JFP **13** (2): 339–414, March 2003. © 2003 Cambridge University Press
DOI: 10.1017/S0956796802004641 Printed in the United Kingdom

Termination of rewriting in the Calculus of Constructions

DARIA WALUKIEWICZ-CHRZĄSZCZ*
(*e-mail:* daria@mimuw.edu.pl)
Institute of Informatics, Warsaw University, ul. Banacha 2,
02-097 Warszawa, Poland
and
Laboratoire de Recherche en Informatique, Université de Paris-Sud,
Bât 490, 91405 Orsay cedex, France

Abstract

We show how to incorporate rewriting into the Calculus of Constructions and we prove that the resulting system is strongly normalizing with respect to beta and rewrite reductions. An important novelty of this paper is the possibility to define rewriting rules over dependently typed function symbols. We prove strong normalization for any term rewriting system, such that all function symbols satisfy the, so called, star dependency condition, and every rule is accepted by the Higher Order Recursive Path Ordering (which is an extension of the method created by Jouannaud and Rubio for the setting of the simply typed lambda calculus). The proof of strong normalization is done by using a typed version of reducibility candidates due to Coquand and Gallier. Our criterion is general enough to accept definitions by rewriting of many well-known higher order functions, for example dependent recursors for inductive types or proof carrying functions. This makes it a very good candidate for inclusion in a proof assistant based on the Curry-Howard isomorphism.

1 Introduction

The aim of this paper is to incorporate a rewriting mechanism into the Calculus of Constructions and to present a method for proving strong normalization of the combined calculus.

Practical motivations of our work concern proof assistants based on the Calculus of Constructions and their extension with rewriting-style function definitions. Defining functions by rewriting, instead of `Fix` and `Cases` (if we take Coq (Barras *et al.*, 1999) as an example), is not only easier but also more powerful, as it can enrich significantly the conversion relation of the system. And the richer the conversion relation is, the more compact and easier to automate the proofs in the system are.

In this respect, our paper contributes to the work on the separation between deduction and computation. Computations are steps that can be mechanically performed and reproduced (like rewriting), and therefore there is no need to keep

* Partly supported by KBN Grant 7 T11C 028 20

them in the proof (they are just hidden in conversion). On the other hand, the reasoning consists of conscious choices of the logical inference rules to be applied.

When combining rewriting with λ-calculus, one has to address the question of mutual relationship of user-defined rewriting and the beta reduction rule. One solution is to treat both equally, using first order (syntactic) matching to decide if a rewrite rule can be applied to a given term, and to consider the reduction relation of the system to be the sum of beta and rewrite relations (Jouannaud & Okada, 1991; Breazu-Tannen & Gallier, 1991). Other approaches consist of applying rewrite rules only to terms in beta normal form (Nipkow, 1991; Loría-Sáenz & Steinbach, 1992; Jouannaud & Rubio, 1998), or using higher order matching (modulo beta) to decide if a rewrite rule can be applied (Klop, 1980; Klop *et al.*, 1993). In our paper we consider higher order rewriting with syntactic matching.

Incorporating rewriting to the Calculus of Constructions is not straightforward, because the definition of rewriting and the definition of the typing system are mutually dependent. Once this problem solved, there are crucial meta-theoretical properties to be studied: normalization, confluence and logical consistency. Our paper focuses on the strong normalization property. To address this question one has to give a termination criterion, that is satisfied by a set of rewriting rules only if the relation generated by these rules and the beta rule is terminating. Ideally, this criterion should be an automatic decision procedure, running in reasonable time and strong enough to accept the rules which are already known to be terminating, like the elimination rules for inductive types (also called recursor rules).

Few techniques are known for proving strong normalization of rewriting in the higher-order setting. Strict functionals presented in van de Pol & Schwichtenberg (1995) are a very powerful method, but a lot of user interaction is subsequently needed for proving their properties when used as interpretations. In general, this concerns also other semantical methods of proving termination.

Syntactical methods are meant to be more practical and to be the base of (partially) automated termination techniques. Avenhaus & Loria-Sáenz (1995) initiated ordering based syntactical methods in the context of simply-typed lambda calculus. Although Lysne & Piris (1995) improves over Avenhaus & Loria-Sáenz (1995), the resulting method is still rather weak.

The General Schema was originally defined in Jouannaud & Okada (1991) in the context of simply typed lambda calculus (see also Jouannaud & Okada (1997)). It was later used in Barbanera *et al.* (1994, 1997), where higher-order rewriting in the Calculus of Constructions was introduced for the first time. The authors restricted themselves to some predefined, basic types and function symbols of algebraic types. The latter result was further extended in Blanqui *et al.* (1999) by strengthening the General Schema and adding a powerful mechanism, called the Computable Closure, designed to use the information offered by Girard's reducibility candidates (the method on which termination proofs are based in this context). This mechanism allowed to take care of complex rules such as recursor rules for Brouwer's ordinals. Very recently, Blanqui (2001) generalized the General Schema to the rewriting on types.

Another method for proving strong normalization of higher-order rewriting is the Higher Order Recursive Path Ordering (HORPO), which is an extension of the first order RPO (Dershowitz, 1982) to higher-order terms. This reduction ordering was presented in Jouannaud & Rubio (1999), in the context of the simply typed lambda calculus. Due to the recursive structure of the HORPO, this method is inherently more powerful than the General Schema, although both methods bear some similarities (like the use of the computable closure).

In this paper, we extend HORPO to the typing discipline of the Calculus of Constructions. We consider a Calculus of Constructions with constants, representing type constructors, constructors and function symbols, and we use (an extended version of) HORPO to control rewriting. Type constructors and constructors are used to build inductive types, function symbols serve to define rewriting. For technical reasons every function symbol has two kinds of arguments: parameters and other arguments. The division of arguments is correct if it satisfies the so called star dependency condition. The division has also an influence on the form of the rules accepted by HORPO. The star dependency condition and its interaction with HORPO are the most important concepts of this paper.

Compared to Blanqui (2001), in our system all constants may have polymorphic and dependent types and consequently we can handle rewriting of dependently typed function symbols. By means of type constructors and constructors we can define small strictly positive inductive types. There is a technical restriction imposed on the form of the types of constructors, but this restriction does not seem to exclude natural examples we have seen so far. Elimination rules for inductive types are presented as rewriting rules and they are meant to be accepted by HORPO. For the moment we cannot accept elimination rules for inductive types with functional arguments (like Brouwer's ordinals) but we hope to do this very soon (so from the elimination rule point of view, we accept only "basic" inductive types). It is important to stress that, just like all previously mentioned papers (except of Blanqui (2001)), we consider only object level rewriting, that is rewriting that does not create types (strong elimination is out of the scope of this paper).

The proof of strong normalization of our calculus is done by using a typed version of the reducibility candidates method due to Coquand & Gallier (1990).

Before we go into details, there are two more points to be explained. The first one concerns the status of inductive types. There are no inductive types in the Calculus of Constructions (contrary to the Calculus of Inductive Constructions (Coquand & Paulin-Mohring, 1990)), which is the starting point of our work. As we have already said, in our calculus, we are able to simulate inductive types and their elimination rules by imposing some positivity conditions on types of constructors and by using rewriting. But inductive types are not necessary to define rewriting in the Calculus of Constructions. One can perfectly imagine the system without the conditions mentioned above, where constructors are just function symbols that do not rewrite. We decided to present the version with inductive types, that is with positivity conditions on types of constructors, because these conditions allow us to accept some rules that would not be accepted otherwise. Accounting for inductive types turned out to be a non-trivial task.

The second point to be discussed concerns the system for which we prove strong normalization. Instead of proving strong normalization for every set of rules accepted by HORPO, we do it for HORPO itself. More precisely, we mix together the Calculus of Constructions with the rewrite relation generated by all valid HORPO judgments and show that the resulting calculus is strongly normalizing. Termination for a particular rewrite system accepted by HORPO follows easily.

The framework we use is described in section 2. Section 3 gives the definition of HORPO and CC+H, which is the system resulting from mixing the Calculus of Constructions with the rewriting relation induced by HORPO. We study part of the meta-theory of CC+H in section 4 and continue in section 5 with the proof of strong normalization. We use this fact to show in section 6 the strong normalization of any CC+R – a Calculus of Constructions with rewriting generated by user-defined rules verifying HORPO. Section 7 provides a discussion on the star dependency condition and its importance for the proof of strong normalization. In section 8 we present several examples illustrating the power and deficiencies of HORPO. Two problems, important from the practical point of view, are discussed in section 9: decidability and modularity. We conclude with the discussion of restrictions we have imposed and potential improvements.

The reader is expected to be familiar with the basics of term rewriting systems (Dershowitz & Jouannaud, 1990) and typed lambda calculi (Barendregt, 1990; Barendregt, 1993), and in particular with the Calculus of Constructions (Coquand & Huet, 1988).

The preliminary version of this paper was presented at the *Workshop on Logical Frameworks and Meta-languages* in Santa Barbara, California, in July 2000.

2 Preliminaries

The calculus that we present in this paper is constructed upon the set of symbols and the typing information provided by the user. This section will explain these requirements and introduce the basic notions used in the rest of the paper.

The section is organized as follows. From a given set of symbols we construct pseudoterms in section 2.1. Then, we recall the definition of the standard Calculus of Constructions (section 2.2). We continue with conditions imposed on types that are assigned to symbols: star dependency condition in section 2.4.1, strict positivity in section 2.4.2 and typing conditions in section 2.4.3. The last part of this section introduces basic definitions concerning relations and orderings.

2.1 Terms

We assume given two disjoint sets of variables: one for small variables ($\mathcal{V}ar^{\star}$) and one for big variables ($\mathcal{V}ar^{\square}$) and three disjoint sets of constants: for type constructors ($\mathcal{TC}$), constructors ($\mathcal{CS}$) and function symbols ($\mathcal{F}$).

Function and constructor symbols will have their arguments classified as *parameters* and other arguments. Consequently, their arities will be pairs of natural numbers. We write $\mathcal{F}^{(r,n)}$ ($\mathcal{CS}^{(r,n)}$) for the set of function (constructor) symbols taking first r parameters and then n other arguments. We simply write $f \in \mathcal{F}^m$ ($c \in \mathcal{CS}^m$) with $m = r + n$, in case the knowledge of r is not needed. An arity of a type constructor is always a natural number. We write $s \in \mathcal{TC}^m$ if a type constructor s is of arity m.

Definition 2.1.1 (*Pseudoterms*)
Pseudoterms are built from the usual constructions and from the constants, respecting the arity. Below, s denotes a type constructor, c a constructor and f a function symbol.

$$a ::= x \mid \star \mid \Box \mid \lambda x : a.a \mid (x : a).a \mid a\, a \mid s(a_1, \dots a_n) \mid c(a_1, \dots a_n) \mid f(a_1, \dots a_n)$$

The pseudoterm $u\, v$, written also $@(u, v)$, denotes the application of u to v. In the sequel we will also use terms of the form $@(u, v_1, \dots v_n)$. If $a = (\dots((u\, v_1)v_2)\dots v_n)$ then $@(u, v_1, \dots v_n)$ is a *left-flattening* of a. Note that u may still be an application. The pseudoterm $(x : u).v$ denotes a product (traditionally, it is written $\Pi x : u.v$). If x does not occur in v, we write it also as $u \to v$. Throughout the paper we will often write $f(\vec{a}, \vec{b})$ to denote a term headed by a symbol $f \in \mathcal{F}^{(r+n)}$ applied to r parameters $\vec{a}$ and n other arguments $\vec{b}$. The same applies to $c \in \mathcal{CS}^{(r,n)}$.

Pseudoterms are identified with finite labeled trees by considering λx and Πx as binary function symbols different for every x. Positions are finite sequences of natural numbers and we use Λ for the root position. The *subterm* of a pseudoterm a at position p is denoted by $a|_p$ and the pseudoterm obtained by replacing $a|_p$ by b is written $a[b]_p$. The strict subterm relation (when $p \neq \Lambda$) is denoted by $\rhd$ and its nonstrict version by $\unrhd$. A subterm at position $p \neq \Lambda$ is called *constructor subterm* if all symbols on the path from the root down to but not including p are constructors. We write $a \rhd b$ if b is a subterm of a and $a \rhd^{CS} b$ if b is a constructor subterm of a.

We denote by $\mathcal{FV}(a)$ the set of free variables of a. This notation easily extends to sets: $\mathcal{FV}(A) = \bigcup_{a \in A} \mathcal{FV}(a)$. We assume that bound variables in a pseudoterm are all different, and are different from the free ones. As usual, we work *modulo* *α-conversion* that is identifying the terms that only differ in their bound variables. The notation $\vec{a}$ denotes a list of pseudoterms, but it will sometimes be used for multisets.

Substitutions are written as in $[a_1/x_1, \dots a_n/x_n]$ or $[\vec{a}/\vec{x}]$. The set $\{x_1, \dots x_n\}$ is called the domain (dom) of the substitution. Substitutions expand to applications defined on all pseudoterms, by replacing each free variable by its image, and avoiding captures. We use letters θ, ρ, μ for substitutions and postfix notation for their application to pseudoterms.

Given two substitutions ρ_1 and ρ_2 with the domains D_1 and D_2, a *parallel composition of* ρ_1 *and* ρ_2, denoted by $\rho_1 \cup \rho_2$, is a substitution with the domain $D_1 \cup D_2$ defined by

$$x(\rho_1 \cup \rho_2) = \begin{cases} x\rho_1 & \text{if } x \in D_1 \\ x\rho_2 & \text{if } x \in D_2 \end{cases}$$

and provided that $D_1 \cap D_2 = \emptyset$.

A *sequential composition*, denoted by $\rho_1\rho_2$, is a substitution defined by

$$x(\rho_1\rho_2) = (x\rho_1)\rho_2$$

A *declaration* is a pair $x : a$ where x is a variable and a is a pseudoterm. An *environment* Γ is an ordered sequence of declarations, where all variables are distinct. For $\Gamma = x_1 : a_1, \dots, x_n : a_n$ the domain of Γ ($dom(\Gamma)$) is the list $x_1, \dots, x_n$, and $\Gamma(x_i) = a_i$. A *sequent* is a pair (Γ, a) consisting of an environment and a pseudoterm. We will write sequents as $\Gamma \vdash a$, since they will be used in situations where a is a well-typed term in the environment Γ in a given typing system.

The sequence of declarations $x_1 : a_1, \dots, x_n : a_n$ may be denoted by $\overrightarrow{x : a}$, if n is irrelevant or clear from the context. Vectors of declarations usually serve to denote environments, but we will sometimes use them to form a long product (shortening $(x_1 : b_1)\dots(x_n : b_n).c$ by $(\overrightarrow{x : b}).c$). By convention, the type $(\overrightarrow{x : b}).c$ is simply c if $(\overrightarrow{x : b})$ is empty. For simplicity, we will also write $(\Gamma).c$ for the product $(\overrightarrow{x : b}).c$ if Γ is the environment $x_1 : b_1, \dots x_n : b_n$.

2.2 Calculus of Constructions

The Calculus of Constructions (CC), originally defined in Coquand & Huet (1988), is the starting point of our work. The typing relation of this system will be used directly to check the types of constants from $(\mathcal{F} \cup \mathcal{CS} \cup \mathcal{TC})$ and to check the types of terms in HORPO (Definition 3.4.1) and computable closure (Definition 3.3.2).

Definition 2.2.1 (*Calculus of Constructions*)

$$\textbf{(ax)} \quad \frac{}{\vdash_{CC} \star : \Box}$$

$$\textbf{(var)} \quad \frac{\Gamma \vdash_{CC} a : p}{\Gamma, x : a \vdash x : a} \qquad (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{(weak)} \quad \frac{\Gamma \vdash_{CC} a : b \qquad \Gamma \vdash_{CC} c : p}{\Gamma, x : c \vdash_{CC} a : b} \qquad (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{(abs)} \quad \frac{\Gamma, x : a \vdash_{CC} b : c \qquad \Gamma \vdash_{CC} (x:a).c : p}{\Gamma \vdash_{CC} \lambda x:a.b : (x:a).c} \qquad (x \notin dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{(app)} \quad \frac{\Gamma \vdash_{CC} a : (x:b).c \qquad \Gamma \vdash_{CC} d : b}{\Gamma \vdash_{CC} a\,d : c[d/x]}$$

$$\textbf{(prod)} \quad \frac{\Gamma \vdash_{CC} a : p \qquad \Gamma, x : a \vdash_{CC} b : q}{\Gamma \vdash_{CC} (x:a).b : q} \qquad (x \notin dom(\Gamma), p, q \in \{\star, \Box\})$$

$$\textbf{(conv)} \quad \frac{\Gamma \vdash_{CC} a : b \qquad \Gamma \vdash_{CC} b' : p}{\Gamma \vdash_{CC} a : b'} \qquad (p \in \{\star, \Box\}, b \overset{*}{\rightarrow}_\beta b' \text{ or } b' \overset{*}{\rightarrow}_\beta b)$$

The relation which occurs in the **(conv)** rule is the beta reduction. It is the reflexive and transitive closure of $\rightarrow_\beta$, which is defined as the compatible closure of the β-reduction rule: $\lambda x:a.b\,c \rightarrow b[c/x]$.

The Calculus of Constructions is confluent and terminating with respect to the beta reduction. As a consequence, the type-checking in this calculus is decidable.

If we want to type-check terms that are headed by a constant (like $f(\vec{a})$) we must add a new rule for it. But first we will provide all constants with types and check that these types are well-typed terms in the Calculus of Constructions.

By *signature* we mean a list of declarations assigning a type (a pseudoterm) to every constant. Because of possible dependencies these types may use constants, for example in the declaration $f : (P : nat \to \star)(n : nat).P(g(n))$ the type of f uses constants nat and g.

Definition 2.2.2
We say that the signature Σ is *well-typed* if every constant of arity m has a type of the form $(x_1 : A_1)\dots(x_m : A_m).B$ and if $\Sigma \vdash_{CC} \star : \Box$ in the original Calculus of Constructions, treating $f(\vec{a})$ like a variable f applied successively to terms $a_1, \dots a_n$.

Given a well-typed signature we can construct the system which apart from the seven usual rules of the Calculus of Constructions contains also:

$$\textbf{(fun)}\ \frac{\Gamma \vdash_{CC} \star : \Box \qquad \forall\, i \quad \Gamma \vdash_{CC} a_i : A_i[\vec{a}/\vec{x}]}{\Gamma \vdash_{CC} f(\vec{a}) : B[\vec{a}/\vec{x}]} \qquad f : \overrightarrow{(x : A)}.B \in \Sigma$$

Since $\Sigma \vdash_{CC} \star : \Box$, the extended system may be seen as the original one where some variables (representing constants of arity m) are always applied to at least m arguments.

Of course this extended version of the Calculus of Constructions is still confluent and terminating with respect to the beta reduction. Consequently the type-checking in this system is decidable. From now on we will refer to it as the Calculus of Constructions and we will call the original one the pure Calculus of Constructions or the Calculus of Constructions without Constants.

In section 3 we will define CC+H – an extension of the Calculus of Constructions by rewriting. To distinguish between the standard and the new systems we will always use the subscript CC, whenever talking about the typing relation of the Calculus of Constructions (like in $\Gamma \vdash_{CC} a : b$). Following the notation used further for CC+H, where the reduction used in the conversion rule needs an environment, we may also write $\Gamma \vdash_{CC} b \to_\beta b'$ for the beta reduction $b \to_\beta b'$ and $\Gamma \vdash_{CC} b \overset{*}{\leftrightarrow} b'$ for the fact that b and b' are beta equal ($\Gamma \vdash b \overset{*}{\leftrightarrow} b'$ will mean that b and b' are convertible in CC+H).

2.3 *Extended signature*

In the previous section the signature was just a list of declarations. For our purposes, it is more convenient to put together the information about a type constructor and its constructors and have a single inductive definition.

Definition 2.3.1 (Extended signature)
An *extended signature* Σ is a (possibly dependent) sequence of declarations, each declaration being of the form:

- $f : \overrightarrow{(x : b)}.c$, where $f \in \mathcal{F}$ and $\vec{b}, c$ are pseudoterms, or

- $Ind[p_1 : P_1, \dots p_r : P_r](s : A := c_1 : C_1, \dots c_m : C_m)$, where $s \in \mathcal{TC}$, $c_1, \dots c_m \in \mathcal{CS}$ (constructors of the inductive type s) and $A, \vec{C}$ are pseudoterms. Variables $p_1, \dots p_r$ are called parameters of the inductive type s.

Moreover, we suppose that every $e \in \mathcal{F} \cup \mathcal{TC} \cup \mathcal{CS}$ has exactly one declaration in Σ.

The type of an $e \in \mathcal{TC} \cup \mathcal{CS}$ is not explicitly mentioned in the extended signature. To obtain it, we have to destructure the inductive definition which includes this e.

$$\textbf{(type constructor)} \quad \frac{Ind[p_1 : P_1, \dots p_r : P_r](s : A := c_1 : C_1, \dots c_m : C_m) \in \Sigma}{s : (p_1 : P_1) \dots (p_r : P_r).A \in \Sigma}$$

$$\textbf{(constructor)} \quad \frac{Ind[p_1 : P_1, \dots p_r : P_r](s : A := c_1 : C_1, \dots c_m : C_m) \in \Sigma}{c_i : (p_1 : P_1) \dots (p_r : P_r).C_i[s(p_1, \dots p_r)/\overline{s}] \in \Sigma}$$

where $\overline{s}$ is a special variable corresponding to the constant s and appearing in C_i's.

Example 1
Inductive definition of polymorphic lists has the form:

$$Ind[A : \star](List : \star := nil : \overline{List}, cons : A \to \overline{List} \to \overline{List})$$

From this definition $List : \star \to \star$, $nil : (A : \star).List(A)$ and $cons : (A : \star)(a : A)(l : List(A)).List(A)$.

Every extended signature Σ may be seen as a simple signature assigning types to constants and such that the declaration for a type constructor precedes the declarations for its constructors.

2.4 Well-formedness conditions

In this section we will present all necessary conditions that must be satisfied by constants and their types (i.e. by the signature) and we will summarize them in the definition of well-formedness (Definition 2.4.3). From the beginning of the next section (section 3) we will assume that we are given a well-formed extended signature.

2.4.1 Star Dependency

The star dependency condition is one of the most important concepts of this paper. The types of function and constructor symbols provided by the user are checked against this condition in order to make sure that the separation of arguments (into parameters and other arguments) is correct.

For function symbols, star dependency condition influences the form of the rewriting rules accepted by HORPO, restricting parameter arguments to be distinct variables (see sections 3.1 and 3.4). This ensures the computability of the type arguments of a function symbol, which is crucial in the proof of strong normalization. For constructors, star dependency makes it possible to define the interpretation of inductive types in section 5.2.1. For a deeper discussion of the star dependency condition we refer the reader to section 7.

Definition 2.4.1
The type $(p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c$ satisfies the *star dependency condition* with respect to $f \in (\mathcal{F}^{(r,n)} \cup \mathcal{CS}^{(r,n)})$ if

$$\mathcal{FV}(b_1,\dots b_n, c) \cap \mathcal{V}ar^{\Box} \subseteq \{p_1,\dots p_r\}$$

In other words, this condition means that every free variable y of $b_1,\dots b_n, c$ is small $(y : A : \star)$ or is a parameter. This condition may be seen as a generalization of the requirement that all variables corresponding to nonparameter arguments, $x_1,\dots x_n$ belong to $\mathcal{V}ar^{\star}$.

Example 2
The type of *map* $\in \mathcal{F}^{(2,2)}$,

$$(A : \star)(B : \star)(f : A \to B)(l : List\ A)(List\ B)$$

satisfies star dependency.

If we suppose that *map* $\in \mathcal{F}^{(1,3)}$, then its type would not satisfy the star dependency condition, as the output type of f depends on the big variable B that would not be a parameter.

Of course, if we suppose that all arguments of *map* are parameters (*map* $\in \mathcal{F}^{(4,0)}$), then its type would obviously satisfy the star dependency condition. On the other hand this would have a negative impact on the form of the rules which can be accepted by HORPO (every argument of map would have to be a variable as can be seen from the forthcoming definition of the parametric rule and HORPO).

2.4.2 *Strict positivity*

As already said in the introduction, we consider only small strictly positive inductive types. The inductive type s, defined by $Ind[p_1 : P_1,\dots p_r : P_r](s : A := c_1 : C_1,\dots c_m : C_m)$, is *small* if s is on the type level of the Calculus of Constructions ($s : T : \Box$ for some T).

In the definition below, we suppose that for every type constructor s there is a special variable $\bar{s}$. It is used to formulate positivity conditions and it will later be substituted by the real s applied do its parameters ($s(\vec{p})$). Positivity conditions are syntactic conditions on the form of A and $\vec{C}$, which are given below (see also Barras *et al.* (1999) and Paulin-Mohring (1993)).

Definition 2.4.2 ($\star$*-arity, positivity condition*)
A pseudoterm A is a $\star$*-arity* if $A = \overrightarrow{(x : b)}.\star$. We denote by $\mathcal{A}_\star$ the set of $\star$-arities.

A *type that is generated by* $\bar{s}$ is a pseudoterm of the form $\overrightarrow{(x : b)}.\bar{s}a_1\dots a_n$.

A type $t = \overrightarrow{(x : b)}.\bar{s}a_1\dots a_n$ generated by $\bar{s}$ satisfies the *positivity condition* if $\bar{s}$ does not occur in any a_i and its occurrences in each domain b_j of the product are *strictly positive*, i.e. each b_j in which $\bar{s}$ occurs is of the form $\overrightarrow{(x : d)}.\bar{s}e_1\dots e_k$ and $\bar{s}$ does not occur in $\vec{d}, \vec{e}$.

We write $t \in Pos^m(\bar{s})$ if t is a type generated by $\bar{s}$ that satisfies the positivity condition and the length of $\overrightarrow{(x : b)}$ is m.

2.4.3 Well-formedness of the extended signature

The following definition summarizes all condition that must be satisfied by constants from $\mathcal{F} \cup \mathcal{TC} \cup \mathcal{CS}$.

Definition 2.4.3 (*Well-formed extended signature*)
An extended signature Σ is well-formed if we can deduce Σ *ok* in the following system:

(start) $\emptyset\ ok$

(fun)
$$\frac{\Sigma\ ok \qquad \Sigma \vdash_{CC} (p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c\ :\ \star}{\Sigma, f\ :(p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c\ \ ok}$$

if $(p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c$ satisfies the star dependency condition w.r.t. $f \in \mathcal{F}^{(r,n)}$.

(ind)
$$\frac{\Sigma\ ok \qquad \Sigma, \Gamma_p \vdash_{CC} A : \Box \qquad \forall i \quad \Sigma, \Gamma_p, \overline{s} : A \vdash_{CC} C_i : \star}{\Sigma, Ind[\Gamma_p](s : A := c_1 : C_1, \dots c_m : C_m)\ \ ok}$$

for $\Gamma_p = p_1 : P_1, \dots p_r : P_r$, if the following conditions hold:

- $s \in \mathcal{TC}^r, \quad \forall i \exists n_i\ c_i \in \mathcal{CS}^{(r,n_i)}$,
- $A \in \mathcal{A}_\star, \quad \forall i\ C_i \in Pos^{n_i}(\overline{s})$,
- $\overrightarrow{(p : P)}.C_i[s(\vec{p})/\overline{s}]$ satisfies the star dependency condition w.r.t. $c_i \in \mathcal{CS}^{(r,n_i)}$

where $\vdash_{CC}$ means that we verify the corresponding typing judgment in the Calculus of Constructions without Constants, treating Σ as an environment, constants from $\mathcal{F} \cup \mathcal{TC} \cup \mathcal{CS}$ as variables and terms headed by constants $e(a_1, \dots a_n)$ like a variable e applied successively to terms $a_1, \dots a_n$.

Obviously, the well-formedness condition given above ensures that the signature is well-typed (i.e. $\Sigma \vdash \star\ :\ \Box$, see Definition 2.2.2). In the sequel we will always assume that our extended signature is well-formed.

Summarizing, the idea is that the user provides the constants, the arities and the signature and its well-formedness is automatically checked. Well-formedness is decidable, because the only difficult part in the its definition is type checking. The latter is decidable since it is the type checking of the pure Calculus of Constructions.

2.5 Relations and Orderings

In this paper, rather than relations on terms we will consider relations on terms in a given environment.

To say that terms a and b are in the relation $\rightarrow$ in the environment Γ we use the notation $\Gamma \vdash a \rightarrow b$ or $a \rightarrow^\Gamma b$, Γ being fixed. We denote by $\overset{*}{\rightarrow}{}^\Gamma$ the reflexive and transitive closure of $\rightarrow^\Gamma$. Unless stated differently, we will consider only those relations that do not distinguish α-convertible terms. Note that given an environment Γ, $\rightarrow^\Gamma$ is an ordinary relation on terms. Consequently $\rightarrow$ is *strongly normalizing*, or *terminating*, if for every Γ, the relation $\rightarrow^{\Gamma}$ is terminating, i.e. if for every term a there is no infinite sequence $a \rightarrow^{\Gamma} a_1 \rightarrow^{\Gamma} a_2 \rightarrow^{\Gamma} \cdots$.

Let us now introduce some useful definitions and extensions of the given (not necessarily transitive) relation $>$ on terms. In these extensions we will often use $=$, which is a syntactic equality on terms.

The *lexicographic extension* of $>$, denoted by $>_{lex}$, is defined as:

$$(a_1, \dots a_n) >_{lex} (b_1, \dots b_n) \quad \text{if} \quad \exists i \in \{1, \dots n\}\ a_i > b_i \text{ and } \forall j < i \ \ a_j = b_j.$$

The *multiset extension* of $>$, denoted by $>_{mul}$ is defined as:

$$M \uplus P >_{mul} M \uplus P' \quad \text{if} \quad P \neq \emptyset \text{ and } \forall b \in P' \ \exists a \in P \ \ a > b.$$

where $\uplus$ denotes the disjoint union.

An important fact is that for a terminating relation $>$, the relations $>_{lex}$ and $>_{mul}$ are also terminating (Dershowitz & Jouannaud, 1990).

Let us recall that a subterm at position p is called a constructor subterm if all symbols on the path from the root down to but not including p are constructors. The constructor subterm relation is denoted by $\rhd^{\mathcal{CS}}$.

The *constructor subterm extension* of $>$, denoted by $>^{\mathcal{CS}}$ is defined as:

$$\begin{array}{lll} a >^{\mathcal{CS}} b & \text{iff} & a > b \ \text{ or} \\ & & \exists c \ \ a \rhd^{\mathcal{CS}} c \text{ and } (c > b \text{ or } c = b). \end{array}$$

We will often use notation $\geqslant$ for $> \cup =$. Following this convention $\geqslant^{\mathcal{CS}}$ is equal to $>^{\mathcal{CS}} \cup =$ and we have $a \geqslant^{\mathcal{CS}} b$ if there is a c such that $a \unrhd^{\mathcal{CS}} c$ and $c \geqslant b$.

An *ordering* is a reflexive, antisymmetric and transitive relation. An ordering $\geqslant$ is *well-founded* if its strict part $>$ (defined as $a > b$ iff $a \geqslant b$ and not $b \geqslant a$) is strongly normalizing.

3 CC+H – Calculus of Constructions with rewriting induced by HORPO

The goal of this paper is twofold: to define a new calculus, obtained by incorporating to the Calculus of Constructions a rewriting mechanism generated by user-defined higher-order rewrite rules satisfying the criterion called Higher Order Recursive Path Ordering (HORPO) and to show that every instance of this calculus is strongly normalizing. To this end, we will actually show that the particular instance of the calculus, obtained by incorporating at once all rules satisfying HORPO, is strongly normalizing.

A difficulty is that rewriting is used in one of the type-checking rules – called conversion – whereas rewriting itself must ensure type preservation, making the definitions of rewriting and typing mutually dependent. This dependency is broken by type-checking the rewrite rules in the Calculus of Constructions (without user-defined rules in conversion). For the discussion about relaxing this assumption see section 9.2.

As already mentioned in the introduction, strong normalization will be shown under the assumption that rewriting operates on the object level. In our formalism, this is ensured by the requirement that for every function symbol f in the signature $f : T : \star$.

Before we define HORPO in section 3.4 we introduce and describe all its components. Throughout these definitions the typing is that of the Calculus of Constructions (as defined in section 2.2).

HORPO generates a relation on terms in an environment and can compare only terms of equal types. Every HORPO judgment has the form $\Gamma \vdash l \succ r \; : \; A$, where Γ is an environment, l, r are terms and A is their type. If the left-hand side l is headed by a function symbol then the whole judgment must be a parametric rule (see below). If in addition, the right-hand side r is an application then it must also verify the application condition (see section 3.2).

An important part of HORPO is the computable closure. It is used to enhance the set of acceptable right-hand sides in HORPO judgments and is defined as a possibly infinite set of terms generated by some computability preserving inference rules from the given initial set (see section 3.3).

Although sections 3.2 and 3.3 are important for the proof of normalization and for the expressive power of HORPO respectively, they are not essential for the understanding of the definition of HORPO. Therefore we advise the reader to skip these two sections for the first reading. On the contrary, the section 3.1 is crucial for the understanding of HORPO.

3.1 Parametricity condition

Parametricity is the general condition imposed on the environment Γ and parameter arguments of l in a HORPO judgment $\Gamma \vdash l \succ r \; : \; A$, in case l is a function symbol headed term.

Definition 3.1.1 (*Parametric rule*)
The quadruple (Γ, l, r, A) is a *parametric rule* if

1. l starts with a function symbol $f : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).c$,
2. $\Gamma = p_1 : P_1, \ldots p_r : P_r, G$,
3. $l = f(p_1, \ldots p_r, u_1, \ldots u_n)$,
4. $\Gamma \vdash_{CC} l \; : \; A$ and $\Gamma \vdash_{CC} r \; : \; A$.

In other words, parametricity requires that all parameter arguments of the left-hand side are different variables. The assumption that HORPO judgments verify parametricity condition is crucial for the proof of strong normalization (see section 5.4 for further explanations and section 7.1 for a counterexample).

Actually, in the above definition we could require $\forall\, i \;\; p_i \in dom(\Gamma)$ and $\forall\, i, j \;\; p_i \neq p_j$ instead of $\Gamma = p_1 : P_1, \ldots p_r : P_r, G$. Even if originally the p's were not in order, we can move them to the beginning of Γ, as P_i depends only on $p_1, \ldots p_{i-1}$ (because the type of f, $(p_1 : P_1) \ldots (p_r : P_r).(x_1 : b_1) \ldots (x_n : b_n).c$, is a well-typed term of the Calculus of Constructions). The fact that P_i does not depend on other variables from Γ than $p_1, \ldots p_{i-1}$ is very important in the proof of strong normalization and for this reason we prefer the explicit statement $\Gamma = p_1 : P_1, \ldots p_r : P_r, G$.

3.2 Application condition

In the definition of HORPO, there is a possibility to compare a term which starts with a function symbol with a term which starts with an application:

$$\overrightarrow{p:P}, G \vdash f(\vec{p},\vec{u}) \succ @(w_0,\dots,w_m) \ : A$$

But for technical reasons not every application can be a right-hand side in such a HORPO comparison. The restricting condition concerns the type of w_0 and it demands that this type is a product, which does not depend on those from $w_1, \dots w_m$ that are big and do not belong to $\vec{p}$. In other words, it expects that for every $i > 0$, the term w_i must be either an object, or a nondependent argument, or must belong to $\vec{p}$.

Definition 3.2.1 (*Application condition*)
The term $@(w_0,\dots,w_m)$ satisfies the *application condition* in the environment $(\overrightarrow{p:P}, G)$, if

$$\overrightarrow{p:P}, G \vdash_{CC} w_0 \ : (y_1 : A_1)\dots(y_m : A_m).B$$

and for all y_i

- $y_i \in \mathcal{V}ar^{\star}$, or
- $y_i \notin \mathcal{FV}(A_{i+1},\dots A_m, B)$, or
- $y_i \in \mathcal{V}ar^{\Box}$, and $w_i \in \vec{p}$.

We denote it by $AppCon(\overrightarrow{p:P}, G \vdash @(w_0,\dots,w_m))$.

Example 3
Suppose that $length \in \mathcal{F}^{(1,1)}$ has type $(A:\star)List(A) \to Nat$ and let us take

$$w_0 = \lambda A:\star.\lambda l:List(A).length(A,l)$$
$$v_0 = \lambda l:List(Nat).length(Nat,l)$$

In any environment $\overrightarrow{p:P}, G$ we have $\overrightarrow{p:P}, G \vdash_{CC} w_0 \ : (A:\star)List(A) \to Nat$ and $\overrightarrow{p:P}, G \vdash_{CC} v_0 \ : List(Nat) \to Nat$

Now, the condition

$$AppCon(\overrightarrow{p:P}, G \vdash @(w_0, Nat, nil(Nat)))$$

does not hold because in the type $(A:\star)List(A) \to Nat$, we have $A \in \mathcal{V}ar^{\Box}$ and $Nat \notin \vec{p}$. On the other hand, application conditions given below are satisfied:

$$AppCon(p:\star, l:List(p) \vdash @(w_0,p,l)) \ \text{ and}$$
$$AppCon(\overrightarrow{p:P}, G \vdash @(v_0, nil(Nat)))$$

3.3 Computable closure

Computable closure is used in the definition of HORPO and serves to enrich the choice of possibilities to accept a given right-hand side of a HORPO judgment $\overrightarrow{p:P}, G \vdash f(\vec{p},\vec{u}) \succ r \ : A$.

Without the computable closure, to accept r_i – an immediate subterm of the right-hand side – we would use the comparison either with the whole left-hand side $f(\vec{p}, \vec{u})$, or with one of its immediate subterms. The role of computable closure is to accept r_i if it results from simple "computability preserving" transformations of $\vec{p}, \vec{u}$ (like, for example, applying u_{j_1} to u_{j_2}). Computability refers here to Tait and Girard celebrated reducibility predicate technique.

The definition of computable closure and that of HORPO are mutually dependent as one of the transformations mentioned above, **Recursive call**, requires some HORPO inequalities to be satisfied. Despite this circularity the definitions are correct, which is shown in Lemma 3.4.2.

Computable closure is a set of quintuples, whose first element is called a *leading term*. We write informally $r_i \in CCl_f(\vec{p}, \vec{u})$ if a tuple led by r_i (verifying some additional conditions) belongs to the computable closure of $f(\vec{p}, \vec{u})$. At the end of this subsection there is a complete proof (in Example 5) that some term belongs to a given computable closure.

In what follows, we will give the formal definition of the computable closure preceded by the definition of (G_p, G)-tuples, that are the quintuples on which computable closure operates. These two definitions are rather complex and at the first reading it may be difficult to guess what their different components are for. In fact, even though the computable closure rules operate on quintuples and have complicated side-conditions, if we restrict our attention to the leading terms the rules turn out to be quite simple (the rules are named after their operations on leading terms). The remaining four components of tuples are necessary to prove "computability preservation" in the fundamental technical Lemma 5.4.7.

Definition 3.3.1 (*(G_p, G)-tuple*)
A quintuple $(t,\ G_m; BV \vdash T,\ \mu)$ is called a (G_p, G)-*tuple* if

1. $\mathcal{FV}(T) \cap \mathcal{V}ar^{\Box} \subseteq G_p \cup BV$,
2. μ is a well-typed substitution from G_p, G_m to G_p, G, such that $\mu|_{G_p} = id_{G_p}$,
3. $G_p, G_m, BV \vdash_{CC} T\ :\ \star/\Box$,
4. $G_p, G, BV\mu \vdash_{CC} t\ :\ T\mu$.

The BV component is an environment consisting of, so-called, bound variables.

In a tuple $(t,\ G_m; BV \vdash T,\ \mu)$ the term t is called the *leading term*. By $\mathcal{FBV}(a)$ we denote those variables from BV that are free in a. A term a is *clean* if $\mathcal{FBV}(a) = \emptyset$.

Example 4
Suppose that $map \in \mathcal{F}^{(2,2)}$ has type $(A : \star)(B : \star)(A \rightarrow B)List(A) \rightarrow List(B)$ and that

$$A : \star, B : \star, f : A \rightarrow B, a : A, l : List(A) \vdash map(A, B, f, cons(A, a, l))\ :\ List(A)$$

holds. Then

$$(f,\ \emptyset; \emptyset \vdash A \rightarrow B,\ [A/A, B/B]) \quad \text{and}$$
$$(cons(A, a,\ l),\ y : A \rightarrow B; \emptyset \vdash List(A),\ [A/A, B/B, f/y])$$

are $(A : \star, B : \star, f : A \rightarrow B, a : A, l : List(A))$-tuples.

In a tuple $(t,\ G_m; BV \vdash T,\ \mu)$, the BV component represents variables in t and T that will later be bound by computable closure rule **Abstraction**, as we are interested only in terms that are clean. This mechanism is used to construct, for example, $\lambda x\!:\!A.@(u,x)$ from $@(u,x)$ by abstracting over a free (BV) variable x.

In fact, the definition above considers two levels at the same time. The first one, where T can be typed and the other, where t is of type $T\mu$. The reason to distinguish these two levels is to ensure that $T\mu$ is of very special form: all big variables of T are included in G_p (assuming BV is empty) and μ affects only small variables of T. This property will be crucial in Lemma 5.4.7 which says roughly that for a tuple $(t,\ G_m; BV \vdash T,\ \mu)$ in the computable closure, t belongs to the interpretation of T under the substitution μ.

Definition 3.3.2 (*Computable Closure*)
Let f have a type $(p_1 : P_1)\dots(p_r : P_r).(x_1 : b_1)\dots(x_n : b_n).c$ and let us suppose that $G_p, G \vdash_{CC} f(\vec{p},\vec{u})\ :\ c[\vec{u}/\vec{x}]$ holds for some environments $G_p = \overrightarrow{p : P}$ and G and for some terms $\vec{u}$.

The *initial set* for $G_p, G \vdash f(\vec{p},\vec{u})$, denoted by $Init(G_p, G \vdash f(\vec{p},\vec{u}))$, is the following set of (G_p, G)-tuples:

$$\{(p_i,\ \emptyset;\emptyset \vdash P_i,\ id_{G_p}) \quad | \quad 1 \leqslant i \leqslant r\} \quad \cup$$
$$\{(u_j,\ x_1 : b_1, \dots x_{j-1} : b_{j-1}; \emptyset \vdash b_j,\ id_{G_p} \cup [\vec{u}/\vec{x}]) \quad | \quad 1 \leqslant j \leqslant n\}$$

The terms $\vec{p}, \vec{u}$ are called *initial terms.*

The *Computable Closure of* $G_p, G \vdash f(\vec{p},\vec{u})$, denoted by $CCl(G_p, G \vdash f(\vec{p},\vec{u}))$, is the smallest set containing $Init(G_p, G \vdash f(\vec{p},\vec{u}))$ and closed under the operations described below:

Introduction of variables

$$\frac{G_p, G_m, BV \vdash_{CC} y\ :\ A, \qquad y \in dom(BV)}{(y,\ G_m; BV \vdash A,\ \mu)}$$

if $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box} \subseteq G_p \cup BV$, and μ is well-typed substitution from G_p, G_m to G_p, G, such that $\mu|_{G_p} = id_{G_p}$.

Abstraction

$$\frac{\begin{array}{cc}(t,\ G_m; BV, y : A \vdash T,\ \mu) & (A\mu,\ G_m; BV \vdash p_2,\ \mu) \\ G_p, G_m, BV, y : A \vdash_{CC} T\ :\ p_1 & G_p, G_m, BV \vdash_{CC} A\ :\ p_2\end{array}}{(\lambda y\!:\!A\mu.t,\ G_m; BV \vdash (y\!:\!A).T,\ \mu)}$$

if $p_1, p_2 \in \{\star, \Box\}$.

Application to a variable

$$\frac{(t,\ G_m; BV \vdash (x\!:\!A).B,\ \mu) \qquad (y,\ G_m; BV \vdash A,\ \mu)}{(@(t,y),\ G_m; BV \vdash B[y/x],\ \mu)}$$

if $y \in BV$.

Application to a parameter

$$\frac{(t,\ G_m; BV \vdash (x\!:\!P).B,\ \mu)}{(@(t,p),\ G_m; BV \vdash B[p/x],\ \mu)}$$

if $p : P \in G_p$.

Application to a small argument

$$\frac{(t_1,\ G_m;BV \vdash (x:A).B,\ \mu) \qquad (t_2,\ G_m;BV \vdash A,\ \mu)}{(@(t_1,t_2),\ G_m, x : A;BV \vdash B,\ \mu \cup [t_2/x])}$$

if $x \in \mathcal{V}ar^{\star}$ and t_2 and A are clean.

Application to a nondependent argument

$$\frac{(t_1,\ G_m;BV \vdash (x:A).B,\ \mu) \qquad (t_2,\ G_m;BV \vdash A,\ \mu)}{(@(t_1,t_2),\ G_m;BV \vdash B,\ \mu)}$$

if $x \notin \mathcal{FV}(B)$.

Precedence

$$\frac{\forall i\ \ (t_i,\ G_m, \overrightarrow{y : d[\vec{p}'/\vec{q}]};BV \vdash d_i[\vec{p}'/\vec{q}],\ \mu \cup [\vec{t}/\vec{y}])}{(g(\vec{p}',\vec{t}),\ G_m, \overrightarrow{y : d[\vec{p}'/\vec{q}]};BV \vdash e[\vec{p}'/\vec{q}],\ \mu \cup [\vec{t}/\vec{y}])}$$

if $f >_{\mathcal{F}} g$, $g : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(y_1 : d_1)\dots(y_m : d_m).e$, $\vec{p}' \subseteq dom(G_p)$ and for every i either t_i is clean or $y_i \notin \mathcal{FV}(d_{i+1},\dots d_m, e)$.

Recursive call

$$\frac{\forall i\ \ (t_i,\ G_m, \overrightarrow{x : b};\emptyset \vdash b_i,\ \mu \cup [\vec{t}/\vec{x}])}{(f(\vec{p},\vec{t}),\ G_m, \overrightarrow{x : b};\emptyset \vdash c,\ \mu \cup [\vec{t}/\vec{x}])}$$

if $G_p, G \vdash (\vec{p},\vec{u}) \succ^{\mathcal{CS}}_{stat} (\vec{p},\vec{t})$, where $stat$ is a status of $f : \overrightarrow{(p : P)}\overrightarrow{(x : b)}.c$ and $\succ^{\mathcal{CS}}_{stat}$ is a $stat$ extension of the constructor extension of $\succ$ (see Section 2.5).

Constructor—decomposition

$$\frac{(c(\vec{a}',\vec{N}),\ G_m;BV \vdash s(\vec{a})\vec{u},\ \mu)}{(a'_i,\ G_m;BV \vdash Q_i[\vec{a}/\vec{q}],\ \mu)}$$

$$\frac{(c(\vec{a}',\vec{N}),\ G_m;BV \vdash s(\vec{a})\vec{u},\ \mu)}{(N_i,\ G_m, \overrightarrow{z : d[\vec{a}/\vec{q}]};BV \vdash d_i[\vec{a}/\vec{q}],\ \mu \cup [\vec{N}/\vec{z}])}$$

if $c : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(z_1 : d_1)\dots(z_k : d_k).s(\vec{q})w_1\dots w_n$ is a constructor of $s : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(x_1 : b_1)\dots(x_n : b_n).\star$, if $\vec{a}$ are clean and if for every i either N_i is clean or $z_i \notin \mathcal{FV}(d_{i+1},\dots d_k)$.

Constructor

$$\frac{\begin{array}{ll}\forall i\ (a'_i,\ G_m;BV \vdash Q_i[\vec{a}/\vec{q}],\ \mu) & \forall i\ (N_i,\ G_m, \overrightarrow{z : d[\vec{a}/\vec{q}]};BV \vdash d_i[\vec{a}/\vec{q}],\ \mu \cup [\vec{N}/\vec{z}]) \\ G_p, G_m, BV \vdash_{CC} s(\vec{a})\vec{u}\ :\ \star & G_p, G, BV\mu \vdash_{CC} c(\vec{a}',\vec{N})\ :\ (s(\vec{a})\vec{u})\mu\end{array}}{(c(\vec{a}',\vec{N}),\ G_m;BV \vdash s(\vec{a})\vec{u},\ \mu)}$$

if $c : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(z_1 : d_1)\dots(z_k : d_k).s(\vec{q})w_1\dots w_n$ is a constructor of $s : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(x_1 : b_1)\dots(x_n : b_n).\star$ and if $\mathcal{FV}(\vec{a},\vec{u}) \cap \mathcal{V}ar^{\square} \subseteq G_p \cup BV$.

Type constructor

$$\frac{\forall i\ (a_i,\ G_m;BV \vdash A_i,\ \mu) \quad \forall i\ (u_i,\ G_m;BV \vdash U_i,\ \mu) \quad G_p, G, BV\mu \vdash_{CC} s(\vec{a})\vec{u}\ :\ \star}{(s(\vec{a})\vec{u},\ G_m;BV \vdash \star,\ \mu)}$$

if $s : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(x_1 : b_1)\dots(x_n : b_n).\star \in \mathcal{TC}$.

Product type

$$\frac{\begin{array}{cc}(A,\ G_m;BV \vdash p_1,\ \mu) & (B,\ G_m;BV,x:C \vdash p_2,\ \mu)\\ G_p,G,BV\mu,x:A \vdash_{CC} B\ :\ p_2 & G_p,G,BV\mu \vdash_{CC} A \overset{*}{\leftrightarrow} C\mu\end{array}}{((x:A).B,\ G_m;BV \vdash p_2,\ \mu)}$$

if $p_1, p_2 \in \{\star, \Box\}$.

Reduction

$$\frac{(t,\ G_m;BV \vdash T,\ \mu)}{(t',\ G_m;BV \vdash T,\ \mu)}$$

if $t \rightarrow_\beta t'$.

Weak1

$$\frac{(t,\ G_m;BV \vdash T,\ \mu) \qquad G_p,G_m \vdash_{CC} A\ :\ \star/\Box \qquad G_p,G \vdash_{CC} M\ :\ A\mu}{(t,\ G_m,x:A;BV \vdash T,\ \mu \cup [M/x])}$$

if $x \notin dom(G_m, BV)$.

Weak2

$$\frac{(t,\ G_m;BV \vdash T,\ \mu) \qquad G_p,G_m,BV \vdash_{CC} A\ :\ \star/\Box}{(t,\ G_m;BV,x:A \vdash T,\ \mu)}$$

if $x \notin dom(G_m, BV)$ and $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box} \subseteq G_p, BV$.

When no confusion can arise, the computable closure $CCl(G_p, G \vdash f(\vec{p}, \vec{u}))$ will simply be written $CCl_f(\vec{l})$, where $\vec{l}$ are the initial terms $(\vec{p}, \vec{u})$. We say that $t \in CCl_f(\vec{l})$ if there is a tuple $(t,\ G_m;BV \vdash T,\ \mu) \in CCl_f(\vec{l})$, such that t and T are clean. In particular, this simpler notation ($t \in CCl_f(\vec{l})$) will be used in the definition of HORPO.

Lemma 3.3.3
Let us suppose that $\overrightarrow{p:P}, G \vdash f(\vec{p}, \vec{u})\ :\ A$ holds. Then every quintuple $(t,\ G_m;BV \vdash T,\ \mu) \in CCl_f(\vec{p}, \vec{u})$ is a $(\overrightarrow{p:P}, G)$-tuple.

Proof
Easy induction on the derivation of the tuple. □

Example 5
Let $map \in \mathcal{F}^{(2,2)}$ be of type $(A\ :\ \star)(B\ :\ \star)(A \rightarrow B)List(A) \rightarrow List(B)$. We will show that $@(f,a) \in CCl(A\ :\ \star, B\ :\ \star, f\ :\ A \rightarrow B, a\ :\ A, l\ :\ List(A) \vdash map(A,B,f,cons(A,a,l)))$. The tuples

$$(f,\ \emptyset;\emptyset \vdash A \rightarrow B,\ [A/A, B/B]) \quad \text{and}$$
$$(cons(A,a,\ l),\ y:A \rightarrow B;\emptyset \vdash List(A),\ [A/A, B/B, f/y])$$

are in the computable closure by the definition of the initial set. By **Weak1** and **Constructor-decomposition** applied respectively to the above tuples, we get

$$(f,\ y:A \rightarrow B;\emptyset \vdash A \rightarrow B,\ [A/A, B/B, f/y]) \quad \text{and}$$
$$(a,\ y:A \rightarrow B;\emptyset \vdash A,\ [A/A, B/B, f/y])$$

Now it is sufficient to use **Application to a nondependent argument** to obtain $(@(f,a),\ y\ :\ A \rightarrow B;\emptyset \vdash B,\ [A/A, B/B, f/y])$. Since $@(f,a)$ and B are obviously clean ($BV = \emptyset$) we obtain $@(f,a) \in CCl_{map}(A, B, f, cons(A,a,l))$.

3.4 HORPO

The Higher Order Recursive Path Ordering (HORPO) presented in definition 3.4.1 below is an adaptation of HORPO (Jouannaud & Rubio, 1999) to the Calculus of Constructions, which was itself a generalization of the well-known RPO (Dershowitz, 1982) to the simply typed lambda calculus (with a polymorphism "à la ML").

A HORPO judgment has the form $\Gamma \vdash l \succ r \ : A$, where Γ is an environment, l, r are terms and A their type. The definition of HORPO involves many different ingredients:

- a well-founded ordering on $\mathcal{F}$, called *precedence*,
- a partition of $\mathcal{F}$ into symbols of lexicographic (Lex) or multiset (Mul) *status*; the status tells us how to compare recursively immediate subterms of two terms headed by the same symbol.
- the parametricity condition (Definition 3.1.1), which must be satisfied if l is a function symbol headed term,
- the application condition (Definition 3.2.1), which must be satisfied if l is a function symbol headed term and r is an application,
- $\succeq$, the union of $\succ$ and $=$, that is the syntactic equality on terms,
- $\succ^{\mathcal{CS}}$, the constructor extension of $\succ$ and $\succeq^{\mathcal{CS}}$ defined as $\succ^{\mathcal{CS}} \cup =$ (see section 2.5),
- $\succ^{\mathcal{CS}}_{lex}$, $\succ^{\mathcal{CS}}_{mul}$ that are respectively lexicographic and multiset extensions of $\succ^{\mathcal{CS}}$ (see section 2.5),
- and finally the computable closure $CCl_f(\vec{l})$ of the term $f(\vec{l})$ with $f \in \mathcal{F}$, which is a set of terms built from $\vec{l}$ by using "computability preserving" operations (see section 3.3).

Compared to RPO (Dershowitz, 1982), HORPO has two new cases allowing to deal with an application and constructor headed term on the right-hand side (in Part I) and four cases corresponding to monotonicity rules for application, abstraction, product and constants from $\mathcal{TC} \cup \mathcal{CS}$ (Part II). Moreover, it has more possibilities to deal with subterms of the right-hand side, which is described by the condition $\mathfrak{P}$:

$$\forall w_i \in \vec{w} \quad \Gamma \vdash f(\vec{l}) \succ w_i \ : A \quad \text{or} \quad \Gamma \vdash l_j \succeq^{\mathcal{CS}} w_i \text{ for some } l_j \text{ or } w_i \in CCl_f(\vec{l})$$

It says that each subterm w_i of the right-hand side is recursively compared either with the whole left-hand side $f(\vec{l})$, or with some constructor subterm, by $\succeq^{\mathcal{CS}}$, or otherwise must belong to the computable closure $CCl_f(\vec{l})$. The choice in $\mathfrak{P}$ is meant to compensate the requirement that HORPO compares only terms of equal types and to prevent the situation where we cannot limit some "evidently smaller" subterm of the right-hand side because there is no left-hand side subterm of the same type.

Definition 3.4.1 (*HORPO*)
Let $>_{\mathcal{F}}$ be a precedence and let $\mathfrak{P}$ denote the condition:

$$\forall w_i \in \vec{w} \quad \Gamma \vdash f(\vec{l}) \succ w_i \ : A \quad \text{or} \quad \Gamma \vdash l_j \succeq^{\mathcal{CS}} w_i \text{ for some } l_j \text{ or } w_i \in CCl_f(\vec{l})$$

By induction with respect to $(|l|, |r|)_{lex}$ we define the strict comparison in HORPO as follows:

$$\Gamma \vdash l \succ r \ : A \ \text{ if}$$
$$\Gamma \vdash_{CC} l \ : A \qquad \Gamma \vdash_{CC} r \ : A \quad \text{and}$$

Part I $\Gamma = (p_1 : P_1) \dots (p_r : P_r), G,$
$l = f(\vec{p}, \vec{u}) = f(l_1, \dots, l_m),$
$f : (x_1 : T_1) \dots (x_{r+n} : T_{r+n}).T \in \mathcal{F}^{(r,n)}$
and

1. $r \in CCl_f(\vec{l})$ or $\Gamma \vdash l_i \succeq^{\mathcal{CS}} r$ for some i, or
2. $r = g(\vec{p'}, \vec{w})$, $g \in \mathcal{F}$, $f >_{\mathcal{F}} g$, $\vec{p'} \subseteq \vec{p}$ and $\mathfrak{P}$, or
3. $r = f(\vec{p}, \vec{w})$, $f \in Mul$ and $\Gamma \vdash (\vec{p}, \vec{u}) \succ^{\mathcal{CS}}_{mul} (\vec{p}, \vec{w})$, or
4. $r = f(\vec{p}, \vec{w})$, $f \in Lex$, $\Gamma \vdash (\vec{p}, \vec{u}) \succ^{\mathcal{CS}}_{lex} (\vec{p}, \vec{w})$ and $\mathfrak{P}$
5. $r = c(\vec{w})$, $c \in \mathcal{CS}$ and $\mathfrak{P}$, or
6. $@(w_0, \dots, w_m)$ is a left-flattening of r, $\mathfrak{P}$ and $AppCon(\overrightarrow{p : P}, G \vdash @(w_0, \dots, w_m))$ hold and
   - $\Gamma \vdash l \succ w_0 \ : T[\vec{l}/\vec{x}]$, or
   - $\Gamma \vdash l_j \succeq^{\mathcal{CS}} w_0$ for some j,

Part II

1. $l = \lambda x : l_1.l_2$, $r = \lambda x : r_1.r_2$ and there exists C such that $\Gamma \vdash_{CC} A \overset{*}{\leftrightarrow} (x : l_1).C$, $\Gamma \vdash_{CC} l_1 \overset{*}{\leftrightarrow} r_1$ and $\Gamma, x : l_1 \vdash l_2 \succ r_2 \ : C$, or
2. $l = (x : l_1).l_2$, $r = (x : r_1).r_2$ and $\Gamma \vdash_{CC} l_1 \overset{*}{\leftrightarrow} r_1$ and $\Gamma, x : l_1 \vdash l_2 \succ r_2 \ : A$, or
3. $l = l_1 \, l_2$, $r = r_1 \, r_2$ and there exist B, C such that $\Gamma \vdash_{CC} A \overset{*}{\leftrightarrow} C[l_2/x]$, $\Gamma \vdash l_1 \succeq r_1 \ : (x : B).C$, $\Gamma \vdash l_2 \succeq r_2 \ : B$ and at least one of these comparisons is strict, or
4. $l = e(l_1, \dots l_n)$, $r = e(r_1, \dots r_n)$, $e : \overrightarrow{(x : b)}.c \in \mathcal{TC} \cup \mathcal{CS}$, for all i $\Gamma \vdash l_i \succeq r_i \ : b_i[\vec{l}/\vec{x}]$ and at least one of these comparisons is strict.

The correctness of this definition follows from the lemma given below:

Lemma 3.4.2
Whenever in the definition of $\Gamma \vdash l \succ r \ : A$ we refer to $\Gamma' \vdash l' \succ r' \ : A'$, we always have $(|l|, |r|) >_{lex} (|l'|, |r'|)$.

Proof
It is easy to check that in all HORPO cases direct recursive calls of $\succ$ are made on smaller terms. In the computable closure, HORPO is used only in the **Recursive call** rule. Note that computable closure is used only when $l = f(\vec{l})$. The **Recursive call** rule proves that $f(\vec{t}) \in CCl_f(\vec{l})$ for some $\vec{t}$ already in $CCl_f(\vec{l})$ and such that $\Gamma \vdash \vec{l} \succ^{\mathcal{CS}}_{stat} \vec{t}$, where *stat* is the status of f. This finishes the proof, since $\vec{l}$ are subterms of l. $\square$

Part I is the heart of HORPO, as its cases 1, 2, 3, 4 correspond to the RPO. In Part I the left-hand side is always a function symbol headed term. The right-hand side can be either a function symbol headed term (cases 2, 3, 4), a constructor headed term (case 5) or an application (case 6). We can also choose a recursive comparison with an immediate subterm of the left-hand side (case 1). In case 6 the appropriate left-flattening has to be chosen non-deterministically. This non-determinism aims at easing the recursive comparison in case of type incompatibility (HORPO compares only terms of the same type).

As a consequence of being defined for symbols that have dependent types, HORPO pays special attention to parameters of function symbols. In Part I the left-hand side must always have different variables $\vec{p}$ on its parameter positions (according to the parametricity condition 3.1.1), and if the right-hand side is also a function symbol headed term then its arguments on parameter positions must be included in $\vec{p}$. This property guarantees that during the rewriting with such a rule, we know exactly all big arguments of the right-hand side (as they may occur only at parameter positions and parameters are passed unchanged) which is very important in the proof of strong normalization.

The use of the application condition (see Section 3.2) in case 6 obeys to the same idea. It says roughly that $f(\vec{p},\vec{u})$ can be compared with $@(t_0, t_1, \ldots t_m)$, only if all big terms within $t_1, \ldots t_n$ belong to $\vec{p}$.

3.5 Examples

Example 6
Let $map \in \mathcal{F}^{(2,2)}$ be of type $(A:\star)(B:\star)(A \to B)List(A) \to List(B)$ with $map \in Mul$ and let the rules be the following:

$$A:\star,\ B:\star,\ f:A\to B\ \vdash\ map(A,B,f,nil(A))\ \to\ nil(B)\ :\ List(B)$$

$$\begin{aligned}&A:\star,\ B:\star,\ f:A\to B,\ a:A,\ l:List(A)\\ &\qquad\vdash\ map(A,B,f,cons(A,a,l))\ \to\ cons(@(f,a),map(A,B,f,l))\ :\ List(B)\end{aligned}$$

The first rule is accepted by case I.5, since $A:\star, B:\star, f:A\to B \vdash B \succeq^{\mathcal{CS}} B$.

For the second rule let us call the environment $A:\star, B:\star, f:A\to B, a:A, l:List(A)$ by Γ. We can use case I.5 and we are left with the following subgoals:

$$@(f,a) \in CCl_{map}(A,B,f,cons(A,a,l))$$

$$\Gamma \vdash map(A,B,f,cons(A,a,l)) \succ map(A,B,f,l) : List(B)$$

For the latter, we can apply case I.3 of HORPO and then show

$$\Gamma \vdash (A,B,f,cons(A,a,l)) \succ^{\mathcal{CS}}_{mul} (A,B,f,l)$$

which follows from

$$\Gamma \vdash cons(A,a,l) \succ^{\mathcal{CS}} l$$

which is true by the definition of the constructor extension.

The formal proof of that $@(f,a) \in CCl_{map}(A,B,f,cons(A,a,l))$ is given in Example 5. Note that without the computable closure it is not possible to accept this rule since there is no subterm of the left-hand side of type B.

Example 7
Suppose that $map \in \mathcal{F}^{(2,2)}$ is the same as in the previous example. The rule

$$C : \star, l : List(C) \vdash map(C, C, \lambda x : C.x, l) \to l : List(C)$$

cannot be accepted by HORPO because it is not parametric (the arguments at parameter positions are not distinct variables)

It may be surprising that the rule from example 7 is rejected by our framework. We do not think this particular rule leads to nontermination, but section 7.1 shows an example (derived from (Girard, 1971)), where one easily gets a nonterminating term precisely because the condition about the arguments at parameter positions is violated.

Our formulation of HORPO is not stable under instantiation (of big variables). The reason is that when comparing $\Gamma \vdash f(\vec{l}) \succ r \ : A$, the definition of HORPO requires that the arguments of f at parameter positions are different variables. Consequently if we take a substitution that equalizes two parameters then the result would not be accepted by HORPO.

Example 8
HORPO is not stable under instantiation, because

$$\begin{aligned}&A : \star, B : \star, f : A \to B, a : A, l : List(A) \vdash \\ &\qquad map(A, B, f, cons(A, a, l)) \succ cons(@(f, a), map(A, B, f, l)) : List(B)\end{aligned}$$

holds (as proved in Example 6), and

$$\begin{aligned}&C : \star, f : C \to C, a : C, l : List(C) \vdash \\ &\qquad map(C, C, f, cons(C, a, l)) \succ cons(@(f, a), map(C, C, f, l)) : List(C)\end{aligned}$$

does not (since parameter arguments of map on the left-hand side are not different variables as required).

Note also that HORPO is not an ordering, as it is not transitive (mainly due to the use of the computable closure). This is not really a problem, since we will show that HORPO is well-founded, hence allowing us to reason with its transitive closure. On the other hand, the transitive closure is of course not decidable. In practice, as shown in (Jouannaud & Rubio, 2001), it may be useful to use two successive steps of HORPO for solving some particular examples. Below, is a counterexample to transitivity.

Example 9
Suppose that $f \in \mathcal{F}^{(1,1)}$ and $g \in \mathcal{F}^{(2,1)}$ have types:

$$\begin{aligned}f &: (A : \star)s_1(A) \to (A \to A)\\ g &: (A : \star)(B : \star)s_2(B) \to (A \to A)\end{aligned}$$

where s_1 and s_2 are inductive types defined by:

$$Ind[A : \star](s_1 : \star := c_1 : (A \to A) \to \overline{s_1})$$
$$Ind[B : \star](s_2 : \star := c_2 : \overline{s_2} \to \overline{s_2})$$

and that $\Gamma = A : \star, B : \star, b : s_2(B)$. Then

$$\Gamma \vdash f(A, c_1(A, g(A, B, c_2(B, b)))) \succ \lambda z : A.@(g(A, B, c_2(B, b)), z) \; : \; A \to A$$

holds by case I.1 of HORPO, and more precisely because $\lambda z : A.@(g(A, B, c_2(B, b)), z) \in CCl_f(A, c_1(A, g(A, B, c_2(B, b))))$ and

$$\Gamma \vdash \lambda z : A.@(g(A, B, c_2(B, b)), z) \succ \lambda z : A.@(g(A, B, b), z) \; : \; A \to A$$

holds by cases II.1 and II.3 of HORPO. But

$$\Gamma \vdash f(A, c_1(A, g(A, B, c_2(B, b)))) \succ \lambda z : A.@(g(A, B, b), z) \; : \; A \to A$$

does not hold.

The proof of the strong normalization (in section 5) will of course concern the rewrite relation generated by HORPO, that is its closure by instantiation and context.

More examples and non-examples can be found in section 8.

3.6 **CC + H**

CC+H is the result of mixing together the judgments of the Calculus of Constructions and of HORPO. Except for the last two rules, all the rules are those of the Calculus of Constructions.

The rule **(rew)** defines rewriting as a closure by substitution and context of valid HORPO judgments. It is similar to first order rewriting, but typability has to be taken care of.

Let a be a term with a potential redex at position p that we want to rewrite using the rule $G \vdash l \succ r \; : \; T$ and substitution θ. Because of abstractions and products we need to allow θ to substitute the free variables in l by terms in which variables bound above p in a may occur. Let us introduce the environment $\mathcal{PV}(a, p)$ (for path variables) consisting of the list of these variables, found along the path from the root to the position p in a, together with their types.

$$\begin{aligned}
\mathcal{PV}(a, \Lambda) &= [\,] \\
\mathcal{PV}(\lambda x : b.c, 1 \cdot q) &= \mathcal{PV}(b, q) \\
\mathcal{PV}(\lambda x : b.c, 2 \cdot q) &= (x : b) :: \mathcal{PV}(c, q) \\
\mathcal{PV}((x : b).c, 1 \cdot q) &= \mathcal{PV}(b, q) \\
\mathcal{PV}((x : b).c, 2 \cdot q) &= (x : b) :: \mathcal{PV}(c, q) \\
\mathcal{PV}(b\, c, 1 \cdot q) &= \mathcal{PV}(b, q) \\
\mathcal{PV}(b\, c, 2 \cdot q) &= \mathcal{PV}(c, q) \\
\mathcal{PV}(e(b_1, \dots, b_n), i \cdot q) &= \mathcal{PV}(b_i, q) \quad \text{if } e \in \mathcal{F} \cup \mathcal{TC} \cup \mathcal{CS}
\end{aligned}$$

Assuming moreover that a is well-typed in the environment Γ, θ must substitute variables from G (the environment of the HORPO judgment) by terms that are well typed in the environment $\Gamma, \mathcal{PV}(a, p)$. Such a substitution θ is written $\theta \; : \; G \to \Gamma, \mathcal{PV}(a, p)$ and is called a well-typed substitution.

The other difference of our calculus with respect to the Calculus of Constructions is the conversion rule **(conv)** which now incorporates not only beta but also the rewrite relation generated by HORPO.

Although the following two definitions are mutually dependent we give them separately for the sake of clarity.

Definition 3.6.1 (*Well-typed substitution*)
A substitution θ is a *well-typed substitution from the environment* $\Delta_1 = x_1 : A_1 \ldots x_n : A_n$ *to the environment* Δ_2, denoted by $\theta : \Delta_1 \to \Delta_2$, if for all $x_i \in dom(\Delta_1)$ we have $\Delta_2 \vdash x_i\theta : A_i[x_1\theta/x_1 \ldots x_{i-1}\theta/x_{i-1}]$.

Definition 3.6.2 (*CC+H*)
Given a precedence and a status function for HORPO, the system CC+H is defined as follows:

$$\textbf{(ax)}\quad \frac{}{\vdash \star : \Box}$$

$$\textbf{(var)}\quad \frac{\Gamma \vdash a : p}{\Gamma, x : a \vdash x : a} \qquad (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{(weak)}\quad \frac{\Gamma \vdash a : b \qquad \Gamma \vdash c : p}{\Gamma, x : c \vdash a : b} \qquad (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{(const)}\quad \frac{\Gamma \vdash_{CC} \star : \Box \quad \forall i \quad \Gamma \vdash a_i : b_i[a_1/x_1, \ldots, a_{i-1}/x_{i-1}]}{\Gamma \vdash e(a_1, \ldots, a_n) : c[a_1/x_1, \ldots, a_n/x_n]} \qquad (e : \overrightarrow{(x : b)}.c \in \Sigma)$$

$$\textbf{(abs)}\quad \frac{\Gamma, x : a \vdash b : c \qquad \Gamma \vdash (x{:}a).c : p}{\Gamma \vdash \lambda x{:}a.b : (x{:}a).c} \qquad (x \notin dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{(app)}\quad \frac{\Gamma \vdash a : (x{:}b).c \qquad \Gamma \vdash d : b}{\Gamma \vdash a\,d : c[d/x]}$$

$$\textbf{(prod)}\quad \frac{\Gamma \vdash a : p \qquad \Gamma, x : a \vdash b : q}{\Gamma \vdash (x{:}a).b : q} \qquad (x \notin dom(\Gamma), p, q \in \{\star, \Box\})$$

$$\textbf{(rew)}\quad \frac{G \vdash l \succ r : T \qquad a|_p = l\theta \qquad \theta : G \to \Gamma, \mathcal{PV}(a,p)}{\Gamma \vdash a[l\theta]_p \to_H a[r\theta]_p}$$

$$\textbf{(conv)}\quad \frac{\begin{array}{ccc} \Gamma \vdash a : b & & \Gamma \vdash b' : p \\ \Gamma \vdash b(\to_\beta \cup \to_H)^* b' & \text{or} & \Gamma \vdash b'(\to_\beta \cup \to_H)^* b \end{array}}{\Gamma \vdash a : b'} \qquad (p \in \{\star, \Box\})$$

In the rules above, $\Gamma \vdash a \to_H b$ stands for the closure of HORPO by instantiation and context. The environment Γ is necessary (see section 2.5), so we could also write $a \to_H^\Gamma b$, but not $a \to_H b$ (the superscript Γ cannot be omitted). The relation $(\to_\beta \cup \to_H)^\Gamma$ stands for the union of beta and the rewrite relations and $((\to_\beta \cup \to_H)^\Gamma)^*$ is used for its reflexive and transitive closure. For simplicity, we write $\Gamma \vdash a \to b$ for $\Gamma \vdash a(\to_\beta \cup \to_H)b$. We also use the notation $\to^\Gamma$ for this relation.

By $\overset{*}{\leftrightarrow}^\Gamma$ we denote the conversion relation: $\Gamma \vdash a \overset{*}{\leftrightarrow} b$ holds if there is a sequence $a = a_0 \overset{*}{\to} a_1 \overset{*}{\leftarrow} a_2 \overset{*}{\to} \ldots \overset{*}{\leftarrow} a_n = b$ such that every a_i satisfies $\Gamma \vdash c_i : a_i$ or

$\Gamma \vdash a_i : p_i$ for some c_i, p_i (like in the premises of **(conv)** rule). Hence conversion is a restricted version of reflexive, symmetric and transitive closure of $\rightarrow$.

According to the definition of rewriting given in the beginning of this section one would expect $\Gamma \vdash a : A$ to be among the premises of **(rew)**. However we prefer not to include this premise because of technical problems in the proof of subject reduction. In fact, the system given above is equivalent (derives the same typing judgments) to the system where $\Gamma \vdash a : A$ is a premise of **(rew)** and the rule **(conv)** is defined for one step of beta or rewrite reduction.

Definition 3.6.3 (*Well-typed terms*)
The set of well-typed terms, denoted by $\mathcal{T}$, is defined by:

$$\mathcal{T} = \{a \mid \exists \Gamma\, \exists b\; \Gamma \vdash a : b\} \cup \{\Box\}$$

where a, b are pseudoterms. A Γ-term is a term which is typable (or is a type of some term) in the environment Γ. Among Γ-terms we distinguish Γ-objects $\{a \mid \exists b\; \Gamma \vdash a : b \text{ and } \Gamma \vdash b : \star\}$. Every Γ-term that is not a Γ-object is called Γ-nonobject.

Lemma 3.6.4
Let Γ_1, Γ_2 and Δ be environments. If $\rho_1 : \Gamma_1 \rightarrow \Delta$ and $\rho_2 : \Gamma_2 \rightarrow \Delta$ are well-typed substitutions such that $dom(\Gamma_1) \cap dom(\Gamma_2) = \emptyset$ then their parallel composition $\rho_1 \cup \rho_2$ is a well-typed substitution from (Γ_1, Γ_2) to Δ.

Similarly if $\rho_1 : \Gamma_1 \rightarrow \Gamma_2$ and $\rho_2 : \Gamma_2 \rightarrow \Delta$ are well-typed substitution then their sequential composition $\rho_1\rho_2$ is a well-typed substitution from Γ_1 to Δ.

Summarizing, we may say that the definitions appear in the following order:

1. $\Gamma \vdash_{CC} a : b$, the definition of the Calculus of Constructions,
2. $\Gamma \vdash a \rightarrow_\beta b$, the definition of beta reduction (the same as $a \rightarrow_\beta b$),
3. $\Gamma \vdash a \succ b : T$, the definition of HORPO, uses 1,
4. $\Gamma \vdash a \rightarrow_H b$, the definition of rewriting, mutually defined with $\Gamma \vdash a : b$ (typing judgments in of CC+H); they use 2 and 3.

4 Basic meta-theory

This section contains some basic meta-theory of $CC + H$. The strong normalization proof is given in the next section.

4.1 Structural properties

We present here some structural properties of CC+H. These are well known properties of the Calculus of Constructions and our rewriting does not change them.

Lemma 4.1.1 (*Substitution property*)
Let $\Gamma_1 \vdash d : a$. Then

$$\begin{array}{rcl} \Gamma_1, x : a, \Gamma_2 \vdash b : c & \text{implies} & \Gamma_1, \Gamma_2[d/x] \vdash b[d/x] : c[d/x] \\ \Gamma_1, x : a, \Gamma_2 \vdash b \rightarrow_H b' & \text{implies} & \Gamma_1, \Gamma_2[d/x] \vdash b[d/x] \rightarrow_H b'[d/x] \end{array}$$

Moreover the same holds for $\rightarrow_\beta$ instead of $\rightarrow_H$.

As a corollary we get that $\Gamma \vdash b : c$ and $\theta : \Gamma \rightarrow \Delta$ imply $\Delta \vdash b\theta : c\theta$.

Lemma 4.1.2
If $\Gamma_1, x : a, y : b, \Gamma_2 \vdash c \;:\; d$ and $x \notin \mathcal{FV}(b)$ then $\Gamma_1, y : b, x : a, \Gamma_2 \vdash c \;:\; d$.

Lemma 4.1.3 (Stripping)
Let $\Gamma \vdash a \;:\; b$. Then:

- if $a = p$ then $p = \star$, $\Gamma \vdash b \overset{*}{\leftrightarrow} \square$,
- if $a = x$ then $x : b' \in \Gamma$, $\Gamma \vdash b' \;:\; p$, $p \in \{\star, \square\}$, $\Gamma \vdash b \overset{*}{\leftrightarrow} b'$
- if $a = f(a_1, \dots a_n)$ then
  $f : (x_1 : b_1) \dots (x_n : b_n).c \in \mathcal{F}^n$,
  $\forall i \quad \Gamma \vdash a_i \;:\; b_i[a_1/x_1, \dots, a_{i-1}/x_{i-1}]$,
  $\Gamma \vdash b \overset{*}{\leftrightarrow} c[a_1/x_1, \dots, a_n/x_n]$
- if $a = s(a_1, \dots a_r)$ then
  $s : (p_1 : P_1) \dots (p_r : P_r).d \in \mathcal{TC}^r, \quad d \in \mathcal{A}_\star$,
  $\forall i \quad \Gamma \vdash a_i \;:\; P_i[a_1/p_1, \dots, a_{i-1}/p_{i-1}]$,
  $\Gamma \vdash b \overset{*}{\leftrightarrow} d[\vec{a}/\vec{x}]$,
- if $a = c(a_1, \dots a_r, a'_1, \dots a'_n)$ then
  $c : (p_1 : P_1) \dots (p_r : P_r).(x_1 : b_1) \dots (x_n : b_n).s(\vec{p})\vec{w} \in \mathcal{CS}^{(r,n)}$
  $s : (p_1 : P_1) \dots (p_r : P_r).(y_1 : d_1) \dots (y_m : d_m).\star \in \mathcal{TC}^r$,
  $\forall i \quad \Gamma \vdash a_i \;:\; P_i[a_1/p_1, \dots, a_{i-1}/p_{i-1}]$,
  $\forall i \quad \Gamma \vdash a'_i \;:\; b_i[a_1/p_1, \dots, a_r/p_r][a'_1/x_1, \dots, a'_{i-1}/x_{i-1}]$,
  c is a constructor of s, $\Gamma \vdash b \overset{*}{\leftrightarrow} (s(\vec{p})\vec{w})[\vec{a}/\vec{p}][\vec{a'}/\vec{x}]$
- if $a = \lambda x : a_1.a_2$ then
  $\exists c \quad \Gamma, x : a_1 \vdash a_2 \;:\; c, \quad \Gamma \vdash (x : a_1).c \;:\; p$,
  $p \in \{\star, \square\}, \quad \Gamma \vdash b \overset{*}{\leftrightarrow} (x : a_1).c$
- if $a = (x : a_1).a_2$ then
  $\Gamma \vdash a_1 \;:\; p_1, \quad \Gamma, x : a_1 \vdash a_2 \;:\; p_2, \quad p_1, p_2 \in \{\star, \square\}$,
  $\Gamma \vdash b \overset{*}{\leftrightarrow} p_2$
- if $a = a_1\, a_2$ then
  $\exists c, d \quad \Gamma \vdash a_1 \;:\; (x : c).d, \quad \Gamma \vdash a_2 \;:\; c, \quad \Gamma \vdash b \overset{*}{\leftrightarrow} d[a_2/x]$

Definition 4.1.4
Terms of CC+H are split into the following categories:

$$\begin{aligned}
Kind &= \{K \in \mathcal{T} \mid \exists \Gamma \quad \Gamma \vdash K \;:\; \square\} \\
Constr &= \{A \in \mathcal{T} \mid \exists \Gamma, \exists K \in Kind \quad \Gamma \vdash A \;:\; K\} \\
Type &= \{T \in \mathcal{T} \mid \exists \Gamma \quad \Gamma \vdash T \;:\; \star\} \\
Obj &= \{M \in \mathcal{T} \mid \exists \Gamma, \exists T \in Type \quad \Gamma \vdash M \;:\; T\}
\end{aligned}$$

By Γ-kind, Γ-constr, … we mean a Γ-term belonging to the appropriate class. Note that *Type* is a special case of *Constr*. The other cases are disjoint by the Classification Lemma : $Kind \cap Type = Constr \cap Obj = \emptyset$.

For the sake of uniformity we will write "a has the type b in the environment Γ" to denote the fact that the judgment $\Gamma \vdash a \;:\; b$ is derivable, even though b may actually be a kind or $\square$.

Now we can describe terms in each class. The letters K, A, T, M denote elements of *Kind*, *Constr*, *Type* and *Obj* respectively; s denotes a type constructor, c a constructor, f a function symbol, x a variable from $\mathcal{V}ar^\star$ and y from $\mathcal{V}ar^\Box$:

$$\begin{array}{rcl}
K & ::= & \star \mid (x:T).K \mid (x:K).K \\
A & ::= & s(a_1,\dots a_n) \mid y \mid (x:T).T \mid (x:K).T \mid \lambda x:T.A \mid \lambda x:K.A \mid A\,M \mid A\,A \\
M & ::= & x \mid f(a_1,\dots a_n) \mid c(a_1,\dots a_n) \mid \lambda x:T.M \mid \lambda x:K.M \mid M\,M \mid M\,A
\end{array}$$

4.2 Subject reduction

To show that $\to^\Gamma$ has the subject reduction property, it suffices to show it for the rewrite reduction ($\to_H^\Gamma$) and the beta reduction ($\to_\beta$) separately. Most of the section is devoted to the subject reduction of $\to_H^\Gamma$, which is rather involved due to the fact that $\to_H^\Gamma$ depends on the environment Γ. For the subject reduction of $\to_\beta$, we only sketch how to reuse the proof given by Barbanera, Fernández and Geuvers (1994, 1997). This proof is more complicated than the proof of subject reduction for the pure Calculus of Constructions, because the relation that is used in the conversion rule of CC+H, $(\to_\beta \cup \to_H)^\Gamma$, is not confluent.

We begin with some auxiliary lemmas:

Lemma 4.2.1 (*Stability under context*)
Relations $\to_\beta$ and $\to_H^\Gamma$ are stable under context, that is if $\Gamma \vdash a\,(\to_\beta \cup \to_H)a'$ then for any environment Δ, term c and position p, such that $\Gamma \subseteq \Delta, \mathcal{PV}(c,p)$ we have $\Delta \vdash c[a]_p(\to_\beta \cup \to_H)c[a']_p$.

Proof
For $\to_\beta$ it is obvious by its definition.

For $\to_H^\Gamma$, let $G \vdash l \to r : T$ and $\theta : G \to \Gamma, \mathcal{PV}(a,q)$ be the rewrite rule and the well-typed substitution that justify $\Gamma \vdash a \to_H a'$. Then the same rule and substitution justify also $\Delta \vdash c[a]_p \to_H c[a']_p$. Indeed, θ may be seen as a well-typed substitution from G to $\Delta, \mathcal{PV}(c[a]_p, p{\cdot}q)$, because $\Gamma, \mathcal{PV}(a,q) \subseteq \Delta, \mathcal{PV}(c[a]_p, p{\cdot}q)$ by the assumption that $\Gamma \subseteq \Delta, \mathcal{PV}(c,p)$ and the fact that $\mathcal{PV}(c[a]_p, p\cdot q) = \mathcal{PV}(c,p), \mathcal{PV}(a,q)$. $\Box$

Lemma 4.2.2
Let $\Gamma, x : b, \Gamma' \vdash a : T$ hold and let c, c' be terms satisfying $\Gamma \vdash c : b$ and $\Gamma \vdash c' : b$. If $\Gamma \vdash c \to_H c'$ then $\Gamma, \Gamma'[c/x] \vdash a[c/x] \overset{*}{\to}_H a[c'/x]$. The same holds for $\to_\beta$ instead of $\to_H^\Gamma$.

Proof
It is obvious for $\to_\beta$ by definition. For $\to_H^\Gamma$, we do the proof by induction on the derivation of $\Gamma, x : b, \Gamma' \vdash a : T$. Below, we give the proof for the case where the last rule in the derivation is **(const)**:

$$\frac{\forall i \quad \Gamma, x : b, \Gamma' \vdash a_i : d_i[a_1/y_1, \dots, a_{i-1}/y_{i-1}]}{\Gamma, x : b, \Gamma' \vdash e(a_1, \dots, a_n) : v[a_1/y_1, \dots, a_n/y_n]} \; (e : (y_1 : d_1)\dots(y_n : d_n).v \in \Sigma)$$

We have to show that $\Gamma, \Gamma'[c/x] \vdash e(a_1, \dots, a_n)[c/x] \overset{*}{\rightarrow}_H e(a_1, \dots, a_n)[c'/x]$. By induction hypothesis, for every $i = 1 \dots n$ $\Gamma, \Gamma'[c/x] \vdash a_i[c/x] \overset{*}{\rightarrow}_H a_i[c'/x]$ holds. Since $e(a_1, \dots, a_n)[c/x] = e(a_1[c/x], \dots a_n[c/x])$, it is sufficient to use Lemma 4.2.1 n times to get the conclusion. □

Definition 4.2.3 (*Rewriting of environments*)
Let $\Gamma = x_1 : A_1 \dots x_n : A_n$ be an environment. We say that $\Gamma \rightarrow_H \Gamma'$ ($\Gamma \rightarrow_\beta \Gamma'$) if there is some i such that $x_1 : A_1 \dots x_{i-1} : A_{i-1} \vdash A_i (\rightarrow_\beta \cup \rightarrow_H) A'_i$ and $\Gamma' = x_1 : A_1 \dots x_{i-1} : A_{i-1}, x_i : A'_i, x_{i+1} : A_{i+1} \dots x_n : A_n$.

4.2.1 *Subject Reduction of rewriting*

Subject reduction of $\rightarrow^\Gamma_H$ will be a consequence of the following lemma.

Lemma 4.2.4
For all environments Γ, Γ' and all terms a, a', b, c, c':

(1). $\Gamma \vdash a : b$ and $\Gamma \vdash a \rightarrow_H a'$ imply $\Gamma \vdash a' : b$,
(2). $\Gamma \vdash a : b$ and $\Gamma \rightarrow_H \Gamma'$ imply $\Gamma' \vdash a : b$,
(3). $\Gamma \vdash c \rightarrow_H c'$ and $\Gamma \rightarrow_H \Gamma'$ imply $\Gamma' \vdash c \rightarrow_H c'$,
(4). $\Gamma \vdash c \rightarrow^* c'$ and $\Gamma \rightarrow_H \Gamma'$ imply $\Gamma' \vdash c \rightarrow^* c'$.

Proof
The proof is done by mutual induction on the derivation of $\Gamma \vdash a : b$, $\Gamma \vdash c \rightarrow_H c'$ and $\Gamma \vdash c \rightarrow^* c'$. Let us detail the proofs of (3) and (4) and the proof of (1) in case the last rule in the derivation of $\Gamma \vdash a : b$ is **(const)**.

(1) **(const)**

$$\frac{\forall i \quad \Gamma \vdash a_i : d_i[a_1/y_1, \dots, a_{i-1}/y_{i-1}]}{\Gamma \vdash e(a_1, \dots, a_n) : v[a_1/y_1, \dots, a_n/y_n]} \quad (e : (y_1 : d_1) \dots (y_n : d_n).v \in \Sigma)$$

Let $\Gamma \vdash e(a_1, \dots, a_n) \rightarrow_H w$. We have to check that $\Gamma \vdash w : v[a_1/y_1, \dots, a_n/y_n]$.

There are two cases: either there is a j such that $\Gamma \vdash a_j \rightarrow_H a'_j$ or $e(\vec{a})$ is a redex itself. In the first case, by induction hypothesis we have $\Gamma \vdash a'_j : d_j[\vec{a}/\vec{y}]$ and $\Gamma \vdash a_i : d_i[\vec{a}/\vec{y}]$ for every i different from j. Let us set $a'_i = a_i$ for $i \neq j$ and $\gamma'_i = [a'_1/y_1, \dots a'_{i-1}/y_{i-1}]$ for all i. To apply the **(const)** rule to $a'_1, \dots a'_n$ we first have to convert the type for a'_i for $i > j$.

Since $e : (y_1 : d_1) \dots (y_n : d_n).v \in \Sigma$, we have $\overrightarrow{y : d} \vdash v : \star$ and for every $i = 1, \dots n$ there is some $p_i \in \{\star, \Box\}$ such that $y_1 : d_1, \dots y_{i-1} : d_{i-1} \vdash d_i : p_i$. We will show that for every i:

$$\Gamma \vdash d_i\gamma'_i : p_i \qquad \text{and} \qquad \Gamma \vdash a'_i : d_i\gamma'_i \qquad\qquad (*)$$

This is trivial for $i < j$, since $a'_i = a_i$. For $i = j$ we have $\Gamma \vdash a_j \rightarrow_H a'_j$ and $\Gamma \vdash a'_j : d_j\gamma'_j$ by induction hypothesis. The judgment $\Gamma \vdash d_j\gamma'_j : p_j$ is again trivial. Now suppose that we have $(*)$ for all natural numbers smaller or equal to some $i \geqslant j$. We will show $(*)$ for $i + 1$.

By $(*)$, γ'_{i+1} is a well-typed substitution from $y_1 : d_1, \dots y_i : d_i$ to Γ and therefore $y_1 : d_1, \dots y_i : d_i \vdash d_{i+1} : p_{i+1}$ implies $\Gamma \vdash d_{i+1}\gamma'_{i+1} : p_{i+1}$ by Substitution property 4.1.1.

Now, by Lemma 4.2.2, $\Gamma \vdash d_{i+1}[\vec{a}/\vec{y}] \overset{*}{\rightarrow}_H d_{i+1}\gamma'_{i+1}$. We can apply the **(conv)** rule to obtain $\Gamma \vdash a'_{i+1} : d_{i+1}\gamma'_{i+1}$ and we have $(*)$. Similarly we get $\Gamma \vdash v\gamma'_{n+1} : \star$ and $\Gamma' \vdash v[\vec{a}/\vec{y}] \overset{*}{\rightarrow}_H v\gamma'_{n+1}$.

Now, having $(*)$ for all $i = 1, \dots n$, we deduce $\Gamma \vdash e(a_1, \dots a'_j, \dots a_n) : v\gamma'_{n+1}$. To obtain the conclusion, we have to use once again the conversion rule together with the facts that $\Gamma' \vdash v[\vec{a}/\vec{y}] \overset{*}{\rightarrow}_H v\gamma'_{n+1}$ and $\Gamma \vdash v\gamma'_{n+1} : \star$.

If $e(\vec{a})$ is a redex itself then, by the definition of rewriting, there is a rule $G \vdash l \rightarrow r : T$ and a well-typed substitution $\theta : G \rightarrow \Gamma$ such that $G \vdash l : T$, $G \vdash r : T$, $e(\vec{a}) = l\theta$, $w = r\theta$ and $\Gamma \vdash T\theta \overset{*}{\leftrightarrow} v[\vec{a}/\vec{y}]$. Applying θ to $G \vdash r : T$ we get $\Gamma \vdash w : T\theta$, which implies, together with $\Gamma \vdash T\theta \overset{*}{\leftrightarrow} v[\vec{a}/\vec{y}]$, that $\Gamma \vdash w : v[\vec{a}/\vec{y}]$.

(3) **(rew)**

$$\frac{G \vdash l \succ r : T \qquad a|_p = l\theta \qquad \theta : G \rightarrow \Gamma, \mathcal{PV}(a,p)}{\Gamma \vdash a[l\theta]_p \rightarrow_H a[r\theta]_p}$$

We have to show that $\Gamma' \vdash a[l\theta]_p \rightarrow_H a[r\theta]_p$ assuming $\Gamma \rightarrow_H \Gamma'$. Let $G = x_1 : A_1, \dots x_n : A_n$.

To this end, we have to check that θ is a well-typed substitution from G to $\Gamma', \mathcal{PV}(a,p)$. By assumption, we know that for every i, $\Gamma, \mathcal{PV}(a,p) \vdash x_i\theta : A_i\theta$. By induction hypothesis we get $\Gamma', \mathcal{PV}(a,p) \vdash x_i\theta : A_i\theta$, since $\Gamma \rightarrow_H \Gamma'$ implies $(\Gamma, \mathcal{PV}(a,p)) \rightarrow_H (\Gamma', \mathcal{PV}(a,p))$. But this means that $\theta : G \rightarrow \Gamma', \mathcal{PV}(a,p)$ and consequently, by **(rew)** $\Gamma' \vdash a[l\theta]_p \rightarrow_H a[r\theta]_p$.

(4) The proof for this case consists of multiple application of case (3) By the definition of $\overset{*}{\rightarrow}$, $\Gamma \vdash c \rightarrow^* c'$ means there is some n such that $\Gamma \vdash c_i(\rightarrow_\beta \cup \rightarrow_H)c_{i+1}$ for $i = 1, \dots n$, $c_0 = c$ and $c_n = c'$.

Let $\Gamma \rightarrow_H \Gamma'$. If $\Gamma \vdash c_i \rightarrow_\beta c_{i+1}$ then obviously $\Gamma' \vdash c_i \rightarrow_\beta c_{i+1}$ as beta reduction does not depend on the environment. Otherwise $\Gamma \vdash c_i \rightarrow_H c_{i+1}$ and by induction hypothesis $\Gamma' \vdash c_i \rightarrow_H c_{i+1}$. Consequently $\Gamma' \vdash c \rightarrow^* c'$. $\square$

4.2.2 Subject reduction of beta

The only difficult point in the proof of the subject reduction for $\rightarrow_\beta$ is to show that for any two convertible product types there is a sequence of conversions passing only through products.

Without confluence, the proof of this fact seems difficult, but fortunately it has already been done in Barbanera *et al.* (1997). Even though that paper concerns only algebraic rewriting, the proof itself depends only on the following properties of term rewriting systems:

1. left-hand and right-hand sides of a rewrite rule must have the same type,
2. rewriting operates on the object level (both sides of the rule are objects),
3. subject reduction for rewriting is satisfied.

Since all these properties are true in our calculus, we can reuse this proof. We state below the property needed for products.

Lemma 4.2.5
Suppose that $\Gamma \vdash (x : c).d \overset{*}{\leftrightarrow} (x : c').d'$ holds. Then there exists a sequence of Γ-terms $(x : c_1).d_1, \dots (x : c_m).d_m$ such that:

1. $(x:c).d = (x:c_1).d_1$,
2. $(x:c_m).d_m = (x:c').d'$,
3. for every $i = 1 \ldots m-1$ we have either $\Gamma \vdash (x:c_i).d_i \rightarrow^* (x:c_{i+1}).d_{i+1}$ or $\Gamma \vdash (x:c_{i+1}).d_{i+1} \rightarrow^* (x:c_i).d_i$.

The intuitive explication of this lemma is that neither $\rightarrow_H^\Gamma$, nor $\rightarrow_\beta$ can create or remove a Π symbol. For $\rightarrow_H^\Gamma$ we use essentially the fact that all rewrites take place at object level. The complete proof is rather long and complicated, and it exploits the commutation properties of rewriting with beta.

Lemma 4.2.6
If $\Gamma \vdash a : b$, $a \rightarrow_\beta a'$ and $\Gamma \rightarrow_\beta \Gamma'$ then $\Gamma' \vdash a' : b$.

Proof
The proof is done by induction on the derivation of $\Gamma \vdash a : b$. The only difficulty occurs in case **(app)**, when we need to show that $(x:c).d \overset{*}{\leftrightarrow}^\Gamma (x:c').d'$ implies $c \overset{*}{\leftrightarrow}^\Gamma c'$ and $d \overset{*}{\leftrightarrow}^{\Gamma,x:c} d'$. But this follows from the previous lemma. □

Theorem 1 (*Subject reduction*)
If $\Gamma \vdash a : b$ and $\Gamma \vdash a \rightarrow a'$ then $\Gamma \vdash a' : b$.

Proof
This follows from Lemmas 4.2.4 and 4.2.6. □

5 Strong normalization

To prove the strong normalization of $\rightarrow^\Gamma$, we use the well known method of “reducibility candidates”. Its untyped version was used in several ways to prove the strong normalization of the pure Calculus of Constructions, but rewriting forces us to use sets of well-typed terms for candidates, as introduced in Coquand & Gallier (1990). The presentation of candidates departs slightly from the original one, following instead Blanqui *et al.* (1999).

The proof follows the standard scheme. We first define candidates, as a subset of strongly normalizing terms closed under some properties. Then we define interpretation of types and we show that every interpretation is a candidate. Compared to Coquand & Gallier (1990), we have to enrich the definition of candidates with the case covering rewriting (actually it is covered by the case for neutral terms) and give the definition of interpretation for inductive types (section 5.2.1).

According to the standard scheme, one then shows the main lemma saying that every term belongs to the interpretation of its type, and concludes that all terms are strongly normalizing. In our case, there are three new kinds of terms for which we have to show the main lemma. Type constructor and constructor headed terms belong to the corresponding interpretations by the definition of the interpretation for inductive types. The case of function symbol headed terms is much more delicate, as these terms may rewrite and, contrary to beta-reduction, there is not enough information stored in the interpretation to solve this case easily. This problem is treated in section 5.4.

It turns out that the only way to show that the function symbol headed term $f(\vec{t})$ belongs to the interpretation of its type is to show the same property for all its reducts. This is done in the Fun Lemma in section 5.4.3, by inspecting all HORPO cases and reasoning by induction on $>_{\mathcal{F}}$ (the precedence) and $(\rightarrow^{\Gamma})_{stat}$ (reductions from $\vec{t}$, with *stat* the status of f). This kind of lemma and induction was already present in Jouannaud & Rubio (1999), but the fact that function symbols have dependent types makes the proof much more involved here.

This section is organized as follows. We begin by the candidates (section 5.1) and we continue with interpretations (section 5.2). Auxiliary lemmas concerning properties of interpretations are given in section 5.3. Section 5.4 is devoted to the reducibility of function symbol headed terms. Fun Lemma is a last part of this subsection. Two other parts concern the notions needed for the Fun Lemma, that is equality of interpretations (section 5.4.1) and constructor subterm lemmas (section 5.4.2). We finish by presenting the proof of the Main Lemma in section 5.5.

5.1 *Family of candidates*

Definition 5.1.1 (*Neutral terms, strongly normalizing terms*)

The set of neutral terms *Neutr* is the set of terms, which are not an abstraction or constructor headed.

We say that the sequent $\Delta \vdash M$ is $\mathcal{SN}$ if there is no infinite sequence of $\rightarrow^{\Delta}$ reductions originating from M. We define:

$$\mathcal{SN}_{\Delta,A} = \{\Delta' \vdash M \mid \Delta' \vdash M \ : \ A,\ \Delta' \supseteq \Delta \text{ and } \Delta' \vdash M \text{ is } \mathcal{SN}\}$$

$$\mathcal{SN}_{\Delta} = \bigcup_{A} \mathcal{SN}_{\Delta,A}$$

Definition 5.1.2 (*Family of candidates*)

A family $\mathcal{C}$ of sets $\mathcal{C}_{\Delta,A}$ where A is a Δ-nonobject, is called a *family of candidates* if sets $\mathcal{C}_{\Delta,A}$ verify the conditions below. Each $C \in \mathcal{C}_{\Delta,A}$ is called a *candidate*. At the same time, we define $C|_{\Delta'}$, for every $\Delta' \supseteq \Delta$, that is the *restriction of the candidate C to the environment Δ'*:

1. $\mathcal{C}_{\Delta,\square} = \{\mathcal{SN}_{\Delta,\square}\}$,
   In this case, we define the *restriction* of $\mathcal{SN}_{\Delta,\square}$ to $\Delta' \supseteq \Delta$ to be $(\mathcal{SN}_{\Delta,\square})|_{\Delta'} = \mathcal{SN}_{\Delta',\square}$,
2. $\mathcal{C}_{\Delta,\star} = \{\mathcal{SN}_{\Delta,\star}\}$,
   In this case, we define the *restriction* of $\mathcal{SN}_{\Delta,\star}$ to $\Delta' \supseteq \Delta$ to be $(\mathcal{SN}_{\Delta,\star})|_{\Delta'} = \mathcal{SN}_{\Delta',\star}$,
3. If A is a type or a kind different from $\star$, then $\mathcal{C}_{\Delta,A}$ is the set of all sets C, each C being a nonempty subset of $\mathcal{SN}_{\Delta,A}$ such that the following properties hold:
   (S1) If $\Delta' \vdash a \in C$ and $\Delta' \vdash a \rightarrow a'$ then $\Delta' \vdash a' \in C$.
   (S2) For every neutral term a, such that $\Delta' \vdash a \ : \ A$ and $\Delta' \supseteq \Delta$, if for every term a' satisfying $\Delta' \vdash a \rightarrow a'$, $\Delta' \vdash a' \in C$ then $\Delta' \vdash a \in C$.
   (S3) If $\Delta' \vdash a \in C$ and $\Delta' \subseteq \Delta''$, then $\Delta'' \vdash a \in C$.

In this case, we define the *restriction* of $C \in \mathcal{C}_{\Delta,A}$ to $\Delta' \supseteq \Delta$ to be $C|_{\Delta'} = \{\Delta'' \vdash M \in C \mid \Delta'' \supseteq \Delta'\}$.

4. When A is in *Constr*$\setminus$*Type* ($\Delta \vdash A : (x:B).D$) then $\mathcal{C}_{\Delta,A}$ is a set of all functions with the following properties:
   (a) If B is a kind, then

   (P1) $h \in \mathcal{C}_{\Delta,A}$ is a function with domain

   $$\{(\Delta' \vdash M,\, C) \mid \Delta' \vdash M : B,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',M}\}$$

   such that $h(\Delta' \vdash M,\, C) \in \mathcal{C}_{\Delta',A\,M}$.

   (P2) $h(\Delta' \vdash M_1,\, C) = h(\Delta' \vdash M_2,\, C)$ whenever $\Delta' \vdash M_1 \overset{*}{\leftrightarrow} M_2$.

   (P3) $h(\tilde{\Delta} \vdash M, C)|_{\Delta'} = h(\Delta' \vdash M, C|_{\Delta'})$ whenever $\Delta \subseteq \tilde{\Delta} \subseteq \Delta'$, $\tilde{\Delta} \vdash M : B$ and $C \in \mathcal{C}_{\tilde{\Delta},M}$.

   In this case, we define the *restriction* of $h \in \mathcal{C}_{\Delta,A}$ to $\Delta' \supseteq \Delta$ ($h|_{\Delta'}$) to be a function with domain $\{(\Delta'' \vdash M, C) \mid \Delta'' \vdash M : B,\ \Delta'' \supseteq \Delta',\ C \in \mathcal{C}_{\Delta'',M}\}$ such that $h|_{\Delta'}(\Delta'' \vdash M, C) = h(\Delta'' \vdash M, C)$.

   (b) If B is a type, then

   (P1) $h \in \mathcal{C}_{\Delta,A}$ is a function with domain $\{\Delta' \vdash M \mid \Delta' \vdash M : B \text{ and } \Delta' \supseteq \Delta\}$ such that $h(\Delta' \vdash M) \in \mathcal{C}_{\Delta',A\,M}$.

   (P2) $h(\Delta' \vdash M_1) = h(\Delta' \vdash M_2)$ whenever $\Delta' \vdash M_1 \overset{*}{\leftrightarrow} M_2$.

   (P3) $h(\tilde{\Delta} \vdash M)|_{\Delta'} = h(\Delta' \vdash M)$ whenever $\Delta \subseteq \tilde{\Delta} \subseteq \Delta'$ and $\tilde{\Delta} \vdash M : B$.

   In this case, we define the *restriction* of $h \in \mathcal{C}_{\Delta,A}$ to $\Delta' \supseteq \Delta$ ($h|_{\Delta'}$) to be a function with domain $\{\Delta'' \vdash M \mid \Delta'' \vdash M : B \text{ and } \Delta'' \supseteq \Delta'\}$ such that $h|_{\Delta'}(\Delta'' \vdash M) = h(\Delta'' \vdash M)$.

Remark 0
Cases 4(a) and 4(b) are very similar, 4(b) being a sort of instance of 4(a) with the second argument (for h) being an empty set. As a consequence, in all proofs to come, we will always consider case 4(a) only.

Remark 1
Note that the case $A = s(\vec{M})$, $s \in \mathcal{TC}$ is hidden in case 3 or 4. Note also that up to now, there is no way to state whether $c(\vec{M})$, $c \in \mathcal{CS}$, belongs to a candidate C. This will be the subject of section 5.2.1.

Remark 2
From the definition of the family it follows immediately that $\mathcal{C}_{\Delta,A} = \mathcal{C}_{\Delta,A'}$ for $\Delta \vdash A \overset{*}{\leftrightarrow} A'$ and that $(C|_{\Delta'})|_{\Delta''} = C|_{\Delta''}$ for every $C \in \mathcal{C}_{\Delta,A}$ and $\Delta \subseteq \Delta' \subseteq \Delta''$.

Lemma 5.1.3 (Restricted candidate)
Let A be a Δ-nonobject, $C \in \mathcal{C}_{\Delta,A}$ and $\Delta \subseteq \Delta'$. Then $C|_{\Delta'} \in \mathcal{C}_{\Delta',A}$.

Proof

- if A is $\Box$ or $\star$, it is obvious,

- if A is a type or a kind different from $\star$ and $C \in \mathcal{C}_{\Delta,A}$ then $C|_{\Delta'} = \{\Delta'' \vdash M \in C \mid \Delta'' \supseteq \Delta'\}$. Of course, $C|_{\Delta'} \in \mathcal{SN}_{\Delta',A}$ and $C|_{\Delta'} \neq \emptyset$. To see that $C|_{\Delta'} \in \mathcal{C}_{\Delta',A}$, we have just to check conditions (S1), (S2), (S3), which is straightforward.
- if $\Delta \vdash A : (x:B).D$, B is a kind and $h \in \mathcal{C}_{\Delta,A}$ then $h|_{\Delta'}$ is a function with domain $\{(\Delta'' \vdash M, C) \mid \Delta'' \vdash M : B,\ \Delta'' \supseteq \Delta',\ C \in \mathcal{C}_{\Delta'',M}\}$ such that $h|_{\Delta'}(\Delta'' \vdash M, C) = h(\Delta'' \vdash M, C)$. To show that $h|_{\Delta'} \in \mathcal{C}_{\Delta',A}$ we have to check that

  (P1) $h|_{\Delta'}(\Delta'' \vdash M,\ C) \in \mathcal{C}_{\Delta'',A\,M}$, which is true by the definition of $h|_{\Delta'}$ and (P1) for h,

  (P2) $h|_{\Delta'}(\Delta'' \vdash M_1,\ C) = h|_{\Delta'}(\Delta'' \vdash M_2,\ C)$ whenever $\Delta'' \vdash M_1 \overset{*}{\leftrightarrow} M_2$, which is true since $h|_{\Delta'}(\Delta'' \vdash M_1,\ C) = h(\Delta'' \vdash M_1,\ C)$, $h|_{\Delta'}(\Delta'' \vdash M_2,\ C) = h(\Delta'' \vdash M_2,\ C)$ and $h(\Delta'' \vdash M_1,\ C) = h(\Delta'' \vdash M_2,\ C)$ by (P2) for h,

  (P3) $h|_{\Delta'}(\tilde\Delta \vdash M, C)|_{\Delta''} = h|_{\Delta'}(\Delta'' \vdash M, C|_{\Delta''})$ whenever $\Delta' \subseteq \tilde\Delta \subseteq \Delta''$, $\tilde\Delta \vdash M : B$ and $C \in \mathcal{C}_{\tilde\Delta,M}$, which is true since

$$\begin{array}{lll} h|_{\Delta'}(\tilde\Delta \vdash M, C)|_{\Delta''} = h(\tilde\Delta \vdash M, C)|_{\Delta''} & & \text{by def. of } h|_{\Delta'} \\ h(\tilde\Delta \vdash M, C)|_{\Delta''} = h(\Delta'' \vdash M, C|_{\Delta''}) & & \text{by (P3) for } h \\ h(\Delta'' \vdash M, C|_{\Delta''}) = h|_{\Delta'}(\Delta'' \vdash M, C|_{\Delta''}) & & \text{by def. of } h|_{\Delta'} \quad \square \end{array}$$

To show that the definition of the family is correct we need to introduce a measure m. By induction on m we will construct a canonical candidate $can_{\Delta,A} \in \mathcal{C}_{\Delta,A}$, which will roughly be $\mathcal{SN}_{\Delta,A}$, if A is a $\square$, a kind or a type, and a function, returning $can_{\Delta,A\,M}$ for a given M, otherwise.

Definition 5.1.4
Let T be a type and K a kind. We define $m(T)$ and $m(K)$ inductively as follows:

$$\begin{array}{rcl} m(\square) &=& 0 \\ m(T) &=& 0 \\ m(K) &=& \begin{cases} 1 & \text{if } K = \star \\ max(m(B), m(D)) + 1 & \text{if } K = (x:B).D \end{cases} \end{array}$$

It is easy to verify that, if $\Delta \vdash K \overset{*}{\leftrightarrow} K'$ then $m(K) = m(K')$. One can also check that $m(K[M/y]) = m(K)$.

Lemma 5.1.5
For every Δ and Δ-nonobject A, the set $\mathcal{C}_{\Delta,A}$ exists and is nonempty.

Proof
We define a canonical member $can_{\Delta,A}$ of $\mathcal{C}_{\Delta,A}$ by induction on $m(K)$, where $\Delta \vdash A : K$. At the same time we verify that $(can_{\Delta,A})|_{\Delta'} = can_{\Delta',A}$ for every $\Delta' \supseteq \Delta$ and that $can_{\Delta,A} = can_{\Delta,A'}$ for $\Delta \vdash A \overset{*}{\leftrightarrow} A'$.

If A is a $\square$, a kind, or a type, then $can_{\Delta,A}$ equals $\mathcal{SN}_{\Delta,A}$ and is nonempty because $\Delta \vdash \star : \square$ and $\Delta, x : A \vdash x : A$ for $A \neq \square$. It is easy to check that the corresponding properties ((S1) … (S3) from the definition of the family) are verified. Obviously, $(can_{\Delta,A})|_{\Delta'} = \mathcal{SN}_{\Delta,A}|_{\Delta'} = \mathcal{SN}_{\Delta',A} = can_{\Delta',A}$. It is also easy to check that $can_{\Delta,A} = can_{\Delta,A'}$ for $\Delta \vdash A \overset{*}{\leftrightarrow} A'$.

If $\Delta \vdash A : (x:B).D$ then $can_{\Delta,A}$ is a function. By induction hypothesis, for every M, such that $\Delta' \vdash M : B$ for some $\Delta' \supseteq \Delta$, $can_{\Delta',A\,M}$ is defined. If B is a kind, we set $can_{\Delta,A}(\Delta' \vdash M, C) = can_{\Delta',A\,M}$ for $\Delta' \vdash M : B$ and $C \in \mathcal{C}_{\Delta',M}$ and, if B is a type, $can_{\Delta,A}(\Delta' \vdash M) = can_{\Delta',A\,M}$. Assume that B is a kind.

(P1) is true by definition.

For (P2), we have to check that $can_{\Delta,A}(\Delta' \vdash M_1, C) = can_{\Delta,A}(\Delta' \vdash M_2, C)$, for $C \in \mathcal{C}_{\Delta',M_1}$ and $\Delta' \vdash M_1 \overset{*}{\leftrightarrow} M_2$. First, note that by the definition of the family $\mathcal{C}_{\Delta',M_1} = \mathcal{C}_{\Delta',M_2}$, which implies $C \in \mathcal{C}_{\Delta',M_2}$ and justifies the correctness of $can_{\Delta,A}(\Delta' \vdash M_2, C)$. By the definition of $can_{\Delta,A}$ it remains to show that $can_{\Delta',AM_1} = can_{\Delta',AM_2}$, which is true by induction hypothesis (as $\Delta' \vdash A\,M_1 \overset{*}{\leftrightarrow} A\,M_2$).

To see that (P3) holds, we have to show that $can_{\Delta,A}(\tilde{\Delta} \vdash M, C)|_{\Delta'} = can_{\Delta,A}(\Delta' \vdash M, C|_{\Delta'})$ whenever $\Delta \subseteq \tilde{\Delta} \subseteq \Delta'$, $\tilde{\Delta} \vdash M : B$ and $C \in \mathcal{C}_{\tilde{\Delta},M}$. By the definition of $can_{\Delta,A}$, $can_{\Delta,A}(\tilde{\Delta} \vdash M, C)|_{\Delta'}$ equals $(can_{\tilde{\Delta},AM})|_{\Delta'}$ and by induction hypothesis it is equal to $can_{\Delta',AM} = can_{\Delta,A}(\Delta' \vdash M, C|_{\Delta'})$.

To check that $(can_{\Delta,A})|_{\Delta'} = can_{\Delta',A}$, it is sufficient to unfold the definition of $can_{\Delta,A}$ and $can_{\Delta',A}$ and to notice that $(can_{\Delta,A\,M})|_{\Delta'} = can_{\Delta',A\,M}$ by induction hypothesis.

We are left to show that $can_{\Delta,A} = can_{\Delta,A'}$ for $\Delta \vdash A \overset{*}{\leftrightarrow} A'$. But for every $\Delta' \vdash M : B$, $C \in \mathcal{C}_{\Delta',M}$

$$can_{\Delta,A}(\Delta' \vdash M, C) = can_{\Delta',A\,M} = can_{\Delta',A'\,M} = can_{\Delta,A'}(\Delta' \vdash M, C)$$

because $can_{\Delta',A\,M} = can_{\Delta',A'\,M}$ by the induction hypothesis applied to $\Delta' \vdash A\,M \overset{*}{\leftrightarrow} A'\,M$. □

Lemma 5.1.6
Let C be a candidate. The sequent $\Delta \vdash @(\lambda x{:}a.b, c)$ belongs to C if $\Delta \vdash b[c/x] \in C$ and a, c are $\mathcal{SN}$.

Proof
By induction on $(\lambda x{:}a.b, c)$ ordered by $(\rightarrow, \rightarrow)_{lex}$ (note that $b \in \mathcal{SN}$, as $b[c/x] \in C$).

Since $@(\lambda x : a.b, c)$ is neutral, it is sufficient (by (S2) from the definition of candidates) to show that every reduct of $\Delta \vdash @(\lambda x{:}a.b, c)$ belongs to C. We have the following reducts:

- $\Delta \vdash @(\lambda x{:}a'.b, c)$, where $\Delta \vdash a \rightarrow a'$. Since $b[c/x] \in C$, by induction hypothesis $\Delta \vdash @(\lambda x{:}a'.b, c) \in C$,
- $\Delta \vdash @(\lambda x{:}a.b', c)$, where $\Delta, x : a \vdash b \rightarrow b'$. Obviously, $\Delta \vdash b[c/x] \rightarrow b'[c/x]$ and by (S1) from the definition of candidates $\Delta \vdash b'[c/x] \in C$. We conclude by induction hypothesis that $\Delta \vdash @(\lambda x{:}a.b', c) \in C$,
- $\Delta \vdash @(\lambda x{:}a.b, c')$, where $\Delta \vdash c \rightarrow c'$. By Lemma 4.2.2, $\Delta \vdash b[c/x] \rightarrow^* b[c'/x]$ and consequently $\Delta \vdash b[c'/x] \in C$. We conclude once again by induction hypothesis.
- $\Delta \vdash b[c/x]$, which belongs to C by assumption. □

Lemma 5.1.7
Let Δ be an environment, A a Δ-nonobject and C a candidate from $\mathcal{C}_{\Delta,A}$. For every variable x and strongly normalizing Δ-terms $\vec{N}$ if $\Delta \vdash x\vec{N} : A$ then $\Delta \vdash x\vec{N} \in C$.

Proof
By induction on $x\vec{N}$ ordered by $\rightarrow$.

Since $x\vec{N}$ is neutral, it is sufficient to show that every reduct of $\Delta \vdash x\vec{N}$ belongs to C. All immediate reducts of $x\vec{N}$ are of the form $x\vec{M}$, where $\Delta \vdash N_i \to M_i$ for some i and $N_j = M_j$ for $j \neq i$. By subject reduction we have $\Delta \vdash x\vec{M} : A$. And by induction hypothesis we get $\Delta \vdash x\vec{M} \in C$. $\square$

5.2 *Interpretation of types*

Definition 5.2.1 (*Candidate assignment*)
Given two valid environments Γ, Δ, and a well-typed substitution $\rho : \Gamma \to \Delta$, a *candidate assignment compatible with* ρ is a function ξ from $\mathcal{V}ar^{\square} \cap dom(\Gamma)$ to the set of candidates, such that, for every variable $x \in \mathcal{V}ar^{\square} \cap dom(\Gamma)$, $x\xi \in \mathcal{C}_{\Delta,x\rho}$.

Given $\Delta' \supseteq \Delta$, we will write $\xi|_{\Delta'}$ for a candidate assignment that associates $(x\xi)|_{\Delta'}$ to every $x \in \mathcal{V}ar^{\square} \cap dom(\Gamma)$.

Now we can define the interpretation $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ where Γ, Δ are environments, A is a Γ-nonobject, $\rho : \Gamma \to \Delta$ is a well-typed substitution and ξ is a candidate assignment compatible with ρ. The definition is by induction on the structure of A.

Definition 5.2.2
In the clauses below we use the following conventions: $K \in Kind$, $T \in Type$, $A, B \in Constr$, $D \in Kind \cup Type$, $M \in Constr \cup Obj$, $N \in Obj$ and $y \in \mathcal{V}ar^{\square}$.

$$
\begin{array}{rcl}
[\![\Gamma \vdash \square]\!]_{\rho,\xi,\Delta} & = & \mathcal{SN}_{\Delta,\square} \\
[\![\Gamma \vdash \star]\!]_{\rho,\xi,\Delta} & = & \mathcal{SN}_{\Delta,\star} \\
[\![\Gamma \vdash y]\!]_{\rho,\xi,\Delta} & = & y\xi \\
[\![\Gamma \vdash A\,B]\!]_{\rho,\xi,\Delta} & = & [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}(\Delta \vdash B\rho, [\![\Gamma \vdash B]\!]_{\rho,\xi,\Delta}) \\
[\![\Gamma \vdash A\,N]\!]_{\rho,\xi,\Delta} & = & [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}(\Delta \vdash N\rho) \\
[\![\Gamma \vdash (x\!:\!K).D]\!]_{\rho,\xi,\Delta} & = & \{\Delta' \vdash M \mid \Delta' \vdash M : ((x\!:\!K).D)\rho,\ \Delta' \supseteq \Delta, \text{ and} \\
 & & \forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''},\ \forall C \in \mathcal{C}_{\Delta'',A}, \\
 & & \Delta'' \vdash M\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C/x],\Delta''}\} \\
[\![\Gamma \vdash (x\!:\!T).D]\!]_{\rho,\xi,\Delta} & = & \{\Delta' \vdash M \mid \Delta' \vdash M : ((x\!:\!T).D)\rho,\ \Delta' \supseteq \Delta, \text{ and} \\
 & & \forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash N \in [\![\Gamma \vdash T]\!]_{\rho,\xi|_{\Delta''},\Delta''}, \\
 & & \Delta'' \vdash M\,N \in [\![\Gamma, x : T \vdash D]\!]_{\rho\cup[N/x],\xi|_{\Delta''},\Delta''}\} \\
[\![\Gamma \vdash \lambda x\!:\!K.B]\!]_{\rho,\xi,\Delta} & = & \lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x : K \vdash B]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'}, \\
 & & \text{a function with domain} \\
 & & \{(\Delta' \vdash A, C) \mid \Delta' \vdash A : K\rho,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',A}\} \\
[\![\Gamma \vdash \lambda x : T.B]\!]_{\rho,\xi,\Delta} & = & \lambda(\Delta' \vdash N).[\![\Gamma, x : T \vdash B]\!]_{\rho\cup[N/x],\xi|_{\Delta'},\Delta'}, \\
 & & \text{a function with domain} \\
 & & \{\Delta' \vdash N \mid \Delta' \vdash N : T\rho,\ \Delta' \supseteq \Delta\}
\end{array}
$$

A careful reader should notice that there is one case missing in the definition above: the interpretation for $s(\vec{M})$ where $s \in \mathcal{TC}$, and in consequence, for any Γ-nonobject containing s. At the same time we did not restrict ρ to assign only terms not containing s. Fortunately, there is no contradiction here, since in order to compute $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ we need to know only the interpretations for the subterms of A . Note also, that there is no problem with ξ, even if $x\rho$ contains s, because ξ uses arbitrary candidates from the family of candidates, and not the interpretation of $x\rho$. The interpretation for $s(\vec{M})$ where $s \in \mathcal{TC}$ will be defined in section 5.2.1.

Regarding restricted candidates $(\xi|_{\Delta'})$ that appear in the definition of interpretation of products and abstractions, it is obvious that when we replace Δ by Δ', such that $\Delta' \supseteq \Delta$, the interpretation $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ becomes $[\![\Gamma \vdash A]\!]_{\rho,\xi|_{\Delta'},\Delta'}$, since, by the definition of interpretations, candidate assignment must assign candidates from $\mathcal{C}_{\Delta',x\rho}$ to every big variable x. Therefore writing $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta'}$ is in principle not correct. Despite this, we will often use this simpler writing, since Δ' carries the information that a restriction is necessary for ξ.

To show that the interpretations defined above are not empty and are candidates we need a technical lemma.

Lemma 5.2.3 (*Equivalence of substitutions*)
Let Γ, Δ be environments, ρ, ρ' well-typed substitutions from Γ to Δ, such that $\Delta \vdash x\rho \overset{*}{\leftrightarrow} x\rho'$ for all $x \in dom(\Gamma)$, and ξ a candidate assignment compatible with ρ and ρ'. If A is a Γ-nonobject that does not contain symbols from $\mathcal{TC}$ then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash A]\!]_{\rho',\xi,\Delta}$.

Proof
By induction on the structure of A □

Lemma 5.2.4 (*Existence of interpretation*)
Let Γ, Δ be environments, A a Γ-nonobject that does not contain symbols from $\mathcal{TC}$, $\rho : \Gamma \to \Delta$ a well-typed substitution and ξ a candidate assignment compatible with ρ. Then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,A\rho}$.

Proof
We will prove the statement of the lemma in parallel with

$$[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}|_{\Delta'} = [\![\Gamma \vdash A]\!]_{\rho,\xi|_{\Delta'},\Delta'}$$

referred by (★). The proof is by induction on the structure of A.

1. If A is $\Box$, $\star$ or $y \in \mathcal{V}ar^{\Box}$ then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,A\rho}$ by the definition of $\mathcal{SN}_{\Delta,\Box}$, $\mathcal{SN}_{\Delta,\star}$ and the fact that ξ is the candidate assignment compatible with ρ.
Of course, if A is a $\Box$ or $\star$ then $\mathcal{SN}_{\Delta,A}|_{\Delta'} = \mathcal{SN}_{\Delta',A}$ and if A is a variable $(A\xi)|_{\Delta'} = A(\xi|_{\Delta'})$.
2. If $A = P\,Q$ we do the proof only for the case where $Q \in Constr$ (the proof for $Q \in Obj$ is very similar) .
Since A is a Γ-term, we have $\Gamma \vdash P : (x\!:\!B).D$ and $\Gamma \vdash Q : B$ for some B and D, by Stripping Lemma 4.1.3. By induction hypothesis $[\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,P\rho}$ and $[\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,Q\rho}$.
By the definition of the family of candidates $[\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta}$ is a function with the domain
$$\{(\Delta' \vdash M, C) \mid \Delta' \vdash M : B\rho,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',M}\}$$
and such that $[\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta}(\Delta' \vdash M, C) \in \mathcal{C}_{\Delta',P\rho M}$.

Thus, $[\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta}) \in \mathcal{C}_{\Delta,P\rho\, Q\rho}$ and consequently $[\![\Gamma \vdash P\ Q]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,(P\ Q)\rho}$.
Finally, $([\![\Gamma \vdash P\ Q]\!]_{\rho,\xi,\Delta})|_{\Delta'} = [\![\Gamma \vdash P\ Q]\!]_{\rho,\xi|_{\Delta'},\Delta'}$, because

$$\begin{aligned}([\![\Gamma \vdash P\ Q]\!]_{\rho,\xi,\Delta})|_{\Delta'} &= ([\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta}))|_{\Delta'} \\ &= ([\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta})|_{\Delta'}(\Delta' \vdash Q\rho, ([\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta})|_{\Delta'}) \qquad \text{by (P3)} \\ &= [\![\Gamma \vdash P]\!]_{\rho,\xi|_{\Delta'},\Delta'}(\Delta' \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi|_{\Delta'},\Delta'}) \qquad \text{by ind.hyp.} \\ &= [\![\Gamma \vdash P\ Q]\!]_{\rho,\xi|_{\Delta'},\Delta'}\end{aligned}$$

3. If $A = (x:K).D$ we do the proof only for the case where K is a kind. We have to check that the set

$$\begin{gathered}\{\Delta' \vdash M \mid \Delta' \vdash M\ :\ ((x:K).D)\rho,\ \Delta' \supseteq \Delta, \text{ and} \\ \forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''},\ \forall C \in \mathcal{C}_{\Delta'',A}, \\ \Delta'' \vdash M\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C/x],\Delta''}\}\end{gathered}$$

is a nonempty set of strongly normalizing terms and that it satisfies conditions (S1), … (S3) of Definition 5.1.2.
First note that by induction hypothesis $[\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''}$ and $[\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C/x],\Delta''}$ are candidates and consequently A and $M\,A$ are strongly normalizing. This also implies that M is strongly normalizing.
To see that the set defined above is nonempty consider $\Delta' = \Delta, y : ((x:K).D)\rho$. Then $\Delta' \vdash y\ :\ ((x:K).D)\rho$ and by Lemma 5.1.7 for every strongly normalizing A such that $\Delta'' \vdash A\ :\ K\rho$ and $\Delta'' \supseteq \Delta'$ we have $\Delta'' \vdash y\,A \in C'$ for any $C' \in \mathcal{C}_{\Delta'',D(\rho\cup[A/x])}$. In particular $\Delta'' \vdash y\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C/x],\Delta''}$ and consequently $\Delta' \vdash y \in [\![\Gamma \vdash (x:K).D]\!]_{\rho,\xi,\Delta}$.
For (S1), let us suppose that $\Delta' \vdash a \in [\![\Gamma \vdash (x:K).D]\!]_{\rho,\xi,\Delta}$ and let us take a' such that $\Delta' \vdash a \to a'$. We will show that $\Delta' \vdash a' \in [\![\Gamma \vdash (x:K).D]\!]_{\rho,\xi,\Delta}$.
By the definition of the interpretation of a product, it is sufficient that we prove $\Delta'' \vdash a'\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ for any Δ'', A and C such that $\Delta'' \supseteq \Delta'$, $\Delta'' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''}$, $C \in \mathcal{C}_{\Delta'',A}$. From the hypothesis that $\Delta' \vdash a \in [\![\Gamma \vdash (x : K).D]\!]_{\rho,\xi,\Delta}$ we have $\Delta'' \vdash a\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ Moreover, by induction hypothesis, we know that $[\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''} \in \mathcal{C}_{\Delta'',D(\rho\cup[A/x])}$. Thus, (S1) for $[\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ implies that $\Delta'' \vdash a'\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$.
To see that $[\![\Gamma \vdash (x:K).D]\!]_{\rho,\xi,\Delta}$ satisfies (S2) let us consider a neutral term a such that $\Delta' \vdash a\ :\ ((x : K).D)\rho$ for some $\Delta' \supseteq \Delta$ and let us suppose that every reduct of a belongs to $[\![\Gamma \vdash (x : K).D]\!]_{\rho,\xi,\Delta}$. We have to show that $\Delta' \vdash a \in [\![\Gamma \vdash (x:K).D]\!]_{\rho,\xi,\Delta}$, which is equivalent, by the definition of interpretation for a product, to $\Delta'' \vdash a\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ for any Δ'', A and C such that $\Delta'' \supseteq \Delta'$, $\Delta'' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''}$, $C \in \mathcal{C}_{\Delta'',A}$.
Since $a\,A$ is neutral and $[\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''} \in \mathcal{C}_{\Delta'',D(\rho\cup[A/x])}$ (by induction hypothesis), it is sufficient to check that every reduct of $(a\,A)$ belongs to $[\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ to get the desired conclusion. Let us prove it by induction on the reduction starting from A (A is $\mathcal{SN}$, because $\Delta'' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''}$). Since a is neutral, every reduct of $a\,A$ is of the form

- $a' \, A$, where a' is a reduct of a. Then $\Delta'' \vdash a' \, A \in [\![\Gamma, x : K \vdash D]\!]_{\rho \cup [A/x], \xi \cup [C/x], \Delta''}$ is an easy consequence of the assumption $\Delta' \vdash a' \in [\![\Gamma \vdash (x : K).D]\!]_{\rho, \xi, \Delta}$.
- $a \, A'$, where $\Delta'' \vdash A \to A'$. Then $\Delta'' \vdash a \, A' \in [\![\Gamma, x : K \vdash D]\!]_{\rho \cup [A'/x], \xi \cup [C/x], \Delta''}$ follows from the induction hypothesis for A' and by Lemma 5.2.3 we get $\Delta'' \vdash a \, A' \in [\![\Gamma, x : K \vdash D]\!]_{\rho \cup [A/x], \xi \cup [C/x], \Delta''}$.

Condition (S3) trivially holds.
We are left to show that $([\![\Gamma \vdash (x : K).D]\!]_{\rho, \xi, \Delta})|_{\Delta'} = [\![\Gamma \vdash (x : K).D]\!]_{\rho, \xi|_{\Delta'}, \Delta'}$ for every $\Delta' \supseteq \Delta$. By the definition of interpretations:

$$\begin{aligned}
&([\![\Gamma \vdash (x : K).D]\!]_{\rho, \xi, \Delta})|_{\Delta'} = \\
&\qquad \{\Delta'' \vdash M \mid \Delta'' \vdash M \ : \ ((x : K).D)\rho, \ \Delta'' \supseteq \Delta' \supseteq \Delta, \text{ and} \\
&\qquad \forall \Delta''' \supseteq \Delta'', \ \forall \Delta''' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho, \xi|_{\Delta'''}, \Delta'''}, \ \forall C \in \mathcal{C}_{\Delta''', A}, \\
&\qquad \Delta''' \vdash M \, A \in [\![\Gamma, x : K \vdash D]\!]_{\rho \cup [A/x], \xi|_{\Delta'''} \cup [C/x], \Delta'''}\}
\end{aligned}$$

$$\begin{aligned}
&[\![\Gamma \vdash (x : K).D]\!]_{\rho, \xi|_{\Delta'}, \Delta'} = \\
&\qquad \{\Delta'' \vdash M \mid \Delta'' \vdash M \ : \ ((x : K).D)\rho, \ \Delta'' \supseteq \Delta', \text{ and} \\
&\qquad \forall \Delta''' \supseteq \Delta'', \ \forall \Delta''' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho, (\xi|_{\Delta'})|_{\Delta'''}, \Delta'''}, \ \forall C \in \mathcal{C}_{\Delta''', A}, \\
&\qquad \Delta''' \vdash M \, A \in [\![\Gamma, x : K \vdash D]\!]_{\rho \cup [A/x], (\xi|_{\Delta'})|_{\Delta'''} \cup [C/x], \Delta'''}\}
\end{aligned}$$

Since $(\xi|_{\Delta'})|_{\Delta'''} = \xi|_{\Delta'''}$ (see Remark 2 after the definition of candidates), the two interpretations given above are equal.

4. If $A = \lambda x : K.B$ we do the proof only for the case where K is a kind. By the definition

$$[\![\Gamma \vdash \lambda x : K.B]\!]_{\rho, \xi, \Delta} = \lambda(\Delta' \vdash A) \lambda C. [\![\Gamma, x : K \vdash B]\!]_{\rho \cup [A/x], \xi|_{\Delta'} \cup [C/x], \Delta'},$$

a function with domain

$$\{(\Delta' \vdash A, C) \mid \Delta' \vdash A \ : \ K\rho, \ \Delta' \supseteq \Delta, \ C \in \mathcal{C}_{\Delta', A}\}$$

and we have to check that $[\![\Gamma \vdash \lambda x : K.B]\!]_{\rho, \xi, \Delta}$ satisfies the condition described in case 4(a) of the definition of the family of candidates (because by Stripping Lemma 4.1.3, $\Gamma \vdash \lambda x : K.B \ : \ (x : K).D$ for some D).
It is clear that $[\![\Gamma \vdash \lambda x : K.B]\!]_{\rho, \xi, \Delta}$ has an adequate domain and codomain, as $[\![\Gamma, x : K \vdash B]\!]_{\rho \cup [A/x], \xi|_{\Delta'} \cup [C/x], \Delta'} \in \mathcal{C}_{\Delta', B(\rho \cup [A/x])}$ by induction hypothesis and $\mathcal{C}_{\Delta', B(\rho \cup [A/x])} = \mathcal{C}_{\Delta', (\lambda x K.B)\rho \, A}$ by the definition of the family of candidates.
For (P2) we have to show that $[\![\Gamma, x : K \vdash B]\!]_{\rho \cup [A/x], \xi|_{\Delta'} \cup [C/x], \Delta'} = [\![\Gamma, x : K \vdash B]\!]_{\rho [A'/x], \xi|_{\Delta'} \cup [C/x], \Delta'}$ whenever $\Delta' \vdash A \overset{*}{\leftrightarrow} A'$. But this follows from the previous lemma (Lemma 5.2.3).
For (P3) it is sufficient to notice that

$$\begin{aligned}
([\![\Gamma \vdash \lambda x : K.B]\!]_{\rho, \xi, \Delta}(\tilde{\Delta} \vdash M, C))|_{\Delta'} &= ([\![\Gamma, x : K \vdash B]\!]_{\rho [M/x], \xi|_{\tilde{\Delta}} \cup [C/x], \tilde{\Delta}})|_{\Delta'} \\
&= [\![\Gamma, x : K \vdash B]\!]_{\rho [M/x], \xi|_{\Delta'} \cup [C|_{\Delta'}/x], \Delta'} \\
&= [\![\Gamma \vdash \lambda x : K.B]\!]_{\rho, \xi, \Delta}(\Delta' \vdash M, C|_{\Delta'})
\end{aligned}$$

where the second equality holds by induction hypothesis for the ($\bigstar$) property.
We are left to show the ($\bigstar$) property. By the definition of interpretation

and restriction:

$$(\llbracket\Gamma \vdash \lambda x:K.B\rrbracket_{\rho,\xi,\Delta})|_{\Delta'} = \lambda(\Delta'' \vdash A)\lambda C.\llbracket\Gamma, x : K \vdash B\rrbracket_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C/x],\Delta''},$$
a function with domain
$$\{(\Delta'' \vdash A, C) \mid \Delta'' \vdash A : K\rho,\ \Delta \subseteq \Delta' \subseteq \Delta'',\ C \in \mathcal{C}_{\Delta'',A}\}$$
$$\llbracket\Gamma \vdash \lambda x:K.B\rrbracket_{\rho,\xi|_{\Delta'},\Delta'} = \lambda(\Delta'' \vdash A)\lambda C.\llbracket\Gamma, x : K \vdash B\rrbracket_{\rho\cup[A/x],(\xi|_{\Delta'})|_{\Delta''}\cup[C/x],\Delta''},$$
a function with domain
$$\{(\Delta'' \vdash A, C) \mid \Delta'' \vdash A : K\rho,\ \Delta' \subseteq \Delta'',\ C \in \mathcal{C}_{\Delta'',A}\}$$

Since $(\xi|_{\Delta'})|_{\Delta''} = \xi|_{\Delta''}$ (this follows easily from the definition of candidates), both functions are equal and we are done. □

In the sequel, we will often write $\llbracket\Gamma \vdash A\rrbracket_{\rho,\xi,\Delta}$ for a candidate assignment ξ that is defined only on $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box}$, even though formally ξ should be given on all variables from $\Gamma \cap \mathcal{V}ar^{\Box}$. The following lemma justifies this writing, saying that ξ may be extended arbitrarily (for example by canonical candidates).

Lemma 5.2.5
Let Γ, Δ be environments, A a Γ-nonobject and ρ a well-typed substitution from Γ to Δ. Moreover, let us suppose that $x \in dom(\Gamma) \cap \mathcal{V}ar^{\Box}$, $x \notin \mathcal{FV}(A)$. Then for every ξ_1, ξ_2, candidate assignments compatible with ρ, that differ only on that x, $\llbracket\Gamma \vdash A\rrbracket_{\rho,\xi_1,\Delta} = \llbracket\Gamma \vdash A\rrbracket_{\rho,\xi_2,\Delta}$.

5.2.1 *Interpretation of inductive types*

To define the interpretation of $s(\vec{a})$, $s \in \mathcal{TC}$, we will first define the interpretation of s, as a least fixpoint of a monotone operator on a complete lattice. Intuitively, the interpretation of an n-ary type constructor s is a function, which given n arguments, returns the subset U of strongly normalizing terms, such that $M \in U$ iff the fact that M rewrites to $c(\vec{N})$, c being a constructor of s, implies that every N_i belongs to the interpretation of its type. Hence, the lattice to be used is the lattice of functions from vectors of terms to the subsets of strongly normalizing terms and it is not difficult to guess that the monotonicity of the operator to construct will follow from the positivity condition imposed on constructors.

Inductive definitions (inductive types and their constructors) were introduced to the signature in some order. The same order will be used to define the interpretations of inductive types. For this reason, during the definition of the interpretation of $s \in \mathcal{TC}$ we suppose that the interpretations of all terms that contain inductive types introduced to Σ beforehand, are given.

Note that, even though formally $s \in \mathcal{TC}^r$ alone is not a valid pseudoterm (it needs r arguments), we will still use the notation $\llbracket\Gamma \vdash s\rrbracket_{\rho,\xi,\Delta}$ to denote the interpretation of s. All the definitions below assume that we want to define $\llbracket\Gamma \vdash s\rrbracket_{\rho,\xi,\Delta}$ for s that is a part of the inductive definition $Ind[\Gamma_p](s : A := c_1 : C_1, \dots c_m : C_m) \in \Sigma$. In this case the type of s equals $(p_1 : P_1)\dots(p_r : P_r).(x_1 : b_1)\dots(x_n : b_n).\star$ and the type of the constructor c_i equals $(p_1 : P_1)\dots(p_r : P_r).(z^i_1 : d^i_1)\dots(z^i_{k_i} : d^i_{k_i}).s(\vec{p})w^i_1 \dots w^i_n$. Recall that, according to our conventions, the first r arguments of s and c_i are parameters.

Let us start with the domain of functions, that are elements of the lattice $L^{\rightarrow}$. It is a set of vectors of pairs, each pair consisting of a sequent and (possibly) a reducibility candidate.

$$\begin{gathered}
Dom = \{(\Delta \vdash a_1, U_1), \dots (\Delta \vdash a_r, U_r), (\nabla^1 \vdash t_1, W_1), \dots (\nabla^n \vdash t_n, W_n) \\
\Delta = \nabla^0 \subseteq \dots \subseteq \nabla^1 \dots \subseteq \nabla^n \\
\forall\, i = 1, \dots r \quad \Delta \vdash a_i \ : \ P_i[a_1/p_1, \dots, a_{i-1}/p_{i-1}], \\
U_i = \emptyset \text{ if } a_i \in Obj \text{ and } U_i \in \mathcal{C}_{\Delta, a_i} \text{ otherwise} \\
\forall\, j = 1, \dots n \quad \nabla^j \vdash t_j \ : \ b_j[\vec{a}/\vec{p}][t_1/x_1, \dots, t_{j-1}/x_{j-1}] \\
W_j = \emptyset \text{ if } t_j \in Obj \text{ and } W_j \in \mathcal{C}_{\nabla^j, t_j} \text{ otherwise}\}
\end{gathered}$$

For $r + n = 0$ let us define Dom as the singleton $\{\cdot\}$.

By $Bot(e)$, $e \in Dom$ we denote the set needed to construct the bottom of the lattice. It is a set of sequents that do not reduce to constructor headed form. Note that these sequents are defined in the environment ∇^n ($\nabla^0 = \Delta$).

$$\begin{gathered}
Bot((\Delta \vdash a_1, U_1), \dots (\Delta \vdash a_r, U_r), (\nabla^1 \vdash t_1, W_1), \dots (\nabla^n \vdash t_n, W_n)) = \\
\{\Delta' \vdash M \in \mathcal{SN}_{\nabla^n, s(\vec{a})\vec{t}} \ \mid \ \Delta' \vdash M \text{ do not reduce to a constructor headed term}\}
\end{gathered}$$

Definition 5.2.6 (*Lattice to compute* $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$)

$$L^{\rightarrow} = \{\varphi : Dom \mapsto \mathcal{P}(\mathcal{SN}) \ \mid \forall e = \langle \overrightarrow{(\Delta \vdash a, U)}, \overrightarrow{(\nabla \vdash t, W)} \rangle \in Dom \qquad Bot(e) \subseteq \varphi(e) \subseteq \mathcal{SN}_{\nabla^n, s(\vec{a})\vec{t}}\}$$

where $\mathcal{P}(\mathcal{SN})$ stands for the powerset of $\mathcal{SN}$.

Lemma 5.2.7
The set $L^{\rightarrow}$ is a complete lattice.

Proof
The minimal element of $L^{\rightarrow}$ is a function which assigns $Bot(e)$ to every $e \in Dom$. The lowest upper bound of φ_i, $i \in I$, is a function which assigns $\bigcup_{i \in I} \varphi_i(e)$ to every $e \in Dom$. A function $\varphi \in L^{\rightarrow}$ is bigger than a function φ' if and only if for every $e \in Dom$ we have $\varphi'(e) \subseteq \varphi(e)$. □

Now, it is time to define the operator F, the least fixpoint of which will be assigned to $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$. The target interpretation of $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$ will be a function, which, given an element from Dom, returns a set containing terms such that if they reduce to $c_i(\vec{a'}, \vec{N})$, then every a'_j and N_k belongs to the interpretation of its type, which may itself depend on $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$.

The operator F makes one step of this recursive definition. It takes $\varphi \in L^{\rightarrow}$ – an "old" interpretation of s – and returns a new one, calculated using φ. More precisely, it returns a function that for every $e \in Dom$ gives the set of terms such that if they reduce to $c_i(\vec{a'}, \vec{N})$, then every a'_j and N_k belongs to the interpretation of its type, calculated with the assumption that s has the interpretation φ. The latter calculation is performed by the operator $Inter(\varphi)$, which takes an environment, a type, a substitution, a candidate assignment and another environment (just like $[\![. \vdash .]\!]_{.,.,.}$).

Recall, that $s : (p_1 : P_1) \dots (p_r : P_r).(x_1 : b_1) \dots (x_n : b_n).\star$ and $c_i : (p_1 : P_1) \dots (p_r : P_r).(z^i_1 : d^i_1) \dots (z^i_{k_i} : d^i_{k_i}).s(\vec{p}) w^i_1 \dots w^i_n$ for every c_i that is a constructor of s.

Definition 5.2.8 (*Monotone operator to compute* $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$)
The monotone operator F to compute $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$ is defined as follows:

$$\begin{aligned} F(\varphi) = \lambda \langle \overrightarrow{(\Delta \vdash a, U)}, \overrightarrow{(\nabla \vdash t, W)} \rangle . \{ \Delta' \vdash M \in \mathcal{SN}_{\nabla^n, s(\vec{a})\vec{t}} \ \mid \\ \text{if } \ \Delta' \vdash M \rightarrow^* c_i(\vec{a}', \vec{N}) \ c_i \text{ being a constructor of s, then} \\ \Delta' \vdash a'_j \in Inter(\varphi)(p_1 : P_1, \dots p_{j-1} : P_{j-1} \vdash P_j)_{[\vec{a}/\vec{p}],[\vec{U}/\vec{p}],\Delta'} \text{ and} \\ \Delta' \vdash N_j \in Inter(\varphi)(\overrightarrow{p : P}, z^i_1 : d^i_1, \dots z^i_{j-1} : d^i_{j-1} \vdash d^i_j)_{[\vec{a}/\vec{p}] \cup [\vec{N}/\vec{z^i}],[\vec{U}/\vec{p}],\Delta'} \} \end{aligned}$$

and $Inter(\varphi)(\hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}}$ is defined by induction on the structure of A in the following way:

1. if $s \notin A$, then $Inter(\varphi)(\hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = [\![\hat{\Gamma} \vdash A]\!]_{\hat{\rho},\hat{\xi},\hat{\Delta}}$,
2. if $A = s(p)v_1, \dots v_n$ then $Inter(\varphi)(\hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = \varphi((\hat{\Delta} \vdash p_1\hat{\rho}, p_1\hat{\xi}), \dots (\hat{\Delta} \vdash p_r\hat{\rho}, p_r\hat{\xi}), (\hat{\Delta} \vdash v_1\hat{\rho}, V_1), \dots (\hat{\Delta} \vdash v_n\hat{\rho}, V_n))$, where $V_i = [\![\hat{\Gamma} \vdash v_i]\!]_{\hat{\rho},\hat{\xi},\hat{\Delta}}$ if $v_i \notin Obj$ and $V_i = \emptyset$ otherwise.
3. if $A = (y : T).B$ and $s \notin T$ then
   (a) if T is a kind then

$$\begin{aligned} Inter(\varphi)(\hat{\Gamma} \vdash (y : T).B)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = \{ \hat{\Delta}' \vdash M \mid \hat{\Delta}' \vdash M \ : ((y : T).B)\hat{\rho}, \\ \hat{\Delta}' \supseteq \hat{\Delta}, \text{ and } \ \forall \hat{\Delta}'' \supseteq \hat{\Delta}' \ \ \forall \hat{\Delta}'' \vdash N \in [\![\hat{\Gamma} \vdash T]\!]_{\hat{\rho},\hat{\xi}|_{\hat{\Delta}''},\hat{\Delta}''} \ \ \forall C \in \mathcal{C}_{\hat{\Delta}'',N}, \\ \hat{\Delta}'' \vdash M\,N \in Inter(\varphi)((\hat{\Gamma}, y : T) \vdash B)_{\hat{\rho} \cup [N/x],\hat{\xi}|_{\hat{\Delta}''} \cup [C/x],\hat{\Delta}''} \} \end{aligned}$$

   (b) if T is a type then

$$\begin{aligned} Inter(\varphi)(\hat{\Gamma} \vdash (y : T).B)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = \{ \hat{\Delta}' \vdash M \mid \hat{\Delta}' \vdash M \ : ((y : T).B)\hat{\rho}, \\ \hat{\Delta}' \supseteq \hat{\Delta}, \text{ and } \ \forall \hat{\Delta}'' \supseteq \hat{\Delta}' \ \ \forall \hat{\Delta}'' \vdash N \in [\![\hat{\Gamma} \vdash T]\!]_{\hat{\rho},\hat{\xi}|_{\hat{\Delta}''},\hat{\Delta}''}, \\ \hat{\Delta}'' \vdash M\,N \in Inter(\varphi)((\hat{\Gamma}, y : T) \vdash B)_{\hat{\rho} \cup [N/x],\hat{\xi}|_{\hat{\Delta}''},\hat{\Delta}''} \} \end{aligned}$$

Although $Inter$ is not completely defined, the part given above suffices to compute F. Indeed, by the definition of inductive type, the types of parameters $(P_1, \dots P_r)$ do not contain s, and in the other input types of c_i, s occurs strictly positively.

Note that in the definition of F, $\Delta' \vdash a'_j$ has to belong to the interpretation computed in the presence of the candidate assignment which assigns to variables $\vec{p}$ candidates $\vec{U}$ and not $[\![\dots \vdash a'_j]\!]$ (which would not give a consistent definition). For this reason $[\![\dots \vdash N_j]\!]$ cannot appear in a candidate assignment part of $Inter$. This is achieved by restricting the form of d^i_j (recall that by definition of the well-formed signature the type of constructor satisfies star dependency condition, which means in our notations that no d^i_j depends on $z^i_m \in \mathcal{V}ar^{\Box}$).

Lemma 5.2.9
The operator F defined above is monotone on $L^{\rightarrow}$.

Proof
The monotonicity of F follows from the monotonicity of $Inter(\varphi)(\Gamma_p, \hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}}$, where Γ_p are parameters of the inductive type s, A is a term that has only strictly positive occurrences of s, $\hat{\rho} : (\Gamma_p, \hat{\Gamma}) \rightarrow \hat{\Delta}$ and $\hat{\xi}$ is a candidate assignment compatible with $\hat{\rho}$. We proceed by induction on the structure of A.

If $s \notin A$ then $Inter(\varphi)(\Gamma_p, \hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = [\![\Gamma_p, \hat{\Gamma} \vdash A]\!]_{\hat{\rho},\hat{\xi},\hat{\Delta}}$ and the monotonicity is obvious. Otherwise, suppose that A has the form $(y_1 : T_1)\dots(y_k : T_k).s(\vec{p})v_1, \dots v_n$ where s does not occur in any v_i, T_i and $\vec{p} = dom(\Gamma_p)$.

If $A = s(\vec{p})v_1, \dots v_n$ then $Inter(\varphi)(\Gamma_p, \hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = \varphi(\overrightarrow{(\hat{\Delta} \vdash p\hat{\rho}, p\hat{\xi})}, \overrightarrow{(\hat{\Delta} \vdash v\hat{\rho}, V)})$, where $V_i = [\![\Gamma_p, \hat{\Gamma} \vdash v_i]\!]_{\hat{\rho},\hat{\xi},\hat{\Delta}}$ if $v_i \notin Obj$ and $V_i = \emptyset$ otherwise. The whole expression is monotone by the definition of the order relation on $L^{\rightarrow}$.

If $A = (y : T).A'$ and T is a kind, then $s \notin T$ and s occurs strictly positively in A'. We have

$$\begin{aligned} Inter(\varphi)(\Gamma_p, \hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = \{&\hat{\Delta}' \vdash M \mid \hat{\Delta}' \vdash M : ((y : T).A')\hat{\rho}, \\ &\hat{\Delta}' \supseteq \hat{\Delta}, \text{ and } \forall \hat{\Delta}'' \supseteq \hat{\Delta}' \ \forall \hat{\Delta}'' \vdash N \in [\![\Gamma_p, \hat{\Gamma} \vdash T]\!]_{\hat{\rho},\hat{\xi}|_{\hat{\Delta}''},\hat{\Delta}''} \ \forall C \in \mathcal{C}_{\hat{\Delta}'',N}, \\ &\hat{\Delta}'' \vdash M\,N \in Inter(\varphi)((\Gamma_p, \hat{\Gamma}, y : T) \vdash A')_{\hat{\rho}\cup[N/x],\hat{\xi}|_{\hat{\Delta}''}\cup[C/x],\hat{\Delta}''}\} \end{aligned}$$

and we know by induction hypothesis that $Inter(\varphi)(\Gamma_p, \hat{\Gamma}, y : T \vdash A')_{\hat{\rho}\cup[N/x],\hat{\xi}|_{\hat{\Delta}''}\cup[C/x],\hat{\Delta}''}$ is monotone. Hence $Inter(\varphi)(\Gamma_p, \hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}}$ is also monotone.

The same reasoning applies if $A = (y : T).A'$ and T is a type. □

Definition 5.2.10 (*Interpretation of an inductive type*)
Let Γ, Δ be environments, s an inductive type, $\rho : \Gamma \rightarrow \Delta$ a well-typed substitution, ξ a candidate assignment compatible with ρ and F an operator defined in definition 5.2.8. Then:

$$[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta} = \mu F$$

where μF is the least fixpoint of F.

Since the types of the constructors of s do not depend on an environment, we get the following corollary:

Corollary 5.2.11
The interpretation $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$ does not depend on Γ, ρ and ξ.

Now, that we have defined the interpretation of an inductive type s we can compute interpretations for every term of CC+H built from symbols introduced to Σ up to s (including s). We start by adding the following case to the definition of interpretation (5.2.2).

$$[\![\Gamma \vdash s(\vec{a})]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}(\Delta \vdash a_1\rho, U_1)\dots(\Delta \vdash a_r\rho, U_r)$$
$$\text{where } \ U_i = \begin{cases} [\![\Gamma \vdash a_i]\!]_{\rho,\xi,\Delta} & \text{if } a_i \text{ is a } \Gamma\text{-nonobject} \\ \emptyset & \text{otherwise} \end{cases}$$

We are left to show that $[\![\Gamma \vdash s(\vec{a})]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,s(\vec{a})\rho}$.

Lemma 5.2.12

Let s have type $(p_1 : P_1)\dots(p_r : P_r).(x_1 : b_1)\dots(x_n : b_n).\star$ and let μF be the least fixpoint of F used to compute the interpretation of s. Then, for every $(\Delta \vdash a_1, U_1), \dots (\Delta \vdash a_r, U_r)$, such that $\forall\ i = 1, \dots r$, $\Delta \vdash a_i : P_i[a_1/p_1, \dots, a_{i-1}/p_{i-1}]$ and $U_i = \emptyset$ if $a_i \in Obj$ and $U_i \in \mathcal{C}_{\Delta, a_i}$ otherwise, we have $\mu F(\Delta \vdash a_1, U_1), \dots (\Delta \vdash a_r, U_r) \in \mathcal{C}_{\Delta, s(\vec{a})}$ and $(\mu F\overrightarrow{(\Delta \vdash a, U)})|_{\Delta'} = \mu F\overrightarrow{(\Delta' \vdash a, U)})$ for $\Delta' \supseteq \Delta$.

Proof

We will show a more general property namely, that for every $(\nabla^1 \vdash t_1, W_1), \dots (\nabla^n \vdash t_n, W_n)$ if

- $\Delta = \nabla^0 \subseteq \nabla^1 \subseteq, \dots \nabla^n$ and
- for every $j = 1, \dots n$, $\nabla^j \vdash t_j : b_j[\vec{a}/\vec{p}][t_1/x_1, \dots, t_{j-1}/x_{j-1}]$ and
- for every $j = 1, \dots n$, $W_j = \emptyset$ if $t_j \in Obj$ and $W_j \in \mathcal{C}_{\nabla^j, t_j}$ otherwise,

then for every $i = 0, \dots n$ we have

- $\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \dots (\nabla^i \vdash t_i, W_i) \in \mathcal{C}_{\nabla^i, s(\vec{a})t_1, \dots t_i}$, and
- $(\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \dots (\nabla^i \vdash t_i, W_i))|_{\Delta'} = \mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1),$ $\dots (\nabla^{i-1} \vdash t_{i-1}, W_{i-1})(\Delta' \vdash t_i, W_i|_{\Delta'})$ for $\Delta' \supseteq \nabla^i$.

The proof is done by induction on i decreasing from n down to 0.

If all arguments are applied to μF then we have to check that the conditions (S1)..(S3) from Definition 5.1.2 of the family of candidates are verified. Let us denote by e the vector $\overrightarrow{(\Delta \vdash a, U)}\overrightarrow{(\nabla \vdash t, W)} \in Dom$. Note that by the definition of F, μFe consist only of strongly normalizing terms and that it is nonempty because unreducible terms like $\nabla^n, x : s(\vec{a})\vec{t} \vdash x$ belong to it.

To check (S1), suppose that $\Delta' \vdash M \in \mu Fe$, with $\Delta' \supseteq \nabla^n$, and $\Delta' \vdash M \to M'$. We will see that $\Delta' \vdash M' \in \mu Fe$

If M' does not rewrite to the constructor headed term, then $\Delta' \vdash M' \in \mu Fe$.

Otherwise, $\Delta' \vdash M' \to^* c_i(\vec{a'}, \vec{N})$, c_i being a constructor of s and s being of type $\overrightarrow{(p : P)}\overrightarrow{(z : d)}.s(\vec{p})\vec{w}$, and we have to check that $\Delta' \vdash a'_j \in Inter(\varphi)(\overrightarrow{p : P} \vdash P_j)_{[\vec{a}/\vec{p}], [\vec{U}/\vec{p}], \Delta'}$ and $\Delta' \vdash N_j \in Inter(\varphi)((\overrightarrow{p : P}, \overrightarrow{z : d}) \vdash d_j)_{[\vec{a}/\vec{p}][\vec{N}/\vec{z}], [\vec{U}/\vec{p}], \Delta'}$ Since, $\Delta' \vdash M' \to^* c_i(\vec{a'}, \vec{N})$ implies $\Delta' \vdash M \to^* c_i(\vec{a'}, \vec{N})$, the required conditions follow directly from the definition of $\Delta' \vdash M \in \mu Fe$.

To check (S2), let us suppose that every reduct M' of a neutral term $\Delta' \vdash M$ belongs to μFe. To show that $\Delta' \vdash M \in \mu Fe$ we have to look at every constructor headed term $c_i(\vec{a'}, \vec{N})$ to which M may rewrite and check some conditions for immediate subterms of $c_i(\vec{a'}, \vec{N})$ (see above). Since M is neutral it must have a reduct M', such that $\Delta' \vdash M \to M' \to^* c_i(\vec{a'}, \vec{N})$. Consequently, the validity of the required conditions follows from the validity of the same conditions for M'.

Condition (S3) obviously holds.

It is also easily seen that $(\mu F\overrightarrow{(\Delta \vdash a, U)}\overrightarrow{(\nabla \vdash t, W)})|_{\Delta'} = \mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \dots$ $(\nabla^{n-1} \vdash t_{n-1}, W_{n-1})(\Delta' \vdash t_n, W_n|_{\Delta'})$.

Suppose now, that we have proved the claim for k, and let us verify it for $k-1$. We have to show that $\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})$ verifies the condition 4(a) or 4(b) from the definition of the family. We will do the proof for the case where b_k is a kind, so we are interested in the condition 4(a).

First, note that $\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})$ is a function with a proper domain $(\{(\Delta' \vdash M, C) \mid \Delta' \vdash M : b_k[\vec{a}/\vec{p}][\vec{t}/\vec{x}],\ \Delta' \supseteq \nabla^{k-1} \text{ and } C \in \mathcal{C}_{\Delta',M}\})$ and a proper codomain, as $\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M, C) \in \mathcal{C}_{\Delta', s(\vec{a})t_1,\dots,t_{k-1}M}$ by induction hypothesis. To check (P2) it is sufficient to show that $\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M_1, C) = \mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M_2, C)$ for $\Delta' \vdash M_1 \overset{*}{\leftrightarrow} M_2$. This follows directly from the definition of F using the fact that $\mathcal{SN}_{\Delta,T_1} = \mathcal{SN}_{\Delta,T_2}$ for $\Gamma \vdash T_1 \overset{*}{\leftrightarrow} T_2$.

For (P3) we have to show that

$$(\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\tilde{\Delta} \vdash M_1, C))|_{\Delta'}$$
$$= \mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M_1, C|_{\Delta'})$$

for $\nabla^{k-1} \subseteq \tilde{\Delta} \subseteq \Delta'$, which is true by induction hypothesis.

It remains to check that $(\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1}))|_{\Delta'}$ equals $\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\Delta' \vdash t_{k-1}, W_{k-1}|_{\Delta'})$. But these are two function that wait for an argument of the form $(\Delta'' \vdash t, W)$ with $\Delta'' \supseteq \Delta'$ and then behave like $(\mu F(\overrightarrow{\Delta \vdash a, U})(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1}))(\Delta'' \vdash t, W)$, so they are equal. $\square$

We should now prove again Lemmas 5.2.3 and 5.2.4 for all Γ-nonobjects that may now contain symbols from $\mathcal{TC}$ up to s, s included.

After having repeated the above procedure for all inductive types from Σ we get the final interpretation. Let us summarize its most important properties concerning inductive types.

Lemma 5.2.13 (*Inductive Type Lemma*)

Let Γ, Δ be environments, ρ a well-typed substitution from Γ to Δ, and ξ a candidate assignment compatible with ρ. Let $s : (p_1 : P_1)\dots(p_r : P_r).(x_1 : b_1)\dots(x_n : b_n).\star$ be an inductive type and $c : (p_1 : P_1)\dots(p_r : P_r).(z_1 : d_1)\dots(z_k : d_k).s(\vec{p})w_1\dots w_n$ its constructor. Under the above assumptions the following facts hold:

Decomposition If $\vec{a}$, $\vec{a'}$, $\vec{N}$, $\vec{u}$ satisfy $\Delta \vdash c(\vec{a'}, \vec{N}) \in [\![\Gamma \vdash s(\vec{a})\vec{u}]\!]_{\rho,\xi,\Delta}$ then:

- $\Delta \vdash a'_i \in [\![\Gamma \vdash P_i[\vec{a}/\vec{p}]]\!]_{\rho,\xi,\Delta}$ for every $i = 1, \dots r$ and
- $\Delta \vdash N_i \in [\![\Gamma, \overrightarrow{z : d[\vec{a}/\vec{p}]} \vdash d_i[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{N}/\vec{z}],\xi,\Delta}$ for every $i = 1, \dots k$.

Composition If $\vec{a}$, $\vec{a'}$, $\vec{N}$, $\vec{u}$ satisfy:

- $\Delta \vdash a'_i \in [\![\Gamma \vdash P_i[\vec{a}/\vec{p}]]\!]_{\rho,\xi,\Delta}$ for every $i = 1, \dots r$ and
- $\Delta \vdash N_i \in [\![\Gamma, \overrightarrow{z : d[\vec{a}/\vec{p}]} \vdash d_i[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{N}/\vec{z}],\xi,\Delta}$ for every $i = 1, \dots k$ and
- $\vec{u}$ are Γ-terms and $\Delta \vdash c(\vec{a'}, \vec{N}) : (s(\vec{a})\vec{u})\rho$

then $\Delta \vdash c(\vec{a'}, \vec{N}) \in [\![\Gamma \vdash s(\vec{a})\vec{u}]\!]_{\rho,\xi,\Delta}$.

The following lemma is a version of Lemma 5.2.3, formulated for all terms of $CC + H$. We will refer to it in the next sections.

Lemma 5.2.14 (*Equivalence of substitutions*)
Let Γ, Δ be environments, ρ, ρ' well-typed substitutions from Γ to Δ, such that $\Delta \vdash x\rho \overset{*}{\leftrightarrow} x\rho'$ for all $x \in dom(\Gamma)$, and ξ a candidate assignment compatible with ρ and ρ'. If A is a Γ-nonobject then $\llbracket \Gamma \vdash A \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma \vdash A \rrbracket_{\rho',\xi,\Delta}$.

5.3 *Auxiliary lemmas*

This subsection presents some technical lemmas that will be used in the sections that follow.

Lemma 5.3.1 (*Extension of* Γ)
Let (Γ_1, Γ_2) and Δ be environments, $\rho : (\Gamma_1, \Gamma_2) \to \Delta$ a well-typed substitution and ξ a candidate assignment compatible with ρ. If A is a (Γ_1, Γ_2)-nonobject, then:

1. $\llbracket \Gamma_1, \Gamma_2 \vdash A \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma_1, x : \sigma, \Gamma_2 \vdash A \rrbracket_{\rho^+,\xi,\Delta}$ where $x \notin FV(A)$, σ is a type and $\rho^+ = \rho \cup [M/x]$ for some M satisfying $\Delta \vdash M : \sigma\rho$
2. $\llbracket \Gamma_1, \Gamma_2 \vdash A \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma_1, x : K, \Gamma_2 \vdash A \rrbracket_{\rho^+,\xi^+,\Delta}$ where $x \notin FV(A)$, K is a kind and $\rho^+ = \rho \cup [M/x]$, $\xi^+ = \xi \cup [C/x]$ for some M satisfying that $\Delta \vdash M : K\rho$ and some $C \in \mathcal{C}_{\Delta,M}$

Proof
By induction on the structure of A. □

Lemma 5.3.2 (*Substitution Property for Interpretations*)
Let $(\Gamma_1, x : T, \Gamma_2)$ and Δ be environments, A a $(\Gamma_1, x : T, \Gamma_2)$-nonobject and M a term satisfying $\Gamma_1 \vdash M : T$. Moreover let $\rho : (\Gamma_1, \Gamma_2[M/x]) \to \Delta$ be a well-typed substitution and ξ a candidate assignment compatible with ρ.

1. If $T = K$ is a kind then
$$\llbracket \Gamma_1, \Gamma_2[M/x] \vdash A[M/x] \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma_1, x : K, \Gamma_2 \vdash A \rrbracket_{\rho \cup [M\rho/x], \xi \cup [\llbracket \Gamma_1 \vdash M \rrbracket_{\rho,\xi,\Delta}/x], \Delta}$$
2. If $T = \sigma$ is a type then
$$\llbracket \Gamma_1, \Gamma_2[M/x] \vdash A[M/x] \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma_1, x : \sigma, \Gamma_2 \vdash A \rrbracket_{\rho \cup [M\rho/x], \xi, \Delta}$$

Proof
Induction on the structure of A. □

Lemma 5.3.3 (*Equivalence of environments*)
Let Γ, Γ' be environments such that $dom(\Gamma) = dom(\Gamma')$ and $\Gamma \vdash \Gamma(x) \overset{*}{\leftrightarrow} \Gamma'(x)$ for all $x \in dom(\Gamma)$. Let Δ be an environment, ρ a well-typed substitution such that $\rho : \Gamma \to \Delta$ and ξ a candidate assignment compatible with ρ. If A is a Γ-nonobject then $\llbracket \Gamma \vdash A \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma' \vdash A \rrbracket_{\rho,\xi,\Delta}$.

Proof
Induction on the structure of A. □

Lemma 5.3.4 (*Equivalence of types*)
Let Γ, Δ be environments, $\rho : \Gamma \to \Delta$ a well-typed substitution and ξ a candidate assignment compatible with ρ. For every Γ-nonobjects A and A', if $\Gamma \vdash A \overset{*}{\leftrightarrow} A'$ then $\llbracket \Gamma \vdash A \rrbracket_{\rho,\xi,\Delta} = \llbracket \Gamma \vdash A' \rrbracket_{\rho,\xi,\Delta}$.

Proof
Induction on the structure of A. □

5.4 Reducibility of function symbol headed terms

Using the candidate method to prove strong normalization, the objective is to show that every term belongs to the interpretation of its type (see Lemma 5.5.2 for the exact formulation). When passing from the pure Calculus of Constructions to the Calculus of Constructions with rewriting most proofs remain unchanged; what is really new is the case of function symbol headed term.

As it was already mentioned, it turns out that the only way to show that the function symbol headed term $f(\vec{t})$ belongs to the interpretation of its type is to check all its reducts. This is done in Fun Lemma 5.4.6 by inspecting all HORPO cases and reasoning by induction on the precedence and on reductions originating from $\vec{t}$, but the proof is technically difficult.

To explain the problem suppose that $f(\vec{t})$ rewrites to $g(\vec{u})$ in HORPO, with $f : \overrightarrow{(x : b)}.c$ and $g : \overrightarrow{(y : d)}.e$. We have to show that $g(\vec{u})$ belongs to the interpretation of the type of $f(\vec{t})$. In fact, the interpretation of the type of $f(\vec{t})$ is just $[\![\overrightarrow{(x : b)} \vdash c]\!]_{[\vec{t}/\vec{x}],\zeta,\Delta}$ where ζ assigns candidates to those $\vec{x}$ that belong to $\mathcal{V}ar^{\Box}$ and Δ is an environment in which $f(\vec{t})$ is well-typed.

Considering $g(\vec{u})$, all one can get from the induction hypothesis are statements of the form $g(\vec{u}) \in [\![\overrightarrow{(y : d)} \vdash e]\!]_{[\vec{u}/\vec{y}],\xi,\Delta}$ for some ξ. Here, two problems arise: how to find ξ and how to prove that $[\![\overrightarrow{(x : b)} \vdash c]\!]_{[\vec{t}/\vec{x}],\zeta,\Delta} = [\![\overrightarrow{(y : d)} \vdash e]\!]_{[\vec{u}/\vec{y}],\xi,\Delta}$. Both problems are solved thanks to the star dependency condition and the form of the HORPO judgments. The substitution ξ is just ζ, as all big $\vec{y}$ have to be parameters and the parameters of the right-hand side are included in the parameters of the left-hand side. The equality of interpretations is true by the Small Substitution Lemma 5.4.1 given in a paragraph below.

In two other parts of this section we give the Constructor Lemma and prove the Fun Lemma and Computable Closure Lemma.

5.4.1 Small substitution lemma

To understand the notion of small substitution we have to go back to the star dependency condition in section 2.4.1. A type $\overrightarrow{(p : P)}\overrightarrow{(x : b)}.c$ satisfies star dependency condition if all free and big variables from $\vec{b}$ and c belong to $\vec{p}$, which may be seen as a generalization of requirements that all $\vec{x}$ are small. A substitution which is defined only on those $\vec{x}$ (and is an identity on $\vec{p}$) will be informally called small.

Recall that the notation $\rho_1\rho_2$ denotes sequential composition of substitutions. The lemma below states that any two types, that are convertible after an application of small substitutions, have equal interpretations.

Lemma 5.4.1 (*Small Substitution Lemma*)
Let G_p, G_1, G_2, G be environments, μ_1, μ_2 be well-typed substitutions and T_1, T_2 terms in beta normal form, such that

1. $\mathcal{FV}(T_1) \cap \mathcal{V}ar^{\Box} \subseteq G_p$, $\mathcal{FV}(T_2) \cap \mathcal{V}ar^{\Box} \subseteq G_p$,
2. $\mu_1 : (G_p, G_1) \to (G_p, G)$, $\mu_2 : (G_p, G_2) \to (G_p, G)$ such that $\mu_1|_{G_p} = \mu_2|_{G_p} = id_{G_p}$,
3. $G_p, G_1 \vdash T_1 : \star/\Box$, $G_p, G_2 \vdash T_2 : \star/\Box$,

If $G_p, G \vdash T_1\mu_1 \overset{*}{\leftrightarrow} T_2\mu_2$ then for every environment Δ, well-typed substitution $\theta : (G_p, G) \to \Delta$ and a candidate assignment ξ compatible with θ, we have $[\![G_p, G_1 \vdash T_1]\!]_{\mu_1\theta,\xi,\Delta} = [\![G_p, G_2 \vdash T_2]\!]_{\mu_2\theta,\xi,\Delta}$. Moreover, if T_1, T_2 are terms of the Calculus of Constructions ($G_p, G_1 \vdash_{CC} T_1 : \star/\Box$, $G_p, G_2 \vdash_{CC} T_2 : \star/\Box$) the same holds without the assumption that T_1, T_2 are in beta normal form.

Proof

The proof is done by induction on the structure of T_1. Because of possible λ's and Π's in T_1 we have to show a more general property, namely that for every n and vectors $\vec{A}, \vec{B}, \vec{\Delta}, \vec{N}, \vec{C}$ of length n, such that $\vec{A}, \vec{B}, \vec{N}$ are terms, $\vec{\Delta}$ are environments and $\vec{C}$ are candidates verifying for all $i = 1, \dots n$

- $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash A_i\mu_1 \overset{*}{\leftrightarrow} B_i\mu_2$ and
- $\Delta_0 = \Delta,\;\; \Delta_i \supseteq \Delta_{i-1}$,
- $\Delta_i \vdash N_i \in [\![G_p, G_1, \overrightarrow{(y : A)} \vdash A_i]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],\xi\cup[\vec{C}/\vec{y}],\Delta^{i-1}}$,
- $C_i \in \mathcal{C}_{\Delta_i,N_i}$ if N_i is not an object and $C_i = \emptyset$ otherwise,

if $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash T_1\mu_1 \overset{*}{\leftrightarrow} T_2\mu_2$ then $[\![G_p, G_1, \overrightarrow{(y : A)} \vdash T_1]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$ $= [\![G_p, G_2, \overrightarrow{(y : B)} \vdash T_2]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$.

The proof is by induction on the structure of T_1.

1. $T_1 = X$, $X \in \{\star, \Box\} \cup \mathcal{V}ar^{\Box}$. Since $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash T_1\mu_1 \overset{*}{\leftrightarrow} T_2\mu_2$ and μ_1, μ_2 are small, $T_2 = X$ and our claim is proved.
2. $T_1 = P_1\,Q_1$. Note that P_1 is neither an abstraction (T_1 is beta normal form) nor a variable which may become an abstraction (in this case $P_1 \in \mathcal{V}ar^{\Box}$ and $P_1\mu = P_1$). Therefore T_2 must be equal to $P_2\,Q_2$ with $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash P_1\mu_1 \overset{*}{\leftrightarrow} P_2\mu_2$, $\;G_p, G, \overrightarrow{(y : A\mu_1)}, x : P_1\mu_1 \vdash Q_1\mu_1 \overset{*}{\leftrightarrow} Q_2\mu_2$.
   By induction hypothesis we have
   $$[\![G_p, G_1, \overrightarrow{(y : A)} \vdash P_1]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n} = [\![G_p, G_2, \overrightarrow{(y : B)} \vdash P_2]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$$
   and if Q_1, Q_2 are nonobjects we have also
   $$[\![G_p, G_1, \overrightarrow{(y : A)} \vdash Q_1]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n} = [\![G_p, G_2, \overrightarrow{(y : B)} \vdash Q_2]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$$
   By the definition of interpretation of an application, this gives us the desired conclusion.
3. $T_1 = (x : P_1).Q_1$. Then $T_2 = (x : P_2).Q_2$ and $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash P_1\mu_1 \overset{*}{\leftrightarrow} P_2\mu_2$, $G_p, G, \overrightarrow{(y : A\mu_1)}, x : P_1\mu_1 \vdash Q_1\mu_1 \overset{*}{\leftrightarrow} Q_2\mu_2$.
   By induction hypothesis we have
   $$[\![G_p, G_1, \overrightarrow{(y : A)} \vdash P_1]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n} = [\![G_p, G_2, \overrightarrow{(y : B)} \vdash P_2]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$$

$$[\![G_p, G_1, \overrightarrow{(y : A)}, x : P_1 \vdash Q_1]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}]\cup[N'/x],(\xi\cup[\vec{C}/\vec{y}]\cup[C'/x])|_{\Delta_{n+1}},\Delta_{n+1}} =$$
$$[\![G_p, G_2, \overrightarrow{(y : B)}, x : P_2 \vdash Q_2]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}]\cup[N'/x],(\xi\cup[\vec{C}/\vec{y}]\cup[C'/x])|_{\Delta_{n+1}},\Delta_{n+1}}$$

for any $\Delta_{n+1} \vdash N' \in [\![G_p, G_1, \overrightarrow{(y : A)} \vdash P_1]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$, $\Delta_{n+1} \supseteq \Delta_n$, $C' \in \mathcal{C}_{\Delta_{n+1},N'}$. By the definition of interpretation of a product type, this gives the desired conclusion.

4. $T_1 = \lambda x : P_1.Q_1$. Then $T_2 = \lambda x : P_2.Q_2$ and $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash P_1\mu_1 \overset{*}{\leftrightarrow} P_2\mu_2$, $G_p, G, \overrightarrow{(y : A\mu_1)}, x : P_1\mu_1 \vdash Q_1\mu_1 \overset{*}{\leftrightarrow} Q_2\mu_2$.
The proof for this case is very similar to the proof given above.
5. $T_1 = s(\vec{M})$ and $s \in \mathcal{TC}$. Then $T_2 = s(\vec{N})$ and for every i we have $G_p, G, \overrightarrow{(y : A\mu_1)} \vdash \vec{M}_i\mu_1 \overset{*}{\leftrightarrow} \vec{N}_i\mu_2$.
By induction hypothesis, $[\![G_p, G_1, \overrightarrow{(y : A)} \vdash M_i]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n} = [\![G_p, G_2, \overrightarrow{(y : B)} \vdash N_i]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$ for every M_i that is not an object. To complete the proof it suffices to check whether $[\![G_p, G_1, \overrightarrow{(y : A)} \vdash s]\!]_{\mu_1\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n} = [\![G_p, G_2, \overrightarrow{(y : B)} \vdash s]\!]_{\mu_2\theta\cup[\vec{N}/\vec{y}],(\xi\cup[\vec{C}/\vec{y}])|_{\Delta_n},\Delta_n}$. But this follows from the Corollary 5.2.11.

If T_1, T_2 are terms of the Calculus of Constructions ($G_p, G_1 \vdash_{CC} T_1 : \star/\Box$, $G_p, G_2 \vdash_{CC} T_2 : \star/\Box$) we can beta normalize them before applying the reasoning presented above. $\Box$

In fact a stronger version of Small Substitution Lemma holds; T_1, T_2 may depend on some big variables that are not parameters, but will be substituted by parameters. This lemma is needed only in some proofs concerning application case (I.6) of HORPO.

Lemma 5.4.2
Let G_p, G_1, G_2, G be environments, μ_1, μ_2 be well-typed substitutions and T_1, T_2 terms in beta normal form, such that for $i = 1, 2$

1. $\mu_i : (G_p, G_i) \to (G_p, G)$, such that $\mu_i|_{G_p} = id_{G_p}$ and if $x \in (\mathcal{FV}(T_i) \cap \mathcal{V}ar^{\Box})$ then $\mu_i(x) \in G_p$
2. $G_p, G_i \vdash T_i : \star/\Box$,

If $G_p, G \vdash T_1\mu_1 \overset{*}{\leftrightarrow} T_2\mu_2$ then for every environment Δ, well-typed substitutions $\theta : (G_p, G) \to \Delta$ and a candidate assignment ξ compatible with θ, we have $[\![G_p, G_1 \vdash T_1]\!]_{\mu_1\theta,\xi_1,\Delta} = [\![G_p, G_2 \vdash T_2]\!]_{\mu_2\theta,\xi_2,\Delta}$, where $x\xi_i = x\xi$ for $x \in G_p$ and $x\xi_i = p\xi$ for $x \in \mathcal{V}ar^{\Box} \setminus G_p$ and such that $x\mu_i = p$. Moreover, if T_1, T_2 are terms of the Calculus of Constructions then the lemma holds, even if T_1, T_2 are not in beta normal form.

Proof
The proof of this new lemma is almost identical to the original one. $\Box$

5.4.2 Constructor subterm lemma

The following property is an easy consequence of Inductive Lemma 5.2.13. We decided to write it down, because it is formulated in terms of notations used in Small Substitution Lemma 5.4.1 and Fun Lemma 5.4.6.

Lemma 5.4.3 (*Constructor Lemma*)
Let $c : (q_1 : Q_1) \dots (q_r : Q_r).(z_1 : d_1) \dots (z_k : d_k).s(\vec{q})w_1 \dots w_n$ be a constructor of $s : (q_1 : Q_1) \dots (q_r : Q_r).(x_1 : b_1) \dots (x_n : b_n).\star$.

Moreover suppose that G_p, G_1, G are environments, μ_1 is a well-typed substitution and $s(\vec{a})\vec{u}$, $c(\vec{a'}, \vec{N})$ terms such that:

1. $\mathcal{FV}(s(\vec{a})\vec{u}) \cap \mathcal{V}ar^{\square} \subseteq G_p$,
2. $\mu_1 : (G_p, G_1) \rightarrow (G_p, G)$, such that $\mu_1|_{G_p} = id_{G_p}$,
3. $G_p, G_1 \vdash_{CC} s(\vec{a})\vec{u} : \star$,
4. $G_p, G \vdash_{CC} c(\vec{a'}, \vec{N}) : (s(\vec{a})\vec{u})\mu_1$.

For every environment Δ, well-typed substitution $\theta : G_p, G \rightarrow \Delta$ and candidate assignment ζ compatible with θ, if

$$\Delta \vdash c(\vec{a'}\theta, \vec{N}\theta) \in [\![G_p, G_1 \vdash s(\vec{a})\vec{u}]\!]_{\mu_1\theta,\zeta,\Delta}$$

then for every $t \in \{\vec{a'}, \vec{N}\}$, there are G_t, T_t and μ_t such that:

1. $\mathcal{FV}(T_t) \cap \mathcal{V}ar^{\square} \subseteq G_p$,
2. $\mu_t : (G_p, G_t) \rightarrow (G_p, G)$, such that $\mu_t|_{G_p} = id_{G_p}$,
3. $G_p, G_t \vdash_{CC} T_t : \star/\square$,
4. $G_p, G \vdash_{CC} t : T_t\mu_t$ and
5. $\Delta \vdash t\theta \in [\![G_p, G_t \vdash T_t]\!]_{\mu_t\theta,\zeta,\Delta}$.

Proof
It is sufficient to apply the Inductive Lemma 5.2.13 to the hypothesis to get

$$\Delta \vdash a'_i\theta \in [\![G_p, G_1 \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\mu_1\theta,\zeta,\Delta}$$

$$\Delta \vdash N_i\theta \in [\![G_p, G_1, \overrightarrow{z : d[\vec{a}/\vec{q}]} \vdash d_i[\vec{a}/\vec{q}]]\!]_{(\mu_1 \cup [\vec{N}/\vec{z}])\theta,\zeta,\Delta}$$

Since $\mathcal{FV}(s(\vec{a})\vec{u}) \cap \mathcal{V}ar^{\square} \subseteq G_p$ and the type of the constructor c satisfies star dependency condition, we have $\mathcal{FV}(Q_i[\vec{a}/\vec{q}]) \cap \mathcal{V}ar^{\square} \subseteq G_p$ and $\mathcal{FV}(d_i[\vec{a}/\vec{q}]) \cap \mathcal{V}ar^{\square} \subseteq G_p$.

Trivially, we have $G_p, G_1 \vdash_{CC} Q_i[\vec{a}/\vec{q}] : \star/\square$, $G_p, G_1, \overrightarrow{z : d[\vec{a}/\vec{q}]} \vdash_{CC} d_i[\vec{a}/\vec{q}] : \star/\square$ and $G_p, G \vdash_{CC} a'_i : Q_i[\vec{a}/\vec{q}]\mu_1$, $G_p, G \vdash_{CC} N_i : d_i[\vec{a}/\vec{q}](\mu_1 \cup [\vec{N}/\vec{z}])$. $\square$

Corollary 5.4.4 (*Constructor Subterm Lemma*)
Under the hypotheses of the previous lemma, for every environment Δ, well-typed substitution $\theta : G_p, G \rightarrow \Delta$ and candidate assignment ζ compatible with θ, if

$$\Delta \vdash c(\vec{a'}\theta, \vec{N}\theta) \in [\![G_p, G_1 \vdash s(\vec{a})\vec{u}]\!]_{\mu_1\theta,\zeta,\Delta}$$

then for every constructor subterm t of $c(\vec{a'}, \vec{N})$ there are G_t, T_t and μ_t such that:

1. $\mathcal{FV}(T_t) \cap \mathcal{V}ar^{\square} \subseteq G_p$,
2. $\mu_t : (G_p, G_t) \rightarrow (G_p, G)$, such that $\mu_t|_{G_p} = id_{G_p}$,
3. $G_p, G_t \vdash_{CC} T_t : \star/\square$,
4. $G_p, G \vdash_{CC} t : T_t\mu_t$ and
5. $\Delta \vdash t\theta \in [\![G_p, G_t \vdash T_t]\!]_{\mu_t\theta,\zeta,\Delta}$.

Proof
First, we will show that T_t is an inductive type if t is a constructor-headed term. From $G_p, G \vdash_{CC} t : T_t\mu_t$ it follows that $T_t\mu_t$ is an inductive type. But $\mathcal{FV}(T_t) \cap \mathcal{V}ar^{\square} \subseteq G_p$ and $\mu_t|_{G_p} = id_{G_p}$ and we conclude that T_t is an inductive type itself.

Repeating the Subterm Lemma as many times as needed we get the conclusion. $\square$

5.4.3 *Fun and computable closure Lemmas*

This subsection is devoted to one of the most important proofs in the paper, namely to the proof of the Fun Lemma. We will show that the reducibility of function symbol headed terms follows from the reducibility of their arguments. In this proof we will use all the machinery introduced before.

The proof of the Fun lemma will be done by inspecting all possible HORPO cases. Since HORPO depends on the computable closure, it is not surprising that there will be a similar lemma concerning tuples from the computable closure. Moreover, since computable closure uses HORPO, this lemma will refer to the induction hypothesis of the Fun Lemma. Hence, we decided to organize the section as follows: first we give the statement of the Fun Lemma, then we state and prove the lemma about the computable closure, and finally, we prove the Fun Lemma.

The following trivial lemma is needed to justify the induction used in the lemmas below.

Lemma 5.4.5
Let us denote by $\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}}$ the union of $\rightarrow^{\Delta}$ and constructor subterm relations.

If $\Gamma \vdash t \succ^{\mathcal{CS}} u$ and $\theta : \Gamma \rightarrow \Delta$ then $\Delta \vdash t\theta(\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})u\theta$. Moreover if $\Delta \vdash t$ is a term such that $\rightarrow^{\Delta}$ terminates starting from t, then $\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}}$ also terminates on t.

Lemma 5.4.6 (*Fun Lemma*)
Let f be a function symbol of type $(p_1 : P_1)\ldots(p_r : P_r).(x_1 : b_1)\ldots(x_n : b_n).c$ such that $\Delta \vdash f(\vec{\alpha}, \vec{\sigma}) : c[\vec{\alpha}/\vec{p}, \vec{\sigma}/\vec{x}]$ holds. For every ζ, a candidate assignment compatible with $[\vec{\alpha}/\vec{p}]$, if $\vec{\alpha}$, $\vec{\sigma}$ satisfy:

- $\Delta \vdash \alpha_j \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{[\vec{\alpha}/\vec{p}],\zeta,\Delta}$, for all $j = 1, \ldots r$,
- $\Delta \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta,\Delta}$, for all $j = 1, \ldots n$,

then $\Delta \vdash f(\vec{\alpha}, \vec{\sigma}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta,\Delta}$.

The proof of the Fun Lemma will be given at the end of the section.

Lemma 5.4.7 (*Computable Closure Lemma*)
Let f be a function symbol of the type $(p_1 : P_1)\ldots(p_r : P_r).(x_1 : b_1)\ldots(x_n : b_n).c$, such that $G_p, G \vdash f(\vec{p}, \vec{u}) : c[\vec{u}/\vec{x}]$ holds for some environments $G_p = \overrightarrow{p : P}$ and G and for some terms $\vec{u}$.

For every environment Δ, well-typed substitution $\theta : G_p, G \rightarrow \Delta$ and candidate assignment ζ compatible with θ, if

$$\Delta \vdash p_j\theta \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta,\Delta} \quad \text{for all } j = 1, \ldots r, \text{ and}$$
$$\Delta \vdash u_j\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta,\Delta} \quad \text{for all } j = 1, \ldots n$$

and if Lemma 5.4.6 holds for all $\Delta \vdash g(\vec{\beta}, \vec{v})$, smaller than $f(\vec{p}\theta, \vec{u}\theta)$ in $(>_{\mathcal{F}}, (\to^{\Delta} \cup \rhd^{\mathcal{CS}})_{stat})_{lex}$, where *stat* is the status of f, then for every $t \in CCl(G_p, G \vdash f(\vec{p}, \vec{u}))$ there exist G_t, T_t, μ_t satisfying:

1. $\mathcal{FV}(T_t) \cap \mathcal{V}ar^{\Box} \subseteq G_p$,
2. $\mu_t \ : \ G_p, G_t \to (G_p, G)$ such that $\mu_t|_{G_p} = id_{G_p}$,
3. $G_p, G_t \vdash_{CC} T_t \ : \ \star/\Box$,
4. $G_p, G \vdash_{CC} t \ : \ T_t\mu_t$ and
5. $\Delta \vdash t\theta \in [\![G_p, G_t \vdash T_t]\!]_{\mu_t\theta,\zeta,\Delta}$.

Proof
We actually prove a slightly more general statement.

We will show that for every $(t,\ G_t; BV \vdash T_t,\ \mu_t) \in CCl(G_p, G \vdash f(\vec{p}, \vec{u}))$, with $BV = (z_1 : A_1) \dots (z_m : A_m)$, and for every $\vec{N}$, $\vec{C}$ and $\Delta = \Delta_0 \subseteq \Delta_1 \dots \subseteq \Delta_m$ satisfying:

$$\forall i \quad \Delta_i \vdash N_i \in [\![G_p, G_t, BV \vdash A_i]\!]_{\mu_t\theta \cup [\vec{N}/\vec{z}], \zeta \cup [\vec{C}/\vec{z}], \Delta_i}$$
$$\forall i \quad C_i \in \mathcal{C}_{\Delta_i, N_i} \ \text{ if } N_i \text{ is not an object and } C_i = \emptyset \text{ otherwise}$$

the following properties hold :

1. $\mathcal{FV}(T_t) \cap \mathcal{V}ar^{\Box} \subseteq G_p \cup BV$,
2. $\mu_t \ : \ G_p, G_t \to (G_p, G)$ such that $\mu_t|_{G_p} = id_{G_p}$,
3. $G_p, G_t, BV \vdash_{CC} T_t \ : \ \star/\Box$,
4. $G_p, G, BV\mu_t \vdash_{CC} t \ : \ T_t\mu_t$,
5. $\Delta_m \vdash t(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_t, BV \vdash T_t]\!]_{\mu_t\theta \cup [\vec{N}/\vec{z}], \zeta \cup [\vec{C}/\vec{z}], \Delta_m}$.

We obtain the conclusion of the lemma by taking an empty set BV.

Properties 1 to 4 hold by the definition of a (G_p, G)-tuple and Lemma 3.3.3. Property 5 is now proved by induction on the derivation of $(t,\ G_t; BV_t \vdash T_t,\ \mu_t) \in CCl(G_p, G \vdash f(\vec{p}, \vec{u}))$.

Initial set By the hypothesis we have

$$\Delta \vdash p_j\theta \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta,\Delta} \quad \text{for all } j = 1, \dots r, \ \text{ and}$$
$$\Delta \vdash u_j\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta,\Delta} \quad \text{for all } j = 1, \dots n$$

Introduction of variables

$$\frac{G_p, G_m, BV \vdash_{CC} y \ : \ A, \qquad y \in dom(BV)}{(y,\ G_m; BV \vdash A,\ \mu)}$$

We have to show that $\Delta_m \vdash y(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash A]\!]_{\mu\theta \cup [\vec{N}/\vec{z}], \zeta \cup [\vec{C}/\vec{z}], \Delta_m}$. Since $y \in dom(BV)$, $y = z_i$ and $y(\theta \cup [\vec{N}/\vec{z}]) = N_i$. We complete the proof using the assumption for N_i.

Abstraction

$$\frac{\begin{array}{cc}(t,\ G_m; BV, y : A \vdash T,\ \mu) & (A\mu,\ G_m; BV \vdash p_2,\ \mu) \\ G_p, G_m, BV, y : A \vdash_{CC} T \ : \ p_1 & G_p, G_m, BV \vdash_{CC} A \ : \ p_2\end{array}}{(\lambda y : A\mu.t,\ G_m; BV \vdash (y : A).T,\ \mu)}$$

To show $\Delta_m \vdash (\lambda y : A\mu.t)(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash (y : A).T]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$ it suffices to check whether $\Delta_{m+1} \vdash @((\lambda y : A\mu.t)(\theta \cup [\vec{N}/\vec{z}]), M) \in [\![G_p, G_m, BV, y : A \vdash T]\!]_{\mu\theta\cup[\vec{N}/\vec{z}]\cup[M/y],\zeta\cup[\vec{C}/\vec{z}]\cup[C'/y],\Delta_{m+1}}$ for any $\Delta_{m+1} \supseteq \Delta_m$, any $\Delta_{m+1} \vdash M \in [\![G_p, G_m, BV \vdash A]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$ and any $C' \in \mathcal{C}_{\Delta_{m+1},M}$.
By induction hypothesis we know that $\Delta_{m+1} \vdash t(\theta \cup [\vec{N}/\vec{z}] \cup [M/y]) \in [\![G_p, G_m, BV, y : A \vdash T]\!]_{\mu\theta\cup[\vec{N}/\vec{z}]\cup[M/y],\zeta\cup[\vec{C}/\vec{z}]\cup[C'/y],\Delta_{m+1}}$. But $@((\lambda y : A\mu.t)(\theta \cup [\vec{N}/\vec{z}]), M) \rightarrow_\beta t(\theta \cup [\vec{N}/\vec{z}] \cup [M/y])$, which gives our claim by Lemma 5.1.6 (together with the fact that $A\mu \in \mathcal{SN}$ that follows from induction hypothesis applied to $(A,\ G_m; BV \vdash p_2,\ \mu)$).

Application to a variable

$$\frac{(t,\ G_m; BV \vdash (x : A).B,\ \mu) \qquad (y,\ G_m; BV \vdash A,\ \mu)}{(@(t, y),\ G_m; BV \vdash B[y/x],\ \mu)}$$

By induction hypothesis we know that

$$\Delta_m \vdash t(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash (x : A).B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$
$$\Delta_m \vdash y(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash A]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

and $y = z_i \in dom(BV)$. By applying the definition of interpretation of a product type we get

$$\Delta_m \vdash @(t(\theta \cup [\vec{N}/\vec{z}]), y(\theta \cup [\vec{N}/\vec{z}])) \\ \in [\![G_p, G_m, BV, x : A \vdash B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}]\cup[N_i/x],\zeta\cup[\vec{C}/\vec{z}]\cup[C_i/x],\Delta_m}$$

Since $y : A$ and $x : A$ are two variables that are equal after applying substitution on terms and candidates (N_i, C_i are the values), we can substitute y for x and conclude that $\Delta_m \vdash @(t, y)(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash B[y/x]]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$.

Application to a parameter

$$\frac{(t,\ G_m; BV \vdash (x : P).B,\ \mu)}{(@(t, p),\ G_m; BV \vdash B[p/x],\ \mu)}$$

where $p : P \in G_p$. The proof is very similar to the proof of the previous case and relies on the fact that $p : P$ (like $y : A$ in the previous case) is already in the environment G_p, G_m, BV.

Application to a small argument

$$\frac{(t_1,\ G_m; BV \vdash (x : A).B,\ \mu) \qquad (t_2,\ G_m; BV \vdash A,\ \mu)}{(@(t_1, t_2),\ G_m, x : A; BV \vdash B,\ \mu \cup [t_2/x])}$$

if $x \in \mathcal{V}ar^\star$ and t_2 and A are clean. By induction hypothesis we know that

$$\Delta_m \vdash t_1(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash (x : A).B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$
$$\Delta_m \vdash t_2(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash A]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

By applying the definition of interpretation of a product type we get

$$\Delta_m \vdash @(t_1(\theta \cup [\vec{N}/\vec{z}]), t_2(\theta \cup [\vec{N}/\vec{z}])) \\ \in [\![G_p, G_m, BV, x : A \vdash B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}]\cup[t_2(\theta\cup[\vec{N}/\vec{z}])/x],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

Since A is clean, $G_p, G_m, BV, x : A = G_p, G_m, x : A, BV$. Moreover $\mu\theta \cup [\vec{N}/\vec{z}] \cup [t_2(\theta \cup [\vec{N}/\vec{z}])/x] = (\mu \cup [t_2/x])\theta \cup [\vec{N}/\vec{z}]$. Then

$$[\![G_p, G_m, BV, x : A \vdash B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}]\cup[t_2(\theta\cup[\vec{N}/\vec{z}])/x],\zeta\cup[\vec{C}/\vec{z}],\Delta_m} = \\ [\![G_p, G_m, x : A, BV \vdash B]\!]_{(\mu\cup[t_2/x])\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

and the proof is complete.

Application to a nondependent argument

$$\frac{(t_1,\ G_m; BV \vdash (x:A).B,\ \mu) \qquad (t_2,\ G_m; BV \vdash A,\ \mu)}{(@(t_1, t_2),\ G_m; BV \vdash B,\ \mu)}$$

if $x \notin \mathcal{FV}(B)$. By induction hypothesis we know that

$$\Delta_m \vdash t_1(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash (x:A).B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$
$$\Delta_m \vdash t_2(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash A]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

By applying the definition of interpretation of a product type we get

$$\Delta_m \vdash @(t_1(\theta \cup [\vec{N}/\vec{z}]), t_2(\theta \cup [\vec{N}/\vec{z}])) \\ \in [\![G_p, G_m, BV, x : A \vdash B]\!]_{\mu\theta\cup[\vec{N}/\vec{z}]\cup[t_2(\theta\cup[\vec{N}/\vec{z}])/x],\zeta\cup[\vec{C}/\vec{z}]\cup[C'/x],\Delta_m}$$

If $x \notin \mathcal{FV}(B)$ we can eliminate it from the environment $G_p, G_m, BV, x : A$ and the substitutions $\mu\theta \cup [\vec{N}/\vec{z}] \cup [t_2(\theta \cup [\vec{N}/\vec{z}])/x]$ and $\zeta \cup [\vec{C}/\vec{z}] \cup [C'/x]$, without changing the interpretation (by Lemma 5.3.1), and get the desired conclusion.

Precedence

$$\frac{\forall i\ \ (t_i,\ G_m, \overrightarrow{y : d[\vec{p}'/\vec{q}]}; BV \vdash d_i[\vec{p}'/\vec{q}],\ \mu \cup [\vec{t}/\vec{y}])}{(g(\vec{p}', \vec{t}),\ G_m, \overrightarrow{y : d[\vec{p}'/\vec{q}]}; BV \vdash e[\vec{p}'/\vec{q}],\ \mu \cup [\vec{t}/\vec{y}])}$$

if $f >_{\mathcal{F}} g$, $g : (q_1 : Q_1) \dots (q_{r'} : Q_{r'}).(y_1 : d_1) \dots (y_m : d_m).e$, $\vec{p}' \subseteq dom(G_p)$ and for every i either t_i is clean or $y_i \notin \mathcal{FV}(d_1, \dots d_m, e)$.
By induction hypothesis we know that for every i, $\Delta_m \vdash t_i(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, \overrightarrow{y : d[\vec{p}'/\vec{q}]}, BV \vdash d_i[\vec{p}'/\vec{q}]]\!]_{(\mu\cup[\vec{t}/\vec{y}])\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$ and since G_m, BV, μ, $[\vec{N}/\vec{z}]$ and $[\vec{C}/\vec{z}]$ are not important in the interpretation of d_i, we have

$$\Delta_m \vdash t_i\theta \in [\![G_p, \overrightarrow{y : d[\vec{p}'/\vec{q}]} \vdash d_i[\vec{p}'/\vec{q}]]\!]_{[\vec{t}/\vec{y}]\theta,\zeta,\Delta_m}$$

By the assumptions of the lemma we know that for all i:

$$\Delta_m \vdash p_i\theta \in [\![G_p \vdash P_i]\!]_{\theta,\zeta,\Delta_m}$$

Since $\vec{p}' \subseteq \vec{p}$, there is a function on indices η such that $p'_i = p_{\eta_i}$ and $\overrightarrow{p : P} \vdash$

$Q_i[\vec{p'}/\vec{q}] \overset{*}{\leftrightarrow} P_{\eta_i}$. From this

$$[\![G_p \vdash P_{\eta_i}]\!]_{\theta,\zeta,\Delta_m} = [\![\overrightarrow{p : P} \vdash Q_i[\vec{p'}/\vec{q}]]\!]_{\theta,\zeta,\Delta_m} = [\![\overrightarrow{q : Q} \vdash Q_i]\!]_{\theta',\zeta',\Delta_m}$$

$$[\![G_p, \overrightarrow{y : d[\vec{p'}/\vec{q}]} \vdash d_i[\vec{p'}/\vec{q}]]\!]_{[\vec{t}/\vec{y}]\theta,\zeta,\Delta_m} = [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash d_i]\!]_{[\vec{t}/\vec{y}]\theta',\zeta',\Delta_m}$$

where $q_j\theta' = p'_j\theta$, $q_j\zeta' = p'_j\zeta$.

Now, it is sufficient to use the assumption about terms that are smaller than $f(\vec{p}\theta, \vec{u}\theta)$ in $(>_{\mathcal{F}}, (\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})_{stat})$ to get,

$$\Delta_m \vdash g(\vec{p'}, \vec{t})\theta \in [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash e]\!]_{[\vec{t}/\vec{y}]\theta',\zeta',\Delta_m} = [\![G_p, \overrightarrow{y : d[\vec{p'}/\vec{q}]} \vdash e[\vec{p'}/\vec{q}]]\!]_{[\vec{t}/\vec{y}]\theta,\zeta,\Delta_m}$$

which is equivalent to the desired conclusion:

$$\Delta_m \vdash g(\vec{p'}, \vec{t})(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, \overrightarrow{y : d[\vec{p'}/\vec{q}]}, BV \vdash e[\vec{p'}/\vec{q}]]\!]_{(\mu\cup[\vec{t}/\vec{y}])\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

Recursive call

$$\frac{\forall i\ \ (t_i,\ G_m, \overrightarrow{x : b}; \emptyset \vdash b_i,\ \mu \cup [\vec{t}/\vec{x}])}{(f(\vec{p}, \vec{t}),\ G_m, \overrightarrow{x : b}; \emptyset \vdash c,\ \mu \cup [\vec{t}/\vec{x}])}$$

if $G_p, G \vdash (\vec{p}, \vec{u}) \succ^{\mathcal{CS}}_{stat} (\vec{p}, \vec{t})$, where *stat* is a status of $f : \overrightarrow{(p : P)}\overrightarrow{(x : b)}.c$ and $\succ^{\mathcal{CS}}_{stat}$ is a *stat* extension of the constructor extension of $\succ$.

This case can be handled in much the same way as the previous one, the only difference being in the decreasing argument. To use the hypothesis of our lemma we need to know that $\Delta \vdash \vec{p}\theta, \vec{u}\theta(\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})_{stat}\vec{p}\theta, \vec{t}\theta$. Fortunately it can be concluded from $G_p, G \vdash (\vec{p}, \vec{u}) \succ^{\mathcal{CS}}_{stat} (\vec{p}, \vec{t})$ and the fact that $(\succ^{\mathcal{CS}}\ \theta) \subseteq (\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})$ (see Lemma 5.4.5).

Constructor–decomposition

$$\frac{(c(\vec{a'}, \vec{N'}),\ G_m; BV \vdash s(\vec{a})\vec{u},\ \mu)}{(a'_i,\ G_m; BV \vdash Q_i[\vec{a}/\vec{q}],\ \mu)}$$

$$\frac{(c(\vec{a'}, \vec{N'}),\ G_m; BV \vdash s(\vec{a})\vec{u},\ \mu)}{(N'_i,\ G_m, \overrightarrow{z : d[\vec{a}/\vec{q}]}; BV \vdash d_i[\vec{a}/\vec{q}],\ \mu \cup [\vec{N'}/\vec{z}])}$$

if $c : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'}).(z_1 : d_1) \ldots (z_k : d_k).s(\vec{q})w_1 \ldots w_n$ is a constructor of $s : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'}).(x_1 : b_1) \ldots (x_n : b_n).\star$, if $\vec{a}$ are clean and if for every i either N'_i is clean or $z_i \notin \mathcal{FV}(d_{i+1}, \ldots d_k)$.

In this case, the desired conclusions are simple consequences of Constructor Lemma 5.4.3 applied to

$$\Delta_m \vdash c(\vec{a'}, \vec{N'})(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash s(\vec{a})\vec{u}]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

which is true by induction hypothesis.

Constructor

$$\frac{\begin{array}{cc} \forall i\ (a'_i,\ G_m; BV \vdash Q_i[\vec{a}/\vec{q}],\ \mu) & \forall i\ (N'_i,\ G_m, \overrightarrow{z : d[\vec{a}/\vec{q}]}; BV \vdash d_i[\vec{a}/\vec{q}],\ \mu \cup [\vec{N'}/\vec{z}]) \\ G_p, G_m, BV \vdash_{CC} s(\vec{a})\vec{u}\ :\ \star & G_p, G, BV\mu \vdash_{CC} c(\vec{a'}, \vec{N'})\ :\ (s(\vec{a})\vec{u})\mu \end{array}}{(c(\vec{a'}, \vec{N'}),\ G_m; BV \vdash s(\vec{a})\vec{u},\ \mu)}$$

if $c : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(z_1 : d_1)\dots(z_k : d_k).s(\vec{q})w_1\dots w_n$ is a constructor of $s : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(x_1 : b_1)\dots(x_n : b_n).\star$ and if $\mathcal{FV}(\vec{a},\vec{u}) \cap \mathcal{V}ar^{\Box} \subseteq G_p \cup BV$.
The desired conclusion is a simple consequence of Inductive Lemma 5.2.13 applied to $G_p, G, BV\mu \vdash_{CC} c(\vec{a'}, \vec{N'}) : (s(\vec{a})\vec{u})\mu$ and

$$\Delta_m \vdash a'_i(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

$$\Delta_m \vdash N'_i(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, \overrightarrow{z : d[\vec{a}/\vec{q}]}, BV \vdash d_i[\vec{a}/\vec{q}]]\!]_{(\mu\cup[\vec{N'}/\vec{z}])\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

that are true for all i by induction hypothesis.

Type constructor

$$\frac{\forall i\,(a_i,\ G_m; BV \vdash A_i,\ \mu) \quad \forall i\,(u_i,\ G_m; BV \vdash U_i,\ \mu) \quad G_p, G, BV\mu \vdash_{CC} s(\vec{a})\vec{u} : \star}{(s(\vec{a})\vec{u},\ G_m; BV \vdash \star,\ \mu)}$$

if $s : (q_1 : Q_1)\dots(q_{r'} : Q_{r'}).(x_1 : b_1)\dots(x_n : b_n).\star \in \mathcal{TC}$.
We must show that $\Delta_m \vdash s(\vec{a})\vec{u}(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash \star]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$.
Since interpretation of $\star$ is just $\mathcal{SN}_{\Delta,\star}$ it suffices that $\vec{a}(\theta \cup [\vec{N}/\vec{z}])$ and $\vec{u}(\theta \cup [\vec{N}/\vec{z}])$ are strongly normalizing. By the induction hypothesis we have

$$\Delta_m \vdash a_i(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash A_i]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

$$\Delta_m \vdash u_i(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, BV \vdash U_i]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

for some $\vec{A}$, $\vec{U}$. We complete the proof using the fact that every interpretation is a candidate, which is a subset of strongly normalizing terms.

Product type

$$\frac{\begin{matrix}(A,\ G_m; BV \vdash p_1,\ \mu) & (B,\ G_m; BV, x : C \vdash p_2,\ \mu)\\ G_p, G, BV\mu, x : A \vdash_{CC} B : p_2 & G_p, G, BV\mu \vdash_{CC} A \overset{*}{\leftrightarrow} C\mu\end{matrix}}{((x : A).B,\ G_m; BV \vdash p_2,\ \mu)}$$

if $p_1, p_2 \in \{\star, \Box\}$.
The proof is very similar to the proof of the previous case.

Reduction

$$\frac{(t,\ G_m; BV \vdash T,\ \mu)}{(t',\ G_m; BV \vdash T,\ \mu)} \quad \text{if } t \to_\beta t'.$$

The proof is immediate as the property of being an element of an interpretation is closed under reduction.

Weak1

$$\frac{(t,\ G_m; BV \vdash T,\ \mu) \qquad G_p, G_m \vdash_{CC} A : \star/\Box \qquad G_p, G \vdash_{CC} M : A\mu}{(t,\ G_m, x : A; BV \vdash T,\ \mu \cup [M/x])}$$

if $x \notin dom(G_m, BV)$. Note that neither BV nor T depend on x. Consequently,

$$[\![G_p, G_m, x : A, BV \vdash T]\!]_{(\mu\cup[M/x])\theta\cup[\vec{N}/\vec{z}],\zeta\cup[C'/x]\cup[\vec{C}/\vec{z}],\Delta_m} = [\![G_p, G_m, BV \vdash T]\!]_{\mu\theta\cup[\vec{N}/\vec{z}],\zeta\cup[\vec{C}/\vec{z}],\Delta_m}$$

(by Lemma 5.3.1) and

$$\Delta_m \vdash t(\theta \cup [\vec{N}/\vec{z}]) \in [\![G_p, G_m, x : A, BV \vdash T]\!]_{(\mu \cup [M/x])\theta \cup [\vec{N}/\vec{z}], \zeta \cup [C'/x] \cup [\vec{C}/\vec{z}], \Delta_m}.$$

Weak2

$$\frac{(t,\ G_m; BV \vdash T,\ \mu) \qquad G_p, G_m, BV \vdash_{CC} A\ :\ \star/\Box}{(t,\ G_m; BV, x : A \vdash T,\ \mu)}$$

if $x \notin dom(G_m, BV)$ and $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box} \subseteq G_p, BV$. The reasoning used in the previous case applies. $\Box$

The following easy lemma will be often used in the proof of Fun Lemma.

Lemma 5.4.8 (*Recursive Comparison in HORPO*)
Suppose that G_p, G_l, G are environments, μ_l is a well-typed substitution and l, T_l are terms satisfying:

1. $\mathcal{FV}(T_l) \cap \mathcal{V}ar^{\Box} \subseteq G_p$,
2. $\mu_l\ :\ (G_p, G_l) \to (G_p, G)$ such that $\mu_l|_{G_p} = id_{G_p}$,
3. $G_p, G_l \vdash_{CC} T_l\ :\ \star/\Box$,
4. $G_p, G \vdash_{CC} l\ :\ T_l\mu_l$ and

For every environment Δ, well-typed substitution $\theta\ :\ G_p, G \to \Delta$ and candidate assignment ζ compatible with θ, if $\Delta \vdash l\theta \in [\![G_p, G_l \vdash T_l]\!]_{\mu_l\theta,\zeta,\Delta}$ and $G_p, G \vdash l \succeq^{\mathcal{CS}} t$ then there exist G_t, T_t and μ_t such that:

1. $\mathcal{FV}(T_t) \cap \mathcal{V}ar^{\Box} \subseteq G_p$,
2. $\mu_t\ :\ (G_p, G_t) \to (G_p, G)$ such that $\mu_t|_{G_p} = id_{G_p}$,
3. $G_p, G_t \vdash_{CC} T_t\ :\ \star/\Box$,
4. $G_p, G \vdash_{CC} t\ :\ T_t\mu_t$ and
5. $\Delta \vdash t\theta \in [\![G_p, G_t \vdash T_t]\!]_{\mu_t\theta,\zeta,\Delta}$.

Proof
By the definition of $\succeq^{\mathcal{CS}}$, $G_p, G \vdash l \succeq^{\mathcal{CS}} t$ means that there is a term s such that $l \trianglerighteq^{\mathcal{CS}} s$ and $G_p, G \vdash s \succ t\ :\ A$ or $s = t$. By Constructor Subterm Lemma 5.4.4, for this s there are G_s, T_s and μ_s satisfying properties 1 to 5 above (with G_l, T_l and μ_l replaced by G_s, T_s and μ_s). Now, if $G_p, G \vdash s \succ t\ :\ A$ then by the condition (S1) from the definition of candidates $\Delta \vdash t\theta \in [\![G_p, G_s \vdash T_s]\!]_{\mu_s\theta,\zeta,\Delta}$. Moreover, $G_p, G \vdash_{CC} t\ :\ T_s\mu_s$ because HORPO compares only terms of equal types. Other properties follow immediately. $\Box$

Proof of Fun Lemma 5.4.6
Let us recall the statement of the lemma:

Let f be a function symbol of type $(p_1 : P_1) \ldots (p_r : P_r).(x_1 : b_1) \ldots (x_n : b_n).c$ such that $\Delta \vdash f(\vec{\alpha}, \vec{\sigma})\ :\ c[\vec{\alpha}/\vec{p}, \vec{\sigma}/\vec{x}]$ holds. For every candidate assignment compatible with $[\vec{\alpha}/\vec{p}]$, ζ, if

- $\Delta \vdash \alpha_j \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{[\vec{\alpha}/\vec{p}],\zeta,\Delta}$, for all $j = 1, \ldots r$,
- $\Delta \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}],\zeta,\Delta}$, for all $j = 1, \ldots n$,

then $\Delta \vdash f(\vec{\alpha}, \vec{\sigma}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}],\zeta,\Delta}$.

The proof is done by nested induction. The first one is on $(f, (\vec{\alpha}, \vec{\sigma}))$ ordered by $(>_{\mathcal{F}}, (\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})_{stat})_{lex}$, where *stat* is either *mul* or *lex* depending on the status of f. Given $\Delta \vdash f(\vec{\alpha}, \vec{\sigma})$ we will check whether all its reducts, i.e. all terms t such that $\Delta \vdash f(\vec{\alpha}, \vec{\sigma}) \rightarrow t$, belong to $[\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}], \zeta, \Delta}$. The second induction is on the size of the reduct. Let q be the rewriting position and $\Gamma \vdash l \succ r \ : \ T$ the HORPO judgment used to rewrite $f(\vec{\alpha}, \vec{\sigma})$ to t.

According to the position q and the HORPO case used to prove $\Gamma \vdash l \succ r \ : \ T$, we distinguish the following possibilities:

1. $q \neq \Lambda$. Let us denote $f(\vec{\alpha}, \vec{\sigma})$ by $f(\vec{d})$. In this case $\Delta \vdash f(\vec{d}) \rightarrow f(\vec{t})$ and there is some i such that $\Delta \vdash d_i \rightarrow t_i$ and $t_j = d_j$ for $j \neq i$. Let us denote by D_i the type (P_i or b_{i-r}) of d_i. By condition (S1) from the definition of candidates we have $\Delta \vdash t_i \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash D_i]\!]_{[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}], \zeta, \Delta}$. Let us denote by γ_i the substitution $[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}]$ with t_i assigned to the ith variable. Since $[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}]$ and γ_i are convertible, Lemma 5.2.14 shows that $\Delta \vdash t_i \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash D_i]\!]_{\gamma_i, \zeta, \Delta}$. By the first induction hypothesis $\Delta \vdash f(\vec{t}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\gamma_i, \zeta, \Delta}$ and we conclude by Lemma 5.2.14 that $\Delta \vdash f(\vec{t}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}], \zeta, \Delta}$.
2. $q = \Lambda$. If $\Gamma \vdash l \succ r \ : \ T$ is the judgment used for rewriting, then

$$\begin{aligned} &\Gamma = (p_1 : P_1) \ldots (p_r : P_r), G \\ &l = f(\vec{l}) = f(\vec{p}, \vec{u}) \\ &l\theta = f(\vec{p}\theta, \vec{u}\theta) = f(\vec{\alpha}, \vec{\sigma}), \qquad t = r\theta \end{aligned}$$

   where θ is a well-typed substitution from $(\overrightarrow{p : P}, G)$ to Δ. Let us denote by μ_1 the substitution $[\vec{u}/\vec{x}]$. Note that $[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}] = \mu_1\theta$. Let us also denote by G_p the environment $\overrightarrow{p : P}$. We have to check all HORPO cases which could have been used for proving $\Gamma \vdash l \succ r \ : \ T$:

   (a) Case I.1: Then either $t \in CCl_f(\vec{p}, \vec{u})$ or $\Gamma \vdash l_i \succeq^{\mathcal{CS}} r$ for some i. In any case there exist G_t, T_t and μ_t such that $G_t, G \vdash t \ : \ T_t\mu_t$ and $\Delta \vdash t\theta \in [\![G_p, G_t \vdash T_t]\!]_{\mu_t\theta, \zeta, \Delta}$ (by Lemma 5.4.7 if $t \in CCl_f(\vec{p}, \vec{u})$ and by Lemma 5.4.8 otherwise). Since $G_p, G \vdash T_t\mu_t \overset{*}{\leftrightarrow} c\mu_1$ and T_t and c are terms of the Calculus of Constructions, Small Substitution Lemma 5.4.1 shows that $[\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\mu_1\theta, \zeta, \Delta} = [\![G_p, G_t \vdash T_t]\!]_{\mu_t\theta, \zeta, \Delta}$. Hence $\Delta \vdash t \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\mu_1\theta, \zeta, \Delta}$.

   (b) Case I.2: Then $f >_{\mathcal{F}} g$, $g : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'}).(y_1 : d_1) \ldots (y_m : d_m).e \in \Sigma$, $\Gamma = (p_1 : P_1) \ldots (p_r : P_r), G$ and

$$\begin{aligned} &l = f(\vec{l}) = f(\vec{p}, \vec{u}), && r = g(\vec{p'}, \vec{w}) \\ &l\theta = f(\vec{p}\theta, \vec{u}\theta) = f(\vec{\alpha}, \vec{\sigma}), && t = r\theta = g(\vec{p'}\theta, \vec{w}\theta) \end{aligned}$$

   We first show that every argument of g (i.e. $p'_i\theta$, $w_i\theta$) belongs to the interpretation of its type. By the definition of HORPO we know that $\vec{p'} \subseteq \vec{p}$. This inclusion can be viewed as function on indices η such that $p'_i = p_{\eta_i}$ and $(p_1 : P_1) \ldots (p_r : P_r) \vdash Q_i[\vec{p'}/\vec{q}] \overset{*}{\leftrightarrow} P_{\eta_i}$. After applying substitution θ to $\vec{p}$ and $\vec{p'}$, we conclude that $\Delta \vdash p'_i\theta$ belongs to $[\![\overrightarrow{p : P} \vdash P_{\eta_i}]\!]_{[\vec{\alpha}/\vec{p}], \zeta, \Delta} = [\![\overrightarrow{p : P} \vdash Q_i[\vec{p'}/\vec{q}]]\!]_{[\vec{\alpha}/\vec{p}], \zeta, \Delta} = [\![\overrightarrow{q : Q} \vdash Q_i]\!]_{[\vec{\alpha'}/\vec{q}], \zeta', \Delta}$, where $\alpha'_i = \alpha_{\eta_i}$ and $q_i\zeta' = p_{\eta_i}\zeta$.

We are done with $\vec{p}'\theta$. For $\vec{w}\theta$, there are three cases to consider, depending how the original rule was accepted. For every w_i we have either $\Gamma \vdash f(\vec{p},\vec{u}) \succ w_i : T$, or $\Gamma \vdash u_j \succeq^{\mathcal{CS}} w_i : b_j[\vec{u}/\vec{x}]$ or $w_i \in CCl_f(\vec{p},\vec{u})$.
If $\Gamma \vdash f(\vec{p},\vec{u}) \succ w_i : T$ then $\Delta \vdash w_i \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\mu_1\theta,\zeta,\Delta}$ by the second induction hypothesis, as $|w_i\theta|$ is smaller than $|t|$. In this case let $G_w = \overrightarrow{x : b}$, $T_w = c$ and $\mu_w = \mu_1$.
Otherwise, either by Lemma 5.4.7 or by Lemma 5.4.8 we get $\Delta \vdash w_i\theta \in [\![G_p, G_w \vdash T_w]\!]_{\mu_w\theta,\zeta,\Delta}$ for some G_w, T_w, μ_w such that $\mathcal{FV}(T_w) \cap \mathcal{V}ar^{\square} \subseteq G_p$, $\mu_w : (G_p, G_w) \rightarrow (G_p, G)$, $\mu_w|_{G_p} = id_{G_p}$ and $G_p, G \vdash w_i : T_w\mu_w$.
Since $G_p, G \vdash T_w\mu_w \overset{*}{\leftrightarrow} d_i[\vec{p}'/\vec{q}][\vec{w}/\vec{y}]$ and T_w and $d_i[\vec{p}'/\vec{q}]$ are terms of the Calculus of Constructions, Small Substitution Lemma shows that $[\![G_p, G_w \vdash T_w]\!]_{\mu_w\theta,\zeta,\Delta} = [\![\overrightarrow{p : P}, \overrightarrow{y : d[\vec{p}'/\vec{q}]} \vdash d_i[\vec{p}'/\vec{q}]]\!]_{[\vec{w}/\vec{y}]\theta,\zeta,\Delta}$. Moreover, by simple renaming $[\![\overrightarrow{p : P}, \overrightarrow{y : d[\vec{p}'/\vec{q}]} \vdash d_i[\vec{p}'/\vec{q}]]\!]_{[\vec{w}/\vec{y}]\theta,\zeta,\Delta} = [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash d_i]\!]_{[\vec{w}/\vec{y}]\theta',\zeta',\Delta}$, where $q_i\theta' = p_{\eta_i}\theta$ (ζ' has been defined few lines above).
Now, we can apply the first induction hypothesis ($f >_{\mathcal{F}} g$) and get $\Delta \vdash g(\vec{p}'\theta, \vec{w}\theta) \in [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash e]\!]_{[\vec{w}/\vec{y}]\theta',\zeta',\Delta} = [\![\overrightarrow{p : P}, \overrightarrow{y : [\vec{p}'/\vec{q}]} \vdash e[\vec{p}'/\vec{q}]]\!]_{[\vec{w}/\vec{y}]\theta,\zeta,\Delta}$.
We complete the proof of this case by noting, that $[\![\overrightarrow{p : P}, \overrightarrow{y : [\vec{p}'/\vec{q}]} \vdash e[\vec{p}'/\vec{q}]]\!]_{[\vec{w}/\vec{y}]\theta,\zeta,\Delta} = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\mu_1\theta,\zeta,\Delta}$ (again by Small Substitution Lemma 5.4.1 using the fact that $e[\vec{p}'/\vec{q}]$ and c are terms of the Calculus of Constructions).

(c) Case I.3: Then $r = f(\vec{p},\vec{w})$, $f \in Mul$ and $t = r\theta$. The proof for the previous case still works here. The only change is in the induction argument, which now relies on $(\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})_{mul}$.

(d) Case I.4: Then $r = f(\vec{p},\vec{w})$, $f \in Lex$ and $t = r\theta$. As above, the proof for case 2(b) can be repeated here. This time the induction argument uses $(\rightarrow^{\Delta} \cup \rhd^{\mathcal{CS}})_{lex}$.

(e) Case I.5: Then $r = c'(\vec{v},\vec{r})$, where c' is a constructor of an inductive type s, $t = r\theta$, and for every $d \in (\vec{v},\vec{r})$:

$$\Gamma \vdash l \succ d : T \quad \text{or} \quad \Gamma \vdash l_j \succeq^{\mathcal{CS}} d \text{ for some } l_j \quad \text{or} \quad d \in CCl_f(\vec{l})$$

Since $G_p, G \vdash c'(\vec{v},\vec{r}) : c\mu_1$ and μ_1 is small ($\mathcal{FV}(c) \cap \mathcal{V}ar^{\square} \subseteq G_p$ and $\mu_1|_{G_p} = id_{G_p}$), $c = s(\vec{a})\vec{u}$ for some $\vec{a}$, $\vec{u}$. By Inductive Lemma 5.2.13, to prove that $\Delta \vdash c'(\vec{v},\vec{r})\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash s(\vec{a})\vec{u}]\!]_{\mu_1\theta,\zeta,\Delta}$ it is sufficient to show

$$\Delta \vdash v_i\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\mu_1\theta,\zeta,\Delta}$$
$$\Delta \vdash r_i\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b}, \overrightarrow{z : d[\vec{a}/\vec{q}]} \vdash d_i[\vec{a}/\vec{q}]]\!]_{(\mu_1 \cup [\vec{r}/\vec{z}])\theta,\zeta,\Delta}$$

provided that $c' : (q_1 : Q_1)\ldots(q_{r'} : Q_{r'}).(z_1 : d_1)\ldots(z_k : d_k).s(\vec{q})w_1\ldots w_n$ and $s : (q_1 : Q_1)\ldots(q_{r'} : Q_{r'}).(x_1 : b_1)\ldots(x_n : b_n).\star$.
By the second induction hypothesis and Lemmas 5.4.7 and 5.4.8 we get that every $\Delta \vdash d\theta$, where $d \in \{\vec{v},\vec{r}\}$, belongs to some $[\![G_p, G_d \vdash T_d]\!]_{\mu_d\theta,\zeta,\Delta}$ where $\mathcal{FV}(T_d) \cap \mathcal{V}ar^{\square} \subseteq G_p$, $\mu_d : (G_p, G_d) \rightarrow (G_p, G)$, $\mu_d|_{G_p} = id_{G_p}$ and $G_p, G \vdash d : T_d\mu_d$.

Since $G_p, G \vdash Q_i[\vec{a}/\vec{p}]\mu_1 \overset{*}{\leftrightarrow} T_d\mu_d$ for $d = v_i$, $G_p, G \vdash d_i[\vec{a}/\vec{p}](\mu_1 \cup [\vec{r}/\vec{z}]) \overset{*}{\leftrightarrow} T_d\mu_d$ for $d = r_i$ and all $Q_i[\vec{a}/\vec{p}]$, $d_i[\vec{a}/\vec{p}]$, T_d are terms of the Calculus of Constructions, we get the desired assertions by the Small Substitution Lemma.

(f) Case I.6: Then

$$\begin{array}{ll} r = \hat{r}_1\,\hat{r}_2, & @(r_0, r_1, \ldots r_m) \text{ is a left flattening of } r \\ t = \hat{r}_1\theta\,\hat{r}_2\theta = \hat{t}_1\,\hat{t}_2, & @(t_0, t_1, \ldots t_m) = @(r_0\theta, r_1\theta, \ldots r_m\theta) \end{array}$$

By the definition of HORPO, we know that for every r_i, $i \geqslant 1$, either $\Gamma \vdash f(\vec{p}, \vec{u}) \succ r_i \ : \ T$, or $\Gamma \vdash u_j \succeq^{\mathcal{CS}} r_i \ : \ b_j[\vec{u}/\vec{x}]$ or $r_i \in CCl_f(\vec{p}, \vec{u})$. In all cases, $\Delta \vdash r_i\theta \in [\![G_p, G_{r_i} \vdash T_{r_i}]\!]_{\theta(\mu_{r_i}),\zeta,\Delta}$ for some G_{r_i}, T_{r_i}, μ_{r_i} such that $\mathcal{FV}(T_{r_i}) \cap \mathcal{V}ar^{\Box} \subseteq G_p$ $\mu_{r_i} \ : \ (G_p, G_{r_i}) \to (G_p, G)$, $\mu_{r_i}\,|_{G_p} = id_{G_p}$ and $G_p, G \vdash r_i \ : \ T_{r_i}\mu_{r_i}$ (the reasoning is exactly the same as in case (b)).

Let us now examine r_0. By the definition of HORPO either $\Gamma \vdash f(\vec{p}, \vec{u}) \succ r_0 \ : \ T$ or $p_j = r_0$ or $\Gamma \vdash u_j \succeq^{\mathcal{CS}} r_0$. In any case, this implies, that $\Delta \vdash r_0\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash T_{r_0}]\!]_{\mu_1\theta,\zeta,\Delta}$ (T_{r_0} equal to some P, b or c).

It's clear that the type of r_0 is functional, and in consequence, that T_{r_0} is convertible with a product $(y_1 : A_1)\ldots(y_m : A_m).B$. Please recall that $\vec{y}$ are subject to several restrictions as $@(r_0, r_1, \ldots r_m)$ satisfy the application condition 3.2.1. These restrictions ensure that for every $z \in (\mathcal{FV}(\vec{A}, B) \cap \mathcal{V}ar^{\Box})$, we have either $z \in \vec{p}$ or $z = y_i$ such that $r_i \in \vec{p}$.

Let us denote by μ_2 the substitution $[\vec{u}/\vec{x}] \cup [\vec{r}/\vec{y}]$ and by ζ_2 the candidate assignment defined by $x\zeta_2 = x\zeta$ for $x \in G_p$ and $x\zeta_2 = p\zeta$ for $x = y_i$ such that $r_i = p$. Note, that $\mu_2\theta = [\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}] \cup [\vec{t}/\vec{y}]$.

By the strong version of Small Substitution Lemma 5.4.2, the interpretation

$$[\![G_p, G_{r_i} \vdash T_{r_i}]\!]_{\mu_{r_i}\theta,\zeta,\Delta}$$

is equal to

$$[\![\overrightarrow{p : P}, \overrightarrow{x : b}, \overrightarrow{y : A} \vdash A_i]\!]_{\mu_2\theta,\zeta_2,\Delta}$$

as A_i, T_r are terms of the Calculus of Constructions, $\overrightarrow{p : P}, G \vdash T_{r_i}\mu_{r_i} \overset{*}{\leftrightarrow} A_i\mu_2$ and $z\mu_2 \in \vec{p}$ for every $z \in \mathcal{FV}(\vec{A}) \cap \mathcal{V}ar^{\Box}$.

By the definition of the interpretation of a product type we get

$$\Delta \vdash @(t_0, t_1, \ldots t_m) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b}, \overrightarrow{y : A} \vdash B]\!]_{\mu_2\theta,\zeta_2,\Delta}$$

and again by Lemma 5.4.2 we conclude that

$$\Delta \vdash @(t_0, t_1, \ldots t_m) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\mu_1\theta,\zeta,\Delta}$$

Since $@(t_0, t_1, \ldots t_m) = @(\hat{t}_1, \hat{t}_2)$, this completes the proof of this case and of this lemma. $\Box$

5.5 Main theorem

This subsection presents the main lemma of the paper stating that every term belongs to the interpretation of its type; as a corollary we get the theorem that every term is $\mathcal{SN}$.

Definition 5.5.1 (*Validity of substitutions*)
Given a well-typed substitution $\rho : \Gamma \to \Delta$ and a candidate assignment ξ compatible with ρ we say that ρ is *valid with respect to* ξ, if for every $x \in dom(\Gamma)$ we have $\Delta \vdash x\rho \in [\![\Gamma \vdash \Gamma(x)]\!]_{\rho,\xi,\Delta}$. We write $[\![\rho, \xi]\!] : \Gamma \to \Delta$ whenever $\rho : \Gamma \to \Delta$, ξ is compatible with ρ and ρ is valid with respect to ξ.

Lemma 5.5.2 (*Main Lemma*)
Let Γ be environment and M, A terms such that $\Gamma \vdash M : A$. Then for every environment Δ and $[\![\rho, \xi]\!] : \Gamma \to \Delta$ we have $\Delta \vdash M\rho \in [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$.

Proof
Induction on the derivation of $\Gamma \vdash M : A$.
(ax)

$$\frac{}{\vdash \star : \Box}$$

This case is obvious, because $\star\rho = \star$ and $[\![\Gamma \vdash \Box]\!]_{\rho,\xi,\Delta} = \mathcal{SN}_{\Box,\Delta}$.
(var)

$$\frac{\Gamma \vdash a : p}{\Gamma, x : a \vdash x : a}\ (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

We have $\Delta \vdash x\rho \in [\![\Gamma, x : a \vdash a]\!]_{\rho,\xi,\Delta}$ by the definition of $[\![\rho, \xi]\!] : (\Gamma, x : a) \to \Delta$.
(weak)

$$\frac{\Gamma \vdash a : b \qquad \Gamma \vdash c : p}{\Gamma, x : c \vdash a : b}\ (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

and we have to show that for $[\![\rho, \xi]\!] : (\Gamma, x : c) \to \Delta$, $\Delta \vdash a\rho \in [\![\Gamma, x : c \vdash b]\!]_{\rho,\xi,\Delta}$ holds.

By induction hypothesis, for every Δ' and every ρ', ξ' such that $[\![\rho', \xi']\!] : \Gamma \to \Delta'$ we have $\Delta' \vdash a\rho' \in [\![\Gamma \vdash b]\!]_{\rho',\xi',\Delta'}$. If we show that $[\![\rho, \xi]\!] : \Gamma \to \Delta$ then we get $\Delta \vdash a\rho \in [\![\Gamma \vdash b]\!]_{\rho,\xi,\Delta}$ and, by Lemma 5.3.1, $\Delta \vdash a\rho \in [\![\Gamma, x : c \vdash b]\!]_{\rho,\xi,\Delta}$.

Let us verify that $[\![\rho, \xi]\!] : \Gamma \to \Delta$. We have to prove that for every $y \in dom(\Gamma)$, $\Delta \vdash y\rho \in [\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi,\Delta}$. We know, from $[\![\rho, \xi]\!] : (\Gamma, x : c) \to \Delta$, that for every $y \in dom(\Gamma)$, $\Delta \vdash y\rho \in [\![\Gamma, x : c \vdash (\Gamma, x : c)(y)]\!]_{\rho,\xi,\Delta}$. As $\Gamma(y) = (\Gamma, x : c)(y)$ and $[\![\Gamma \vdash (\Gamma, x : c)(y)]\!]_{\rho,\xi,\Delta} = [\![\Gamma, x : c \vdash (\Gamma, x : c)(y)]\!]_{\rho,\xi,\Delta}$ by Lemma 5.3.1, the proof of this case is finished.
(const)

$$\frac{\forall i \quad \Gamma \vdash a_i : b_i[a_1/x_1, \ldots, a_{i-1}/x_{i-1}]}{\Gamma \vdash e(a_1, \ldots, a_n) : c[a_1/x_1, \ldots, a_n/x_n]}\ (e : (x_1 : b_1) \ldots (x_n : b_n).c \in \Sigma_n)$$

This case has three subcases, for $s \in \mathcal{TC}$, $c \in \mathcal{CS}$, $f \in \mathcal{F}$.

Inductive type Let us start by specializing the typing rule to an inductive type s of type $(p_1 : P_1) \ldots (p_r : P_r).e$, with $e = (x_1 : b_1) \ldots (x_n : b_n).\star$.

$$\frac{\forall i \quad \Gamma \vdash a_i : P_i[\vec{a}/\vec{p}]}{\Gamma \vdash s(a_1, \ldots a_r) : e[\vec{a}/\vec{p}]}$$

We have to prove that $\Delta \vdash s(\vec{a}\rho) \in [\![\Gamma \vdash e[\vec{a}/\vec{p}]]\!]_{\rho,\xi,\Delta}$, assuming that $\Delta \vdash a_i\rho \in [\![\Gamma \vdash P_i[\vec{a}/\vec{p}]]\!]_{\rho,\xi,\Delta}$ holds for all a_i.

We will show a more general property namely, that for every $(\Delta^1 \vdash d_1, C_1), \dots (\Delta^n \vdash d_n, C_n)$, if $\Delta = \Delta^0 \subseteq \Delta^1 \dots \subseteq \Delta^n$ and for every $j = 1, \dots n$, $\Delta^j \vdash d_j \in [\![\Gamma, \overrightarrow{x : b} \vdash b_j[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{d}/\vec{x}], \xi \cup [\vec{C}/\vec{x}]|_{\Delta^j}, \Delta^j}$ and $C_j = \emptyset$ if $d_j \in Obj$ and $C_j \in \mathcal{C}_{\Delta^j, d_j}$ otherwise, then for every $i = 1, \dots n$ we have $\Delta^i \vdash s(\vec{a}\rho)d_1, \dots d_i \in [\![\Gamma, \overrightarrow{x : b} \vdash (x_{i+1} : b_{i+1}[\vec{a}/\vec{p}]) \dots (x_n : b_n[\vec{a}/\vec{p}]).\star]\!]_{\rho \cup [\vec{d}/\vec{x}], \xi \cup [\vec{C}/\vec{x}]|_{\Delta^i}, \Delta^i}$.
The proof is done by induction on i decreasing from n down to 0.
For $i = n$, the claim is obviously true, as the interpretation of $\star$ is equal to $\mathcal{SN}_{\Delta^n, \star}$ and we know by definition of interpretation that $\vec{a}\rho$ and $\vec{d}$ are strongly normalizing.
Now, let us prove the claim for $i - 1$, supposing that it holds for $i > 0$. Let us denote by T the type $(x_{i+1} : b_{i+1}[\vec{a}/\vec{p}]) \dots (x_n : b_n[\vec{a}/\vec{p}]).\star$ and suppose that b_i is a kind (for b_i a type the proof is similar). We have to show that $\Delta^{i-1} \vdash s(\vec{a}\rho)d_1, \dots d_{i-1} \in [\![\Gamma, \overrightarrow{x : b} \vdash (x_i : b_i[\vec{a}/\vec{p}]).T]\!]_{\rho \cup [\vec{d}/\vec{x}], \xi \cup [\vec{C}/\vec{x}], \Delta^{i-1}}$. By the definition of interpretation, $[\![\Gamma, \overrightarrow{x : b} \vdash (x_i : b_i[\vec{a}/\vec{p}]).T]\!]_{\rho \cup [\vec{d}/\vec{x}], \xi \cup [\vec{C}/\vec{x}], \Delta^{i-1}}$ equals to the set $\{\Delta' \vdash M \mid \Delta' \vdash M : ((x_i : b_i[\vec{a}/\vec{p}]).T)\rho \cup [\vec{d}/\vec{x}],\ \Delta' \subseteq \Delta^{i-1}$, and $\forall \Delta'' \subseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma, \overrightarrow{x : b} \vdash b_i[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{d}/\vec{x}], (\xi \cup [\vec{C}/\vec{x}])|_{\Delta''}, \Delta''},\ \forall D \in \mathcal{C}_{\Delta'', A},\quad \Delta'' \vdash M\,A \in [\![\Gamma, \overrightarrow{x : b} \vdash T]\!]_{\rho \cup [\vec{d}/\vec{x}] \cup [A/x], (\xi \cup [\vec{C}/\vec{x}] \cup [D/x])|_{\Delta''}, \Delta''}\}$ A direct application of the induction hypothesis shows that $s(\vec{a}\rho)d_1, \dots d_{i-1}$ belongs to the set described above.

Constructor As in the previous subcase, let us start by specializing the typing rule for c of type $(p_1 : P_1) \dots (p_r : P_r).(z_1 : d_1) \dots (z_k : d_k).s(\vec{p})w_1 \dots w_n$, c being a constructor of an inductive type $s : (p_1 : P_1) \dots (p_r : P_r).(x_1 : b_1) \dots (x_n : b_n).\star$.

$$\frac{\forall i \quad \Gamma \vdash a_i \ : \ P_i[\vec{a}/\vec{p}] \qquad\qquad \forall j \quad \Gamma \vdash t_j \ : \ d_j[\vec{a}/\vec{p}, \vec{t}/\vec{z}]}{\Gamma \vdash c(\vec{a}, \vec{t}) \ : \ s(\vec{a})w_1[\vec{a}/\vec{p}, \vec{t}/\vec{z}], \dots w_n[\vec{a}/\vec{p}, \vec{t}/\vec{z}]}$$

We have to show that $\Delta \vdash c(\vec{a}\rho, \vec{t}\rho) \in [\![\Gamma \vdash s(\vec{a})\overrightarrow{w[\vec{a}/\vec{p}, \vec{t}/\vec{z}]}]\!]_{\rho, \xi, \Delta}$, assuming that $\Delta \vdash a_i\rho \in [\![\Gamma \vdash P_i[\vec{a}/\vec{p}]]\!]_{\rho, \xi, \Delta}$ for all a_i and $\Delta \vdash t_j\rho \in [\![\Gamma \vdash d_j[\vec{a}/\vec{p}, \vec{t}/\vec{z}]]\!]_{\rho, \xi, \Delta}$ for all t_j. Since $[\![\Gamma \vdash d_j[\vec{a}/\vec{p}, \vec{t}/\vec{z}]]\!]_{\rho, \xi, \Delta} = [\![\Gamma, \overrightarrow{z : d[\vec{a}/\vec{p}]} \vdash d_j[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{t}\rho/\vec{z}], \xi, \Delta}$, by Lemma 5.3.2, we complete the proof by applying the **composition** case of Lemma 5.2.13.

Function The rule for function symbols is exactly the same as for constructors. If $f : (p_1 : P_1) \dots (p_r : P_r).(x_1 : b_1) \dots (x_n : b_n).c$ then it is as follows:

$$\frac{\forall i \quad \Gamma \vdash a_i \ : \ P_i[\vec{a}/\vec{p}] \qquad\qquad \forall j \quad \Gamma \vdash t_j \ : \ b_j[\vec{a}/\vec{p}, \vec{t}/\vec{x}]}{\Gamma \vdash f(\vec{a}, \vec{t}) \ : \ c[\vec{a}/\vec{p}, \vec{t}/\vec{x}]}$$

But the proof that $\Delta \vdash f(\vec{a}\rho, \vec{t}\rho) \in [\![\Gamma \vdash c[\vec{a}/\vec{p}, \vec{t}/\vec{x}]]\!]_{\rho, \xi, \Delta}$ is completely different from the previous one, because a function symbol headed term may rewrite.
By induction hypothesis we know that $\Delta \vdash a_i\rho \in [\![\Gamma \vdash P_i[\vec{a}/\vec{p}]]\!]_{\rho, \xi, \Delta}$ for all a_i and $\Delta \vdash t_i\rho \in [\![\Gamma \vdash b_i[\vec{a}/\vec{p}, \vec{t}/\vec{x}]]\!]_{\rho, \xi, \Delta}$ for all t_i. By Lemma 5.3.2, we know also that $[\![\Gamma \vdash P_i[\vec{a}/\vec{p}]]\!]_{\rho, \xi, \Delta} = [\![\overrightarrow{p : P} \vdash P_i]\!]_{[\vec{a}\rho/\vec{p}], [\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho, \xi, \Delta}}/\vec{p}], \Delta}$ and $[\![\Gamma \vdash b_i[\vec{a}/\vec{p}, \vec{t}/\vec{x}]]\!]_{\rho, \xi, \Delta} = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_i]\!]_{[\vec{a}\rho/\vec{p}, \vec{t}\rho/\vec{x}], [\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho, \xi, \Delta}}/\vec{p}], \Delta}$.
We can now apply the Fun Lemma (with $\alpha = a\rho$, $\sigma = t\rho$ and $\zeta = [\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho, \xi, \Delta}/\vec{p}}])$

to obtain $\Delta \vdash f(\vec{a}\rho, \vec{t}\rho) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{a}\rho/\vec{p}, \vec{t}\rho/\vec{x}], [\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}], \Delta}$. We conclude by Lemma 5.3.2 that $\Delta \vdash f(\vec{a}\rho, \vec{t}\rho) \in [\![\Gamma \vdash c[\vec{a}/\vec{p}, \vec{t}/\vec{x}]]\!]_{\rho,\xi,\Delta}$.

(abs)

$$\frac{\Gamma, x : a \vdash b \ : \ c \qquad \Gamma \vdash (x : a).c \ : \ p}{\Gamma \vdash \lambda x : a.b \ : \ (x : a).c} \ (x \notin dom(\Gamma), p \in \{\star, \square\})$$

We will do the proof only for the case when a is a kind.

We have to check that $\Delta \vdash \lambda x : a\rho.b\rho \in [\![\Gamma \vdash (x : a).c]\!]_{\rho,\xi,\Delta}$. By the definition of interpretations, it is equivalent to verifying that $\forall \Delta'' \sqsubseteq \Delta$, $\forall \Delta'' \vdash N \in [\![\Gamma \vdash a]\!]_{\rho,\xi|_{\Delta''},\Delta''}$, $\forall C \in \mathcal{C}_{\Delta'',N}$, $\Delta'' \vdash (\lambda x : a\rho.b\rho)\, N \in [\![\Gamma, x : a \vdash c]\!]_{\rho \cup [N/x], \xi|_{\Delta''} \cup [C/x], \Delta''}$.

By induction hypothesis we know that for every ρ', ξ' such that $[\![\rho', \xi']\!] \ : \ (\Gamma, x : a) \to \Delta''$ we have $\Delta'' \vdash b\rho' \in [\![\Gamma, x : a \vdash c]\!]_{\rho',\xi',\Delta''}$. Let us take:

$$\rho' = \rho \cup [N/x], \quad \xi' = \xi|_{\Delta''} \cup [C/x]$$

If these ρ' and ξ' are compatible, then we have $\Delta'' \vdash b(\rho \cup [N/x]) \in [\![\Gamma, x : a \vdash c]\!]_{\rho \cup [N/x], \xi|_{\Delta''} \cup [C/x], \Delta''}$. Hence, $\Delta'' \vdash (\lambda x : a\rho.b\rho)\, N \in [\![\Gamma, x : a \vdash c]\!]_{\rho \cup [N/x], \xi|_{\Delta''} \cup [C/x], \Delta''}$ by Lemma 5.1.6 ($N \in \mathcal{SN}$ by assumption and $a\rho \in \mathcal{SN}$ by induction hypothesis).

We are left to check whether $[\![\rho', \xi']\!] \ : \ (\Gamma, x : a) \to \Delta''$, i.e. $\Delta'' \vdash y\rho' \in [\![(\Gamma, x : a) \vdash (\Gamma, x : a)(y)]\!]_{\rho',\xi',\Delta''}$ for all $y \in dom(\Gamma, x : a)$. If $y = x$ then $\Delta'' \vdash N \in [\![\Gamma, x : a \vdash a]\!]_{\rho',\xi',\Delta''}$ is a consequence of applying Lemma 5.3.1 to $\Delta'' \vdash N \in [\![\Gamma \vdash a]\!]_{\rho,\xi|_{\Delta''},\Delta''}$. If $y \neq x$ then $[\![\rho, \xi]\!] \ : \ \Gamma \to \Delta$ implies $\Delta \vdash y\rho \in [\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi,\Delta}$. To deduce $\Delta'' \vdash y(\rho \cup [N/x]) \in [\![\Gamma, x : a \vdash (\Gamma, x : a)(y)]\!]_{\rho \cup [N/x], \xi|_{\Delta''} \cup [C/x], \Delta''}$ it is sufficient to note that:

- $\Delta'' \vdash y\rho \in [\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi|_{\Delta''},\Delta''}$ because, by the definition of restriction, we have $[\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi|_{\Delta''},\Delta''} = ([\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi,\Delta})|_{\Delta''}$,
- $[\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi|_{\Delta''},\Delta''} = [\![\Gamma, x : a \vdash (\Gamma, x : a)(y)]\!]_{\rho \cup [N/x], \xi|_{\Delta''} \cup [C/x], \Delta''}$, from Lemma 5.3.1 and because $x \notin \Gamma(y)$.

(prod)

$$\frac{\Gamma \vdash a \ : \ p \qquad \Gamma, x : a \vdash b \ : \ q}{\Gamma \vdash (x : a).b \ : \ q} \ (x \notin dom(\Gamma), p, q \in \{\star, \square\})$$

We have to check that $\Delta \vdash (x : a\rho).b\rho \in [\![\Gamma \vdash q]\!]_{\rho,\xi,\Delta}$. Note that $[\![\Gamma \vdash q]\!]_{\rho,\xi,\Delta} = \mathcal{SN}_{\Delta,q}$, as $q = \star$ or $q = \square$. Trivially $\Delta \vdash (x : a\rho).b\rho \ : \ q$. We have also $\Delta \vdash (x : a\rho).b\rho \in \mathcal{SN}_{\Delta}$, because, by induction hypothesis, $\Delta \vdash a\rho \in \mathcal{SN}_{\Delta,p}$, and $\Delta, x : a\rho \vdash b\rho \in \mathcal{SN}_{(\Delta, x : a\rho),q}$.

(app)

$$\frac{\Gamma \vdash a \ : \ (x : b).c \qquad \Gamma \vdash d \ : \ b}{\Gamma \vdash a\, d \ : \ c[d/x]}$$

We will do the proof only for the case when b is a kind.

By induction hypothesis, we know that $\Delta \vdash a\rho \in [\![\Gamma \vdash (x : b).c]\!]_{\Delta,\xi,\rho}$ and $\Delta \vdash d\rho \in [\![\Gamma \vdash b]\!]_{\Delta,\xi,\rho}$. By the definition of the interpretation of a product type, we get $\Delta \vdash a\rho\, d\rho \in [\![\Gamma, x : b \vdash c]\!]_{\rho \cup [d\rho/x], \xi \cup [[\![\Gamma \vdash d]\!]_{\rho,\xi,\Delta}/x], \Delta}$ and by applying Lemma 5.3.2 we conclude that $\Delta \vdash a\rho\, d\rho \in [\![\Gamma \vdash c[d/x]]\!]_{\rho,\xi,\Delta}$.

(conv)

$$\frac{\Gamma \vdash a : b \qquad \Gamma \vdash b' : p}{\Gamma \vdash a : b'} \; (p \in \{\star, \Box\} \text{ and } \Gamma \vdash b \overset{*}{\leftrightarrow} b')$$

The conclusion follows from the induction hypothesis ($\Delta \vdash a\rho \in [\![\Gamma \vdash b]\!]_{\rho,\xi,\Delta}$) and Lemma 5.3.4 by which $[\![\Gamma \vdash b]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash b']\!]_{\rho,\xi,\Delta}$. $\Box$

Theorem 2
Every term of CC+H is strongly normalizing.

Proof
The term $\Box$ is obviously strongly normalizing. For every other term M there are Γ and A such that $\Gamma \vdash M : A$ and we will use Lemma 5.5.2 to show that it is strongly normalizing.

Let's take an identity substitution $\rho_{id} : \Gamma \to \Gamma$ and a canonical candidate assignment ξ_{can}:

$$x\rho_{id} = x \quad \text{for all } x \in dom(\Gamma)$$
$$x\xi_{can} = can_{\Gamma,x} \quad \text{for all } x \in dom(\Gamma) \cap \mathcal{V}ar^{\Box}$$

where canonical candidates are defined in Lemma 5.1.5

These ρ_{id} and ξ_{can} trivially satisfy $[\![\rho_{id}, \xi_{can}]\!] : \Gamma \to \Gamma$. Hence, by Lemma 5.5.2, $\Gamma \vdash M \in [\![\Gamma \vdash A]\!]_{\rho_{id},\xi_{can},\Gamma}$. It completes the proof, as every interpretation is a candidate and every candidate is a subset of the set of strongly normalizing terms. $\Box$

6 CC + R – Calculus of Constructions with rewriting generated by user-defined rules

In the previous sections we have proved that the Calculus of Constructions with rewriting defined by HORPO is terminating. But in practice, we do not want to consider all HORPO. We have instead some user-defined rules and we want to be sure that the Calculus of Constructions with the conversion enriched by the rewrite relation induced by these rules is terminating. For a given term rewriting system R, let us call such a system CC+R. In this section we will show, that for the rules orientable by HORPO, CC+R is terminating.

Of course, we suppose that the rewrite rules come together with the signature containing all constants with their arities and types and that this signature is well-formed according to Definition 2.4.3 (the latter condition can be checked automatically).

The form of the rules must correspond to the form expected by HORPO. The general and quite standard condition is that the left-hand and right-hand sides (l, r) have the same type in the environment consisting of the free variables of l and r. Additionally, the left-hand side must start with a function symbol and its parameter arguments must be distinct variables.

Definition 6.0.3 (*Rule and term rewriting system*)
A rule is a quadruple (G, l, r, T) where:

1. $G \vdash_{CC} l : T$ and $G \vdash_{CC} r : T$,

2. $l = f(\vec{p}, \vec{a})$ and $G = (p_1 : P_1)\dots(p_r : P_r), G'$, provided $f \in \mathcal{F}^{(r,n)}$ and $f : (p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c \in \Sigma$.

The rules are written as $G \vdash l \to r : T$. A *term rewriting system* is simply a set of rules.

Definition 6.0.4 (*CC+R*)
Given a term rewriting system R, the system CC+R is defined just like CC+H, with two exceptions. In case **(rew)** we use rules of R instead of HORPO judgments, and consequently in case **(conv)** we replace $\to_H$ by $\to_R$.

$$\textbf{(rew)} \quad \frac{G \vdash l \to r : T \in R \qquad a|_p = l\theta \qquad \theta : G \to \Gamma, \mathcal{PV}(a,p)}{\Gamma \vdash a[l\theta]_p \to_R a[r\theta]_p}$$

$$\textbf{(conv)} \quad \frac{\begin{array}{cc} \Gamma \vdash a : b & \Gamma \vdash b' : p \\ \Gamma \vdash b(\to_\beta \cup \to_R)^* b' & \text{or} \quad \Gamma \vdash b'(\to_\beta \cup \to_R)^* b \end{array}}{\Gamma \vdash a : b'} \quad (p \in \{\star, \Box\})$$

Just like in the case of CC+H, we write $\Gamma \vdash a \to_{R\cup\beta} b$ for any of $\Gamma \vdash a \to_R b$ and $a \to_\beta b$. We denote by $\overset{*}{\to}{}^{\Gamma}_{R\cup\beta}$ the reflexive, transitive closure of $\to^{\Gamma}_{R\cup\beta}$, and by $\overset{*}{\leftrightarrow}{}^{\Gamma}_{R\cup\beta}$ the conversion relation of CC+R.

Lemma 6.0.5
Given a term rewriting system R, if there is a precedence and status such that every rule $\Gamma \vdash l \to r : T \in R$ verifies $\Gamma \vdash l > r : T$ then the following statements hold:

1. every term of CC+R is a term of CC+H,
2. the reduction relation is richer in CC+H than in CC+R ($\to^{\Gamma}_{R\cup\beta} \subseteq \to^{\Gamma}$),
3. the conversion relation is richer in CC+H than in CC+R ($\overset{*}{\leftrightarrow}{}^{\Gamma}_{R\cup\beta} \subseteq \overset{*}{\leftrightarrow}{}^{\Gamma}$),
4. the strong normalization of CC+H implies the strong normalization of CC+R.

7 On the star dependency condition

This section provides an explanation for imposing star dependency (see Definition 2.4.1) on function and constructor symbols, and parametricity (see Definition 3.1.1) on HORPO judgments.

7.1 Star dependency, parametricity and HORPO

In our paper, the general condition imposed on the left-hand side of HORPO judgments and rewrite rules comes in two parts: star dependency and parametricity conditions. Both these conditions follow from the proof of the Fun Lemma. In this proof we need to know that, if the left-hand side of the rewrite rule starts with a function symbol f, then all meaningful type arguments of f must be different variables (where the ith argument is meaningful for $f : \overrightarrow{(x : b)}.c$ if x_i appears somewhere in c or b_j for $j > i$). The role of the star dependency condition is to find those meaningful arguments (they are included in parameters) and the role of parametricity condition is to ensure that they are distinct variables (all parameter arguments must be distinct variables).

Let us see with an example that these conditions are important; let

$$J : (A : \star)(B : \star)A \to B \to A \qquad \text{have a rule}$$

$$J(X,\ X,\ a,\ b) \to b$$

The arguments on parameter positions are not distinct variables, therefore star dependency is not satisfied. But let us try to use this rule in the Fun Lemma.

In the proof of the Fun Lemma, by the hypothesis about arguments of J we would know that a belongs to the interpretation of A ($\Delta \vdash a \in [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$), b belongs to the interpretation of B ($\Delta \vdash b \in [\![\Gamma \vdash B]\!]_{\rho,\xi,\Delta}$) and ρ satisfies $A\rho = B\rho = X$. The goal would be to prove that $J(X,\ X,\ a,\ b)$ belongs to the interpretation of A and we would try to do it by showing that b – the only reduct of $J(X,\ X,\ a,\ b)$ – belongs to this interpretation. But this could not be done, since the information that $\Delta \vdash b \in [\![\Gamma \vdash B]\!]_{\rho,\xi,\Delta} = B\xi$ in general would not imply that $\Delta \vdash b \in [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = A\xi$, because the candidate $A\xi$ may be different from $B\xi$ even if $A\rho = B\rho$.

We would therefore not succeed to do the proof of the Fun Lemma, because the fact that the right-hand side is of adequate type (b is of type X) would not be enough here.

Looking at the type of J, $(A : \star)(B : \star)A \to B \to A$, we see that the second argument, B, is the type of the fourth one, and the first argument, A, is the type of the result.

In our rule $J(X,\ X,\ a,\ b) \to b$, both A and B are substituted by X, and this is why we can return the fourth argument, b, as a result. We can say that b is of adequate type only "accidentally". And it turns out that this coincidence leads to nontermination, as shown below.

Let $U = (\beta : \star).\beta \to \beta$ and let us define the term c of type U:

$$c = \lambda\beta : \star.\lambda x : \beta.J(\beta,\ U \to U,\ x,\ t)$$

where $t = \lambda z : U.(z\ (U \to U)\ (\lambda x : U.x)\ z)$.

Consider now the term $c\ (U \to U)\ (\lambda x : U.x)\ c$ of type U.

$$\begin{aligned} &c\ (U \to U)\ (\lambda x : U.x)\ c \to_\beta^* \\ &\qquad J(U \to U,\ U \to U,\ \lambda x : U.x,\ t)\, c \to \\ &\qquad\qquad t\, c \to_\beta c\ (U \to U)\ (\lambda x : U.x)\ c \to \ldots \end{aligned}$$

So, we have found a nonterminating reduction sequence.

The example of the function symbol J and its rule derives from the one presented by Girard (1971); it was shown to the author by Christine Paulin.

The conclusion is that even very simple rules that do not have distinct variables as meaningful type arguments may lead to nontermination.

Now, let us briefly discuss the difference between linearity and the requirement that arguments on parameter positions are distinct variables. First, let

$$f : (n : nat)(List\ n) \to (m : nat)(List\ m) \to List\ n \qquad \text{have a rule}$$

$$f(n,\ l_1,\ n,\ l_2) \to l_2$$

This rule is nonlinear, but f has no parameters. It is a valid rewrite rule and it is accepted by HORPO. Let now

$$f : (A : \star)A \to A \to A \qquad \text{have a rule}$$

$$f(A,\ a,\ f(A,\ b,\ c)) \to a$$

From the type of f it follows that the first argument must be a parameter. In this rule we have two occurrences of A on parameter positions, but they appear in different function calls. Consequently, it is a valid rule accepted by HORPO. Finally, let

$$f : (A : \star)(a : A)(n : nat) \to A \qquad \text{have a rule}$$

$$f(nat,\ a,\ n) \to n$$

The rule is linear, but is not accepted, because f must have at least one parameter and, consequently, the first argument of f should be a variable (in fact if we try to do the proof of the Fun Lemma with this rule, we will encounter the same problems as for the rule $J(X,\ X,\ a,\ b) \to b$).

Therefore, star dependency condition is not equivalent to linearity of parameter variables.

7.2 Star dependency and inductive types

The star dependency condition on constructors is needed to define the interpretation of inductive types.

In our framework, the interpretation of inductive types has to be defined in such a way that the reducibility of the constructor headed term $c(\vec{u})$ implies the reducibility of every u. To see this, consider the elimination rule for natural numbers:

$$nat_{dep}(P, z, f, s(n)) \to @(f, n, nat_{dep}(P, z, f, n))$$

During the proof that $nat_{dep}(P, z, f, s(n))$ belongs to the interpretation of its type, I, one assumes that P, z, f and $s(n)$ belong to the interpretations of their types, and tries to show that every reduct of $nat_{dep}(P, z, f, s(n))$ belongs to I.

The fact that $@(f, n, nat_{dep}(P, z, f, n)) \in I$ can be obtained only by analyzing its direct subterms and combining interpretations to which they belong. In particular in our example, we are led to the problem how to deduce that n belongs to the interpretation of its type from the hypothesis that so does $s(n)$. In other words, we want to know that “reaching under a constructor” is a safe operation from the reducibility point of view.

In the definition of the interpretation of the inductive type s, if a constructor headed term $c(\vec{a'}, \vec{N})$ belongs to this interpretation, then every a'_j and N_j must belong to a certain, explicitly written interpretation. More precisely, if $s : (p_1 : P_1)\dots(p_r : P_r).(x_1 : b_1)\dots(x_n : b_n).\star$ and $c : (p_1 : P_1)\dots(p_r : P_r).(z_1 : d_1),\dots(z_k : d_k).s(\vec{p})w_1\dots w_n$, then a'_j must belong to the interpretation of P_j, and N_j to the interpretation of d_j. Since P_j and d_j may have big free variables, defining their interpretations requires an adequate candidate assignment. Candidates assigned to big parameters are just candidates corresponding to parameters of s and they are given. But how to get candidates for big non-parameter variables, that would not contradict the well-foundedness of the definition of interpretation? In particular, assigning the interpretation of N_i to $z_i \in \mathcal{V}ar^{\square}$ is no good, because the interpretation of N_i may not be defined yet.

Summarizing, we know how to define the interpretation of an inductive type if every input type of its constructor depends only on those big variables that are parameters. And this is exactly what the star dependency condition states.

In the literature, there are other proofs of strong normalization for calculi which contain the Calculus of Constructions and inductive types with their elimination rules (traditionally, small elimination is an object-level rewriting rule and large (or strong) elimination is a type-level rule). In the two examples that we know (Werner, 1994; Stefanova, 1999), there is no additional restriction (apart from positivity) on types of constructors in order to define the interpretation of inductive types. But neither of the authors needed to safely reach under a constructor.

In (1994) Benjamin Werner defines the interpretation of inductive types so as to have small elimination for free, and he can do so because eliminations are the only rewriting rules in his system.

The definition of interpretation of inductive types, presented by Milena Stefanova in (1999), resembles very much to ours, but instead of giving explicitly the candidate assignment and interpretations to which belong immediate subterms of a constructor headed term, the author assumes only that an adequate candidate assignment exists. This is sufficient for small elimination (and in fact we could do it as well), but it is not enough for reaching under a constructor, which seems necessary when we have not only eliminations but also other rules as we do.

On the other hand, it turns out that in both papers (Werner, 1994; Stefanova, 1999) large elimination requires an additional condition on types of constructors. In Werner (1994), constructors must be *small*, i.e. every argument of a constructor must be either a parameter or an object, and in Stefanova (1999), the type of constructors must satisfy the *safeness* condition, which allows a little bit more inductive types than the small constructors condition.

Both conditions, that is small constructors and safeness, are strictly stronger than the star dependency, which is not surprising, since large elimination is much more powerful than reaching under a constructor (in our framework, even if we can reach to some big term N, we cannot return this N as a result – the right-hand side of a rewrite rule – because we consider only object-level rewriting).

To conclude, we think that reaching under a constructor in the context of the object-level rewriting is something between small and large eliminations and star dependency is the necessary condition which makes it possible.

8 Examples

Before we give some examples of dependently typed rules, let us mention the fact that all positive examples given in Jouannaud & Rubio (1999) are also accepted by our HORPO.

In the examples below, for the sake of clarity we resigned from parentheses after type constructors (we write *List A* and not *List*(A)). We also omit types in HORPO judgments, whenever they are clear form the context.

Example 10 (*Dependent elimination on natural numbers*)
Let $nat_{dep} : (P : nat \to \star)(u : P\ 0)(v : (n : nat)(P\ n) \to (P\ s(n)))(m : nat)(P\ m) \in \mathcal{F}^{(1,3)}$ have the multiset status and let us denote by Γ the environment:

$$P : nat \to \star,\ z : P\ 0,\ f : (n : nat)(P\ n) \to (P\ s(n)),\ n : nat$$

Consider the rules:

$$\Gamma \vdash nat_{dep}(P,z,f,0) \to z\ \ : P\ 0$$
$$\Gamma \vdash nat_{dep}(P,z,f,s(n)) \to @(f,n,nat_{dep}(P,z,f,n))\ \ : P\ s(n)$$

The first rule is easily oriented by case I.1 of HORPO, as z is an immediate subterm of the left-hand side of the rule.

To verify the second rule, we first use case I.6 of HORPO and we are left to show that $\Gamma \vdash f \succeq^{\mathcal{CS}} f$, $\Gamma \vdash s(n) \succeq^{\mathcal{CS}} n$ and $nat_{dep}(P,z,f,n) \in CCl_{nat_{dep}}(P,z,f,s(n))$. Case I.6 is applicable because $AppCon(\Gamma \vdash @(f,n,nat_{dep}(P,z,f,n)))$ holds (as $\Gamma \vdash f : (n : nat)(P\ n) \to (P\ s(n))$ and nat and $(P\ n)$ are of type $\star$).

The first two inequalities are trivial. Let us detail the proof of $nat_{dep}(P,z,f,n) \in CCl_{nat_{dep}}(P,z,f,s(n))$. By the definition of the initial set the tuples

$$(P,\ \emptyset;\emptyset \vdash nat \to \star,\ [P/P])$$
$$(z,\ \emptyset;\emptyset \vdash P\ 0,\ [P/P])$$
$$(f,\ u : P\ 0;\emptyset \vdash (n : nat)(P\ n) \to (P\ s(n)),\ [P/P,z/u])$$
$$(s(n),\ u : P\ 0, v : (n : nat)(P\ n) \to (P\ s(n));\emptyset \vdash nat,\ [P/P,z/u,f/v])$$

are in $CCl_{nat_{dep}}(P,z,f,s(n))$. By applying **Constructor–decomposition** to the last tuple, we get another tuple in the computable closure, namely:

$$(n,\ u : P\ 0, v : (n : nat)(P\ n) \to (P\ s(n));\emptyset \vdash nat,\ [P/P,z/u,f/v])$$

Now we can apply **Recursive call** to obtain:

$$(nat_{dep}(P,z,f,n),\ u : P\ 0, v : (n : nat)(P\ n) \to (P\ s(n)), m : nat;\emptyset \vdash P\ m,\ \gamma)$$

where $\gamma = [P/P,z/u,f/v,n/m]$. The use of this rule is justified, because $\Gamma \vdash (P,z,f,s(n)) \succ^{\mathcal{CS}}_{mul} (P,z,f,n)$ (as $\Gamma \vdash s(n) \succ^{\mathcal{CS}} n$). Since $nat_{dep}(P,z,f,n)$ and $P\ m$ are clean we conclude that $nat_{dep}(P,z,f,n) \in CCl_{nat_{dep}}(P,z,f,s(n))$.

Note that we could not compare $nat_{dep}(P,z,f,s(n))$ with $nat_{dep}(P,z,f,n)$ by case I.3 of HORPO, because these terms have different types ($P\ s(n)$ and $P\ n$).

Example 11 (*Dependent elimination on lists*)
Let $list_{dep} : (A : \star)(P : (List\ A) \to \star)(P\ nil(A)) \to ((a : A)(l : List\ A)(P\ l) \to (P\ cons(A,\ ,a,\ l))) \to (l : List\ A)(P\ l)$ be a symbol with two parameters and the multiset status.

Let us denote by Γ the environment:

$$A:\star,\quad P:List\ A\rightarrow\star,\quad f_1:P\ nil(A),$$
$$f_2:(a:A)(l:List\ A)(P\ l)\rightarrow(P\ cons(A,\ a,\ l)),\quad a:A,\quad l:(List\ A)$$

and let us consider the rules:

$$\begin{array}{l}\Gamma\ \vdash\ list_{dep}(A,P,f_1,f_2,nil(A))\ \rightarrow\ f_1\ :\ P\ nil(A)\\ \Gamma\ \vdash\ list_{dep}(A,P,f_1,f_2,cons(A,\ a,\ l))\\ \qquad\qquad\qquad\rightarrow\ @(f_2,a,l,list_{dep}(A,P,f_1,f_2,l))\ :\ P\ cons(A,\ a,\ l)\end{array}$$

The first rule is easily oriented, as f_1 is an immediate subterm of the left-hand side of the rule (case I.1 of HORPO).

To verify the second rule, we first use case I.6 of HORPO and we are left to show that $\Gamma\vdash f_2\succeq^{\mathcal{CS}} f_2$, $\Gamma\vdash cons(A,\ a,\ l)\succeq^{\mathcal{CS}} l$, $\Gamma\vdash cons(A,\ a,\ l)\succeq^{\mathcal{CS}} a$ and $list_{dep}(A,P,f_1,f_2,l)\in CCl_{list_{dep}}(A,P,f_1,f_2,cons(A,\ a,\ l))$. The application condition holds, since $\Gamma\vdash f_2\ :\ (a:A)(l:List\ A)(P,\ l)\rightarrow(P\ cons(A,\ a,\ l))$ and A, $List\ A$ and $(P\ l)$ are of type $\star$.

The first two inequalities are trivial. In the third one we really use the constructor part of $\succeq^{\mathcal{CS}}$, since $cons(A,\ a,\ l)$ and a have different types. Note that it is the only way to limit this a, as there are no immediate subterms of the left-hand side $list_{dep}(A,P,f_1,f_2,cons(A,\ a,\ l))$ of type A. To get the fourth assertion, we first decompose the constructor $cons$, to get $a,l\in CCl_{list_{dep}}(A,P,f_1,f_2,cons(A,\ a,\ l))$, and then apply $list_{dep}$ to its arguments (it is possible, as (P,f_1,f_2,l) is smaller than $(P,f_1,f_2,cons(A,\ a,\ l))$ in $\succ^{\mathcal{CS}}_{mul}$). Note that, like in the previous example, we could not compare the whole left-hand side $list_{dep}(A,P,f_1,f_2,cons(A,\ a,\ l))$ with $list_{dep}(A,P,f_1,f_2,l)$ by case I.3 of HORPO.

Example 12 (*fold_left and fold_right on lists with length*)
Let us take the following inductive definition:

$$\begin{array}{l}Ind[A:\star](LList:nat\rightarrow\star\ :=Lnil:\overline{LList}\ 0,\\ \qquad\qquad\qquad\qquad\qquad Lcons:(n:nat)A\rightarrow\overline{LList}\ n\rightarrow\overline{LList}\ s(n))\end{array}$$

and suppose that $fold\_left,fold\_right\ :\ (A:\star)(B:\star)(A\rightarrow B\rightarrow B)\rightarrow(n:nat)(LList\ A\ n)\rightarrow B\rightarrow B\in\mathcal{F}^{(2,4)}$ have lexicographic status.

Let us denote by Γ the environment

$$A:\star,\quad B:\star,\quad f:A\rightarrow B\rightarrow B,\quad a:A,\quad n:nat,\quad l:LList\ A\ n,\quad x:B$$

and let us consider the rules:

$$\begin{array}{l}\Gamma\ \vdash\ fold\_left(A,B,f,0,Lnil(A,\ 0),x)\ \rightarrow\ x\ :\ B\\ \Gamma\ \vdash\ fold\_left(A,B,f,s(n),Lcons(A,s(n),a,l),x)\\ \qquad\qquad\qquad\rightarrow\ fold\_left(A,B,f,n,l,@(f,a,x))\ :\ B\end{array}$$

$$\begin{array}{l}\Gamma\ \vdash\ fold\_right(A,B,f,0,Lnil(A,\ 0),x)\ \rightarrow\ x\ :\ B\\ \Gamma\ \vdash\ fold\_right(A,B,f,s(n),Lcons(A,s(n),a,l),x)\\ \qquad\qquad\qquad\rightarrow\ @(f,a,fold\_right(A,B,f,n,l,x))\ :\ B\end{array}$$

The first rules of $fold\_left$, $fold\_right$ are trivially accepted by case I.1 of HORPO.

To orient the second rule of *fold_left* we use the case lex (I.4) of HORPO and we are led to show $\Gamma \vdash s(n) \succ^{\mathcal{CS}} n$, $\Gamma \vdash Lcons(A,\ s(n),\ a,\ l) \succeq^{\mathcal{CS}} l$, which are obvious and $\Gamma \vdash fold\_left(A, B, f, s(n), Lcons(A, s(n), a, l), x) \succ @(f, a, x)\ :\ B$ that can be done by case I.6. The second rule of *fold_right* may be handled in a similar way as the second rule of elimination on lists.

Example 13 (*Membership in a list*)
Given an element a and a list l, the result of the *mem* function will be the proof that a is in l or the proof that every element of l is different from a. To formalize this we need to introduce inductive types representing disjunction, equality, false, membership and verification of a predicate by all elements of the list. The type $\sim A$ stands for $A \to False$.

$$Ind[A:\star, B:\star](Or:\star\ :=\ left : A \to \overline{Or},\ right : B \to \overline{Or})$$
$$Ind[A:\star, x:A](Eq : A \to \star\ :=\ refl\_eq : \overline{Eq}\ x)$$
$$Ind[\,](False:\star\ :=\ )$$

$$\begin{aligned} Ind[A:\star, a:A](In : (List\ A) \to \star\ :=&\\ &in\_hd : (l : (List\ A))(\overline{In}\ cons(A,\ a,\ l)),\\ &in\_tl : (l : (List\ A))(b:A)(\overline{In}\ l) \to (\overline{In}\ cons(A,\ b,\ l)) \end{aligned}$$

$$\begin{aligned} &Ind[A:\star, P : A \to \star](AllS : (List\ A) \to \star\ :=\\ &\quad allS\_nil : (\overline{AllS}\ nil(A)),\\ &\quad allS\_cons : (a:A)(l:(List\ A))(P\ a) \to (\overline{AllS}\ l) \to (\overline{AllS}\ cons(A,\ a,\ l))) \end{aligned}$$

Let *eq_ind*, *mem* and *if* have the following types:

$$\begin{aligned} eq_{ind}:&\quad (A:\star)(a:A)(P:A\to\star)(b:A)(Eq\ A\ a\ b) \to (P\ a) \to (P\ b) \in \mathcal{F}^{(3,3)},\\ mem:&\quad (A:\star)(a:A) \to ((c,d:A)(Or\ (Eq\ A\ c\ d)\ (\sim Eq\ A\ c\ d))) \to\\ &\quad (l:(List\ A))(Or\ (In\ A\ a\ l)\ (AllS\ A\ (\lambda x{:}A.\sim Eq\ A\ a\ x)\ l)) \in \mathcal{F}^{(1,3)},\\ if:&\quad (A:\star)(a,b:A)(l:List\ A) \to (Or\ (Eq\ A\ a\ b)(\sim Eq\ A\ a\ b)) \to\\ &\quad (Or\ (In\ A\ a\ l)\ (AllS\ A\ (\lambda x{:}A.\sim Eq\ A\ a\ x)\ l)) \to (Or\ (In\ A\ a\\ &\quad cons(A,\ b,\ l))\ (AllS\ A\ (\lambda x{:}A.\sim Eq\ A\ a\ x)\ cons(A,\ b,\ l))) \in \mathcal{F}^{(1,5)} \end{aligned}$$

And let us consider the rules (all given in the environment Γ):

$$\Gamma \equiv A:\star,\ x:A,\ P:A\to\star,\ h:(Eq\ A\ x\ x),\ eqDec:(c,d:A)(Or\ (Eq\ A\ c\ d)$$
$$(\sim Eq\ A\ c\ d))),\ b:A,\ l:List\ A,\ d_l:\Psi,\ d_r:\sim\Psi,\ i:(Or\ \Phi_3,\ \Upsilon_3),\ e_l:\Phi_3,\ e_r:\Upsilon_3$$

1. $\Gamma \vdash eq_{ind}(A,\ x,\ P,\ x,\ h,\ p) \to p :\ P\ x,$
2. $\Gamma \vdash mem(A,\ a,\ eqDec,\ nil(A))$
 $\to right(\Phi_1,\ \Upsilon_1,\ allS\_nil(A,\ (\lambda x{:}A.\sim Eq\ A\ a\ x))) :\ Or\ \Phi_1\ \Upsilon_1,$
3. $\Gamma \vdash mem(A,\ a,\ eqDec,\ cons(A,\ b,\ l))$
 $\to if(A,\ a,\ b,\ l,\ @(eqDec, a, b),\ mem(A,\ a,\ eqDec,\ l)) :\ Or\ \Phi_2\ \Upsilon_2$
4. $\Gamma \vdash if(A,\ a,\ b,\ l,\ left(\Psi,\ \sim\Psi,\ d_l),\ i)$
 $\to left(\Phi_2,\ \Upsilon_2,\ eq_{ind}(A,\ a,\ (\lambda x{:}A.In\ A\ a\ cons(A,\ x,\ l)),\ b,\ d_l,$
 $in\_hd(A,\ a,\ l))) :\ Or\ \Phi_2\ \Upsilon_2$

5. $\Gamma \vdash if(A,\ a,\ b,\ l,\ right(\Psi,\ \sim\Psi,\ d_r),\ left(\Phi_3,\ \Upsilon_3,\ e_l))$
 $\rightarrow left(\Phi_2,\ \Upsilon_2,\ in\_tl(A,\ a,\ l,\ b,\ e_l)) : Or\ \Phi_2\ \Upsilon_2$
6. $\Gamma \vdash if(A,\ a,\ b,\ l,\ right(\Psi,\ \sim\Psi,\ d_r),\ right(\Phi_3,\ \Upsilon_3,\ e_r))$
 $\rightarrow right(\Phi_2,\ \Upsilon_2,\ allS\_cons(A,\ (\lambda x{:}A. \sim Eq\ A\ a\ x),\ b,\ l,\ d_r,\ e_r)) : Or\ \Phi_2\ \Upsilon_2,$

where

$$\Psi = Eq\ A\ a\ b$$
$$\Phi_1 = In\ A\ a\ nil(A)$$
$$\Upsilon_1 = AllS\ A\ (\lambda x{:}A. \sim Eq\ A\ a\ x)\ nil(A)$$
$$\Phi_2 = In\ A\ a\ cons(A,\ b,\ l)$$
$$\Upsilon_2 = AllS\ A\ (\lambda x{:}A. \sim Eq\ A\ a\ x)\ cons(A,\ b,\ l)$$
$$\Phi_3 = In\ A\ a\ l$$
$$\Upsilon_3 = AllS\ A\ (\lambda x{:}A. \sim Eq\ A\ a\ x)\ l$$

For HORPO we choose the precedence $mem >_{\mathcal{F}} if >_{\mathcal{F}} eq_{ind}$ and we give to mem the multiset status. Let us concentrate on the second, third and fourth rules.

After applying case I.5 of HORPO to the second rule we are left to show that $allS\_nil(A,\ (\lambda x : A. \sim Eq\ A\ a\ x))$, Φ_1 and Υ_1 belong to the $CCl_{mem}(A, a, eqDec, nil(A))$. Let us denote the terms $(A, a, eqDec, nil(A))$ by $\vec{t}$. By the definition of the initial set, terms A, a, $eqDec$ and $nil(A)$ belong to $CCl_{mem}(\vec{t})$. To get $\Phi_1 \in CCl_{mem}(\vec{t})$, it suffices to apply constructor type rule to A, a, $nil(A)$. For Υ_1 and $allS\_nil(A, (\lambda x : A. \sim Eq\ A\ a\ x))$ the only difficulty is to check whether $(\lambda x : A. \sim Eq\ A\ a\ x) \in CCl_{mem}(\vec{t})$. The proof of this fact goes like this: by the type constructor composition rule we can get $Eq\ A\ a\ x$ with a free variable x. To obtain the negation, we should first introduce False to $CCl_{mem}(\vec{t})$ again by the type constructor composition rule and then use the product rule. We finish by abstracting the variable x. Once we have $(\lambda x : A. \sim Eq\ A\ a\ x) \in CCl_{mem}(\vec{t})$ it is sufficient to apply type constructor rule to get $\Upsilon_1 \in CCl_{mem}(\vec{t})$ and constructor composition rule to get $allS\_nil(A, (\lambda x{:}A. \sim Eq\ A\ a\ x)) \in CCl_{mem}(\vec{t})$.

In the third rule, since $mem >_{\mathcal{F}} if$, we are left to show that every immediate subterm of if is comparable with some subterm of mem or belongs to its computable closure. Terms A and a are just subterms of the left-hand side and b, l can be compared with $cons(A,\ b,\ l)$ by means of $\succ^{\mathcal{CS}}$. We will show that $@(eqDec, a, b)$ and $mem(A,\ a,\ eqDec,\ l)$ belong to the computable closure of mem. Note that we cannot use directly case mul (I.3) of HORPO to compare $mem(A,\ a,\ eqDec,\ l)$ with the whole left-hand side, as the types do not match. By the definition of the initial set, terms A, a, $eqDec$, b and l belong to $CCl_{mem}(A, a, eqDec, cons(A, b, l))$. By the application rule we get $@(eqDec, a, b)$ in the computable closure and by the recursive call rule $mem(A,\ a,\ eqDec,\ l) \in CCl_{mem}(A, a, eqDec, cons(A, b, l))$. The application of the recursive call rule is correct as $(A, a, eqDec, cons(A,\ b,\ l)) \succ^{\mathcal{CS}}_{mul} (A, a, eqDec, l)$.

For the fourth rule we first apply case I.5 of HORPO and we are led to show that Φ_2, Υ_2 and eq_{ind} applied to its arguments belong to the computable closure of if. The proof for Φ_2 is simple and the one for Υ_2 resembles very much to the proof given above that $\Upsilon_1 \in CCl_{mem}(A, a, eqDec, nil(A))$. For the eq_{ind}, it is obvious that $CCl_{if}(A, a, b, l, left(\Psi, \sim \Psi, d_l), i)$ contains A, a, b, l, d_l as they are either immediate or constructor subterms of the left-hand side. We will show that $(\lambda x : A.In\ A\ a\ cons(A,\ x,\ l))$ and $in\_hd(A,\ a,\ l)$ also belong to the computable closure. For the latter it is sufficient to use the constructor composition rule, for the former we have to first use the constructor composition rule to build $(cons\ A\ x\ l)$ with a free variable x, then the type constructor composition rule to have $(In\ A\ a\ cons(A,\ x,\ l))$ and finally the abstraction rule over x. We complete the proof by applying the precedence rule for eq_{ind}.

The acceptance proofs for other rules are similar to the proofs we have described.

The remainder of this section is devoted to examples which do not fit in our current framework.

Example 14 (*Constructors that do not satisfy star dependency condition*)

$$\begin{aligned}
&Ind[](StrangeList : \star \ := \\
&\quad nil : \overline{StrangeList} \\
&\quad cons : (X : \star)X \to \overline{StrangeList} \to \overline{StrangeList}
\end{aligned}$$

In this example the star dependency condition is violated, because X is a big variable that appears in the type of another argument of *cons*, but it is not a parameter. Consequently, the definition of well-formedness 2.4.3 is not satisfied and this inductive definition is not accepted in our system.

In practice types violating the star dependency condition are rare; we have searched Coq's contribution files, but have not found any (*nota bene*, this example is accepted by Coq).

Example 15 (*Elimination on Brouwer's ordinals*)
Let $ord_{ind} : (P : \star)P \to (ord \to P \to P) \to ((nat \to ord) \to (nat \to P) \to P) \to ord \to P \in \mathcal{F}^{(1,4)}$ have a rule:

$$\begin{aligned}
&P : \star, f_1 : P, f_2 : ord \to P \to P, f_3 : ((nat \to ord) \to (nat \to P) \to P), \\
&F : nat \to ord \vdash \\
&\quad ord_{ind}(P, f_1, f_2, f_3, lim(F)) \to @(f_3, F, \lambda y : nat.\, ord_{ind}(P, f_1, f_2, f_3, @(F, y)))
\end{aligned}$$

In the recursive call of ord_{ind} there is a term $@(F, y)$ where y is a variable bound above ord_{ind}. Our method is not sufficient here, but we hope to extend it very soon using ordinals related to the fixpoint construction of the interpretation of inductive types, like in (Werner, 1994).

9 Practical issues

The long term practical motivation of our work is the incorporation of rewriting in interactive theorem provers such as Coq (Barras *et al.*, 1999). To be suitable for implementation in such a framework, the termination criterion must have two important properties: decidability and modularity.

The first requirement is obvious. The second would allow the user to treat definitions by rewriting just like normal definitions (s)he is used to. It would be possible to enter one rewriting system, work with it, and then enter a new one on top of the first one. The modular termination criterion would then assure him(her) that the entire system is strongly normalizing.

9.1 Decidability

Concerning the first practical requirement, HORPO without computable closure is decidable in polynomial time (for a given precedence and status, and if we do not count the conversion used in the type checking). Full HORPO, computable closure included, is surely semi-decidable, but we cannot guarantee decidability. In fact, if we want to find out whether a given Γ-term t belongs to $CCl_f(\vec{l})$ using the rules given in Definition 3.3.2, we are faced with a potentially infinite nondeterminism.

If we try to proceed bottom up we first have to choose a term T and a substitution μ with the only constraint that $\Gamma \vdash t : T\mu$. Even though some (most?) choices are clearly wrong it is not obvious whether we can limit ourselves to a finite class. Similar problems arise when trying to apply backward the rules which do not enjoy the subterm property (like **Application to a nondependent argument** or **Reduction**).

On the other hand if we try the top down method, we have the same problem of potentially unbounded nondeterminism in the rule **Constructor**. Moreover we do not clearly know when to stop the research of a derivation, due to the rules like **Reduction** or **Constructor decomposition**, which may decrease the size of a leading term.

The trivial remedy to the problem of undecidability in an implementation is to limit to a given constant the number of rule applications. A better solution, in our opinion, would be to find out a decidable restriction of the computable closure, permitting at the same time to simplify its definition. We believe that removing the **Reduction** rule, and imposing small restrictions on a few others would permit such a simplification together with a strategy leading to a polynomial algorithm. But this problem obviously needs further work.

9.2 Modularity

As presented so far, the result of this paper applies to the case where the set of rules is given from the beginning.

In fact, we can do more, because our method for proving termination is *modular*. Modularity means that rules may be given step-by-step, a whole set at a time, possibly using the rules given at a previous stage for typing the new ones. Consider for example the rules for *plus* and *append* (inductive type *list* corresponds to integer lists with length).

$$\begin{aligned}
plus \quad &: \quad nat \to nat \to nat \\
&\quad\;\; plus\ O\ n \to n \\
append \quad &: \quad (n, m : nat)\ (list\ n) \to (list\ m) \to (list\ (plus\ n\ m)) \\
&\quad\;\; append\ O\ m\ nil\ lm \to lm
\end{aligned}$$

Unless we have the rule for *plus* in the conversion relation of the system, the types of the left-hand and right-hand side of the rule for *append* are different. Consequently, HORPO would reject this rule.

Our method of proving termination is modular because we can use it for proving the strong normalization of every CCR_n, defined by:

$$\begin{array}{rcl} CCR_0 & = & CC \\ CCR_n & = & CCR_{n-1} + R_n \end{array}$$

where R_n is a set of rewrite rules, potentially defined on symbols from the new signature Σ_n, and type-checked with CCR_{n-1}.

To this end, we can repeat the trick employed in the whole paper. Instead of proving the strong normalization for the system with user-defined rules, we will show the termination of the system based on the certain version of HORPO. Just like R_n is build on top of R_{n-1}, we will construct a hierarchy of HORPOs $H_0, \ldots$ H_n, one on top of another, and we will show that if every R_n satisfies H_n then the termination of CCR_n follows from the termination of $CC + H_n$. More formally, let H_0, Σ_0 and $>^0_{\mathcal{F}}$ be empty and H_n be a HORPO that uses $CC + H_{n-1}$ for type checking, whose precedence $>^n_{\mathcal{F}}$ includes $>^{n-1}_{\mathcal{F}}$ and which is defined on the signature $\Sigma_n \supseteq \Sigma_{n-1}$. We will show that every $CC + H_n$ is terminating by induction on n.

For $n = 0$, $CC + H_0$ is terminating because the Calculus of Constructions is terminating.

For $n \geqslant 1$, the following schema works:

1. verification of the new signature Σ_n in $CC + H_{n-1}$,
2. construction of H_n,
3. construction of the system $CC + H_n$,
4. proof of strong normalization of $CC + H_n$.

It is obvious how to do the three first steps. Note that all we need is the definition of $CC + H_{n-1}$ and not its decidability.

For the proof of strong normalization of $CC + H_n$, we claim that it is the proof for $CC + H$ where we replace everywhere H by H_n and use $CC + H_{n-1}$ instead of CC for typing. To see this, one has to realize how the proof of strong normalization for $CC + H$ relies on the fact that the signature and HORPO use CC for typing.

The only place where the Calculus of Constructions is explicitly mentioned in the proof of $CC + H$ is the Small Substitution Lemma 5.4.1. This lemma works only for terms that are β-normalizing, in particular for terms of CC. Later on, we use the Small Substitution Lemma in Fun Lemma 5.4.6, and we use it on terms that either come from the signature or result from the Computable Closure (Definition 3.3.2) or the Constructor Subterm Lemma 5.4.4. Fortunately, if the signature and HORPO use CC for typing, then the Computable Closure and the Constructor Subterm Lemma return terms typable in CC.

It turns out that the relationship described above, between terms resulting from the Computable Closure and the Constructor Subterm Lemma and the type system used to check the signature and HORPO, is true for any $CC + H_m$.

Consequently, in the proof of strong normalization of $CC + H_n$, the Small Substitution Lemma is used only on terms of $CC + H_{n-1}$ and it holds for these terms, because $CC + H_{n-1}$ is terminating by induction hypothesis.

Now, let us suppose that for every m, the term rewriting system R_m is accepted by H_m or, if we treat R's and H's like relations, that

$$\forall m\ R_m \subseteq H_m$$

By the definition of H_m we have also:

$$\forall m\ H_{m-1} \subseteq H_m$$

Consequently, $R_1 \cup R_2 \cup \ldots R_m \subseteq H_1 \cup \ldots H_m = H_m$ and the strong normalization of $CC + R_1 + \ldots R_m = CCR_m$ follows from the strong normalization of $CC + H_m$ (the argument is similar to the one used in Lemma 6.0.5).

Formally, the proof of modularity would therefore require to repeat all the definitions and proofs done for $CC + H$, assuming that the signature and HORPO use the system CCX with a certain relation $\succ_X$ included in HORPO and such that $(\rightarrow_X \cup \rightarrow_\beta)$ is terminating.

10 Conclusions

We have defined an extension of the Calculus of Constructions by higher-order rewrite rules of possibly dependent types and presented a criterion (HORPO) which guarantees the strong normalization of the resulting calculus.

The restrictions we impose are twofold:

1. We consider rewriting at the object-level only, and consequently we use only inductive types of star arity.
2. Types of function symbols and constructors of inductive types have to satisfy the star dependency condition.

The first point is a choice: rewriting at the level of types is a difficult question as shown in Werner (1994), Dowek *et al.* (1998) and Dowek & Werner (1999). Very recently, Blanqui (2001) generalized the General Schema to the rewriting on types. Adapting ideas from his paper to our framework would raise some important technical difficulties, since a basic assumption made by Blanqui is confluence which cannot be satisfied by HORPO.

It is not clear how important the second restriction is. At present, it is crucial for the proof of normalization and we believe that it should be difficult to weaken it significantly. Moreover, it seems that star dependency on constructor types is not a severe restriction.

In our paper we have not addressed the problem of logical consistency, but we believe that the strong normalization proof is a step towards it. In fact, in presence of rewriting two consistency problems arise: whether there is an uninhabited type and whether the term structure does not collapse. For both problems some conditions on the types of constants will surely be needed, and for the latter one, probably some confluence properties will be necessary too.

Another interesting problem would be to check if the results presented in this paper extend to the Calculus of Constructions with beta and eta reductions in the conversion.

Acknowledgments

I wish to thank Jean-Pierre Jouannaud for his careful reading and Christine Paulin for fruitful discussions about inductive types and for the example in section 7.1. I am also grateful to the anonymous referees for their useful comments and suggestions.

References

Avenhaus, J. and Loria-Sáenz, C. (1995) *A Reduction Ordering for Higher-order Terms*. Technical report SR-95-03, University of Kaiserslautern.

Barbanera, F., Fernández, M. and Geuvers, H. (1994) Modularity of strong normalization and confluence in the λ-algebraic-cube. *Proc. 9th IEEE Symp. Logic in Computer Science*, pp. 406–415. Paris, France.

Barbanera, F., Fernández, M. and Geuvers, H. (1997) Modularity of strong normalization in the algebraic-λ-cube. *J. Functional Programming*, **7**(6), 613–660.

Barendregt, K. (1990) Functional programming and lambda calculus. In: van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, vol. B, pp. 321–364. North-Holland.

Barendregt, H. (1993) Typed lambda calculi. In: Abramsky *et al*., editors, *Handbook of Logic in Computer Science*. Oxford University Press.

Barras, B., Boutin, S., Cornes, C., Courant, J., Coscoy, Y., Delahaye, D., de Rauglaudre, D., Filliâtre, J. C., Giménez, E., Herbelin, H., Huet, G., Laulhère, H., Loiseleur, P., Muñoz, C., Murthy, C., Parent, C., Paulin, C., Saïbi, A. and Werner, B. (1999) *The Coq Proof Assistant Reference Manual – Version V6.3*.

Blanqui, F., Jouannaud, J.-P. and Okada, M. (1999) The calculus of algebraic constructions. In: Narendran, P. and Rusinowitch, M., editors, *10th Intl. Conf. on Rewriting Techniques and Applications: Lecture Notes in Computer Science 1631*. Springer-Verlag.

Blanqui, F. (2001) Definitions by rewriting in the calculus of constructions. *Proc. 16th IEEE Symp. Logic in Computer Science*. Boston, MA.

Breazu-Tannen, V. and Gallier, J. (1991) Polymorphic rewriting conserves algebraic strong normalization. *Theor. Comput. Sci.* **83**(1), 3–28.

Coquand, T. and Gallier, J. (1990) A proof of strong normalization for the theory of constructions using a Kripke-like interpretation. In: Huet, G. and Plotkin, G., editors, *Preliminary Proceedings 1st Int. Workshop on Logical Frameworks*, pp. 479–497. Antibes, France.

Coquand, T. and Huet, G. (1988) The calculus of constructions. *Infor. & Computation*, **76**(2/3), 95–120.

Coquand, T. and Paulin-Mohring, C. (1990) Inductively defined types. In: Martin-Löf, P. and Mints, G., editors, *Proceedings of Colog'88: Lecture Notes in Computer Science 417*. Springer-Verlag.

Dershowitz, N. (1982) Orderings for term rewriting systems. *Theor. Comput. Sci.* **17**(3), 279–301.

Dershowitz, N. and Jouannaud, J.-P. (1990) Rewrite systems. In: van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, vol. B, pp. 243–309. North-Holland.

Dowek, G., Hardin, T. and Kirchner, C. (1998) *Theorem Proving Modulo*. Rapport de recherche INRIA 3400.

Dowek, G. and Werner, B. (1999) Proof normalization modulo. In: Altenkirch, T., Naraschewski, W. and Rues, B., editors, *Types for Proofs and Programs 98: Lecture Notes in Computer Science 1657*, pp. 62–77. Springer-Verlag.

Girard, J.-Y. (1971) Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la téorie des types. In: Fenstad, J. E., editor, *Proceedings 2nd Scandinavian Logic Symposium*, pp. 63–92. North-Holland.

Jouannaud, J.-P. and Okada, M. (1991) Executable higher-order algebraic specification languages. *Proc. 6th IEEE Symp. Logic in Computer Science*, pp. 350–361. Amsterdam, The Netherlands.

Jouannaud, J.-P. and Okada, M. (1997) Abstract data type systems. *Theor. Comput. Sci.* **173**(2), 349–391.

Jouannaud, J.-P. and Rubio, A. (1998) Rewrite orderings for higher-order terms in η-long β-normal form and the recursive path ordering. *Theor. Comput. Sci.* **208**(1–2), 3–31.

Jouannaud, J.-P. and Rubio, A. (1999) The higher-order recursive path ordering. In: Longo, G., editor, *Fourteenth Annual IEEE Symposium on Logic in Computer Science*. Trento, Italy. IEEE Press.

Jouannaud, J.-P. and Rubio, A. (2001) Higher-order recursive path orderings "à la carte". *First Japanese Conference on Logic and Computation*, Sendai.

Klop, J. W. (1980) *Combinatory reduction systems.* Mathematical Centre Tracts 127, Mathematisch Centrum, Amsterdam.

Klop, J. W., van Oostrom, V. and van Raamsdonk, F. (1993) Combinatory Reduction Systems: introduction and survey. *Theor. Comput. Sci.* **121**(1–2), 279–308.

Loría-Sáenz, C. and Steinbach, J. (1992) Termination of combined (rewrite and λ-calculus) systems. In: *Proc. 3rd Int. Workshop on Conditional Term Rewriting Systems: Lecture Notes in Computer Science 656*, pp. 143–147. Springer-Verlag.

Lysne, O. and Piris, J. (1995) A termination ordering for higher order rewrite systems. In: Hsiang, J., editor, *6th Intl. Conf. on Rewriting Techniques and Applications: Lecture Notes in Computer Science 914*, pp. 26–40. Springer-Verlag.

Nipkow, T. (1991) Higher order critical pairs. In: Kahn, G., editor, *Proc. 6th IEEE Symp. on Logic in Comp. Science*, pp. 342–349. Amsterdam, The Netherlands.

Paulin-Mohring, C. (1993) Inductive definitions in the system COQ. *Typed Lambda Calculi and Applications: Lecture Notes in Computer Science 664*, pp. 328–345. Springer-Verlag.

Stefanova, M. (1999) *Properties of Typing Systems.* PhD thesis, University of Nijmegen.

van de Pol, J. and Schwichtenberg, H. (1995) Strict functionals for termination proofs. *Typed Lambda Calculi and Applications: Lecture Notes in Computer Science 902*, pp. 350–364. Springer-Verlag.

Werner, B. (1994) *Méta-théorie du calcul des constructions inductives.* Thèse de doctorat, Université Paris VII.