Type Inference for Pure Type Systems

Paula Severi
Eduardo Acevedo 1139. Montevideo. Uruguay
severi@cmat.edu.uy

September 3, 1997

Abstract

In this paper we define a type inference semi-algorithm for singly sorted pure type systems. For that, we define the notion of pure type systems without the Π-condition and a mapping from pure type systems without the Π-condition to pure type systems. This allows us to prove the two main results: first that weak normalisation is preserved by the extension, second the correctness of the type inference semi-algorithm.

1 Introduction

A type inference semi-algorithm can be constructed if we have a set of typing rules that is syntax directed. The proof of the equivalence (soundness and completeness) between the original rules and the corresponding syntax directed ones is problematic for some pure type systems (see [vBJMP93] and also [Pol93]). Here we present a *syntax directed set of rules for singly sorted pure type systems* similar to the one presented in [vBJMP93]. As in [vBJMP93], we use an auxiliary system to check the Π-condition, i.e. the premise that the product $(\Pi x{:}A.\ B)$ should be well-typed in the abstraction rule. The auxiliary system we use to define the syntax directed set of rules is the corresponding pure type system without the Π-condition. The convenience of this system is that it permits us to prove soundness and completeness of the original rules for pure type systems with respect to the syntax directed ones.

This paper is organised as follows. In section 2, we define the class of pure type systems without the Π-condition. A pure type system without the Π-condition - $\lambda^{\omega}(S)$ - can be considered as an extension of the corresponding system with the Π-condition - $\lambda(S)$ -. The new terms of the extension are the abstractions $(\lambda x{:}A.\ b)$ whose type $(\Pi x{:}A.\ B)$ is not typable and applications of the form $(F\ a)$ whose operator F has a type that is not typable. Hence we are led to consider types that are not typable. We call them *toptypes*. In section 3, we prove the basic properties of pure type systems without the Π-condition, e.g. weak subject reduction theorem. In section 4, we analyse the shape of a toptype. We give a characterisation of the set of toptypes and prove weak type reduction. In the case of singly sorted specifications we give another characterisation of the set of toptypes and prove that it is closed under substitution and β-reduction. In section 5, we consider those β-redexes whose abstraction has a type that is not typable and we call them *illegal redexes*. We introduce a mapping φ that contracts all the illegal redexes of a term. Using φ we prove that weak normalisation is preserved by the extension from $\lambda(S)$ to $\lambda^{\omega}(S)$ in the case that S is singly sorted. In section 6, we define a syntax directed set of rules for singly sorted pure type systems. In section 7, we define a partial function **type** that computes the type of a term in a singly sorted pure type system based upon the syntax directed set of rules defined in the previous section.

2 Pure Type Systems without the Π-condition

In this section we define the notion of pure type systems without the Π-condition. For pure type systems the type of an abstraction $(\lambda x{:}A.\ b)$ is a dependent product $(\Pi x : A.\ B)$ and the abstraction rule has the premise that the product $(\Pi x{:}A.\ B)$ should be well-typed. This premise is called the Π-condition. The problem with the Π-condition is that sometimes it is not well-adapted to induction. The idea is to remove this condition in order to define a syntax directed set of rules and prove the equivalence of the original rules with the syntax directed ones.

We recall the definition of specification and pure type systems (see [Bar92]).

Definition 2.1. A *specification* is a triple $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ such that

1. $\boldsymbol{S}$ is a set of symbols called *sorts*,
2. $\boldsymbol{A} \subseteq \boldsymbol{S} \times \boldsymbol{S}$ called set of *axioms*,
3. $\boldsymbol{R} \subseteq \boldsymbol{S} \times \boldsymbol{S} \times \boldsymbol{S}$ called set of *rules*.

We say that a sort s is a *topsort* if there is no s' such that $(s, s') \in \boldsymbol{A}$.

Definition 2.2. Let $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ be a specification. The specification S is called *singly sorted* if

1. $(s_1, s_2), (s_1, s_3) \in \boldsymbol{A}$ implies $s_2 = s_3$
2. $(s_1, s_2, s_3), (s_1, s_2, s_4) \in \boldsymbol{R}$ implies $s_3 = s_4$.

Definition 2.3. The set $\mathcal{T}$ of *pseudoterms* and the set $\mathcal{C}$ of *pseudocontexts* are defined as follows.

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

where V is a set of variables, $\boldsymbol{S}$ is the set of sorts and ϵ is the empty context.

Syntactic equality is denoted by $=$ and β-conversion is denoted by $=_\beta$.

Definition 2.4. Let $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ be a specification. The *typing relation* $\vdash_S$ (or $\vdash$ for short) is the smallest relation on $\mathcal{C} \times \mathcal{T} \times \mathcal{T}$ closed under the following rules.

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

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

A *pure type system* is denoted by $\lambda(S)$ and is defined as $(\mathcal{T}, \mathcal{C}, \rightarrow_\beta, \vdash)$.

A pseudoterm b is *typable* in Γ if there exists B such that $\Gamma \vdash b : B$. A pseudoterm b is *typable* (or a *term*) if there exists Γ such that b is typable in Γ. A pseudoterm B is a *type* if there are Γ and b such that $\Gamma \vdash b : B$. A pseudocontext Γ is *legal* if there are A and B such that $\Gamma \vdash A : B$.

The Π-*condition* is the premise $\Gamma \vdash (\Pi x{:}A.\ B) : s$ of the abstraction rule.

We remove the Π-condition from the abstraction rule and we obtain a new typing relation called $\vdash^\omega$.

Definition 2.5. Let S be a specification. The typing relation $\vdash^\omega$ (or $\vdash^\omega_S$) is the smallest relation defined by the same rules as in definition 2.4 except for the abstraction rule that is replaced by the following one.

$$(abstraction)\dfrac{\Gamma, x : A \vdash^\omega b : B}{\Gamma \vdash^\omega (\lambda x{:}A.\ b) : (\Pi x{:}A.\ B)}$$

A *pure type system without the* Π*-condition* is denoted by $\lambda^\omega(S)$ and is defined as $(\mathcal{T}, \mathcal{C}, \to_\beta, \vdash^\omega)$.

We can define the notions of typable, of term and of type as for pure type systems.

Note that if a term is typable in $\lambda(S)$ then it is typable in $\lambda^\omega(S)$. Therefore $\lambda^\omega(S)$ is an extension of $\lambda(S)$.

The converse is not true. There may be terms typable in $\lambda^\omega(S)$ which are not typable in $\lambda(S)$. For example, the term $\lambda x{:}{*}.(* \to *)$ is not typable in the calculus of constructions but it has type $(* \to \Box)$ in the corresponding system without the Π-condition.

3 Basic Properties

In this section we prove a weak version of the subject reduction theorem for pure type systems without the Π-condition. Since correctness of types does not hold, we prove some weaker lemmas that we call correctness of contexts and correctness of types for variables. Finally we prove uniqueness of types for singly sorted specifications.

Lemma 3.1. (Generation Lemma)

1. If $\Gamma \vdash^\omega s : D$ then there exists s' such that $(s, s') \in \boldsymbol{A}$ and $D =_\beta s'$.
2. If $\Gamma \vdash^\omega x : D$ then there are s, b and B such that $\Gamma \vdash^\omega B : s$, $x : B \in \Gamma$ and $B =_\beta D$.
3. If $\Gamma \vdash^\omega (\Pi x : A.B) : D$ then there is a rule $(s_1, s_2, s_3) \in \boldsymbol{R}$ such that $\Gamma \vdash^\omega A : s_1$, $\Gamma, x{:}A \vdash^\omega B : s_2$ and $D =_\beta s_3$.
4. If $\Gamma \vdash^\omega (\lambda x{:}A.\ b) : D$ then there is a B such that $\Gamma, x{:}A \vdash^\omega b : B$ and $D =_\beta (\Pi x{:}A.\ B)$.
5. If $\Gamma \vdash^\omega (b \quad a) : D$ then there are A, B such that $\Gamma \vdash^\omega b : (\Pi x{:}A.\ B)$, $\Gamma \vdash^\omega a : A$ and $D =_\beta B[x := a]$.

All parts of the generation lemma are proved by induction on the derivation of a term.

Lemma 3.2. (Correctness of Contexts) If $\Gamma, x{:}A, \Gamma' \vdash^\omega b : B$ then there exists an s such that $\Gamma \vdash^\omega A : s$.

This lemma is proved by induction on the derivation of $\Gamma, x{:}A, \Gamma' \vdash^\omega b : B$.

Lemma 3.3. (Correctness of Types for Variables) If $\Gamma \vdash^\omega x : A$ then there exists s such that $\Gamma \vdash^\omega A : s$.

This lemma is proved by induction on the derivation of $\Gamma \vdash^{\omega} x : A$.

Lemma 3.4. Let $\Gamma \vdash^{\omega} (\lambda x{:}A.\ b) : D$. Then there exists a B such that $\Gamma, x : A \vdash^{\omega} b : B$. Moreover either $D = (\Pi x{:}A.\ B)$ or $\Gamma \vdash D : s$ and $D =_{\beta} (\Pi x{:}A.\ B)$ for some sort s.

This lemma is proved by induction on the derivation of $\Gamma \vdash^{\omega} (\lambda x{:}A.\ b) : D$.

Lemma 3.5. If $\Gamma \vdash^{\omega} A : s$ and $\Gamma \vdash a : A$ then $\Gamma \vdash A : s'$ for some sort s'.

Proof: By the lemma of correctness of types for pure type systems, we have that $\Gamma \vdash A : s''$ or $A = s''$ for some s''. In case $A = s''$, we have that (s'', s) is an axiom. $\square$

Lemma 3.6. (Thinning Lemma) Let $\Gamma' \vdash^{\omega} b : B$.

$$\text{If } \Gamma \vdash^{\omega} a : A \text{ and } \Gamma \subseteq \Gamma' \text{ then } \Gamma' \vdash^{\omega} a : A.$$

This lemma is proved by induction on the derivation of $\Gamma \vdash^{\omega} a : A$.

Lemma 3.7. (Substitution Lemma) If $\Gamma \vdash^{\omega} a : A$ and $\Gamma, x{:}A, \Gamma' \vdash^{\omega} b : B$ then $\Gamma, \Gamma'[x := a] \vdash^{\omega} b[x := a] : B[x := a]$.

This lemma is proved by induction on the derivation of $\Gamma, x{:}A, \Gamma' \vdash^{\omega} b : B$.

Lemma 3.8. (Correctness of Domains) If $\Gamma \vdash^{\omega} F : \Pi x : A.\ B$ then $\Gamma \vdash^{\omega} A : s$ for some sort s.

Proof: If $(\Pi x{:}A.\ B)$ is typable in Γ then by generation lemma we have that $\Gamma \vdash^{\omega} A : s$ for some sort s. If $(\Pi x{:}A.\ B)$ is not typable in Γ then we can prove by induction on the derivation of $\Gamma \vdash^{\omega} F : \Pi x{:}A.\ B$ that if $\Pi x{:}A.\ B = \Pi x_1 : A_1 \ldots x_n{:}A_n.\ D$ then for all i there exists s_i such that $\Gamma, x_1{:}A_1, \ldots x_{i-1}{:}A_{i-1} \vdash^{\omega} A_i : s_i$. $\square$

Theorem 3.9. (Weak Subject Reduction Theorem)
Let $\Gamma \vdash^{\omega} e : E$.

a) If $e \rightarrow_{\beta} e'$ then there exists E' such that $E \twoheadrightarrow_{\beta} E'$ and $\Gamma \vdash^{\omega} e' : E'$.

b) If $\Gamma \rightarrow_{\beta} \Gamma'$ then $\Gamma' \vdash^{\omega} e : E$.

Proof: The two statements are proved by simultaneous induction on the derivation of $\Gamma \vdash^{\omega} e : E$. We only consider the statement a) for the case of the application rule.

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

There are several cases when we reduce the application $(b \quad a)$.

1. Suppose $(b \quad a) \to_\beta (b' \quad a)$ with $b \to_\beta b'$. By induction we have that $\Gamma \vdash^\omega b' : (\Pi x{:}A'.\ B')$ for $A \twoheadrightarrow_\beta A'$ and $B \twoheadrightarrow_\beta B'$. By lemma 3.8 we have that $\Gamma \vdash^\omega A' : s$ for some sort s. Therefore we can apply the conversion rule and we obtain $\Gamma \vdash^\omega a : A'$. Then we apply the application rule and we have a derivation of $\Gamma \vdash^\omega (b \quad a) : B'[x := a]$.

2. Suppose $(b \quad a) \to_\beta (b \quad a')$ with $a \to_\beta a'$. By induction we have that $\Gamma \vdash^\omega a' : A'$ for $A \twoheadrightarrow_\beta A'$. By lemma 3.8 we have that $\Gamma \vdash^\omega A : s$. Therefore we can apply the conversion rule and we have that $\Gamma \vdash^\omega a' : A$. Then we apply the application rule and we obtain a derivation of $\Gamma \vdash^\omega (b \quad a') : B[x := a']$.

3. Suppose $(\lambda x : A'.\ d)a \to_\beta d[x := a]$. By lemma 3.4 we have that $\Gamma, x : A' \vdash^\omega d : B'$. Moreover by lemma 3.4 we know there are two possibilities, either $(\Pi x : A.\ B) = (\Pi x : A'.\ B')$ or $(\Pi x : A.\ B) =_\beta (\Pi x : A'.\ B')$ and $\Gamma \vdash^\omega (\Pi x{:}A.\ B) : s$.

   (a) Suppose $(\Pi x{:}A.\ B) = (\Pi x{:}A'.\ B')$. Hence $A = A'$ and $B = B'$. By the substitution lemma we have that $\Gamma \vdash^\omega d[x := a] : B[x := a]$.

   (b) Suppose $(\Pi x{:}A.\ B) =_\beta (\Pi x{:}A'.\ B')$ and $\Gamma \vdash^\omega (\Pi x{:}A.\ B) : s$ for some s. By lemma 3.8 we have that $\Gamma \vdash^\omega A' : s'$ for some s'. Therefore we can apply the conversion rule and we have that $\Gamma \vdash^\omega a : A'$. By the substitution lemma we have that $\Gamma \vdash^\omega d[x := a] : B'[x := a]$. Since $\Gamma \vdash^\omega (\Pi x{:}A.\ B) : s$, it follows from the generation lemma that $\Gamma, x{:}A \vdash^\omega B : s_2$ for some sort s_2. By the substitution lemma we have that $\Gamma \vdash^\omega B[x := a] : s_2$ and by conversion rule we have that $\Gamma \vdash^\omega d[x := a] : B[x := a]$.

The rest of the cases are easy to prove. □

The next example shows that subject reduction does not hold for pure type systems without the Π-condition. If $\Gamma \vdash^\omega e : E$ and $e \to_\beta e'$ then e' may not have type E. This happens when the type E is not typable and hence we cannot apply conversion rule.

Example 3.10. Let $\Gamma =< \gamma : * >$.

$$\begin{array}{rlll} e & = & (\lambda\alpha{:}*.\lambda x{:}\alpha.*)((\lambda\alpha{:}*.\alpha)\gamma) & \\ & \to_\beta & (\lambda\alpha{:}*.\lambda x{:}\alpha.*)\gamma & = \quad e' \end{array}$$

In the calculus of constructions without the Π-condition, we have that $\Gamma \vdash^\omega e : (\lambda\alpha{:}*.\alpha)\gamma \to \Box$ and $\Gamma \vdash^\omega e' : \gamma \to \Box$ but not $\Gamma \vdash^\omega e' : (\lambda\alpha{:}*.\alpha)\gamma \to \Box$ (the latter can be proved by some meta-theoretical reasoning).

Theorem 3.11. (Uniqueness of Types) Let S be a singly sorted specification. If $\Gamma \vdash^{\omega} a : A$ and $\Gamma \vdash^{\omega} a : B$ then $A =_{\beta} B$.

This theorem is proved by induction on the derivation of $\Gamma \vdash^{\omega} a : A$.

4 Description of Toptypes

Toptypes are types that are not typable. In this section we study the form of a toptype for pure type systems without the Π-condition and prove weak type reduction. Also we prove for singly sorted specifications that the set of toptypes is closed under β-reduction and under substitution.

Definition 4.1. We say that A is *a toptype in* Γ if there exists a such that $\Gamma \vdash^{\omega} a : A$ and $\Gamma \not\vdash^{\omega} A : -$ and that A is a *toptype* if there are a and Γ such that $\Gamma \vdash^{\omega} a : A$ and $\Gamma \not\vdash^{\omega} A : -$.

For example, $(A \to \Box)$ is a toptype in the calculus of constructions without the Π-condition.

Definition 4.2. We say that the product $\Pi x : A.\ B$ *can be formed* in the context Γ if there exists an s such that $\Gamma \vdash^{\omega} (\Pi x{:}A.\ B) : s$.

We define the set $\mathcal{M}_{\Gamma}$ of 'potential' toptypes in Γ. The set $\mathcal{M}_{\Gamma}$ is not exactly the set of toptypes. If an element is in the set $\mathcal{M}_{\Gamma}$ then it is not typable in Γ. However it may or may not be inhabited.

Definition 4.3. Let $\Gamma \in \mathcal{C}$. We define the set $\mathcal{M}_{\Gamma}$ as the smallest subset of $\mathcal{T}$ satisfying the following clauses.

1. if s is a topsort then $s \in \mathcal{M}_{\Gamma}$,
2. if $\Gamma, x{:}A \vdash^{\omega} B : s$ for some s and the product $(\Pi x{:}A.\ B)$ cannot be formed in Γ then $(\Pi x{:}A.\ B) \in \mathcal{M}_{\Gamma}$.
3. if $B \in \mathcal{M}_{\Gamma, x:A}$ then $(\Pi x{:}A.\ B) \in \mathcal{M}_{\Gamma}$.

Definition 4.4. Let $\Gamma \in \mathcal{C}$.

1. A mapping $depth_{\Gamma} : \mathcal{M}_{\Gamma} \to \mathbb{N}$ is defined as follows.

$$\begin{array}{rcll} depth_{\Gamma}(s) & = & 0 & \text{if } s \text{ is a topsort} \\ depth_{\Gamma}(\Pi x{:}A.\ B) & = & 1 & \text{if } \Gamma, x{:}A \vdash^{\omega} B : s \text{ for some } s \\ & & & \text{and } (\Pi x{:}A.\ B) \text{ cannot be} \\ & & & \text{formed in } \Gamma \\ depth_{\Gamma}(\Pi x{:}A.\ B) & = & depth_{\Gamma, x:A}(B) + 1 & \text{otherwise} \end{array}$$

2. We extend the mapping *depth* to pseudoterms as follows.

$$depth_\Gamma(A) = 0 \text{ for } A \in \mathcal{T} - \mathcal{M}_\Gamma \text{ and } \Gamma \in \mathcal{C}.$$

Next we prove that a pseudoterm in the set $\mathcal{M}_\Gamma$ is a sequence of products whose 'heart' is either a topsort or it is a term of type s. The number of outermost Π's which are before this 'legal heart' is given by the function $depth_\Gamma$.

Lemma 4.5. Let $\Gamma \in \mathcal{C}$ and $A \in \mathcal{T}$. The following two statements are equivalent:

1. $A \in \mathcal{M}_\Gamma$, $n = depth_\Gamma(A)$,
2. there are $A_1, \ldots, A_n, B$ such that $A = \Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n.B$ where

   either $n \geq 0$ and B is a topsort s_0

   or $n \geq 1$ and $\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \vdash^\omega B : s$ for some sort s and $(\Pi x_n{:}A_n.\ B)$ cannot be formed in $\Gamma, x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1}$.

Proof:

$(1 \Rightarrow 2)$. This is proved by induction on $A \in \mathcal{M}_\Gamma$.

$(2 \Rightarrow 1)$. There are two cases:

1. Suppose $n \geq 0$ and B is a topsort s_0. We apply once the first clause and then n times the third clause.
2. Suppose that $\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \vdash^\omega B : s$ for some sort s and the product $(\Pi x_n{:}A_n.\ B)$ cannot be formed in the context $\Gamma, x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1}$.

   We apply once the second clause and then n times the third clause.

□

Next lemma says that the pseudoterms in the set $\mathcal{M}_\Gamma$ are not typable in the context Γ.

Lemma 4.6. If $A \in \mathcal{M}_\Gamma$ then $\Gamma \not\vdash^\omega A : -$.

This is proved by induction on $A \in \mathcal{M}_\Gamma$.

Lemma 4.7. Let $\Gamma \vdash^\omega a : A$. If there is no sort s such that $\Gamma \vdash^\omega B[x := a] : s$ then there is no sort s such that $\Gamma \vdash^\omega (\Pi x{:}A.\ B) : s$.

Proof: Suppose $\Gamma \vdash^{\omega} (\Pi x{:}A.\ B) : s'$. By the generation lemma we have that $\Gamma, x{:}A \vdash^{\omega} B : s$. By the substitution lemma we have that $\Gamma \vdash^{\omega} B[x := a] : s$. □

Next we prove that a type in Γ that does not have a sort as type is in the set $\mathcal{M}_{\Gamma}$.

Lemma 4.8. Suppose that $\Gamma \vdash^{\omega} a : A$.

If there is no sort s such that $\Gamma \vdash A : s$ then $A \in \mathcal{M}_{\Gamma}$.

Proof: This lemma is proved by induction on the derivation of $\Gamma \vdash^{\omega} a : A$. We will prove the case of the application.

$$\text{(application)}\ \frac{\Gamma \vdash f : (\Pi x{:}D.\ E) \quad \Gamma \vdash d : D}{\Gamma \vdash (f \quad d) : E[x := d]}.$$

If there is no s such that $\Gamma \vdash^{\omega} E[x := d] : s$ then by lemma 4.7 there is no s such that $\Gamma \vdash (\Pi x{:}D.\ E) : s$.

By induction we have that $(\Pi x{:}D.\ E) \in \mathcal{M}_{\Gamma}$. By lemma 4.5 we have that $E = \Pi x_1 : A_1 \ldots \Pi x_n : A_n.H$ where either H is a topsort s_0 or $\Gamma, x : D, x_1 : A_1, \ldots, x_n{:}A_n \vdash^{\omega} H : s$ for some sort s.

Then $E[x := d] = \Pi x_1 : A_1[x := d] \ldots \Pi x_n : A_n[x := d].H[x := d]$ where $H[x := d]$ is either a topsort s_0 or $\Gamma, x_1{:}A_1[x := d], \ldots, x_n{:}A_n[x := d] \vdash^{\omega} H[x := d] : s$.

1. Suppose that $H[x := d]$ is a topsort s_0. By lemma 4.5 we have that $E[x := d] \in \mathcal{M}_{\Gamma}$.

2. Suppose that $\Gamma, x_1{:}A_1[x := d], \ldots, x_n{:}A_n[x := d] \vdash^{\omega} H[x := d] : s$. There should be a natural number k with $1 \leq k \leq n$ and a sort s' such that

   a) $\Gamma, x_1{:}A_1[x := d] \ldots x_k{:}A_k[x := d] \vdash^{\omega} (\Pi x_{k+1}{:}A_{k+1} \ldots \Pi x_n{:}A_n.\ B)[x := d] : s'$ and

   b) the product $(\Pi x_k{:}A_k \ldots \Pi x_n{:}A_n.\ B)[x := d]$ cannot be formed in the context $\Gamma, x_1{:}A_1[x := d] \ldots x_{k-1}{:}A_{k-1}[x := d]$.

   Therefore by lemma 4.5 we have that $E[x := d] \in \mathcal{M}_{\Gamma}$.

The rest of the cases are easy to prove. □

The next theorem says that a type is not typable in a context Γ if and only if it is in the set $\mathcal{M}_{\Gamma}$. Hence, A is a toptype in Γ if and only if A is a type in Γ and it is in the set $\mathcal{M}_{\Gamma}$.

Theorem 4.9. (Description of Toptypes) Let $\Gamma \vdash^{\omega} a : A$. The following statements are equivalent:

1. $\Gamma \not\vdash^{\omega} A : -$,

2. There is no sort s such that $\Gamma \vdash A : s$,

3. $A \in \mathcal{M}_{\Gamma}$.

Proof: The proof of ($1 \Rightarrow 2$) is trivial. The implication ($2 \Rightarrow 3$) is lemma 4.8. The implication ($3 \Rightarrow 1$) is lemma 4.6. $\square$

Next lemma says that a term e whose type E is a toptype reduces to another term e' which has a sequence of outermost abstractions. The number of outermost λ's in this term is the same number of Π's in E given by the function *depth*.

Lemma 4.10. Let $\Gamma \vdash^{\omega} e : E$ and $\Gamma \not\vdash^{\omega} E : -$ be such that $depth(E) = n$ and $E = (\Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n.\ B)$. Then there exists $e' = (\lambda x_1{:}A_1 \ldots \lambda x_n{:}A_n.\ b)$ such that $e \twoheadrightarrow_{\beta} e'$ and $\Gamma \vdash^{\omega} e' : E$.

Proof: This is proved by induction on the derivation of $\Gamma \vdash^{\omega} e : E$. Only one case is considered.

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

Suppose that $depth(\Pi x{:}A.\ B) = m + 1$. By induction hypothesis we have that $b \twoheadrightarrow_{\beta} b'$, $b' = (\lambda x{:}A.\ \lambda x_1{:}A_1 \ldots \lambda x_m{:}A_m.\ d)$ and $\Gamma \vdash^{\omega} b' : (\Pi x{:}A.\ B)$ for some $A_1, \ldots A_m$ and d.

By lemma 4.5 and substitution lemma, we have that $depth_{\Gamma}(B[x := a]) = n \leq m = depth_{\Gamma,x:A}(B)$.

Therefore $(b\ \ a) \twoheadrightarrow_{\beta} d$ and $d = (\lambda x_1{:}A_1[x := a] \ldots \lambda x_m{:}A_m[x := a].\ d[x := a])$. By the substitution lemma we have that $\Gamma \vdash^{\omega} d : B[x := a]$. $\square$

Theorem 4.11. (Weak Type Reduction Theorem)
If $\Gamma \vdash^{\omega} e : E$ and $E \rightarrow_{\beta} E'$ then there exists e' such that $e \twoheadrightarrow_{\beta} e'$ and $\Gamma \vdash^{\omega} e' : E'$.

Proof: There are two possibilities, either E' is a toptype or not.

1. Suppose E' is not a toptype. By theorem 4.9, $\Gamma \vdash^{\omega} E' : s$. Applying the conversion rule, we have that $\Gamma \vdash e : E'$.

2. Suppose E' is a toptype. Then E is a toptype and $E = (\Pi x_1{:}A_1 \ldots \Pi x_n{:} A_n.\ B)$ with $n = depth(E)$.

   By lemma 4.10 there exists e' such that $e \twoheadrightarrow_{\beta} e' = (\lambda x_1{:}A_1 \ldots \lambda x_n{:}A_n.\ b)$ and $\Gamma \vdash^{\omega} e' : E$. Hence $\Gamma, x_1{:}A_1 \ldots, x_n{:}A_n \vdash^{\omega} b : B$.

   (a) If $A_i \rightarrow_{\beta} A'_i$ then by the weak subject reduction theorem $\Gamma, x_1 : A_1 \ldots x_i{:}A'_i, \ldots, x_n{:}A_n \vdash^{\omega} b : B$.

(b) If $B \to_\beta B'$ then by lemma 4.5 and the weak subject reduction theorem we have that $\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \vdash^\omega B' : s$.

Applying the conversion rule we have that $\Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \vdash^\omega b : B'$.

□

For singly sorted specifications, we give another characterisation of the set of toptypes. Using this characterisation we prove that for singly sorted specifications, the toptypes are closed under β-reduction and substitution.

Next we define the set $\mathcal{N}_\Gamma$ similar to the set $\mathcal{M}_\Gamma$. The elements in $\mathcal{N}_\Gamma$ are not typable and they may or may not be inhabited.

Definition 4.12. We define the set $\mathcal{N}_\Gamma$ as the smallest subset of $\mathcal{T}$ satisfying the following clauses.

1. if s is a topsort then $s \in \mathcal{N}_\Gamma$,
2. if there are sorts s_1, s_2 such that $\Gamma \vdash^\omega A : s_1$, that $\Gamma, x{:}A \vdash^\omega B : s_2$ and there is no s_3 such that $(s_1, s_2, s_3) \in \boldsymbol{R}$ then $(\Pi x{:}A.\ B) \in \mathcal{N}_\Gamma$.
3. if $B \in \mathcal{N}_{\Gamma, x:A}$ then $(\Pi x{:}A.\ B) \in \mathcal{N}_\Gamma$.

Note that definitions 4.3 and 4.12 differ only in the second clause. For the case of singly sorted specifications, to require that the product $(\Pi x{:}A.\ B)$ cannot be formed in Γ is equivalent to require that there are sorts s_1, s_2 such that $\Gamma \vdash^\omega A : s_1$, that $\Gamma, x : A \vdash^\omega B : s_2$ and there is no sort s_3 such that $(s_1, s_2, s_3) \in \boldsymbol{R}$.

Note that the inclusion $\mathcal{M}_\Gamma \subset \mathcal{N}_\Gamma$ holds for any specification. If the specification is not singly sorted, $\mathcal{N}_\Gamma$ may have more elements than $\mathcal{M}_\Gamma$.

Example 4.13. Let $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ be the non-singly sorted specification such that $\boldsymbol{S} = \{0, 1, 2\}$, $\boldsymbol{A} = \{(0,1), (0,2)\}$ and $\boldsymbol{R} = \{(2,2,2)\}$. We have that $(0 \to 0) \in \mathcal{N}_\Gamma$ but $(0 \to 0) \notin \mathcal{M}_\Gamma$.

Lemma 4.14. Let S be a singly sorted specification. Then $\mathcal{M}_\Gamma = \mathcal{N}_\Gamma$.

$[\mathcal{M}_\Gamma \subset \mathcal{N}_\Gamma]$. It is proved by induction on the definition of $D \in \mathcal{M}_\Gamma$. $[\mathcal{N}_\Gamma \subset \mathcal{M}_\Gamma]$. It is proved by induction on the definition of $D \in \mathcal{N}_\Gamma$.

As a consequence of the previous lemma, the set $\mathcal{N}_\Gamma$ has the same properties as the set $\mathcal{M}_\Gamma$ if the specification is singly sorted.

Lemma 4.15. Let S be a singly sorted specification.

$$\text{If } A \in \mathcal{N}_\Gamma \text{ then } \Gamma \not\vdash^\omega A : -.$$

Proof: This follows from lemmas 4.6 and 4.14. □

The next theorem states that a type is in $\mathcal{N}_\Gamma$ if and only if this type is not typable. Hence A is a toptype in Γ if and only if A is a type and it is in the set $\mathcal{N}_\Gamma$.

Theorem 4.16. Let S be a singly sorted specification and $\Gamma \vdash^\omega a : A$.
The following statements are equivalent:

1. $\Gamma \not\vdash^\omega A : -$.

2. $A \in \mathcal{N}_\Gamma$.

Proof: This follows from theorem 4.9 and lemma 4.14. □

In the following lemma we prove that the set $\mathcal{N}_\Gamma$ is closed under β-reduction.

Lemma 4.17. If $D \in \mathcal{N}_\Gamma$ and $D \twoheadrightarrow_\beta D'$ then $D' \in \mathcal{N}_\Gamma$.

Proof: The following two statements are proved by induction on the definition of $D \in \mathcal{N}_\Gamma$.

a) if $D \rightarrow_\beta D'$ then $D' \in \mathcal{N}_\Gamma$,

b) if $\Gamma \rightarrow_\beta \Gamma'$ then $D \in \mathcal{N}_{\Gamma'}$.

We only prove statement **a)** when the pseudoterm D is $(\Pi x{:}A.\ B)$ and there are sorts s_1, s_2 such that $\Gamma \vdash^\omega A : s_1$, $\Gamma, x{:}A \vdash B : s_2$ and there is no sort s_3 such that $(s_1, s_2, s_3) \in \boldsymbol{R}$.

If $A \rightarrow_\beta A'$ then it follows from weak subject reduction theorem that $\Gamma \vdash^\omega A' : s_1$ and $\Gamma, x{:}A' \vdash^\omega B : s_2$. Hence $(\Pi x{:}A'.\ B) \in \mathcal{N}_\Gamma$.

Similarly if $B \rightarrow_\beta B'$ we have that $(\Pi x{:}A.\ B') \in \mathcal{N}_\Gamma$. □

As a consequence of the previous lemma, we have that the set of toptypes is closed under β-reduction.

Theorem 4.18. **(β-closure of toptypes)** Let S be a singly sorted specification.

$$\text{If } A \text{ is a toptype and } A \twoheadrightarrow_\beta A' \text{ then } A' \text{ is a toptype.}$$

Proof: If A is a toptype then there are Γ and a such that $\Gamma \vdash^\omega a : A$ and $\Gamma \not\vdash^\omega A : -$. By theorem 4.16 we have that $A \in \mathcal{N}_\Gamma$. By lemma 4.17 we have that $A' \in \mathcal{N}_\Gamma$. By lemma 4.15 we have that $\Gamma \not\vdash^\omega A' : -$. By the weak type reduction theorem we have that there exists a' such that $\Gamma \vdash a' : A'$. Hence A' is a toptype. □

Corollary 4.19. Let S be a singly sorted specification and $\Gamma \vdash^\omega b : B$.
If there exists a sort s such that $\Gamma \vdash^\omega B' : s$ and $B' =_\beta B$ then $\Gamma \vdash^\omega B : s$.

Proof: By Church-Rosser theorem, there exists a common reduct D_0 of B and B'. By weak subject reduction theorem we have that $\Gamma \vdash^{\omega} D_0 : s$. By theorems 4.9 and 4.18 we have that $\Gamma \vdash^{\omega} B : s'$ for some s'. By weak subject reduction $\Gamma \vdash^{\omega} D_0 : s'$ and by uniqueness of types we have that $s = s'$. $\square$

In the following, we prove that the set of toptypes is closed under substitution.

Lemma 4.20. Let S be a singly sorted specification. Let $\Gamma \vdash d : D$.

$$\text{If } E \in \mathcal{N}_{\Gamma, y:D, \Gamma'} \text{ then } E[y := d] \in \mathcal{N}_{\Gamma, \Gamma'[y:=d]}.$$

Proof: This is proved by induction on $E \in \mathcal{N}_{\Gamma, y:D, \Gamma'}$. Only one case is considered.

Suppose that E is $(\Pi x : A.\ B)$ and that there are sorts s_1, s_2 such that $\Gamma, y\!:\!D, \Gamma' \vdash^{\omega} A : s_1$ and $\Gamma, y\!:\!D, \Gamma', x\!:\!A \vdash B : s_2$. Moreover there is no sort s_3 such that $(s_1, s_2, s_3) \in \boldsymbol{R}$. Substitution lemma yields $\Gamma, \Gamma' \vdash^{\omega} A[y := d] : s_1$ and $\Gamma, \Gamma'[y := d], x\!:\!A[y := d] \vdash B[y := d] : s_2$. Hence $(\Pi x\!:\!A.\ B)[y := d] \in \mathcal{N}_{\Gamma, \Gamma'[y:=d]}$. $\square$

Theorem 4.21. (Substitution on Toptypes) Let S be a singly sorted specification. Suppose that $\Gamma, y\!:\!D, \Gamma' \vdash e : E$ and that $\Gamma \vdash d : D$.

$$\text{If } \Gamma, y\!:\!D, \Gamma' \not\vdash^{\omega} E : - \text{ then } \Gamma, \Gamma'[y := d] \not\vdash^{\omega} E[y := d] : -.$$

Proof: By theorem 4.16 we have that $E \in \mathcal{N}_{\Gamma, y:D, \Gamma'}$. By lemma 4.20 we have that $E[y := d] \in \mathcal{N}_{\Gamma, \Gamma'[y:=d]}$. By lemma 4.15 we have that $\Gamma, \Gamma'[y := d] \not\vdash^{\omega} E[y := d] : -$. $\square$

Corollary 4.22. Let S be a singly sorted specification. Let $\Gamma \vdash^{\omega} F : (\Pi x\!:\! A.\ B)$, $\Gamma \vdash^{\omega} B[x := a] : s$ and $\Gamma \vdash^{\omega} a : A$. Then $\Gamma, x\!:\!A \vdash^{\omega} B : s$.

The next example shows that when the specification is not singly sorted, a toptype may become typable after substitution.

Example 4.23. The following specification is not singly sorted:

S	$\boldsymbol{S}$	$0, 1, 2$
	$\boldsymbol{A}$	$0 : 1, 0 : 2, 1 : 2$
	$\boldsymbol{R}$	$(2, 2)$

Given the context $< x\!:\!1 >$, the term $(x \to x)$ is a toptype. Substituting x by 0 we obtain $(0 \to 0)$ which has type 2.

5 Normalisation for β-reduction

In this section we define the notion of illegal redex and a function $\varphi : \mathcal{C} \times \mathcal{T} \to \mathcal{T}$ that contracts the illegal redexes of a term (we write $\varphi_\Gamma(a)$ instead of $\varphi(\Gamma, a)$). We prove that φ_Γ is a strategy for the β-reduction. Moreover we prove that for singly sorted specifications, if a term b is typable in $\lambda^\omega(S)$ then $\varphi_\Gamma(b)$ does not contain illegal redexes. Finally we prove that weak normalisation of $\lambda(S)$ implies weak normalisation of $\lambda^\omega(S)$ if S is singly sorted.

Definition 5.1. Let S be a singly sorted specification.

We say that an abstraction $\lambda x{:}A.b$ is *illegal in the context* Γ if there exists D such that $\Gamma \vdash^\omega \lambda x{:}A.b : D$ and $\Gamma \not\vdash^\omega D : -$.

An abstraction is *legal* if it is typable in Γ and it is not illegal in Γ.

We say that $(\lambda x : A.b)a$ is an *illegal β-redex in* Γ if the abstraction $\lambda x : A.b$ is illegal in the context Γ.

Lemma 5.2. Let S be a singly sorted specification. An abstraction $\lambda x{:}A.b$ is legal in the context Γ if and only if there exists D such that $\Gamma \vdash^\omega \lambda x{:}A.b : D$ and $\Gamma \vdash^\omega D : s$.

Proof: ($\Rightarrow$) Obvious.

($\Leftarrow$) Since $\Gamma \vdash^\omega \lambda x{:}A.b : D$, the abstraction $\lambda x{:}A.b$ is typable in Γ. Suppose there exists D' such that $\Gamma \vdash^\omega \lambda x{:}A.b : D'$. By the uniqueness of types theorem we have that $D' =_\beta D$. Corollary 4.19 yields $\Gamma \vdash^\omega D' : s$. $\square$

We define the mapping φ that contracts the illegal redexes of a term.

Definition 5.3. We define $\varphi : \mathcal{C} \times \mathcal{T} \to \mathcal{T}$ as follows.

$$\begin{array}{rcl}
\varphi_\Gamma(x) & = & x \\
\varphi_\Gamma(s) & = & s \\
\varphi_\Gamma(a\ b) & = & \left\{ \begin{array}{ll} a_0[x := \varphi_\Gamma(b)] & \text{if } \varphi_\Gamma(a) = \lambda x{:}A.a_0 \text{ is an illegal} \\ & \text{abstraction in } \Gamma \\ & \\ (\varphi_\Gamma(a)\ \varphi_\Gamma(b)) & \text{otherwise} \end{array} \right. \\
\varphi_\Gamma(\lambda x{:}A.\ a) & = & (\lambda x{:}\varphi_\Gamma(A).\ \varphi_{\Gamma, x:A}(a)) \\
\varphi_\Gamma(\Pi x{:}A.\ B) & = & (\Pi x{:}\varphi_\Gamma(A).\ \varphi_{\Gamma, x:A}(B))
\end{array}$$

Sometimes we write $\varphi(a)$ instead of $\varphi_\Gamma(a)$.

Definition 5.4. We define $\varphi : \mathcal{C} \to \mathcal{C}$ as follows.

$$\begin{array}{rcl}
\varphi(\epsilon) & = & \epsilon \\
\varphi(\Gamma, x{:}A) & = & \varphi(\Gamma), x{:}\varphi_\Gamma(A)
\end{array}$$

The following lemma states that φ_Γ is the identity on $\lambda(S)$.

Lemma 5.5. Let S be a singly sorted specification. If $\Gamma \vdash a : A$ then $\varphi(\Gamma) = \Gamma$, $\varphi_\Gamma(a) = a$ and $\varphi_\Gamma(A) = A$.

The following lemma states that φ_Γ is a strategy for β-reduction.

Lemma 5.6. For all Γ, $a \twoheadrightarrow_\beta \varphi_\Gamma(a)$.

Lemma 5.7. Let S be a singly sorted specification. Suppose $\Gamma, x{:}A, \Gamma' \vdash^\omega b : B$ and $\Gamma \vdash^\omega a : A$. Then b is an illegal abstraction in $\Gamma, x{:}A, \Gamma'$ if and only if $b[x := a]$ is an illegal abstraction in $\Gamma, \Gamma'[x := a]$.

Proof: $(\Rightarrow)$. Suppose $b = \lambda y{:}C.\ d$ is an illegal abstraction in $\Gamma, x{:}A, \Gamma'$. It follows from lemma 3.7 and theorem 4.21 that $b[x := a]$ is an illegal abstraction in $\Gamma, \Gamma'[x := a]$.

$(\Leftarrow)$. Suppose that $b[x := a]$ is an illegal abstraction.

Suppose that b were not an abstraction. Then $b = x$ and $a = b[x := a] = \lambda y{:}C.\ d$ is a legal abstraction in Γ as $\Gamma \vdash^\omega A : s$. By lemmas 3.6 and 5.2, $b[x := a]$ is a legal abstraction in $\Gamma, \Gamma'[x := a]$.

Suppose that b is an abstraction but legal in $\Gamma, x{:}A, \Gamma'$. By lemmas 3.7 and 5.2 we have that $b[x := a]$ is a legal abstraction in $\Gamma, \Gamma'[x := a]$.

Therefore b is an illegal abstraction in $\Gamma, x{:}A, \Gamma'$.

□

Lemma 5.8. Let S be a singly sorted specification. If $\Gamma, x{:}A, \Gamma' \vdash^\omega b : B$ and $\Gamma \vdash^\omega a : A$ then $\varphi_{\Gamma, x:A, \Gamma'}(b)[x := \varphi_\Gamma(a)] = \varphi_{\Gamma, \Gamma'[x:=\varphi_\Gamma(a)]}(b[x := a])$.

Proof: This is proved by induction on the derivation of $\Gamma, x{:}A, \Gamma' \vdash^\omega b : B$. We prove the case that $b = (c\ d)$ is an application. By induction hypothesis we have that $\varphi(c)[x := \varphi(a)] = \varphi(c[x := a])$ and $\varphi(d)[x := \varphi(a)] = \varphi(d[x := a])$.

By weak subject reduction theorem, we have that $\varphi(c)$ is typable in $\Gamma, x{:}A, \Gamma'$ and $\varphi(a)$ is typable in Γ. Since $\Gamma \vdash^\omega A : s$,we have that $\Gamma \vdash^\omega \varphi(a) : A$. By lemma 5.7, $\varphi(c)$ is an illegal abstraction in $\Gamma, x{:}A, \Gamma'$ if and only if $\varphi(c[x := a])$ is an illegal abstraction in $\Gamma, \Gamma'[x := \varphi(a)]$.

It is easy to derive now that $\varphi(c\ d)[x := \varphi(a)] = \varphi((c\ d)[x := a])$.

The rest of the cases are easy to prove. □

The next example shows that if the specification is not singly sorted $\varphi(b[x := a])$ may not be syntactically equal to $\varphi(b)[x := \varphi(a)]$.

Example 5.9. The following specification is not singly sorted:

$$S \begin{array}{|lc|} \hline \boldsymbol{S} & 0, 1, 2 \\ \boldsymbol{A} & 0:1, 0:2, 1:2 \\ \boldsymbol{R} & (2, 2) \\ \hline \end{array}$$

We take $\Gamma = < x : 1, z : x >$ and $b = (\lambda y{:}x.y)z$. Note that b contains an illegal abstraction but $b[x := 0]$ does not. Hence $\varphi(b[x := 0]) \neq \varphi(b)[x := \varphi(0)]$.

In the following theorem, we prove that if e is typable in $\lambda^\omega(S)$ then $\varphi(e)$ does not contain illegal redexes when S is a singly sorted specification. The value $\varphi(e)$ is of the form $\lambda x_1{:}E_1.\ \ldots \lambda x_n{:}E_n.\ d$ with d and $E_1, \ldots E_n$ typable in $\lambda(S)$.

Theorem 5.10. (Preservation of the typing relation)
Let S be a singly sorted specification.
If $\Gamma \vdash^\omega e : E$, $n = \mathit{depth}(E)$ and $E = \Pi x_1{:}E_1 \ldots \Pi x_n{:}E_n.D$ then

$$\varphi(\Gamma, x_1{:}E_1 \ldots x_n{:}E_n) \vdash d : \varphi(D) \text{ with } \varphi(e) = \lambda x_1{:}\varphi(E_1) \ldots \lambda x_n{:}\varphi(E_n).d.$$

Proof: This property is proved by induction on the derivation of $\Gamma \vdash^\omega e : E$. We consider the cases of the abstraction, the application and the conversion rule.

- (abstraction) $\dfrac{\Gamma, x : A \vdash^\omega b : B}{\Gamma \vdash^\omega (\lambda x{:}A.\ b) : (\Pi x{:}A.\ B)}$.

  There are two possibilities, either $(\Pi x{:}A.\ B)$ is a toptype or not.

  1. Suppose $\Gamma \vdash^\omega (\Pi x{:}A.\ B) : s$. By the generation lemma we have that $\Gamma, x{:}A \vdash^\omega B : s_2$, $\Gamma \vdash^\omega A : s_1$ and $(s_1, s_2, s) \in \boldsymbol{R}$.
     By definition 4.4 and theorem 4.9, we have that $\mathit{depth}_{\Gamma, x:A}(B) = 0$. So by induction we have that $\varphi(\Gamma, x : A) \vdash \varphi(b) : \varphi(B)$.
     By lemma 5.6 and weak subject reduction theorem we deduce that $\varphi(\Gamma, x{:}A) \vdash^\omega \varphi(B) : s_2$. By lemma 3.5 we have that $\varphi(\Gamma, x{:}A) \vdash \varphi(B) : s'_2$ for some sort s'_2. Since $\lambda^\omega(S)$ is an extension of $\lambda(S)$, we have that $\varphi(\Gamma, x : A) \vdash^\omega \varphi(B) : s'_2$. By uniqueness of types, $s_2 = s'_2$.
     By lemma 5.6 and weak subject reduction theorem we deduce that $\varphi(\Gamma) \vdash^\omega \varphi(A) : s_1$. Since $\varphi(\Gamma), x{:}\varphi(A)$ is a legal context in $\lambda(S)$, we have that $\varphi(\Gamma) \vdash \varphi(A) : s'_1$ for some sort s'_1. Since $\lambda^\omega(S)$ is an extension of $\lambda(S)$, we have that $\varphi(\Gamma) \vdash^\omega \varphi(A) : s'_1$. By uniqueness of types, $s_1 = s'_1$.
     Since $s_1 = s'_1$ and $s_2 = s'_2$, we have that $(s'_1, s'_2, s) = (s_1, s_2, s) \in \boldsymbol{R}$.
     We obtain the following derivation:

     $$\dfrac{\varphi(\Gamma, x{:}A) \vdash \varphi(b) : \varphi(B) \quad \dfrac{\varphi(\Gamma) \vdash \varphi(A) : s_1 \quad \varphi(\Gamma, x{:}A) \vdash \varphi(B) : s_2}{\varphi(\Gamma) \vdash \varphi(\Pi x{:}A.\ B) : s}}{\varphi(\Gamma) \vdash \varphi(\lambda x{:}A.\ b) : \varphi(\Pi x{:}A.\ B)}$$

  2. Suppose $\Gamma \not\vdash^\omega (\Pi x{:}A.\ B) : -$. There are two possibilities, either B is a toptype or not.
     (a) Suppose $\Gamma, x{:}A \vdash^\omega B : s$. By definition 4.4 and theorem 4.9, we have that $\mathit{depth}_{\Gamma, x:A}(B) = 0$. By induction we have that:
     $$\varphi(\Gamma, x{:}A) \vdash \varphi(b) : \varphi(B)$$

where $\varphi(\lambda x{:}A.\ b) = \lambda x{:}\varphi(A).\ \varphi(b)$

(b) Suppose $\Gamma, x{:}A \not\vdash^{\omega} B : -$. It follows from induction that

$$\varphi(\Gamma, x{:}A, x_1{:}A_1 \ldots x_m{:}A_m) \vdash b' : \varphi(B')$$

$$\text{where} \quad \begin{array}{l} m = depth_{\Gamma, x:A}(B) \\ B = \Pi x_1{:}A_1 \ldots \Pi x_m{:}A_m.B', \\ \varphi(b) = \lambda x_1{:}\varphi(A_1) \ldots \lambda x_m{:}\varphi(A_m).b' \end{array}$$

- (application) $\dfrac{\Gamma \vdash^{\omega} b : (\Pi x{:}A.\ B) \quad \Gamma \vdash^{\omega} a : A}{\Gamma \vdash^{\omega} (b \ \ a) : B[x := a]}$.

  There are two possibilities: either $B[x := a]$ is a toptype in Γ or not.

  1. Suppose $\Gamma \vdash^{\omega} B[x := a] : s$.

     (a) Suppose $\Gamma \vdash^{\omega} (\Pi x{:}A.\ B) : s'$. So $depth_{\Gamma}(\Pi x{:}A.\ B) = 0$. By induction we have that
     $$\varphi(\Gamma) \vdash \varphi(b) : \varphi(\Pi x{:}A.\ B)$$
     By the generation lemma we have that $\Gamma \vdash^{\omega} A : s_1$. By induction we have that $\varphi(\Gamma) \vdash \varphi(a) : \varphi(A)$.
     Using the weak subject reduction theorem, we deduce that $\Gamma \vdash^{\omega} \varphi(b) : \Pi x{:}A'.\ B'$ and $\Gamma \vdash^{\omega} \Pi x{:}A'.\ B' : s'$ with $A \twoheadrightarrow_{\beta} A'$ and $B \twoheadrightarrow_{\beta} B'$. Hence $\varphi(b)$ is not an illegal abstraction in Γ and $\varphi(b\ a) = (\varphi(b)\ \varphi(a))$.
     Hence we have the following derivation:
     $$\frac{\varphi(\Gamma) \vdash \varphi(b) : \varphi(\Pi x{:}A.\ B) \quad \varphi(\Gamma) \vdash \varphi(a) : \varphi(A)}{\varphi(\Gamma) \vdash \varphi(b \ \ a) : \varphi(B)[x := \varphi(a)]}$$
     By lemma 5.8 we have that $\varphi(B)[x := \varphi(a)] = \varphi(B[x := a])$.

     (b) Suppose $\Gamma \not\vdash^{\omega} (\Pi x{:}A.\ B) : -$.
     By lemma 3.8 we have that $\Gamma \vdash^{\omega} A : s_1$. By induction we have that $\varphi(\Gamma) \vdash \varphi(a) : \varphi(A)$.
     By corollary 4.22 we have that $\Gamma, x{:}A \vdash^{\omega} B : s_2$. By induction we have that $\varphi(\Gamma, x{:}A) \vdash b' : \varphi(B)$ with $\varphi(b) = \lambda x{:}\varphi(A).b'$.
     We have that $\Gamma \vdash^{\omega} \lambda x{:}\varphi(A).b' : (\Pi x{:}\varphi(A).\ \varphi(B))$. By theorem 4.18 we have that $\Gamma \not\vdash^{\omega} (\Pi x{:}\varphi(A).\ \varphi(B)) : -$. Therefore $\varphi(b) = \lambda x{:}\varphi(A).b'$ is an illegal abstraction in the context Γ and $\varphi(b \ \ a) = b'[x := \varphi(a)]$.
     By the substitution lemma,
     $$\varphi(\Gamma) \vdash b'[x := \varphi(a)] : \varphi(B)[x := \varphi(a)]$$
     By lemma 5.8, we have that $\varphi(B)[x := \varphi(a)] = \varphi(B[x := a])$.

  2. Suppose $\Gamma \not\vdash^{\omega} B[x := a] : -$.
     It follows from lemma 4.7 that $\Gamma \not\vdash^{\omega} (\Pi x{:}A.\ B) : -$.

By lemma 3.8 we have that $\Gamma \vdash^{\omega} A : s_1$. By induction we have that $\varphi(\Gamma) \vdash \varphi(a) : \varphi(A)$ and that $\varphi(\Gamma, x : A, x_1 : A_1, \ldots, x_m : A_m) \vdash b' : \varphi(B_0)$ with $m = depth_{\Gamma,x:A}(B)$, $\varphi(b) = \lambda x{:}\varphi(A).\ \lambda x_1{:}\varphi(A_1) \ldots \lambda x_m{:} \varphi(A_m).b'$ and $B = \Pi x_1{:}A_1 \ldots \Pi x_m{:}A_m.\ B_0$.
Weak subject reduction theorem yields $\Gamma \vdash^{\omega} \varphi(b) : (\Pi x{:}A'.\ B')$ with $A \twoheadrightarrow_{\beta} A'$ and $B \twoheadrightarrow_{\beta} B'$. By theorem 4.18 we have that $\Gamma \not\vdash^{\omega} (\Pi x{:} A'.\ B') : -$. Therefore $\varphi(b)$ is an illegal abstraction and then the value of $\varphi(b\ a)$ is computed as follows.

$$\varphi(b\ a) = \lambda x_1{:}\varphi(A_1)[x := \varphi(a)] \ldots \lambda x_m{:}\varphi(A_m)[x := \varphi(a)].b'[x := \varphi(a)]$$

By lemmas 3.7 and 5.8 we have that $b'[x := \varphi(a)]$ has type $\varphi(B_0[x := a])$ in the context $\varphi(\Gamma), x_1{:}\varphi(A_1[x := a]), \ldots, x_m{:}\varphi(A_m[x := a])$.
By theorem 4.21, we have that $m = depth_{\Gamma}(B[x := a]) = depth_{\Gamma,x:A}(B)$.

- (conversion) $\dfrac{\Gamma \vdash^{\omega} b : B \quad \Gamma \vdash^{\omega} A : s \quad B =_{\beta} A}{\Gamma \vdash^{\omega} b : A}$

  By corollary 4.19, we have that $\Gamma \vdash^{\omega} B : s$. By induction we have that $\varphi(\Gamma) \vdash \varphi(b) : \varphi(B)$ and $\varphi(\Gamma) \vdash \varphi(A) : s$. After applying the conversion rule, we obtain $\varphi(\Gamma) \vdash \varphi(b) : \varphi(A)$.

The rest of the cases are easy to prove. □

We say that $\lambda(S)$ (or $\lambda^{\omega}(S)$) is *weakly normalising* if for all Γ, A, B such that $\Gamma \vdash A : B$ (or $\Gamma \vdash^{\omega} A : B$) then A and B have normal form.

Corollary 5.11. (Weak Normalisation) Let S be a singly sorted specification.
If $\lambda(S)$ is weakly normalising then so is $\lambda^{\omega}(S)$.

6 Syntax Directed Rules

In this section we define a syntax directed set of rules for any singly sorted pure type system. This system will be used in the following section to define a function that infers the type.

A set of rules is called *syntax directed* if the last rule in the derivation of the type of a term is determined by the structure of the term and of the context. The rules for pure type systems are not syntax directed since the last rule in the derivation can be the conversion or the weakening rules besides the corresponding structural rule. In order to make these sets of rules syntax directed, we should remove the non-structural rules (like the weakening and the conversion ones) and keep only the structural rules (the ones for term constructors). The system obtained by eliminating the non-structural rules should be equivalent to the original one. Therefore, the weakening rule is not removed but restricted to variables or constants and eventhough the conversion rule is removed, reduction or conversion is needed in the premises of almost all the rules. The equivalence (soundness and completeness) between the syntax directed set of rules and the original one for pure type systems is not easy to be proved. In order to be able to prove soundness and completeness we do not check the Π-condition in the same system but in a weaker one. The weaker system is the pure type system without the Π-condition.

First we recall the notion of weak head reduction. The *weak head β-reduction* is defined by the following rules.

$$(\lambda x{:}A.\ b)a \to_\beta^{wh} b[x := a] \qquad \frac{F \to_\beta^{wh} F'}{(F\ a) \to_\beta^{wh} (F'\ a)}$$

We define a typing relation $\vdash_{sd}^{\omega}$ whose rules are syntax directed and is proved to be 'equivalent' to the typing relation $\vdash^{\omega}$ for pure type systems without the Π-condition. We denote that $\Gamma \vdash A : C$ and $C \twoheadrightarrow B$ by $\Gamma \vdash A :\twoheadrightarrow B$ for an arbitrary typing relation $\vdash$ and a rewrite relation $\to$.

Definition 6.1. The typing relation $\vdash_{sd}^{\omega}$ is defined as the smallest relation on $\mathcal{C} \times \mathcal{T} \times \mathcal{T}$ closed under the following rules.

$$(axiom) \qquad \epsilon \vdash_{sd}^{\omega} s_1 : s_2 \qquad \text{for } (s_1, s_2) \in \boldsymbol{A}$$

$$(start) \qquad \frac{\Gamma \vdash_{sd}^{\omega} A :\twoheadrightarrow_\beta s}{\Gamma, x : A \vdash_{sd}^{\omega} x : A} \qquad \text{where } x \text{ is } \Gamma\text{-fresh}$$

$$(weakening) \qquad \frac{\Gamma \vdash_{sd}^{\omega} b : B \quad \Gamma \vdash_{sd}^{\omega} A :\twoheadrightarrow_\beta s}{\Gamma, x : A \vdash_{sd}^{\omega} b : B} \qquad \begin{array}{l}\text{where } x \text{ is } \Gamma\text{-fresh}\\ \text{and } b \in \mathbf{C} \cup V\end{array}$$

$$(formation) \qquad \frac{\Gamma \vdash_{sd}^{\omega} A :\twoheadrightarrow_\beta s_1 \quad \Gamma, x : A \vdash_{sd}^{\omega} B :\twoheadrightarrow_\beta s_2}{\Gamma \vdash_{sd}^{\omega} (\Pi x{:}A.B) : s_3} \qquad \text{for } (s_1, s_2, s_3) \in \boldsymbol{R}$$

$$(abstraction) \qquad \frac{\Gamma, x : A \vdash_{sd}^{\omega} b : B}{\Gamma \vdash_{sd}^{\omega} (\lambda x{:}A.\ b) : (\Pi x{:}A.\ B)}$$

$$(application) \qquad \frac{\Gamma \vdash_{sd}^{\omega} b :\twoheadrightarrow_\beta^{wh} (\Pi x{:}A.\ B) \quad \Gamma \vdash_{sd}^{\omega} a : A'}{\Gamma \vdash_{sd}^{\omega} (b\ \ a) : B[x := a]} \qquad A =_\beta A'$$

where $s \in \boldsymbol{S}$.

Since the conversion rule is removed, reduction or conversion is needed in the premises of almost all the rules. In the start, the weakening and the product rules, the types are reduced to some sort. In the application rule, the type of F is reduced to $(\Pi x{:}A.\ B)$ and the type A' of a should be convertible to A.

Note that these rules are syntax directed if the specification is singly sorted. If the specification is not singly sorted, given a context Γ and a term b, the type of b may not be unique and the last rule to be applied in the derivation of the type of b is not determined by the shape of b and Γ.

Lemma 6.2. If $\Gamma \vdash^{\omega} A : D$ and $D \twoheadrightarrow_{\beta} s$ then $\Gamma \vdash^{\omega} A : s$.

Proof: Suppose that $\Gamma \vdash^{\omega} D : s'$ for some s'. By the weak subject reduction theorem, $\Gamma \vdash^{\omega} s : s'$ and applying the conversion rule, we obtain $\Gamma \vdash^{\omega} A : s$.

If D is a toptype, we have that $D = s$. □

Theorem 6.3. (Soundness)
If $\Gamma \vdash^{\omega}_{sd} e : E$ then $\Gamma \vdash^{\omega} e : E$.

Proof: The proof proceeds by induction on the derivation of $\Gamma \vdash^{\omega}_{sd} e : E$. Only two cases are considered.

- (product) $\dfrac{\Gamma \vdash^{\omega}_{sd} A :\twoheadrightarrow_{\beta} s_1 \quad \Gamma, x : A \vdash^{\omega}_{sd} B :\twoheadrightarrow_{\beta} s_2}{\Gamma \vdash^{\omega}_{sd} (\Pi x{:}A.B) : s_3}$ for $(s_1, s_2, s_3) \in \boldsymbol{R}$.

  By induction hypothesis and lemma 6.2, we have $\Gamma \vdash^{\omega} A : s_1$ and $\Gamma, x : A \vdash^{\omega} B : s_2$. Hence $\Gamma \vdash^{\omega} (\Pi x{:}A.B) : s_3$.

- (application) $\dfrac{\Gamma \vdash^{\omega}_{sd} b :\twoheadrightarrow^{wh}_{\beta} (\Pi x{:}A.\ B) \quad \Gamma \vdash^{\omega}_{sd} a : A'}{\Gamma \vdash^{\omega}_{sd} (b \ \ a) : B[x := a]}$ with $A =_{\beta} A'$.

  By induction hypothesis, we have that $\Gamma \vdash^{\omega} b : D$ and $D \twoheadrightarrow^{wh}_{\beta} (\Pi x{:}A.\ B)$. Also $\Gamma \vdash^{\omega} a : A'$. There are two cases.

  1. Suppose that $\Gamma \vdash^{\omega} D : s$ for some s. By weak subject reduction, we have that $\Gamma \vdash^{\omega} (\Pi x{:}A.\ B) : s$. Generation lemma yields $\Gamma \vdash^{\omega} A : s'$ for some s'. Hence $\Gamma \vdash b : (\Pi x{:}A.\ B)$ and $\Gamma \vdash^{\omega} a : A$. So $\Gamma \vdash^{\omega} (b \ \ a) : B[x := a]$.
  2. Suppose that D is a toptype in Γ. By the description of toptypes theorem, we have that $D \in \mathcal{M}_{\Gamma}$ and so $D = \Pi x{:}A''.\ B''$ for some A'' and B''. The weak head normal form of D is D itself, so $A = A''$ and $B'' = B$.

□

The proof of completeness is straightforward.

Theorem 6.4. (Completeness)
If $\Gamma \vdash^{\omega} e : E$ then there exists E' such that $\Gamma \vdash^{\omega}_{sd} e : E'$ and $E =_{\beta} E'$.

Next we define a typing relation $\vdash_{sd}$ whose rules are syntax directed. We prove that if S is singly sorted then $\vdash_{sd}$ is 'equivalent' to the typing relation $\vdash$ for pure type systems. The idea is to use an auxiliary system to check for the Π-condition [vBJMP93] so that we can prove the 'equivalence' between $\vdash_{sd}$ and $\vdash$. In our case, the auxiliary system is the corresponding pure type system without the Π-condition.

Definition 6.5. The typing relation $\vdash_{sd}$ is the smallest relation on $\mathcal{C} \times \mathcal{T} \times \mathcal{T}$ closed under the same rules as in definition 6.1 except that the abstraction rule is replaced by the following one.

$$(abstraction) \quad \frac{\Gamma, x : A \vdash_{sd} b : B \quad \Gamma \vdash^{\omega}_{sd} (\Pi x{:}A.\ B) :\twoheadrightarrow_{\beta} s}{\Gamma \vdash_{sd} (\lambda x{:}A.\ b) : (\Pi x{:}A.\ B)}$$

where $s \in \boldsymbol{S}$.

Observe that in the abstraction rule we have the requirement that $(\Pi x{:}A.\ B)$ should be typable in $\vdash^{\omega}_{sd}$.

Theorem 6.6. (Soundness) Let S be a singly sorted specification.
If $\Gamma \vdash_{sd} e : E$ then $\Gamma \vdash e : E$.

Proof: We prove only the case of the abstraction rule.

$$\text{(abstraction)} \ \frac{\Gamma, x : A \vdash_{sd} b : B \quad \Gamma \vdash^{\omega}_{sd} (\Pi x{:}A.\ B) :\twoheadrightarrow s}{\Gamma \vdash_{sd} (\lambda x{:}A.\ b) : (\Pi x{:}A.\ B)}$$

By induction hypothesis we have that $\Gamma, x{:}A \vdash b : B$. By lemma 5.5, we have that $\varphi(\Gamma) = \Gamma$, $\varphi(A) = A$, $\varphi(b) = b$ and $\varphi(B) = B$. By theorem 6.3 and lemma 6.2 we have that $\Gamma \vdash^{\omega} (\Pi x{:}A.\ B) : s$. By theorem 5.10 we have that $\Gamma \vdash (\Pi x{:}A.\ B) : s$. $\square$

Theorem 6.7. (Completeness) If $\Gamma \vdash e : E$ then there exists E' such that $\Gamma \vdash_{sd} e : E'$ and $E =_{\beta} E'$.

Proof: We prove only the case of the abstraction rule.

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

By induction hypothesis we have that $\Gamma, x{:}A \vdash_{sd} b : B'$ for some B' such that $B =_{\beta} B'$. By the previous theorem (soundness) we have that $\Gamma, x{:}A \vdash b : B'$.

If $\Gamma \vdash (\Pi x{:}A.\ B) : s$ then there exists s_1 and s_2 such that $(s_1, s_2, s) \in \boldsymbol{R}$, $\Gamma \vdash A : s_1$ and $\Gamma, x{:}A \vdash B : s_2$. By correctness and unicity of types we have that $\Gamma, x{:}A \vdash B' : s_2$. Hence $\Gamma \vdash (\Pi x{:}A.\ B') : s$. By theorem 6.4 we have that $\Gamma \vdash^{\omega}_{sd} (\Pi x{:}A.\ B') : D$ and $D \twoheadrightarrow_{\beta} s$. $\square$

7 Type Inference

In this section we define a semi-algorithm of type inference for the class of singly sorted pure type systems. A semi-algorithm of type inference is a partial function or program that terminates and yields the type of a term if the term is typable and it may not terminate otherwise.

We cannot expect to find a terminating algorithm for the class of singly sorted pure type systems since typability (and also conversion) for some non-normalising pure type systems is not decidable [MR86] [CH94]. Hence we define a type inference semi-algorithm, for the class of singly sorted pure type systems, including the non-normalising ones.

In order to define such a semi-algorithm, we use the syntax directed set of rules presented in section 6. Although those rules are syntax directed when the specification is singly sorted, they are not yet deterministic. There are several conditions, called *side conditions*, that should be solved. For these conditions we have to verify if some element belongs to some of the sets in a specification, or we have to perform β-reduction or check if two types are β-convertible.

If the sets $\boldsymbol{S}$, $\boldsymbol{A}$ and $\boldsymbol{R}$ of the specification are not recursively enumerable (recursive) then the conditions $s \in \boldsymbol{S}$, $(s_1, s_2) \in \boldsymbol{A}$ and $(s_1, s_2, s_3) \in \boldsymbol{R}$ are not semi-decidable (decidable). In order to have a semi-algorithm we have to assume that these sets are recursively enumerable.

There are several ways of β-reducing or checking β-conversion. We should specify in which way we reduce and how we check if two types are convertible. We write semi-algorithms or partial functions to compute the weak head normal form and to check for β-conversion. The first one computes the weak head normal form of a term if it exists and it may not terminate otherwise. The second one checks if two terms are β-convertible. The termination of the semi-algorithm of type inference depends on the termination of these two functions.

Now we write a function $\mathbf{whnf} : \mathcal{T} \to \mathcal{T}$ that computes the weak head β-normal form if it exists. This function is a semi-algorithm, i.e. it may not terminate only in case the term is not weak head normalising.

$$\begin{aligned} \mathbf{whnf}(a) &= a \text{ if } a \text{ is in weak head normal form} \\ \mathbf{whnf}((\lambda x{:}A.\ b)\ a\ d_1\ \ldots d_n) &= \mathbf{whnf}(b[x := a]d_1 \ldots d_n) \end{aligned}$$

Note that a weak head normal form can be either an abstraction $(\lambda x{:}A.\ b)$, or a product $(\Pi x{:}A.\ B)$, or an application $(b\ a_1\ \ldots a_n)$ where b is a sort or a variable or a product.

Now we write a function $\mathbf{equal} : \mathcal{T} \times \mathcal{T} \to \mathbf{Bool}$ that checks if two terms are β-convertible. Two terms are β-convertible if the set of common-reducts is non-empty.

$$\mathbf{equal}(a, b) = \mathbf{comreduct}(\{a\}, \{b\}) \neq \emptyset$$

The function $\mathbf{comreduct} : \mathcal{P}(\mathcal{T}) \times \mathcal{P}(\mathcal{T}) \to \mathcal{P}(\mathcal{T})$ computes a set of common-reducts for subsets X, Y of $\mathcal{T}$.

$$\mathbf{comreduct}(X, Y) = \begin{cases} X \cap Y & \text{if } X \cap Y \neq \emptyset \\ \mathbf{comreduct}(Y, X \cup \mathcal{G}_{\to_\beta}(X)) & \text{otherwise} \end{cases}$$

where $\mathcal{G}_{\rightarrow_\beta}(a) = \{d \mid a \rightarrow_\beta d\}$

Note that Y takes the role of X in the else-part.

The function **equal** is a semi-algorithm, i.e. it may not terminate only in case the terms are not β-convertible.

The function **equal** can be improved. In [Coq91] (see also [Mag94]) a function to check $\beta\eta$-conversion is defined on normalising terms. The idea is to compute the weak head normal forms of the terms and compare their heads. In spite of being efficient, it gives a set of common-reducts only if both terms are normalising. In [Sev96], other functions to check β-conversion are defined and compared.

Now we define a function $\mathbf{type}_S : \mathcal{C} \times \mathcal{T} \rightarrow \mathcal{T}_\perp$ (or just **type**) that computes the type of a term (up to β-conversion) in a singly sorted pure type system. This function is obtained from the syntax directed set of rules defined in section 6 for pure type systems.

$$
\begin{array}{lcll}
\mathbf{type}(\epsilon, s) & = & s' & \text{if } (s, s') \in \boldsymbol{A} \\[1ex]
\mathbf{type}(<\Gamma, x : A>, x) & = & A & \text{if } \mathbf{whnf}(\mathbf{type}(\Gamma, A)) \in \boldsymbol{S} \\
 & & & \text{ and } x \text{ is } \Gamma\text{-fresh} \\[1ex]
\mathbf{type}(<\Gamma, x : A>, b) & = & \mathbf{type}(\Gamma, b) & \text{if } b \in \boldsymbol{S} \cup V,\ b \neq x\ x \text{ is } \Gamma\text{-fresh and} \\
 & & & \mathbf{whnf}(\mathbf{type}(\Gamma, A)) \in \boldsymbol{S} \\[1ex]
\mathbf{type}(\Gamma, (\Pi x{:}A.B)) & = & s_3 & \text{if } \mathbf{whnf}(\mathbf{type}(\Gamma, A)) = s_1, \\
 & & & \mathbf{whnf}(\mathbf{type}(<\Gamma, x : A>, B)) = s_2, \\
 & & & \text{and } (s_1, s_2, s_3) \in \boldsymbol{R} \\[1ex]
\mathbf{type}(\Gamma, (\lambda x{:}A.\ b)) & = & (\Pi x{:}A.\ B) & \text{if } \mathbf{type}(<\Gamma, x : A>, b) = B \text{ and} \\
 & & & \mathbf{whnf}(\mathbf{type}^\omega(\Gamma, \Pi x{:}A.\ B)) \in \boldsymbol{S} \\[1ex]
\mathbf{type}(\Gamma, (b\ \ a)) & = & B[x := a] & \text{if } \mathbf{whnf}(\mathbf{type}(\Gamma, b)) = (\Pi x{:}A.\ B), \\
 & & & \mathbf{type}(\Gamma, a) = A' \text{ and } \mathbf{equal}(A, A') \\[1ex]
\mathbf{type}(\Gamma, a) & = & \perp & \text{otherwise}
\end{array}
$$

The function $\mathbf{type}^\omega_S : \mathcal{C} \times \mathcal{T} \rightarrow \mathcal{T}_\perp$ is defined exactly like **type** except for the case that corresponds to the abstraction rule. In that case, the condition $\mathbf{whnf}(\mathbf{type}^\omega(\Gamma, \Pi x{:}A.\ B)) = s \in \boldsymbol{S}$ is removed. The function $\mathbf{type}^\omega$ computes the type of a term in a pure type system without the Π-condition.

The first part of the following theorem says that if the term has a type in a singly sorted pure type system then this is computed by **type** (up to β-conversion). The second part says that the value $\mathbf{type}(\Gamma, a) \neq \perp$ is the type of a in Γ in a singly sorted pure type system.

Theorem 7.1. (Correctness of 'type') Let $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ be singly sorted such that the sets $\boldsymbol{S}$, $\boldsymbol{A}$ and $\boldsymbol{R}$ are recursively enumerable.

1. If $\Gamma \vdash a : A$ then $\mathbf{type}(\Gamma, a)$ terminates and $\mathbf{type}(\Gamma, a) =_\beta A$.
2. If $\mathbf{type}(\Gamma, a)$ terminates and yields A then $\Gamma \vdash a : A$.

Type checking can be solved from type inference. Using the function **type** that infers the type of a term, we write another function $\mathbf{check} : \mathcal{C} \times \mathcal{T} \times \mathcal{T} \to Bool$ that checks if a term has certain type.

$$\mathbf{check}(\Gamma, a, A) = \mathbf{whnf}(\mathbf{type}(\Gamma, A)) \in \boldsymbol{S} \text{ and } \mathbf{equal}(\mathbf{type}(\Gamma, a), A)$$

We have the following conclusion which has been proved before in [vBJMP93] and [Pol96].

Theorem 7.2. (Decidability of Type Inference and Type Checking) Let $S = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ be singly sorted such that the sets $\boldsymbol{S}$, $\boldsymbol{A}$ and $\boldsymbol{R}$ are recursive. If $\lambda(S)$ is β-weakly normalising then type inference and type checking in $\lambda(S)$ are decidable.

Proof: Suppose that $\lambda(S)$ is normalising. Then **whnf** and **equal** are applied to normalising terms and hence they terminate. Since the sets of the specification are recursive, we have that $\mathbf{type}^\omega$ terminates and so do **type** and **check**. Therefore type inference and type checking for $\lambda(S)$ are decidable. $\square$

8 Conclusions

We think that the method we have followed in this paper, i.e. the consideration of the system without the Π-condition as an auxiliary system, can be easily adapted to other extension of pure type systems to solve the problem of type inference. In [Sev96], the method has been applied to the extension of pure type systems with definitions. In that case, we have considered pure type systems with definitions and without the Π-condition as an auxiliary system.

Several syntax directed sets of rules for pure type systems are studied in [vBJMP93]. Our definition of a syntax directed set of rules follows the idea in [vBJMP93] of using an auxiliary system to check for the Π-condition. In that paper, the auxiliary system is much weaker than $\lambda(S)$. In our case, the auxiliary system is $\lambda^\omega(S)$, which is very close to $\lambda(S)$ (in some cases like for the extended calculus of constructions both systems coincide). Moreover, $\lambda^\omega(S)$ preserves some properties of $\lambda(S)$ like normalisation.

In [Pol93] a syntax directed set of rules for bijective pure type systems is presented. The class of bijective pure type systems includes all systems of the λ-cube and is a proper subclass of the class we study here, the class of singly sorted pure type systems. The class of bijective pure type systems does not include any of the systems of the family of AUTOMATH as described on page 216 and 217 in [Bar92].

Decidability of type checking for normalising pure type systems whose set of sorts is *finite* is proved in [vBJ93]. In that paper, a type inference algorithm is defined that computes the normal form in all the rules. A discussion on the side-conditions can be found in [Pol96]. In this paper, decidability of type inference is proved for normalising pure type systems that are either singly sorted or semi-full under the assumption that the sets forming the specification are *recursive*. In theorem 7.2, we have given a new proof of the same result for the singly sorted but not for the semi-full pure type systems.

References

[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.

[CH94] T. Coquand and H. Herbelin. A-translation and looping combinators in pure type systems. *Journal of Functional Programming*, 4(1):77–88, 1994.

[Coq91] T. Coquand. An algorithm for testing conversion in type theory. In *Logical Frameworks*, 1991.

[Mag94] L. Magnusson. *The Implementation of Alf-a proof editor based on Martin-Löf's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Chalmers University of Technology, 1994.

[MR86] A. R. Meyer and M. B. Reinhold. 'type' is not a type: Preliminary report. In *Principles of Programming Languages*, pages 287–295. ACM, 1986.

[Pol93] E. Poll. A type checker for bijective pure type systems. Computing Science Note CS-9322, Eindhoven University of Technology, 1993.

[Pol96] R. Pollack. A Verified Type Checker. In B. Bjerner, M. Larsson, and B. Nordström, editors, *Proceedings of the 7th Nordic Workshop on Programming Theory*, 86. Göteborg University and Chalmers University of Technology, 1996.

[Sev96] P. Severi. *Normalisation on Lambda Calculus and its relation to Type Inference.* PhD thesis, Eindhoven University of Technology, 1996.

[vBJ93] B. van Benthem Jutting. Typing in pure type systems. *Information and Computation*, 105:30–41, 1993.

[vBJMP93] B. van Benthem Jutting, J. McKinna, and R. Pollack. Checking algorithms for pure type systems. In H. Barendregt and T. Nipkov, editors, *Types for Proofs and Programs*, volume 806, 1993.