A})\{a\}_\Gamma
\end{array}
$$

  The map $\mid \_ \mid\_$ is similar to the projection $\mid \_ \mid\_$. The value $\{a\}_\Gamma$ is a term that is obtained from a by unfolding all the definitions occuring in Γ and in a. However $\{\_\}\_$ differs from $\mid - \mid_-$ in the value given to $\{x{=}a{:}A\ \mathit{in}\ b\}_\Gamma$. Instead of removing the local definition it is translated to a β-redex, i.e. an application and an abstraction. Actually this is an special β-redex called K**-redex**(see [?]).

- The 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 to terms that are typable in a slighter larger PTS, i.e. if $\mathcal{S}'$ is a completion of $\mathcal{S}$ and $\Gamma \vdash_{\lambda\mathcal{S}\delta} a : A$ then $\{\Gamma\} \vdash_{\lambda\mathcal{S}'} \{a\}_\Gamma : \{A\}_\Gamma$.

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

  (C1) $\boldsymbol{S} \subseteq \boldsymbol{S}'$, $\boldsymbol{A} \subseteq \boldsymbol{A}'$, $\boldsymbol{R} \subseteq \boldsymbol{R}'$, and

  (C2) $\mathcal{S}'$ is full, i.e. for all s_1, $s_2 \in \boldsymbol{S}'$ there exists $s_3 \in \boldsymbol{S}'$ such that $(s_1, s_2, s_3) \in \boldsymbol{R}'$, and

(C3) for all $s \in \boldsymbol{S}$ there is a $s' \in \boldsymbol{S}'$ such that $(s : s') \in \boldsymbol{A}'$ (i.e. the sorts of S are not topsorts in S').

For example, the system λC_∞ is a completion of λC and itself.

Theorem 5.2. *Let $\mathcal{S} = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ and $\mathcal{S}' = (\boldsymbol{S}', \boldsymbol{A}', \boldsymbol{R}')$ be such that $\mathcal{S}'$ is a completion of $\mathcal{S}$. If the PTS $\lambda\mathcal{S}'$ is β-strongly normalising, then the DPTS $\lambda\mathcal{S}_\delta$ is $\beta\delta$-strongly normalising.*

Corollary 5.3. *$(\lambda C_\infty)_\delta$ is $\beta\delta$-strongly normalizing*

Proof: The system λC_∞ is a subsystem of ECC and ECC is strongly normalising (see [?]). $\square$

The system λC_∞ is a completion of all strongly normalising PTS's given in [?] and then all these systems extended with definitions are strongly normalizing. In particular the Calculus of Constructions extended with definitions is $\beta\delta$-strongly normalizing.

6 Conclusions and Further Research

In the case that definitions appear only in the context, strong normalization is very easy to prove. It is proved using the projection mapping $|\_|_\_$. Howewer complications appear when definitions occur in terms. We can not use the mapping $|\_|_\_$ above because it may contract an infinite $\beta\delta$-reduction sequence into a finite β-reduction sequence. We have proved that the Calculus of Constructions extended with definitions is $\beta\delta$-strongly normalizing and also that strong normalization is preserved for a wide spectrum of PTS's. We do not know of any strongly normalising PTS λS for which theorem **??** cannot be used to prove strong normalization of λS_δ.
It would be nicer to prove that for **any** β-strongly normalising PTS its extension is $\beta\delta$-strongly normalising. In order to prove this general statement for any PTS it would be sufficient to prove:
If S is β-strongly normalizing then there exits S' a completion of S that is β-strongly normalizing.

In other paper we prove that the property of $\beta\delta$-strong normalization for a DPTS is **equivalent** to the property of strong normalization for the same PTS extended with the combinator **K**. The extension of a PTS to include the combinator **K** is defined as follows. We add

- a new constant **K** to the set of pseudoterms
- a reduction rule

  $\mathbf{K}\ a\ b \to_k a \quad (\mathbf{k})$

- the following derivation rules:

$$(\mathbf{K}-formation) \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (\mathbf{K}\ a\ b) : A}$$

$$(\mathbf{K}-conversion) \qquad \frac{\Gamma \vdash b : B \quad \Gamma \vdash B' : s \quad \Gamma \vdash B =_k B'}{\Gamma \vdash b : B'}$$

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

It is not difficult to prove that:

Theorem 6.1. Equivalence between $SN_{\beta k}$ and $SN_{\beta\delta}$
λS extended with $\mathbf{K}$ is βk-strongly normalizing, iff the DPTS λS_δ is $\beta\delta$-strongly normalizing.

Using theorem **??** and the notion of saturated sets one can prove directly (without using the fact that the Calculus of Constructions is strongly normalizing) that the Calculus of Constructions extended with definitions is $\beta\delta$-strongly normalizing.

References

[Bar85] H. Barendregt. *The Lambda Calculus. Its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North Holland. Amsterdam, 1985.

[Bar92] H. 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, pages 117–309. Oxford University Press, 1992.

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

[dV87] R. de Vrijer. Exactly estimating functionals and strong normalization. In *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*, volume 90, pages 479–493, 1987.

[ea91] 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 dans l'arithmétique d'ordre supérieur.* PhD thesis, Université Paris VII, 1972.

[HA91] L. Helmink and R. M. C. Ahn. Goal directed proof construction in type theory. In *Procs. of the first Workshop on Logical Frameworks.* Cambridge University Press, 1991.

[HP89] R. Harper and R. Pollack. Type checking, universe polymorphism and typical ambiguity in the calculus of constructions. In *TAPSOFT'89*, volume 352 of *LNCS*, pages 241–256. Springer, 1989.

[LP92] Z. Luo and R. 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 *The fourth Annual Symposium on 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.

[RHP93] F. Honsell R. Harper and G. Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

[SP93] P. Severi and E. Poll. Pure type systems with definitions. Computing Science Note CS-9324, Eindhoven University of Technology, 1993.

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