Termination of Rewriting in the Calculus of Constructions

Doctoral Thesis

Daria Walukiewicz-Chrząszcz

Institute of Informatics,
Warsaw University,
and
Laboratoire de Recherche en Informatique,
Université de Paris-Sud

Written under the supervision of Professor Jean-Pierre Jouannaud and Professor Jerzy Tiuryn

Warsaw, December 2002

To My Parents

Acknowledgments

Most of all I would like to express my gratitude to my French supervisor Professor Jean-Pierre Jouannaud who introduced me into the research area and provided many valuable ideas. He has been a great source of help and encouragement through all these years. I would also like to thank my Polish supervisor Professor Jerzy Tiuryn for his concern and support. I could always count on his help and advice.

I thank Christine Paulin, Frédéric Blanqui and Jacek Chrząszcz for their insights and many helpful discussions. Particularly, I am in debt to Christine for what I have learned from her about inductive types; to Frédéric for stimulating discussions about necessary termination conditions; and to Jacek for his patience in listening to my detailed explanations of the proof of strong normalization, and for his many good suggestions. Gilles Dowek, Benjamin Werner and Femke van Raamsdonk were interested in my work and gave me many valuable remarks.

The members of research groups: ZLS in Warsaw and Démons in Orsay provided a wonderful and motivating environment for work and study.

I thank the anonymous referees of [90] which is the basis of the substantial part of the thesis.

I gratefully acknowledge the financial support of the French government and Polish KBN grant no. 4 T11C 024 22.

Contents

Chapter 1

Introduction

Term rewriting and typed λ-calculus are two powerful, complementary models of computation. They give a theoretical basis for functional programming languages (ML, Haskell, Scheme), specification languages (OBJ, Maude, Elan) and theorem provers (Coq, Isabelle, Agda).

Typed λ-calculus is a model of functional computations based on a built-in rewrite rule called β-reduction. It has a logical interpretation via the Curry-Howard isomorphism where types are understood as formulas and terms are understood as proofs. Data structures can be directly represented by using the impredicativity of the underlying type system. On the practical side, these representations are both awkward and inefficient.

Term rewriting is a model of algebraic computations, based on user-defined rewrite rules. It has a logical interpretation via Birkhoff's theorem which makes sense when the rewrite rules are confluent. Term rewriting does not directly handle functions, but is well-suited to handle data structures seen as algebraic data types.

Putting these two formalisms together is a natural idea, coming from both the lambda-calculus and the term rewriting communities. The goal of the research is to design calculi that inherit the advantages of both formalisms. Ideally, such a calculus should be able to handle higher order objects and allow easy manipulation of data types. The fundamental properties required from such a calculus are subject reduction, confluence and strong normalization.

Gödel system T may be considered as one of the first examples of a combination of a typed λ-calculus with rewriting. It unifies simply typed λ-calculus and the rules for recursors on natural numbers. Gödel's T had many followers: Martin-Löf's type theory [65, 66], Mendler's system of recursive types [69, 68] and Coquand and Paulin's calculus of inductive constructions [32]. In all these typed λ-calculi the elimination schemes of recursively/inductively defined types are expressed as rewrite rules.

The study of simply-typed λ-calculus augmented with arbitrary first-order rewriting was initiated by Breazu-Tannen in the late 80's [22] and it turned out to be very satisfactory from both confluence and strong normalization points of view.

Incorporating higher-order rewriting is much more challenging and still not completely understood. The systematic study of higher-order rewriting was started by Klop and his combinatory reduction systems [57]. It was followed by Nipkow who developed significantly the theory of confluence [72] and by Jouannaud and Okada who proposed the termination criterion called the general recursive schema [50]. This criterion, originally designed for the simply typed λ-calculus was later lifted by other authors to the whole λ-cube [6] and pure type systems [12]. Later Jouannaud and Rubio extended the recursive path ordering — used for proving termination of first-order rewriting — to higher-order simply typed terms [53]. This method turned out to be strictly more powerful than the general schema.

The subject of this thesis is to extend the work of Jouannaud and Rubio to the calculus of constructions. We present a dependent version of the higher-order recursive path ordering (HORPO) and prove the strong normalization for the calculus of constructions augmented with rewrite rules accepted by HORPO. A practical motivation of this work is the perspective of extending the Coq proof assistant [87], based on the calculus of constructions, with a rewriting mechanism. In particular, this would allow to smoothly incorporate decision procedures within the proof development process. On the theoretical side, the contribution of this thesis can be seen as the design of a well-founded ordering on higher-order terms that can be used to show termination of higher-order calculi without explicitly using the complex Tait and Girard's method of reducibility candidates.

In the following we give a more detailed presentation of the notions and results mentioned above. In particular we discuss successive extensions of the calculus of constructions increasing its expressive power. Our work can be seen as a one more step in this succession.

1.1 Term Rewriting

Term rewriting originated from the need of efficient equality handling in the first-order deduction systems [84, 59, 61]. Later on, it was successfully used as a programming paradigm [73, 36, 37, 42, 28, 21]. In this section we will give a brief introduction to term rewriting; for a detailed exposition the reader is referred to [38, 58, 3].

1.1.1 First-order rewriting

Rewriting is an abstract model of computing with algebras, that is, a set of values equipped with a set of operations whose properties are axiomatized by equations. These equations can be used for reasoning, by applying Leibniz's law for replacing equals by equals, in a highly nondeterministic way. Using equations only in one direction leads to the notion of *rewrite rules* and yields a computational mechanism for the expressions of the abstract language. A set of rewrite rules is a *rewrite* system.

The process of replacing left-hand sides of rewrite rules by the corresponding right-hand sides may continue until an expression that cannot be reduced any further is possibly obtained. This process is called *normalization* and its result is a *normal form*. When every reduction sequence eventually ends up in a normal form, the term rewriting system is said to be *terminating*, or *strongly normalizing*. If in addition all reductions from a given term lead to the same normal form, the system is said to be *convergent*. The property underlying uniqueness of the result is called *confluence*: it ensures that every two divergent reduction sequences of common origin can be joined.

For example, groups can be defined as algebras over the signature $\{e, {}^{-1}, \cdot\}$ with the following set of equations:

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad x \cdot x^{-1} = e \qquad x \cdot e = x \tag{1.1}$$

Orienting the equations from left to right yields a terminating, non-confluent rewrite system, since $(x \cdot e) \cdot x^{-1}$ rewrites both to e and $x \cdot (e \cdot x^{-1})$ which are different normal forms. There is however a method, called Knuth-Bendix completion, which allows to compute a convergent set of rules from the oriented equations while preserving the original equational theory. This is important, because the obtained rewrite system can then be used to decide whether two terms are equal by simply comparing their normal forms. Of course not all decidable equational theories can be decided by mere rewriting, and therefore Knuth-Bendix completion does not succeed in general.

1.1.2 Termination proof methods

Termination turns out to be a fundamental property of rewrite systems, since it is instrumental in both the existence of a normal form, and in its uniqueness: many algorithms for checking confluence rely on termination. Termination is of course undecidable, but there are sophisticated proof methods, among which we distinguish semantical, transformational and syntactical methods.

Semantical methods consist in finding an appropriate interpretation in a well-founded domain, such that every reduction step decreases the interpretation. The practical difficulty here is in the choice of the well-founded domain and the interpretation. A special case is given by polynomial interpretations: the domain is then the set of polynomials over some interval of natural numbers compared by the natural extension to polynomials of the ordering on natural numbers. The left to right orientation of the group equations (1.1) can easily be treated that way.

Transformational methods are simply non-termination preserving transformations of term rewriting systems. Given such a transformation Φ, one can deduce the termination of a rewrite system R by proving by any suitable method the termination of $\Phi(R)$.

Syntactical methods are based upon the direct use of well-founded orderings on terms. Practice favors the use of reduction orderings, which are closed under substitution instance and context application: it then suffices to check that the left-hand side of each rule is greater than the corresponding right-hand side in the ordering.

An important example of syntactical method is the *recursive path ordering* (thereafter called RPO) due to Dershowitz [35]. It is an ordering on terms built upon a given well-founded ordering on function symbols, called the *precedence*, and a *status* for each function symbol, either lexicographic or multiset. We give and explain its definition below, since it is the basis of the technique presented in this thesis.

Definition 1.1.1 *Let $>_{\mathcal{F}}$ be the precedence on the signature $\mathcal{F}$, and let every symbol have status Lex or Mul. The recursive path ordering is defined as follows:*

$$l = f(l_1, \dots l_m) \succeq_{rpo} g(r_1, \dots r_n) = r$$

if and only if either of the following holds

1. $l_i \succeq_{rpo} r$ *for some* $i \in [1..m]$,
2. $f >_{\mathcal{F}} g$ *and* $l \succ_{rpo} r_i$ *for all* $i \in [1..n]$,
3. $f = g$ *and*
   (a) $f \in Mul$ *and* $(l_1, \dots l_m)(\succeq_{rpo})_{mul}(r_1, \dots r_n)$

(b) $f \in Lex$, $(l_1, \dots l_m)(\succeq_{rpo})_{lex}(r_1, \dots r_n)$ *and* $l \succ_{rpo} r_i$ *for all* $i \in [1..n]$,

where $s \succ_{rpo} t$ *if* $s \succeq_{rpo} t$ *and* $t \not\succeq_{rpo} s$.

The main idea underlying RPO is that two terms l and r are compared by first looking at their root symbols, and then recursively comparing the collections of their immediate subterms. If the root of l is smaller, then the only way l can dominate r is by having a subterm itself dominating r. If the root of r is smaller, then l must dominate each of the subterms of r. When the roots are equal, the collections of their immediate subterms are recursively compared, according to the status of the root. These collections can be seen as multisets, leading to a multiset comparison, or as tuples, leading to a lexicographic comparison.

Oriented equations defining groups (1.1) can be compared by letting $\cdot >_{\mathcal{F}} e$ and $\cdot \in Lex$.

Implementing RPO is easy, and the comparison of two terms (given precedence and status) can be done in time $\mathcal{O}(n^2)$. Another advantage is that it is rather easy to guess the adequate precedence and statuses in practice, although it is NP-complete in the number of function symbols.

It turns out that semantical methods can be incorporated to RPO by comparing the whole terms (instead of their roots) in some additional well-founded (semantical) ordering.

1.1.3 Higher-order rewriting

In the framework of first-order rewriting, bound variables do not exist as such. However, bound variables appear naturally in logic, mathematics and computer science. We need them to express the β-reduction rule $(\lambda x.u)v \rightarrow u[v/x]$, tautologies for the predicate logic like $\forall x(P(x) \wedge Q(x)) \Leftrightarrow (\forall x P(x)) \wedge (\forall x Q(x))$, or to define the operational semantics of programming languages: for example, the rule $o.m_i \rightarrow B_i[o/\mathit{self}]$ expresses the method invocation in the object-oriented programming style.

Higher-order features are also needed when functions can take other functions as arguments or return them as values. Let us consider the definition of *map* which applies a function given as a first argument to all elements of a list:

$$\begin{array}{rcl} map(f, nil) & \rightarrow & nil \\ map(f, cons(a, l)) & \rightarrow & cons(f(a), map(f, l)) \end{array}$$

The symbol f above is used both as a variable and as a function symbol. Hence it is a higher-order variable.

There are two ways of dealing with bound and higher-order variables. In the first approach higher-order concepts are encoded in a first-order framework, which has the advantage of possibly reusing tools and methods designed for the first-order case. The examples of such translations are, among others, Combinatory Logic [33] and explicit substitutions [1, 62, 55, 56, 20]. In the second approach, higher-order variables and binders are explicitly introduced, giving rise to higher-order rewriting. The most prominent example is rewriting on λ-terms. We distinguish here three main variants.

First, there are combinatory reduction systems (CRS) — an untyped approach introduced by Klop in his thesis [57]. Two other approaches, higher order rewriting systems (HRS) and algebraic-functional systems (AFS), are typed and they differ in the use of the λ-calculus conversions (α, β and η).

Higher-order rewriting systems

The notion of higher-order rewriting systems was introduced by Nipkow in [72]. In this approach, rewriting operates on equivalence classes of terms modulo the conversions, which is done by using terms in η-long β-normal form. Besides, to ensure that higher-order pattern matching is decidable (which is still an open problem in the general case), the left-hand sides of rules must be *patterns*, actually making the higher-order matching linear. The rules in this approach are pairs of terms of the same basic type.

Many results concerning confluence have been generalized from the first-order setting to the higher-order case [67, 89, 82, 75].

Compared to the first-order case, the theory of termination is significantly less developed. Two important methods have been investigated: termination models [80, 81] and higher-order versions of the recursive path ordering [52, 53, 83].

In his thesis [81] van de Pol proves that a HRS is terminating if it has a termination model. A termination model is given by the interpretation of base types as well-founded partial orders with some additional property that guarantees the existence of so called weakly monotonic and strict functionals. These functionals are consequently used as interpretations for function symbols and rules. The difficulty in using this approach often lies not in finding suitable interpretations but in proving their properties.

Two initial attempts to generalize the recursive path ordering to the higher-order case by Loria-Saenz [63] and by Jouannaud and Rubio [52] did not lead to convincing practical methods. The successful approach came later, as a by-product of an attempt to define a higher-order recursive path ordering for higher-order rewriting [53]. This is explained in the next paragraph about algebraic-functional systems. Another adaptation of [53] to higher-order rewriting (and Klop's combinatory reduction systems) is presented in [83].

Algebraic-functional systems

In this approach, the user-defined rewrite rules and the conversion rules of the λ-calculus are combined, i.e., β reductions are performed explicitly and not as a part of rewriting steps: plain (syntactic) matching is used here to fire a rule.

Breazu-Tannen initiated the study of plain higher-order rewriting by investigating the confluence properties of the simply-typed λ-calculus augmented by first-order rewrite rules [22]. The investigation of termination properties followed closely. The works by Breazu-Tannen and Gallier [23], Okada [74], Dougherty [39] showed that first-order rewriting can be safely added to polymorphic λ-calculus.

The addition of higher-order rewrite rules to the polymorphic λ-calculus was then considered by Jouannaud and Okada [50, 51]. Using Tait and Girard's computability predicate method, the authors show the termination of the rewrite relation generated by the union of the β-rule and the user-defined rewrite rules. Rewrite rules are assumed to follow the so-called general recursive schema, a generalization of primitive recursion of higher type. These results were later lifted to more powerful type systems, including the calculus of constructions [6, 7]. Finally, the general schema was reformulated [18, 19], which was instrumental in extending these concepts further to dependent type calculi [16, 17].

The general schema is not explicitly based on term orderings. An extension of the recursive path ordering to simply typed higher-order terms (Higher Order Recursive Path Ordering) was proposed by Jouannaud and Rubio [53]. Due to the recursive structure of HORPO, this method is strictly more powerful than the general schema (in the simply typed case), although both methods bear some similarities. In their work, terms of different types could not be compared, a weakness that is overcome in a more recent work [54]. The true goal of the present thesis is to generalize this ordering to the calculus of constructions framework.

1.2 The typed λ-calculus

The lambda calculus was introduced by Church as a foundation for mathematics [25, 26]. Church's aim was to design a logical system based on the notion of function and function application. In this system, the term $\lambda x.u$ denotes a function of (formal) parameter x and body u. Application of a term v to a term w is written $v\,w$. In case v is a function $\lambda x.u$, (one step of) the function evaluation is modeled by the β-rule, $(\lambda x.u)w = u[w/x]$, according to which the (intermediate) result of the computation is obtained by replacing the variable x by the argument w in the body u.

Because λ-calculus is indeed a theory of partial recursive functions, its use as a logical system yields paradoxes. To restore logical consistency, Church proposed a version of λ-calculus in which functions are stratified according to the types of their arguments and result, application being then restricted to arguments of appropriate type [27].

The resulting calculus, called simply typed λ-calculus, has many nice properties. First, every term has a unique normal form, because β-reduction on typed terms is terminating and confluent. This makes the theory decidable, since showing equality of terms reduces to comparing their normal forms. Second, it turns out that the simply typed λ-calculus reveals intriguing analogies with the implicational fragment of natural deduction: a proof of a proposition ϕ can be seen as a lambda term of type ϕ, introduction of implication becomes a λ-abstraction, and elimination of implication is application. In other words, the lambda-calculus can be seen as an abstract data structure for the proofs in this logical fragment. Moreover, normal proofs, for which an introduction is never immediately followed by an elimination, correspond to lambda terms in normal forms, and the cut elimination theorem for natural deduction corresponds to the normalization property of the typed λ-calculus.

The above phenomenon, whose various aspects have been discovered by Curry and Feys [33], Howard [48] and de Bruijn [34], is called the Curry-Howard isomorphism. Its importance lies in its use during the past decades as a way to cross-fertilize logical and computational calculi. For example, type systems corresponding to the second-order and higher-order propositional logics are respectively Girard's system F of polymorphic types and its extension F_ω [47]. Encoding predicate logics requires the introduction of dependent types, that is, types that can depend upon terms.

The λ-cube [14, 88, 8] is a collection of eight type systems for the lambda calculus, which include those mentioned above. Every direction in the three-dimensional space corresponds to one of the possible dependencies: terms on types (polymorphism), types on terms (dependent types) and types on types (type constructors). The system combining all three possibilities, therefore lying at the top of the cube, is Coquand and Huet's calculus of constructions [31].

All these systems enjoy the main properties of the simply typed λ-calculus and are therefore decidable theories of total functions enjoying strong normalization. Of course, more complex type systems give additional expressivity. From the extensional point of view, the simply typed λ-calculus can only represent polynomial functions (with zero-test). An important expressivity gain is achieved in system F (and then in F_ω) in which all recursive functions provably total in second-order (respectively higher-order) arithmetic can be represented. No more functions can be obtained by adding dependent types. However, expressivity can also be discussed from the intentional point of view. In [60] Krivine presents an algorithm for the minimum of two numbers that cannot be expressed in system F. Of course, the untyped lambda term, representing the algorithm, is typable in system F but not with the desired type $Nat \rightarrow Nat \rightarrow Nat$.

Another reason for the quest for more expressive type systems comes from the use of the previously mentioned typed λ-calculi as a basis for proof assistants in the corresponding logics. In these logics, finding a proof of a given proposition (finding an inhabitant of the type) is undecidable, but checking a proof (type-checking) is decidable. Proof assistants come therefore in two parts: a proof checking algorithm called the kernel of the system, and a proof search component, made of general and domain-dependent heuristics called tactics.

In dependent type calculi, two types may differ by β-convertible subterms. For example, if IsEven is a predicate, then $\mathit{IsEven}(2+2)$ and $\mathit{IsEven}(4)$ are convertible to each other, assuming a suitable definition for $+$. This phenomenon is actually accounted for by a specific typing rule of these calculi called conversion. This rule states that if a term t has type T in an environment Γ and if T is β-convertible to T', then t has also type T' in the same environment:

$$\frac{\Gamma \vdash t : T \quad T =_\beta T' \quad T' \text{ is a type}}{\Gamma \vdash t : T'}$$

Note that the use of conversion is not reflected by the proof t, which remains unchanged. Conversion can therefore be seen as a mean to separate reasoning from computing: while the proof term must record all deduction steps, the conversion steps can be hidden away since β-conversion can be performed automatically by checking the normal forms of T and T'. Extending conversion is therefore a key towards having more automatic proof assistants, making them more user-friendly.

For more detailed accounts on typed λ-calculus and its use in automated reasoning the reader is referred to [9, 10].

1.3 The calculus of constructions and its extensions

There are several extensions of the pure calculus of constructions which follow the lines presented above. Of course, the more complex the system is, the closer we are to undecidable type-checking, or even to a paradox. For example, extending the conversion with the extensionality axiom (saying that functions are equal iff they are equal as input-output relations) results in a consistent system with undecidable type-checking. The assumption that $\star$ is a sort of all types, including itself, results in an inconsistent system.

Decidability of type-checking and the absence of paradoxes usually follow from other more basic properties: subject reduction (called also type preservation), confluence and strong normalization.

Before discussing four extensions of the calculus of constructions, we briefly introduce the basics of its syntax by means of examples and discuss its limitations.

1.3.1 The pure calculus of constructions

The set of terms of the pure calculus of constructions is built from two constants, $\star$ and $\Box$, variables and three operations, application, abstraction and product; these are written as $u\,w$, $\lambda x\!:\!u.w$ and $(x\!:\!u).w$ respectively. If w does not depend on x then the term $(x\!:\!u).w$ can be written as $u \to w$.

Construction of terms is governed by typing rules that operate on judgments of the form $\Gamma \vdash t : T$, where Γ is an environment and t, T are terms. According to these rules $\star$ is meant to be a sort of types and propositions, and $\Box$ a sort of $\star$ and other kinds like $\star \to \star$ or $(A \to \star) \to \star$.

In the pure calculus of constructions one can derive

$$\vdash (X\!:\!\star).X \ : \star$$

which means that $(X\!:\!\star).X$ is a type. It is called an empty type, since there is no term t such that $\vdash t : (X\!:\!\star).X$.

One can also derive

$$A : \star,\ P : A \to \star,\ a : A \vdash P\,a \ : \star$$

which states that given a set A and a predicate P, $P\,a$ is a proposition (true if inhabited and false otherwise).

The precise definition of the pure calculus of constructions and its properties are given in Section 2.1.

The pure calculus of constructions captures both the notion of computation (via the inclusion of functional programs written in typed λ calculus) and of proof (via the Curry-Howard isomorphism). Being such a powerful system, it was used as the basis of CoC, the preliminary version of the proof assistant Coq.

Constructing proofs interactively in CoC showed the limits of the calculus especially when the representation of data is concerned. The natural way to represent data is by defining inductive types, which come in three parts: a complete list of constructors for terms of that type, an appropriate induction principle, and a mechanism for writing functions by some form of primitive recursion.

In the pure calculus of constructions it is possible to define natural numbers, lists, booleans and other inductive types by using the so-called impredicative encoding. There is indeed a systematic procedure that, given a set of typed constructors for some inductive type produces a term of the pure calculus of constructions representing this inductive type [79]. For example, natural numbers can be represented as follows:

$$\begin{aligned} Nat &\equiv (C:\star)C \to (C \to C) \to C \\ 0 &\equiv \lambda C{:}\star.\lambda x{:}C.\lambda f{:}C \to C.x \\ s &\equiv \lambda n{:}Nat.\lambda C{:}\star.\lambda x{:}C.\lambda f{:}C \to C.n\,C\,(f\,x)\,f \end{aligned}$$

It turns out that it is possible to define all primitive recursive functions on Nat. Nevertheless this coding has some important drawbacks concerning both logical and computational aspects. Let us list them briefly:

- One cannot prove that 0 is different from 1.
- Induction principles (for example on natural numbers) are not provable.
- The predecessor function cannot be computed in constant time.
- For some inductive types, there are more closed terms in normal form inhabiting the coding than terms built only from constructors.

1.3.2 The calculus of inductive constructions

The above mentioned problems were considered by Coquand and Paulin: in order to represent inductive types they proposed to add to the calculus new kinds of syntactic objects: inductive definitions introducing a type and its constructors and elimination schemes for that type [32].

For example, the inductive definition of the type of natural numbers becomes

$$Ind(Nat : \star := 0 : Nat,\ s : Nat \to Nat)$$

meaning that Nat is an inductive type with two constructors 0 of type Nat and s of type $Nat \to Nat$. The types of constructors are subject to the strict positivity condition: for every constructor of an inductive type I types of its arguments must be of the form $\overrightarrow{(x : b)}I$, where I does not appear in $\vec{b}$.

The terms corresponding to the elimination schemes are more complicated as they encompass different variants of elimination (dependent, non-dependent, weak, strong) at the same time. To present their power and computational behavior it is more convenient to use the format of recursors. For natural numbers one can use

$$NatElim_{dep} : (P : Nat \to \star)(P\ 0) \to ((m : Nat)(P\ m) \to (P\ (s\ m))) \to \\ (n : Nat)(P\ n)$$

representing the well-known induction schema on natural numbers,

$$NatElim_{nondep} : (P : \star)P \to (Nat \to P \to P) \to Nat \to P$$

representing the recursor on natural numbers, just like in Gödel system T and

$$NatElim_{Tnondep} : (P : \Box)P \to (Nat \to P \to P) \to Nat \to P$$

representing the possibility of creating types (more precisely objects of type P) by induction on natural numbers. The elimination of such kind is called strong elimination.[1]

For all these eliminators the associated reduction rules are the same:

$$\begin{aligned} NatElim\ f_0\ f_1\ 0 &\to f_0 \\ NatElim\ f_0\ f_1\ (s\ n) &\to f_1\ n\ (NatElim\ f_0\ f_1\ n) \end{aligned} \tag{1.2}$$

They are called ι-reduction rules and become part of the conversion rule of the system.

[1] Formally the type of $NatElim_{Tnondep}$ is not a correct term of the calculus of constructions. One solution is to consider a collection of recursors, another one is to introduce a new sort as the type of $\Box$.

Using strong elimination one can show that 0 is different from 1, for example by defining a predicate $P : Nat \to \star$, such that $(P\ 0)$ is convertible to $True$ (an inductive type with one constructor $i : True$) and $(P\ (s\ 0))$ to $False$ (inductive type without constructors). Strong elimination is an important modification of the underlying calculus and to keep the system consistent it is allowed only for inductive types whose constructors do not take types or predicates as arguments [76, 77].

The calculus of inductive constructions preserves all essential meta-theoretical properties of the original system: it is terminating, confluent, logically consistent and has decidable type-checking [92].

1.3.3 The calculus of constructions with definitions by pattern matching

In the calculus of inductive constructions all functions have to be defined by means of elimination schemes. While being a significant improvement over impredicative encodings used in the pure calculus of constructions, this approach has shown some drawbacks.

The first one is the expected computational behavior. For example, $plus$ can be defined as

$$plus(x, y) = NatElim_{nondep}(Nat, y, \lambda n\!:\!Nat.\lambda m\!:\!Nat.s(m), x)$$

Now, the term $plus(s(x), y)$ reduces to $s(NatElim_{nondep}(Nat, y, \lambda n\!:\!Nat.\lambda m\!:\!Nat.s(m), x))$ and one needs to fold back this expression to get the expected $s(plus(x, y))$.

Another one is readability. The definition of $plus$ above is rather complicated. Sometimes, it is not even clear how to use $NatElim_{nondep}$ in order to define certain functions which can be simply characterized by pattern matching expressions. A typical example is a boolean function that checks whether a natural number is even or not:

$$\begin{array}{rcl} even(0) & = & true \\ even(s(0)) & = & false \\ even(s(s(x))) & = & even(x) \end{array}$$

A natural solution to these problems is to replace the elimination schemes by two separate mechanisms: one for pattern-matching (or case-analysis), and one for constructing well-founded recursive functions.

Definitions by pattern-matching were added to the Martin-Löf's type theory by Coquand [29]. Consequently, constants may be defined not only explicitly, by giving a term, but also implicitly by their defining equations of the form $f(u_1, \dots u_n) = e$. These equations are actually computation rules and they must satisfy two requirements. The first one is that rules are exhaustive and mutually exclusive, a property whose verification needs in general higher-order unification. The second one concerns strong normalization. As a way to deal with this requirement Coquand suggested to use the structural ordering induced by the inductive types definitions. More precisely he proposed that for every function symbol f there is an index $i \leq n$ such that in every recursive call $f(v_1, \dots v_n)$ in e the term v_i is structurally smaller than u_i. Being structurally smaller is defined as a transitive closure of the relation: $a_i \vec{t}$ is smaller than $c(a_1, \dots a_m)$, where c is a constructor.

At the same time Coquand noticed that the approach needs to be refined if an impredicative type theory is concerned, because there are functions, satisfying the above condition, that do not terminate. Consider for example a type V with one constructor v and a function f defined by

$$Ind(V : \star := v : ((A : \star)(A \to A)) \to V)$$

$$f(v(h)) \to f(h\ V\ v(h))$$

Then the term $f(v(\lambda A{:}\star.\lambda x{:}A.x))$ is non-terminating.

An adaptation of the above idea to the context of the calculus of constructions is due to Christine Paulin and was implemented in the proof assistant Coq in 1994. The problem of checking ambiguity and exhaustivity is eliminated by choosing a simple format of case analysis using $Cases$ operator. The recursive definitions are built using the fixpoint operator Fix. Normalization is guaranteed within the type-checking rule of the Fix. According to it, every recursive call must operate on arguments that are structurally smaller at recursive positions (the positions in a constructor of the inductive type I is recursive if its type contains I). Such definitions are called *guarded-by-destructors*. The problematic type V above is now accepted, but the function f is not (as the first position in v is not recursive). The relationship between the calculus with fixpoints and the one using elimination schemes is presented in [43].

Although very natural and general enough to accept many interesting functions, the Fix rule with its guard condition causes problems, both at the practical and the theoretical levels. First, it complicates the meta-theoretical investigation of the system, and indeed, no complete proof of strong normalization exists at the moment for arbitrary inductive types. Second, from the perspective of a proof assistant, since the side condition in the rule for Fix applies to the whole proof-term, proof checking cannot be done incrementally anymore. For these reasons several versions of type annotations were proposed to hide the guardedness condition in the type system [44, 78, 45], but none of them is completely satisfactory. Some more complete results exist for the simply-typed λ-calculus [11, 2].

1.3.4 The algebraic λ-cube

The algebraic λ-cube arises as a combination of the traditional λ-cube with a mechanism to define algebraic functions by means of rewrite rules.

Originally algebraic-functional combinations were seen as programming languages and built upon non-dependent calculi such as the simply typed λ-calculus, the second-order λ-calculus, or the higher-order λ-calculus [4, 5, 22, 23, 24, 50]. Algebraic type systems with dependent types were first introduced for the systems of the algebraic λ-cube in [6, 7] and later in their general form in [12], where the underlying calculus is any single-sorted pure type system.

Rewriting in the algebraic λ-cube is built upon rules whose both sides are algebraic terms, i.e. constructed from function symbols, variables and the application symbol, of the same type. Moreover, all function symbols are restricted to have types built from the set of basic algebraic types using $\rightarrow$. Of course this kind of rewriting may be not only first-order, but also higher-order.

Rewriting in the system makes it possible to define functions directly by their computation rules. For example *plus* can be defined as

$$\begin{array}{rcl} plus(0,x) & \rightarrow & x \\ plus(s(y),x) & \rightarrow & s(plus(y,x)) \end{array}$$

Moreover one can easily add other rules for plus like

$$plus(x,0) \rightarrow x \quad \text{or} \quad plus(plus(x,y),z) \rightarrow plus(x,plus(y,z))$$

When a dependent calculus is considered, rewriting is not only used to easily define functions but also to enrich the conversion of the system. For *plus*, this means that both $plus(x,0)$ and $plus(0,x)$ are identified (convertible) with x, which is not possible in the pure calculus of constructions nor in its extension with definitions by pattern matching. Consequently, more terms are typable (like $(\lambda x : P\,(plus(n,0)).\,x)h$ in an environment where $P : Nat \rightarrow \star$ and $h : P\,0$) and more computations are done automatically in the conversion rule.

Strong normalization for the algebraic λ-cube was first shown in [6, 7] under the conditions adopted from [50]. According to them, first-order rules must be non-duplicating and strongly normalizing on first-order algebraic terms and higher-order rules must satisfy the general schema. In [6, 7] there are also proofs of subject reduction and confluence.

The results presented in [7] were further extended in [18] by reformulating the general schema and adding a new powerful mechanism, called the computable closure.

One can consider algebraic rewriting not only over a λ-cube but over an arbitrary pure type system. The resulting algebraic type systems were studied by many authors. Strong normalization of algebraic type systems is known for rules satisfying the general schema of Jouannaud and Okada [12] and for the systems where separately the underlying pure type system and the term rewriting system are strongly normalizing [13].

1.3.5 The calculus of constructions with dependent rewriting

A natural continuation of the previously mentioned approaches is to consider rewriting without restriction to algebraic terms and to deal with function symbols whose types may be dependent.

Using dependently typed constants one may define the inductive type of lists of natural numbers of a given length *Listl* together with its constructors *nill* and *consl*:

$$\begin{array}{rcl} Listl & : & Nat \to \star \\ nill & : & Listl\, 0 \\ consl & : & (n : Nat)Nat \to (Listl\, n) \to (Listl\, s(n)) \end{array}$$

The *map* function on such lists is defined as follows:

$$\begin{array}{rcl} map(f, 0, nill) & \to & nill \\ map(f, s(n), consl(n, a, l)) & \to & consl(n, f(a), map(f, n, l)) \end{array}$$

map being a symbol of type $(Nat \to Nat) \to (n : Nat)(Listl\, n) \to (Listl\, n)$.

One can also imagine functions that operate on proofs, for example one that given an element a and a list l returns a proof that a appears in l or the proof that all elements of l are different from a (see Example 13 in Section 7.2).

Elimination schemes of the calculus of inductive constructions or guarded-by-destructors definitions are other examples of rewrite rules built upon dependently typed symbols.

Rules (1.2) for $NatElim_{dep}$, $NatElim_{nondep}$ (cf. page 18), or the rules for *map* represent *rewriting on objects*, since they concern function symbols which applied to their arguments are objects (terms of type T with $T : \star$). The same rules (1.2) for $NatElim_{Tnondep}$, corresponding to strong elimination, represent *rewriting on types*, since $NatElim_{Tnondep}$ applied to its argument is a type (term of type $T : \Box$).

Dependent rewriting in the calculus of constructions (without restriction to elimination schemes or guarded definitions) was considered in [91] and its journal version [90]. In these papers, an extension of HORPO from [53] to the calculus of constructions was presented and the strong normalization for the calculus of constructions with rewriting induced by HORPO was proved. The rewriting considered in these works was rewriting on objects. This thesis is an elaboration and extension of those papers.

Later, Blanqui studied dependent rewriting in [16, 17]. He extended the general schema from [18] and considered rewriting not only on objects but also on types. A detailed comparison with [17] is presented in Chapter 8.

1.4 Contributions

In this thesis we show how to incorporate dependent, higher-order, object-level rewriting into the calculus of constructions and we present a method for proving strong normalization of the combined calculus.

Incorporating dependent rewriting into the calculus of constructions is not straightforward and requires some explanations. The main problem is that the definitions of rewriting and the typing system are mutually dependent. On the one hand, rewriting is used in the conversion rule, hence the typing depends on rewriting. On the other hand, rewriting should operate on well typed terms only; its definition must satisfy some typing condition.

These constrains have motivated the author to introduce dependent rewriting to the calculus of constructions in two steps. First, we define the notion of signature, introducing correctly typed function symbols and inductive types, and HORPO — the acceptance criterion for rewrite rules. Both these definitions use the type-checking of the calculus of constructions with just β-equality in the conversion rule. Next, we simultaneously define the rewrite relation and CC+H — the calculus of constructions with rewriting accepted by HORPO. This two-step approach helps handle the circularity mentioned in the previous paragraph.

An interesting aspect of the above presentation is that CC+H and the proof of strong normalization are done not for a particular rewrite system accepted by HORPO, but for the whole HORPO itself. So, 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. This implies strong normalization of any set of rewrite rules accepted by HORPO

The status of inductive types in our work is similar to the one in the calculus of constructions with definitions by pattern-matching. Inductive types are built from type constructors and constructors satisfying the strict positivity condition and elimination schemes are just special cases of rewrite rules. Accepting in HORPO the corresponding elimination schemes can be done for an important subsystem of the calculus of inductive constructions.

An important technical contribution of this work concerns the form of the rewrite rules. The left-hand side of a rewrite rule starts with a function symbol, which is applied to some parameters and other arguments. The division into parameters and the rest of the arguments is correct if it satisfies the *star dependency* condition, which roughly says that all type arguments are parameters. We consider only *parametric rules*, which means that each parameter is a different variable. The notions of star dependency and parametricity together with their interactions with HORPO are two major concepts of our work. In Section 6.1 there is an example of an innocently looking rule (for Girard's operator J) which does not satisfy the parametricity condition and makes the system non-terminating.

The definition of HORPO presented in our thesis extends to the calculus of constructions the definition introduced by Jouannaud and Rubio for the simply-typed λ-calculus [53]. Its main part resembles very much the original recursive path ordering of Dershowitz, but there are also some additional mechanisms dealing with inductive objects and taking advantage of the fact that the computability predicate method is used for proving termination.

According to the standard vision of constructors as symbols that do not rewrite, the left-hand sides of HORPO judgments cannot be constructor-headed terms. In order to incorporate into HORPO the structural ordering generated by inductive definitions, the *constructor extension* and *recursive constructor extension* of HORPO are introduced. They are based on the notions of the structurally smaller term of Coquand and recursive positions of Paulin, which are discussed in Section 1.3.3.

The computable closure is another mechanism used in HORPO. It turned out to be very useful already in the context of the simply-typed λ-calculus [19]. Its aim is to build as many terms as possible, whose computability follows from the computability of the given initial set of terms.

The strong normalization of the calculus of constructions with HORPO is proved using a typed version of Girard's reducibility candidates method proposed by Coquand and Gallier [30]. The system has the subject reduction property, but its proof is complex since the rewrite relation generated by the β-rule and HORPO is obviously not confluent. The proof of subject reduction is adopted, with very little changes, from [7].

Like the original recursive path ordering, HORPO with its constructor extensions and computable closure is a decidable criterion for accepting rewrite rules. Putting aside the convertibility tests (i.e. β-equality), checking a rewrite rule has a polynomial complexity. Decidability and relatively low complexity of HORPO provide an important argument for considering CC+H as a basis for the incorporation of rewriting in interactive theorem provers such as Coq [87].

1.5 Road map

In Chapter 2 we describe the setting of our work: we recall the definition of the pure calculus of constructions, introduce the notion of the signature and present all preliminary conditions on constants, like star dependency or strict positivity.

Chapter 3 is devoted to HORPO and the notions it uses: the parametricity condition, the computable closure and the constructor extensions of HORPO. At the end of this chapter the calculus CC+H is formulated, which is the calculus of constructions with rewriting induced by HORPO.

Chapter 4 studies the basic meta-theory of CC+H. An important part of this chapter is concerned with the proof of subject reduction.

Chapter 5 proves the strong normalization theorem for CC+H, using a typed version of Girard's reducibility candidates method.

Chapter 6 provides a discussion on the role of the star dependency condition used both for function and constructor symbols and its use in the proof of strong normalization.

Chapter 7 shows how to pass from CC+H to CC+R — the calculus of constructions with rewriting induced by user-defined rewrite rules satisfying HORPO. The definition of CC+R is followed by many examples that do or do not fit in our framework. In the last section of this chapter we show which elimination rules for inductive types are accepted in HORPO.

Chapter 8 presents a detailed comparison with [17], where the general schema is used as the method for proving termination of higher-order rewriting in the calculus of constructions.

Chapter 9 considers some practical issues of the proposed system. We show the decidability of HORPO (together with its constructor extensions and computable closure) and briefly discuss the modularity of building the system CC+H and proving its termination.

In Chapter 10, we discuss further research topics and directions.

Chapter 2

Setting

The calculus presented in this paper is built from a predefined set of function symbols equipped with typing information. This chapter will explain these prerequisites and introduce the basic notions used in the rest of the thesis.

The section is organized as follows. In Section 2.1 we recall the definition of the standard calculus of constructions. In Section 2.2 we introduce the notion of signature assigning arities and types to constants. We continue with the conditions imposed on these types: star dependency in Section 2.3.1, strict positivity in Section 2.3.2 and typing conditions in Section 2.3.3. Finally, we present a version of the calculus of constructions with constants from a given signature.

The last part of this section (2.5.3) introduces basic definitions concerning relations and orderings.

2.1 The pure calculus of constructions

The calculus of constructions, originally defined in [31], is the starting point of our work. We will refer to it as the *pure calculus of constructions* in order to distinguish the original calculus from its later version where some additional constants are added. This version with constants, given in Section 2.4, will be called the *calculus of constructions*.

All notations given in this section are mostly standard. All other notations used in this thesis are presented in Sections 2.4 and 2.5.3.

We assume given two disjoint sets of variables: one for *small variables* ($\mathcal{V}ar^\star$) and one for *big variables* ($\mathcal{V}ar^\Box$). Let $\mathcal{V}ar = \mathcal{V}ar^\star \cup \mathcal{V}ar^\Box$.

The set of pseudoterms is given by the following grammar:

$$a ::= x \mid \star \mid \Box \mid (a\,a) \mid (\lambda x\!:\!a.a) \mid ((x\!:\!a).a)$$

The pseudoterms $\star$ and $\Box$ are constants called *sorts*. The pseudoterm $u\,v$, denotes the application of u to v. The pseudoterm $\lambda x\!:\!u.v$ is a lambda abstraction. The pseudoterm $(x\!:\!u).v$ denotes the dependent product (traditionally, it is written $\Pi x{:}u.\,v$). If x does not occur in v, we write it also as $u \to v$.

For convenience, we will omit parentheses whenever possible, assuming that product (and hence arrow) associates to the right and application to the left.

We denote by $\mathcal{FV}(a)$ the set of free variables of a. This notation naturally extends to sets : $\mathcal{FV}(A) = \bigcup_{a\in A} \mathcal{FV}(a)$. We use also $\mathcal{FV}^{\Box}(A)$ for $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box}$. We assume that bound variables in a pseudoterm are all different, and are different from the free ones. As usual, terms that only differ in their bound variables are identified. The notation $\vec{a}$ denotes a sequence of pseudoterms, but it will sometimes be used for multisets. The size of a pseudoterm a is denoted by $|a|$.

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

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 $\Gamma = x_1 : a_1, \ldots, x_n : a_n$ the domain $dom(\Gamma)$ of Γ is the list $x_1, \ldots, x_n$, and we write $\Gamma(x_i) = a_i$.

The sequence of declarations $x_1 : a_1, \ldots, x_n : a_n$ 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 (abbreviating $(x_1 : b_1) \ldots (x_n : b_n).c$ by $\overrightarrow{(x : b)}.c$). By convention, the type $\overrightarrow{(x : b)}.c$ is simply c if $\overrightarrow{(x : b)}$ is empty.

The typing rules given below derive *judgments* of the form $\Gamma \vdash_{CC_0} a : b$ stating that a has type b in the environment Γ in the pure calculus of constructions.

Definition 2.1.1 (Pure calculus of constructions)

(ax) $$\frac{}{\vdash_{CC_0} \star : \Box}$$

(var) $$\frac{\Gamma \vdash_{CC_0} a : p}{\Gamma, x : a \vdash_{CC_0} x : a} \qquad (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{\textit{(weak)}}\ \frac{\Gamma \vdash_{CC_0} a : b \qquad \Gamma \vdash_{CC_0} c : p}{\Gamma, x : c \vdash_{CC_0} a : b} \quad (x \in \mathcal{V}ar^p \setminus dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{\textit{(abs)}}\ \frac{\Gamma, x : a \vdash_{CC_0} b : c \qquad \Gamma \vdash_{CC_0} (x\!:\!a).c : p}{\Gamma \vdash_{CC_0} \lambda x\!:\!a.b : (x\!:\!a).c} \quad (x \notin dom(\Gamma), p \in \{\star, \Box\})$$

$$\textbf{\textit{(app)}}\ \frac{\Gamma \vdash_{CC_0} a : (x\!:\!b).c \qquad \Gamma \vdash_{CC_0} d : b}{\Gamma \vdash_{CC_0} a\, d : c[d/x]}$$

$$\textbf{\textit{(prod)}}\ \frac{\Gamma \vdash_{CC_0} a : p \qquad \Gamma, x : a \vdash_{CC_0} b : q}{\Gamma \vdash_{CC_0} (x\!:\!a).b : q} \quad (x \notin dom(\Gamma), p, q \in \{\star, \Box\})$$

$$\textbf{\textit{(conv)}}\ \frac{\Gamma \vdash_{CC_0} a : b \qquad \Gamma \vdash_{CC_0} b' : p}{\Gamma \vdash_{CC_0} a : b'} \quad (p \in \{\star, \Box\}, b \overset{*}{\leftrightarrow}_\beta b')$$

The relation $\overset{*}{\leftrightarrow}_\beta$ used in the **(conv)** rule is called *beta conversion*. It is the reflexive, symmetric and transitive closure of beta reduction, denoted by $\rightarrow_\beta$, which is defined as the compatible closure of the β-reduction rule: $(\lambda x\!:\!a.b)\, c \rightarrow b[c/x]$.

The pure calculus of constructions is confluent and terminating with respect to beta reduction. As a consequence, the relation $\Gamma \vdash_{CC_0} a : b$, called type-checking, is decidable in this calculus.

2.2 Signature

In the pure calculus of constructions there are only two constants $\star$ and $\Box$, with $\vdash_{CC_0} \star : \Box$. The goal of this paper it to prove the strong normalization of the calculus of constructions extended with functions defined by rewriting. To this end, we need additional constant symbols in the calculus.

In this section we define the notion of signature, which is a sequence of declarations of functions and inductive types. These declarations provide types and arities for three kinds of constants we consider: function symbols, type constructors and constructors. There will be no possibility to define mutually inductive types.

Since we want to add constants to the pure calculus of constructions without losing good meta-theoretical properties, the types of these constants cannot be arbitrary. In the definition of a well-formed signature (Def. 2.3.3) we will impose these types to be well-typed terms of the pure calculus of constructions. On the other hand, we must allow constants to appear in types of other constants. To fulfill these two requirements, we will treat constants as variables while type-checking their types in the pure calculus of construction.

In Section 2.4, given a well-formed signature, we build a version of the calculus of constructions where the sets of function symbols, type constructors, constructors and variables are all pairwise disjoint. This calculus will consist of the rules presented in the previous section together with an explicit rule for constant application, ensuring that arity constraints are satisfied.

In our framework an *arity* is either a natural number or a pair of natural numbers. This pair is used for function symbols and constructors and it corresponds to the division of arguments into parameters and operands (other arguments).

Definition 2.2.1 (Signature) *A* signature Σ *is a sequence of declarations, each declaration being of the form:*

- $Fun[p_1 : P_1, \ldots p_r : P_r](f^{(r,n)} : (x_1 : b_1) \ldots (x_n : b_n).c)$*, where* $f, \vec{p}, \vec{x} \in \mathcal{V}ar$ *and* $\vec{P}$*,* $\vec{b}$*,* c *are pseudoterms. This is a declaration of a function symbol* f *of arity* (r,n) *and of type* $(p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).c$*.*

- $Ind[p_1 : P_1, \ldots p_r : P_r](s^r : A := c_1^{(r,k_1)} : C_1, \ldots c_m^{(r,k_m)} : C_m)$*, where* $s, \vec{p}, \vec{c} \in \mathcal{V}ar$*,* $\vec{P}$*,* A *are pseudoterms and* $\vec{C}$ *are pseudoterms containing a special local variable* $\overline{s}$*. This is a declaration of a type constructor* s *of arity* r *and constructor symbols* $\vec{c}$*, each* c_i *being of arity* (r, k_i)*. The type of* s *is* $(p_1 : P_1) \ldots (p_r : P_r).A$ *and the type of* c_i*'s are* $(p_1 : P_1) \ldots (p_r : P_r).C_i[(sp_1, \ldots p_r)/\overline{s}]$*.*

Let $E(\Sigma)$ *be the set of declared symbols. It is assumed that every* $e \in E(\Sigma)$ *belongs to exactly one declaration. Moreover, we require that for every occurrence of* e *in a pseudoterm in* Σ*, if* e *is of arity* r *then it is applied to at least* r *pseudoterms and if* e *is of arity* (r,n) *then it is applied to at least* $r+n$ *pseudoterms.*

Given a signature Σ we may transform it to the environment Γ assigning types to the symbols declared in Σ. The order of declarations in the environment Γ corresponds to the order of declarations in Σ, and the declaration for a type constructor comes before the declarations for its constructors.

Example 1 *Consider the following signature declaring polymorphic lists and a map function.*

$$\begin{array}{rcl}\Sigma & = & Ind[A:\star](List^1 : \star := nil^{(1,0)} : \overline{List},\ cons^{(1,2)} : A \to \overline{List} \to \overline{List}) \\ & & Fun[A:\star, B:\star](map^{(2,2)} : (A \to B) \to (List\ A) \to (List\ B))\end{array}$$

The corresponding environment is

$$\begin{array}{rcl} \mathit{List} & : & \star \to \star, \\ \mathit{nil} & : & (A:\star).(\mathit{List}\ A), \\ \mathit{cons} & : & (A:\star)A \to (\mathit{List}\ A) \to (\mathit{List}\ A), \\ \mathit{map} & : & (A:\star)(B:\star)(A \to B) \to (\mathit{List}\ A) \to (\mathit{List}\ B) \end{array}$$

From now on, we will write $e : \overrightarrow{(x:b)}.c \in \Sigma$ if $e \in E(\Sigma)$ and $\overrightarrow{(x:b)}.c$ is its type. Similarly we will write $\Sigma \vdash_{CC_0} a : b$ meaning that in the calculus of constructions a has a type b in the environment corresponding to Σ.

2.3 Well-formed signatures

In this section, we present all conditions that must be satisfied by the declarations given in a signature. Briefly speaking:

1. The types of function symbols must satisfy the star dependency condition.

2. The types of constructors must be strictly positive and must satisfy the star dependency condition.

2.3.1 Star dependency

The star dependency condition is one of the key concepts of the work. Function and constructor symbols have their arguments classified as *parameters* and *operands* (other arguments). Star dependency is used to check whether this separation of arguments is correct.

For function symbols, star dependency influences the form of the rewriting rules accepted by HORPO, restricting parameter arguments to be distinct variables (see Section 3.1.3 and 3.1.4). This condition is crucial in the proof of strong normalization.

For constructors, star dependency makes it possible to define the interpretation of an inductive type (see Section 5.2.1). For a deeper discussion on the star dependency condition we refer the reader to Chapter 6.

Definition 2.3.1 *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 the arity* (r, n) *if*

$$\mathcal{FV}^{\square}(b_1, \dots b_n, c) \subseteq \{p_1, \dots p_r\}$$

In other words this condition means that all dependencies in $b_1, \ldots b_n, c$ are the dependencies on small variables ($x_i \in \mathcal{V}ar^\star$) or parameters. This condition may be seen as a generalization of the small constructor requirement that all variables corresponding to non-parameter arguments, $x_1, \ldots x_n$ belong to $\mathcal{V}ar^\star$.

Example 2 *The type of map*

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

satisfies star dependency w.r.t the arity $(2, 2)$.

If map had arity $(1, 3)$*, then its type would not satisfy the star dependency condition, as the output type of* f *depends on the big variable* B *that would no more be a parameter.*

Of course, if all arguments of map were parameters (i.e. map is of arity $(4, 0)$*) then its type would obviously satisfy the star dependency condition. On the other hand this would have a quite negative impact on the form of the rules for map acceptable by HORPO (see Definition 3.1.2).*

2.3.2 Strict positivity

Given an inductive declaration $Ind[p_1 : P_1, \ldots p_r : P_r](s^r : A := c_1^{(r,k_1)} : C_1, \ldots c_m^{(r,k_m)} : C_m)$, *strict positivity* is a syntactic condition on the form of $\vec{C}$, which is essentially the same as in [87, 77].

Definition 2.3.2 *A* type generated by $\overline{s}$ *is a pseudoterm of the form* $\overrightarrow{(x : b)}.\overline{s}a_1 \ldots a_n$.

A type $t = \overrightarrow{(x : b)}.\overline{s}a_1 \ldots a_n$ *generated by* $\overline{s}$ *satisfies the* positivity condition *if* $\overline{s}$ *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* $\overline{s}$ *occurs is of the form* $\overrightarrow{(x : d)}.\overline{s}e_1 \ldots e_k$ *and* $\overline{s}$ *does not occur in* $\vec{d}, \vec{e}$.

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

Example 3 *The types generated by* $\overline{s}$ *given below do not satisfy the positivity condition:*

1. $(\overline{s} \to \overline{s}) \to \overline{s}$,
2. $((\overline{s} \to Nat) \to Nat) \to \overline{s}$,

3. $\overline{s} \to (List\ \overline{s}) \to \overline{s}$.

In fact, the first occurrence of $\overline{s}$ *in 1 is negative and the first occurrence of* $\overline{s}$ *in 2 is positive, but not strict. In 3 an occurrence of* $\overline{s}$ *under List is not allowed.*

On the other hand the following types generated by $\overline{s}$ *satisfy the positivity condition:*

1. $\overline{s} \to \overline{s}\ \in Pos^1(\overline{s})$,
2. $(A : \star)A \to \overline{s} \to \overline{s}\ \in Pos^3(\overline{s})$,
3. $(Nat \to \overline{s}) \to \overline{s}\ \in Pos^1(\overline{s})$.

2.3.3 Well-formedness of the signature

The following definition summarizes all conditions that must be satisfied by the signature.

Definition 2.3.3 (Well-formed signature) *A signature* Σ *is well-formed if we can deduce* Σ *ok using the following rules:*

(start) $\emptyset$ *ok*

(fun)

$$\frac{\Sigma\ ok \qquad \Sigma \vdash_{CC_0} (p_1 : P_1) \dots (p_r : P_r)(x_1 : b_1) \dots (x_n : b_n).c\ :\ \star}{\Sigma, Fun[p_1 : P_1, \dots p_r : P_r](f^{(r,n)} : (x_1 : b_1) \dots (x_n : b_n).c)\ \ ok}$$

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.* (r, n).

(ind)

$$\frac{\Sigma\ ok \qquad \Sigma, \Gamma_p \vdash_{CC} A : \square \qquad \forall i \quad \Sigma, \Gamma_p, \overline{s} : A \vdash_{CC} C_i : \star}{\Sigma, Ind[\Gamma_p](s^r : A := c_1^{(r,k_1)} : C_1, \dots c_m^{(r,k_m)} : C_m)\ \ ok}$$

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

- $A = \overrightarrow{(x : b)}.\star$,
- $\forall i\ C_i \in Pos^{k_i}(\overline{s})$,
- $\forall i\ \overrightarrow{(p : P)}.C_i[s\vec{p}/\overline{s}]$ *satisfies the star dependency condition w.r.t.* (r, k_i).

2.4 Calculus of constructions with constants

From now on, we assume a given well-formed signature Σ. We extract three sets of constants declared in Σ: the set of function symbols $\mathcal{F}$, of type constructors $\mathcal{TC}$ and of constructors $\mathcal{CS}$. Moreover, we assume that they are disjoint from the sets of variables $\mathcal{V}ar^\star$ and $\mathcal{V}ar^\Box$.

In this section, we construct the calculus of constructions consisting of the rules presented in Section 2.1 and of a rule for constant application which ensures that arity constraints are satisfied.

As already said, function and constructor symbols have their arguments classified as parameters and operands (their arities are 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 2.4.1 (Pseudoterms) *Pseudoterms are built with the usual constructions and constants. Below, s denotes a type constructor, c a constructor and f a function symbol.*

$$\begin{aligned} a ::= x \mid \star \mid \Box \mid (\lambda x\!:\!a.a) \mid ((x\!:\!a).a) \mid (a\, a) \mid s(a_1, \ldots a_n) \\ \mid c(a_1, \ldots a_n) \mid f(a_1, \ldots a_n) \end{aligned}$$

For convenience, we will omit parentheses around lambda abstraction, product and application whenever possible.

We will write $@(u, v)$ for the application of u to v making the application operator explicit. We will even use notation $@(u, v_1, \ldots v_n)$ for application $(\ldots((u\, v_1)v_2) \ldots v_n)$. If $a = (\ldots((u\, v_1)v_2) \ldots v_n)$ then $@(u, v_1, \ldots v_n)$ is called a *left-flattening* of a. Note that u may still be an application. 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 operands $\vec{b}$. The same applies to $c \in \mathcal{CS}^{(r,n)}$.

Pseudoterms are identified with finite trees whose nodes are labeled with constants, applications and λx and Πx considered 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 $\rhd$ (in $a \rhd b$, term b is a subterm of a) and its non-strict version by $\unrhd$.

Free variables of a term headed by a constant e are given by $\mathcal{FV}(e(a_1,\ldots a_n)) = \mathcal{FV}(a_1) \cup \ldots \cup \mathcal{FV}(a_n)$. For other pseudoterms free variables are defined as in the pure calculus of constructions.

The notions of substitution and environment extend to pseudoterms with constants in an obvious way. Recall that substitutions are written $[a_1/x_1,\ldots a_n/x_n]$ or $[\vec{a}/\vec{x}]$ and that we use postfix notation for their application. We use letters θ, ρ, μ for substitutions. Recall also that an environment Γ is an ordered sequence of declarations, where all variables in the domain are distinct. We write $\Gamma \subseteq \Gamma'$ if Γ and Γ' are environments such that $x \in dom(\Gamma')$ for all $x \in dom(\Gamma)$, and $\Gamma(x) = \Gamma'(x)$. A *sequent* $\Gamma \vdash a$ is a pair consisting of an environment and a pseudoterm.

Typing rules given below derive *judgments* of the form $\Gamma \vdash_{CC} a : b$ stating that a has type b in the environment Γ in the calculus of constructions.

Definition 2.4.2 (Calculus of constructions)

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

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

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

$$\textit{\textbf{(const)}} \quad \frac{\Gamma \vdash_{CC} \star : \Box \quad \forall\, i = 1,\ldots m \quad \Gamma \vdash_{CC} a_i : b_i[\vec{a}/\vec{x}]}{\Gamma \vdash_{CC} e(\vec{a}) : c[\vec{a}/\vec{x}]} \qquad (e : (x_1 : b_1) \ldots (x_m : b_m).c \in \Sigma)$$

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

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

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

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

In rule **(const)** the judgment $\Gamma \vdash_{CC} \star : \Box$ ensures the validity of Γ in case e has no argument (according to Σ).

It is important to notice that all good meta-theoretical properties of the pure calculus of constructions are preserved by the addition of constants. This follows from the fact that the **(const)** rule is just a serialized version of the **(app)** rule and that arity constraints are invariant under beta reduction. For a given environment Γ and two pseudoterms a and b, respecting the arities, we have the following equivalence:

$$\Gamma \vdash_{CC} a : b \quad \text{if and only if} \quad \Sigma, \Gamma \vdash_{CC_0} a : b$$

This shows that the calculus with constants can be seen as a subsystem of the pure calculus of constructions and that it inherits the desired meta-theoretical properties, like subject reduction, confluence, strong normalization and decidability of type-checking.

For these reasons, we will refer, from now on, to the calculus with constants as the calculus of constructions (while still calling the one from Section 2.1 the pure calculus of constructions).

2.5 Auxiliary definitions

2.5.1 Constructor subterms

Definition 2.5.1 (Constructor subterm) *A subterm at a 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 $a \rhd^{\mathcal{CS}} b$ if b is a constructor subterm of a.*

Another useful relation is *recursive constructor subterm.*

Definition 2.5.2 *The i-th position in the constructor $c : \overrightarrow{(p : P)}\overrightarrow{(z : d)}.s(\vec{p})\vec{w}$ of an inductive type s is* recursive *if d_i is of the form $(x_1 : T_1) \dots (x_n : T_n).s(\vec{p})\vec{v}$. It is called a* nonfunctional recursive position *if $n = 0$; otherwise it is called a* functional recursive position

Definition 2.5.3 (Recursive constructor subterm) *A constructor subterm at position $p \neq \Lambda$ is called* recursive *if every term on the path from the root down to p (p included) is at a recursive position of the corresponding constructor. The recursive constructor subterm relation is denoted by $\rhd^{\mathcal{RCS}}$.*

Of course every recursive constructor subterm is also a constructor subterm.

Example 4 *Let* $cons : (A : \star)A \to List(A) \to List(A)$ *be a constructor,* $f : (A : \star)A \to A$ *be a function symbol and let*

$$t = cons(A, a, cons(A, b, cons(A, f(A, c), nil(A))))$$

Then $cons(A, b, cons(A, f(A, c), nil(A)))$, $cons(A, f(A, c), nil(A))$ *and* $nil(A)$ *are recursive constructor subterms of* t*. Apart from the occurrences of* A *and* c *which are under* f*, all other subterms are constructor subterms.*

2.5.2 Composing substitutions

Definition 2.5.4 *Given two substitutions* ρ_1 *and* ρ_2 *with the domains* D_1 *and* D_2 *such that* $D_1 \cap D_2 = \emptyset$*, 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*

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

Definition 2.5.5 *Given two substitutions* ρ_1 *and* ρ_2 *with the domains* D_1 *and* D_2*, a* sequential composition*, denoted by* $\rho_1\rho_2$*, is a substitution defined by*

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

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

Let us now introduce some useful definitions and extensions of a given (not necessarily transitive) relation $>$ on terms. In these extensions $=$ denotes the syntactic equality of terms. We will also use the notation $\geq$ for the union of $>$ and equality.

The *lexicographic composition* of $>_1, \ldots >_n$, denoted by $(>_1, \ldots >_n)_{lex}$, is defined as:

$$(a_1, \ldots a_n)(>_1, \ldots >_n)_{lex}(b_1, \ldots b_n) \quad \text{if} \quad \exists i \in \{1, \ldots n\}\ a_i >_i b_i \\ \text{and} \quad \forall j < i \quad a_j = b_j$$

For a given n, the *lexicographic extension* of $>$, denoted by $>_{lex}$, is a lexicographic composition of n copies of $>$.

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 relations $>, >_1, \ldots >_n$, the relations $(>_1, \ldots >_n)_{lex}$, $>_{lex}$ and $>_{mul}$ are also terminating [38].

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

Chapter 3

Calculus of constructions with rewriting induced by HORPO

The goal of this thesis 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 so-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 (as defined in Subsection 2.4). For the discussion about relaxing this assumption see Section 9.2.

As it was already mentioned in the introduction, strong normalization will be shown under the assumption that rewriting operates at the object level. In our formalism, it is ensured by the condition $f : T : \star$ in the definition of a well-formed signature.

3.1 The dependent Higher-Order Recursive Path Ordering

Our definition of the Higher Order Recursive Path Ordering needs six ingredients: the precedence $>_{\mathcal{F}}$ (Section 3.1.1), the status (Section 3.1.2), the notion of a parametric rule (Section 3.1.3), the relation HORPO, written $\succ$

(Section 3.1.4), the computable closure $CCl_f(\vec{l})$ (Section 3.2) and the constructor extensions, $\succ^{\mathcal{CS}}$ and $\succ^{\mathcal{RCS}}$, of HORPO (Section 3.3). The first three of them can be presented independently, but the last three are mutually dependent and are formally defined simultaneously. In all definitions in this section, we use type-checking in the calculus of constructions, that is the relation $\vdash_{CC}$.

3.1.1 Precedence

From now on, we assume that the set of function symbols $\mathcal{F}$ is partially ordered by a well-founded relation $>_{\mathcal{F}}$, called *precedence*.

3.1.2 Status

We assume also that there is a total function from $\mathcal{F}$ to $\{Lex, Mul\}$, called *status*. We write $f \in Lex$ ($f \in Mul$) if the function symbol f has a lexicographic (multiset) status. A status tells us how to compare recursively immediate subterms of two terms headed by the same function symbol.

3.1.3 Parametric rules

Definition 3.1.1 (Parametric rule) *A* parametric rule *is a quadruple* (Γ, l, r, A)*, also written* $\Gamma \vdash l \to r : A$*, such that*

1. *l is headed by a function symbol $f \in \mathcal{F}^{(r',n)}$ of type $(p_1 : P_1) \dots (p_{r'} : P_{r'})(x_1 : b_1) \dots (x_n : b_n).c$,*
2. *$\Gamma = p_1 : P_1, \dots p_r : P_r, G$,*
3. *$l = f(p_1, \dots p_{r'}, u_1, \dots u_n)$,*
4. *$\Gamma \vdash_{CC} l : A$, $\Gamma \vdash_{CC} r : A$.*

In other words, in a parametric rule all parameter arguments of the left-hand side are different variables.

3.1.4 HORPO

The Higher Order Recursive Path Ordering (HORPO) presented in Definition 3.1.2 below is an adaptation to the calculus of constructions of HORPO from [53], which was itself a generalization to the simply typed lambda calculus of the well-known RPO [35] (cf. 1.1.2).

HORPO is a relation on terms in an environment and can compare only terms of equal types. Every judgment of HORPO has the form $\Gamma \vdash l \succ r : A$ where Γ is an environment, l, r are terms and A is their type. The term l is called bigger or the left-hand side and the term r smaller or the right-hand side of the judgment $\Gamma \vdash l \succ r : A$.

The definition of HORPO will use its own reflexive extension $\succeq$, constructor subterm extension $\succ^{\mathcal{CS}}$, recursive constructor subterm extension $\succ^{\mathcal{RCS}}$, their multiset and lexicographic extensions and the computable closure $CCl_f(\vec{l})$ of the set of terms $\vec{l}$ w.r.t. the function symbol f. For now, it is by no means important to know the exact definitions of the computable closure $CCl_f(\vec{l})$: taking $\vec{l}$ itself is enough for understanding HORPO. Knowing the exact definition of the constructor extensions is not necessary either: it is sufficient to remember that both contain HORPO properly.

Compared to RPO, HORPO has two new cases allowing to deal with an application and constructor headed term on the right (in Part I) and four cases corresponding to monotonicity rules for application, abstraction, product and constants from $\mathcal{TC} \cup \mathcal{CS}$ (in Part II). Moreover, it has more possibilities to deal with subterms from the smaller side, which is described by the predicate $\mathfrak{P}$. The multiple choices in $\mathfrak{P}$ aim to balance the strong requirement that HORPO compares only terms of equal types.

Definition 3.1.2 (HORPO) *The higher order recursive path ordering is defined as follows:*

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

Part I $\quad l = f(\vec{p}, \vec{u}) = f(\vec{l})$, (Γ, l, r, A) *is a parametric rule and*

1. $r \in CCl_f(\vec{l})$ *or* $\Gamma \vdash l_i \succeq^{\mathcal{CS}} r$ *or* $\Gamma \vdash l_i \succeq^{\mathcal{RCS}} r$ *for some* i, *or*
2. $r = g(\vec{t}, \vec{w})$, $g \in \mathcal{F}$, $f >_{\mathcal{F}} g$ *and* $\mathfrak{P}(\vec{t}, \vec{w})$, *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. $r = f(\vec{p}, \vec{w})$, $f \in Lex$, $\Gamma \vdash (\vec{p}, \vec{u}) \succ^{\mathcal{RCS}}_{lex} (\vec{p}, \vec{w})$ *and* $\mathfrak{P}(\vec{w})$, *or*
5. $r = c(\vec{w})$, $c \in \mathcal{CS}$ *and* $\mathfrak{P}(\vec{w})$, *or*
6. $@(w_0, \ldots, w_m)$ *is a left-flattening of* r *and* $\mathfrak{P}(\vec{w})$,

where

$$\mathfrak{P}(W) \;=\; \forall w \in W \quad \Gamma \vdash f(\vec{l}) \succ w : A \quad \textit{or} \quad \Gamma \vdash l_j \succeq^{\mathcal{CS}} w \textit{ for some } l_j$$
$$\textit{or} \quad \Gamma \vdash l_j \succeq^{\mathcal{RCS}} w \textit{ for some } l_j \quad \textit{or} \quad w \in CCl_f(\vec{l})$$

Part II

1. $l = \lambda x : l_1.l_2$, $r = \lambda x : r_1.r_2$, $l_1 \overset{*}{\leftrightarrow}_\beta r_1$ *and* $\Gamma, x : l_1 \vdash l_2 \succ r_2 \ : C$ *for some* C*, or*

2. $l = (x : l_1).l_2$, $r = (x : r_1).r_2$ *and* $l_1 \overset{*}{\leftrightarrow}_\beta r_1$ *and* $\Gamma, x : l_1 \vdash l_2 \succ r_2 \ : A$*, or*

3. $l = l_1\, l_2$, $r = r_1\, r_2$, $\Gamma \vdash l_1 \succeq r_1 \ : (x : B).C$, $\Gamma \vdash l_2 \succeq r_2 \ : B$*, for some* B *and* C*, and at least one of the two comparisons is strict, or*

4. $l = e(l_1, \ldots l_n)$, $r = e(r_1, \ldots r_n)$, $e : \overrightarrow{(x : b)}.c \in \mathcal{TC} \cup \mathcal{CS}$*, 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.*

Part I is the heart of HORPO, as its cases 1, 2, 3, 4 correspond to RPO. In Part I the left-hand side is always a function symbol headed term, while 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). More precisely, if we want to show that $\Gamma \vdash l \succ r \ : A$ and $l = f(\vec{l})$ with $f \in \mathcal{F}$ then the only way to do it is to show that one of the conditions described in Part I is satisfied.

If we want to use case 2 then r should start with a function symbol g smaller than f in the precedence and all immediate subterms of r should satisfy the predicate $\mathfrak{P}$. In cases 3 and 4, r should start with the same function symbol as l and their immediate subterms should be comparable in the lexicographic extension of $\succ^{\mathcal{RCS}}$ or multiset extension of $\succ^{\mathcal{CS}}$. In case of a lexicographic comparison, since the comparison stops when a strict inequality is found, all (other) immediate subterms of r must be controlled by the comparisons performed in $\mathfrak{P}$.

As in [53], in case 6 an appropriate left-flattening has to be chosen non-deterministically. This non-determinism aims to ease the recursive comparison in case of type incompatibilities (HORPO compares only terms of the same type). Independently of the form of r we may always try to use case 1, which is satisfied when either $r \in CCl_f(\vec{l})$ or r is smaller (or equal) than one of the immediate subterms of l in some constructor extension of HORPO.

Part II may be considered as some kind of closure by context of the relation generated by Part I. One can imagine the definition of HORPO without Part II, but then it would not be possible to show $x : Nat \vdash f(s(plus(x,1))) \succ f(s(x)) \ : Nat$, even though $x : Nat \vdash plus(x,1) \succ x \ : Nat$ holds. On the other hand even with Part II, HORPO is not stable under context. The reason is that we use the calculus of constructions for type-checking (with beta equality in conversion) while we use HORPO reductions in types. Consider the example: $l2 = plus(0,0)$, $r2 = 0$ and $l_1 = r_1 = \lambda x{:}Nat.P(x)$, where $P : (x{:}Nat).Or(x = 0, x > 0)$ is the function that for every x gives a proof that $x = 0$ or $x > 0$. Then $\Gamma \vdash l2 \succ r2 \, : \, Nat$ and $l1 = r1$, but $l_1\, l_2$ and $r_1\, r_2$ are incomparable in HORPO because they have different types $P(plus(0,0))$ and $P(0)$ (these types will be convertible only in CC+H). The real closure by context (and by substitution) will be done in the **(rew)** rule in the definition of CC+H (Definition 3.6.2).

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 $\vec{p}$ in its parameter positions (according to the parametricity condition 3.1.1). Fortunately, the parameter arguments of the right-hand side in case 2 of Part I do not have to be variables, which allows us to consider rules like $l : List(Nat) \vdash length\_nat(l) \succ length(Nat, l) \, : \, Nat$, where $length\_nat : List(Nat) \to Nat$ and $length : (A : \star)List(A) \to Nat$.

3.2 The computable closure

The computable closure is used in Part I of HORPO and in the definition of its recursive constructor subterm extension $\succ^{\mathcal{RCS}}$.

Given a sequent $\Gamma \vdash f(\vec{l})$ (the left-hand side of a HORPO judgment), the computable closure $CCl(\Gamma \vdash f(\vec{l}))$, written also $CCl_f(\vec{l})$, is the closure of the set $\vec{l}$ under some computability preserving operations. Computability refers here to Tait and Girard celebrated reducibility predicate technique.

The computable closure is constructed in three steps. First, in the initial step the only available terms are $\vec{l}$. Then all their constructor subterms are added, to form the accessible set. The last step consists in building new terms from existing ones by using operations like abstraction, application, application of a constructor and so on. The rules for constructing these terms recall very much the typing rules for the calculus of constructions with some additional side conditions. Note that the side conditions of one of the rules, **Recursive call**, contain HORPO comparisons, making HORPO and the computable closure mutually dependent.

For technical reasons the computable closure is not a set of terms, but a set of triples consisting of an environment, a term and a type, called Γ-tuples.

Definition 3.2.1 (Γ-tuple) *A triple* $(BV, \; t, \; T)$ *is called a* Γ-tuple *if* $\Gamma, BV \vdash_{CC} t \, : \, T$. *The* BV *component is an environment consisting of variables to be bound.*

The environment BV allows keeping track of those (temporarily) free variables that will become bound later on. The type information T will be very useful in the proof of strong normalization.

We write $t \in CCl_f(\vec{l})$ to express the existence of a term T such that the tuple $(\emptyset,\ t,\ T)$ belongs to $CCl(\Gamma \vdash f(\vec{l}))$, the environment Γ being clear from the context.

Definition 3.2.2 (Computable Closure) *Let f have the type $(p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).c$ and let us suppose that $\Gamma \vdash_{CC} f(\vec{p}, \vec{u}) : c[\vec{u}/\vec{x}]$, where $\Gamma = \overrightarrow{p : P}, G$, for some environment G.*

I. *The* initial set $Init(\Gamma \vdash f(\vec{p}, \vec{u}))$ *is*

$$\{(\emptyset,\ p_i,\ P_i) \mid 1 \leq i \leq r\} \ \cup\ \{(\emptyset,\ u_j,\ b_j[\vec{u}/\vec{x}]) \mid 1 \leq j \leq n\}$$

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

II. *The* accessible set $Acc(\Gamma \vdash f(\vec{p}, \vec{u}))$ *is the closure of $Init(\Gamma \vdash f(\vec{p}, \vec{u}))$ under the following two rules:*

$$\frac{(BV,\ c(\vec{a'}, \vec{N}),\ s(\vec{a})\vec{u})}{(BV,\ a'_i,\ Q_i[\vec{a}/\vec{q}])} \qquad\qquad \frac{(BV,\ c(\vec{a'}, \vec{N}),\ s(\vec{a})\vec{u})}{(BV,\ N_i,\ d_i[\vec{a}/\vec{q}, \vec{N}/\vec{z}])}$$

where $c : \overrightarrow{(q : Q)}\overrightarrow{(z : d)}.s(\vec{q})\vec{w}$ is a constructor of $s : \overrightarrow{(q : Q)}\overrightarrow{(y : e)}.\star$.

III. *The* computable closure of $\Gamma \vdash f(\vec{p}, \vec{u})$, *denoted by $CCl(\Gamma \vdash f(\vec{p}, \vec{u}))$, is the closure of $Acc(\Gamma \vdash f(\vec{p}, \vec{u}))$ under the following rules:*

Axiom

$$\frac{}{(\emptyset,\ \star,\ \Box)}$$

Var

$$\frac{\Gamma, BV \vdash_{CC} A : p}{((BV, x : A),\ x,\ A)} \qquad (x \in \mathcal{V}ar^p \backslash dom(\Gamma, BV),\ p \in \{\star, \Box\})$$

Weak

$$\frac{(BV,\ t,\ T) \qquad \Gamma, BV \vdash_{CC} A : p}{((BV, x : A),\ t,\ T)} \qquad (x \in \mathcal{V}ar^p \backslash dom(\Gamma, BV),\ p \in \{\star, \Box\})$$

Abstraction

$$\frac{((BV, y:A),\ t,\ T) \qquad (BV,\ A,\ p_2) \qquad \Gamma, BV, y:A \vdash_{CC} T : p_1}{(BV,\ \lambda y\!:\!A.t,\ (y\!:\!A).T)} \quad (x \notin dom(\Gamma, BV),\ p_1, p_2 \in \{\star, \Box\})$$

Application

$$\frac{(BV,\ t_1,\ (x\!:\!A).B) \qquad (BV,\ t_2,\ A)}{(BV,\ @(t_1, t_2),\ B[t_2/x])}$$

Precedence

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ \ (BV,\ t_i,\ Q_i[\vec{t}/\vec{q}]) \qquad \forall i\ \ (BV,\ w_i,\ d_i[\vec{t}/\vec{q}, \vec{w}/\vec{y}])}{(BV,\ g(\vec{t}, \vec{w}),\ e[\vec{t}/\vec{q}, \vec{w}/\vec{y}])}$$

if $f >_{\mathcal{F}} g$ and $g : \overrightarrow{(q:Q)}\overrightarrow{(y:d)}.e$.

Recursive call

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ \ (BV,\ w_i,\ b_i[\vec{w}/\vec{x}])}{(BV,\ f(\vec{p}, \vec{w}),\ c[\vec{w}/\vec{x}])}$$

if $\Gamma \vdash \vec{p}, \vec{u} (\succ^{\mathcal{RCS}}_{BV, f(\vec{p}, \vec{u})})_{lex} \vec{p}, \vec{w}$ or $\Gamma \vdash \vec{p}, \vec{u} \succ^{\mathcal{CS}}_{mul} \vec{p}, \vec{w}$ depending on the status of f. Terms $\vec{p}, \vec{u}$ are initial terms. The relations $\succ^{\mathcal{RCS}}_{BV, f(\vec{p}, \vec{u})}$ and $\succ^{\mathcal{CS}}$ will be defined in Section 3.3.

Constructor

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ (BV,\ a_i,\ Q_i[\vec{a}/\vec{q}]) \qquad \forall i\ (BV,\ N_i,\ d_i[\vec{a}/\vec{q}, \vec{N}/\vec{z}])}{(BV,\ c(\vec{a}, \vec{N}),\ (s(\vec{q})\vec{w})[\vec{a}/\vec{q}, \vec{N}/\vec{z}])}$$

$c : \overrightarrow{(q:Q)}\overrightarrow{(z:d)}.s(\vec{q})\vec{w}$ is a constructor of $s : \overrightarrow{(q:Q)}\overrightarrow{(y:e)}.\star$.

Type constructor

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ (BV,\ a_i,\ Q_i[\vec{a}/\vec{q}])}{(BV,\ s(\vec{a}),\ \overrightarrow{(y:e)}.\star)} \quad (s : \overrightarrow{(q:Q)}\overrightarrow{(y:e)}.\star \in \mathcal{TC})$$

Product

$$\frac{(BV,\ A,\ p_1) \quad ((BV, x:A),\ B,\ p_2)}{(BV,\ (x\!:\!A).B,\ p_2)} \quad (x \notin dom(\Gamma, BV),\ p_1, p_2 \in \{\star, \Box\})$$

Reduction

$$\frac{(BV,\ t,\ T)}{(BV,\ t',\ T)} \qquad (t \rightarrow_\beta t')$$

Conversion

$$\frac{(BV,\ t,\ T) \qquad \Gamma, BV \vdash_{CC} T' : p \qquad \mathcal{T} \overset{*}{\leftrightarrow}_\beta T'}{(BV,\ t,\ T')} \quad (p \in \{\star, \Box\})$$

The following lemma states the correctness of the above rules.

Lemma 3.2.3 *Every triple $(BV,\ t,\ T) \in CCl(\Gamma \vdash f(\vec{p}, \vec{u}))$ is a Γ-tuple.*

Proof
Easy induction on the derivation of the tuple. $\Box$

From the definition of Γ-tuples it follows that every term $t \in CCl(\Gamma \vdash f(\vec{l}))$ is typable in the calculus of constructions in the environment Γ.

Let us see an example that shows in details how to use the computable closure.

Example 5 *Let $map \in \mathcal{F}^{(2,2)}$ be of type $(A : \star)(B : \star)(A \rightarrow B) \rightarrow List(A) \rightarrow List(B)$. We will show that the terms $@(f,\ a)$ and $\lambda x{:}C.@(f,\ x)$ belong to $CCl(C : \star,\ D : \star, f : C \rightarrow D,\ a : C,\ l : List(C) \vdash map(C,\ D,\ f,\ cons(C,\ a,\ l)))$.*

$@(f,\ a)$: *Note that the tuples $(\emptyset,\ f,\ C \rightarrow D)$ and $(\emptyset,\ cons(C,\ a,\ l),\ List(C))$ belong to the initial set. Therefore, $(\emptyset,\ a,\ C)$ belongs to the accessible set. Using **Application**, we get $(\emptyset,\ @(f,a),\ D)$ in the computable closure.*

$\lambda x{:}C.@(f,\ x)$: *The tuples $(x : C,\ x,\ C)$ and $(x : C,\ f,\ C \rightarrow D)$ belong to the computable closure by **Var** and **Weak** rules respectively, since $C : \star,\ D : \star, f : C \rightarrow D,\ a : C,\ l : List(C) \vdash_{CC} C : \star$. Now by the **Application** rule we have*

$$(x : C,\ @(f,\ x),\ D)$$

*and by the **Abstraction** rule*

$$(\emptyset,\ \lambda x{:}C.@(f,\ x),\ C \rightarrow D)$$

*The use of **Abstraction** rule is correct because $(x : C,\ C,\ \star)$ is in the computable closure as a weakening of the initial tuple $(\emptyset,\ C,\ \star)$ and since $C : \star,\ D : \star, f : C \rightarrow D,\ a : C,\ l : List(C), x : C \vdash_{CC} D : \star$.*

3.3 Constructor extensions of HORPO

Definition 3.3.1 (Constructor subterm extension of HORPO) *The judgment* $\Gamma \vdash u \succ^{\mathcal{CS}} w$ *holds if one of the following is true:*

1. $\Gamma \vdash u \succ w \ : T$ *for some* T*, or*

2. u *has a constructor subterm* a ($u \rhd^{\mathcal{CS}} a$) *such that* $\Gamma \vdash a \succeq w \ : T$ *for some* T.

This kind of extension is used to show that $n : Nat \vdash f(s(n)) \succ f(n) \ : Nat$ (because $n : Nat \vdash s(n) \succ^{\mathcal{CS}} n$) where $f : Nat \rightarrow Nat \in \mathcal{F}$ and $s : Nat \rightarrow Nat$ is a constructor of Nat. Recall that we do not have $n : Nat \vdash s(n) \succ n \ : Nat$ since s is a constructor and not a function symbol.

The recursive constructor subterm extension of HORPO, denoted by $\succ^{\mathcal{RCS}}$, is a little more complicated to define than the constructor subterm extension, since it uses a recursive constructor subterm of the left-hand side and a left-flattening of the right-hand side of the judgment. Moreover terms building that left-flattening must belong to the computable closure of the given sequent.

Definition 3.3.2 (Recursive constructor subterm extension) *The judgment* $\Gamma \vdash u(\succ^{\mathcal{RCS}}_{BV, f(\vec{l})})w$ *holds if one of the following is true:*

1. $\Gamma \vdash u \succ w \ : T$ *for some* T*, or*

2. $u \rhd^{\mathcal{RCS}} a$ *for some* a*,* $w = @(v_0, v_1, \ldots v_k)$ *and there exist* $A_1, \ldots A_k, B$ *such that*

   - $\Gamma \vdash a \succeq v_0 \ : \overrightarrow{(z : A)}.B$ *and*
   - $(BV, v_i, A_i[\vec{v}/\vec{z}]) \in CCl(\Gamma \vdash f(\vec{l}))$ *for* $i = 1, \ldots k$.

The recursive constructor extension is necessary if we want to show that $x : Nat \rightarrow Ord \vdash f(lim(x)) \succ f(x\,1) \ : Ord$ where $f : Ord \rightarrow Ord \in \mathcal{F}$ and $lim : (Nat \rightarrow Ord) \rightarrow Ord$ is a constructor of Ord.

In the definition of HORPO, the BV component is always empty and $f(\vec{l})$ is clear from the context, so we simply write $\Gamma \vdash u \succ^{\mathcal{RCS}} w$. On the other hand, in the definition of the computable closure, we use the full notation $\Gamma \vdash u(\succ^{\mathcal{RCS}}_{BV, f(\vec{l})})w$.

In HORPO, apart from $\mathfrak{P}$, constructor extensions $\succ^{\mathcal{CS}}$ and $\succ^{\mathcal{RCS}}$ are used in cases 3 and 4 corresponding to Mul and Lex comparisons respectively.

Note that $\succ^{\mathcal{CS}}$ and $\succ^{\mathcal{RCS}}$ do not subsume each other — on the one hand, every recursive constructor subterm is a constructor subterm, on the other hand $\succ^{\mathcal{RCS}}$ allows to use HORPO and computable closure under the first application symbol on the right. Unfortunately we cannot consider $(\succ^{\mathcal{CS}} \cup \succ^{\mathcal{RCS}})$ for *Lex* or *Mul* since later in the proof we would need to know that every term strongly normalizing in the relation generated by $\succ \cup \rightarrow_\beta$ is also strongly normalizing in the relation generated by $\succ \cup \rightarrow_\beta \cup \rhd^{\mathcal{CS}} \cup \rhd^{\mathcal{RCS}}$ and we cannot prove it. But we can prove this property for $\succ \cup \rightarrow_\beta \cup \rhd^{\mathcal{CS}}$ and $\succ \cup \rightarrow_\beta \cup \rhd^{\mathcal{RCS}}$ separately.

We decided to use $\succ^{\mathcal{RCS}}$ for *Lex* because the terms from the same position of the same function symbol are usually of (almost) the same type. For *Mul* the $\succ^{\mathcal{CS}}$ extension seems more natural.

3.4 Well-definedness of HORPO

As we have seen, HORPO, the computable closure and the constructor extensions of HORPO are mutually dependent. Formally, the mathematical objects corresponding to these definitions are the smallest relations satisfying the given conditions. By Tarski's theorem they exist and can be described as the least fixpoint of a monotone operator constructed directly from the definitions of HORPO, the computable closure and the constructor subterm extensions of HORPO.

We will not present here the details of the above reasoning. Instead, in Section 9.1 we will present a stronger result. It turns out that there is a measure assigned to HORPO, the computable closure and the constructor extensions of HORPO judgments, such that all recursive calls operate on arguments with a smaller measure. This will imply the decidability of HORPO.

3.5 Examples

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

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

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

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

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

For the second rule let us write Γ for the environment $A : \star, B : \star, f : A \to B, a : A, l : List(A)$. 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 proof 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 type B on the left-hand side.

Example 7 *Suppose that $map \in \mathcal{F}^{(2,2)}$ has the same type as in example 6. 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 non-termination, but Section 6.1 shows an example (derived from [46]), where one easily gets a non-terminating term precisely because the condition about the arguments at parameter positions is violated.

It follows that HORPO is not stable under instantiation of big variables:

Example 8 *Assuming $map \in \mathcal{F}^{(2,2)}$ has the same type as in example 6*

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

holds (as proved in Example 6), but

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

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, it is not clear whether the transitive closure is decidable. In practice, as shown in [54], 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:*

$$f : (A : \star)s_1(A) \to (A \to A)$$
$$g : (A : \star)(B : \star)s_2(B) \to (A \to A)$$

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

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

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.

More examples and non-examples can be found in Section 7.2.

Of course the proof of strong normalization (in Section 5) will apply to the rewrite relation generated by HORPO, that is its closure by instantiation and context.

3.6 CC+H — Calculus of constructions with rewriting induced by HORPO

CC+H is the result of mixing together the judgments of the calculus of constructions and those 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 one wants 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.

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

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 from the environment $\Delta_1 = x_1 : A_1 \ldots x_n : A_n$ to the environment Δ_2 is a* well-typed substitution, *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) *The system CC+H is defined as follows:*

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

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

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

$$\textit{(const)}\quad \frac{\Gamma \vdash \star : \Box \quad \forall i = 1, \dots m \quad \Gamma \vdash a_i : b_i[\vec{a}/\vec{x}]}{\Gamma \vdash e(\vec{a}) : c[\vec{a}/\vec{x}]}$$
$$(e : (x_1 : b_1) \dots (x_m : b_m).c \in \Sigma)$$

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

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

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

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

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

In the rules above, $\Gamma \vdash a \to_H b$ stands for the closure of HORPO by instantiation and context. Mentioning the environment Γ is necessary (see Section 2.5.3), so we could also write $a \to_H^\Gamma b$, but not $a \to_H b$. The relation $(\to_\beta \cup \to_H)^\Gamma$ stands for the union of beta and the rewrite relations. For simplicity, we write $\Gamma \vdash a \to b$, or even $a \to^\Gamma b$ for $\Gamma \vdash a(\to_\beta \cup \to_H)b$. We use $\xrightarrow{*}^\Gamma$ for the reflexive and transitive closure of $\to^\Gamma$.

According to the definition of rewriting given at 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 which then arise in the proof of subject reduction. And 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:*

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

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

$$\begin{aligned}
&\Gamma\text{-}\mathit{kind} = \{K \in \mathcal{T} \mid \ \Gamma \vdash K : \Box\}\\
&\Gamma\text{-}\mathit{constr} = \{A \in \mathcal{T} \mid \ \exists K \in \Gamma\text{-}\mathit{kind} \quad \Gamma \vdash A : K\}\\
&\Gamma\text{-}\mathit{type} = \{T \in \mathcal{T} \mid \ \Gamma \vdash T : \star\}\\
&\Gamma\text{-}\mathit{obj} = \{M \in \mathcal{T} \mid \ \exists T \in \Gamma\text{-}\mathit{type} \quad \Gamma \vdash M : T\}
\end{aligned}$$

Note that a Γ-type is a special case of Γ-constr.

By Kind, Constr, Type *and* Obj *we denote the sum for all Γ of Γ-kind, Γ-constr, Γ-type and Γ-obj respectively.*

For the sake of uniformity we will write "a has 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 $\Box$.

Definition 3.6.4 (Well-formed environment) *An environment Γ is* well-formed *if $\Gamma \vdash a : b$ holds for some terms a and b. In the sequel, we will consider only well-formed environments.*

Definition 3.6.5 (Conversion relation) *Terms a and b are* convertible *in the environment Γ, denoted by $\Gamma \vdash a \stackrel{*}{\leftrightarrow} b$ if there is a natural number n and Γ-terms $a_0, a_1, \ldots a_n$ such that:*

- $a = a_0$, $b = a_n$,
- $\Gamma \vdash a_i(\rightarrow_\beta^* \cup \rightarrow_H^*)a_{i+1}$ *or* $\Gamma \vdash a_{i+1}(\rightarrow_\beta^* \cup \rightarrow_H^*)a_i$ *for all* $i = 0 \ldots n-1$.

Summarizing, let us recall the order of major definitions that lead to the system CC+H.

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. $\Gamma \vdash a \rightarrow_H b$ and $\Gamma \vdash a : b$, the mutually defined rewrite relation and typing judgments of CC+H; they use 2 and 3.

Chapter 4

Basic meta-theory

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

The most difficult result in this chapter concerns the subject reduction property of $\to_\beta \cup \to_H$. The usual proof of subject reduction for the calculus of constructions could not be repeated here since it heavily relies on confluence and $\to_H$ is not confluent. Fortunately, a proof not relying on confluence has already been done. In [7] Barbanera, Fernández and Geuvers show that subject reduction holds for the calculus of constructions extended with algebraic rewriting. Although our rewriting is not algebraic, the proof can be easily adapted to our needs. Most of the lemmas from Sections 4.2 and all Section 4.3.1 have the counterparts in [7] and their proofs are strongly inspired by the original ones.

The chapter is organized as follows: in the first section we show some structural properties, like substitutivity, thinning or the generation lemma. The next section contains introductory results needed in the proof of subject reduction. Section 4.3 is devoted to the proof of subject reduction itself. In the last section we describe some properties of conversion that will be directly used in the proof of strong normalization.

4.1 Structural properties

We recall here some well-known structural properties of the calculus of constructions which are still valid in CC+H.

Lemma 4.1.1 (Free variable lemma) *Let* $\Gamma = x_1 : A_1, \ldots x_n : A_n$ *and let* $\Gamma \vdash b : c$*. Then:*

1. $x_1, \ldots x_n$ *are all distinct,*

2. $\mathcal{FV}(b, c) \subseteq \{x_1, \ldots x_n\}$,

3. $\mathcal{FV}(A_i) \subseteq \{x_1, \ldots x_{i-1}\}$ *for* $i = 1 \ldots n$.

Proof
By induction of the derivation $\Gamma \vdash b : c$. $\square$

Lemma 4.1.2 (Start lemma) *Let* Γ *be an environment and* a, b *terms. If* $\Gamma \vdash a : b$ *then* $\Gamma \vdash \star : \square$.

Proof
By induction on the derivation of $\Gamma \vdash a : b$. We treat two typical cases.

1. The last rule in the derivation was **(weak)**

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

   We have to show that $\Gamma, x : c \vdash \star : \square$. By induction hypothesis $\Gamma \vdash \star : \square$. Since we have $\Gamma \vdash c : p$ we may apply **(weak)** rule and get $\Gamma, x : c \vdash \star : \square$.

2. The last rule in the derivation was **(app)**

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

   by induction hypothesis we get immediately $\Gamma \vdash \star : \square$. $\square$

Lemma 4.1.3 (Substitution lemma) *Let* $\Gamma_1 \vdash d : a$. *Then*

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

Moreover the same holds for $\to_\beta$ *instead of* $\to_H$.

Proof
By mutual induction on the derivation of $\Gamma_1, x : a, \Gamma_2 \vdash b : c$ and $\Gamma_1, x : a, \Gamma_2 \vdash b \to_H b'$. We treat two typical cases for $\Gamma_1, x : a, \Gamma_2 \vdash b : c$ and the case of beta and rewrite reduction in b.

1. The last rule in the derivation was **(var)** rule.

   (a) Γ_2 is empty and the last step in the derivation was

   $$\frac{\Gamma_1 \vdash a \,:\, p}{\Gamma_1, x : a \vdash x \,:\, a} \qquad (x \in \mathcal{V}ar^p \setminus dom(\Gamma_1), p \in \{\star, \Box\})$$

   We have to show that $\Gamma_1 \vdash x[d/x] \,:\, a[d/x]$, but this is simply the hypothesis of the lemma.

   (b) Γ_2 is not empty and the last step was

   $$\frac{\Gamma_1, x : a, \Gamma_2 \vdash e \,:\, p}{\Gamma_1, x : a, \Gamma_2, y : e \vdash y \,:\, e} \ (y \in \mathcal{V}ar^p \backslash dom(\Gamma_1, x : a, \Gamma_2), p \in \{\star, \Box\})$$

   We have to show that $\Gamma_1, \Gamma_2[d/x], y : e[d/x] \vdash y \,:\, e[d/x]$, but this follows directly from $\Gamma_1, \Gamma_2[d/x] \vdash e[d/x] \,:\, p$, which is true by induction hypothesis.

2. The last rule in the derivation was **(app)** rule

   $$\frac{\Gamma_1, x : a, \Gamma_2 \vdash b_1 \,:\, (y{:}c_1).c_2 \qquad \Gamma_1, x : a, \Gamma_2 \vdash b_2 \,:\, c_1}{\Gamma_1, x : a, \Gamma_2 \vdash b_1\, b_2 \,:\, c_2[b_2/y]}$$

   By the induction hypothesis we have

   $$\Gamma_1, \Gamma_2[d/x] \vdash b_1[d/x] \,:\, ((y{:}c_1).c_2)[d/x]$$
   $$\Gamma_1, \Gamma_2[d/x] \vdash b_2[d/x] \,:\, c_1[d/x]$$

   and by **(app)** rule

   $$\Gamma_1, \Gamma_2[d/x] \vdash b_1[d/x]\, b_2[d/x] \,:\, c_2[d/x][b_2[d/x]/y]$$

   Since $c_2[d/x][b_2[d/x]/y] = (c_2[b_2/y])[d/x]$ (because $y \notin \mathcal{FV}(d)$ and x, y are different variables) we get the desired conclusion.

3. For the case of reduction, let us start with $\to_H$. The judgment $\Gamma_1, x : a, \Gamma_2 \vdash b \to_H b'$ implies that there is a HORPO judgment $G \vdash l \succ r \,:\, T$, a position p and a well-typed substitution $\theta : G \to \Gamma_1, x : a, \Gamma_2, \mathcal{PV}(b, p)$ such that $b|_p = l\theta$. To check that $\Gamma_1, \Gamma_2[d/x] \vdash b[d/x] \to_H b'[d/x]$ it is sufficient to note that $b[d/x] = b[d/x][l\theta[d/x]]_p$ and $b'[d/x] = b'[d/x][r\theta[d/x]]_p$ and then show that the sequential

composition of θ and $[d/x]$ is a well-typed substitution from G to $\Gamma_1, \Gamma_2[d/x], \mathcal{PV}(b[d/x], p)$. But for every $z_i : A_i$ in G, we have

$$\Gamma_1, \Gamma_2[d/x], \mathcal{PV}(b[d/x], p) \vdash z_i\theta[d/x] \,:\, A_i\theta[d/x]$$

by induction hypothesis applied to $\Gamma_1, x : a, \Gamma_2, \mathcal{PV}(b, p) \vdash z_i\theta \,:\, A_i\theta$.

For $\rightarrow_\beta$ the proof follows the same line. Note that $\Gamma_1, x : a, \Gamma_2 \vdash b \rightarrow_\beta b'$ implies that there is a position p such that $b|_p = (\lambda y : u.v)\, w$ and $b' = a[v[w/y]]_p$. Of course, we have $(b[d/x])|_p = (\lambda y : u[d/x].v[d/x])\, w[d/x]$ and $b'[d/x] = b[d/x][v[d/x][w[d/x]/y]]_p$. Consequently $\Gamma_1, \Gamma_2[d/x] \vdash b[d/x] \rightarrow_\beta b'[d/x]$. $\Box$

Lemma 4.1.4 *For all environments Γ_1, Γ_2, Γ_3 such that $dom(\Gamma_2) \cap dom(\Gamma_3) = \emptyset$ and $\Gamma_1, \Gamma_2 \vdash \star \,:\, \Box$ and all terms a, b, a'*

$$\begin{array}{rcl} \Gamma_1, \Gamma_3 \vdash a \,:\, b & \textit{implies} & \Gamma_1, \Gamma_2, \Gamma_3 \vdash a \,:\, b \\ \Gamma_1, \Gamma_3 \vdash a \rightarrow_H a' & \textit{implies} & \Gamma_1, \Gamma_2, \Gamma_3 \vdash a \rightarrow_H a' \end{array}$$

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

Proof
By mutual induction on the derivation of $\Gamma_1, \Gamma_3 \vdash a \,:\, b$ and $\Gamma_1, \Gamma_3 \vdash a \rightarrow_H a'$. We treat two typical cases for $\Gamma_1, \Gamma_3 \vdash a \,:\, b$ and the case of reduction:

1. The last rule in the derivation was the **(var)** rule

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

   We have to consider two subcases:

   (a) $\Gamma_1 = \Gamma, x : a$, $\Gamma_3 = \emptyset$. We want to show that $\Gamma, x : a, \Gamma_2 \vdash x \,:\, a$. To this end we have to weaken the judgment $\Gamma, x : a \vdash x \,:\, a$ with all variables from $\Gamma_2 = y_1 : b_1, \ldots y_n : b_n$. Since $\Gamma_1, \Gamma_2 \vdash \star \,:\, \Box$ we have $\Gamma, y_1 : b_1, \ldots y_i : b_i \vdash b_{i+1} \,:\, p_{i+1}$ with $p_{i+1} \in \{\star, \Box\}$ for all i. By induction hypothesis we get $\Gamma, x : a, y_1 : b_1, \ldots y_i : b_i \vdash b_{i+1} \,:\, p_{i+1}$ for all i. From $\Gamma, x : a \vdash x \,:\, a$ and $\Gamma, x : a \vdash b_1 \,:\, p_1$ we obtain $\Gamma, x : a, y_1 : b_1 \vdash x \,:\, a$ by the **(weak)** rule. Repeating this step n times we finally get $\Gamma, x : a, \Gamma_2 \vdash x \,:\, a$.

(b) $\Gamma_1 = \Gamma'$, $\Gamma_3 = \Gamma'', x : a$ with $\Gamma = \Gamma', \Gamma''$. We want to show that $\Gamma', \Gamma_2, \Gamma'', x : a \vdash x : a$. By induction hypothesis applied to $\Gamma \vdash a : p$ and $\Gamma, \Gamma_2 \vdash \star : \Box$ we get $\Gamma', \Gamma_2, \Gamma'' \vdash a : p$. Since $x \notin dom(\Gamma', \Gamma_2, \Gamma'')$ (recall that $\Gamma_2 \cap \Gamma_3 = \emptyset$ and $x \in dom(\Gamma_3)$) we may use the **(weak)** rule and get the desired conclusion $\Gamma', \Gamma_2, \Gamma'', x : a \vdash x : a$.

2. The last rule in the derivation was the **(prod)** rule

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

Let $\Gamma = \Gamma_1, \Gamma_3$. By induction hypothesis we have $\Gamma_1, \Gamma_2, \Gamma_3 \vdash a : p$ and $\Gamma_1, \Gamma_2, \Gamma_3, x : a \vdash b : q$ (since it may be assumed that $x \notin \Gamma_2$). Now it is sufficient to apply **(prod)** to get $\Gamma_1, \Gamma_2, \Gamma_3 \vdash (x{:}a).b : q$.

3. For the case of reduction, the judgment $\Gamma_1, \Gamma_3 \vdash a \to_H a'$ implies that there is a HORPO judgment $G \vdash l \succ r : T$, a position p and a well-typed substitution $\theta : G \to \Gamma, \mathcal{PV}(a, p)$ such that $a_p = l\theta$. To check that $\Gamma_1, \Gamma_2, \Gamma_3 \vdash a \to_H a'$ it is sufficient to check that θ is also a well-typed substitution from G to $\Gamma_1, \Gamma_2, \Gamma_3, \mathcal{PV}(a, p)$. But this is true, since for every $z_i : A_i$ in G, $\Gamma_1, \Gamma_2, \Gamma_3 \vdash z_i\theta : A_i\theta$ follows by induction hypothesis from $\Gamma_1, \Gamma_3 \vdash z_i\theta : A_i\theta$.

For $\to_\beta$ the statement of the lemma is obviously true. $\Box$

Lemma 4.1.5 (Thinning lemma) *Let Γ and Γ' be environments such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash \star : \Box$. Then for all a, b,*

$$\Gamma \vdash a : b \quad \textit{implies} \quad \Gamma' \vdash a : b$$
$$\Gamma \vdash a \to_H a' \quad \textit{implies} \quad \Gamma' \vdash a \to_H a'$$

Moreover the same holds for $\to_\beta$ instead of $\to_H$.

Proof
It follows easily ($\Gamma = \Gamma_1$, $\Gamma' = (\Gamma_1, \Gamma_2)$, $\Gamma_3 = \emptyset$) from the previous lemma.
$\Box$

The following corollary is an easy consequence of Substitution and Thinning.

Corollary 4.1.6 *If $\Gamma \vdash b : c$ and $\theta : \Gamma \to \Delta$ then $\Delta \vdash b\theta : c\theta$.*

Lemma 4.1.7 *Let Γ_1, Γ_2 and Δ be environments. If $\rho_1 : \Gamma_1 \to \Delta$ and $\rho_2 : \Gamma_2 \to \Delta$ 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 $\rho_1 : \Gamma_1 \to \Gamma_2$ and $\rho_2 : \Gamma_2 \to \Delta$ are well-typed substitution then their sequential composition $\rho_1\rho_2$ is a well-typed substitution from Γ_1 to Δ.

Proof
For the parallel composition it is sufficient to show that $\Delta \vdash x(\rho_1 \cup \rho_2) : A(\rho_1 \cup \rho_2)$ for every $x : A \in (\Gamma_1 \cup \Gamma_2)$. Since Γ_1 and Γ_2 are disjoint we can assume that $x : A \in \Gamma_1$ without loss of generality. But then $x(\rho_1 \cup \rho_2) = x\rho_1$ and $A(\rho_1 \cup \rho_2) = A\rho_1$ and we conclude using the hypothesis that $\rho_1 : \Gamma_1 \to \Delta$.

For the sequential composition one has to show that $\Delta \vdash x(\rho_1\rho_2) : A(\rho_1\rho_2)$ for every $x : A \in \Gamma_1$. By assumption we have $\Gamma_2 \vdash x\rho_1 : A\rho_1$. By Corollary 4.1.6 we get the conclusion. $\square$

Lemma 4.1.8 (Generation lemma) *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 $a = f(a_1, \ldots a_n)$ then
  $f : (x_1 : b_1) \ldots (x_n : b_n).c \in \mathcal{F}^n$,
  $\forall i \quad \Gamma \vdash a_i : b_i[a_1/x_1, \ldots, a_{i-1}/x_{i-1}]$,
  $\Gamma \vdash b \overset{*}{\leftrightarrow} c[a_1/x_1, \ldots, a_n/x_n]$
- if $a = s(a_1, \ldots a_r)$ then
  $s : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).\star \in \mathcal{TC}^r$,
  $\forall i \quad \Gamma \vdash a_i : P_i[a_1/p_1, \ldots, a_{i-1}/p_{i-1}]$,
  $\Gamma \vdash b \overset{*}{\leftrightarrow} ((x_1 : b_1) \ldots (x_n : b_n).\star)[\vec{a}/\vec{p}]$,
- if $a = c(a_1, \ldots a_r, a'_1, \ldots a'_n)$ then
  $c : (p_1 : P_1) \ldots (p_r : P_r).(x_1 : b_1) \ldots (x_n : b_n).s(\vec{p})\vec{w} \in \mathcal{CS}^{(r,n)}$
  $s : (p_1 : P_1) \ldots (p_r : P_r).(y_1 : d_1) \ldots (y_m : d_m).\star \in \mathcal{TC}^r$,
  $\forall i \quad \Gamma \vdash a_i : P_i[a_1/p_1, \ldots, a_{i-1}/p_{i-1}]$,
  $\forall i \quad \Gamma \vdash a'_i : b_i[a_1/p_1, \ldots, a_r/p_r][a'_1/x_1, \ldots, a'_{i-1}/x_{i-1}]$,
  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
  $$\exists c \;\; \Gamma, x : a_1 \vdash a_2 \;:\; c, \quad \Gamma \vdash (x\!:\!a_1).c \;:\; p,$$
  $$p \in \{\star, \Box\}, \quad \Gamma \vdash b \overset{*}{\leftrightarrow} (x\!:\!a_1).c$$

- if $a = (x\!:\!a_1).a_2$ then
  $$\Gamma \vdash a_1 \;:\; p_1, \quad \Gamma, x : a_1 \vdash a_2 \;:\; p_2, \quad p_1, p_2 \in \{\star, \Box\},$$
  $$\Gamma \vdash b \overset{*}{\leftrightarrow} p_2$$

- if $a = a_1\, a_2$ then
  $$\exists c, d \;\; \Gamma \vdash a_1 \;:\; (x\!:\!c).d, \quad \Gamma \vdash a_2 \;:\; c, \quad \Gamma \vdash b \overset{*}{\leftrightarrow} d[a_2/x]$$

Proof
Consider a derivation of $\Gamma \vdash a \;:\; b$. The rules of conversion and weakening do not change the term a. We can follow the branch of the derivation until the term a is introduced for the first time. This can only be done by the typing rule corresponding to the form of a, for example if $a = \lambda x\!:\!a_1.a_2$ the rule is **(abs)**.

In each case the conclusion of the rule is $\Gamma' \vdash a \;:\; b'$ with $\Gamma' \subseteq \Gamma$ and $\Gamma \vdash b \overset{*}{\leftrightarrow} b'$. The statement of the lemma follows by inspection of the used rule and the Thinning Lemma. $\Box$

4.2 Auxiliary lemmas for subject reduction

This section contains proofs of technical properties of conversion that will be used in the proof of subject reduction in the next section.

Lemma 4.2.1 *Let $\Gamma \vdash a \;:\; b$. Then there is $p \in \{\star, \Box\}$, such that $b = p$ or $\Gamma \vdash b \;:\; p$.*

Proof
By induction on the derivation of $\Gamma \vdash a \;:\; b$. The most interesting case is when the last rule is **(app)**.

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

Then $\Gamma \vdash (x\!:\!b).c \;:\; p$ by induction, and hence by Generation (Lemma 4.1.8), $\Gamma, x : b \vdash c \;:\; p'$ for some $p' \in \{\star, \Box\}$. Now by Substitution Lemma 4.1.3, we conclude that $\Gamma \vdash c[d/x] \;:\; p'$. $\Box$

Lemma 4.2.2 *Let $\Gamma \vdash a : b$. If c is a subterm of a then there is Γ' and d such that $\Gamma' \vdash c : d$.*

Proof
By induction on the structure of a, using the Generation Lemma. $\square$

Lemma 4.2.3 *Let Γ be an environment, $p \in \{\star, \square\}$ and M be a term such that $\Gamma \vdash M \overset{*}{\leftrightarrow} p$. Then $M = p$.*

Proof
Suppose that $M \neq p$. Then there is an $N \neq p$ such that $\Gamma \vdash N \overset{*}{\rightarrow}_H p$ or $N \overset{*}{\rightarrow}_\beta p$. In case of $\rightarrow_H$ reduction let us also consider N' such that $\Gamma \vdash N \overset{*}{\rightarrow}_H N' \rightarrow_H p$

If $p = \square$ the first case is not possible, since it would imply that there is a judgment $G \vdash l \succ r \ : T$ and a substitution θ such that $l\theta = N'$ and $r\theta = \square$. This would mean that $\Gamma \vdash \square : T$ which is not possible. In case of beta reduction $N = B\,C$ with $\square$ occurring as subterm of B or C. But every subterm has a type and $\square$ does not have one. Contradiction.

If $p = \star$ then $\Gamma \vdash N' \rightarrow_H \star$ implies $\Gamma \vdash \star : T$ with $\Gamma \vdash T : \star$. This is not possible, since by Generation $\Gamma \vdash T \overset{*}{\leftrightarrow} \square$, by the above reasoning $T = \square$ and hence $\Gamma \vdash \square : \star$. If we suppose that $N \overset{*}{\rightarrow}_\beta \star$ then $N = B\,C$ with $\star$ occurring as subterm of B or C in applied ($\star P$ or $P\vec{Q}\star$) or abstracted form ($\lambda\vec{y}\!:\!\vec{A}.\star$). In either case we find by Generation that $\square$ has a type, which is not possible. $\square$

Lemma 4.2.4 *Let Γ be an environment and $\overrightarrow{(x : A)}.\star$ be a Γ-term. Then $\Gamma \vdash \overrightarrow{(x : A)}.\star : \star$ is not possible.*

Proof
By Generation Lemma there must be a $p \in \{\star, \square\}$ such that $\Gamma, \overrightarrow{x : A} \vdash \star : p$ and $\Gamma, \overrightarrow{x : A} \vdash \star \overset{*}{\leftrightarrow} p$. By Lemma 4.2.3 $p = \star$, which implies $\Gamma, \overrightarrow{x : A} \vdash \star : \star$. By Generation Lemma we have $\Gamma \vdash \star \overset{*}{\leftrightarrow} \square$ but this is impossible by Lemma 4.2.3. $\square$

Lemma 4.2.5 *Let Γ be an environment and M be a Γ-term. We have $M = \overrightarrow{(x : A)}.\star$ for some $\vec{A}$ if and only if $\Gamma \vdash M : \square$.*

Proof
Suppose that $M = \overrightarrow{(x:A)}.\star$ and $\Gamma \vdash M : T$ for some T. By Generation Lemma there is a $p \in \{\star, \Box\}$ such that $\Gamma \vdash T \overset{*}{\leftrightarrow} p$. By the Lemma 4.2.3 $T = p$. From Lemma 4.2.4 $p = \Box$.

Suppose now that $\Gamma \vdash M : \Box$. We consider all possible forms of M. It can be neither a variable, nor a constant-headed term, nor an abstraction. In fact $\Gamma \vdash P\,Q : \Box$ is also impossible because otherwise $\Box$ would be a subterm of the well-typed term, which would imply that $\Box$ has a type. Of course $M = \star$ satisfies the statement of the lemma. The only possibility left is $M = \overrightarrow{(x:C)}.D$ with D that is not a product. If $\overrightarrow{(x:C)} = \emptyset$ then D must be a $\star$. Otherwise, by Generation Lemma $\Gamma, \overrightarrow{x:C} \vdash D : p$ with $\Gamma \vdash p \overset{*}{\leftrightarrow} \Box$. By Lemma 4.2.3 $p = \Box$ and consequently $D = \star$. $\Box$

Lemma 4.2.6 *Let Γ be an environment and $\overrightarrow{(x:A)}.\star$ and Q be Γ-terms. Assuming that $\Gamma \vdash \overrightarrow{(x:A)}.\star : \Box$ we have:*

1. *If $\Gamma \vdash \overrightarrow{(x:A)}.\star(\overset{*}{\rightarrow}_\beta \cup \overset{*}{\rightarrow}_H)Q$ then $Q = \overrightarrow{(x:A')}.\star$.*
2. *If $\Gamma \vdash Q(\overset{*}{\rightarrow}_\beta \cup \overset{*}{\rightarrow}_H)\overrightarrow{(x:A)}.\star$ then $Q = \overrightarrow{(x:A')}.\star$.*

Proof
The statement 1 is quite obvious since neither β-reduction, nor rewriting can destroy the head product.

For 2, suppose first that Q β-reduces to $\overrightarrow{(x:A)}.\star$. Let $Q = (\lambda\vec{y}{:}\vec{B}.P)\,\vec{Q}$. Then either P or $\vec{Q}$ contain a subterm of the form $\overrightarrow{(z:C)}.\star$. We even know that this subterm occurs as $\lambda q{:}R.\overrightarrow{(z:D)}.\star$ or as $R(\overrightarrow{(z:D)}.\star)$. But neither is possible for typing reasons.

If Q rewrites to $\overrightarrow{(x:A)}.\star$ and the reduction is nontrivial then let us examine Q', the last term before $\overrightarrow{(x:A)}.\star$ ($\Gamma \vdash Q \overset{*}{\rightarrow}_H Q' \rightarrow_H \overrightarrow{(x:A)}.\star$). There is a HORPO judgment $G \vdash l \succ r : T$ and a substitution θ, such that $Q' = l\theta$ and $\overrightarrow{(x:A)}.\star = r\theta$. This immediately eliminates all cases from Part II of HORPO except the second one (for the product). But in this case Q' is a product $\overrightarrow{(x:A')}.\star$ and it is well-typed ($\Gamma \vdash Q' : T\theta$), so we conclude by induction hypothesis. If we suppose that $G \vdash l \succ r : T$ comes from Part I then $G \vdash r : T : \star$ and we get $\Gamma \vdash \overrightarrow{(x:A)}.\star : T\theta : \star$. By Generation there is $p \in \{\star, \Box\}$ such that $\Gamma \vdash p \overset{*}{\leftrightarrow} T\theta$. By Lemma 4.2.3 $T\theta = p$ and we have $\Gamma \vdash p : \star$ which is not possible ($\Box$ does not have a type and $\Gamma \vdash \star : \star$ is not possible by Lemma 4.2.4). □

Corollary 4.2.7 *Let Γ be an environment and M, N be Γ-terms. If $\Gamma \vdash M : \Box$ and $\Gamma \vdash M \overset{*}{\leftrightarrow} N$ then $\Gamma \vdash N : \Box$.*

Proof
Let $\Gamma \vdash M = M_1(\rightarrow_\beta^* \cup \rightarrow_H^*)M_2({}_H\overset{*}{\leftarrow} \cup {}_\beta\overset{*}{\leftarrow})\ldots(\rightarrow_\beta^* \cup \rightarrow_H^*)M_n = N$ be the conversion sequence between M and N. Note that all M_i are well-typed. Moreover by Lemma 4.2.5 $M = \overrightarrow{(x:A)}.\star$. By induction on n we can show that $M_i = \overrightarrow{(x:A^i)}.\star$ (by Lemma 4.2.6) and that $\Gamma \vdash M_i : \Box$ (by Lemma 4.2.5). This finishes the proof, as $N = M_n$. □

Lemma 4.2.8 *Let Γ be an environment and M, N be Γ-terms such that $\Gamma \vdash M : p$ and $\Gamma \vdash N : p'$ with $p, p' \in \{\star, \Box\}$. If $\Gamma \vdash M \overset{*}{\leftrightarrow} N$ then $p = p'$*

Proof
Suppose that $\Gamma \vdash M : \Box$. We will show that $p' = \Box$.

By Lemma 4.2.7 $\Gamma \vdash N : \Box$ and by Lemma 4.2.5 $N = \overrightarrow{(x:A)}.\star$. If $p' = \star$ then we get $\Gamma \vdash \overrightarrow{(x:A)}.\star : \star$ which is not possible by Lemma 4.2.4. □

Lemma 4.2.9 (Classification lemma) *In CC+H we have*

$$\Gamma\text{-}kind \cap \Gamma\text{-}type = \Gamma\text{-}constr \cap \Gamma\text{-}obj = \emptyset$$

Proof
For the former it is sufficient to prove the following:

$$\Gamma \vdash M : p \quad \text{and} \quad \Gamma \vdash M : p' \quad \text{imply} \quad p = p'$$

which is true by Lemma 4.2.8. For the latter we have to show

$$\Gamma \vdash M : B : p \quad \text{and} \quad \Gamma \vdash M : B' : p' \quad \text{imply} \quad p = p'$$

It can be shown by induction on the structure of M using Generation Lemma and Lemma 4.2.8. We do the proof only for two typical cases:

- if $M = x$ and $x \in \mathcal{V}ar^{p_0}$ then by Generation there is A such that $x : A \in \Gamma$, $\Gamma \vdash A : p_0$ and $\Gamma \vdash A \overset{*}{\leftrightarrow} B$. By Lemma 4.2.8 $p_0 = p$. Similarly we get $p_0 = p'$. Hence $p = p'$.

- if $M = a\,b$ then by Generation $\Gamma \vdash a : (x:c).d : p_1$ and $\Gamma \vdash b : c$ for some c and d such that $\Gamma \vdash B \overset{*}{\leftrightarrow} d[b/x]$. Again by Generation we have $\Gamma \vdash c : p_2$ and $\Gamma, x : c \vdash d : p_1$. By substitution $\Gamma \vdash d[b/x] : p_1$. By Lemma 4.2.8 $\Gamma \vdash B : p_1$ and $p_1 = p$. If we do the same for B' we find that $\Gamma \vdash a : (x:c').d' : p_1'$, $\Gamma \vdash b : c'$ and $p_1' = p'$. But by induction hypothesis $p_1 = p_1'$. $\square$

The following two definitions are used locally in this section, to prove Lemma 4.2.14.

Definition 4.2.10 (Ordering on paths) *Recall that paths are sequences of natural numbers. The ordering $<_{path}$ is defined as the lexicographic extension of the ordering $<$ on natural numbers:*

$$n_1 \cdot \ldots \cdot n_k <_{path} m_1 \cdot \ldots \cdot m_l \quad \textit{if and only if}$$

- *$k < l$ and for all $i = 1, \ldots k$, $n_i = m_i$, or*
- *there exists $i \in \{1, \ldots k\}$ such that $n_i < m_i$ and for all $j < i$ $n_j = m_j$.*

In the first case we say that $n_1 \cdot \ldots \cdot n_k$ is a prefix *of $m_1 \cdot \ldots \cdot m_l$.*

Of course, $p \leq_{path} q$ if and only if $p <_{path} q$ or $p = q$.

It is well-known that $\leq_{path}$ is a total ordering on paths.

Definition 4.2.11 *Let Γ be an environment, b a term and q a path in b. We say that b is* q-well-typed in Γ *if $\Gamma, \mathcal{PV}(b,q) \vdash \star : \Box$ holds.*

If b is p-well-typed for every path p such that $p \leq_{path} q$ then b is called up-to-q-well-typed.

Finally, if b is p-well-typed for every path p such that

- *$p \leq_{path} q$, or*
- *q is a prefix of p.*

then b is called totally-q-well-typed.

Lemma 4.2.12 *The following properties hold:*

1. *If $\Gamma \vdash b : T$ then b is q-well-typed in Γ for any path q in b. Consequently, b is also up-to-q-well-typed and totally-q-well-typed for any q in b.*

2. *If b is up-to-q-well-typed in Γ then it is totally-p-well-typed for every $p <_{path} q$ such that p is not a prefix of q.*

3. *If b is totally-q-well-typed in Γ, $b = b[c]_q$ and $\Gamma, PV(b,q) \vdash c \rightarrow_H c'$ then $b[c']_q$ is up-to-q-well-typed.*

Proof
Obvious. $\square$

Lemma 4.2.13 *Let b be a totally-q-well-typed term in Γ. If $b = b[c]_q$ and $\Delta \vdash c \rightarrow_H c'$ with $\Delta \subseteq \Gamma, \mathcal{PV}(b,q)$ then $\Gamma \vdash b[c]_q \rightarrow_H b[c']_q$ and $b[c']_q$ is up-to-q-well-typed. Moreover, if $\Delta \vdash c \overset{\Lambda}{\rightarrow}_H c'$ (the rewriting is in the root) then $b[c']_q$ is totally-q-well-typed.*

Proof
The judgment $\Delta \vdash c \rightarrow_H c'$ implies that there is a HORPO judgment $G \vdash l \succ r : T$, a position p and a well-typed substitution $\theta : G \rightarrow \Delta, \mathcal{PV}(c,p)$ such that $c = c[l\theta]_p$ and $c' = c[r\theta]_p$.

To show that $\Gamma \vdash b[c]_q \rightarrow_H b[c']_q$ we will use the same HORPO judgment $G \vdash l \succ r : T$ and the substitution θ. Let us show that $\theta : G \rightarrow \Gamma, \mathcal{PV}(b, q \cdot p)$. First note that $\mathcal{PV}(b, q \cdot p) = \mathcal{PV}(b,q), \mathcal{PV}(c,p)$ and $\Gamma, \mathcal{PV}(b,q) = \Delta, \Delta'$ for some Δ' (by assumption that $\Delta \subseteq \Gamma, \mathcal{PV}(b,q)$). Consequently, we have to check whether $\theta : G \rightarrow \Delta, \Delta', \mathcal{PV}(c,p)$, i.e. whether $\Delta, \Delta', \mathcal{PV}(c,p) \vdash x_i\theta : A_i\theta$ for all $x_i : A_i \in G$. But this follows by Lemma 4.1.4 applied to $\Delta, \mathcal{PV}(c,p) \vdash x_i\theta : A_i\theta$ (which is true since $\theta : G \rightarrow \Delta, PV(c,p)$) and $\Delta, \Delta' \vdash \star : \square$.

The term $b[c']_q$ is up-to-q-well-typed by case 3 of Lemma 4.2.12.

Let us show that $b[c']_q$ is totally-q-well-typed if $\Delta \vdash c \overset{\Lambda}{\rightarrow}_H c'$. Of course, $\mathcal{PV}(b[c']_q, q \cdot q') = \mathcal{PV}(b,q), \mathcal{PV}(r\theta, q')$ and we have to show that $\Gamma, \mathcal{PV}(b,q), \mathcal{PV}(r\theta, q') \vdash \star : \square$ for every path q' in $r\theta$. But $r\theta$ is a well-typed term ($\Delta \vdash r\theta : T\theta$) and consequently, by case 1 of Lemma 4.2.12, $\Delta, \mathcal{PV}(r\theta, q') \vdash \star : \square$ for every q'. To conclude, it is sufficient to note that $\Gamma, \mathcal{PV}(b,q) = \Delta, \Delta'$ and use Lemma 4.1.4. $\square$

Lemma 4.2.14 *Let $\Gamma, x : a \vdash b : T$, $\Gamma, x : a \vdash b \rightarrow_H b'$, $\Gamma \vdash c : a$ and $\Gamma \vdash c \rightarrow_H c'$. Then $\Gamma \vdash b[c/x] \overset{*}{\rightarrow}_H b'[c'/x]$.*

Proof
The rewriting $\Gamma, x : a \vdash b \rightarrow_H b'$ implies that there is a HORPO judgment

$G \vdash l \succ r \ : T$, a position q and a well-typed substitution $\theta : G \to \Gamma, x : a, \mathcal{PV}(b, q)$ such that $b = b[l\theta]_q$ and $b' = b[r\theta]_q$.

Let $p_1, \ldots p_n$ be a lexicographically sorted sequence of occurrences of x in b. Note that none of $p_1, \ldots p_n$ is a proper prefix/suffix of another position from $p_1, \ldots p_n$. Moreover, let us suppose that q is the prefix of the positions $p_{i+1}, \ldots p_{j-1}$.

Let us denote $b[c/x]$ by d. By Substitution (Lemma 4.1.3) $\Gamma \vdash d : T[c/x]$ and by case 1 of Lemma 4.2.12 d is totally-p_n-well-typed. By Lemma 4.2.13 $\Gamma \vdash d \to_H d[c']_{p_n}$ and $d[c']_{p_n}$ is totally-p_{n-1}-well-typed (since $p_{n-1} <_{path} p_n$). We can continue like this up to p_j. Hence

$$\Gamma \vdash d \to_H d[c']_{p_n} \to_H \ldots \to_H d[c']_{p_n}[c']_{p_{n-1}} \ldots [c']_{p_j}$$

Let us denote $d[c']_{p_n}[c']_{p_{n-1}} \ldots [c']_{p_j}$ by d'. Of course it is totally-q-well-typed and $d'|_q = b[c/x]|_q = l\theta[c/x]$. Since $\Gamma, \mathcal{PV}(b[c/x], q) \vdash l\theta[c/x] \xrightarrow{\Lambda}_H r\theta[c/x]$, by Lemma 4.2.13 we get

$$\Gamma \vdash d' \to_H d'[r\theta[c/x]]_q$$

and $d'[r\theta[c/x]]_q$ is a totally-q-well-typed

Let us denote $d'[r\theta[c/x]]_q$ by d'' and let $q_1, \ldots q_m$ be a lexicographically sorted sequence of occurrence of x in $r\theta$. Of course d'' is totally-qq_m-well-typed. Like before, by Lemma 4.2.13 we have

$$\Gamma \vdash d'' \to_H d''[c']_{q \cdot q_m} \to_H \ldots \to_H d''[c']_{q \cdot q_m}[c']_{q \cdot q_{m-1}} \ldots [c']_{q \cdot q_1}$$

Let us denote $d''[c']_{q \cdot q_m}[c']_{q \cdot q_{m-1}} \ldots [c']_{q \cdot q_1}$ by d'''. Note that $d''' = d'[r\theta[c'/x]]_q$ and that it is up-to-q-well-typed. Consequently, it is totally-p_i-well-typed and we can continue to rewrite at $p_{i-1}, \ldots p_1$ resulting in

$$\Gamma \vdash d''' \to_H d'''[c']_{p_i} \to_H \ldots \to_H d'''[c']_{p_i}[c']_{p_{i-1}} \ldots [c']_{p_1} = b'[c'/x] \quad \square$$

Definition 4.2.15 (Rewriting and conversion of environments) *Let $\Gamma = x_1 : A_1 \ldots x_n : A_n$ and $\Gamma' = x_1 : A'_1 \ldots x_n : A'_n$ be environments. Then*

1. *$\Gamma \to \Gamma'$ if there is an i such that $\overrightarrow{x : A} \vdash A_i \to A'_i$ and $A_j = A'_j$ for $j \neq i$.*
2. *$\Gamma \overset{*}{\Leftrightarrow}_1 \Gamma'$ if there is an i such that $\overrightarrow{x : A} \vdash A_i (\overset{*}{\to} \cup \overset{*}{\leftarrow})^* A'_i$ and $A_j = A'_j$ for $j \neq i$.*

3. *$\Gamma \overset{*}{\leftrightarrow}_1 \Gamma'$ if there is an i such that $\overrightarrow{x : A} \vdash A_i \overset{*}{\leftrightarrow} A'_i$ and $A_j = A'_j$ for $j \neq i$.*

4. *$\Gamma \overset{*}{\leftrightarrow} \Gamma'$ is a transitive closure of $\Gamma \overset{*}{\leftrightarrow}_1 \Gamma'$.*

The difference between $\Gamma \overset{*}{\Leftrightarrow}_1 \Gamma'$ and $\Gamma \overset{*}{\leftrightarrow}_1 \Gamma'$ concerns the well-typedness of the sequence of reductions between A_i and A'_i. In $\Gamma \overset{*}{\Leftrightarrow}_1 \Gamma'$ we know nothing about types of the terms on this sequence, but in $\Gamma \overset{*}{\leftrightarrow}_1 \Gamma'$ we know that A_i and A'_i are convertible (there are peaks and valleys that are well-typed terms in the environment $\overrightarrow{x : A}$).

Lemma 4.2.16 *Let Γ, Γ' be environments such that $\Gamma \vdash \star : \Box$, $\Gamma' \vdash \star : \Box$ and $\Gamma \overset{*}{\Leftrightarrow}_1 \Gamma'$ and let a, b be terms.*

1. *If $\Gamma \vdash a : b$ then $\Gamma' \vdash a : b$.*

2. *If $\Gamma \vdash a \to_H b$ then $\Gamma' \vdash a \to_H b$.*

Moreover the same holds for $\to_\beta$ instead of $\to_H$.

Proof
By mutual induction on the derivations of $\Gamma \vdash a : b$ and $\Gamma \vdash a \to b$. We treat below three typical cases.

1. The last rule in the derivation of $\Gamma \vdash a : b$ was **(var)**

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

   According to the definition of rewriting of environments we distinguish two cases:

   (a) $\Gamma, x : c \overset{*}{\Leftrightarrow}_1 \Gamma', x : c'$, because $\Gamma \overset{*}{\Leftrightarrow}_1 \Gamma'$ and $c = c'$. Then by induction hypothesis we have $\Gamma' \vdash c : p$ and by rule **(var)** $\Gamma', x : c \vdash x : c$.

   (b) $\Gamma, x : c \overset{*}{\Leftrightarrow}_1 \Gamma', x : c'$, because $\Gamma = \Gamma'$ and $\Gamma \vdash c(\overset{*}{\to} \cup \overset{*}{\leftarrow})^* c'$. We use the well-formedness of $\Gamma, x : c'$ to deduce $\Gamma \vdash c' : p'$. The desired conclusion $\Gamma, x : c' \vdash x : c'$ follows by rule **(var)**.

2. The last rule in the derivation of $\Gamma \vdash a : b$ was **(abs)**

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

   Since $\Gamma \overset{*}{\Leftrightarrow}_1 \Gamma'$ we have also $\Gamma, x : a \overset{*}{\Leftrightarrow}_1 \Gamma', x : a$. By induction hypothesis $\Gamma', x : a \vdash b : c$ and $\Gamma' \vdash (x{:}a).c : p$. Applying rule **(abs)** we get $\Gamma' \vdash \lambda x{:}a.b : (x{:}a).c$.

3. The reduction $\Gamma \vdash a \rightarrow b$ may be either β-reduction or rewrite reduction. For β-reduction it is obvious, since β-reduction does not depend on environment. For rewriting, let us suppose that there is a HORPO judgment $G \vdash l \succ r \ : T$, a position p and a substitution $\theta : G \rightarrow \Gamma, \mathcal{PV}(a,p)$ such that $a|_p = l\theta$ and $b = a[r\theta]_p$. Since $\theta : G \rightarrow \Gamma, PV(a,p)$, for every $x_i : A_i \in G$ we have $\Gamma, \mathcal{PV}(a,p) \vdash x_i\theta : A_i\theta$. By induction hypothesis we have also $\Gamma', \mathcal{PV}(a,p) \vdash x_i\theta : A_i\theta$. Consequently $\theta : G \rightarrow \Gamma', \mathcal{PV}(a,p)$ and $\Gamma' \vdash a \rightarrow_H b$. $\square$

Lemma 4.2.17 *Let Γ be an environment and $(x\!:\!A).B$ be a term (possibly not well-typed). If $\Gamma \vdash (x\!:\!A).B \rightarrow Q$ then $Q = (x\!:\!C).D$ and*

1. *$\Gamma \vdash A \rightarrow C$ and $B = D$, or*
2. *$A = C$ and $\Gamma, x : A \vdash B \rightarrow D$, or*
3. *$\Gamma \vdash A \overset{*}{\leftrightarrow} C$ and $\Gamma, x : A \vdash B \rightarrow D$.*

Proof
The only possibility to reduce in the root position is to use second case of Part II of HORPO. But then $(x\!:\!A).B$ and Q are well-typed, $Q = (x\!:\!C).D$ with $\Gamma \vdash A \overset{*}{\leftrightarrow} C$, $\Gamma, x : A \vdash B \rightarrow D$ and case 3 holds. Otherwise the reduction must take place in A or B resulting in 1 or 2. $\square$

Lemma 4.2.18 *Let Γ be an environment and $(x : A).B$, $(x : C).D$ be Γ-terms. If $\Gamma \vdash (x\!:\!A).B \rightarrow^* (x\!:\!C).D$ then $\Gamma \vdash A \overset{*}{\leftrightarrow} C$, $\Gamma, x : A \vdash B \overset{*}{\leftrightarrow} D$ and $\Gamma, x : C \vdash B \overset{*}{\leftrightarrow} D$.*

Proof
The base case of this lemma (one reduction step) is done in Lemma 4.2.17. Now we will show how to put together conclusions for many reductions steps.

Consider the reduction path from $(x\!:\!A).B$ to $(x\!:\!C).D$

$$(x\!:\!A).B = (x\!:\!E_1).F_1 \rightarrow \ldots (x\!:\!E_n).F_n = (x\!:\!C).D$$

By Lemma 4.2.17, at each reduction step $E_i = E_{i+1}$ or $E_i \rightarrow E_{i+1}$ or $\Gamma \vdash E_i \overset{*}{\leftrightarrow} E_{i+1}$ with E_i and E_{i+1} well-typed in Γ. Consequently $\Gamma \vdash A \overset{*}{\leftrightarrow} C$, because A and C are Γ-terms. Regarding the reductions on F's, they have always the form $\Gamma, x : E_i \vdash F_i \rightarrow F_{i+1}$. To put them together we have to convert the environments $\Gamma, x : E_i$ to one common environment, for example

$\Gamma, x : A$ or $\Gamma, x : C$. This can be done using Lemma 4.2.16, since $\Gamma, x : A$ and $\Gamma, x : C$ are well-formed (because $\Gamma, x : A \vdash \star : \Box$ and $\Gamma, x : C \vdash \star : \Box$ follow from the fact that $(x{:}A).B$ and $(x{:}C).D$ are Γ-terms). Summarizing, for all i we have $\Gamma, x : A \vdash F_i \to F_{i+1}$ or $F_i = F_{i+1}$, which implies $\Gamma, x : A \vdash B {\to^*} D$ (similarly one gets $\Gamma, x : C \vdash B {\to^*} D$). $\Box$

Lemma 4.2.19 *Let Γ be an environment and $(x : A).B$, $(x : C).D$ be Γ-terms. If $\Gamma \vdash (x{:}A).B \overset{*}{\leftrightarrow} (x{:}C).D$ through products, i.e., if there exists a natural number m and a sequence of Γ-terms $(x{:}c_1).d_1, \ldots (x{:}c_m).d_m$ such that:*

1. *$(x{:}c_1).d_1 = (x{:}A).B$,*
2. *$(x{:}c_m).d_m = (x{:}C).D$,*
3. *for every $i = 1 \ldots m-1$ either $\Gamma \vdash (x : c_i).d_i {\to^*} (x : c_{i+1}).d_{i+1}$ or $\Gamma \vdash (x{:}c_{i+1}).d_{i+1} {\to^*} (x{:}c_i).d_i$.*

then $\Gamma \vdash A \overset{*}{\leftrightarrow} C$ and $\Gamma, x : A \vdash B \overset{*}{\leftrightarrow} D$.

Proof
By Lemma 4.2.18 every $\Gamma \vdash (x : c_i).d_i {\to^*} (x : c_{i+1}).d_{i+1}$ implies that $\Gamma \vdash c_i \overset{*}{\leftrightarrow} c_{i+1}$ and $\Gamma, x : c_i \vdash d_i {\to^*} d_{i+1}$. Using Lemma 4.2.16 we get $\Gamma, x : A \vdash d_i {\to^*} d_{i+1}$ for all i. Putting all these judgments together results in $\Gamma \vdash A \overset{*}{\leftrightarrow} C$ and $\Gamma, x : A \vdash B \overset{*}{\leftrightarrow} D$. $\Box$

Corollary 4.2.20 *Let Γ be an environment and $(x : A).B$, $(x : C).D$ be Γ-kinds ($\Gamma \vdash (x{:}A).B : \Box$ and $\Gamma \vdash (x{:}C).D : \Box$). If $\Gamma \vdash (x{:}A).B \overset{*}{\leftrightarrow} (x{:}C).D$ then $\Gamma \vdash A \overset{*}{\leftrightarrow} C$ and $\Gamma, x : A \vdash B \overset{*}{\leftrightarrow} D$.*

Proof
By Lemma 4.2.6 all terms on the reduction/expansion sequence from $(x : A).B$ to $(x{:}C).D$ are products. We conclude by Lemma 4.2.19. $\Box$

4.3 Subject reduction

To show that $\to^{\Gamma}$ has the subject reduction property, it suffices to show it for the rewrite reduction ($\to^{\Gamma}_{H}$) and the beta reduction ($\to_{\beta}$) separately.

This will be done in Sections 4.3.2 and 4.3.3. The main difficulty in these proofs is to show that any two convertible product types can be joined by a sequence of conversions passing only through products. This is proved in Section 4.3.1, by adapting the results from [6, 7].

4.3.1 Conversion paths between products

As already mentioned, we follow closely the proof from [7]. Even though the latter 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 have the same type,

2. rewriting operates on object level.

Since these properties are true in our calculus, we can adapt the proof from [7] to our rewriting.

Definition 4.3.1 *Let Γ be an environment and $\lambda x\!:\!A.M$ a Γ-term. It is*

1. 0-abstraction *if A is a Γ-type and M is an $\Gamma, x : A$-object,*

2. 2-abstraction *if A is a Γ-kind and M is an $\Gamma, x : A$-object,*

3. P-abstraction *if A is a Γ-type and M is an $\Gamma, x : A$-constructor,*

4. ω-abstraction *if A is a Γ-kind and M is an $\Gamma, x : A$-constructor.*

Moreover this choice of abstractions is exhaustive.

We can decorate the λs correspondingly and we have the notions of β_0-reduction, β_2-reduction, β_P-reduction and β_ω-reduction, just by restricting reduction to the redexes with the appropriate subscript attached to the symbol λ. We use an arrow with a superscript above it to denote these restricted reductions (or the union of some of them), like in $\to_\beta^{P\omega}$.

We also have the usual notion of *β-weak-head-reduction*: a term M β-weak-head-reduces to M', notation $M \to_{wh} M'$, if M is itself a β-redex, say $M = (\lambda x : A.P)\, Q$ and M' is obtained by contracting $(\lambda x : A.P)\, Q$ (the *β-head-redex*). In the following we shall not consider (weak)-head-reductions, where the head-redex is a rewrite redex. Therefore, we omit explicit reference to β and refer to β-weak-head-reduction as *weak-head-reduction*. As usual, we define $\xrightarrow{*}_{wh}$ as the transitive and reflexive closure of $\to_{wh}$. Note that a term of the form $(x\!:\!A).B$ is in *weak-head-normal-form* (whnf).

Lemma 4.3.2 *For environments Γ and Γ' and terms P, P', D we have:*

1. $\Gamma \vdash P \,:\, D$ *and* $P \to_{\beta}^{P\omega} P'$ *imply* $\Gamma \vdash P' \,:\, D$,

2. $\Gamma \vdash P \,:\, D$ *and* $\Gamma \to_{\beta}^{P\omega} \Gamma'$ *imply* $\Gamma' \vdash P \,:\, D$.

Proof

The proof is by simultaneous induction on the derivation. Just as in the usual proof for β, the only interesting case is when the last rule is **(app)** with $P = (\lambda x{:}A.M)\, N$ and $P' = M[N/x]$. We then have

$$\frac{\Gamma \vdash \lambda x{:}A.M \,:\, (x{:}B).C \qquad \Gamma \vdash N \,:\, B}{\Gamma \vdash (\lambda x{:}A.M)\, N \,:\, C[N/x]}$$

where the λ is either λ_P or λ_ω. By applying Generation Lemma 4.1.8 to the first premise, we find

$$\begin{aligned} &\Gamma, x : A \vdash M \,:\, C' \\ &\Gamma \vdash (x{:}A).C' \,:\, p \\ &\Gamma \vdash (x{:}A).C' \overset{*}{\leftrightarrow} (x{:}B).C \end{aligned} \tag{4.1}$$

with $p \in \{\star, \Box\}$. So again by Generation

$$\begin{aligned} \Gamma \vdash A \,:\, p_1 \\ \Gamma, x : A \vdash C' \,:\, p \end{aligned} \tag{4.2}$$

for some $p_1 \in \{\star, \Box\}$. By the fact that we have P or ω-redex, we know that M is a $(\Gamma, x : A)$-constructor and hence that $(x : A).C'$ and $(x : B).C$ are kinds. By Lemma 4.2.20

$$\Gamma \vdash A \overset{*}{\leftrightarrow} B \tag{4.3}$$

$$\Gamma, x : A \vdash C' \overset{*}{\leftrightarrow} C \tag{4.4}$$

So, applying the conversion rule to (4.2), $\Gamma \vdash N \,:\, B$ and (4.3), we get

$$\Gamma \vdash N \,:\, A. \tag{4.5}$$

By applying substitution (Lemma 4.1.3) to (4.1) and (4.5) we get

$$\Gamma \vdash M[N/x] \,:\, C'[N/x]. \tag{4.6}$$

By applying Lemma 4.2.1 to the first premise $\Gamma \vdash \lambda x{:}A.M \,:\, (x{:}B).C$, we find $\Gamma \vdash (x{:}B).C \,:\, p'$ for some $p' \in \{\star, \Box\}$ and hence by Generation

$$\Gamma, x : B \vdash C \,:\, p' \tag{4.7}$$

Now apply Substitution Lemma to (4.7) and $\Gamma \vdash N : B$ to get

$$\Gamma \vdash C[N/x] : p'. \tag{4.8}$$

Apply the conversion rules to (4.6) and (4.8) (using $\Gamma \vdash C'[N/x] \overset{*}{\leftrightarrow} C[N/x]$, which follows from (4.4) and Lemma 4.1.3) to conclude

$$\Gamma \vdash M[N/x] : C[N/x]$$

and we are done. $\Box$

Lemma 4.3.3 *If $\Gamma \vdash Q : A$, $D[Q/x] = \star$, and $\Gamma \vdash (x : A).D : p$ with $p \in \{\star, \Box\}$ then Q cannot be $\star$.*

Proof
If $Q = \star$ then $A = \Box$ by Generation and Lemma 4.2.3. Hence, $\Box$ is a subterm of well-typed term $(x:A).D$ which is impossible by Lemma 4.2.2. $\Box$

Lemma 4.3.4 *If $\Gamma \vdash (\lambda x:C.P)\, Q : p \in \{\star, \Box\}$, then it is a P or ω-redex.*

Proof
By Generation there are A, B and D such that

$$\Gamma, x : C \vdash P : B,$$
$$\Gamma \vdash Q : A,$$
$$\Gamma \vdash \lambda x:C.P : (x:A).D,$$
$$\Gamma \vdash (x:C).B \overset{*}{\leftrightarrow} (x:A).D,$$
$$\Gamma \vdash D[Q/x] \overset{*}{\leftrightarrow} p$$

If $p = \Box$ then either Q must be $\Box$, which is impossible since Q has a type A and $\Box$ has no type, or D must be $\Box$, which is also impossible since otherwise $(x:A).\Box$ would have a type.

Hence $D[Q/x] = \star$ and so $D = \star$, because Q cannot be $\star$ by Lemma 4.3.3. We have $\Gamma \vdash (x : A).D : \Box$ and by Lemma 4.2.7 $\Gamma \vdash (x : C).B : \Box$. By Generation $\Gamma, x : C \vdash B : \Box$ and P is a constructor. We conclude that λ is decorated with P or ω and we are done. $\Box$

As a consequence, we find that subject reduction holds for weak-head-reduction on types and kinds.

Corollary 4.3.5 *If $\Gamma \vdash M : p$ with $p \in \{\star, \Box\}$ and $M \stackrel{*}{\to}_{wh} N$ then $\Gamma \vdash N : p$.*

Moreover, the true weak-head-reduction is possible only if $p = \star$ (if $p = \Box$ then $M = N$).

We now formulate three useful lemmas on the interaction between $\stackrel{*}{\to}_H$, $\stackrel{*}{\to}_\beta$ and $\stackrel{*}{\to}_{wh}$. The first one is formulated on pseudoterms, while the other ones are working on well-typed terms.

Lemma 4.3.6 (Commutativity of $\stackrel{*}{\to}_{wh}$ with $\stackrel{*}{\to}_\beta$) *If $M \stackrel{*}{\to}_{wh} P$ and $M \stackrel{*}{\to}_\beta N$ then there exists Q such that*

$$P \stackrel{*}{\to}_\beta Q \quad \textit{and} \quad N \stackrel{*}{\to}_{wh} Q$$

Proof
Recall that weak-head-reduction refers only to β-weak-head-reduction. We prove the lemma for the case of a one step weak-head-reduction: if $M \to_{wh} P$ and $M \stackrel{*}{\to}_\beta N$ then $P \stackrel{*}{\to}_\beta Q$ and $N \to_{wh} Q$ for some Q. This immediately implies the general case for $M \stackrel{*}{\to}_{wh} P$. So, let $M \to_{wh} P$, say $M = (\lambda x : A.B)\, C$. If $N = (\lambda x : A'.B')\, C'$ with $A \stackrel{*}{\to}_\beta A'$, $B \stackrel{*}{\to}_\beta B'$ and $C \stackrel{*}{\to}_\beta C'$ we are done by taking $Q = B'[C'/x]$. If in the reduction from M to N the head-redex is contracted we are also done, by taking $Q = N$.
$\Box$

Lemma 4.3.7 (Commutativity of $\stackrel{*}{\to}_{wh}$ with $\stackrel{*}{\to}_H$) *For Γ an environment, M a Γ-type or Γ-kind and P, N terms if $M \stackrel{*}{\to}_{wh} P$ and $\Gamma \vdash M \stackrel{*}{\to}_H N$ then there exists Q such that*

$$\Gamma \vdash P \stackrel{*}{\to}_H Q \quad \textit{and} \quad N \stackrel{*}{\to}_{wh} Q$$

Proof
We prove the lemma for the case of a one step weak-head-reduction: if $M \to_{wh} P$ and $\Gamma \vdash M \stackrel{*}{\to}_H N$ then $\Gamma \vdash P \stackrel{*}{\to}_H Q$ and $N \to_{wh} Q$ for some Q. This immediately implies the general case, since P is a Γ-type or Γ-kind by Corollary 4.3.5. Moreover only if M is a Γ-type the true weak-head reduction may arise; otherwise $M = P$ and $Q = N$.

So, let $M \to_{wh} P$, say $M = (\lambda x : A.B)\, C$. Then $N = (\lambda x : A'.B')\, C'$ with $\Gamma \vdash A \stackrel{*}{\leftrightarrow} A'$, $\Gamma, x : A \vdash B \stackrel{*}{\to}_H B'$ and $\Gamma \vdash C \stackrel{*}{\to}_H C'$. Let us take $Q = B'[C'/x]$. We are left to show that $\Gamma \vdash B[C/x] \stackrel{*}{\to}_H B'[C'/x]$.

By assumption, $\Gamma \vdash (\lambda x{:}A.B)\,C : \star$ holds. Then by Generation we find U, W and D such that

$$\Gamma, x : A \vdash B : W,$$
$$\Gamma \vdash C : U,$$
$$\Gamma \vdash \lambda x{:}A.B : (x{:}A).W,$$
$$\Gamma \vdash (x{:}A).W \overset{*}{\leftrightarrow} (x{:}U).D,$$
$$\Gamma \vdash D[C/x] \overset{*}{\leftrightarrow} \star$$

Hence $D[C/x] = \star$ and so $D = \star$, because C cannot be $\star$ (by Lemma 4.3.3). So, we have $\Gamma \vdash (x : U).D : \Box$, $\Gamma \vdash (x : A).W : \Box$ and by Lemma 4.2.20 $\Gamma \vdash A \overset{*}{\leftrightarrow} U$. By the conversion rule $\Gamma \vdash C : A$.

Once we have $\Gamma, x : A \vdash B : W$ and $\Gamma \vdash C : A$ it is sufficient to use Lemma 4.2.14 to get $\Gamma \vdash B[C/x] \overset{*}{\rightarrow}_H B'[C'/x]$. $\Box$

Lemma 4.3.8 (Postponement of $\overset{*}{\rightarrow}_H$ with respect to $\overset{*}{\rightarrow}_{wh}$) *For Γ an environment, M a Γ-type or Γ-kind and P, N terms if $\Gamma \vdash M \overset{*}{\rightarrow}_H N$ and $N \overset{*}{\rightarrow}_{wh} P$ then there exists Q such that*

$$M \overset{*}{\rightarrow}_{wh} Q \quad \textit{and} \quad \Gamma \vdash Q \overset{*}{\rightarrow}_H P$$

Proof
We prove the lemma for the case of a one step weak-head-reduction: if $\Gamma \vdash M \overset{*}{\rightarrow}_H N$ and $N \rightarrow_{wh} P$ then $M \rightarrow_{wh} Q$ and $\Gamma \vdash Q \overset{*}{\rightarrow}_H P$ for some Q. This immediately implies the general case, since Q is a Γ-type or Γ-kind by Corollary 4.3.5.

From $N \rightarrow_{wh} P$ we conclude that $N = (\lambda x : A.B)\,C$ and $P = B[C/x]$ for some A, B, C. As M is a Γ-kind or Γ-type, we know that no rewrite rule can be applied in the root position (we have only rewriting on objects). Hence, $M = (\lambda x : A'.B')\,C'$ with $\Gamma \vdash A' \overset{*}{\leftrightarrow} A$, $\Gamma, x : A' \vdash B' \overset{*}{\rightarrow}_H B$ and $\Gamma \vdash C' \overset{*}{\rightarrow}_H C$. Now, take $Q = B'[C'/x]$. Then obviously $M \rightarrow_{wh} Q$. The proof that $\Gamma \vdash Q \overset{*}{\rightarrow}_H P$, i.e. $\Gamma \vdash B'[C'/x] \overset{*}{\rightarrow}_H B[C/x]$ proceeds exactly as in the previous lemma (we have $\Gamma \vdash (\lambda x{:}A'.B')\,C' : \star$ by assumption that M is a Γ-type or Γ-kinds that truly weak-head-reduces). $\Box$

Lemma 4.3.9 *Let Γ be an environment and E, F and $(x{:}A).B$ be Γ-types or Γ-kinds. Then there is a term Q satisfying the diagrams given below. Moreover this Q is of the form $(x{:}A').B'$ and it is a Γ-type or Γ-kind.*

(I)
$$\begin{tikzcd}
E \arrow[r, "*" near end, "H"'] \arrow[d, "wh"', "*"' near end] & F \arrow[d, dotted, "wh", "*" near end] \\
(x:A).B \arrow[r, dotted, "*" near end, "H"'] & Q
\end{tikzcd}$$

(II)
$$\begin{tikzcd}
E \arrow[r, "*" near end, "\beta"'] \arrow[d, "wh"', "*"' near end] & F \arrow[d, dotted, "wh", "*" near end] \\
(x:A).B \arrow[r, dotted, "*" near end, "\beta"'] & Q
\end{tikzcd}$$

(III)
$$\begin{tikzcd}
E \arrow[r, "*" near end, "H"'] \arrow[d, dotted, "wh"', "*"' near end] & F \arrow[d, "wh", "*" near end] \\
Q \arrow[r, dotted, "*" near end, "H"'] & (x:A).B
\end{tikzcd}$$

(IV)
$$\begin{tikzcd}
E \arrow[r, "*" near end, "\beta"'] \arrow[d, dotted, "wh"', "*"' near end] & F \arrow[d, "wh", "*" near end] \\
Q \arrow[r, dotted, "*" near end, "\beta"'] & (x:A).B
\end{tikzcd}$$

Proof

First note that the term Q is always a Γ-type or Γ-kind by Corollary 4.3.5, as a reduct of F or E which are Γ-types or Γ-kinds.

For diagrams (I) and (II) it is of the form $(x:A').B'$ because it results from reduction starting at $(x:A).B$ (and neither $\to_\beta$ nor $\to_H$ can destroy a product).

The existence of Q for diagram (I) and (II) follows easily from Lemma 4.3.7 and 4.3.6 respectively.

For diagram (III) let us consider the reduction sequence from Γ-type or Γ-kind Q to $(x:A).B$. By induction on the length of this sequence we may show that all terms on it are products. Indeed, $(x:A).B$ is a product and since all rewrite rules are defined on object-level, every Q_n such that $\Gamma \vdash Q_n \to_H (x:A_{n-1}).B_{n-1}$ must be of the form $(x:A_n).B_n$. The existence of Q for diagram (III) follows from Lemma 4.3.8.

For diagram (IV) we use the well-known Standardization Theorem. It states that if $M \stackrel{*}{\to}_\beta N$, then there is a 'standard' reduction path from M to N, i.e. a path that reduces outside-in. If one considers such a reduction from E to $(x:A).B$, then either $E = (x:A').B'$ (and $Q = E$) or there must be a reduction step $(\lambda x:C.D)\,P \to_\beta (x:A').B'$ on the path (for some C, D, P, A' and B'). But then the reduction path from E to $(\lambda x:C.D)\,P$ must consist of weak-head-reductions only, because any other reduction would block the possibility to contract $(\lambda x:C.D)\,P$. So we are done. $\square$

Lemma 4.3.10 (Main Lemma for Subject Reduction) *Let Γ be an environment and $(x:A).B$ and $(x:A').B'$ two Γ-terms. If $\Gamma \vdash (x:A).B \overset{*}{\leftrightarrow} (x:A').B'$ then there exist a natural number m and a sequence of Γ-terms $(x:c_1).d_1, \ldots (x:c_m).d_m$ such that:*

1. $(x:c_1).d_1 = (x:A).B$,
2. $(x:c_m).d_m = (x:A').B'$,
3. $\Gamma \vdash (x:c_i).d_i(\rightarrow^*_\beta \cup \rightarrow^*_H)(x:c_{i+1}).d_{i+1}$ *or* $\Gamma \vdash (x:c_{i+1}).d_{i+1}(\rightarrow^*_\beta \cup \rightarrow^*_H)(x:c_i).d_i$ *for every* $i = 1 \ldots m-1$.

Proof

By the definition of $\Gamma \vdash (x:A).B \overset{*}{\leftrightarrow} (x:C).D$ there is a natural number n (we may assume that it is odd without loss of generality) and Γ-terms $E_1, \ldots E_n$ such that

1. $\Gamma \vdash E_1(\rightarrow^*_\beta \cup \rightarrow^*_H)(x:A).B$,
2. $\Gamma \vdash E_n(\rightarrow^*_\beta \cup \rightarrow^*_H)(x:A').B'$,
3. for every even i $\Gamma \vdash E_{i-1}(\rightarrow^*_\beta \cup \rightarrow^*_H)E_i$ and $\Gamma \vdash E_{i+1}(\rightarrow^*_\beta \cup \rightarrow^*_H)E_i$.

The following diagram shows how to transform this E-sequence into the sequence of products (every ρ_i stands for β or H). Note that all arrows exist and that products are well-typed in Γ due to Lemma 4.3.9.

$$\begin{tikzcd}
 & E_1 \arrow[dl, "\rho_1"', "*" near end] \arrow[d, "wh", "*"' near end] \arrow[dr, "\rho_2", "*"' near end] & & E_3 \arrow[dl, "\rho_3"', "*" near end] \arrow[d, "wh", "*"' near end] & \cdots \\
(x:A).B & (x:C_1).D_1 \arrow[l, "\rho_1", "*"' near end] \arrow[dr, "\rho_2"', "*" near end] & E_2 \arrow[d, "wh", "*"' near end] & (x:C_3).D_3 \arrow[dl, "\rho_3", "*"' near end] & \cdots \\
 & & (x:C_2).D_2 & &
\end{tikzcd}$$

$\square$

Corollary 4.3.11 *Let Γ be an environment and $(x:A).B$, $(x:C).D$ be Γ-terms. If $\Gamma \vdash (x:A).B \overset{*}{\leftrightarrow} (x:C).D$ then $\Gamma \vdash A \overset{*}{\leftrightarrow} C$ and $\Gamma, x:A \vdash B \overset{*}{\leftrightarrow} D$.*

Proof
By Lemma 4.3.10 we know that there exists a sequence of reductions and expansions going through well-typed products. To get the desired conclusion it is sufficient to use Lemma 4.2.19. $\square$

Lemma 4.3.12 (Uniqueness of types) *Let Γ be an environment and a, b_1 and b_2 be terms. If $\Gamma \vdash a : b_1$ and $\Gamma \vdash a : b_2$ then $\Gamma \vdash b_1 \overset{*}{\leftrightarrow} b_2$.*

Proof
By induction on the structure of a. We treat the case of variable and application. Other cases can be solved similarly.

If $a = x$ then by Generation there exist c such that $x : c \in \Gamma$, $x \in \mathcal{V}ar^p$ and $\Gamma \vdash c : p$. Moreover $\Gamma \vdash b_1 \overset{*}{\leftrightarrow} c$ and $\Gamma \vdash b_2 \overset{*}{\leftrightarrow} c$. Hence $\Gamma \vdash b_1 \overset{*}{\leftrightarrow} b_2$.

If $a = c\,d$ then by Generation there are u_1, u_2, w_1, w_2 such that $\Gamma \vdash c : (x{:}u_1).w_1$, $\Gamma \vdash d : u_1$, $\Gamma \vdash b_1 \overset{*}{\leftrightarrow} w_1[d/x]$ and $\Gamma \vdash c : (x{:}u_2).w_2$, $\Gamma \vdash d : u_2$, $\Gamma \vdash b_2 \overset{*}{\leftrightarrow} w_2[d/x]$. By induction hypothesis $\Gamma \vdash (x : u_1).w_1 \overset{*}{\leftrightarrow} (x : u_2).w_2$. By Lemma 4.3.11 $\Gamma, x : u_1 \vdash w_1 \overset{*}{\leftrightarrow} w_2$ and by Lemma 4.1.3 $\Gamma \vdash w_1[d/x] \overset{*}{\leftrightarrow} w_2[d/x]$. Consequently $\Gamma \vdash b_1 \overset{*}{\leftrightarrow} b_2$. $\square$

4.3.2 The subject reduction property of rewriting

Subject reduction of $\to_H^\Gamma$ will be a consequence of the following lemma.

Lemma 4.3.13 *For all environments Γ, Γ' and all terms a, a', b, c, c':*

(1). $\Gamma \vdash a : b$ *and* $\Gamma \vdash a \to_H a'$ *imply* $\Gamma \vdash a' : b$,

(2). $\Gamma \vdash a : b$ *and* $\Gamma \to_H \Gamma'$ *imply* $\Gamma' \vdash a : b$,

(3). $\Gamma \vdash c \to_H c'$ *and* $\Gamma \to_H \Gamma'$ *imply* $\Gamma' \vdash c \to_H c'$,

(4). $\Gamma \vdash c \to^* c'$ *and* $\Gamma \to_H \Gamma'$ *imply* $\Gamma' \vdash c \to^* c'$.

Proof
The proof is done by mutual induction on the derivation of $\Gamma \vdash a : b$, $\Gamma \vdash c \to_H c'$ and $\Gamma \vdash c \to^* c'$.

(1) We give the details only for the case when the last step in the derivation was **(const)**. The other cases are similar or simpler.

$$\frac{\Gamma \vdash \star : \square \qquad \forall i \quad \Gamma \vdash a_i : d_i[\vec{a}/\vec{y}]}{\Gamma \vdash e(\vec{a}) : v[\vec{a}/\vec{y}]} \qquad (e : (y_1 : d_1) \ldots (y_n : d_n).v \in \Sigma)$$

Let $\Gamma \vdash e(a_1, \ldots, a_n) \to_H w$. We have to check that $\Gamma \vdash w : v[\vec{a}/\vec{y}]$.

There are two cases: either there is a j such that $\Gamma \vdash a_j \to_H a'_j$ 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 $\gamma'_i = [a'_1/y_1, \ldots a'_{i-1}/y_{i-1}]$ for all i. To apply the **(const)** rule to $a'_1, \ldots a'_n$ we first have to convert the type for a'_i for $i > j$.

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

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

This is trivial for $i < j$, since $a'_i = a_i$. For $i = j$ we have $\Gamma \vdash a_j \to_H a'_j$ 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 $i \geq j$. We will show $(*)$ for $i + 1$.

By $(*)$, γ'_{i+1} is a well-typed substitution from $y_1 : d_1, \ldots y_i : d_i$ to Γ and therefore $y_1 : d_1, \ldots y_i : d_i \vdash d_{i+1} : p_{i+1}$ implies $\Gamma \vdash d_{i+1}\gamma'_{i+1} : p_{i+1}$ by Substitution property 4.1.3. Now, by Lemma 4.2.14, $\Gamma \vdash d_{i+1}[\vec{a}/\vec{y}] \overset{*}{\to}_H d_{i+1}\gamma'_{i+1}$. We can apply **(conv)** 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 $\Gamma \vdash v[\vec{a}/\vec{y}] \overset{*}{\to}_H v\gamma'_{n+1}$.

Now, having $(*)$ for all $i = 1, \ldots n$, we deduce $\Gamma \vdash e(a_1, \ldots a'_j, \ldots a_n) : v\gamma'_{n+1}$. To obtain the conclusion, we have to use once again the conversion rule together with the facts that $\Gamma \vdash v[\vec{a}/\vec{y}] \overset{*}{\to}_H v\gamma'_{n+1}$ 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 $G \vdash l \to r : T$ and a well-typed substitution $\theta : G \to \Gamma$ such that $G \vdash l : T$, $G \vdash r : T$, $e(\vec{a}) = l\theta$, $w = r\theta$ and $\Gamma \vdash T\theta \overset{*}{\leftrightarrow} v[\vec{a}/\vec{y}]$ (by Lemma 4.3.12). 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 $\Gamma \vdash w : v[\vec{a}/\vec{y}]$.

(2) All cases are easy. We show the proof when **(weak)** is the last rule in the derivation of $\Gamma, x : c \vdash a : b$.

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

There are two subcases: when $\Gamma, x : c \to \Gamma', x : c$ and when $\Gamma, x : c \to \Gamma, x : c'$ with $\Gamma \vdash c \to c'$. In the first case we conclude by induction hypothesis of (2). In the second case we get $\Gamma \vdash c' : p$ by induction hypothesis (1) and $\Gamma, x : c' \vdash a : b$ by **(weak)**.

(3) Rewriting is defined by the rule **(rew)**

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

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

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 $\Gamma \to_H \Gamma'$ implies $(\Gamma, \mathcal{PV}(a,p)) \to_H (\Gamma', \mathcal{PV}(a,p))$. But this means that $\theta \,:\, G \to \Gamma', \mathcal{PV}(a,p)$ and consequently, by **(rew)** $\Gamma' \vdash a[l\theta]_p \to_H a[r\theta]_p$.

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

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

4.3.3 The subject reduction property of beta reduction

Lemma 4.3.14 *Let Γ and Γ' be environments and a, a' and b terms. Then*

(1). $\Gamma \vdash a \,:\, b$ *and* $a \to_\beta a'$ *imply* $\Gamma \vdash a' \,:\, b$,

(2). $\Gamma \vdash a \,:\, b$ *and* $\Gamma \to_\beta \Gamma'$ *imply* $\Gamma' \vdash a \,:\, b$,

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 $\Gamma \vdash (x:c).d \stackrel{*}{\leftrightarrow} (x:c').d'$ implies $\Gamma \vdash c \stackrel{*}{\leftrightarrow} c'$ and $\Gamma, x : c \vdash d \stackrel{*}{\leftrightarrow} d'$. But this follows from Corollary 4.3.11. The rest of the proof for **(app)** is identical to the one already presented in Lemma 4.3.2. $\Box$

Theorem 1 (Subject reduction) *Let Γ be an environment and a, a' and b be terms. If $\Gamma \vdash a \,:\, b$ and $\Gamma \vdash a \to a'$ then $\Gamma \vdash a' \,:\, b$.*

Proof
This follows from Lemma 4.3.13 and Lemma 4.3.14. $\Box$

4.4 Conversion properties

The lemmas in this section describe main properties of the conversion relation which will be needed in the proof of strong normalization and especially for the definition of candidates.

Lemma 4.4.1 *Let* Γ *be an environment.*

1. *If* $\Gamma \vdash a : b_1$ *and* $\Gamma \vdash b_1 \rightarrow^* b_2$ *then* $\Gamma \vdash a : b_2$.

2. *If* $\Gamma \vdash a : b_1$*,* $\Gamma \vdash b_2 \rightarrow^* b_1$ *and* $\Gamma \vdash b_2 : c$ *then* $\Gamma \vdash a : b_2$.

Proof
For 1, by Lemma 4.2.1 $b_1 = p$ or $\Gamma \vdash b_1 : p$ for some $p \in \{\star, \Box\}$. If $b_1 = p$ then $b_2 = p$ (by Lemma 4.2.3) and we are done. Otherwise, by subject reduction, $\Gamma \vdash b_2 : p$ and using **(conv)** rule we get $\Gamma \vdash a : b_2$.

For 2, like above by Lemma 4.2.1 $b_1 = p$ or $\Gamma \vdash b_1 : p$ for some $p \in \{\star, \Box\}$. If $b_1 = p$ then $b_2 = p$ (by Lemma 4.2.3) and we are done. Otherwise, by subject reduction $\Gamma \vdash b_1 : c$ and by Lemma 4.3.12 $\Gamma \vdash c \overset{*}{\leftrightarrow} p$. By Lemma 4.2.3 $c = p$ and using **(conv)** rule we get $\Gamma \vdash a : b_2$. $\Box$

Lemma 4.4.2 *Let* Γ *be an environment and* a_1*,* a_2 *and* b *be terms. If* $\Gamma \vdash a_1 : b$ *and* $\Gamma \vdash a_1 \overset{*}{\leftrightarrow} a_2$ *then* $\Gamma \vdash a_2 : b$.

Proof
By the definition of $\Gamma \vdash a_1 \overset{*}{\leftrightarrow} a_2$ there is a natural number n (without loss of generality we can assume that it is odd) and Γ-terms $c_1, \dots c_n$ ($\forall i \quad \Gamma \vdash c_i : d_i$) such that

1. $c_1 = a_1$,

2. $c_n = a_2$,

3. $\Gamma \vdash c_{i-1} \rightarrow^* c_i$ and $\Gamma \vdash c_{i+1} \rightarrow^* c_i$ for every even i.

By subject reduction for every even i we have $\Gamma \vdash c_i : d_{i-1}$ and $\Gamma \vdash c_i : d_{i+1}$. By Uniqueness Lemma $\Gamma \vdash d_i \overset{*}{\leftrightarrow} d_{i-1}$ and $\Gamma \vdash d_i \overset{*}{\leftrightarrow} d_{i+1}$. Consequently $\Gamma \vdash d_1 \overset{*}{\leftrightarrow} d_n$ and $\Gamma \vdash a_2 : b$. $\Box$

Lemma 4.4.3 (Classes of Γ-terms are disjoint) *The classes of* Γ*-objects,* Γ*-types,* Γ*-constructors that are not* Γ*-types and* Γ*-kinds are pairwise disjoint.*

Proof
Classification Lemma shows that

$$\Gamma\text{-kind} \cap \Gamma\text{-type} = \Gamma\text{-constr} \cap \Gamma\text{-obj} = \emptyset$$

If we suppose that $\Gamma\text{-obj} \cap \Gamma\text{-type} \neq \emptyset$ then we find A, such that $\Gamma \vdash A : B : \star$, for some B, and $\Gamma \vdash A : \star$. By Lemma 4.3.12 we have $\Gamma \vdash B \overset{*}{\leftrightarrow} \star$, which by Lemma 4.2.3 implies $B = \star$ and $\Gamma \vdash \star : \star$ which is impossible.

If we suppose that $\Gamma\text{-obj} \cap \Gamma\text{-kind} \neq \emptyset$ we can repeat the reasoning presented above and get $\Gamma \vdash \Box : \star$ which is also impossible.

If we suppose that $\Gamma\text{-constr} \cap \Gamma\text{-kind} \neq \emptyset$ we can once again repeat the same reasoning and get $\Gamma \vdash \Box : \Box$ which is impossible.

Of course Γ-constructors that are not Γ-types are disjoint with Γ-types.
$\Box$

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 $\mathcal{V}ar^\Box$:

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

The four statements below state that being an object, a type, a constructor or a kind is closed under conversion, substitution or environment extension.

Lemma 4.4.4 *Let Γ be an environment and let A be a Γ-term. Suppose that A is a Γ-object, or a Γ-type, or a Γ-constructor or a Γ-kind.*

1. *If $\Gamma \vdash A \overset{*}{\leftrightarrow} A'$ then A' belongs to the corresponding Γ-class.*
2. *If $\Gamma \overset{*}{\leftrightarrow} \Gamma'$ then A belongs to the corresponding Γ'-class.*
3. *If $\rho : \Gamma \to \Delta$ is a well-typed substitution then $A\rho$ belongs to the corresponding Δ-class.*
4. *If $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash \star : \Box$ then A belongs to the corresponding Γ'-class.*

Proof
For 1 we will show the proof for two cases: if $\Gamma \vdash A \to^* A'$ and if $\Gamma \vdash A' \to^* A$ with $\Gamma \vdash A' : T$ for some T. The general case ($\Gamma \vdash A \overset{*}{\leftrightarrow} A'$) follows easily.

If $\Gamma \vdash A \rightarrow^* A'$ and $\Gamma \vdash A : B$ for some B then by subject reduction $\Gamma \vdash A' : B$. Consequently A' is in the same Γ-class.

If $\Gamma \vdash A' \rightarrow^* A$ with $\Gamma \vdash A' : T$ then by subject reduction $\Gamma \vdash A : T$. Now if we suppose that $\Gamma \vdash A : p$ for some $p \in \{\star, \Box\}$ then by Lemmas 4.3.12 and 4.2.3 $T = p$ and $\Gamma \vdash A' : p$ (so we are done with the cases of Γ-type and Γ-kind). If $\Gamma \vdash A : B : p$ (A is an object or constructor) then by Lemma 4.3.12 $\Gamma \vdash T \overset{*}{\leftrightarrow} B$. By Lemma 4.2.1 applied to $\Gamma \vdash A' : T$ we get $T = p'$ or $\Gamma \vdash T : p'$. If $T = p'$ then $T = B$ and $\Gamma \vdash A' : B : p$. Otherwise by Lemma 4.2.8 $p = p'$ and $\Gamma \vdash A' : T : p$ and consequently A' and A are in the same Γ-class.

Cases 2, 3 and 4 are easy consequences of Lemmas 4.2.16, 4.1.3 and 4.1.5 respectively. $\Box$

Chapter 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 [30].

The proof follows the standard scheme. We first define candidates, as subsets 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 [30] we have to account for rewriting 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. Both type constructor headed 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, 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 [53], but the fact that function symbols have dependent types makes the proof much more complicated here.

This chapter is organized as follows. We begin with the definition of 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. 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) *The set of neutral terms* Neutr *is the set of terms, which are not an abstraction or constructor headed.*

Please recall that $\rightarrow$ stands for $\rightarrow_H \cup \rightarrow_\beta$, i.e., the rewrite relation generated by HORPO and β-reduction.

Definition 5.1.2 (Strongly normalizing terms) *Given an environment Δ the relation $\rightarrow_{sn}^{\Delta}$ is defined as follows*

$$\Delta' \vdash u \rightarrow_{sn}^{\Delta} \Delta'' \vdash w \textit{ if and only if } \Delta \subseteq \Delta' \subseteq \Delta'' \textit{ and } \Delta'' \vdash u \rightarrow w$$

We say that the sequent $\Delta \vdash M$ is strongly normalizing *($\mathcal{SN}$) if there is no infinite sequence $(\Delta_j \vdash M_j)_{j=0}^{\infty}$, such that $\Delta_j \vdash M_j \rightarrow_{sn}^{\Delta} \Delta_{j+1} \vdash M_{j+1}$ holds for every j, with $\Delta_0 = \Delta$ and $M_0 = M$.*

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

As one can see, our definition of $\mathcal{SN}$ is stronger than usual. This is due to the fact that on the right-hand side of a HORPO judgment defining a function symbol f, one may have recursive calls of f under a λ, and therefore the environment may grow with reduction.

Fortunately, this stronger definition of $\mathcal{SN}$ has all expected properties:

Lemma 5.1.3 (Basic properties of $\mathcal{SN}$)

1. *If $\Delta \subseteq \Delta'$ then $\mathcal{SN}_{\Delta} \supseteq \mathcal{SN}_{\Delta'}$.*

2. *If* $\Delta \subseteq \Delta' \subseteq \Delta''$ *and* $\Delta'' \vdash u \in \mathcal{SN}_{\Delta}$ *then* $\Delta'' \vdash u \in \mathcal{SN}_{\Delta'}$.

3. *If* $\Delta' \vdash u \in \mathcal{SN}_{\Delta,A}$ *and* $\Delta' \vdash u \to w$ *then* $\Delta' \vdash w \in \mathcal{SN}_{\Delta,A}$.

4. *Let* $\Delta \subseteq \Delta'$ *and* a *be a neutral term satisfying* $\Delta' \vdash a : A$. *If for every environment* Δ'' *and term* a' *such that* $\Delta'' \vdash a \to a'$, $\Delta'' \vdash a' \in \mathcal{SN}_{\Delta,A}$ *then* $\Delta' \vdash a \in \mathcal{SN}_{\Delta,A}$.

5. *Let* $\Delta \subseteq \Delta' \subseteq \Delta''$. *If* $\Delta' \vdash u \in \mathcal{SN}_{\Delta,A}$ *then* $\Delta'' \vdash u \in \mathcal{SN}_{\Delta,A}$.

Proof
Straightforward. $\square$

Definition 5.1.4 *Every* Δ*-term that is not a* Δ*-object is called* Δ-non-object.

Definition 5.1.5 (Family of candidates) *A family* $\mathcal{C}$ *of sets* $\mathcal{C}_{\Delta,A}$ *where* A *is a* Δ*-non-object, is called a* family of candidates *if the sets* $\mathcal{C}_{\Delta,A}$ *satisfy the conditions below. Each* $C \in \mathcal{C}_{\Delta,A}$ *is called a* candidate. *At the same time, for every* $\Delta' \supseteq \Delta$ *we define the* restriction $C|_{\Delta'}$ 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 $\Delta' \vdash a \to a'$ 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 environment Δ'' and term a' satisfying $\Delta'' \vdash a \to a'$, $\Delta'' \vdash a' \in C$ 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*$\backslash$*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 $\Delta' \supseteq \Delta$ to be the function $h|_{\Delta'}$ 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 $\{\Delta' \vdash M \mid \Delta' \vdash M : B$ 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 $\Delta' \supseteq \Delta$ to be the function $h|_{\Delta'}$ with domain $\{\Delta'' \vdash M \mid \Delta'' \vdash M : B$ 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 Subsection 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 5.1.6 (Restricted candidate) *Let A be a Δ-non-object, $C \in \mathcal{C}_{\Delta,A}$ 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 A is a kind such that $\Delta \vdash A : (x:B).D$ 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 $\Delta'' \vdash M_1 \stackrel{*}{\leftrightarrow} M_2$, 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

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

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 $\Box$, a kind or a type, and a function, returning $can_{\Delta,A\,M}$ for a given M, otherwise.

Definition 5.1.7 *Let Γ be an environment, T be a Γ-type and K a Γ-kind. We define $m(\Gamma \vdash T)$ and $m(\Gamma \vdash K)$ inductively as follows:*

$$\begin{array}{lcl} m(\Gamma \vdash \Box) & = & 0 \\ m(\Gamma \vdash T) & = & 0 \\ m(\Gamma \vdash K) & = & \left\{ \begin{array}{ll} 1 & \textit{if } K = \star \\ max(m(\Gamma \vdash B), m(\Gamma, x : B \vdash D)) + 1 & \textit{if } K = (x:B).D \end{array} \right. \end{array}$$

Lemma 5.1.8 *Let Γ, Γ' be environments such that $\Gamma \overset{*}{\leftrightarrow} \Gamma'$. For every Γ-non-object A we have $m(\Gamma \vdash A) = m(\Gamma' \vdash A)$.*

Proof
First note that if A is a Γ-kind (Γ-type) then it is also a Γ'-kind (Γ'-type). If A is a $\Box$ or a Γ-type then $m(\Gamma \vdash A) = m(\Gamma' \vdash A)$. Otherwise, A is a Γ-kind, $A = \overrightarrow{(x : T)}.\star$ and n is the length of $\overrightarrow{x : T}$. Let us proceed by induction on n.

If $n = 0$ then $m(\Gamma \vdash A) = 1 = m(\Gamma' \vdash A)$. Otherwise, let us call A' the term $(x_2 : T_2) \ldots (x_n : T_n).\star$.

Since $\Gamma \vdash A : \Box$ we have $\Gamma \vdash T_1 : p$ and $\Gamma, x : T_1 \vdash A' : \Box$ by Generation Lemma. We have also $\Gamma' \vdash T_1 : p$, $\Gamma', x : T_1 \vdash A' : \Box$ and by induction hypothesis $m(\Gamma \vdash T_1) = m(\Gamma' \vdash T_1)$ and $m(\Gamma, x : T_1 \vdash A') = m(\Gamma', x : T_1 \vdash A')$. Consequently, $m(\Gamma \vdash A) = m(\Gamma' \vdash A)$. $\Box$

Lemma 5.1.9 *Let Γ be an environment and A, B be Γ-non-objects such that $\Gamma \vdash A \overset{*}{\leftrightarrow} B$. Then $m(\Gamma \vdash A) = m(\Gamma \vdash B)$.*

Proof
First note that A and B must be both $\Box$, Γ-kinds or Γ-types. If A is a $\Box$ or a Γ-type then $m(\Gamma \vdash A) = 0 = m(\Gamma \vdash B)$. Otherwise A is a Γ-kind, $A = \overrightarrow{(x : T)}.\star$, $B = \overrightarrow{(x : U)}.\star$ and n is the length of $\overrightarrow{x : T}$ and $\overrightarrow{x : U}$. Let us proceed by induction on n.

If $n = 0$ then $m(\Gamma \vdash A) = 1 = m(\Gamma \vdash B)$. Otherwise let us call A' and B' the terms $(x_2 : T_2) \ldots (x_n : T_n).\star$ and $(x_2 : U_2) \ldots (x_n : U_n).\star$ respectively.

Since $\Gamma \vdash A \overset{*}{\leftrightarrow} B$ we have also $\Gamma \vdash T_1 \overset{*}{\leftrightarrow} U_1$ and $\Gamma, x : T_1 \vdash A' \overset{*}{\leftrightarrow} B'$. By induction hypothesis:

$$m(\Gamma \vdash T_1) = m(\Gamma \vdash U_1)$$
$$m(\Gamma, x : T_1 \vdash A') = m(\Gamma, x : T_1 \vdash B')$$

and by the previous lemma $m(\Gamma, x : T_1 \vdash B') = m(\Gamma, x : U_1 \vdash B')$. Consequently, by definition of the measure $m(\Gamma \vdash A) = m(\Gamma \vdash B)$. $\Box$

Lemma 5.1.10 *Let Γ, Δ be environments, A a Γ-non-object and ρ a well-typed substitution from Γ to Δ. Then $m(\Gamma \vdash A) = m(\Delta \vdash A\rho)$.*

Proof
First note that if A is a $\Box$, a Γ-kind or a Γ-type then $A\rho$ is respectively a $\Box$, a

Δ-kind or a Δ-type. If A is a $\Box$ or a Γ-type then $m(\Gamma \vdash A) = 0 = m(\Delta \vdash A\rho)$. Otherwise A is a Γ-kind, $A = \overrightarrow{(x:T)}.\star$ and n is the length of $\overrightarrow{x:T}$. Let us proceed by induction on n.

If $n = 0$ then $m(\Gamma \vdash A) = 1 = m(\Delta \vdash A\rho)$. Otherwise, let us call A' the term $(x_2 : T_2) \ldots (x_n : T_n).\star$.

Since $\Gamma \vdash A : \Box$ we have $\Gamma \vdash T_1 : p$ and $\Gamma, x : T_1 \vdash A' : \Box$ by Generation Lemma. By induction hypothesis:

$$m(\Gamma \vdash T_1) = m(\Delta \vdash T_1\rho)$$
$$m(\Gamma, x : T_1 \vdash A') = m(\Delta, x : T_1\rho \vdash A'\rho)$$

Consequently, by definition of the measure $m(\Gamma \vdash A) = m(\Delta \vdash A\rho)$. $\Box$

Lemma 5.1.11 *Let Γ, Γ' be environments such that $\Gamma \subseteq \Gamma'$ and A a Γ-non-object. Then $m(\Gamma \vdash A) = m(\Gamma' \vdash A)$.*

Proof
The desired equality follows from the fact that if A is a $\Box$, a kind or a type in environment Γ then A is respectively a $\Box$, a kind or a type in environment Γ'. $\Box$

Lemma 5.1.12 *For every Δ and Δ-non-object 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 $m(\Delta \vdash K)$, 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 $\Box$, a kind, or a type, then $can_{\Delta,A}$ equals $\mathcal{SN}_{\Delta,A}$ and is nonempty because $\Delta \vdash \star : \Box$ and $\Delta, x : A \vdash x : A$ for $A \neq \Box$. It is easy to check that the corresponding properties ((S1) ... (S3) from the definition of the family) are verified (see Lemma 5.1.3). 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 $\Delta' \vdash A\,M \overset{*}{\leftrightarrow} A'\,M$. $\square$

Let us prove some more properties of $\mathcal{SN}_\Delta$ that will be later needed in Lemmas 5.1.17, 5.4.7 and 5.5.2.

Lemma 5.1.13 *If $\Delta' \vdash a \in \mathcal{SN}_\Delta$, $\Delta' \vdash c \in \mathcal{SN}_\Delta$, $\Delta' \vdash b[c/x] \in \mathcal{SN}_\Delta$ and $(\lambda x\!:\!a.b)\,c$ is a Δ'-term then $\Delta' \vdash \lambda x\!:\!a.b \in \mathcal{SN}_\Delta$,*

Proof

Let us suppose that there is an infinite sequence of $\to_{sn}^{\Delta}$ reductions $(\Delta_j \vdash \lambda x\!: a_j.b_j)_{j=0}^{\infty}$ with $\Delta_0 = \Delta'$ and $a_0 = a$, $b_0 = b$. Since $\Delta' \vdash a \in \mathcal{SN}_\Delta$ there must be an index k from which reduction do not occur on a's. Consequently for $j > k$ we have $\Delta_{j+1}, x : a_k \vdash b_j \to b_{j+1}$ with $\Delta_k \vdash a \overset{*}{\leftrightarrow} a_k$. If we substitute c on x we get an infinite sequence of $\to_{sn}^{\Delta}$ reductions $(\Delta_j \vdash b_j[c/x])_{j=k}^{\infty}$. But $\Delta_k \vdash b_k[c/x] \in \mathcal{SN}_\Delta$ because $\Delta' \vdash b[c/x] \in \mathcal{SN}_\Delta$. Contradiction. $\square$

Lemma 5.1.14 *Let $s : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).\star$ be a type constructor and $\Delta_1 \subseteq \ldots \Delta_r \subseteq \Delta_{r+n}$ be environments. If for all i $\Delta_i \vdash a_i \in \mathcal{SN}_{\Delta_i, P_i[\vec{a}/\vec{p}]}$ and $\Delta_{r+i} \vdash t_i \in \mathcal{SN}_{\Delta_{r+i}, b_i[\vec{a}/\vec{p}][\vec{t}/\vec{x}]}$ then $\Delta_{r+n} \vdash s(\vec{a})\vec{t} \in \mathcal{SN}_{\Delta_{r+n}}$.*

Proof
Let us suppose that there is an infinite sequence of $\rightarrow_{sn}^{\Delta_{r+n}}$ reductions $(\nabla_j \vdash s(\vec{a}^j \vec{t}^j))_{j=0}^{\infty}$ with $\nabla_0 = \Delta_{r+n}$ and $\vec{a}^0 = \vec{a}$, $\vec{t}^0 = \vec{t}$. Since $\vec{\nabla}$ extends Δ_{r+n} it also extends any Δ_i. Since $\Delta_1 \vdash a_1 \in \mathcal{SN}_{\Delta_1}$ there must be an index k_1 from which reduction do not occur on a_1's. Considering $(\nabla_j \vdash s(\vec{a}^j \vec{t}^j))_{j=k_1}^{\infty}$ we find k_2 such that the reductions do not occur on a_1's and a_2's. After repeating this step $r+n$ times, reductions cannot occur at all. Contradiction. □

Lemma 5.1.15 *For every Δ-term $(x:a).b$, if $\Delta \vdash a \in \mathcal{SN}_\Delta$ and $\Delta, z : a \vdash b[z/x] \in \mathcal{SN}_{\Delta,z:a}$ then $\Delta \vdash (x:a).b \in \mathcal{SN}_\Delta$.*

Proof
Let us suppose that there is an infinite sequence of $\rightarrow_{sn}^{\Delta}$ reductions $(\Delta_j \vdash (x:a_j).b_j)_{j=0}^{\infty}$ with $\Delta_0 = \Delta$ and $a_0 = a$, $b_0 = b$. Without loss of generality let us assume that $z \notin \Delta_j$ for any j.

Every Δ_j may be seen as Δ, ∇_j. Let us define Δ'_j as $\Delta, z : a, \nabla_j$. If we consider the sequence $(\Delta'_j \vdash (x : a_j).b_j)_{j=0}^{\infty}$ there must be an index k from which reduction do not occur on a's. Consequently for all $j > k$ we have $\Delta'_{j+1}, x : a_k \vdash b_j \rightarrow b_{j+1}$ with $\Delta'_k \vdash a \overset{*}{\leftrightarrow} a_k$. If we substitute z on x we get $\Delta'_{j+1} \vdash b_j[z/x] \rightarrow b_{j+1}[z/x]$, i.e an infinite sequence of $\rightarrow_{sn}^{\Delta,z:A}$ reductions $(\Delta'_j \vdash b_j[z/x])_{j=k}^{\infty}$. But $\Delta'_k \vdash b_k[z/x] \in \mathcal{SN}_{\Delta,z:A}$ because $\Delta, z : A \vdash b[z/x] \in \mathcal{SN}_{\Delta,z:A}$. Contradiction. □

Lemma 5.1.16 *Subterms of strongly normalizing terms are strongly normalizing. In particular:*

1. *if $\Delta \vdash (x:a).b \in \mathcal{SN}_\Delta$ then $\Delta \vdash a \in \mathcal{SN}_\Delta$,*
2. *if $\Delta \vdash a\, b \in \mathcal{SN}_\Delta$ then $\Delta \vdash a, \Delta \vdash b \in \mathcal{SN}_\Delta$,*
3. *if $\Delta \vdash c(\vec{a}) \in \mathcal{SN}_\Delta$, $c \in \mathcal{CS}$ then for every i $\Delta \vdash a_i \in \mathcal{SN}_\Delta$.*

Proof
Let us suppose that there is an infinite sequence of $\rightarrow_{sn}^{\Delta}$ reductions $(\Delta_j \vdash a_j)_{j=0}^{\infty}$ with $\Delta_0 = \Delta$ and $a_0 = a$. Then we have also an infinite reduction sequence $(\Delta_j \vdash (x:a_j).b)_{j=0}^{\infty}$ which contradicts the fact $\Delta \vdash (x:a).b \in \mathcal{SN}_\Delta$.

The same reasoning applies to application and constructor. □

Lemma 5.1.17 *Let Δ be an environment, A a Δ-non-object and C a candidate from $\mathcal{C}_{\Delta,A}$. For every $\Delta' \supseteq \Delta$, if $\Delta' \vdash b[c/x] \in C$, $\Delta' \vdash a \in \mathcal{SN}_\Delta$, $\Delta' \vdash c \in \mathcal{SN}_\Delta$ and $\Delta' \vdash @(\lambda x\!:\!a.b, c) : A$ then $\Delta' \vdash @(\lambda x\!:\!a.b, c) \in C$.*

Proof
By induction on $(\Delta' \vdash \lambda x : a.b, \Delta' \vdash c)$ ordered by $(\rightarrow_{sn}^{\Delta}, \rightarrow_{sn}^{\Delta})_{lex}$ (note that $\Delta' \vdash \lambda x\!:\!a.b \in \mathcal{SN}_\Delta$ by Lemma 5.1.13).

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

- $\Delta'' \vdash @(\lambda x\!:\!a'.b, c)$, where $\Delta'' \vdash a \rightarrow a'$. Since $\Delta'' \vdash b[c/x] \in C$, by (S3) applied to the assumption $\Delta' \vdash b[c/x] \in C$, by induction hypothesis we get $\Delta'' \vdash @(\lambda x\!:\!a'.b, c) \in C$,

- $\Delta'' \vdash @(\lambda x : a.b', c)$, where $\Delta'', x : a \vdash b \rightarrow b'$. Obviously, $\Delta'' \vdash b[c/x] \rightarrow b'[c/x]$ and by (S3) and (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 4.2.14, $\Delta'' \vdash b[c/x] \rightarrow^* b[c'/x]$ 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 (S3) applied to the assumption $\Delta' \vdash b[c/x] \in C$. $\square$

Lemma 5.1.18 *Let Δ be an environment, A a Δ-non-object and C a candidate from $\mathcal{C}_{\Delta,A}$. For every environment Δ', variable x and Δ'-terms $\vec{N}$ from $\mathcal{SN}_\Delta$ if $\Delta' \vdash x\vec{N} : A$ then $\Delta' \vdash x\vec{N} \in C$.*

Proof
By induction on $\vec{N}$ ordered by $(\rightarrow_{sn}^{\Delta})_{lex}$.

Since $x\vec{N}$ is neutral, it is sufficient to show that every reduct of $\Delta'' \vdash x\vec{N}$, $\Delta'' \supseteq \Delta'$ belongs to C. All immediate reducts of $\Delta'' \vdash x\vec{N}$ are of the form $\Delta'' \vdash x\vec{M}$, where $\Delta'' \vdash N_i \rightarrow M_i$ 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 well-formed environments* Γ, Δ, *and a well-typed substitution* $\rho : \Gamma \to \Delta$, *a* candidate assignment compatible with ρ *is a function* ξ *from* $\mathcal{V}ar^{\Box} \cap dom(\Gamma)$ *to the set of candidates, such that, for every variable* $x \in \mathcal{V}ar^{\Box} \cap dom(\Gamma)$, $x\xi \in \mathcal{C}_{\Delta,x\rho}$.

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

Now we can define the interpretation $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ where Γ, Δ are environments, A is a Γ-non-object, $\rho : \Gamma \to \Delta$ 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* $y \in \mathcal{V}ar^{\Box}$.

$$
\begin{array}{rcl}
[\![\Gamma \vdash \Box]\!]_{\rho,\xi,\Delta} & = & \mathcal{SN}_{\Delta,\Box} \\
[\![\Gamma \vdash \star]\!]_{\rho,\xi,\Delta} & = & \mathcal{SN}_{\Delta,\star} \\
[\![\Gamma \vdash y]\!]_{\rho,\xi,\Delta} & = & y\xi \\
[\![\Gamma \vdash A\,B]\!]_{\rho,\xi,\Delta} & = & [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}(\Delta \vdash B\rho, [\![\Gamma \vdash B]\!]_{\rho,\xi,\Delta}) \\
[\![\Gamma \vdash A\,N]\!]_{\rho,\xi,\Delta} & = & [\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}(\Delta \vdash N\rho)
\end{array}
$$

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

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-non-object 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 indeed 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 $x \in \mathit{dom}(\Gamma)$, and ξ a candidate assignment compatible with ρ. If A is a Γ-non-object that does not contain symbols from $\mathcal{TC}$ then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash A]\!]_{\rho',\xi,\Delta}$.*

Proof
First note that ξ is compatible with ρ' and consequently the interpretation $[\![\Gamma \vdash A]\!]_{\rho',\xi,\Delta}$ is correct. We do the proof by induction on the structure of A.

1. If A is $\Box$, $\star$ or $y \in \mathcal{V}ar^{\Box}$ then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ equals $\mathcal{SN}_{\Delta,\Box}$, $\mathcal{SN}_{\Delta,\star}$ or $A\xi$ respectively, and obviously this equals $[\![\Gamma \vdash A]\!]_{\rho',\xi,\Delta}$.

2. If $A = P\,Q$ we do the proof only for the case where $Q \in \mathit{Constr}$ (the proof for $Q \in \mathit{Obj}$ is very similar).

   By induction hypothesis, we have:

$$\begin{aligned}
[\![\Gamma \vdash P\,Q]\!]_{\rho,\xi,\Delta} &= [\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta}) \\
&= [\![\Gamma \vdash P]\!]_{\rho',\xi,\Delta}(\Delta \vdash Q\rho', [\![\Gamma \vdash Q]\!]_{\rho',\xi,\Delta}) \\
&= [\![\Gamma \vdash P\,Q]\!]_{\rho',\xi,\Delta}
\end{aligned}$$

The second equality is valid because $[\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash Q]\!]_{\rho',\xi,\Delta}$ by induction hypothesis and because $\Delta \vdash Q\rho \overset{*}{\leftrightarrow} Q\rho'$ by the hypothesis about ρ and ρ'.

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

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

equals

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

But $\Delta' \vdash ((x:B).D)\rho \overset{*}{\leftrightarrow} ((x:B).D)\rho'$ and consequently if $\Delta' \vdash M \,:\, ((x:B).D)\rho$ then also $\Delta' \vdash M \,:\, ((x:B).D)\rho'$. Moreover, by induction hypothesis $[\![\Gamma \vdash B]\!]_{\rho,\xi|_{\Delta''},\Delta''} = [\![\Gamma \vdash B]\!]_{\rho',\xi|_{\Delta''},\Delta''}$ and $[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'/x],\Delta''} = [\![\Gamma, x : B \vdash D]\!]_{\rho'\cup[A/x],\xi|_{\Delta''}\cup[C'/x],\Delta''}$.

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

$$\begin{gathered}[\![\Gamma \vdash \lambda x:B.D]\!]_{\rho,\xi,\Delta} = \lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x:B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'} \\ \text{a function with domain} \\ \{(\Delta' \vdash A, C) \mid \Delta' \vdash A \,:\, B\rho,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',A}\}\end{gathered}$$

and we have to check that it equals

$$\begin{gathered}[\![\Gamma \vdash \lambda x:B.D]\!]_{\rho',\xi,\Delta} = \lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x:B \vdash D]\!]_{\rho'\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'} \\ \text{a function with domain} \\ \{(\Delta' \vdash A, C) \mid \Delta' \vdash A \,:\, B\rho',\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',A}\}\end{gathered}$$

But $\Delta' \vdash A \,:\, B\rho$ is equivalent to $\Delta' \vdash A \,:\, B\rho'$ and $[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'} = [\![\Gamma, x : B \vdash D]\!]_{\rho'\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'}$ by induction hypothesis. $\square$

Lemma 5.2.4 (Correctness of interpretation) *Let* Γ, Δ *be environments,* A *a* Γ*-non-object 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*

- $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,A\rho}$ *and*
- *for every* $\Delta' \supseteq \Delta$ $\;[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}|_{\Delta'} = [\![\Gamma \vdash A]\!]_{\rho,\xi|_{\Delta'},\Delta'}$.

Proof
We will prove both statements of the lemma in parallel. 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*.

   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 Generation Lemma 4.1.8. 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

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

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

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

is a nonempty set of strongly normalizing terms and that it satisfies conditions (S1), ... (S3) of Definition 5.1.5.

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 (see Lemma 5.1.16).

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 5.1.18 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 every $C' \in \mathcal{C}_{\Delta'',D(\rho\cup[A/x])}$. 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 to 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 prove that $\Delta' \vdash a \in [\![\Gamma \vdash (x:K).D]\!]_{\rho,\xi,\Delta}$. We will show that $\Delta'' \vdash a\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ for any Δ'', A and C such that $\Delta'' \supseteq \Delta'$, $\Delta'' \vdash A \in [\![\Gamma \vdash K]\!]_{\rho,\xi|_{\Delta''},\Delta''}$, $C \in \mathcal{C}_{\Delta'',A}$. By the definition of interpretation for a product it will give the desired conclusion. The proof will be done by induction on $\Delta'' \vdash A$ in $\to_{sn}^{\Delta}$ ($\Delta'' \vdash A \in \mathcal{SN}_{\Delta}$ because it belongs to some interpretation).

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. Since a is neutral, every reduct of $\Delta''' \vdash a\,A$, $\Delta''' \supseteq \Delta''$ is of the form

- $\Delta''' \vdash a'\,A$, where $\Delta''' \vdash a \to a'$. Then $\Delta''' \vdash a'\,A \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$ is an easy consequence of the assumption about reducts of a, namely $\Delta''' \vdash a' \in [\![\Gamma \vdash (x\!:\!K).D]\!]_{\rho,\xi,\Delta}$.
- $\Delta''' \vdash a\,A'$, 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 $\Delta''' \vdash a\,A' \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta'''}$. Since $[\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta'''} = [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}|_{\Delta'''}$ by induction hypothesis, we get the desired conclusion $\Delta''' \vdash a\,A' \in [\![\Gamma, x : K \vdash D]\!]_{\rho\cup[A/x],\xi\cup[C/x],\Delta''}$.

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

$$\begin{aligned}
[\![\Gamma \vdash \lambda x\!:\!K.B]\!]_{\rho,\xi,\Delta} = \lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x : K \vdash B]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'} \\
\text{a function with domain} \\
\{(\Delta' \vdash A, C) \mid \Delta' \vdash A : K\rho,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',A}\}
\end{aligned}$$

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 Generation Lemma 4.1.8, $\Gamma \vdash \lambda x : K.B \; : \; (x : K).D$ 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.

We are left to show the second property. By the definition of interpretation and restriction:

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

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

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

In the sequel, we will often write $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ for a candidate assignment ξ that is defined only on $\mathcal{FV}(A) \cap \mathcal{V}ar^{\square}$, even though formally ξ should be given on all variables from $\Gamma \cap \mathcal{V}ar^{\square}$. 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-non-object and ρ a well-typed substitution from Γ to Δ. Moreover, let us suppose that $z \in dom(\Gamma) \cap \mathcal{V}ar^{\Box}$, $z \notin \mathcal{FV}(A)$. Then for every ξ_1, ξ_2, candidate assignments compatible with ρ, that differ only on that z, $[\![\Gamma \vdash A]\!]_{\rho,\xi_1,\Delta} = [\![\Gamma \vdash A]\!]_{\rho,\xi_2,\Delta}$.*

Proof
We do the proof by induction on the structure of A.

1. If A is $\Box$, $\star$ or $y \in \mathcal{V}ar^{\Box}$ then $[\![\Gamma \vdash A]\!]_{\rho,\xi_1,\Delta}$ equals $\mathcal{SN}_{\Delta,\Box}$, $\mathcal{SN}_{\Delta,\star}$ or $A\xi_1$ respectively, and obviously this equals $[\![\Gamma \vdash A]\!]_{\rho,\xi_2,\Delta}$, as $y \in \mathcal{FV}(y)$.

2. If $A = P\,Q$ we do the proof only for the case where $Q \in$ *Constr.*

   By induction hypothesis, we have:

$$\begin{aligned}
[\![\Gamma \vdash P\,Q]\!]_{\rho,\xi_1,\Delta} &= [\![\Gamma \vdash P]\!]_{\rho,\xi_1,\Delta}(\Delta \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi_1,\Delta}) \\
&= [\![\Gamma \vdash P]\!]_{\rho,\xi_2,\Delta}(\Delta \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi_2,\Delta}) \\
&= [\![\Gamma \vdash P\,Q]\!]_{\rho,\xi_2,\Delta}
\end{aligned}$$

   The second equality is valid by induction hypothesis and because $z \notin \mathcal{FV}(P, Q)$.

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

$$\begin{gathered}
\{\Delta' \vdash M \mid \Delta' \vdash M \,:\, ((x\!:\!B).D)\rho,\ \Delta' \supseteq \Delta,\ \text{and} \\
\forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma \vdash B]\!]_{\rho,\xi_1|_{\Delta''},\Delta''},\ \forall C' \in \mathcal{C}_{\Delta'',A}, \\
\Delta'' \vdash M\,A \in [\![\Gamma, x : B \vdash D]\!]_{\rho \cup [A/x],\xi_1|_{\Delta''} \cup [C'/x],\Delta''}\}
\end{gathered}$$

   equals

$$\begin{gathered}
\{\Delta' \vdash M \mid \Delta' \vdash M \,:\, ((x\!:\!B).D)\rho,\ \Delta' \supseteq \Delta,\ \text{and} \\
\forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma \vdash B]\!]_{\rho,\xi_2|_{\Delta''},\Delta''},\ \forall C' \in \mathcal{C}_{\Delta'',A}, \\
\Delta'' \vdash M\,A \in [\![\Gamma, x : B \vdash D]\!]_{\rho \cup [A/x],\xi_2|_{\Delta''} \cup [C'/x],\Delta''}\}
\end{gathered}$$

   Since $z \notin \mathcal{FV}(B, D)$, by induction hypothesis we get $[\![\Gamma \vdash B]\!]_{\rho,\xi_1|_{\Delta''},\Delta''} = [\![\Gamma \vdash B]\!]_{\rho,\xi_2|_{\Delta''},\Delta''}$ and $[\![\Gamma, x : B \vdash D]\!]_{\rho \cup [A/x],\xi_1|_{\Delta''} \cup [C'/x],\Delta''} = [\![\Gamma, x : B \vdash D]\!]_{\rho \cup [A/x],\xi_2|_{\Delta''} \cup [C'/x],\Delta''}$ and the proof is finished.

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

$$[\![\Gamma \vdash \lambda x\!:\!B.D]\!]_{\rho,\xi_1,\Delta} = \lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x : B \vdash D]\!]_{\rho \cup [A/x],\xi_1|_{\Delta'} \cup [C/x],\Delta'}$$

   a function with domain

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

and we have to check that it equals

$$\begin{array}{l} [\![\Gamma \vdash \lambda x\!:\!B.D]\!]_{\rho,\xi_2,\Delta} = \lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi_2|_{\Delta'}\cup[C/x],\Delta'} \\ \qquad\qquad\text{a function with domain} \\ \qquad\qquad\{(\Delta' \vdash A, C) \mid \Delta' \vdash A \,:\, B\rho,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',A}\} \end{array}$$

But $[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi_1|_{\Delta'}\cup[C/x],\Delta'} = [\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi_2|_{\Delta'}\cup[C/x],\Delta'}$ by induction hypothesis. $\square$

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 in 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 $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$ to denote the interpretation of s. All the definitions below assume that we want to define $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$ for s that is a part of the inductive definition

$$Ind[(p_1 : P_1)\dots(p_r : P_r)](s : A := c_1 : C_1, \dots c_m : C_m)$$

from Σ, with $A = (x_1 : b_1)\dots(x_n : b_n).\star$ and $C_i = (z_1^i : d_1^i)\dots(z_{k_i}^i : d_{k_i}^i).s(\vec{p})w_1^i \dots w_n^i$. 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 $(p_1 : P_1)\dots(p_r : P_r)(z_1^i : d_1^i)\dots(z_{k_i}^i : d_{k_i}^i).s(\vec{p})w_1^i \dots w_n^i$. 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 $L^{\to}$. It is a set of vectors of pairs, each pair consisting of a sequent and (possibly) a reducibility candidate.

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

If $r+n=0$ let us define Dom to be 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 strongly normalizing sequents that do not reduce to constructor headed form. Note that these sequents are defined in the environment ∇^n ($\nabla^0 = \Delta$).

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

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

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

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 $\{\varphi_i\}_{i\in I}$, 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 $\varphi'(e) \subseteq \varphi(e)$. $\square$

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 $s : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).\star$ and $c_i : (p_1 : P_1) \ldots (p_r : P_r)(z_1^i : d_1^i) \ldots (z_{k_i}^i : d_{k_i}^i).s(\vec{p})w_1^i \ldots w_n^i$ 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:*

$$F(\varphi) = \lambda\langle\overrightarrow{(\Delta \vdash a, U)}, \overrightarrow{(\nabla \vdash t, W)}\rangle.\{\Delta' \vdash M \in \mathcal{SN}_{\nabla^n, s(\vec{a})\vec{t}} \mid$$
$$\text{if } \Delta'' \vdash M \rightarrow^* c_i(\vec{a'}, \vec{N}),\ c_i \text{ being a constructor of } s \text{ and } \Delta'' \supseteq \Delta', \text{ then}$$
$$\Delta'' \vdash a'_j \in \mathit{Inter}(\varphi)(p_1 : P_1, \dots p_{j-1} : P_{j-1} \vdash P_j)_{[\vec{a}/\vec{p}],[\vec{U}|_{\Delta''}/\vec{p}],\Delta''} \text{ and}$$
$$\Delta'' \vdash N_j \in \mathit{Inter}(\varphi)(\overrightarrow{p : P}, z_1^i : d_1^i, \dots z_{j-1}^i : d_{j-1}^i \vdash d_j^i)_{[\vec{a}/\vec{p}]\cup[\vec{N}/\vec{z^i}],[\vec{U}|_{\Delta''}/\vec{p}],\Delta''}\}$$

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

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

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

   (b) *if T is a type then*

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

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 $[\![... \vdash a'_j]\!]$ (which would not give a consistent definition). For this reason $[\![... \vdash N_j]\!]$ cannot appear in a candidate assignment part of *Inter*. This is achieved by restricting the form of d^i_j (recall that by the definition of the well-formed signature every 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 $(y_1 : T_1) \ldots (y_k : T_k).s(\vec{p})v_1, \ldots v_n$ where s does not occur in any v_i, T_i and $\vec{p} = dom(\Gamma_p)$.

If $A = s(\vec{p})v_1, \ldots v_n$ 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

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

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

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 constructors of s (and hence the definition of F) 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.

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

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 $(p_1 : P_1) \ldots (p_r : P_r).(x_1 : b_1) \ldots (x_n : b_n).\star$ and let μF be the least fixpoint of F used to compute the interpretation of s. Then, for every $(\Delta \vdash a_1, U_1), \ldots (\Delta \vdash a_r, U_r)$, such that for every $i = 1, \ldots r$,*

- *$\Delta \vdash a_i : P_i[a_1/p_1, \ldots, a_{i-1}/p_{i-1}]$,*
- *$U_i = \emptyset$ if $a_i \in Obj$ and $U_i \in \mathcal{C}_{\Delta,a_i}$ otherwise,*

we have $\mu F(\Delta \vdash a_1, U_1), \ldots (\Delta \vdash a_r, U_r) \in \mathcal{C}_{\Delta,s(\vec{a})}$ and for all $\Delta' \supseteq \Delta$ $(\mu F\overrightarrow{(\Delta \vdash a, U)})|_{\Delta'} = \mu F(\Delta \vdash a_1, U_1), \ldots (\Delta \vdash a_{r-1}, U_{r-1})(\Delta' \vdash a_r, U_r|_{\Delta'})$.

Proof

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

- $\Delta = \nabla^0 \subseteq \nabla^1 \subseteq, \ldots \nabla^n$ and
- for every $j = 1, \ldots n$, $\nabla^j \vdash t_j : b_j[\vec{a}/\vec{p}][t_1/x_1, \ldots, t_{j-1}/x_{j-1}]$ and
- for every $j = 1, \ldots n$, $W_j = \emptyset$ 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
- for all $\Delta' \supseteq \nabla^i$, $(\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \dots (\nabla^i \vdash t_i, W_i))|_{\Delta'} = \mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \dots (\nabla^{i-1} \vdash t_{i-1}, W_{i-1})(\Delta' \vdash t_i, W_i|_{\Delta'})$.

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 the definition 5.1.5 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 irreducible 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 a constructor headed term, then $\Delta' \vdash M' \in \mu Fe$.

Otherwise, $\Delta'' \vdash M' {\to^*} c_i(\vec{a'}, \vec{N})$, for some $\Delta'' \supseteq \Delta'$, $c_i : \overrightarrow{(p : P)}\overrightarrow{(z : d)}.s(\vec{p})\vec{w}$ being a constructor of s, and we have to check that

$$\Delta'' \vdash a'_j \in Inter(\varphi)(\overrightarrow{p : P} \vdash P_j)_{[\vec{a}/\vec{p}],[\vec{U}/\vec{p}]|_{\Delta''},\Delta''} \quad \text{and}$$
$$\Delta'' \vdash N_j \in Inter(\varphi)((\overrightarrow{p : P}, \overrightarrow{z : d}) \vdash d_j)_{[\vec{a}/\vec{p}][\vec{N}/\vec{z}],[\vec{U}/\vec{p}]|_{\Delta''},\Delta''}$$

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 ($\Delta'' \vdash M \to c_i(\vec{a'}, \vec{N})$, $\Delta'' \supseteq \Delta'$) and check some conditions for immediate subterms of $c_i(\vec{a'}, \vec{N})$ (see above). Since M is neutral there must be M', such that $\Delta'' \vdash M \to M' {\to^*} c_i(\vec{a'}, \vec{N})$. Of course, $\Delta'' \vdash M'$ is a reduct of $\Delta' \vdash M$. Consequently, the validity of the required conditions follows from the validity of the same conditions for $\Delta'' \vdash M' \in \mu Fe$.

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 $\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \ldots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})$ verifies (P1), i.e. is a function with a proper domain $(\{(\Delta' \vdash M, C) \mid \Delta' \vdash M : b_k[\vec{a}/\vec{p}][\vec{t}/\vec{x}],\ \Delta' \supseteq \nabla^{k-1}$ and $C \in \mathcal{C}_{\Delta', M}\})$ and a proper codomain, as $\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \ldots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M, C)$ belongs to $\mathcal{C}_{\Delta', s(\vec{a})t_1, \ldots t_{k-1}M}$ by induction hypothesis. To check (P2) it is sufficient to show that

$$\begin{aligned}\mu F\overrightarrow{(\Delta \vdash a, U)}&(\nabla^1 \vdash t_1, W_1), \ldots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M_1, C) = \\ &\mu F\overrightarrow{(\Delta \vdash a, U)}(\nabla^1 \vdash t_1, W_1), \ldots (\nabla^{k-1} \vdash t_{k-1}, W_{k-1})(\Delta' \vdash M_2, C)\end{aligned}$$

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

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

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

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

The two lemmas given below show the relationship between $Inter(\mu F)$ and the real interpretation. They are the basis of Lemma 5.3.7 which expresses the idea that reaching under a constructor is safe from the interpretation point of view.

Lemma 5.2.13 *Let Γ, Δ be environments, ρ a well-typed substitution from Γ to Δ, and ξ a candidate assignment compatible with ρ. Let $s : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).\star$ be an inductive type and $c : (p_1 : P_1) \ldots (p_r : P_r)(z_1 : d_1) \ldots (z_k : d_k).s(\vec{p})w_1 \ldots w_n$ its constructor. If $\vec{a}$, $\vec{N}$ are such that for every $i = 1, \ldots r$, $\Gamma \vdash a_i : P_i[\vec{a}/\vec{p}]$ and for every $j = 1, \ldots k$,*

$\Delta \vdash N_j : d_j[\vec{a}\rho/\vec{p}][\vec{N}/\vec{z}]$ *then for all* i, j:

$$Inter(\mu F)(\overrightarrow{p:P} \vdash P_i)_{[\vec{a}\rho/\vec{p}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta} = [\![\overrightarrow{p:P} \vdash P_i]\!]_{[\vec{a}\rho/\vec{p}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta}$$

$$Inter(\mu F)(\overrightarrow{p:P}, \overrightarrow{z:d} \vdash d_j)_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta} =$$
$$[\![\overrightarrow{p:P}, \overrightarrow{z:d} \vdash d_j]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta}$$

Proof

Let us denote $[\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}]$ by ζ. By the positivity condition imposed on types of constructors we know that $s \notin P_i$ and that d_j is of form $\overrightarrow{(y:T)}.s(\vec{p})\vec{v}$ with $s \notin \{\vec{T}, \vec{v}\}$.

By the definition of $Inter$, if $s \notin A$ then immediately $Inter(\mu F)(\hat{\Gamma} \vdash A)_{\hat{\rho},\hat{\xi},\hat{\Delta}} = [\![\hat{\Gamma} \vdash A]\!]_{\hat{\rho},\hat{\xi},\hat{\Delta}}$. We proceed by induction on the structure of d_j.

If $d_j = s(\vec{p})\vec{v}$ then

$$Inter(\mu F)(\overrightarrow{p:P}, \overrightarrow{z:d} \vdash d_j)_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta} =$$
$$\mu F\overrightarrow{(\Delta \vdash a\rho, p\zeta)}\overrightarrow{(\Delta \vdash v([\vec{a}\rho/\vec{p}] \cup [\vec{N}/\vec{z}]), z\zeta')}$$

where $z_i\zeta' = [\![\overrightarrow{p:P}, \overrightarrow{z:d} \vdash v_i]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta}$ if $z_i \in \mathcal{V}ar^{\Box}$. On the other hand, by the definition of interpretation of an application, $[\![\overrightarrow{p:P}, \overrightarrow{z:d} \vdash s(\vec{p})\vec{v}]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta}$ equals

$$[\![\overrightarrow{p:P}, \overrightarrow{z:d} \vdash s(\vec{p})]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta}\overrightarrow{(\Delta \vdash v([\vec{a}\rho/\vec{p}] \cup [\vec{N}/\vec{z}]), z\zeta')}$$

But by the definition of interpretation of s, $[\![\overrightarrow{p:P}, \overrightarrow{z:d} \vdash s(\vec{p})]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta}$ is equal to $\mu F\overrightarrow{(\Delta \vdash a\rho, p\zeta)}$ and this case of the lemma is proved.

If $d_j = (y:T).B$ and T is a kind then $Inter(\mu F)(\overrightarrow{p:P}, \overrightarrow{z:d} \vdash d_j)_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta}$ equals

$$\{\Delta' \vdash M \mid \Delta' \vdash M : ((y:T).B)[\vec{a}\rho/\vec{p}] \cup [\vec{N}/\vec{z}],\ \Delta' \supseteq \Delta, \text{ and } \forall \Delta'' \supseteq \Delta'$$
$$\forall \Delta'' \vdash N' \in [\![\overrightarrow{p:P}, \overrightarrow{z:d} \vdash T]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta|_{\Delta''},\Delta''} \ \ \forall C \in \mathcal{C}_{\Delta'',N'},$$
$$\Delta'' \vdash M\,N' \in Inter(\mu F)((\overrightarrow{p:P}, \overrightarrow{z:d}, y:T) \vdash B)_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}]\cup[N'/x],\zeta|_{\Delta''}\cup[C/x],\Delta''}\}$$

By induction hypothesis the last *Inter* equals

$$[\![\overrightarrow{p:P},\overrightarrow{z:d},y:T\vdash B]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}]\cup[N'/x],\zeta|_{\Delta''}\cup[C/x],\Delta''}$$

and we get $[\![\overrightarrow{p:P},\overrightarrow{z:d}\vdash(y{:}T).B]\!]_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],\zeta,\Delta}$.

If T is a type the proof is very similar. $\square$

Lemma 5.2.14 *Let Γ, Δ be environments, ρ a well-typed substitution from Γ to Δ, and ξ a candidate assignment compatible with ρ. Let $s:(p_1:P_1)\ldots(p_r:P_r)(x_1:b_1)\ldots(x_n:b_n).\star$ be an inductive type and $c:(p_1:P_1)\ldots(p_r:P_r)(z_1:d_1)\ldots(z_k:d_k).s(\vec{p})w_1\ldots w_n$ its constructor. Suppose that $\vec{a}$, $\vec{a'}$, $\vec{N}$, $\vec{u}$, $\vec{U}$ satisfy the following*

- *$\vec{u}$ are Γ-terms and $\Delta\vdash c(\vec{a'},\vec{N}):(s(\vec{a})\vec{u})\rho$ holds,*
- *$\vec{U}$ are such that $U_i\in\mathcal{C}_{\Delta,u_i\rho}$ if $u_i\notin\mathrm{Obj}$ and $U_i=\emptyset$ otherwise.*

Under the above assumptions $\Delta\vdash c(\vec{a'},\vec{N})\in[\![\Gamma\vdash s(\vec{a})]\!]_{\rho,\xi,\Delta}\overrightarrow{(\Delta\vdash u\rho,U)}$ if and only if

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

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

$$\Delta\vdash c(\vec{a'},\vec{N})\in\mu F\overrightarrow{(\Delta\vdash a\rho,[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta})}\overrightarrow{(\Delta\vdash u\rho,U)}$$

and by the definition of a fixpoint ($\mu F=F(\mu F)$)

- $\Delta\vdash a'_i\in \mathit{Inter}(\mu F)(\overrightarrow{p:P}\vdash P_i)_{[\vec{a}\rho/\vec{p}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta}$ for every $i=1,\ldots r$ and
- $\Delta\vdash N_i\in \mathit{Inter}(\mu F)(\overrightarrow{p:P},\overrightarrow{z:d}\vdash d_j)_{[\vec{a}\rho/\vec{p}]\cup[\vec{N}/\vec{z}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta}$ for every $i=1,\ldots k$.

Now by Lemma 5.2.13 we get

- $\Delta\vdash a'_i\in[\![\overrightarrow{p:P}\vdash P_i]\!]_{[\vec{a}\rho/\vec{p}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta}$ for every $i=1,\ldots r$ and

- $\Delta \vdash N_i \in [\![\overrightarrow{p : P}, \overrightarrow{z : d} \vdash d_j]\!]_{[\vec{a}\rho/\vec{p}] \cup [\vec{N}/\vec{z}], [\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}], \Delta}$ for every $i = 1, \ldots k$,

which are the desired conclusion.

For the reverse implication it is sufficient to reverse all the steps presented above. $\square$

All that was so far presented in this section concerned one inductive type s. To proceed we should prove again Lemmas 5.2.3 and 5.2.4, this time for all Γ-non-objects that may now contain symbols from $\mathcal{TC}$ up to s, s included. Their respective proofs should be extended by one case for $A = s(\vec{a})$ which trivially follows from Corollary 5.2.11 and Lemma 5.2.12.

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

Lemma 5.2.15 (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. If A is a Γ-non-object then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash A]\!]_{\rho',\xi,\Delta}$.*

Lemma 5.2.16 (Correctness of interpretation) *Let A be a Γ-non-object, $\rho : \Gamma \to \Delta$ and ξ a candidate assignment compatible with ρ. Then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} \in \mathcal{C}_{\Delta,A\rho}$.*

Some other useful properties of interpretation will be given in Section 5.3.

5.2.2 Ordering induced by interpretation of inductive types

Inductive construction of $[\![\Gamma \vdash s]\!]_{\rho,\xi,\Delta}$, as a least fixpoint of the monotone operator F, induces a well-founded ordering on sequents. This ordering is based on the comparison of ordinals, which correspond to the least number of iterations of F needed to reach a given sequent. A similar construction has been considered in [92].

Looking at the definition of $F(\varphi)((\overrightarrow{\Delta \vdash a, U}), (\overrightarrow{\nabla \vdash t, W}))$ we see that it does not depend on $\vec{W}$ and uses $\vec{t}$ only for typing. On the other hand $\vec{a}$ and $\vec{U}$ (the parameter part of the element of Dom) remain unchanged in all recursive calls of F. Consequently, the ordering induced by $((\overrightarrow{\Delta \vdash a, U}), (\overrightarrow{\nabla \vdash t, W}))$ depends really on $\vec{A} = (\overrightarrow{\Delta \vdash a, U})$ and it can compare terms of type $s(\vec{a})\vec{t}$ for different $\vec{t}$. The ordering will be called $>^{\Delta}_{deg(\vec{A})}$ and its domain $U_{(s,\Delta,\vec{A})}$.

We start by defining a transfinite iteration of F.

Definition 5.2.17 *Let φ be an element of the lattice $L^{\rightarrow}$. We define $F^{\nu}(\varphi)$ by transfinite induction on ν:*

- $F^0(\varphi) = \varphi$,
- *for every ordinal ν, $F^{\nu+1}(\varphi) = F(F^{\nu}(\varphi))$,*
- *for every set of ordinals $\mathfrak{A}$, $F^{lim(\mathfrak{A})}(\varphi)(e) = \bigcup_{\nu \in \mathfrak{A}} F^{\nu}(\varphi)(e)$.*

By the Tarski theorem, there is an ordinal ν such that $\mu F = F^{\nu}(Bot)$. Moreover, we may assign to every $\Delta \vdash u \in \mu F(e)$ the ordinal corresponding to the least number of iterations needed to reach this u.

Definition 5.2.18 *Let $\Delta \vdash u \in \mu F(e)$. We define the* degree $deg(\Delta \vdash u, e)$ *as the least ordinal ν such that $\Delta \vdash u \in F^{\nu}(Bot)(e)$.*

Lemma 5.2.19 *For every $\Delta \vdash u \in \mu F(e)$, $deg(\Delta \vdash u, e)$ is 0 if u does not reduce to constructor headed form, and a successor ordinal otherwise.*

Proof
The first part of the lemma is trivial. For the second one, if we suppose that $deg(\Delta \vdash u, e) = lim(\mathfrak{A})$ then

$$\Delta \vdash u \in F^{lim(\mathfrak{A})}(Bot)(e) = \bigcup_{\nu \in \mathfrak{A}} F^{\nu}(Bot)(e)$$

and consequently $\Delta \vdash u \in F^{\nu}(Bot)(e)$ for some $\nu < deg(\Delta \vdash u, e)$ which is impossible. $\square$

The lemmas below describe some basic properties of degree.

Lemma 5.2.20 *For every environments $\Delta \subseteq \hat{\Delta} \subseteq \nabla_1$, we have*

1. $\Delta' \vdash u \in \mu F(\overrightarrow{(\Delta \vdash a, U)}, \overrightarrow{(\nabla \vdash t, W)})$ *if and only if* $\Delta' \vdash u \in \mu F$ $(\overrightarrow{(\hat{\Delta} \vdash a, U|_{\hat{\Delta}})}, \overrightarrow{(\nabla \vdash t, W)})$, *and*

2. $deg(\Delta' \vdash u, (\overrightarrow{(\Delta \vdash a, U)}, \overrightarrow{(\nabla \vdash t, W)})) = deg(\Delta' \vdash u, (\overrightarrow{(\hat{\Delta} \vdash a, U|_{\hat{\Delta}})}, \overrightarrow{(\nabla \vdash t, W)}))$.

Proof
Both conclusions follow from the fact that for all φ we have

$$F(\varphi)(\overrightarrow{(\Delta \vdash a, U)}, \overrightarrow{(\nabla \vdash t, W)}) = F(\varphi)(\overrightarrow{(\hat{\Delta} \vdash a, U|_{\hat{\Delta}})}, \overrightarrow{(\nabla \vdash t, W)}) \quad \Box$$

Lemma 5.2.21 *Let $\Delta \vdash u \in \mu F(e)$. For every $\Delta' \supseteq \Delta$ we have $\Delta' \vdash u \in \mu F(e)$ and $deg(\Delta' \vdash u, e) \geq deg(\Delta \vdash u, e)$.*

Proof
First note that by lemma 5.2.12 $\mu F(e)$ is a candidate. By (S3) property of candidates $\Delta' \vdash u \in \mu F(e)$.

By the definition of degree we have $\Delta \vdash u \in F^{deg(\Delta \vdash u, e)}(Bot)(e)$. If $\Delta \vdash u$ does not reduce to constructor headed term in any environment bigger than Δ than $\Delta' \vdash u$ does not either and $deg(\Delta \vdash u, e) = deg(\Delta' \vdash u, e) = 0$. Otherwise $deg(\Delta \vdash u, e)$ is a successor ordinal $\nu + 1$ and whenever u reduces to $c(\vec{a})$ in some environment $\Delta'' \supseteq \Delta$ these a's belong to the $Inter(F^{\nu})$ applied to the appropriate arguments. Since $\Delta' \supseteq \Delta$, all constructor headed terms $c(\vec{b})$ we can reach from $\Delta' \vdash u$ are also reducts of $\Delta \vdash u$ and all these b's belong to $Inter(F^{\nu})$. Consequently, $\Delta' \vdash u \in F^{\nu+1}(Bot)(e)$ and by the definition of the degree $deg(\Delta' \vdash u, e) \leq \nu + 1 = deg(\Delta \vdash u, e)$. $\Box$

Lemma 5.2.22 *Let $\Delta' \vdash u \in \mu F(e)$. If $\Delta' \vdash u \rightarrow^{\Delta}_{sn} \Delta'' \vdash w$ then $\Delta'' \vdash w \in \mu F(e)$ and $deg(\Delta' \vdash u, e) \geq deg(\Delta'' \vdash w, e)$.*

Proof
By the definition of $\rightarrow^{\Delta}_{sn}$ we have $\Delta'' \supseteq \Delta'$ and $\Delta'' \vdash u \rightarrow w$.

First note that by lemma 5.2.12 $\mu F(e)$ is a candidate. By (S3) property for candidates $\Delta' \vdash u \in \mu F(e)$ implies $\Delta'' \vdash u \in \mu F(e)$. By (S1) property $\Delta'' \vdash w \in \mu F(e)$.

By Lemma 5.2.21 $deg(\Delta' \vdash u, e) \geq deg(\Delta'' \vdash u, e)$. We will show that $deg(\Delta'' \vdash u, e) \geq deg(\Delta'' \vdash w, e)$

By the definition of the degree we have $\Delta'' \vdash u \in F^{deg(\Delta'' \vdash u, e)}(Bot)(e)$. If u does not reduce to constructor headed term than w does not either and $deg(\Delta'' \vdash u, e) = deg(\Delta'' \vdash w, e) = 0$. Otherwise $deg(\Delta'' \vdash u, e)$ is a successor ordinal $\nu + 1$ and whenever u reduces to $c(\vec{a})$, these a's belong to the $Inter(F^{\nu})$ applied to the appropriate arguments. Since w is a reduct of u, all constructor headed terms $c(\vec{b})$ we can reach from w are also reducts of u and all these b's belong to $Inter(F^{\nu})$. Consequently, $\Delta'' \vdash w \in F^{\nu+1}(Bot)(e)$ and by the definition of the degree $deg(\Delta'' \vdash w, e) \leq \nu + 1 = deg(\Delta'' \vdash u, e)$.
□

The following definition presents the domain of the ordering $>^{\Delta}_{deg(\vec{A})}$. Given the parameter part $\vec{A}$ of Dom, $U_{(s,\Delta,\vec{A})}$ is the image of the subset $\{(\vec{A}, \_)\}$ of Dom with respect to μF.

Definition 5.2.23 *Let* $\vec{A} = \overrightarrow{(\Delta \vdash a, U)}$. *We define*

$$U_{(s,\Delta,\vec{A})} = \{\Delta' \vdash u \mid \Delta' \supseteq \Delta,\ \exists \vec{\nabla}, \vec{t}, \vec{W}\ (\vec{A}, \overrightarrow{(\nabla \vdash t, W)}) \in Dom \quad \Delta' \vdash u \in \mu F(\vec{A}, \overrightarrow{(\nabla \vdash t, W)})\}$$

Now we will prove that the degree of a sequent does not depend on the $\overrightarrow{(\nabla \vdash t, W)})$ components.

Lemma 5.2.24 *Let* $\Delta' \vdash u : s(\vec{a})\vec{t}$. *If* $\Delta' \vdash u \in U_{(s,\Delta,\vec{A})}$ *then* $\Delta' \vdash u \in \mu F(\vec{A}, \overrightarrow{(\Delta' \vdash t, W)})$ *for any* $\vec{W}$ *such that* $W_i \in \mathcal{C}_{\Delta', t_i}$ *if* t_i *is a non-object and* $W_i = \emptyset$ *otherwise.*

Proof
By the definition of F, for all φ $F(\varphi)(\vec{A}, \overrightarrow{(\nabla \vdash t, W)})$ does not depend on $\vec{W}$ and uses $\vec{t}$ only for type-checking $\Delta' \vdash u : s(\vec{a})\vec{t}$. Consequently, any $\vec{W}$ such that $W_i \in \mathcal{C}_{\Delta', t_i}$ if t_i is a non-object and $W_i = \emptyset$ will satisfy $\Delta' \vdash u \in \mu F(\vec{A}, \overrightarrow{(\Delta' \vdash t, W)})$. □

Lemma 5.2.25 *Let* $\Delta' \vdash u \in U_{(s,\Delta,\vec{A})}$. *If* $\vec{\nabla}^1, \vec{t}^1, \vec{W}^1$ *and* $\vec{\nabla}^2, \vec{t}^2, \vec{W}^2$ *are such that*

- $(\vec{A}, \overrightarrow{(\nabla^1 \vdash t^1, W^1)}) \in Dom$, $(\vec{A}, \overrightarrow{(\nabla^2 \vdash t^2, W^2)}) \in Dom$,

- $\Delta' \vdash u \in \mu F(\vec{A}, \overrightarrow{(\nabla^1 \vdash t^1, W^1)})$, $\Delta' \vdash u \in \mu F(\vec{A}, \overrightarrow{(\nabla^2 \vdash t^2, W^2)})$

then $deg(\Delta' \vdash u, (\vec{A}, \overrightarrow{(\nabla^1 \vdash t^1, W^1)})) = deg(\Delta' \vdash u, (\vec{A}, \overrightarrow{(\nabla^2 \vdash t^2, W^2)}))$.

Proof
First note that according to the definition of F we have $\Delta' \supseteq \nabla^1_n$ and $\Delta' \supseteq \nabla^2_n$, where n is the length of $\vec{\nabla}$.

Directly from the definition of F it follows that for all φ

$$(F(\varphi)(\vec{A}, \overrightarrow{(\nabla^1 \vdash t^1, W^1)}))|_{\Delta'} = F(\varphi)(\vec{A}, \overrightarrow{(\Delta' \vdash t^1, W^1|_{\Delta'})})$$

Similarly for $\vec{\nabla}^2$ and $\vec{t}^2$. Moreover, for all φ $F(\varphi)(\vec{A}, \overrightarrow{(\Delta' \vdash t^1, W^1|_{\Delta'})}) = F(\varphi)(\vec{A}, \overrightarrow{(\Delta' \vdash t^2, W^2|_{\Delta'})})$.

Summarizing, for all φ we have

$$F(\varphi)(\vec{A}, \overrightarrow{(\nabla^1 \vdash t^1, W^1)}) = F(\varphi)(\vec{A}, \overrightarrow{(\nabla^2 \vdash t^2, W^2)})$$

which implies the equality of degrees. $\Box$

Once we have this lemma we may define the degree of a sequent in $U_{(s,\Delta,\vec{A})}$ given only $\vec{A}$ part.

Definition 5.2.26 *Let* $\Delta' \vdash u \in U_{(s,\Delta,\vec{A})}$. *We define* $deg(\Delta' \vdash u, \vec{A})$ *as* $deg(\Delta' \vdash u, (\vec{A}, \overrightarrow{(\nabla \vdash t, W)}))$ *for any any* $\vec{\nabla}, \vec{t}, \vec{W}$ *such that* $(\vec{A}, \overrightarrow{(\nabla \vdash t, W)}) \in Dom$ *and* $\Delta' \vdash u \in \mu F(\vec{A}, \overrightarrow{(\nabla \vdash t, W)})$.

Now we may define the ordering $>^{\Delta}_{deg(\vec{A})}$ on $U_{(s,\Delta,\vec{A})}$:

Definition 5.2.27 *Given two sequents* $\Delta' \vdash u$, $\Delta'' \vdash w$ *from* $U_{(s,\Delta,\vec{A})}$ *we say that* u has a greater degree in $\vec{A}$ than w, *denoted by* $\Delta' \vdash u >^{\Delta}_{deg(\vec{A})} \Delta'' \vdash w$ *if*

$$deg(\Delta' \vdash u, \vec{A}) > deg(\Delta'' \vdash w, \vec{A})$$

Of course for a given $\vec{A}$, $>^{\Delta}_{deg(\vec{A})}$ is well-founded on $U_{(s,\Delta,\vec{A})}$, because it is based on the comparison of ordinals. Next lemma shows that $\rightarrow^{\Delta}_{sn} \cup >^{\Delta}_{deg(\vec{A})}$ is also well-founded.

Lemma 5.2.28 *Let us denote by* $(\rightarrow^{\Delta}_{sn} \cup >^{\Delta}_{deg(\vec{A})})$ *the union of* $\rightarrow^{\Delta}_{sn}$ *and* $>^{\Delta}_{deg(\vec{A})}$ *relations. Then:*

1. *if* $\Delta' \vdash u \in U_{(s,\Delta,\vec{A})}$ *and* $\Delta' \vdash u(\rightarrow^{\Delta}_{sn} \cup >^{\Delta}_{deg(\vec{A})})\Delta'' \vdash w$ *then* $\Delta'' \vdash w \in U_{(s,\Delta,\vec{A})}$ *and*

2. $(\rightarrow^{\Delta}_{sn} \cup >^{\Delta}_{deg(\vec{A})})$ *is well-founded on* $U_{(s,\Delta,\vec{A})}$.

Proof
The first statement is true for $>^{\Delta}_{deg(\vec{A})}$ by its definition and for $\rightarrow^{\Delta}_{sn}$ by Lemma 5.2.22.

For the second one, let us suppose that there is an infinite sequence $(\Delta_j \vdash u_j)_{j=0}^{\infty}$ of elements from $U_{(s,\vec{A},\Delta)}$.

Note, that for all j we have $deg(\Delta_j \vdash u_j, \vec{A}) \geq deg(\Delta_{j+1} \vdash u_{j+1}, \vec{A})$ (if $\Delta_j \vdash u_j \rightarrow^{\Delta}_{sn} \Delta_{j+1} \vdash u_{j+1}$ then it follows from Lemma 5.2.22, otherwise it is obvious by the definition of $>^{\Delta}_{deg(\vec{A})}$).

So, along $(\Delta_j \vdash u_j)_{j=0}^{\infty}$ the degrees decrease and they decrease in a strict way at every $>^{\Delta}_{deg(\vec{A})}$ comparison. Since there cannot be an infinite strictly decreasing sequence of ordinals, in $(\Delta_j \vdash u_j)_{j=0}^{\infty}$ from some point k there are only $\rightarrow^{\Delta}_{sn}$ reductions. But this is impossible, as $\Delta_k \vdash u_k$ belongs to $\mu F(e_k)$ for some e_k and by Lemma 5.2.12 $\mu F(e_k)$ is a candidate, i.e. a set of strongly normalizing terms with respect to $\rightarrow^{\Delta}_{sn}$. $\square$

The lemmas given below show that reaching under a constructor on recursive positions decreases the degree.

Lemma 5.2.29 (Recursive Position Lemma) *Let* Δ *be an environment,* $\vec{a}$ *terms and* $\vec{I}$ *candidates such that for all* i, $I_i \in \mathcal{C}_{\Delta,a_i}$. *Let us call* $\vec{A}$ *the vector of pairs* $\overrightarrow{(\Delta \vdash a, I)}$.

Suppose also that $c : \overrightarrow{(q:Q)}\overrightarrow{(z:d)}.s(\vec{q})\vec{w}'$ *is a constructor of inductive type* s, $\Delta' \supseteq \Delta$, $\vec{N}$ *are* Δ'*-terms such that* $\Delta' \vdash c(\vec{a}, \vec{N}) \in U_{(s,\Delta,\vec{A})}$ *and that* j *is a recursive position in* c *(with* $d_j = \overrightarrow{(x:T)}.s(\vec{q})\vec{w}$*).*

If j *is a nonfunctional recursive position in* c *(*$|\overrightarrow{(x:T)}| = 0$*) then*

1. $\Delta' \vdash N_j \in U_{(s,\Delta,\vec{A})}$,

2. $\Delta' \vdash c(\vec{a}, \vec{N}) >^{\Delta}_{deg(\vec{A})} \Delta' \vdash N_j$.

If j *is a functional position (*$\overrightarrow{(x:T)} = (x_1 : T_1)\dots(x_m : T_m)$, $m \geq 1$*) then for every* $\Delta'' \supseteq \Delta'$, *every* $v_1, \dots v_m$ *and every candidate assignment* ξ *such that* $[\vec{I}/\vec{q}]|_{\Delta''} \cup \xi$ *is compatible with* $[\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}] \cup [\vec{v}/\vec{x}]$, *if for all* i, $\Delta'' \vdash v_i \in [\![\overrightarrow{q:Q}, \overrightarrow{z:d}, \overrightarrow{x:T} \vdash T_i]\!]_{[\vec{a}/\vec{q}]\cup[\vec{N}/\vec{z}]\cup[\vec{v}/\vec{x}],[\vec{I}/\vec{q}]|_{\Delta''}\cup\xi,\Delta''}$ *then*

1. $\Delta'' \vdash N_j\, v_1\, \ldots v_m \in U_{(s,\Delta,\vec{A})}$,

2. $\Delta' \vdash c(\vec{a},\vec{N}) >^{\Delta}_{deg(\vec{A})} \Delta'' \vdash N_j\, v_1\, \ldots v_m$.

Proof
By the definition of $U_{(s,\Delta,\vec{A})}$

$$\Delta' \vdash c(\vec{a},\vec{N}) \in \mu F(\vec{A},\overrightarrow{(\nabla \vdash t, W)})$$

for some $\vec{N}$, $\vec{t}$, $\vec{W}$. Let us denote $(\vec{A},\overrightarrow{(\nabla \vdash t, W)})$ by e. Of course, $\mu F(e) = F^{deg(\Delta' \vdash c(\vec{a},\vec{N}),e)}(e)$.

By the definition of F, for every recursive position j we have

$$\Delta' \vdash N_j \in Inter(F^{\nu}(Bot))(\overrightarrow{q:Q},\overrightarrow{z:d} \vdash d_j)_{[\vec{a}/\vec{q}]\cup[\vec{N}/\vec{z}],[\vec{I}/\vec{q}],\Delta'}$$

for some $\nu < deg(\Delta' \vdash c(\vec{a},\vec{N}),e)$.

If j is a nonfunctional recursive position ($d_j = s(\vec{q})\vec{w}$) then

$$Inter(F^{\nu}(Bot))(\overrightarrow{q:Q},\overrightarrow{z:d} \vdash d_j)_{[\vec{a}/\vec{q}]\cup[\vec{N}/\vec{z}],[\vec{I}/\vec{q}],\Delta'} = F^{\nu}(Bot)(e')$$

where $e' = (\overrightarrow{(\Delta' \vdash a, I|_{\Delta'})},\overrightarrow{(\Delta' \vdash w([\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}]))})$. Of course we have $deg(\Delta' \vdash N_j, e') \leq \nu < deg(\Delta' \vdash c(\vec{a},\vec{N}),e)$.

By Lemma 5.2.20 if $\Delta' \vdash N_j \in F^{\nu}(Bot)(e')$ then

$$\Delta' \vdash N_j \in F^{\nu}(Bot)(\overrightarrow{(\Delta \vdash a, I)},\overrightarrow{(\Delta' \vdash w([\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}]))})$$

and $deg(\Delta' \vdash N_j, e') = deg(\Delta' \vdash N_j, (\overrightarrow{(\Delta \vdash a, I)},\overrightarrow{(\Delta' \vdash w([\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}]))}))$. Consequently we get the desired conclusions:

1. $\Delta' \vdash N_j \in U_{(s,\Delta,\vec{A})}$,

2. $\Delta' \vdash c(\vec{a},\vec{N}) >^{\Delta}_{deg(\vec{A})} \Delta' \vdash N_j$.

If j is a functional recursive position ($d_j = (x_1 : T_1)\ldots(x_m : T_m).s(\vec{q})\vec{w}$) and Δ'', $v_1, \ldots v_m$ and ξ satisfy

$$\Delta'' \vdash v_i \in [\![\overrightarrow{q:Q},\overrightarrow{z:d},\overrightarrow{x:T} \vdash T_i]\!]_{[\vec{a}/\vec{q}]\cup[\vec{N}/\vec{z}]\cup[\vec{v}/\vec{x}],[\vec{I}/\vec{q}]|_{\Delta''}\cup\xi,\Delta''}$$

for all i, then by the definition of $Inter$ we have

$$\Delta'' \vdash N_j\, v_1\, \ldots v_m \in$$
$$Inter(F^{\nu}(Bot))(\overrightarrow{q : Q}, \overrightarrow{z : d}, \overrightarrow{x : T} \vdash s(\vec{q})\vec{w})_{[\vec{a}/\vec{q}]\cup[\vec{N}/\vec{z}]\cup[\vec{v}/\vec{x}],[\vec{I}/\vec{q}]|_{\Delta''}\cup\xi,\Delta''} =$$
$$F^{\nu}(Bot)(e'')$$

where $e'' = (\overrightarrow{(\Delta'' \vdash a, I|_{\Delta''})}, \overrightarrow{(\Delta'' \vdash w([\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}] \cup [\vec{v}/\vec{x}]))})$. Like before, $deg(\Delta'' \vdash N_j\, v_1\, \ldots v_m, e'') \leq \nu < deg(\Delta' \vdash c(\vec{a}, \vec{N}), e)$.

By Lemma 5.2.20 if $\Delta'' \vdash N_j\, v_1\, \ldots v_m \in F^{\nu}(Bot)(e'')$ then $\Delta'' \vdash N_j\, v_1\, \ldots v_m \in F^{\nu}(Bot)(\overrightarrow{(\Delta \vdash a, I)}, \overrightarrow{(\Delta' \vdash w([\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}]))})$ and $deg(\Delta'' \vdash N_j\, v_1\, \ldots v_m, e') = deg(\Delta'' \vdash N_j\, v_1\, \ldots v_m, (\overrightarrow{(\Delta \vdash a, I)}, \overrightarrow{(\Delta' \vdash w([\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}]))}))$. Consequently we get the desired conclusions:

1. $\Delta'' \vdash N_j\, v_1\, \ldots v_m \in U_{(s,\Delta,\vec{A})}$,

2. $\Delta' \vdash c(\vec{a}, \vec{N}) >^{\Delta}_{deg(\vec{A})} \Delta'' \vdash N_j\, v_1\, \ldots v_m$. $\square$

Lemma 5.2.30 *Suppose that $\Delta \vdash u : T$ and that u has a recursive constructor subterm at position $p = p_1 \cdot \ldots p_n$. Then:*

- *T is convertible with $s(\vec{a})\vec{t}$ for some type constructor s and some terms $\vec{a}$, $\vec{t}$,*
- *every term on the path p is headed by some constructor of s and $\vec{a}$ are its parameter arguments,*
- *$p_1, \ldots p_n$ are recursive positions and $p_1, \ldots p_{n-1}$ are in addition nonfunctional ones.*

Proof
Obvious. $\square$

Lemma 5.2.31 *Let Δ be an environment, $\vec{a}$ terms and $\vec{I}$ candidates such that for all i, $I_i \in \mathcal{C}_{\Delta,a_i}$. Let us call $\vec{A}$ the vector of pairs $\overrightarrow{(\Delta \vdash a, I)}$.*

Suppose also that $\Delta' \supseteq \Delta$, u satisfies $\Delta' \vdash u : s(\vec{a})\vec{t}$ for some $\vec{t}$ and $\Delta' \vdash u \in U_{(s,\Delta,\vec{A})}$. Moreover suppose that N_j is a recursive constructor subterm of u at position $p = p_1 \cdot \ldots p_n$, $u|_{p_1 \cdot \ldots p_{n-1}} = c(\vec{a}, \vec{N})$, $c : \overrightarrow{(q : Q)}\overrightarrow{(z : d)}.s(\vec{q})\vec{w}'$ and $d_j = \overrightarrow{(x : T)}.s(\vec{q})\vec{w}$.

If j is a nonfunctional recursive position in c ($|\overrightarrow{(x : T)}| = 0$) then

1. $\Delta' \vdash N_j \in U_{(s,\Delta,\vec{A})}$,

2. $\Delta' \vdash u >^{\Delta}_{deg(\vec{A})} \Delta' \vdash N_j$.

If j is a functional position ($\overrightarrow{(x:T)} = (x_1 : T_1)\dots(x_m : T_m)$, $m \geq 1$) then for every $\Delta'' \supseteq \Delta'$, every $v_1, \dots v_m$ and every candidate assignment ξ such that $[\vec{I}/\vec{q}]|_{\Delta''} \cup \xi$ is compatible with $[\vec{a}/\vec{q}] \cup [\vec{N}/\vec{z}] \cup [\vec{v}/\vec{x}]$, if for all i, $\Delta'' \vdash v_i \in [\![\overrightarrow{q:Q}, \overrightarrow{z:d}, \overrightarrow{x:T} \vdash T_i]\!]_{[\vec{a}/\vec{q}]\cup[\vec{N}/\vec{z}]\cup[\vec{v}/\vec{x}],[\vec{I}/\vec{q}]|_{\Delta''}\cup\xi,\Delta''}$ then

1. $\Delta'' \vdash N_j\, v_1\, \dots v_m \in U_{(s,\Delta,\vec{A})}$,

2. $\Delta' \vdash u >^{\Delta}_{deg(\vec{A})} \Delta'' \vdash N_j\, v_1\, \dots v_m$.

Proof

By induction on the length of the path $p = p_1 \cdot \dots p_n$.

If $n = 1$ then the desired conclusion follows immediately from Recursive Position Lemma 5.2.29. Otherwise, let us suppose that for all paths of length $n-1$ the lemma is true and let use see what we can deduce for n. In other words, let us suppose that N_j is a recursive constructor subterm of u at position $p = p_1 \cdot \dots p_n$ and that it is an immediate subterm of $c(\vec{a}, \vec{N})$ ($u|_{p_1 \cdots p_{n-1}} = c(\vec{a}, \vec{N})$).

By Lemma 5.2.30, $c(\vec{a}, \vec{N})$ is also a recursive constructor subterm of u and in addition it is a nonfunctional one. Consequently, by inductive hypothesis:

1. $\Delta' \vdash c(\vec{a}, \vec{N}) \in U_{(s,\Delta,\vec{A})}$,
2. $\Delta' \vdash u >^{\Delta}_{deg(\vec{A})} \Delta' \vdash c(\vec{a}, \vec{N})$.

Now, since p_n is a recursive position in c, we may use lemma 5.2.29 and get all the properties we want, except that $\Delta' \vdash N_j$ (or $\Delta'' \vdash N_j\, v_1\, \dots v_m$ if j is functional) is smaller than $\Delta' \vdash c(\vec{a}, \vec{N})$ and not directly $\Delta' \vdash u$. But we have already $\Delta' \vdash u >^{\Delta}_{deg(\vec{A})} \Delta' \vdash c(\vec{a}, \vec{N})$. $\square$

5.3 Auxiliary lemmas

This section 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)-non-object and $(\Gamma_1, y : T, \Gamma_2)$ is well-formed then:*

$$[\![\Gamma_1, \Gamma_2 \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma_1, y : T, \Gamma_2 \vdash A]\!]_{\rho^+,\xi^+,\Delta}$$

where $\rho^+ = \rho \cup [M/y]$ for some M satisfying $\Delta \vdash M : T\rho$ and $\xi^+ = \xi$ if $y \in \mathcal{V}ar^\star$ and $\xi^+ = \xi \cup [C/y]$ for some $C \in \mathcal{C}_{\Delta,M}$ otherwise.

Proof
By induction on the structure of A. We will consider only the case where $y \in \mathcal{V}ar^\Box$.

1. If A is $\Box$, $\star$ or $z \in \mathcal{V}ar^\Box$ then $[\![\Gamma_1, \Gamma_2 \vdash A]\!]_{\rho,\xi,\Delta}$ equals $\mathcal{SN}_{\Delta,\Box}$, $\mathcal{SN}_{\Delta,\star}$ or $A\xi$ respectively, and obviously this equals $[\![\Gamma_1, y : T, \Gamma_2 \vdash A]\!]_{\rho^+,\xi^+,\Delta}$.

2. If $A = P\,Q$ we do the proof only for the case where $Q \in$ *Constr*.

   By induction hypothesis, we have:

$$\begin{aligned}
[\![\Gamma_1, \Gamma_2 \vdash P\,Q]\!]_{\rho,\xi,\Delta} &= [\![\Gamma_1, \Gamma_2 \vdash P]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q\rho, [\![\Gamma_1, \Gamma_2 \vdash Q]\!]_{\rho,\xi,\Delta}) \\
&= [\![\Gamma_1, y : T, \Gamma_2 \vdash P]\!]_{\rho^+,\xi^+,\Delta} \\
&\qquad\quad (\Delta \vdash Q\rho^+, [\![\Gamma_1, y : T, \Gamma_2 \vdash Q]\!]_{\rho^+,\xi^+,\Delta}) \\
&= [\![\Gamma_1, y : T, \Gamma_2 \vdash P\,Q]\!]_{\rho^+,\xi^+,\Delta}
\end{aligned}$$

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

$$\begin{gathered}
\{\Delta' \vdash N \mid \Delta' \vdash N : ((x:B).D)\rho,\ \Delta' \supseteq \Delta,\ \text{and} \\
\forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma_1, \Gamma_2 \vdash B]\!]_{\rho,\xi|_{\Delta''},\Delta''},\ \forall C' \in \mathcal{C}_{\Delta'',A}, \\
\Delta'' \vdash N\,A \in [\![\Gamma_1, \Gamma_2, x : B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'/x],\Delta''}\}
\end{gathered}$$

   equals

$$\begin{gathered}
\{\Delta' \vdash N \mid \Delta' \vdash N : ((x:B).D)\rho^+,\ \Delta' \supseteq \Delta,\ \text{and} \\
\forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma_1, y : T, \Gamma_2 \vdash B]\!]_{\rho^+,\xi^+|_{\Delta''},\Delta''},\ \forall C' \in \mathcal{C}_{\Delta'',A}, \\
\Delta'' \vdash N\,A \in [\![\Gamma_1, y : T, \Gamma_2, x : B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta''}\cup[C'/x],\Delta''}\}
\end{gathered}$$

But $((x:B).D)\rho = ((x:B).D)\rho^+$ because $(x:B).D$ is a Γ_1,Γ_2-non-object. By induction hypothesis we have $[\![\Gamma_1,\Gamma_2 \vdash B]\!]_{\rho,\xi|_{\Delta''},\Delta''} = [\![\Gamma_1, y:T,\Gamma_2 \vdash B]\!]_{\rho^+,\xi^+|_{\Delta''},\Delta''}$ and $[\![\Gamma_1,\Gamma_2, x:B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'|_{\Delta''}/x],\Delta''} = [\![\Gamma_1, y:T,\Gamma_2, x:B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta''}\cup[C'|_{\Delta''}/x],\Delta''}$. Consequently $[\![\Gamma_1,\Gamma_2 \vdash (x\!:\!B).D]\!]_{\rho,\xi,\Delta} = [\![\Gamma_1, y:T,\Gamma_2 \vdash (x\!:\!B).D]\!]_{\rho^+,\xi^+,\Delta}$.

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

$$
\begin{array}{l}
[\![\Gamma_1,\Gamma_2 \vdash \lambda x\!:\!B.D]\!]_{\rho,\xi,\Delta} = \\
\qquad\qquad \lambda(\Delta' \vdash A)\lambda C'.[\![\Gamma_1,\Gamma_2, x:B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C'/x],\Delta'} \\
\qquad\qquad\qquad \text{a function with domain} \\
\qquad\qquad \{(\Delta' \vdash A, C') \mid \Delta' \vdash A : B\rho,\ \Delta' \supseteq \Delta,\ C' \in \mathcal{C}_{\Delta',A}\}
\end{array}
$$

and we have to check that it equals

$$
\begin{array}{l}
[\![\Gamma_1, y:T,\Gamma_2 \vdash \lambda x\!:\!B.D]\!]_{\rho^+,\xi^+,\Delta} = \\
\qquad \lambda(\Delta' \vdash A)\lambda C'.[\![\Gamma_1, y:T,\Gamma_2, x:B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta'}\cup[C'/x],\Delta'} \\
\qquad\qquad\qquad \text{a function with domain} \\
\qquad\qquad \{(\Delta' \vdash A, C') \mid \Delta' \vdash A : B\rho^+,\ \Delta' \supseteq \Delta,\ C' \in \mathcal{C}_{\Delta',A}\}
\end{array}
$$

But $B\rho = B\rho^+$ because B is a Γ_1,Γ_2-non-object. By induction hypothesis we have $[\![\Gamma_1,\Gamma_2, x:B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C'/x],\Delta'} = [\![\Gamma_1, y:T,\Gamma_2, x:B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta'}\cup[C'/x],\Delta'}$. Consequently $[\![\Gamma_1,\Gamma_2 \vdash \lambda x:B.D]\!]_{\rho,\xi,\Delta} = [\![\Gamma_1, y:T,\Gamma_2 \vdash \lambda x\!:\!B.D]\!]_{\rho^+,\xi^+,\Delta}$. $\Box$

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

$$[\![\Gamma_1,\Gamma_2[M/y] \vdash A[M/y]]\!]_{\rho,\xi,\Delta} = [\![\Gamma_1, y:T,\Gamma_2 \vdash A]\!]_{\rho^+,\xi^+,\Delta}$$

where $\rho^+ = \rho\cup[M\rho/y]$ and $\xi^+ = \xi$ if $y \in \mathcal{V}ar^\star$ and $\xi^+ = \xi\cup[[\![\Gamma_1 \vdash M]\!]_{\rho,\xi,\Delta}/y]$ otherwise.

Proof

Induction on the structure of A. We will consider only the case where $y \in \mathcal{V}ar^\Box$.

1. If A is $\Box$, $\star$, y or $z \in \mathcal{V}ar^{\Box}$ such that $z \neq y$ then $[\![\Gamma_1, \Gamma_2[M/y] \vdash A[M/y]]\!]_{\rho,\xi,\Delta}$ equals $\mathcal{SN}_{\Delta,\Box}$, $\mathcal{SN}_{\Delta,\star}$, $[\![\Gamma_1 \vdash M]\!]_{\rho,\xi,\Delta}$ or $z\xi$ respectively, and obviously this equals $[\![\Gamma_1, y : T, \Gamma_2 \vdash A]\!]_{\rho^+,\xi^+,\Delta}$.

2. If $A = P\,Q$ we do the proof only for the case where $Q \in \mathit{Constr}$.

   By induction hypothesis, we have:

$$\begin{array}{l}
[\![\Gamma_1, \Gamma_2[M/y] \vdash (P\,Q)[M/y]]\!]_{\rho,\xi,\Delta} = \\
[\![\Gamma_1, \Gamma_2[M/y] \vdash P[M/y]]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q[M/y]\rho, [\![\Gamma_1, \Gamma_2[M/y] \vdash Q[M/y]]\!]_{\rho,\xi,\Delta}) \\
\quad = [\![\Gamma_1, y : T, \Gamma_2 \vdash P]\!]_{\rho^+,\xi^+,\Delta}(\Delta \vdash Q\rho^+, [\![\Gamma_1, y : T, \Gamma_2 \vdash Q]\!]_{\rho^+,\xi^+,\Delta}) = \\
\hfill [\![\Gamma_1, y : T, \Gamma_2 \vdash P\,Q]\!]_{\rho^+,\xi^+,\Delta}
\end{array}$$

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

$$\begin{array}{c}
\{\Delta' \vdash N \mid \Delta' \vdash N \,:\, ((x\!:\!B).D)[M/y]\rho,\ \Delta' \supseteq \Delta,\ \text{and} \\
\forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma_1, \Gamma_2[M/y] \vdash B[M/y]]\!]_{\rho,\xi|_{\Delta''},\Delta''},\ \forall C' \in \mathcal{C}_{\Delta'',A}, \\
\Delta'' \vdash N\,A \in [\![\Gamma_1, \Gamma_2[M/y], x : B[M/y] \vdash D[M/y]]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'/x],\Delta''}\}
\end{array}$$

   equals

$$\begin{array}{c}
\{\Delta' \vdash N \mid \Delta' \vdash N \,:\, ((x\!:\!B).D)\rho^+,\ \Delta' \supseteq \Delta,\ \text{and} \\
\forall \Delta'' \supseteq \Delta',\ \forall \Delta'' \vdash A \in [\![\Gamma_1, y : T, \Gamma_2 \vdash B]\!]_{\rho^+,\xi^+|_{\Delta''},\Delta''},\ \forall C' \in \mathcal{C}_{\Delta'',A}, \\
\Delta'' \vdash N\,A \in [\![\Gamma_1, y : T, \Gamma_2, x : B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta''}\cup[C'/x],\Delta''}\}
\end{array}$$

   But $((x : B).D)[M/y]\rho = ((x : B).D)\rho^+$ By induction hypothesis we have $[\![\Gamma_1, \Gamma_2[M/y] \vdash B[M/y]]\!]_{\rho,\xi|_{\Delta''},\Delta''} = [\![\Gamma_1, y : T, \Gamma_2 \vdash B]\!]_{\rho^+,\xi^+|_{\Delta''},\Delta''}$ and $[\![\Gamma_1, \Gamma_2[M/y], x : B[M/y] \vdash D[M/y]]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'|_{\Delta''}/x],\Delta''} = [\![\Gamma_1, y : T, \Gamma_2, x : B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta''}\cup[C'|_{\Delta''}/x],\Delta''}$. Consequently, $[\![\Gamma_1, \Gamma_2[M/y] \vdash ((x\!:\!B).D)[M/y]]\!]_{\rho,\xi,\Delta} = [\![\Gamma_1, y : T, \Gamma_2 \vdash (x\!:\!B).D]\!]_{\rho^+,\xi^+,\Delta}$.

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

$$\begin{array}{c}
[\![\Gamma_1, \Gamma_2[M/y] \vdash (\lambda x\!:\!B.D)[M/y]]\!]_{\rho,\xi,\Delta} = \lambda(\Delta' \vdash A)\lambda C'. \\
[\![\Gamma_1, \Gamma_2[M/y], x : B[M/y] \vdash D[M/y]]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C'/x],\Delta'} \\
\text{a function with domain} \\
\{(\Delta' \vdash A, C') \mid \Delta' \vdash A \,:\, B[M/y]\rho,\ \Delta' \supseteq \Delta,\ C' \in \mathcal{C}_{\Delta',A}\}
\end{array}$$

and we have to check that it equals

$$[\![\Gamma_1, y : T, \Gamma_2 \vdash \lambda x\!:\!B.D]\!]_{\rho^+,\xi^+,\Delta} = \lambda(\Delta' \vdash A)\lambda C'.$$
$$[\![\Gamma_1, y : T, \Gamma_2, x : B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta'}\cup[C'/x],\Delta'}$$
$$\text{a function with domain}$$
$$\{(\Delta' \vdash A, C') \mid \Delta' \vdash A \,:\, B\rho^+,\ \Delta' \supseteq \Delta,\ C' \in \mathcal{C}_{\Delta',A}\}$$

But $(B[M/y])\rho = B\rho^+$ and by induction hypothesis we have

$$[\![\Gamma_1, \Gamma_2[M/y], x : B[M/y] \vdash D[M/y]]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C'/x],\Delta'} =$$
$$[\![\Gamma_1, y : T, \Gamma_2, x : B \vdash D]\!]_{\rho^+\cup[A/x],\xi^+|_{\Delta'}\cup[C'/x],\Delta'}$$

Consequently, $[\![\Gamma_1, \Gamma_2[M/y] \vdash (\lambda x\!:\!B.D)[M/y]]\!]_{\rho,\xi,\Delta} = [\![\Gamma_1, y : T, \Gamma_2 \vdash \lambda x\!:\!B.D]\!]_{\rho^+,\xi^+,\Delta}$. $\square$

Corollary 5.3.3 (Substitution Lemma) *Let Γ_0, Γ and Δ be environments, $\mu : \Gamma_0 \to \Gamma$ and $\rho : \Gamma \to \Delta$ well-typed substitution and ξ a candidate assignment compatible with ρ. For every Γ_0-non-object A the following equality holds*

$$[\![\Gamma \vdash A\mu]\!]_{\rho,\xi,\Delta} = [\![\Gamma_0 \vdash A]\!]_{\mu\rho,\zeta,\Delta}$$

where $x\zeta = [\![\Gamma \vdash x\mu]\!]_{\rho,\xi,\Delta}$.

Proof
The proof is done by using repeatedly the previous lemma. Let $\Gamma_0 = x_1 : A_1, \ldots x_n : A_n$.

By Lemma 5.3.1 $[\![\Gamma_0 \vdash A]\!]_{\mu\rho,\zeta,\Delta} = [\![\Gamma, \Gamma_0 \vdash A]\!]_{\rho\cup\mu\rho,\xi\cup\zeta,\Delta}$ Now we can start eliminating Γ_0 according to the Lemma 5.3.2 from x_1 to x_n. Every step of elimination is correct because $\Gamma \vdash x_i\mu \,:\, A_i\mu$ by the definition of μ. Let us denote $[x_i\mu/x_i, \ldots x_n\mu/x_n]$ by μ_i and $[[\![\Gamma \vdash x_i\mu]\!]_{\rho,\xi,\Delta}/x_i, \ldots [\![\Gamma \vdash x_n\mu]\!]_{\rho,\xi,\Delta}/x_n]$ by ζ_i.

$$[\![\Gamma, x_1 : A_1, \ldots x_n : A_n \vdash A]\!]_{\rho\cup\mu\rho,\xi\cup\zeta,\Delta} =$$
$$[\![\Gamma, x_2 : A_2[x_1\mu/x_1], \ldots x_n : A_n[x_1\mu/x_1] \vdash A[x_1\mu/x_1]]\!]_{\rho\cup\mu_2\rho,\xi\cup\zeta_2,\Delta} =$$
$$\vdots$$
$$[\![\Gamma, x_n : A_n[x_1\mu/x_1, \ldots x_{n-1}\mu/x_{n-1}] \vdash A[x_1\mu/x_1, \ldots x_{n-1}\mu/x_{n-1}]]\!]_{\rho\cup\mu_n\rho,\xi\cup\zeta_n,\Delta}$$
$$= [\![\Gamma \vdash A\mu]\!]_{\rho,\xi,\Delta} \quad \square$$

Corollary 5.3.4 (Simple Substitution Lemma) *Let $\overrightarrow{p:P}$, G_0, G and Δ be environments, $\rho : (\overrightarrow{p:P}, G) \to \Delta$ and $\mu : (\overrightarrow{p:P}, G_0) \to (\overrightarrow{p:P}, G)$ well-typed substitution, satisfying $x\mu = x$ for $x \in \overrightarrow{p:P}$, and ξ a candidate assignment compatible with ρ only on variables from $\vec{p}$. If $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box} \subseteq \vec{p}$ then*

$$[\![\overrightarrow{p:P}, G \vdash A\mu]\!]_{\rho,\xi^+,\Delta} = [\![\overrightarrow{p:P}, G_0 \vdash A]\!]_{\mu\rho,\xi,\Delta}$$

holds for any ξ^+ defined as

$$x\xi^+ = \begin{cases} x\xi & x \in \vec{p} \\ C_x & \textit{otherwise} \end{cases}$$

where C_x, $x \in dom(G)$, is an arbitrary candidate from $\mathcal{C}_{\Delta,G(x)\rho}$.

Proof
First note that the candidate assignments in the interpretations above are correct. Indeed, if ξ is compatible with ρ on $\vec{p}$ then ξ^+ is compatible with the whole ρ. Moreover, since $\mathcal{FV}(A) \cap \mathcal{V}ar^{\Box} \subseteq \vec{p}$, the interpretation $[\![\overrightarrow{p:P}, G_0 \vdash A]\!]_{\mu\rho,\hat{\xi},\Delta}$ does not depend on the values of $\hat{\xi}$ on variables outside $\vec{p}$ (see Lemma 5.2.5) and we may write simply ξ.

Now we can apply Substitution Lemma 5.3.3 with $\Gamma_0 = \overrightarrow{p:P}, G_0$, $\Gamma = \overrightarrow{p:P}, G$ and $\xi = \xi^+$ and check that the resulting ζ equals ξ on variables from $\vec{p}$. $\Box$

Lemma 5.3.5 (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-non-object then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma' \vdash A]\!]_{\rho,\xi,\Delta}$.*

Proof
First note that $\rho : \Gamma' \to \Delta$ and the interpretation $[\![\Gamma' \vdash A]\!]_{\rho,\xi,\Delta}$ is correct. The proof is done by straightforward induction on the structure of A.

Let us detail one of the cases, for $A = P\,Q$, where $Q \in \mathit{Constr}$. By induction hypothesis, we have:

$$\begin{aligned}
[\![\Gamma \vdash P\,Q]\!]_{\rho,\xi,\Delta} &= [\![\Gamma \vdash P]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q\rho, [\![\Gamma \vdash Q]\!]_{\rho,\xi,\Delta}) \\
&= [\![\Gamma' \vdash P]\!]_{\rho,\xi,\Delta}(\Delta \vdash Q\rho, [\![\Gamma' \vdash Q]\!]_{\rho,\xi,\Delta}) \\
&= [\![\Gamma' \vdash P\,Q]\!]_{\rho,\xi,\Delta} \quad \Box
\end{aligned}$$

Lemma 5.3.6 (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-non-objects A and A', if $\Gamma \vdash A \overset{*}{\leftrightarrow} A'$ then $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash A']\!]_{\rho,\xi,\Delta}$.*

Proof
First we will show that $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash A']\!]_{\rho,\xi,\Delta}$ whenever $\Gamma \vdash A \to A'$. The desired conclusion will follow by applying this fact as many times as needed according to $\Gamma \vdash A \overset{*}{\leftrightarrow} A'$ (recall that $\Gamma \vdash A \overset{*}{\leftrightarrow} A'$ means that there is a sequence of $\to$ and $\leftarrow$ reductions passing through well-typed terms).

The proof is done by induction on the structure of A. The typical case is when reduction takes place in the proper subterm of A. The reduction may also happen at the root position, but then A must be a beta redex. Let us detail the proofs for two cases: when $A = (x:B).D$ and $A = (\lambda x:B.D)\, b$.

- If $A = (x : B).D$ and $\Gamma \vdash A \to A'$ then either $\Gamma \vdash B \to B'$ and $A' = (x:B').D$, or $\Gamma, x : B \vdash D \to D'$ and $A' = (x:B).D'$. We will do the proof only for the case when B is a kind. The interpretation $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ is equal to:

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

  If $A' = (x:B').D$ then by induction hypothesis $[\![\Gamma \vdash B]\!]_{\rho,\xi|_{\Delta''},\Delta''} = [\![\Gamma \vdash B']\!]_{\rho,\xi|_{\Delta''},\Delta''}$ and by Lemma 5.3.5 $[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'/x],\Delta''} = [\![\Gamma, x : B' \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta''}\cup[C'/x],\Delta''}$.
  If $A' = (x:B).D'$ then by induction hypothesis

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

  In both cases of A' it follows that $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash A']\!]_{\rho,\xi,\Delta}$.

- If $A = (\lambda x:B.D)\, b$ and $A' = D[b/x]$ then

$$[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta} = [\![\Gamma \vdash \lambda x:B.D]\!]_{\rho,\xi,\Delta}(\Delta \vdash b\rho, [\![\Gamma \vdash b]\!]_{\rho,\xi,\Delta})$$

  Since $[\![\Gamma \vdash \lambda x:B.D]\!]_{\rho,\xi,\Delta}$ equals

$$\begin{gathered}\lambda(\Delta' \vdash A)\lambda C.[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[A/x],\xi|_{\Delta'}\cup[C/x],\Delta'} \\ \text{a function with domain} \\ \{(\Delta' \vdash A, C) \mid \Delta' \vdash A \,:\, B\rho,\ \Delta' \supseteq \Delta,\ C \in \mathcal{C}_{\Delta',A}\}\end{gathered}$$

the interpretation $[\![\Gamma \vdash A]\!]_{\rho,\xi,\Delta}$ results in

$$[\![\Gamma, x : B \vdash D]\!]_{\rho\cup[b\rho/x],\xi\cup[[\![\Gamma\vdash b]\!]_{\rho,\xi,\Delta}/x],\Delta}$$

which, by Lemma 5.3.2, equals $[\![\Gamma \vdash D[b/x]]\!]_{\rho,\xi,\Delta}$ and $[\![\Gamma \vdash A']\!]_{\rho,\xi,\Delta}$. $\square$

The following lemma presents the essence of the definition of interpretation for inductive types. These interpretations are built in such a way that reaching under a constructor and building a constructor headed term preserve the reducibility (and more precisely the fact of being an element of the interpretation of its type).

Lemma 5.3.7 (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 $\Delta \vdash c(\vec{a'}, \vec{N}) \in [\![\Gamma \vdash s(\vec{a})\vec{u}]\!]_{\rho,\xi,\Delta}$.

Proof
Let us start by **decomposition** case. If $\Delta \vdash c(\vec{a'}, \vec{N}) \in [\![\Gamma \vdash s(\vec{a})\vec{u}]\!]_{\rho,\xi,\Delta}$ then also

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

where $U_i = [\![\Gamma \vdash u_i]\!]_{\rho,\xi,\Delta}$ if $u_i \notin Obj$ and $U_i = \emptyset$ otherwise. We can use Lemma 5.2.14 and get

- $\Delta \vdash a'_i \in [\![\overrightarrow{p : P} \vdash P_i]\!]_{[\vec{a}\rho/\vec{p}],[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}],\Delta}$ for every $i = 1, \dots r$ and

- $\Delta \vdash N_i \in [\![\overrightarrow{p : P}, \overrightarrow{z : d} \vdash d_j]\!]_{[\vec{a}\rho/\vec{p}] \cup [\vec{N}/\vec{z}], [\overrightarrow{[\![\Gamma \vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}], \Delta}$ for every $i = 1, \ldots k$,

Now we use Lemma 5.3.2 to get the desired conclusion.

In the **composition** case we have to show that $\Delta \vdash c(\vec{a'}, \vec{N}) \in [\![\Gamma \vdash s(\vec{a})\vec{u}]\!]_{\rho,\xi,\Delta}$. To this end it is sufficient to reverse all the steps presented above. $\square$

Note the difference in the formulation of two cases of the lemma. In the **decomposition** case given constructor headed term and an interpretation to which it belongs we can find the interpretations for its immediate subterms. In the **composition** case given immediate subterms and the interpretations to which they belong to reconstruct the interpretation for the whole term we must additionally know some terms $\vec{u}$.

Lemma 5.3.8 (Constructor Lemma) *Let* Γ, Δ *be environments,* $\theta : \Gamma \to \Delta$ *a well-typed substitution,* ζ *a candidate assignment compatible with* θ*. Suppose that* $c(\vec{a}, \vec{N})$ *is a term satisfying* $\Gamma \vdash_{CC} c(\vec{a}, \vec{N}) : s(\vec{a})\vec{v}$ *and* $\Delta \vdash c(\vec{a}, \vec{N})\theta \in [\![\Gamma \vdash s(\vec{a})\vec{v}]\!]_{\theta,\zeta,\Delta}$*. Then for every* $t \in \{\vec{a}, \vec{N}\}$ *there is* T *such that:*

- $\Gamma \vdash_{CC} t : T$, *and*
- $\Delta \vdash t\theta \in [\![\Gamma \vdash T]\!]_{\theta,\zeta,\Delta}$.

Proof
Let us suppose that $c : (q_1 : Q_1) \ldots (q_r : Q_r)(z_1 : d_1) \ldots (z_k : d_k).s(\vec{q})\vec{w}$. By Inductive Type Lemma 5.3.7 we have

$$\Delta \vdash a_i\theta \in [\![\Gamma \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\theta,\zeta,\Delta}$$
$$\Delta \vdash N_j\theta \in [\![\Gamma, \overrightarrow{z : d[\vec{a}/\vec{q}]} \vdash d_j[\vec{a}/\vec{q}]]\!]_{(\theta \cup [\vec{N}\theta/\vec{z}]),\zeta,\Delta}$$

By Lemma 5.3.2 we know that $[\![\Gamma, \overrightarrow{z : d[\vec{a}/\vec{q}]} \vdash d_j[\vec{a}/\vec{q}]]\!]_{(\theta \cup [\vec{N}\theta/\vec{z}]),\zeta,\Delta} = [\![\Gamma \vdash d_j[\vec{a}/\vec{q}][\vec{N}/\vec{z}]]\!]_{\theta,\zeta,\Delta}$. Of course, for all i, j $\Gamma \vdash_{CC} a_i : Q_i[\vec{a}/\vec{q}]$ and $\Gamma \vdash_{CC} N_j : d_j[\vec{a}/\vec{q}][\vec{N}/\vec{z}]$. This completes the proof of this lemma. $\square$

Lemma 5.3.9 (Constructor Subterm Lemma) *Let* Γ, Δ *be environments,* $\theta : \Gamma \to \Delta$ *a well-typed substitution,* ζ *a candidate assignment compatible with* θ*. Suppose also that* $\Gamma \vdash_{CC} u : U$ *and* $\Delta \vdash u\theta \in [\![\Gamma \vdash U]\!]_{\theta,\zeta,\Delta}$ *hold. Then for every constructor subterm of* u, t, *there is* T *such that:*

- $\Gamma \vdash_{CC} t : T$, *and*

- $\Delta \vdash t\theta \in [\![\Gamma \vdash T]\!]_{\theta,\zeta,\Delta}$.

Proof
By applying repeatedly Lemma 5.3.8. $\square$

5.4 Fun and Computable Closure Lemmas

This section is devoted to one of the most important proofs in the paper, namely to the proof of Fun Lemma. We will show that the reducibility of a function symbol headed term follows from the reducibility of its arguments. In this proof we will use all the machinery introduced up to this point.

We start by presenting well-founded orderings, that will be used to justify the induction in the lemmas to come.

Lemma 5.4.1 *Let Δ be an environment and let us denote by $\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}}$ the union of $\rightarrow_{sn}^{\Delta}$ and constructor subterm relations. Then:*

1. *if $\Delta' \vdash t \in \mathcal{SN}_\Delta$ and $\Delta' \vdash t(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})\Delta'' \vdash t'$ then $\Delta'' \vdash t' \in \mathcal{SN}_\Delta$*
2. *$(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})$ and $(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})^+$ are well-founded on $\mathcal{SN}_\Delta$.*
3. *for every $\Delta'' \supseteq \Delta' \supseteq \Delta$ if $\Gamma \vdash t \succ^{\mathcal{CS}} u$, $\rho : \Gamma \to \Delta''$ and $t\rho$ is a Δ'-term then $\Delta' \vdash t\rho(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})^+\Delta'' \vdash u\rho$*

Proof

The first statement is easy since reducts and constructor subterms of strongly normalizing terms are strongly normalizing (see Lemmas 5.1.3 and 5.1.16).

For the second one, let us suppose that there is an infinite sequence of $(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})$ reductions $(\Delta_j \vdash t_j)_{j=0}^{\infty}$, such that both $\rhd^{\mathcal{CS}}$ and $\rightarrow_{sn}^{\Delta}$ appear infinitely often (otherwise the conclusion is immediate). Let us consider an $\rhd^{\mathcal{CS}}$ step followed by an $\rightarrow_{sn}$ step: $\Delta_1 \vdash a \rhd^{\mathcal{CS}} \Delta_1 \vdash b \rightarrow_{sn}^{\Delta} \Delta_2 \vdash c(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})\Delta_3 \vdash d$. By the definition of $\rightarrow_{sn}^{\Delta}$ we know that $\Delta_2 \vdash b \rightarrow c$. Since $\rightarrow$ is stable under context, we have $\Delta_2 \vdash a[b]_p \rightarrow^+ a[c]_p$ (where p is the position at which b occurs in a) and $\Delta_1 \vdash a(\rightarrow_{sn}^{\Delta})^+\Delta_2 \vdash a[c]_p$. Of course $\Delta_2 \vdash a[c]_p(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})\Delta_3 \vdash a[d]_p$. Consequently, every $\rhd^{\mathcal{CS}}$ step can be eliminated and one gets an infinite sequence of $\rightarrow_{sn}^{\Delta}$ starting in t_1. But this contradicts the assumption that $t_1 \in \mathcal{SN}_\Delta$.

For the third one there are three cases to be considered: either $\Gamma \vdash t \succ u : T$ or there is an a satisfying $t \rhd^{\mathcal{CS}} a$ and $\Gamma \vdash a \succ u : T$ or $a = u$. Note that $t \rhd^{\mathcal{CS}} a$ implies that $a\rho$ is a Δ'-term.

In the case when $\Gamma \vdash t \succ u : T$, we have $\Delta'' \vdash t\rho \rightarrow u\rho$ and consequently $\Delta' \vdash t\rho \rightarrow_{sn}^{\Delta} \Delta'' \vdash u\rho$.

If $t \rhd^{\mathcal{CS}} a$ and $\Gamma \vdash a \succ u : T$ then $t\rho \rhd^{\mathcal{CS}} a\rho$ and $\Delta'' \vdash a\rho \rightarrow u\rho$. Summarizing we have $\Delta' \vdash t\rho \rhd^{\mathcal{CS}} \Delta' \vdash a\rho \rightarrow_{sn}^{\Delta} \Delta'' \vdash u\rho$, i.e. two steps of $(\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})$.

If $t \rhd^{\mathcal{CS}} a$ and $a = u$ then $\Delta' \vdash t\rho \rhd^{\mathcal{CS}} \Delta' \vdash a\rho = \Delta' \vdash u\rho$. $\square$

The following relation is the basis of induction in Fun Lemma. The domain of this relation is a dependent product of suitable subsets of $\mathcal{SN}$. According to the status of the function symbol, the relation is either a multiset or lexicographic extension of an enforced version of $\rightarrow_{sn}^{\Delta}$: $\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}}$ for multiset function symbols and $\rightarrow_{sn}^{\Delta} \cup >_{deg}^{\Delta}$ for inductive positions of lexicographic symbols.

Definition 5.4.2 *Let f be a function symbol of type $(p_1 : P_1) \dots (p_r : P_r)(x_1 : b_1) \dots (x_n : b_n).c$, Δ an environment, ρ a well-typed substitution from $\overrightarrow{p : P}$ to Δ and ζ a candidate assignment compatible with ρ.*

We define the dependent product $DepD_{(f,\Delta,\rho,\zeta)}$ and a relation $\leadsto_{(f,\rho,\zeta)}^{\Delta}$ on $DepD_{(f,\Delta,\rho,\zeta)}$ according to the status of f:

1. *if $f \in Mul$ then $DepD_{(f,\Delta,\rho,\zeta)} = \mathcal{SN}_{\Delta}^{r+n}$ (the product of $r+n$ copies of $\mathcal{SN}_{\Delta}$) and $\leadsto_{(f,\rho,\zeta)}^{\Delta} = ((\rightarrow_{sn}^{\Delta} \cup \rhd^{\mathcal{CS}})^{+})_{mul}$,*

2. *if $f \in Lex$ then*

$$
\begin{aligned}
DepD_{(f,\Delta,\rho,\zeta)} = \{ & (\Delta_1 \vdash t_1, \dots \Delta_r \vdash t_r, \nabla_1 \vdash v_1 \dots \nabla_n \vdash v_n) \mid \\
& \Delta = \Delta_0 \subseteq \Delta_1 \subseteq \dots \Delta_r \subseteq \nabla_1 \subseteq \dots \nabla_n \text{ and for all } i \\
& \Delta \vdash t_i : P_i[\vec{t}/\vec{p}] \text{ and } \nabla_i \vdash v_i : b_i[\vec{t}/\vec{p}][\vec{v}/\vec{x}] \\
& \Delta_i \vdash p_i\rho \rightarrow^{*} t_i \\
& \Delta_i \vdash t_i \in \mathcal{SN}_{\Delta} \\
& \nabla_i \vdash v_i \in D_{(f,\Delta,\rho,\zeta)}(i)(\overrightarrow{\Delta \vdash t}, \nabla_1 \vdash v_1, \dots \nabla_{i-1} \vdash v_{i-1}) \}
\end{aligned}
$$

$$
D_{(f,\Delta,\rho,\zeta)}(i)(\overrightarrow{\Delta \vdash t}, \overrightarrow{\nabla \vdash v}) = \begin{cases} U_{(s,\nabla_{i-1},\vec{A}^i)} & \text{if } b_i = s(\vec{\alpha}')\vec{t}' \ s \in \mathcal{TC} \\ \mathcal{SN}_{\nabla_{i-1}} & \text{otherwise} \end{cases}
$$

where $U_{(s,\nabla,\vec{A})}$ is defined in 5.2.23 and $\vec{A}^i = \overrightarrow{(\nabla_{i-1} \vdash \alpha'([\vec{t}/\vec{p}] \cup [\vec{v}/\vec{x}]), I)}$ with $I_k = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash \alpha'_k]\!]_{([\vec{t}/\vec{p}] \cup [\vec{v}/\vec{x}]),\zeta,\nabla_{i-1}}$.

The ordering $\leadsto_{(f,\rho,\zeta)}^{\Delta}$ is a dependent product of relations $(\rightarrow_{sn}^{\Delta}, \dots \rightarrow_{sn}^{\Delta}, \leadsto_1, \dots \leadsto_n)_{lex}$ where $\leadsto_i$ is a relation on $D_{(f,\Delta,\rho\zeta)}(i)(\overrightarrow{\Delta \vdash t}, \overrightarrow{\nabla \vdash v})$ and depends on sequents $\overrightarrow{\Delta \vdash t}, \overrightarrow{\nabla \vdash v}$.

$$
\leadsto_i = \begin{cases} \rightarrow_{sn}^{\nabla_{i-1}} \cup >_{deg(\vec{A}^i)}^{\nabla_{i-1}} & \text{if } b_i = s(\vec{\alpha}')\vec{t}' \ s \in \mathcal{TC} \\ \rightarrow_{sn}^{\nabla_{i-1}} & \text{otherwise} \end{cases}
$$

Lemma 5.4.3 *The relation $\leadsto_{(f,\rho,\zeta)}^{\Delta}$ is well-founded on $DepD_{(f,\Delta,\rho,\zeta)}$.*

Proof
According to the status of f, the relation $\rightsquigarrow^{\Delta}_{(f,\rho,\zeta)}$ is either a multiset extension of $(\to^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})$, or a lexicographic dependent composition of $(\to^{\Delta}, \ldots \to^{\Delta}, (\to^{\nabla_0}_{sn} \cup >^{\nabla_0}_{deg(\vec{A}^1)}), \ldots (\to^{\nabla_{n-1}}_{sn} \cup >^{\nabla_{n-1}}_{deg(\vec{A}^n)})$.

The relation $(\to^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})$ is well-founded on $\mathcal{SN}_{\Delta}$ by Lemma 5.4.1 and consequently $(\to^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})_{mul}$ is well-founded on $\mathcal{SN}^{r+n}_{\Delta}$.

For every $\vec{t}, \vec{v}$, the relation $(\to^{\nabla_{k-1}}_{sn} \cup >^{\nabla_{k-1}}_{deg(\vec{A}^k)})$ is well-founded on the set $D_{(f,\Delta,\rho,\zeta)}(k)(\overrightarrow{\Delta \vdash t}, \overrightarrow{\nabla \vdash v})$ by Lemma 5.2.28 The proof that lexicographic composition of well-founded dependent relations is well-founded is standard.
□

Corollary 5.4.4 *The relation* $(>_{\mathcal{F}}, \emptyset, \emptyset, \emptyset, \rightsquigarrow^{\Delta}_{(f,\rho,\zeta)})_{lex}$ *is well-founded on the set DepDom, which consists of all tuples* $(f, \Delta, \rho, \zeta, T)$ *satisfying*

1. $f \in \mathcal{F}^{(r,n)}$, $f : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).c$,
2. Δ *is an environment,*
3. $\rho : \overrightarrow{p : P} \to \Delta$,
4. ζ *is a candidate assignment compatible with* ρ,
5. $T \in DepD_{(f,\Delta,\rho,\zeta)}$.

as a lexicographic (dependent) composition of well founded relations.

Remark In the relation defined above the second, third and fourth components are given only because of dependencies and they cannot be used as the decreasingness argument.

The proof of Fun lemma will be done by inspecting all possible HORPO cases. Since HORPO depends on its own extensions and the computable closure, there will be other lemmas using the induction hypothesis from Fun Lemma. The Computable Closure Lemma 5.4.7 and Recursive Comparison Lemma 5.4.8 correspond more or less to the $\mathfrak{P}(W)$ part of HORPO and their aim is to show that every term satisfying $\mathfrak{P}(W)$ belongs to the interpretation of its type. The Recursive Constructor Subterm Lemma 5.4.6 provides in addition the decreasingness argument and shows that the term on the right-hand side is smaller in $>^{\Delta}_{deg(\vec{A})}$ then the term on the left-hand side, therefore allowing to use $\succ^{\mathcal{RCS}}$ in *Lex* case in Fun Lemma. The corresponding lemma for constructor subterm and $\succ^{\mathcal{CS}}$ is obvious.

The rest of the section is organized as follows: first we give the statement of Fun Lemma, then we state and prove the lemmas about recursive constructor subterm extension, computable closure, and recursive comparison and finally we prove Fun Lemma itself.

Lemma 5.4.5 (Fun Lemma) *For every function symbol f of type $(p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c$, environment Δ, well-typed substitution $\rho : \overrightarrow{p : P} \to \Delta$ and candidate assignment ζ_ρ compatible with ρ, if $\vec{\Delta}$, $\vec{\alpha}$, $\vec{\nabla}$, $\vec{\sigma}$ satisfy $\Delta = \Delta_0 \subseteq \Delta_1 \subseteq \dots \Delta_r \subseteq \nabla_1 \subseteq \dots \nabla_n$, $\Delta_i \vdash p_i\rho \rightarrow^* \alpha_i$ for all i and*

- $\Delta_j \vdash \alpha_j \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{[\vec{\alpha}/\vec{p}],\zeta_\rho,\Delta_{j-1}}$, *for $j = 1,\dots r$,*
- $\nabla_j \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_{j-1}}$, *for $j = 1,\dots n$,*

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

The proof will be done by induction on $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash \alpha}, \overrightarrow{\nabla \vdash \sigma}))$ ordered by $(>_{\mathcal{F}}, \emptyset, \emptyset, \emptyset, \rightsquigarrow^{\Delta}_{(f,\rho,\zeta_\rho)})_{lex}$.

The lemma presented below states that $>_{deg}$ relation between term u_k and its recursive constructor subterm r (following from Lemma 5.2.31) is stable under substitution. Moreover, if u_k belongs to the interpretation of its type then the same property holds for r.

Lemma 5.4.6 (Recursive Constructor Subterm Lemma) *Let f be a function symbol of type $(p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c$, Δ an environment, ρ a well-typed substitution from $\overrightarrow{p : P}$ to Δ and ζ_ρ a candidate assignment compatible with ρ. Let $\Delta_1,\dots\Delta_r,\nabla_1,\dots\nabla_n$ be environments satisfying $\Delta = \Delta_0 \subseteq \Delta_1 \subseteq \dots \Delta_r \subseteq \nabla_1 \subseteq \dots \nabla_n$.*

Moreover, let $\Gamma = \overrightarrow{p : P}, G$ be an environment, $\vec{u}$ Γ-terms satisfying $\Gamma \vdash_{CC} f(\vec{p},\vec{u}) : c[\vec{u}/\vec{x}]$, $\theta : \Gamma \to \nabla_n$ a well-typed substitution and ζ a candidate assignment compatible with θ such that for all i $\Delta_i \vdash p_i\rho \rightarrow^* p_i\theta$, $\zeta_\rho|_{\nabla_n}(p_i) = \zeta(p_i)$ and

$$\Delta_j \vdash p_j\theta \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta_\rho,\Delta_{j-1}} \quad \text{for } j = 1,\dots r, \quad \text{and}$$
$$\nabla_j \vdash u_j\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{j-1}} \quad \text{for } j = 1,\dots n$$

If $b_k = s(\vec{a})\vec{t}$ and r is a recursive subterm of u_k such that $\Gamma \vdash_{CC} r : (x_1 : V_1)\dots(x_m : V_m).B$ then for all Γ''', Δ''', σ, ξ and $v_1,\dots v_m$ verifying

- $\Delta''' \supseteq \nabla_k$, $(\theta \cup \sigma) : \Gamma, \Gamma''' \to \Delta'''$, $\zeta \cup \xi$ *compatible with* $(\theta \cup \sigma)$,

- $\Gamma, \Gamma''' \vdash_{CC} v_i : V_i[\vec{v}/\vec{x}]$,
- $\Delta''' \vdash v_i(\theta \cup \sigma) \in [\![\Gamma, \Gamma''' \vdash V_i[\vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$

we have

1. $\Delta''' \vdash (r\, v_1\, \ldots v_m)(\theta \cup \sigma) \in [\![\Gamma, \Gamma''' \vdash B[\vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$ *and*
2. $\nabla_k \vdash u_k\theta >^{\nabla_{k-1}}_{deg(\vec{A})} \Delta''' \vdash (r\, v_1\, \ldots v_m)(\theta \cup \sigma)$

where $\vec{A} = \overrightarrow{(\nabla_{k-1} \vdash a[\vec{u}/\vec{x}]\theta, I)}$ *with* $I_i = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{k-1}}$.

Proof

Let us denote by Δ' the environment ∇_{k-1} and by Δ'' the environment ∇_k. By the hypothesis we know that $\Delta'' \vdash u_k\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash s(\vec{a})\vec{t}]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'}$ which is equivalent to

$$\begin{aligned}\Delta'' \vdash u_k\theta \in \mu F(&\overrightarrow{(\Delta' \vdash a[\vec{u}/\vec{x}]\theta, [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash a]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})},\\ &\overrightarrow{(\Delta' \vdash t[\vec{u}/\vec{x}]\theta, [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash t]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})})\\ &= \mu F(\vec{A}, \overrightarrow{(\Delta' \vdash t[\vec{u}/\vec{x}]\theta, [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash t]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})})\end{aligned}$$

Let us denote the pair $(\vec{A}, \overrightarrow{(\Delta' \vdash t[\vec{u}/\vec{x}]\theta, [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash t]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})})$ by e.

Since r is a recursive subterm of u, by Lemma 5.2.30 there is a subterm of u of the form $c(\vec{a}[\vec{u}/\vec{x}], \vec{M})$, with $c : \overrightarrow{(q : Q)}\overrightarrow{(z : d)}.s(\vec{q})\vec{w}$, whose immediate subterm at recursive position j is r ($r = M_j$). Of course, we have $\Gamma \vdash r : d_j([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}])$.

If j is a nonfunctional position then $d_j = s(\vec{q})\vec{w}$ and $\Gamma \vdash_{CC} B \overset{*}{\leftrightarrow} (s(\vec{q})\vec{w})([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}])$. From Lemma 5.2.31 it follows that:

1. $\Delta'' \vdash r\theta \in U_{(s,\Delta',\vec{A})}$.
2. $\Delta'' \vdash u_k\theta >^{\Delta'}_{deg(\vec{A})} \Delta'' \vdash r\theta$

Concerning degrees, by Lemma 5.2.21 $\Delta'' \vdash r\theta \geq^{\Delta'}_{deg(\vec{A})} \Delta''' \vdash r\theta$ and since $r\theta = r(\theta \cup \sigma)$ we get

$$\Delta'' \vdash u_k\theta >^{\Delta'}_{deg(\vec{A})} \Delta''' \vdash r(\theta \cup \sigma)$$

which is the second of the desired conclusions.

For the first conclusion, note that by Lemma 5.2.21 $\Delta''' \vdash r\theta \in U_{(s,\Delta',\vec{A})}$. By Lemma 5.2.24 $\Delta''' \vdash r\theta$ belongs to

$$\mu F(\vec{A}, \overrightarrow{(\Delta''' \vdash w[\vec{a}[\vec{u}/\vec{x}]\theta/\vec{q}, \vec{M}\theta/\vec{z}], W)})$$

for any appropriate $\vec{W}$ and by Lemma 5.2.20 to

$$\mu F(\overrightarrow{(\Delta''' \vdash a[\vec{u}/\vec{x}]\theta, [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'}|_{\Delta'''})}, \overrightarrow{(\Delta''' \vdash w[\vec{a}[\vec{u}/\vec{x}]\theta/\vec{q}, \vec{M}\theta/\vec{z}], W)})$$

By Lemma 5.2.4 $([\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})|_{\Delta'''} = [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'''}$ and by Lemma 5.3.4 $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'''} = [\![\Gamma, \Gamma''' \vdash a_i[\vec{u}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$. Now, let us take $W_i = [\![\Gamma, \Gamma''' \vdash w_i[\vec{a}[\vec{u}/\vec{x}]/\vec{q}, \vec{M}/\vec{z}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$ if w_i is a non-object and $W_i = \emptyset$ otherwise. Then

$$\begin{aligned}&\mu F(\overrightarrow{(\Delta''' \vdash a[\vec{u}/\vec{x}]\theta, [\![\Gamma, \Gamma''' \vdash a[\vec{u}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''})}, \overrightarrow{(\Delta''' \vdash w[\vec{a}[\vec{u}/\vec{x}]\theta/\vec{q}, \vec{M}\theta/\vec{z}], W)}) \\ &\quad = [\![\Gamma, \Gamma''' \vdash (s(\vec{q})\vec{w})[\vec{a}[\vec{u}/\vec{x}]/\vec{q}, \vec{M}/\vec{z}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''} = [\![\Gamma, \Gamma''' \vdash B]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}\end{aligned}$$

Consequently we have $\Delta''' \vdash r\theta \in [\![\Gamma, \Gamma''' \vdash B]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$, which is the desired conclusion.

If j is a functional position then $d_j = \overrightarrow{(x:T)}.s(\vec{q})\vec{w}$ and since $d_j([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}])$ and $(x_1 : V_1) \dots (x_m : V_m).B$ are both types of r we have $\Gamma, \Gamma''' \vdash_{CC} V_i[\vec{v}/\vec{x}] \overset{*}{\leftrightarrow} T_i([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}] \cup [\vec{v}/\vec{x}])$ for all i and $\Gamma, \Gamma''' \vdash_{CC} B[\vec{v}/\vec{x}] \overset{*}{\leftrightarrow} (s(\vec{q})\vec{w})([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}] \cup [\vec{v}/\vec{x}])$.

By assumption we have $\Delta''' \vdash v_i(\theta\cup\sigma) \in [\![\Gamma, \Gamma''' \vdash V_i[\vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$ for all i. By Lemma 5.3.6 we have

$$[\![\Gamma, \Gamma''' \vdash V_i[\vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''} = [\![\Gamma, \Gamma''' \vdash T_i([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}] \cup [\vec{v}/\vec{x}])]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$$

and by Lemma 5.3.3

$$\begin{aligned}&[\![\Gamma, \Gamma''' \vdash T_i([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}] \cup [\vec{v}/\vec{x}])]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''} = \\ &\qquad [\![\overrightarrow{q:Q}, \overrightarrow{z:d}, \overrightarrow{x:T} \vdash T_i]\!]_{([\vec{a}[\vec{u}/\vec{x}]\theta/\vec{q}] \cup [\vec{M}\theta/\vec{z}] \cup [\vec{v}(\theta\cup\sigma)/\vec{x}]),\hat{\xi},\Delta'''}\end{aligned}$$

where $y\hat{\xi} = [\![\Gamma, \Gamma''' \vdash y([\vec{a}[\vec{u}/\vec{x}]/\vec{q}] \cup [\vec{M}/\vec{z}] \cup [\vec{v}/\vec{x}])]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$ for $y \in \{\vec{q}, \vec{z}, \vec{x}\} \cap \mathcal{V}ar^\Box$.

To use Lemma 5.2.31 we have to check that $q_i\hat{\xi} = I_i|_{\Delta'''}$. We have $q_i\hat{\xi} = [\![\Gamma \vdash a_i[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\Delta'''}$ and $I_i|_{\Delta'''} = ([\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})|_{\Delta'''}$ which equals $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'''}$ by Lemma 5.2.4 and $[\![\Gamma \vdash a_i[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\Delta'''}$ by Lemma 5.3.4.

So, we can use Lemma 5.2.31 and get

1. $\Delta''' \vdash (r\, v_1 \, \ldots v_m)(\theta \cup \sigma) \in U_{(s,\Delta',\vec{A})}$ and

2. $\Delta'' \vdash u_k\theta >^{\Delta'}_{deg(\vec{A})} \Delta''' \vdash (r\, v_1 \, \ldots v_m)(\theta \cup \sigma)$.

To show that $\Delta''' \vdash (r\, v_1 \, \ldots v_m)(\theta \cup \sigma) \in [\![\Gamma, \Gamma''' \vdash B[\vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$, we proceed like in the nonfunctional case. By Lemma 5.2.24 $\Delta''' \vdash (r\, v_1 \, \ldots v_m)(\theta \cup \sigma)$ belongs to

$$\mu F(\vec{A}, \overrightarrow{(\Delta''' \vdash w([\vec{a}[\vec{u}/\vec{x}]\theta/\vec{q}] \cup [\vec{M}\theta/\vec{z}] \cup [\vec{v}(\theta \cup \sigma)/\vec{x}]), W)})$$

for any appropriate $\vec{W}$, and by Lemma 5.2.20 to

$$\begin{aligned}\mu F(&\overrightarrow{(\Delta''' \vdash a[\vec{u}/\vec{x}]\theta, ([\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})|_{\Delta'''})},\\ &\overrightarrow{(\Delta''' \vdash w[\vec{a}[\vec{u}/\vec{x}]\theta/\vec{q}, \vec{M}\theta/\vec{z}, \vec{v}(\theta \cup \sigma)/\vec{x}], W)})\end{aligned}$$

By Lemma 5.2.4 $([\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'})|_{\Delta'''} = [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\Delta'''}$ and by Lemma 5.3.4 $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash a_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta,\Delta'''} = [\![\Gamma, \Gamma''' \vdash a_i[\vec{u}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$. Now, let us denote $a_i[\vec{u}/\vec{x}]$ by $\vec{a'_i}$ and let us take

$$W_i = [\![\Gamma, \Gamma''' \vdash w_i[\vec{a}[\vec{u}/\vec{x}]/\vec{q}, \vec{M}/\vec{z}, \vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$$

if w_i is a non-object and $W_i = \emptyset$ otherwise. Then

$$\begin{aligned}&\mu F(\overrightarrow{(\Delta''' \vdash a'\theta, [\![\Gamma, \Gamma''' \vdash a']\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''})},\\ &\overrightarrow{(\Delta''' \vdash w[\vec{a}'\theta/\vec{q}, \vec{M}\theta/\vec{z}, \vec{v}(\theta \cup \sigma)/\vec{x}], [\![\Gamma, \Gamma''' \vdash w[\vec{a}'/\vec{q}, \vec{M}/\vec{z}, \vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''})})\\ &= [\![\Gamma, \Gamma''' \vdash (s(\vec{q})\vec{w})[\vec{a}'/\vec{q}, \vec{M}/\vec{z}, \vec{v}/\vec{x}]]\!]_{\theta\cup\xi,\zeta\cup\xi,\Delta'''} = [\![\Gamma, \Gamma''' \vdash B[\vec{v}/\vec{x}]]\!]_{\theta\cup\xi,\zeta\cup\xi,\Delta'''}\end{aligned}$$

Consequently we have $\Delta''' \vdash (r\, v_1 \, \ldots v_m)(\theta \cup \sigma) \in [\![\Gamma, \Gamma''' \vdash B[\vec{v}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\Delta'''}$.
□

Lemma 5.4.7 (Computable Closure Lemma) *Let f be a function symbol of type $(p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).c$, Δ an environment, ρ a well-typed substitution from $\overrightarrow{p:P}$ to Δ and ζ_ρ a candidate assignment compatible with ρ. Let $\Delta_1, \ldots \Delta_r, \nabla_1, \ldots \nabla_n$ be environments satisfying $\Delta = \Delta_0 \subseteq \Delta_1 \subseteq \ldots \Delta_r \subseteq \nabla_1 \subseteq \ldots \nabla_n$.*

Moreover, let $\Gamma = \overrightarrow{p:P}, G$ be an environment, $\vec{u}$ Γ-terms satisfying $\Gamma \vdash_{CC} f(\vec{p}, \vec{u}) : c[\vec{u}/\vec{x}]$, $\theta : \Gamma \to \nabla_n$ a well-typed substitution and ζ a *candidate assignment compatible with* θ *such that for all* i $\Delta_i \vdash p_i\rho \rightarrow^* p_i\theta$, $\zeta_\rho|_{\nabla_n}(p_i) = \zeta(p_i)$ *and*

$$\Delta_j \vdash p_j\theta \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta_\rho,\Delta_{j-1}} \quad \textit{for } j = 1, \ldots r, \quad \textit{and}$$
$$\nabla_j \vdash u_j\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{j-1}} \quad \textit{for } j = 1, \ldots n$$

Suppose also that for every $(g, \Delta', \rho', \zeta', (\overrightarrow{\Delta' \vdash \beta}, \overrightarrow{\nabla' \vdash \nu}))$ *smaller than* $(f, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla \vdash u\theta}))$ *in* $(>_{\mathcal{F}}, \emptyset, \emptyset, \emptyset, \rightsquigarrow^{\Delta}_{(f,\rho,\zeta_\rho)})_{lex}$ *if* $g : \overrightarrow{(q : Q)}\overrightarrow{(y : d)}.e \in \mathcal{F}^{(r',n')}$, $\Delta = \Delta'_0 \subseteq \Delta'_1 \ldots \Delta'_{r'} \subseteq \nabla'_1 \ldots \nabla'_{n'}$, $\Delta' \vdash q_i\rho' \rightarrow^{\Delta'}_{sn} \Delta'_i \vdash \beta_i$ *for all* i *and*

- $\Delta'_j \vdash \beta_j \in [\![\overrightarrow{q : Q} \vdash Q_j]\!]_{[\vec{\beta}/\vec{q}],\zeta',\Delta'_{j-1}}$ *for* $j = 1, \ldots r'$,
- $\nabla'_j \vdash \nu_j \in [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash d_j]\!]_{[\vec{\beta}/\vec{q}]\cup[\vec{\nu}/\vec{y}],\zeta',\nabla'_{j-1}}$ *for* $j = 1, \ldots n'$,

then $\nabla_{n'} \vdash g(\vec{\beta}, \vec{\nu}) \in [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash e]\!]_{[\vec{\beta}/\vec{q}]\cup[\vec{\nu}/\vec{y}],\zeta',\nabla_{n'}}$

Under these assumptions for every $t \in CCl(\Gamma \vdash f(\vec{p}, \vec{u}))$ *there exists* T *satisfying:*

1. $\Gamma \vdash_{CC} t : T$ *and*
2. $\nabla_n \vdash t\theta \in [\![\Gamma \vdash T]\!]_{\theta,\zeta,\nabla_n}$.

Proof
To get the conclusion of Lemma 5.4.7 we have to prove slightly more general statement.

We will show that for every $(BV,\ t,\ T) \in CCl(\Gamma \vdash f(\vec{p}, \vec{u}))$ with $BV = y_1 : A_1, \ldots y_m : A_m$ and for every $\hat{\Delta}_1, \ldots \hat{\Delta}_m$, σ and ξ satisfying

$$\hat{\Delta}_i \vdash y_i\sigma \in [\![\Gamma, BV \vdash A_i]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_i}, \quad \hat{\Delta}_i \supseteq \hat{\Delta}_{i-1},$$
$$y_i\xi \in \mathcal{C}_{\hat{\Delta}_i, y_i\sigma} \text{ if } \ y_i \in \mathcal{V}ar^{\Box} \text{ and } y_i\xi = \emptyset \text{ otherwise}$$

(and assuming $\hat{\Delta}_0 = \nabla_n$), we have:

1. $\Gamma, BV \vdash_{CC} t : T$ and
2. $\hat{\Delta}_m \vdash t(\theta \cup \sigma) \in [\![\Gamma, BV \vdash T]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$

We obtain the conclusion of our lemma when BV is empty.

Property 1 holds by the definition of Γ-tuple and Lemma 3.2.3. The second one can be proved by induction on the derivation of $(BV,\ t,\ T) \in CCl(\Gamma \vdash f(\vec{p},\vec{u}))$.

Terms from the initial set $Init(\Gamma \vdash f(\vec{p},\vec{u}))$ satisfy the demanded property, because by the hypothesis we have:

$$\Delta_j \vdash p_j\theta \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\Delta_{j-1}} \quad \text{for } j=1,\ldots r, \text{ and}$$
$$\nabla_j \vdash u_j\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{j-1}} \quad \text{for } j=1,\ldots n$$

Note, that $p_j\theta = p_j(\theta\cup\sigma)$ and $u_j\theta = u_j(\theta\cup\sigma)$ because $\mathcal{FV}(\vec{p},\vec{u})\cap BV = \emptyset$. By property (S3) from the definition of the family of candidates and the properties of restriction we have

$$\hat{\Delta}_m \vdash p_j(\theta\cup\sigma) \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\hat{\Delta}_m} \quad \text{for } j=1,\ldots r, \text{ and}$$
$$\hat{\Delta}_m \vdash u_j(\theta\cup\sigma) \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\hat{\Delta}_m} \quad \text{for } j=1,\ldots n$$

Since $[\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\hat{\Delta}_m} = [\![\Gamma \vdash P_j]\!]_{\theta,\zeta,\hat{\Delta}_m}$ by Lemma 5.3.4 and $[\![\Gamma \vdash P_j]\!]_{\theta,\zeta,\hat{\Delta}_m} = [\![\Gamma, BV \vdash P_j]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ by Lemma 5.3.1 (similarly $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\hat{\Delta}_m} = [\![\Gamma, BV \vdash b_j[\vec{u}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$) we get the desired conclusions

$$\hat{\Delta}_m \vdash p_j(\theta\cup\sigma) \in [\![\Gamma, BV \vdash P_j]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} \quad \text{for } j=1,\ldots r, \text{ and}$$
$$\hat{\Delta}_m \vdash u_j(\theta\cup\sigma) \in [\![\Gamma, BV \vdash b_j[\vec{u}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} \quad \text{for } j=1,\ldots n$$

Accessible set, $Acc(\Gamma \vdash f(\vec{p},\vec{u}))$, is obtained from the initial one by applying the rules

$$\frac{(BV,\ c(\vec{a},\vec{N}),\ s(\vec{a})\vec{u})}{(BV,\ a_i,\ Q_i[\vec{a}/\vec{q}])} \qquad\qquad \frac{(BV,\ c(\vec{a},\vec{N}),\ s(\vec{a})\vec{u})}{(BV,\ N_i,\ d_i[\vec{a}/\vec{q},\vec{N}/\vec{z}])}$$

where $c : (q_1 : Q_1)\ldots(q_{r'} : Q_{r'})(z_1 : d_1)\ldots(z_k : d_k).s(\vec{q})w_1\ldots w_n$ is a constructor of $s : (q_1 : Q_1)\ldots(q_{r'} : Q_{r'})(x_1 : b'_1)\ldots(x_{n'} : b'_{n'}).\star$.

In this case, we get

$$\hat{\Delta}_m \vdash a_i(\theta\cup\sigma) \in [\![\Gamma, BV \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$
$$\hat{\Delta}_m \vdash N_i(\theta\cup\sigma) \in [\![\Gamma, BV, \overrightarrow{z:d[\vec{a}/\vec{q}]} \vdash d_i[\vec{a}/\vec{q}]]\!]_{\theta\cup\sigma\cup[\vec{N}/\vec{z}],\zeta\cup\xi,\hat{\Delta}_m}$$

by the **decomposition** case of Inductive Type Lemma 5.3.7 applied to $\hat{\Delta}_m \vdash c(\vec{a},\vec{N})(\theta\cup\sigma) \in [\![\Gamma, BV \vdash s(\vec{a})\vec{u}]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$, which is true by induction hypothesis. Since by Lemma 5.3.2

$$[\![\Gamma, BV, \overrightarrow{z:d[\vec{a}/\vec{q}]} \vdash d_i[\vec{a}/\vec{q}]]\!]_{\theta\cup\sigma\cup[\vec{N}/\vec{z}],\zeta\cup\xi,\hat{\Delta}_m} = [\![\Gamma, BV \vdash d_i[\vec{a}/\vec{q},\vec{N}/\vec{z}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$

the proof of this case is complete.

Other tuples in $CCl(\Gamma \vdash f(\vec{p}, \vec{u}))$ result from applying the following rules:

Axiom

$$\overline{(\emptyset,\ \star,\ \Box)}$$

It is easily seen that $\hat{\Delta}_m \vdash \star(\theta \cup \sigma) \in [\![\Gamma, BV \vdash \Box]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$, since the interpretation of $\Box$ equals $\mathcal{SN}_{\hat{\Delta}_m,\Box}$.

Var

$$\frac{\Gamma, BV \vdash_{CC} A : p}{((BV, x : A),\ x,\ A)} \qquad (x \in \mathcal{V}ar^p \backslash dom(\Gamma, BV),\ p \in \{\star, \Box\})$$

The fact that $\hat{\Delta}_m \vdash x(\theta \cup \sigma) \in [\![\Gamma, BV, x : A \vdash A]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ follows from the hypothesis about σ and ξ.

Weak

$$\frac{(BV,\ t,\ T) \qquad \Gamma, BV \vdash_{CC} A : p}{((BV, x : A),\ t,\ T)} \ (x \in \mathcal{V}ar^p \backslash dom(\Gamma, BV),\ p \in \{\star, \Box\})$$

By induction hypothesis we have $\hat{\Delta}_m \vdash t(\theta\cup\sigma) \in [\![\Gamma, BV \vdash T]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ and by Lemma 5.3.1 we get $\hat{\Delta}_{m+1} \vdash t(\theta \cup \sigma^+) \in [\![\Gamma, BV, x : A \vdash T]\!]_{\theta\cup\sigma^+,\zeta\cup\xi^+,\hat{\Delta}_{m+1}}$ for any $\sigma^+ = \sigma\cup[M/x]$, $\xi^+ = \xi\cup[C/x]$, $\hat{\Delta}_{m+1} \supseteq \hat{\Delta}_m$, M satisfying that $\hat{\Delta}_{m+1} \vdash M : A(\theta \cup \sigma)$ and any $C \in \mathcal{C}_{\hat{\Delta}_{m+1},M}$.

Abstraction

$$\frac{\begin{array}{c}((BV, y : A),\ t,\ T) \qquad (BV,\ A,\ p_2)\\ \Gamma, BV, y : A \vdash_{CC} T : p_1\end{array}}{(BV,\ \lambda y\!:\!A.t,\ (y\!:\!A).T)} (x \notin dom(\Gamma, BV),\ p_1, p_2 \in \{\star, \Box\})$$

To show that $\hat{\Delta}_m \vdash (\lambda y\!:\!A.t)(\theta \cup \sigma) \in [\![\Gamma, BV \vdash (y\!:\!A).T]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ it suffices to check whether $\hat{\Delta}_{m+1} \vdash @((\lambda y\!:\!A.t)(\theta \cup \sigma), M) \in [\![\Gamma, BV, y : A \vdash T]\!]_{\theta\cup\sigma\cup[M/y],\zeta\cup\xi\cup[C/y],\hat{\Delta}_{m+1}}$ for any $\hat{\Delta}_{m+1} \supseteq \hat{\Delta}_m$, any $\hat{\Delta}_{m+1} \vdash M \in [\![\Gamma, BV \vdash A]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ and any $C \in \mathcal{C}_{\hat{\Delta}_{m+1},M}$.

By induction hypothesis we know that $\hat{\Delta}_{m+1} \vdash t(\theta \cup \sigma \cup [M/y]) \in [\![\Gamma, BV, y : A \vdash T]\!]_{\theta\cup\sigma\cup[M/y],\zeta\cup\xi\cup[C/y],\hat{\Delta}_{m+1}}$. But $@((\lambda y : A.t)(\theta \cup \sigma), M) \rightarrow_\beta t(\theta \cup \sigma \cup [M/y])$, which gives our claim by Lemma 5.1.17 (with $\Delta = \hat{\Delta}_m$, $\Delta' = \hat{\Delta}_{m+1}$; $\hat{\Delta}_{m+1} \vdash A \in \mathcal{SN}_{\hat{\Delta}_{m+1}}$ because $\hat{\Delta}_m \vdash A \in \mathcal{SN}_{\hat{\Delta}_m}$ by induction hypothesis applied to $(BV,\ A,\ p_2)$ and $\hat{\Delta}_{m+1} \supseteq \hat{\Delta}_m$).

Application

$$\frac{(BV,\ t_1,\ (x\!:\!A).B) \qquad (BV,\ t_2,\ A)}{(BV,\ @(t_1,t_2),\ B[t_2/x])}$$

By induction hypothesis we know that

$$\hat{\Delta}_m \vdash t_1(\theta \cup \sigma) \in [\![\Gamma, BV \vdash (x\!:\!A).B]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$
$$\hat{\Delta}_m \vdash t_2(\theta \cup \sigma) \in [\![\Gamma, BV \vdash A]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$

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

$$\hat{\Delta}_m \vdash @(t_1,t_2)(\theta \cup \sigma) \in [\![\Gamma, BV, x : A \vdash B]\!]_{\theta\cup\sigma\cup[t_2(\theta\cup\sigma)/x],\zeta\cup\xi\cup\xi',\hat{\Delta}_m}$$

where $\xi' = [[\![\Gamma, BV \vdash t_2]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}/x]$ if t_2 is a non-object and $\xi' = \emptyset$ otherwise.

Now, by Lemma 5.3.2

$$[\![\Gamma, BV, x : A \vdash B]\!]_{\theta\cup\sigma\cup[t_2/x],\zeta\cup\xi\cup[[\![\Gamma,BV\vdash t_2]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}/x],\hat{\Delta}_m} = [\![\Gamma, BV \vdash B[t_2/x]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$

and the proof is complete.

Precedence

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ \ (BV,\ t_i,\ Q_i[\vec{t}/\vec{q}]) \qquad \forall i\ \ (BV,\ w_i,\ d_i[\vec{t}/\vec{q},\vec{w}/\vec{y}])}{(BV,\ g(\vec{t},\vec{w}),\ e[\vec{t}/\vec{q},\vec{w}/\vec{y}])}$$

if $f >_{\mathcal{F}} g$, $g : (q_1 : Q_1) \dots (q_{r'} : Q_{r'})(y_1 : d_1) \dots (y_{n'} : d_{n'}).e$.

By induction hypothesis we know that

$$\hat{\Delta}_m \vdash t_i(\theta \cup \sigma) \in [\![\Gamma, BV \vdash Q_i[\vec{t}/\vec{q}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} \ \text{ for all } i = 1, \dots r'$$
$$\hat{\Delta}_m \vdash w_i(\theta \cup \sigma) \in [\![\Gamma, BV \vdash d_i[\vec{t}/\vec{q},\vec{w}/\vec{y}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} \ \text{ for all } i = 1, \dots n'$$

Moreover, by Lemma 5.3.3 we have

$$[\![\Gamma, BV \vdash Q_i[\vec{t}/\vec{q}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} = [\![\overrightarrow{q : Q} \vdash Q_i]\!]_{[\vec{t}(\theta\cup\sigma)/\vec{q}],\zeta',\hat{\Delta}_m}$$
$$[\![\Gamma, BV \vdash d_i[\vec{t}/\vec{q},\vec{w}/\vec{y}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} = [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash d_i]\!]_{[\vec{t}(\theta\cup\sigma)/\vec{q},\vec{w}(\theta\cup\sigma)/\vec{y}],\zeta',\hat{\Delta}_m}$$

with $q_i\zeta' = [\![\Gamma, BV \vdash t_i]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ and $y_i\zeta' = [\![\Gamma, BV \vdash w_i]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$.

Note that $(g, \hat{\Delta}_m, [\vec{t}(\theta \cup \sigma)/\vec{q}], \zeta', (\overrightarrow{\hat{\Delta}_m \vdash t(\theta \cup \sigma)}, \overrightarrow{\hat{\Delta}_m \vdash w(\theta \cup \sigma)}))$ belongs to *DepDom*, as

1. $\hat{\Delta}_m \vdash t_i(\theta \cup \sigma), \hat{\Delta}_m \vdash w_i(\theta \cup \sigma) \in \mathcal{SN}_{\hat{\Delta}_m}$, because they belong to the corresponding interpretations and interpretations are subsets of strongly normalizing terms,
2. if $d_i = s(\vec{\alpha'})\vec{t'}$ then $\hat{\Delta}_m \vdash w_i(\theta \cup \sigma) \in U_{(s,\hat{\Delta}_m,\vec{A^i})}$, where

$$\vec{A^i} = \overrightarrow{(\hat{\Delta}_m \vdash \alpha'[\vec{t}(\theta \cup \sigma)/\vec{q}, \vec{w}(\theta \cup \sigma)/\vec{y}], I)} \text{ with}$$
$$I_k = [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash \alpha'_k]\!]_{[\vec{t}(\theta\cup\sigma)/\vec{q},\vec{w}(\theta\cup\sigma)/\vec{y}],\zeta',\hat{\Delta}_m}$$

because

$$[\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash s(\vec{\alpha'})\vec{t'}]\!]_{[\vec{t}(\theta\cup\sigma)/\vec{q},\vec{w}(\theta\cup\sigma)/\vec{y}],\zeta',\hat{\Delta}_m} = \mu F(\vec{A^i}, \overrightarrow{(\hat{\Delta}_m \vdash t'[\vec{t}(\theta \cup \sigma)/\vec{q}, \vec{w}(\theta \cup \sigma)/\vec{y}], W)})$$

with $W_k = [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash t'_k]\!]_{[\vec{t}(\theta\cup\sigma)/\vec{q},\vec{w}(\theta\cup\sigma)/\vec{y}],\zeta',\hat{\Delta}_m}$

Moreover $(g, \hat{\Delta}_m, [\vec{t}(\theta\cup\sigma)/\vec{q}], \zeta', (\overrightarrow{\hat{\Delta}_m \vdash t(\theta \cup \sigma)}, \overrightarrow{\hat{\Delta}_m \vdash w(\theta \cup \sigma)}))$ is smaller than $(f, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla \vdash u\theta}))$ because $f >_{\mathcal{F}} g$.

Now, it is sufficient to use the hypothesis of the lemma about elements of $DepDom$ to get

$$\hat{\Delta}_m \vdash g(\vec{t}, \vec{w})(\theta \cup \sigma) \in [\![\overrightarrow{q : Q}, \overrightarrow{y : d} \vdash e]\!]_{[\vec{t}(\theta\cup\sigma)/\vec{q},\vec{w}(\theta\cup\sigma)/\vec{y}],\zeta',\hat{\Delta}_m}$$

which is equivalent to the desired conclusion $\hat{\Delta}_m \vdash g(\vec{t}, \vec{w})(\theta \cup \sigma) \in [\![\Gamma, BV \vdash e[\vec{t}/\vec{q}, \vec{w}/\vec{y}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ once again by Lemma 5.3.3.

Recursive call

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i \ \ (BV,\ w_i,\ b_i[\vec{w}/\vec{x}])}{(BV,\ f(\vec{p}, \vec{w}),\ c[\vec{w}/\vec{x}])}$$

and $\Gamma \vdash \vec{p}, \vec{u} (\succ^{\mathcal{RCS}}_{BV,f(\vec{p},\vec{u})})_{lex} \vec{p}, \vec{w}$ if f has lexicographic status or $\Gamma \vdash \vec{p}, \vec{u} \succ^{\mathcal{CS}}_{mul} \vec{p}, \vec{w}$ if f has a multiset status.

If f has multiset status then by assumptions of the lemma we have

$$\Delta_i \vdash p_i\theta \in [\![\overrightarrow{p : P} \vdash P_i]\!]_{\theta,\zeta_\rho,\Delta_{i-1}} \quad \text{for } i = 1, \dots r$$

By induction hypothesis

$$\hat{\Delta}_m \vdash w_i(\theta \cup \sigma) \in [\![\Gamma, BV \vdash b_i[\vec{w}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} \quad \text{for } i = 1, \dots n$$

and by Lemma 5.3.4 for every i

$$[\![\Gamma, BV \vdash b_i[\vec{w}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m} = [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_i]\!]_{[\vec{p}(\theta\cup\sigma)/p]\cup[\vec{w}(\theta\cup\sigma)/\vec{x}],\zeta',\hat{\Delta}_m}$$

with $p_i\zeta' = [\![\Gamma, BV \vdash p_i]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ and $x_i\zeta' = [\![\Gamma, BV \vdash w_i]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$. Since for all i $p_i\zeta' = p_i\zeta_\rho$ and $p_i(\theta\cup\sigma) = p_i\theta$ we get

$$\hat{\Delta}_m \vdash w_i(\theta\cup\sigma) \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_i]\!]_{[\vec{p}\theta/p]\cup[\vec{w}(\theta\cup\sigma)/\vec{x}],\zeta_\rho,\hat{\Delta}_m} \quad \text{for } i = 1, \ldots n$$

The proof that $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\hat{\Delta}_m \vdash w(\theta\cup\sigma)})) \in DepDom$ is easy, as for all i $\Delta_i \vdash p_i\theta$ and $\hat{\Delta}_m \vdash w_i(\theta\cup\sigma)$ belong to $\mathcal{SN}_\Delta$.

For the decreasingness argument, note that $\Gamma \vdash \vec{p}, \vec{u} \succ^{\mathcal{CS}}_{mul} \vec{p}, \vec{w}$ implies that $\vec{w}$ are Γ-terms and that $w_i(\theta\cup\sigma) = w_i\theta$. We have to show $\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla \vdash u\theta}((\rightarrow^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})^+)_{mul}\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\hat{\Delta}_m \vdash w\theta}$. For this, it is sufficient that $\Gamma \vdash u_i \succ^{\mathcal{CS}} w_k$ implies $\nabla_i \vdash u_i\theta(\rightarrow^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})^+\hat{\Delta}_m \vdash w_k\theta$, which is true by Lemma 5.4.1.

We may now use the assumption of the lemma about elements of $DepDom$ to get $\hat{\Delta}_m \vdash f(\vec{p}, \vec{w})(\theta\cup\sigma) \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash c]\!]_{\theta\cup[\vec{w}(\theta\cup\sigma)/\vec{x}],\zeta,\hat{\Delta}_m}$ and consequently (by Lemma 5.3.4) $\hat{\Delta}_m \vdash f(\vec{p}, \vec{w})(\theta\cup\sigma) \in [\![\Gamma, BV \vdash c[\vec{w}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$.

If f has lexicographic status then $\Gamma \vdash \vec{p}, \vec{u}(\succ^{\mathcal{RCS}}_{BV,f(\vec{p},\vec{u})})_{lex}\vec{p}, \vec{w}$ and there exists an index k such that $\Gamma \vdash u_k(\succ^{\mathcal{RCS}}_{BV,f(\vec{p},\vec{u})})w_k$ and $w_i = u_i$ for $i < k$. By the assumption of the lemma we have

$$\Delta_i \vdash p_i\theta \in [\![\overrightarrow{p:P} \vdash P_i]\!]_{\theta,\zeta_\rho,\Delta_{i-1}} \quad \text{for } i = 1, \ldots r$$

We have also

$$\nabla_i \vdash w_i\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_i]\!]_{[\vec{w}/\vec{x}]\theta,\zeta_\rho,\nabla_{i-1}}$$

for $i < k$ and

$$\hat{\Delta}_m \vdash w_k\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_k]\!]_{[\vec{w}/\vec{x}]\theta,\zeta_\rho,\nabla_{k-1}}$$

By induction hypothesis for $i > k$ we have

$$\hat{\Delta}_m \vdash w_i(\theta\cup\sigma) \in [\![\Gamma, BV \vdash d_i[\vec{w}/\vec{x}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$

and consequently (the reasoning is like in Mul case) we get

$$\hat{\Delta}_m \vdash w_i(\theta\cup\sigma) \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_i]\!]_{[\vec{p}\theta/p]\cup[\vec{w}(\theta\cup\sigma)/\vec{x}],\zeta_\rho,\hat{\Delta}_m}$$

The proof that $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \nabla_1 \vdash w_1(\theta \cup \sigma), \ldots \nabla_{k-1} \vdash w_{k-1}(\theta \cup \sigma), \hat{\Delta}_m \vdash w_k(\theta \cup \sigma), \ldots \hat{\Delta}_m \vdash w_n(\theta \cup \sigma))) \in \mathit{DepDom}$ goes more or less like in **Precedence** case. More precisely

1. $\forall j\ \Delta \vdash p_j\theta \in \mathcal{SN}_\Delta$, $\forall i < k\ \nabla_i \vdash w_i(\theta \cup \sigma) \in \mathcal{SN}_{\nabla_{i-1}}$, $\hat{\Delta}_m \vdash w_k(\theta \cup \sigma) \in \mathcal{SN}_{\nabla_{k-1}}$, $\forall i > k\ \hat{\Delta}_m \vdash w_i(\theta \cup \sigma) \in \mathcal{SN}_{\hat{\Delta}_m}$, because these sequents belong to corresponding interpretations and interpretations are subsets of strongly normalizing terms,
2. if $b_i = s(\vec{\alpha'})\vec{t'}$ then $\Delta' \vdash w_i(\theta \cup \sigma) \in U_{(s, \Delta'', \vec{A^i})}$, where $\Delta' = \nabla_i$, $\Delta'' = \nabla_{i-1}$ for $i < k$, $\Delta' = \hat{\Delta}_m$, $\Delta'' = \nabla_{k-1}$ for $i = k$ and $\Delta' = \Delta'' = \hat{\Delta}_m$ for $i > k$ and
$$\vec{A^i} = \overrightarrow{(\Delta'' \vdash \alpha'([\vec{p}\theta/\vec{p}] \cup [\vec{w}(\theta \cup \sigma)/\vec{x}]), I)} \text{ with}$$
$$I_l = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash \alpha'_l]\!]_{[\vec{p}\theta/\vec{p}] \cup [\vec{w}(\theta \cup \sigma)/\vec{x}], \zeta_\rho, \Delta''}$$
because
$$[\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash s(\vec{\alpha'})\vec{t'}]\!]_{[\vec{p}\theta/p] \cup [\vec{w}(\theta \cup \sigma)/x], \zeta_\rho, \Delta''} = \mu F(\vec{A^i}, \overrightarrow{(\Delta'' \vdash t'[\vec{p}\theta/\vec{p}] \cup [\vec{w}(\theta \cup \sigma)/\vec{x}], W)})$$
with $W_l = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash t'_l]\!]_{[\vec{p}\theta/\vec{p}] \cup [\vec{w}(\theta \cup \sigma)/\vec{x}], \zeta_\rho, \Delta''}$

For the decreasingness argument, it is sufficient to prove that $\nabla_k \vdash u_k\theta(\rightarrow_{sn}^{\nabla k-1} \cup >_{deg(\vec{A^k})}^{\nabla_{k-1}})\hat{\Delta}_m \vdash w_k(\theta \cup \sigma)$ assuming $\Gamma \vdash u_k(\succ_{BV, f(\vec{p}, \vec{u})}^{\mathcal{RCS}})w_k$. If we have $\Gamma \vdash u_k(\succ_{BV, f(\vec{p}, \vec{u})}^{\mathcal{RCS}})w_k$ because $\Gamma \vdash u_k \succ w_k\ : b_k[\vec{w}/\vec{x}]$ holds then obviously w_k is a Γ-term, $w_k(\theta \cup \sigma) = w_k\theta$ and $\hat{\Delta}_m \vdash u_k\theta \rightarrow w_k\theta$. Consequently, $\nabla_k \vdash u_k\theta \rightarrow_{sn}^{\nabla_{k-1}} \hat{\Delta}_m \vdash w_k(\theta \cup \sigma)$.

Otherwise, $b_k = s(\vec{\alpha'})\vec{t'}$, there is a recursive subterm of u_k, a, and a left-flattening of w_k, $@(v_0, v_1, \ldots v_m)$, such that $\Gamma \vdash a \succeq v_0 : \overrightarrow{(z : T)}.B$ for some $\vec{T}, B$ and the tuples $(BV,\ v_i,\ T_i[\vec{v}/\vec{z}])$, $i = 1 \ldots m$, are already in the computable closure $CCl_f(\vec{p}, \vec{u})$. By induction hypothesis applied to these tuples we have $\hat{\Delta}_m \vdash v_i(\theta \cup \sigma) \in [\![\Gamma, BV \vdash T_i[\vec{v}/\vec{z}]]\!]_{\theta \cup \sigma, \zeta \cup \xi, \hat{\Delta}_m}$ for all i. By assumption we have $\nabla_k \vdash u_k\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_k]\!]_{[\vec{u}/\vec{x}]\theta, \zeta_\rho, \nabla_{k-1}}$. We may apply now Lemma 5.4.6 (with $\Delta''' = \hat{\Delta}_m$, $\Gamma''' = BV$) and get $\nabla_k \vdash u_k\theta >_{deg(\vec{A})}^{\nabla_{k-1}} \hat{\Delta}_m \vdash (a\, v_1\ \ldots v_m)(\theta \cup \sigma)$ for $\vec{A} = \overrightarrow{(\nabla_{k-1} \vdash \alpha'[\vec{u}/\vec{x}]\theta, I)}$ with $I_i = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash \alpha'_i]\!]_{[\vec{u}/\vec{x}]\theta, \zeta_\rho, \nabla_{k-1}}$.

Since $\alpha'_i[\vec{u}/\vec{x}]\theta = \alpha'_i[\vec{p}\theta/\vec{p}, \vec{w}\theta/\vec{x}]$ (as $u_j = w_j$ for all $j < k$) we have $\vec{A} = \vec{A}^k$. By Lemma 5.2.22 we get $\nabla_k \vdash u_k\theta >^{\nabla_{k-1}}_{deg(\vec{A}^k)} \hat{\Delta}_m \vdash (v_0\, v_1\, \ldots v_m)(\theta \cup \sigma)$, which is equivalent to $\nabla_k \vdash u_k\theta >^{\nabla_{k-1}}_{deg(\vec{A}^k)} \hat{\Delta}_m \vdash w_k(\theta \cup \sigma)$. That completes the decreasingness argument.

Like for Mul case we may now use the assumption of the lemma about elements of $DepDom$ to get $\hat{\Delta}_m \vdash f(\vec{p}, \vec{w})(\theta \cup \sigma) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\theta \cup [\vec{w}(\theta \cup \sigma)/\vec{x}], \zeta, \hat{\Delta}_m}$ and consequently (by Lemma 5.3.4) $\hat{\Delta}_m \vdash f(\vec{p}, \vec{w})(\theta \cup \sigma) \in [\![\Gamma, BV \vdash c[\vec{w}/\vec{x}]]\!]_{\theta \cup \sigma, \zeta \cup \xi, \hat{\Delta}_m}$.

Constructor

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ (BV,\ a_i,\ Q_i[\vec{a}/\vec{q}]) \qquad \forall i\ (BV,\ N_i,\ d_i[\vec{a}/\vec{q}, \vec{N}/\vec{z}])}{(BV,\ c(\vec{a}, \vec{N}),\ (s(\vec{q})\vec{w})[\vec{a}/\vec{q}, \vec{N}/\vec{z}])}$$

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 $s : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'})(x'_1 : b'_1) \ldots (x'_{n'} : b'_{n'}).\star$.

By induction hypothesis we know that:

$$\forall i \ \ \hat{\Delta}_m \vdash a_i(\theta \cup \sigma) \in [\![\Gamma, BV \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\theta \cup \sigma, \zeta \cup \xi, \hat{\Delta}_m}$$
$$\forall i \ \ \hat{\Delta}_m \vdash N_i(\theta \cup \sigma) \in [\![\Gamma, BV \vdash d_i[\vec{a}/\vec{q}, \vec{N}/\vec{z}]]\!]_{\theta \cup \sigma, \zeta \cup \xi, \hat{\Delta}_m}$$

By Lemma 5.3.2 and the star dependency condition imposed on the type of constructor c, according to which $\mathcal{FV}^{\Box}(\vec{d}) \subseteq \vec{q}$

$$[\![\Gamma, BV \vdash d_i[\vec{a}/\vec{q}, \vec{N}/\vec{z}]]\!]_{\theta \cup \sigma, \zeta \cup \xi, \hat{\Delta}_m} = [\![\Gamma, BV, \overrightarrow{z : d[\vec{a}/\vec{q}]} \vdash d_i[\vec{a}/\vec{q}]]\!]_{\theta \cup \sigma \cup [\vec{N}/\vec{z}], \zeta \cup \xi, \hat{\Delta}_m}$$

The desired conclusion follows from the **composition** case of Inductive Type Lemma 5.3.7 applied to the facts mentioned above.

Type constructor

$$\frac{\Gamma, BV \vdash_{CC} \star : \Box \qquad \forall i\ (BV,\ a_i,\ Q_i[\vec{a}/\vec{q}])}{(BV,\ s(\vec{a}),\ (x'_1 : b'_1) \ldots (x'_{n'} : b'_{n'}).\star)}$$

if $s : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'})(x'_1 : b'_1) \ldots (x'_{n'} : b'_{n'}).\star \in \mathcal{TC}$.

We have to show that $\hat{\Delta}_m \vdash s(\vec{a})(\theta \cup \sigma) \in [\![\Gamma, BV \vdash (x'_1 : b'_1) \ldots (x'_{n'} : b'_{n'}).\star]\!]_{\theta \cup \sigma, \zeta \cup \xi, \hat{\Delta}_m}$. By the definition of the interpretation of the product type this is equivalent to $\hat{\nabla}_n \vdash s(\vec{a})(\theta \cup \sigma)\vec{u} \in [\![\Gamma, BV, \overrightarrow{x' : b'} \vdash$

$\star]\!]_{\theta\cup\sigma\cup[\vec{u}/\vec{x}'],\zeta\cup\xi\cup[\vec{C}/\vec{x}'],\hat{\nabla}_n}$ for any $\hat{\nabla}_1,\ldots\hat{\nabla}_n$, $\vec{u}$, $\vec{C}$ satisfying $\hat{\Delta}_m \subseteq \hat{\nabla}_1 \ldots \subseteq \hat{\nabla}_n$, $\hat{\nabla}_i \vdash u_i \in [\![\Gamma, BV, \overrightarrow{x' : b'} \vdash b'_i]\!]_{\theta\cup\sigma\cup[\vec{u}/\vec{x}'],\zeta\cup\xi\cup[\vec{C}/\vec{x}'],\hat{\nabla}_i}$ and $C_i \in \mathcal{C}_{\hat{\nabla}_i,u_i}$.

Since the interpretation of $\star$ is just $\mathcal{SN}_{\hat{\nabla}_n,\star}$ it suffices that $\hat{\nabla}_n \vdash s(\vec{a})(\theta\cup\sigma)\vec{u} \in \mathcal{SN}_{\hat{\nabla}_n}$. By induction hypothesis we have

$$\hat{\Delta}_m \vdash a_i(\theta\cup\sigma) \in [\![\Gamma, BV \vdash Q_i[\vec{a}/\vec{q}]]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$$

which implies by the definition of interpretation that $\hat{\Delta}_m \vdash a_i(\theta\cup\sigma) \in \mathcal{SN}_{\hat{\Delta}_m}$ for all i. Similarly, $\hat{\nabla}_i \vdash u_i \in \mathcal{SN}_{\hat{\nabla}_i}$ for all i. This gives the desired conclusion by Lemma 5.1.14.

Product

$$\frac{(BV,\ A,\ p_1) \qquad (BV, x : A,\ B,\ p_2)}{(BV,\ (x\!:\!A).B,\ p_2)} \quad (x \notin dom(\Gamma, BV),\ p_1, p_2 \in \{\star, \Box\})$$

We have to show that $\hat{\Delta}_m \vdash ((x\!:\!A).B)(\theta\cup\sigma) \in [\![\Gamma, BV \vdash p_2]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$. Since the interpretation of p_2 is just $\mathcal{SN}_{\hat{\Delta}_m,p_2}$ by Lemma 5.1.15 it suffices that $\hat{\Delta}_m \vdash A(\theta\cup\sigma) \in \mathcal{SN}_{\hat{\Delta}_m}$ and $\hat{\Delta}_m, z : A(\theta\cup\sigma) \vdash B(\theta\cup\sigma\cup[z/x]) \in \mathcal{SN}_{\hat{\Delta}_m,z:A(\theta\cup\sigma)}$. But this follows from $\hat{\Delta}_m \vdash A(\theta\cup\sigma) \in [\![\Gamma, BV \vdash p_1]\!]_{\theta\cup\sigma,\zeta\cup\xi,\hat{\Delta}_m}$ and $\hat{\Delta}_m, z : A(\theta\cup\sigma) \vdash B(\theta\cup\sigma\cup[z/x]) \in [\![\Gamma, BV, x : A \vdash p_2]\!]_{\theta\cup\sigma\cup[z/x],\zeta\cup\xi\cup[C'/x],\hat{\Delta}_m,z:A(\theta\cup\sigma)}$, which are true for any $C' \in \mathcal{C}_{\hat{\Delta}_m,z}$ by induction hypothesis.

Reduction

$$\frac{(BV,\ t,\ T)}{(BV,\ t',\ T)} \qquad (t \to_\beta t')$$

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

Conversion

$$\frac{(BV,\ t,\ T) \qquad \Gamma, BV \vdash_{CC} T' : p \qquad \mathcal{T} \overset{*}{\leftrightarrow}_\beta T'}{(BV,\ t,\ T')} \qquad (p \in \{\star, \Box\})$$

The proof is immediate since by Lemma 5.3.6 the interpretations of convertible types are equal and β-equality on terms of the calculus of constructions is contained in convertibility. $\Box$

Lemma 5.4.8 (Recursive Comparison Lemma) *Let f be a function symbol of type $(p_1 : P_1)\dots(p_r : P_r)(x_1 : b_1)\dots(x_n : b_n).c$, Δ an environment, ρ a well-typed substitution from $\overrightarrow{p:P}$ to Δ and ζ_ρ a candidate assignment compatible with ρ. Let $\Delta_1,\dots\Delta_r,\nabla_1,\dots\nabla_n$ be environments satisfying $\Delta = \Delta_0 \subseteq \Delta_1 \subseteq \dots\Delta_r \subseteq \nabla_1 \subseteq \dots\nabla_n$.*

Moreover, let $\Gamma = \overrightarrow{p:P}, G$ be an environment, $\vec{u}$ Γ-terms satisfying $\Gamma \vdash_{CC} f(\vec{p},\vec{u}) : c[\vec{u}/\vec{x}]$, $\theta : \Gamma \to \nabla_n$ a well-typed substitution and ζ a candidate assignment compatible with θ such that for all i $\Delta_i \vdash p_i\rho \to^* p_i\theta$, $\zeta_\rho|_{\nabla_n}(p_i) = \zeta(p_i)$ and

$$\Delta_j \vdash p_j\theta \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\Delta_{j-1}} \quad \textit{for } j = 1,\dots r, \quad \textit{and}$$
$$\nabla_j \vdash u_j\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{j-1}} \quad \textit{for } j = 1,\dots n$$

Suppose also that for every $(g, \Delta', \rho', \zeta', (\overrightarrow{\Delta' \vdash \beta}, \overrightarrow{\nabla' \vdash \nu}))$ smaller than $(f, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla \vdash u\theta}))$ in $(>_{\mathcal{F}}, \emptyset, \emptyset, \emptyset, \leadsto^{\Delta}_{(f,\rho,\zeta_\rho)})_{lex}$ if $g : \overrightarrow{(q:Q)}\overrightarrow{(y:d)}.e \in \mathcal{F}^{(r',n')}$, $\Delta = \Delta'_0 \subseteq \Delta'_1 \dots \Delta'_{r'} \subseteq \nabla'_1 \dots \nabla'_{n'}$, $\Delta' \vdash q_i\rho' \to^{\Delta'}_{sn} \Delta'_i \vdash \beta_i$ for all i and

- $\Delta'_j \vdash \beta_j \in [\![\overrightarrow{q:Q} \vdash Q_j]\!]_{[\vec{\beta}/\vec{q}],\zeta',\Delta'_{j-1}}$ *for $j = 1,\dots r'$,*
- $\nabla'_j \vdash \nu_j \in [\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash d_j]\!]_{[\vec{\beta}/\vec{q}]\cup[\vec{\nu}/\vec{y}],\zeta',\nabla'_{j-1}}$ *for $j = 1,\dots n'$,*

then $\nabla_{n'} \vdash g(\vec{\beta},\vec{\nu}) \in [\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash e]\!]_{[\vec{\beta}/\vec{q}]\cup[\vec{\nu}/\vec{y}],\zeta',\nabla_{n'}}$

Under these assumptions for every t resulting from $\vec{l} = (\vec{p},\vec{u})$ in one of the following ways:

1. *$\Gamma \vdash l_k \succeq^{\mathcal{CS}} t$ for some l_k,*
2. *$\Gamma \vdash l_k \succeq^{\mathcal{RCS}} t$ for some l_k,*
3. *$t \in CCl_f(\vec{l})$,*

there is T such that:

- *$\Gamma \vdash_{CC} t : T$, and*
- *$\nabla_n \vdash t\theta \in [\![\Gamma \vdash T]\!]_{\theta,\zeta,\nabla_n}$.*

Proof

First note that $\Delta_j \vdash p_j\theta \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\Delta_{j-1}}$ implies $\nabla_n \vdash p_j\theta \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\nabla_n}$ and $\nabla_j \vdash u_j\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{j-1}}$ implies

$\nabla_n \vdash u_j\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_n}$ by (S3) condition of candidates and the properties of restriction. Moreover, $[\![\overrightarrow{p:P} \vdash P_j]\!]_{\theta,\zeta_\rho,\nabla_n} = [\![\Gamma \vdash P_j]\!]_{\theta,\zeta,\nabla_n}$ and $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_n} = [\![\Gamma \vdash b_j[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ by Lemma 5.3.4.

If $t = l_k$ for some k then obviously $\Gamma \vdash_{CC} t : P_k$ (or $\Gamma \vdash_{CC} t : b_{k-r}[u/x]$ if $k > r$), and $\nabla_n \vdash t\theta \in [\![\Gamma \vdash P_k]\!]_{\theta,\zeta,\nabla_n}$ (or $\nabla_n \vdash t\theta \in [\![\Gamma \vdash b_{k-r}]\!]_{\theta,\zeta,\nabla_n}$ if $k > r$).

If $\Gamma \vdash l_k \succ^{\mathcal{CS}} t$ then either $\Gamma \vdash l_k \succ t : A$ for some A or there is a term a such that $l_k \rhd^{\mathcal{CS}} a$ and $\Gamma \vdash a \succeq t : A$ for some A. In the former case we conclude by the definition of the family of candidates (condition (S1)) and the fact that HORPO compares only terms of equal types. In the latter case we first have to use Constructor Subterm Lemma 5.3.9 to get $\nabla_n \vdash a\theta \in [\![\Gamma \vdash A]\!]_{\theta,\zeta,\nabla_n}$ and then by the same condition (S1) $\nabla_n \vdash t\theta \in [\![\Gamma \vdash A]\!]_{\theta,\zeta,\nabla_n}$.

If $\Gamma \vdash l_k \succ^{\mathcal{RCS}} t$ then there is a recursive subterm r of l_k and a left-flattening $@(v_0, v_1, \ldots v_m)$ of t, such that $\Gamma \vdash r \succeq v_0 : \overrightarrow{(x:A)}.B$ and $v_i \in CCl_f(\vec{l})$ for all i. By Lemma 5.4.7 we have $\nabla_n \vdash v_i\theta \in [\![\Gamma \vdash A_i[\vec{v}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ for all i. We may use Lemma 5.4.6 (with $\Delta''' = \nabla_n$, $\Gamma''' = \emptyset$) to get $\nabla_n \vdash (r\, v_1 \, \ldots v_m)\theta \in [\![\Gamma \vdash B[\vec{v}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$. It is sufficient now to use already mentioned condition (S1) to get the desired conclusion $\nabla_n \vdash (v_0\, v_1 \, \ldots v_m)\theta \in [\![\Gamma \vdash B[\vec{v}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ and $\nabla_n \vdash t\theta \in [\![\Gamma \vdash B[\vec{v}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$.

If $t \in CCl_f(\vec{l})$, the desired conclusion follows directly from Computable Closure Lemma 5.4.7. $\square$

Proof of Lemma 5.4.5

Let us recall the statement of the lemma:

For every function symbol f of type $(p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).c$, environment Δ, well-typed substitution $\rho : \overrightarrow{p:P} \to \Delta$ and candidate assignment ζ_ρ compatible with ρ, if $\vec{\Delta}$, $\vec{\alpha}$, $\vec{\nabla}$, $\vec{\sigma}$ satisfy $\Delta = \Delta_0 \subseteq \Delta_1 \subseteq \ldots \Delta_r \subseteq \nabla_1 \subseteq \ldots \nabla_n$, $\Delta_i \vdash p_i\rho {\to^*} \alpha_i$ for all i and

- $\Delta_j \vdash \alpha_j \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{[\vec{\alpha}/\vec{p}],\zeta_\rho,\Delta_{j-1}}$, for $j = 1, \ldots r$,
- $\nabla_j \vdash \sigma_j \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_{j-1}}$, for $j = 1, \ldots n$,

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

The proof is done by nested induction. The first one is on $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash \alpha}, \overrightarrow{\nabla \vdash \sigma}))$ ordered by $(>_{\mathcal{F}}, \emptyset, \emptyset, \emptyset, \leadsto^{\Delta}_{(f,\rho,\zeta_\rho)})_{lex}$.

Given $\nabla_n \vdash f(\vec{\alpha}, \vec{\sigma})$ we will check whether all its reducts, i.e. all terms t such that $\nabla_n \vdash f(\vec{\alpha}, \vec{\sigma}) \to t$, belong to $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_n}$. The second induction is on the size of that t.

Accordingly to the position *pos*, where the rewriting took place, and the way the judgment, which served to rewrite, was accepted by HORPO we distinguish the following cases:

1. $pos \neq \Lambda$. We will use $f(\vec{d})$ for $f(\vec{\alpha}, \vec{\sigma})$. In this case $\nabla_n \vdash f(\vec{d}) \to f(\vec{t})$ and there is an i such that $\nabla_n \vdash d_i \to t_i$ and $t_j = d_j$ for $j \neq i$.

   If $i \leq r$ then by (S3) from the definition of candidates and the properties of restriction we have $\nabla_n \vdash \alpha_i \in [\![\overrightarrow{p : P} \vdash P_i]\!]_{[\vec{\alpha}/\vec{p}],\zeta_\rho,\Delta_{i-1}}$, $\nabla_n \vdash \alpha_j \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{[\vec{\alpha}/\vec{p}],\zeta_\rho,\nabla_n}$ for $j > i$ and $\nabla_n \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_n}$ for all j. By condition (S1) $\nabla_n \vdash t_i \in [\![\overrightarrow{p : P} \vdash P_i]\!]_{[\vec{\alpha}/\vec{p}],\zeta_\rho,\Delta_{i-1}}$. Let us denote by γ_i the substitution $[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}]$ with t_i assigned to the i-th variable. Since $[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}]$ and γ_i are convertible, Lemma 5.2.15 shows that

   $$\nabla_n \vdash \alpha_j \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\gamma_i,\zeta_\rho,\nabla_n} \text{ for } \; j > i$$
   $$\nabla_n \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{\gamma_i,\zeta_\rho,\nabla_n}$$

   By induction hypothesis $\nabla_n \vdash f(\vec{t}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{\gamma_i,\zeta_\rho,\nabla_n}$ and we conclude from Lemma 5.2.15 that $\nabla_n \vdash f(\vec{t}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_n}$. The use of induction hypothesis is correct since $(f, \Delta, \rho, \zeta_\rho, (\Delta_1 \vdash \alpha_1, \ldots \Delta_{i-1} \vdash \alpha_{i-1}, \nabla_n \vdash \alpha_i, \ldots \nabla_n \vdash \alpha_r, \overrightarrow{\nabla_n \vdash \sigma})) \in$ *DepDom* and it is smaller than $(f, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash \alpha}, \overrightarrow{\nabla \vdash \sigma}))$, as $\Delta_i \vdash \alpha_i \to_{sn}^{\Delta} \nabla_n \vdash t_i$.

   If $i > r$ the reasoning is very similar. Using (S1) and (S3) we get

   $$\Delta_j \vdash \alpha_j \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{[\vec{\alpha}/\vec{p}],\zeta_\rho,\Delta_{j-1}} \text{ for all } j$$
   $$\nabla_j \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_{j-1}} \text{ for } \; j < i - r$$
   $$\nabla_n \vdash t_i \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_{i-r}]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_{i-r-1}}$$
   $$\nabla_n \vdash \sigma_j \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_n} \text{ for } \; j > i - r$$

   Using substitution γ_i, Lemma 5.2.15 and induction hypothesis we arrive to show $\nabla_n \vdash f(\vec{t}) \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{\alpha}/\vec{p}]\cup[\vec{\sigma}/\vec{x}],\zeta_\rho,\nabla_n}$. The use of induction hypothesis is correct since $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash \alpha}, \nabla_1 \vdash \sigma_1, \ldots \nabla_{i-r-1} \vdash \sigma_{i-r-1}, \nabla_n \vdash t_i, \ldots \nabla_n \vdash \sigma_n)) \in$ *DepDom* and it is smaller than $(f, \rho, \zeta_\rho, (\Delta \vdash \alpha, \nabla \vdash \sigma))$, as $\nabla_{i-r} \vdash \sigma_{i-r} \to_{sn}^{\nabla_{i-r-1}} \nabla_n \vdash t_i$.

2. $pos = \Lambda$. Note that the second part of HORPO was already considered in the previous case. Consequently if $\Gamma \vdash l \succ r\ :\ T$ is the judgment which serves to rewrite then

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

where θ is a well-typed substitution from Γ to ∇_n. Note that $[\vec{\alpha}/\vec{p}] \cup [\vec{\sigma}/\vec{x}] = [\vec{u}/\vec{x}]\theta$.

Note that $\Delta \vdash p_j\theta \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta_\rho,\Delta}$ implies $\nabla_n \vdash p_j\theta \in [\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta_\rho,\nabla_n}$ and $\nabla_j \vdash u_j\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{j-1}}$ implies $\nabla_n \vdash u_j\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_n}$ by property (S3) of candidates and the properties of restriction. Moreover, by Lemma 5.3.4 $[\![\overrightarrow{p : P} \vdash P_j]\!]_{\theta,\zeta_\rho,\nabla_n} = [\![\Gamma \vdash P_j]\!]_{\theta,\zeta,\nabla_n}$, $[\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_j]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_n} = [\![\Gamma \vdash b_j[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ and $[\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_n} = [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ for any ζ extending ζ_ρ, for example

$$x\zeta = \begin{cases} \zeta_\rho|_{\nabla_n}(x) & x \in \vec{p} \\ can_{\nabla_n, x\theta} & \text{otherwise} \end{cases}$$

Summarizing we have:

- $\nabla_n \vdash p_j\theta \in [\![\Gamma \vdash P_j]\!]_{\theta,\zeta,\nabla_n}$ for all $j = 1, \ldots r$,
- $\nabla_n \vdash u_j\theta \in [\![\Gamma \vdash b_j[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ for all $j = 1, \ldots n$,

We will try to show that $\nabla_n \vdash r\theta \in [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$. We examine the cases of HORPO, by which the judgment $\Gamma \vdash l \succ r\ :\ T$ was accepted:

(a) $\Gamma \vdash l \succ r\ :\ T$ was accepted by case I.1 of HORPO.
Then either $r \in CCl_f(\vec{p}, \vec{u})$, or $\Gamma \vdash l_i \succeq^{\mathcal{CS}} r$ or $\Gamma \vdash l_i \succeq^{\mathcal{RCS}} r$ for some i. In any case, by Lemma 5.4.8, we have $\nabla_n \vdash r\theta \in [\![\Gamma \vdash T_r]\!]_{\theta,\zeta,\nabla_n}$ with $\Gamma \vdash_{CC} T_r \overset{*}{\leftrightarrow} c[\vec{u}/\vec{x}]$ and by Lemma 5.3.6 $\nabla_n \vdash r\theta \in [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$.

(b) $\Gamma \vdash l \succ r\ :\ T$ was accepted by case I.2 of HORPO. Then $f >_{\mathcal{F}} g$, $g : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'})(y_1 : d_1) \ldots (y_m : d_m).e$ and

$$\begin{array}{ll} l = f(\vec{l}) = f(\vec{p}, \vec{u}), & r = g(\vec{r}) = g(\vec{t}, \vec{w}) \\ l\theta = f(\vec{p}\theta, \vec{u}\theta) = f(\vec{\alpha}, \vec{\sigma}), & t = r\theta = g(\vec{t}\theta, \vec{w}\theta) \end{array}$$

We will show that every argument of g (i.e. $r_i\theta$) belongs to the interpretation of its type.

By the definition of HORPO we know that for every r_i one of the following possibilities holds: $\Gamma \vdash f(\vec{l}) \succ r_i \ : \ c[\vec{u}/\vec{x}]$ or $r_i \in CCl_f(\vec{l})$, or $\Gamma \vdash l_j \succeq^{\mathcal{CS}} r_i$ or $\Gamma \vdash l_j \succeq^{\mathcal{RCS}} r_i$ for some j.

If $\Gamma \vdash f(\vec{l}) \succ r_i \ : \ c[\vec{u}/\vec{x}]$ then $\nabla_n \vdash r_i\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash c]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_n}$ by second induction hypothesis, as $|r_i\theta|$ is smaller than $|t|$.

Otherwise, by Lemma 5.4.8, $\nabla_n \vdash r_i\theta \in [\![\Gamma \vdash T_i]\!]_{\theta,\zeta,\nabla_n}$ for some T_i such that $\Gamma \vdash_{CC} r_i : T_i$.

Summarizing we have:

- $\nabla_n \vdash t_j\theta \in [\![\Gamma \vdash Q_i[\vec{t}/\vec{q}]]\!]_{\theta,\zeta,\nabla_n}$, for all $j = 1, \dots r'$,
- $\nabla_n \vdash w_j\theta \in [\![\Gamma \vdash d_j[\vec{t}/\vec{q}][\vec{w}/\vec{y}]]\!]_{\theta,\zeta,\nabla_n}$, for all $j = 1, \dots m$,

which, by Lemma 5.3.3, is equivalent to:

- $\nabla_n \vdash t_j\theta \in [\![\overrightarrow{q:Q} \vdash Q_i]\!]_{[\vec{t}\theta/\vec{q}],\xi,\nabla_n}$, for all $j = 1, \dots r'$,
- $\nabla_n \vdash w_j\theta \in [\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash d_j]\!]_{[\vec{t}\theta/\vec{q}]\cup[\vec{w}\theta/\vec{y}],\xi,\nabla_n}$, for all $j = 1, \dots m$,

where $q_i\xi = [\![\Gamma \vdash t_i]\!]_{\theta,\zeta,\nabla_n}$ and $y_i\xi = [\![\Gamma \vdash w_i]\!]_{\theta,\zeta,\nabla_n}$.

Note that $(g, \nabla_n, [\vec{t}\theta/\vec{q}], \xi, (\overrightarrow{\nabla_n \vdash t\theta}, \overrightarrow{\nabla_n \vdash w\theta}))$ belongs to $DepDom$, as

i. $\nabla_n \vdash t_i\theta, \nabla_n \vdash w_i\theta \in \mathcal{SN}_{\nabla_n}$, because they belong to the corresponding interpretations and interpretations are subsets of strongly normalizing terms,

ii. if $d_i = s(\vec{\alpha'})\vec{t'}$ then $\nabla_n \vdash w_i\theta \in U_{(s,\nabla_n,\vec{A^i})}$, where

$$\vec{A^i} = \overrightarrow{(\nabla_n \vdash \alpha'[\vec{t}\theta/\vec{q}, \vec{w}\theta/\vec{y}], I)} \text{ with}$$
$$I_k = [\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash \alpha'_k]\!]_{[\vec{t}\theta/\vec{q},\vec{w}\theta/\vec{y}],\xi,\nabla_n}$$

because

$$[\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash s(\vec{\alpha'})\vec{t'}]\!]_{[\vec{t}\theta/\vec{q},\vec{w}\theta/\vec{y}],\xi,\nabla_n} = \mu F(\vec{A^i}, \overrightarrow{(\nabla_n \vdash t'[\vec{t}\theta/\vec{q}, \vec{w}\theta/\vec{y}], W)})$$

with $W_k = [\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash t'_k]\!]_{[\vec{t}\theta/\vec{q},\vec{w}\theta/\vec{y}],\xi,\nabla_n}$

Moreover $(g, \nabla_n, [\vec{t}\theta/\vec{q}], \xi, (\overrightarrow{\nabla_n \vdash t\theta}, \overrightarrow{\nabla_n \vdash w\theta}))$ is strictly smaller than $(f, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla \vdash u\theta}))$ because $f >_{\mathcal{F}} g$. Consequently, we can apply induction hypothesis and get $\nabla_n \vdash g(\vec{t}\theta, \vec{w}\theta) \in [\![\overrightarrow{q:Q}, \overrightarrow{y:d} \vdash e]\!]_{[\vec{t}\theta/\vec{q}]\cup[\vec{w}\theta/\vec{y}],\xi,\nabla_n}$. Using once again Lemma 5.3.3 we get $\nabla_n \vdash g(\vec{t}\theta, \vec{w}\theta) \in [\![\Gamma \vdash e[\vec{t}/\vec{q}][\vec{w}/\vec{y}]]\!]_{\theta,\zeta,\nabla_n}$. But $[\![\Gamma \vdash e[\vec{t}/\vec{q}][\vec{w}/\vec{y}]]\!]_{\theta,\zeta,\nabla_n} = [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ by Lemma 5.3.6 and because $\Gamma \vdash_{CC} e[\vec{t}/\vec{q}][\vec{w}/\vec{y}] \overset{*}{\leftrightarrow} c[\vec{u}/\vec{x}]$.

(c) $\Gamma \vdash l \succ r \ : T$ was accepted by case I.3 of HORPO. Then $r = f(\vec{p}, \vec{w})$, $f \in Mul$ and $t = r\theta$.

By the definition of HORPO and the definition of the multiset extension, we know that for every w_j either $\Gamma \vdash l_i \succ^{\mathcal{CS}} w_j$ or $l_i = w_j$ for some i.

If $l_i = w_j$ for some i then easily $\nabla_n \vdash w_j\theta \in [\![\Gamma \vdash b_j[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$. Otherwise, we get $\nabla_n \vdash w_j\theta \in [\![\Gamma \vdash b_j[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ by Lemma 5.4.8. By Lemma 5.3.4, for all $j = 1, \ldots n$ we have:

$$\nabla_n \vdash w_j\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_j]\!]_{[\vec{p}\theta/\vec{p}]\cup[\vec{w}\theta/\vec{x}],\zeta_\rho,\nabla_n}$$

By the hypothesis we have $\Delta_j \vdash p_j\theta \in [\![\overrightarrow{p:P} \vdash P_j]\!]_{[\vec{p}\theta/\vec{p}],\zeta_\rho,\Delta_{j-1}}$, for all $j = 1, \ldots r$.

In order to use the induction hypothesis, we need to know that $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla_n \vdash w\theta}))$ belongs to $DepDom$ and $\vec{u}\theta((\rightarrow^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})^+)_{mul}\vec{w}\theta$. The first requirement reduces to $\overrightarrow{\Delta \vdash p\theta}, \overrightarrow{\nabla_n \vdash w\theta} \in \mathcal{SN}_\Delta$ which is obviously true, since $\vec{p}\theta, \vec{t}\theta$ belong to interpretations of their types and interpretations are strongly normalizing sets. For the second one we have to show that $\nabla_i \vdash u_i\theta(\rightarrow^{\Delta}_{sn} \cup \rhd^{\mathcal{CS}})^+\nabla_n \vdash w_k\theta$ whenever $\Gamma \vdash u_i \succ^{\mathcal{CS}} w_k$. But this is true by Lemma 5.4.1.

Now, we can apply induction hypothesis and get $\nabla_n \vdash f(\vec{p}, \vec{w})\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash c]\!]_{[\vec{p}\theta/\vec{p}]\cup[\vec{w}\theta/\vec{x}],\zeta_\rho,\nabla_n}$. As $[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash c]\!]_{[\vec{p}\theta/\vec{p}]\cup[\vec{w}\theta/\vec{x}],\zeta_\rho,\nabla_n} = [\![\Gamma \vdash c[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ by Lemma 5.3.4 and $[\![\Gamma \vdash c[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n} = [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$, by Lemma 5.3.6 applied to $\Gamma \vdash_{CC} c[\vec{w}/\vec{x}] \overset{*}{\leftrightarrow} c[\vec{u}/\vec{x}]$, the proof of this case is complete.

(d) $\Gamma \vdash l \succ r \ : T$ was accepted by case I.4 of HORPO. Then $r = f(\vec{p}, \vec{w})$, $f \in Lex$ and $t = r\theta$.

By the definition of HORPO and the definition of the lexicographic extension, we know that there exists k such that $\Gamma \vdash$

$u_k \succ^{\mathcal{RCS}} w_k$ and $u_i = w_i$ for $i < k$. Moreover for $i > k$ one of the following cases holds $\Gamma \vdash f(\vec{l}) \succ r_i \ : c[\vec{u}/\vec{x}]$ or $r_i \in CCl_f(\vec{l})$, or $\Gamma \vdash l_j \succeq^{\mathcal{CS}} r_i$ or $\Gamma \vdash l_j \succeq^{\mathcal{RCS}} r_i$ for some j.
By the assumption of the lemma we have

$$\Delta_i \vdash p_i\theta \in [\![\overrightarrow{p:P} \vdash P_i]\!]_{\theta,\zeta_\rho,\Delta_{i-1}} \quad \text{for } i = 1, \ldots r$$

We have also

$$\nabla_i \vdash w_i\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_i]\!]_{[\vec{w}/\vec{x}]\theta,\zeta_\rho,\nabla_{i-1}}$$

for $i < k$. By Lemma 5.4.8 for $i \geq k$ we have

$$\nabla_n \vdash w_i\theta \in [\![\Gamma \vdash b_i[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$$

and consequently (by Lemma 5.3.4)

$$\nabla_n \vdash w_i\theta \in [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash b_i]\!]_{[\vec{p}\theta/p]\cup[\vec{w}\theta/\vec{x}],\zeta_\rho,\nabla_n}$$

Let us prove that $(f, \Delta, \rho, \zeta_\rho, (\overrightarrow{\Delta \vdash p\theta}, \nabla_1 \vdash w_1\theta, \ldots \nabla_{k-1} \vdash w_{k-1}\theta,$ $\nabla_n \vdash w_k\theta, \ldots \nabla_n \vdash w_n\theta)) \in DepDom$. We have

i. $\forall j \ \Delta_j \vdash p_j\theta \in \mathcal{SN}_\Delta$, $\forall i < k \ \nabla_i \vdash w_i\theta \in \mathcal{SN}_{\nabla_{i-1}}$, $\nabla_n \vdash w_k\theta \in \mathcal{SN}_{\nabla_{k-1}}$, $\forall i > k \ \nabla_n \vdash w_i\theta \in \mathcal{SN}_{\nabla_n}$, because these sequents belong to the corresponding interpretations and interpretations are subsets of strongly normalizing terms,

ii. if $b_i = s(\vec{\alpha'})\vec{t'}$ then $\Delta' \vdash w_i\theta \in U_{(s,\Delta'',\vec{A^i})}$, where $\Delta' = \nabla_i$, $\Delta'' = \nabla_{i-1}$ for $i < k$, $\Delta' = \nabla_n$, $\Delta'' = \nabla_{k-1}$ for $i = k$ and $\Delta' = \Delta'' = \nabla_n$ for $i > k$. and

$$\begin{aligned} \vec{A^i} &= \overrightarrow{(\Delta'' \vdash \alpha'([\vec{p}\theta/\vec{p}] \cup [\vec{w}\theta/\vec{x}]), I)} \text{ with} \\ I_l &= [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash \alpha'_l]\!]_{[\vec{p}\theta/\vec{p}]\cup[\vec{w}\theta/\vec{x}],\zeta_\rho,\Delta''} \end{aligned}$$

because

$$\begin{aligned} &[\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash s(\vec{\alpha'})\vec{t'}]\!]_{[\vec{p}\theta/p]\cup[\vec{w}\theta/x],\zeta_\rho,\Delta''} = \\ &\qquad\qquad \mu F(\vec{A^i}, \overrightarrow{(\Delta'' \vdash t'[\vec{p}\theta/\vec{p}] \cup [\vec{w}\theta/\vec{x}], W)}) \end{aligned}$$

with $W_l = [\![\overrightarrow{p:P}, \overrightarrow{x:b} \vdash t'_l]\!]_{[\vec{p}\theta/\vec{p}]\cup[\vec{w}\theta/\vec{x}],\zeta_\rho,\Delta''}$.

To use the induction hypothesis, we need to know that $\overrightarrow{\nabla \vdash u\theta}(\rightsquigarrow_1, \ldots \rightsquigarrow_{r+n})_{lex}(\nabla_1 \vdash w_1\theta, \ldots \nabla_{k-1} \vdash w_{k-1}\theta, \nabla_n \vdash w_k\theta, \ldots \nabla_n \vdash w_n\theta)$. We will show that $\nabla_k \vdash u_k\theta(\rightarrow_{sn}^{\nabla_{k-1}} \cup >_{deg(\vec{A}^k)}^{\nabla_{k-1}})\nabla_n \vdash w_k\theta$.

If $\Gamma \vdash u_k \succ^{\mathcal{RCS}} w_k$ holds because $\Gamma \vdash u_k \succ w_k \ : b_k[\vec{w}/\vec{x}]$ then obviously $\nabla_n \vdash u_k\theta \rightarrow w_k\theta$ and $\nabla_k \vdash u_k\theta \rightarrow_{sn}^{\nabla_{k-1}} \nabla_n \vdash w_k\theta$.

Otherwise, $b_k = s(\vec{\alpha'})\vec{t'}$, there is a recursive subterm of u_k, a, and a left-flattening of w_k, $@(v_0, v_1, \ldots v_m)$, such that $\Gamma \vdash a \succeq v_0 \ : \overrightarrow{(z : T)}.B$ for some $\vec{T}, B$ and the tuples $(\emptyset,\ v_i,\ T_i[\vec{v}/\vec{z}])$, $i = 1 \ldots m$, are in the computable closure $CCl_f(\vec{l})$. From Lemma 5.4.7 it follows that for all i $\nabla_n \vdash v_i \in [\![\Gamma \vdash T_i[\vec{v}/\vec{z}]]\!]_{\theta,\zeta,\nabla_n}$. By assumption we have $\nabla_k \vdash u_k\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash b_k]\!]_{[\vec{u}/\vec{x}]\theta,\zeta_\rho,\nabla_{k-1}}$. We may apply now Lemma 5.4.6 (with $\Delta''' = \nabla_n$, $\Gamma''' = \emptyset$) and get $\nabla_k \vdash u_k\theta >_{deg(\vec{A})}^{\nabla_{k-1}} \nabla_n \vdash (a\, v_1 \ \ldots v_m)\theta$ for $\vec{A} = \overrightarrow{(\nabla_{k-1} \vdash \alpha'[\vec{u}/\vec{x}]\theta, I)}$ with $I_i = [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash \alpha'_i]\!]_{[\vec{u}/\vec{x}]\theta,\zeta,\nabla_{k-1}}$. Since $\alpha'_i[\vec{u}/\vec{x}]\theta = \alpha'_i[\vec{p}\theta/\vec{p}, \vec{w}\theta/\vec{x}]$ (as $u_j = w_j$ for all $j < k$) we have $\vec{A} = \vec{A}^k$. By Lemma 5.2.22 we conclude $\nabla_k \vdash u_k\theta >_{deg(\vec{A}^k)}^{\nabla_{k-1}} \Delta_m \vdash (v_0\, v_1 \ \ldots v_m)\theta$, which is equivalent to $\nabla_k \vdash u_k\theta >_{deg(\vec{A}^k)}^{\nabla_{k-1}} \Delta_m \vdash w_k\theta$. That completes the decreasingness argument.

Now we can apply induction hypothesis and get $\nabla_n \vdash f(\vec{p}, \vec{w})\theta \in [\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{p}\theta/\vec{p}] \cup [\vec{w}\theta/\vec{x}],\zeta_\rho,\nabla_n}$. As $[\![\overrightarrow{p : P}, \overrightarrow{x : b} \vdash c]\!]_{[\vec{p}\theta/\vec{p}] \cup [\vec{w}\theta/\vec{x}],\zeta_\rho,\nabla_n} = [\![\Gamma \vdash c[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ by Lemma 5.3.4 and $[\![\Gamma \vdash c[\vec{w}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n} = [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$, by Lemma 5.3.6 applied to $\Gamma \vdash_{CC} c[\vec{w}/\vec{x}] \overset{*}{\leftrightarrow} c[\vec{u}/\vec{x}]$, the proof of this case is complete.

(e) $\Gamma \vdash l \succ r \ : T$ was accepted by case I.5 of HORPO. Then $r = c'(\vec{t}, \vec{r})$, where c' is a constructor of an inductive type s, $t = r\theta$, and for every $d \in (\vec{t}, \vec{r})$ we have $\Gamma \vdash f(\vec{l}) \succ d \ : T$ or $\Gamma \vdash l_j \succeq^{\mathcal{CS}} d$ for some l_j or $\Gamma \vdash l_j \succeq^{\mathcal{RCS}} d$ for some l_j or $d \in CCl_f(\vec{l})$.

As in case (b), by the second induction hypothesis and Lemma 5.4.8 we can deduce that

$$\nabla_n \vdash t_i\theta \in [\![\Gamma \vdash Q_i[\vec{t}/\vec{q}]]\!]_{\theta,\zeta,\nabla_n}$$

$$\nabla_n \vdash r_i\theta \in [\![\Gamma \vdash d_i[\vec{t}/\vec{q}][\vec{w}/\vec{z}]]\!]_{\theta,\zeta,\nabla_n}$$

provided that $s : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'})(x'_1 : b'_1) \ldots (x'_{n'} : b'_{n'}).\star$ and $c' : (q_1 : Q_1) \ldots (q_{r'} : Q_{r'})(z_1 : d_1) \ldots (z_k : d_k).s(\vec{q})w_1 \ldots w_{n'}$.

By Lemma 5.3.2 we get $[\![\Gamma \vdash d_i[\vec{t}/\vec{q}][\vec{w}/\vec{z}]]\!]_{\theta,\zeta,\nabla_n} = [\![\Gamma, \overrightarrow{z : d[\vec{t}/\vec{q}]} \vdash d_i[\vec{t}/\vec{q}]]\!]_{(\theta \cup [\vec{r}\theta/\vec{z}]),\zeta,\nabla_n}$.

Consequently we have

$$\nabla_n \vdash t_i\theta \in [\![\Gamma \vdash Q_i[\vec{t}/\vec{q}]]\!]_{\theta,\zeta,\nabla_n}$$
$$\nabla_n \vdash r_i\theta \in [\![\Gamma, \overrightarrow{z : d[\vec{t}/\vec{q}]} \vdash d_i[\vec{t}/\vec{q}]]\!]_{(\theta\cup[\vec{r}\theta/\vec{z}]),\zeta,\nabla_n}$$

which is enough to apply the **composition** case of Lemma 5.3.7 and get $\nabla_n \vdash c'(\vec{t},\vec{r})\theta \in [\![\Gamma \vdash (s(\vec{q})\vec{w})[\vec{t}/\vec{q},\vec{r}/\vec{z}]]\!]_{\theta,\zeta,\nabla_n} = [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$.

(f) $\Gamma \vdash l \succ r : T$ was accepted by case I.6 of HORPO. 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, we have either $\Gamma \vdash f(\vec{l}) \succ r_i : T_i$, or $\Gamma \vdash l_j \succeq^{\mathcal{CS}} r_i : t_i$ or $\Gamma \vdash l_j \succeq^{\mathcal{RCS}} r_i$ or $r_i \in CCl_f(\vec{l})$. In any case, it follows that $\nabla_n \vdash r_0\theta \in [\![\Gamma \vdash \overrightarrow{(y : A)}.B]\!]_{\theta,\zeta,\nabla_n}$ and $\nabla_n \vdash r_i\theta \in [\![\Gamma \vdash A_i[\vec{r}/\vec{y}]]\!]_{\theta,\zeta,\nabla_n}$ (from the second induction hypothesis or Lemma 5.4.8).

By Lemma 5.3.2 $[\![\Gamma \vdash A_i[\vec{r}/\vec{y}]]\!]_{\theta,\zeta,\nabla_n} = [\![\Gamma, \overrightarrow{y : A} \vdash A_i]\!]_{\theta\cup[\vec{r}\theta/\vec{y}],\zeta\cup\xi,\nabla_n}$, with $y\xi = [\![\Gamma \vdash r_i]\!]_{\theta,\zeta,\nabla_n}$ for $y \in \mathcal{V}ar^{\Box}$.

By the definition of interpretation of the product type $\nabla_n \vdash @(r_0, r_1, \ldots r_m)\theta \in [\![\Gamma, \overrightarrow{y : A} \vdash B]\!]_{\theta\cup[\vec{r}\theta/\vec{y}],\zeta\cup\xi,\nabla_n}$. We now apply the same Lemma 5.3.2 and get $\nabla_n \vdash @(r_0, r_1, \ldots r_m)\theta \in [\![\Gamma \vdash B[\vec{r}/\vec{y}]]\!]_{\theta,\zeta,\nabla_n}$. But $\Gamma \vdash_{CC} @(r_0, r_1, \ldots r_m) : c[\vec{u}/\vec{x}]$ and $\Gamma \vdash_{CC} c[\vec{u}/\vec{x}] \overset{*}{\leftrightarrow} B[\vec{r}/\vec{y}]$ and we get the desired conclusion $\nabla_n \vdash @(r_0, r_1, \ldots r_m)\theta \in [\![\Gamma \vdash c[\vec{u}/\vec{x}]]\!]_{\theta,\zeta,\nabla_n}$ by Lemma 5.3.6. $\Box$

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)

$$\frac{\Gamma \vdash \star : \Box \qquad \forall i \quad \Gamma \vdash a_i : b_i[\vec{a}/\vec{x}]}{\Gamma \vdash e(\vec{a}) : c[\vec{a}/\vec{x}]} \qquad (e : (x_1 : b_1) \ldots (x_n : b_n).c \in \Sigma_n)$$

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

type constructor Let us start by specializing the typing rule for a type constructor $s : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).\star$.

$$\frac{\Gamma \vdash \star : \Box \qquad \forall i \quad \Gamma \vdash a_i : P_i[\vec{a}/\vec{p}]}{\Gamma \vdash s(a_1, \ldots a_r) : ((x_1 : b_1) \ldots (x_n : b_n).\star)[\vec{a}/\vec{p}]}$$

We have to prove that $\Delta \vdash s(\vec{a}\rho) \in [\![\Gamma \vdash ((x_1 : b_1) \ldots (x_n : b_n).\star)[\vec{a}/\vec{p}]]\!]_{\rho,\xi,\Delta}$ 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 $(\Delta^1 \vdash d_1, C_1), \ldots (\Delta^n \vdash d_n, C_n)$, if $\Delta = \Delta^0 \subseteq \Delta^1 \ldots \subseteq \Delta^n$, $\Delta^j \vdash d_j \in [\![\Gamma, \overrightarrow{x : b} \vdash b_j[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{d}/\vec{x}], (\xi \cup [\vec{C}/\vec{x}])|_{\Delta^j}, \Delta^j}$ and $C_j = \emptyset$ if $d_j \in Obj$ and $C_j \in \mathcal{C}_{\Delta^j, d_j}$ otherwise, then we have $\Delta^i \vdash s(\vec{a}\rho)d_1, \ldots d_i \in [\![\Gamma, \overrightarrow{x : b} \vdash (x_{i+1} : b_{i+1}[\vec{a}/\vec{p}]) \ldots (x_n : b_n[\vec{a}/\vec{p}]).\star]\!]_{\rho \cup [\vec{d}/\vec{x}], (\xi \cup [\vec{C}/\vec{x}])|_{\Delta^i}, \Delta^i}$ for every $i = 1, \ldots n$.

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 $\overrightarrow{\Delta \vdash a\rho}$ and $\Delta^1 \vdash d_1, \ldots \Delta^n \vdash d_n$ are strongly normalizing (see Lemma 5.1.14).

Now, let us prove the claim for $i - 1$, supposing that it holds for $i > 0$. Let us denote by T the type $(x_{i+1} : b_{i+1}[\vec{a}/\vec{p}]) \ldots (x_n : b_n[\vec{a}/\vec{p}]).\star$ and suppose that b_i is a kind (if b_i is a type the proof is similar). We have to show that $\Delta^{i-1} \vdash s(\vec{a\rho})d_1, \ldots d_{i-1} \in [\![\Gamma, \overrightarrow{x : b} \vdash (x_i : b_i[\vec{a}/\vec{p}]).T]\!]_{\rho \cup [\vec{d}/\vec{x}], (\xi \cup [\vec{C}/\vec{x}])|_{\Delta^{i-1}}, \Delta^{i-1}}$. By the definition of interpretation, $[\![\Gamma, \overrightarrow{x : b} \vdash (x_i : b_i[\vec{a}/\vec{p}]).T]\!]_{\rho \cup [\vec{d}/\vec{x}], (\xi \cup [\vec{C}/\vec{x}])|_{\Delta^{i-1}}, \Delta^{i-1}}$ equals to the set $\{\Delta' \vdash M \mid \Delta' \vdash M : ((x_i : b_i[\vec{a}/\vec{p}]).T)\rho \cup [\vec{d}/\vec{x}],\ \Delta' \supseteq \Delta^{i-1}$, and $\forall \Delta'' \supseteq \Delta'$, $\forall \Delta'' \vdash A \in [\![\Gamma, \overrightarrow{x : b} \vdash b_i[\vec{a}/\vec{p}]]\!]_{\rho \cup [\vec{d}/\vec{x}], (\xi \cup [\vec{C}/\vec{x}])|_{\Delta''}, \Delta''}$, $\forall D \in \mathcal{C}_{\Delta'', A}$, $\Delta'' \vdash M\,A \in [\![\Gamma, \overrightarrow{x : b} \vdash T]\!]_{\rho \cup [\vec{d}/\vec{x}] \cup [A/x], (\xi \cup [\vec{C}/\vec{x}] \cup [D/x])|_{\Delta''}, \Delta''}\}$. A direct application of the induction hypothesis shows that $s(\vec{a\rho})d_1, \ldots d_{i-1}$ belongs to the set described above.

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

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

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 5.3.7.

function symbol 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:

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

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 Fun Lemma (with $\Delta_i=\nabla_i=\Delta$, $\alpha_i=a_i\rho$, $\sigma_i=t_i\rho$ and $\zeta_\rho=[\overrightarrow{[\![\Gamma\vdash a]\!]_{\rho,\xi,\Delta}}/\vec{p}])$ 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)

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

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 $\forall \Delta'' \supseteq \Delta$, $\forall \Delta'' \vdash N \in [\![\Gamma \vdash a]\!]_{\rho,\xi|_{\Delta''},\Delta''}$, $\forall C \in \mathcal{C}_{\Delta'',N}$, $\Delta'' \vdash (\lambda x : a\rho . b\rho)\, N \in [\![\Gamma, x : a \vdash c]\!]_{\rho \cup [N/x], \xi|_{\Delta''} \cup [C/x], \Delta''}$.

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:

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

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 5.1.17, because $\Delta'' \vdash N \in \mathcal{SN}_{\Delta''}$ by assumption and $\Delta'' \vdash a\rho \in \mathcal{SN}_{\Delta''}$ by Lemma 5.1.16 applied to $\Delta'' \vdash (x : a).c\rho \in \mathcal{SN}_{\Delta''}$ which follows from induction hypothesis for $(x : a).c$.

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 $y \in \mathit{dom}(\Gamma, x : a)$. 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:

- $y(\rho \cup [N/x]) = y\rho$ and $(\Gamma, x : a)(y) = \Gamma(y)$,
- $\Delta'' \vdash y\rho \in [\![\Gamma \vdash \Gamma(y)]\!]_{\rho,\xi|_{\Delta''},\Delta''}$ 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)

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

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 $q = \Box$. Trivially $\Delta \vdash (x : a\rho).b\rho : q$. By induction hypothesis we have $\Delta \vdash a\rho \in \mathcal{SN}_\Delta$ and $\Delta, z : a\rho \vdash b\rho \in \mathcal{SN}_{\Delta, z : a\rho}$. By Lemma 5.1.15 this gives $\Delta \vdash (x : a\rho).b\rho \in \mathcal{SN}_\Delta$.

(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 5.3.6 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}:

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

where canonical candidates are defined in Lemma 5.1.12

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. $\Box$

Chapter 6

On the role of star dependency

This chapter provides an explanation for imposing star dependency (see Def. 2.3.1) on function and constructor symbols, and parametricity (see Def. 3.1.1) on HORPO judgments.

6.1 Star dependency, parametricity and HORPO

Left-hand sides of HORPO judgments and rewrite rules are assumed to satisfy two conditions: star dependency and parametricity. 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 i-th 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 the 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 parametricity 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 cannot 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 non-termination, 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.

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

So, we have found a non-terminating reduction sequence.

The example of the function symbol J and its rule derives from the one presented by J.-Y. Girard in [46]; 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 non-termination. Of course we do not say that every rule not satisfying the star dependency condition leads to non-termination. The question of knowing which of them are terminating is an interesting topic of future research.

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

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

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

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

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

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

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

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

$$f(Nat,\ a,\ n) \rightarrow n$$

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 $J(X,\ X,\ a,\ b) \rightarrow b$).

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

6.2 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:

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

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 of deducing that n belongs to the interpretation of its type from the hypothesis that $s(n)$ does. 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 $s : (p_1 : P_1) \ldots (p_r : P_r)(x_1 : b_1) \ldots (x_n : b_n).\star$ and $c : (p_1 : P_1) \ldots (p_r : P_r)(z_1 : d_1), \ldots (z_k : d_k).s(\vec{p})w_1 \ldots w_n$, 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 $z_i \in \mathcal{V}ar^{\Box}$ 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 [92] Benjamin Werner defines the interpretation of inductive types so as to have small elimination for free. For example, the interpretation of natural numbers states that for every suitable arguments $nat_{dep}(P, z, f, s(n))$ belongs to it.

The definition of the interpretation of inductive types, presented by Milena Stefanova in [86], 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. Fortunately, this is sufficient for small elimination rules.

In both papers, [92, 86], there is no additional restriction (apart from positivity) on types of constructors in order to define the interpretation of inductive types. Reaching under a constructor on arbitrary position is not possible, but the authors do not need it, because eliminations are the only rewrite rules in their systems.

On the other hand, it turns out that in both papers [92, 86] large elimination requires an additional condition on types of constructors. In [92] constructors must be *small*, i.e. every argument of a constructor must be either a parameter or an object, and in [86] 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 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 — i.e., as the right-hand side of a rewrite rule — because we consider only object-level rewriting).

Another paper concerned with strong normalization of elimination rules for inductive types is [17], where Frédéric Blanqui studies rewriting in the calculus of constructions. In this paper a condition similar to safeness of Stefanova is used to define interpretation of inductive types in such a way that reaching under a constructor is safe from the reducibility point of view. This condition is needed for both small and large eliminations. The role of safeness in [17] the same as the role of star dependency in our work: to know exactly the candidates to which belong immediate subterms of a constructor headed term.

An interesting question is whether a Stefanova-style or Werner-style definitions can be used to define interpretation for inductive types while safely reaching under a constructor only on some subset of positions. The recent result of Blanqui [15] shows that this is possible for Werner-style interpretations on the set of accessible positions. His accessible positions are exactly those whose type do not depend on non-parameter big arguments.

To conclude, we think that reaching under a constructor is something between small and large eliminations and that star dependency (and more generally, non-dependence on big non-parameter arguments, in case we are interested in reaching under a constructor only on some positions) is the necessary condition which makes it possible.

Chapter 7

Calculus of constructions with user-defined rewrite rules

In Chapter 5 we 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.

In the next section we will formally define such a calculus and justify its strong normalization. In Section 7.2 we present examples of interesting rewrite systems accepted (or not) by HORPO and in Section 7.3 we discuss which elimination rules of inductive types may be proved terminating by our method.

7.1 CC+R — Calculus of constructions with rewriting induced by user-defined rules

In this section, given a set of user-defined rewrite rules, R, we build the calculus of constructions with conversion based on the rewrite relation $\rightarrow_R$, called CC+R. We show that CC+R is terminating if all rules in R are accepted by HORPO.

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 2.3.3 (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 7.1.1 (Rule and term rewriting system) *A rewrite rule is a quadruple (G, l, r, T) satisfying parametricity condition 3.1.1.*

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

Definition 7.1.2 (CC+R) *Given a term rewriting system R, the system CC+R is defined just like CC+H, with two exceptions. In* **(rew)** *we use rules of R instead of HORPO judgments, and consequently in* **(conv)** *we replace $\to_H$ by $\to_R$.*

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

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

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 7.1.3 *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 $\Gamma \vdash l \succ r : T$ then the following statements hold:*

1. *every term of CC+R is a term of CC+H,*
2. $\to^\Gamma_{R\cup\beta} \subseteq \to^\Gamma$,
3. $\overset{*}{\leftrightarrow}{}^\Gamma_{R\cup\beta} \subseteq \overset{*}{\leftrightarrow}{}^\Gamma$,
4. *the strong normalization of CC+H implies the strong normalization of CC+R.*

Proof
Obvious. $\Box$

7.2 Examples

Before we give some examples of dependently typed rules, let us mention the fact that all examples accepted in [53] are also accepted by our HORPO.

In the examples below, for the sake of clarity we omit parentheses for type constructors (writing *List A* for *List*(A)). We also omit types in the 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 $nat_{dep}(P, z, f, n) \in CCl_{nat_{dep}}(P, z, f, s(n))$.

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

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

are in $CCl_{nat_{dep}}(P, z, f, s(n))$. By the definition of the accessible set,

$$(\emptyset,\ n,\ Nat) \in CCl_{nat_{dep}}(P, z, f, s(n))$$

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

$$(\emptyset,\ nat_{dep}(P, z, f, n),\ P\ n) \in CCl_{nat_{dep}}(P, z, f, s(n))$$

The use of this rule is justified, because $\Gamma \vdash (P, z, f, s(n)) \succ^{\mathcal{CS}}_{mul} (P, z, f, n)$ (as $\Gamma \vdash s(n) \succ^{\mathcal{CS}} n$).

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:*

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

and let us consider the rules:

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

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 first two inequalities are trivial.

For the third one we need to use the constructor part of $\succeq^{\mathcal{CS}}$, since $cons(A,\ a,\ l)$ and a have different types. Note that it is the only possible 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, because $(P, f_1, f_2, cons(A,\ a,\ l)) \succ^{\mathcal{CS}}_{mul} (P, f_1, f_2, l)$). 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:*

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

and suppose that both fold_left and fold_right $: (A : \star)(B : \star)(A \to B \to B) \to (n : Nat)(LList\ A\ n) \to B \to B \in \mathcal{F}^{(2,4)}$ *have a lexicographic status.*

Let Γ *be the environment*

$$A : \star,\ B : \star,\ f : A \to B \to B,\ a : A,\ n : Nat,\ l : (LList\ A\ n),\ x : B$$

and let us consider the rules:

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

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

The first rules of respectively *fold_left* and *fold_right* are trivially accepted by case I.1 of HORPO.

For the second rule of *fold_left* we use the case lex (I.4) of HORPO and we are led to show $\Gamma \vdash s(n) \succeq^{\mathcal{CS}} n$, $\Gamma \vdash \mathit{Lcons}(A,\ s(n),\ a,\ l) \succeq^{\mathcal{CS}} l$, which are obvious and $\Gamma \vdash \mathit{fold\_left}(A, B, f, s(n), \mathit{Lcons}(A, s(n), a, l), x) \succ @(f, a, x) \ : \ B$ that can be done by case I.6.

The second rule of *fold_right* can 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 a proof that a is in l or a 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 $A \rightarrow False$.*

$$\begin{array}{l} \mathit{Ind}[A:\star, B:\star](Or:\star \ := \ left : A \rightarrow \overline{Or},\ right : B \rightarrow \overline{Or}) \\ \mathit{Ind}[A:\star, x:A](Eq : A \rightarrow \star \ := \ refl\_eq : \overline{Eq}\ x) \\ \mathit{Ind}[\,](False:\star \ := \,) \end{array}$$

$$\begin{array}{l} \mathit{Ind}[A:\star, a:A](In : (\mathit{List}\ A) \rightarrow \star \ := \\ \qquad\qquad \mathit{in\_hd} : (l : (\mathit{List}\ A))(\overline{In}\ \mathit{cons}(A,\ a,\ l)), \\ \qquad\qquad \mathit{in\_tl} : (l : (\mathit{List}\ A))(b : A)(\overline{In}\ l) \rightarrow (\overline{In}\ \mathit{cons}(A,\ b,\ l)) \end{array}$$

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

Let eq_ind, mem and mem_cons have the following types:

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

Let

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

be the environment common to all the following rules:

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. $\Gamma\vdash mem(A,\ a,\ eqDec,\ cons(A,\ b,\ l))$
   $\to\ mem\_cons(A,\ a,\ b,\ l,\ @(eqDec,a,b),\ mem(A,\ a,\ eqDec,\ l)):$
   $Or\ \Phi_2\ \Upsilon_2$

4. $\Gamma\vdash mem\_cons(A,\ a,\ b,\ l,\ left(\Psi,\ \sim\Psi,\ d_l),\ i)$
   $\to left(\Phi_2,\ \Upsilon_2,\ eq_{ind}(A,\ a,\ (\lambda x{:}A.In\ A\ a\ cons(A,\ x,\ l)),\ b,\ d_l,$
   $in\_hd(A,\ a,\ l))):\ Or\ \Phi_2\ \Upsilon_2$

5. $\Gamma\vdash mem\_cons(A,\ a,\ b,\ l,\ right(\Psi,\ \sim\Psi,\ d_r),\ left(\Phi_3,\ \Upsilon_3,\ e_l))$
   $\to left(\Phi_2,\ \Upsilon_2,\ in\_tl(A,\ a,\ l,\ b,\ e_l)):\ Or\ \Phi_2\ \Upsilon_2$

6. $\Gamma\vdash mem\_cons(A,\ a,\ b,\ l,\ right(\Psi,\ \sim\Psi,\ d_r),\ right(\Phi_3,\ \Upsilon_3,\ e_r))$
   $\to right(\Phi_2,\ \Upsilon_2,\ allS\_cons(A,\ (\lambda x{:}A.\sim Eq\ A\ a\ x),\ b,\ l,\ d_r,\ e_r)):$
   $Or\ \Phi_2\ \Upsilon_2,$

where

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

We choose the precedence $mem >_{\mathcal{F}} mem\_cons >_{\mathcal{F}} eq_{ind}$ and the multiset status for mem. Let us consider 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 $CCl_{mem}(A, a, eqDec, nil(A))$. 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 **Type constructor** to A, a and $nil(A)$. 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 **Type constructor** we 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 **Type constructor** and then use **Product**. 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** to get $\Upsilon_1 \in CCl_{mem}(\vec{t})$ and **Constructor** to get $allS\_nil(A, (\lambda x{:}A. \sim Eq\ A\ a\ x)) \in CCl_{mem}(\vec{t})$.

For the third rule, since $mem >_{\mathcal{F}} mem\_cons$, we are left to show that every immediate subterm of mem_cons 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 initial and accessible sets, terms A, a, $eqDec$, b and l belong to $CCl_{mem}(A, a, eqDec, cons(A, b, l))$. By **Application**, $@(eqDec, a, b) \in CCl_{mem}(A, a, eqDec, cons(A, b, l))$ and by **Recursive call** $mem(A,\ a,\ eqDec,\ l) \in CCl_{mem}(A, a, eqDec, cons(A, b, l))$. The application of **Recursive call** 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 *mem_cons*. 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 eq_{ind}, it is obvious that $CCl_{mem\_cons}(A, a, b, l, left(\Psi, \sim \Psi, d_l), i)$ contains A, a, b, l, d_l as they are either immediate or constructor subterms of the left-hand side. We now 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 **Constructor**, for the former we first use **Constructor** yielding $(cons\ A\ x\ l)$ (with a free variable x), then **Type constructor** to obtain $(In\ A\ a\ cons(A,\ x,\ l)) \in CCl_{mem\_cons}(A, a, b, l, left(\Psi, \sim \Psi, d_l), i)$ and finally **Abstraction** over x to get $(\lambda x : A.In\ A\ a\ cons(A,\ x,\ l)) \in CCl_{mem\_cons}(A, a, b, l, left(\Psi, \sim \Psi, d_l), i)$. 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.

Example 14 (Elimination rules for Brouwer's ordinals) *Let us define an inductive type Ord as:*

$$\begin{aligned} Ind[](Ord : \star\ :=\ & zero : \overline{Ord} \\ & succ : \overline{Ord} \to \overline{Ord} \\ & lim : (Nat \to \overline{Ord}) \to \overline{Ord}) \end{aligned}$$

and suppose that $ord_{rec} : (P : \star)P \to (Ord \to P \to P) \to ((Nat \to Ord) \to (Nat \to P) \to P) \to Ord \to P \in \mathcal{F}^{(1,4)}$ *has a lexicographic status.*

Let us denote by Γ *the environment* $P : \star, f_1 : P, f_2 : Ord \to P \to P, f_3 : ((Nat \to Ord) \to (Nat \to P) \to P), n : Nat, F : Nat \to Ord$.

Consider the rules:

$$\begin{aligned} &\Gamma \vdash ord_{rec}(P, f_1, f_2, f_3, zero) \to f_1\ : P \\ &\Gamma \vdash ord_{rec}(P, f_1, f_2, f_3, succ(n)) \to @(f_2, n, ord_{rec}(P, f_1, f_2, f_3, n))\ : P \\ &\Gamma \vdash ord_{rec}(P, f_1, f_2, f_3, lim(F)) \to \\ &\qquad\qquad @(f_3, F, \lambda y{:}Nat.\, ord_{rec}(P, f_1, f_2, f_3, @(F, y))) : P \end{aligned}$$

The first two rules recall the rules for nat_{dep} and are easily accepted by HORPO. For the third one, there is a recursive call of ord_{rec} on $@(F, y)$ where y is a variable bound above ord_{rec}. Let us show that this rule is also accepted by HORPO.

By applying case I.6 of HORPO we are left with $\Gamma \vdash f_3 \succeq^{\mathcal{CS}} f_3$, $\Gamma \vdash lim(F) \succeq^{\mathcal{CS}} F$ and $\lambda y\!:\!Nat.\, ord_{rec}(P, f_1, f_2, f_3, @(F, y)) \in CCl_{ord_{rec}}(P, f_1, f_2, f_3, lim(F))$.

The first two inequalities are trivial. Let us detail the proof for the computable closure judgment. By the definition of the initial set the tuples

$$(\emptyset,\ P,\ \star)$$
$$(\emptyset,\ f_1,\ P)$$
$$(\emptyset,\ f_2,\ Ord \to P \to P)$$
$$(\emptyset,\ f_3,\ ((Nat \to Ord) \to (Nat \to P) \to P))$$
$$(\emptyset,\ lim(F),\ Ord)$$

are in $CCl_{ord_{rec}}(P, f_1, f_2, f_3, lim(F))$. We have also $(\emptyset,\ F,\ Nat \to Ord) \in CCl_{ord_{rec}}(P, f_1, f_2, f_3, lim(F))$ by the definition of the accessible set and $(y : Nat,\ y,\ Nat) \in CCl_{ord_{rec}}(P, f_1, f_2, f_3, lim(F))$ by **Var**. Consequently by **Application** one gets

$$(y : Nat,\ @(F, y),\ Ord) \in CCl_{ord_{rec}}(P, f_1, f_2, f_3, lim(F))$$

After weakening initial tuples to $(y : Nat,\ P,\ \star)$, $(y : Nat,\ f_1,\ P)$, $(y : Nat,\ f_2,\ Ord \to P \to P)$, $(y : Nat,\ f_3,\ ((Nat \to Ord) \to (Nat \to P) \to P))$ we can apply **Recursive call** and obtain:

$$(y : Nat,\ ord_{rec}(P, f_1, f_2, f_3, @(F, y)),\ P) \in CCl_{ord_{rec}}(P, f_1, f_2, f_3, lim(F))$$

The use of this rule is justified, since

$$\Gamma \vdash P, f_1, f_2, f_3, lim(F) (\succ^{\mathcal{RCS}}_{y:Nat, ord_{rec}(P, f_1, f_2, f_3, lim(F))})_{lex} P, f_1, f_2, f_3, @(F, y)$$

The only nontrivial part in the above judgment is

$$\Gamma \vdash lim(F) (\succ^{\mathcal{RCS}}_{y:Nat, ord_{rec}(P, f_1, f_2, f_3, lim(F))}) @(F, y)$$

But F is a recursive constructor subterm of $lim(F)$, $@(F, y)$ is a left-flattening of $@(F, y)$, $\Gamma \vdash F \succeq F\ :\ Nat \to Ord$ holds and $(y : Nat,\ y,\ Nat)$ is already in the computable closure. Finally we use **Abstraction** to get the result.

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

Example 15 (Constructors that do not satisfy star dependency)

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

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 is not a parameter. Consequently, the definition of well-formedness 2.3.3 is not satisfied and this inductive definition is not accepted in our system.

For the same reasons we cannot accept an inductive definition of the greatest fixpoint.

$$Ind[F : \star \to \star](nu : \star := CoIt : (X : \star)(X \to (FX)) \to X \to \overline{nu})$$

In practice, types violating the star dependency condition are quite rare; we have searched Coq's theories and found any; in Coq's contribution files, we have found 11.

There is also a class of inductive types that do satisfy star dependency, but whose elimination rules are not accepted by HORPO. These are types of the form $s : \overrightarrow{(p : P)}\overrightarrow{(x : b)}.\star$ such that at least one of the variables in $\vec{x}$ belongs to $\mathcal{V}ar^{\square}$. Let us see two examples of such a type:

Example 16

$$\begin{aligned} &Ind[\,](Sum : \star \to \star\ := \\ &\qquad B : Bool \to \overline{Sum}\, Bool \\ &\qquad N : Nat \to \overline{Sum}\, Nat) \end{aligned}$$

This type represents a disjoint sum of Bool and nat with the property that already the type tells whether one has to do with booleans or natural numbers. The type itself is accepted in our framework, but its elimination rules are not.

We have $sum_{dep} : (P : (A : \star)(Sum\, A) \to \star)((b : Bool)P\, Bool\, B(b)) \to ((n : Nat)P\, Nat\, N(n)) \to (A : \star)(u : Sum\, A)(P\, A\, u)$. Let us denote by Γ the environment $P : (A : \star)(SumA) \to \star, f_1 : (b : Bool)P\, Bool\, B(b), f_2 : (n : Nat)P\, Nat\, N(n), b : Bool, n : Nat$. The rules are the following:

$$\begin{aligned} \Gamma \vdash & sum_{dep}(P, f_1, f_2, Bool, B(b)) \to @(f_1, b)\ : P\, Bool\, B(b) \\ \Gamma \vdash & sum_{dep}(P, f_1, f_2, Nat, N(n)) \to @(f_2, n)\ : P\, Nat\, N(n) \end{aligned}$$

Intuitively, the occurrence of *Bool* (respectively *Nat*) as an argument of sum_{dep} is only needed to type-check $B(b)$ (respectively $N(n)$) and not to do elimination. Nevertheless the two rules are rejected. Indeed by the star dependency condition sum_{dep} can be given an arity $(4, 1)$ or $(5, 0)$ but for either of these choices the rules are non-parametric.

For the same reasons we cannot accept in HORPO elimination rule for the John-Major equality:

$$\begin{aligned}&Ind[A:\star, x:A](JMeq:(B:\star)B\rightarrow\star :=\\ &\qquad JMeq\_refl:\overline{JMeq}\,A\,x)\end{aligned}$$

where

$$JMeq_{dep}:(A:\star)(a:A)(P:(B:\star)B\rightarrow\star)(P\ A\ a)\rightarrow(B:\star)(b:B)(P\ B\ b)$$

has one rule

$$\begin{aligned}&A:\star, a:A, P:(B:\star)B\rightarrow\star, h:(P\ A\ a)\vdash\\ &\qquad JMeq_{dep}(A,a,P,h,A,a,JMeq\_refl(A,a))\rightarrow h\end{aligned}$$

7.3 Elimination rules for parametric inductive types

The notion of inductive type is inherently related to the notion of elimination rules which provide the semantics of inductive types as the smallest collections containing all terms build from constructors. In the previous section we presented several examples of elimination rules and here we show formally which elimination rules of inductive types can be accepted by HORPO.

Even though writing elimination rules for a given inductive type seems pretty standard, the actual form of the rules depends in a significant way on the chosen approach to inductive types. It turns out that we get different rules, depending whether inductive types are anonymous, named, defined with parameters, defined in environments or depending on the positivity condition.

The reference paper on elimination rules for inductive types in the calculus of constructions is [92]. Unfortunately, in this work the inductive types are quite different from ours, since they are anonymous. Our inductive types and their elimination rules follow rather those actually implemented in Coq [87].

Let us see which elimination rules can be written for inductive types in our framework and how they are accepted by HORPO. Without loss of generality we will consider only dependent eliminations; the non-dependent ones are simpler both to define and to accept in HORPO.

For the needs of this section suppose that Σ is a well-formed signature containing only inductive declarations.

First Σ is extended by adding the function symbols for which the elimination rules will be later defined. For every inductive type $s \in \Sigma$ we add the function symbol $Elim_s$ called eliminator. This extended signature is called Σ_{Elim}. Whenever needed, we will distinguish the typing relation of the calculus of constructions with constants for different signatures by adding an index, for example like in $\Gamma \vdash_{CC_\Sigma} a : b$.

The eliminator takes the parameters $\overrightarrow{p : P}$ of the inductive type, the elimination predicate Q, the elimination functions $\vec{f}$, one for each constructor, the arguments of the inductive type $\vec{x}$ and the inductive object u (of type $s(\vec{p})\vec{x}$) to be eliminated. To define the type of $Elim_s$ formally, we first need to know the types of $\vec{f}$. They are calculated using $\Delta\{\_\}$ below:

Definition 7.3.1 *Let $C \in Pos(\overline{s})$. Then $\Delta\{C, Q, u\}$ is defined by induction* *on the structure of C:*

$$\begin{array}{rcl}\Delta\{(\overline{s}\,\vec{t}),Q,u\} & = & Q\vec{t}u \\ \Delta\{(z{:}B).D,Q,u\} & = & (z:B)\Delta\{D,Q,u\,z\} \qquad\qquad\qquad \textit{if}\ \ \overline{s}\notin B \\ \Delta\{(z{:}B).D,Q,u\} & = & (z:\overrightarrow{(y:V)}(\overline{s}\,\vec{q}))(\overrightarrow{(y:V)}(Q\vec{q}(z\vec{y}))) \to \Delta\{D,Q,u\,z\} \\ & & \qquad\qquad\qquad\qquad\qquad \textit{if}\ \ B=\overrightarrow{(y:V)}.\overline{s}\,\vec{q}\end{array}$$

Definition 7.3.2 (Function symbol $Elim_s$) *Let $Ind[\Gamma_p](s^r : A := c_1^{(r,k_1)} : C_1, \dots c_m^{(r,k_m)} : C_m) \in \Sigma$ with $s : (p_1 : P_1)\dots(p_r : P_r).(x_1 : b_1)\dots(x_n : b_n).\star$, $C_i = \overrightarrow{(z:d)}.\overline{s}\vec{w}$ and $c_i : \overrightarrow{(p:P)}C_i[s(\vec{p})/\overline{s}]$. The type of $Elim_s$ is*

$$\begin{array}{l}\overrightarrow{(p:P)}(Q:\overrightarrow{(x:b)}(s(\vec{p})\vec{x})\to\star)\Delta\{C_1,Q,c_1\vec{p}\}[s(\vec{p})/\overline{s}]\to\dots \\ \qquad\qquad \Delta\{C_m,Q,c_m\vec{p}\}[s(\vec{p})/\overline{s}]\to(x_1:b_1)\dots(x_n:b_n)(u:s(\vec{p})\vec{x})(Q\,\vec{x}\,u)\end{array}$$

Looking at the type of $Elim_s$ we can see that the last argument u comes after the terms $\vec{x}$ which specify the type of u. The types of these $\vec{x}$ may be either types or kinds. Consequently, for the type of $Elim_s$ to satisfy the star dependency condition, the arity must be $(r+1+m+n,1)$. But then, parametricity imposes that in every rule for $Elim_s$ the first $r+1+m+n$ arguments are variables, which is not compatible with pattern matching over u.

For all these reasons, we will consider in the sequel inductive types for which $A=\overrightarrow{(x:b)}.\star$ with $\vec{x}\in\mathcal{V}ar^\star$.

Definition 7.3.3 *An inductive type s from the declaration $Ind[\Gamma_p](s^r : A := c_1^{(r,k_1)} : C_1, \dots c_m^{(r,k_m)} : C_m) \in \Sigma$ is* plain *if $A=\overrightarrow{(x:b)}.\star$ and $\vec{x}\in\mathcal{V}ar^\star$.*

We have seen examples of non-plain inductive type in Example 16. All other examples in Section 7.2 were plain.

Lemma 7.3.4 *The signature Σ augmented by declarations of $Elim_s$ for every plain inductive type s is well-formed.*

Proof
Of course, for every type constructor $s:\overrightarrow{(p:P)}\overrightarrow{(x:b)}.\star$ with $\vec{x}\in\mathcal{V}ar^\star$, function symbol $Elim_s$ belongs to $\mathcal{F}^{(r+1+m,n+1)}$ and it satisfies star dependency condition with respect to this arity.

We are left to show that the type of $Elim_s$ is a well-typed term of the calculus of constructions with constants from Σ. This follows easily from the fact that for every i

$$\Sigma, \overrightarrow{p:P}, Q : (\overrightarrow{(x:b)}(s(\vec{p})\vec{x}) \to \star) \vdash_{CC_0} \Delta\{C_i, Q, c_i\vec{p}\}[s(\vec{p})/\overline{s}] \ : \star$$

that we proceed to prove below.

Let $C = \overrightarrow{(z:d)}.\overline{s}\,\vec{t} \in Pos^k(\overline{s})$ and $c : \overrightarrow{(p:P)}C[s(\vec{p})/\overline{s}]$. Moreover, let us denote the environment $z_1 : d_1[s(\vec{p})/\overline{s}], \dots z_i : d_i[s(\vec{p})/\overline{s}]$ by Δ_i, the type $(z_{i+1} : d_{i+1}) \dots (z_k : d_k).\overline{s}\,\vec{t}$ by C^i and the vector $z_1 \dots z_i$ by Z_i.

We will show by induction on i decreasing from k to 0 that

$$\Sigma, \overrightarrow{p:P}, Q : (\overrightarrow{(x:b)}(s(\vec{p})\vec{x}) \to \star), \Delta_i \vdash_{CC_0} \Delta\{C^i, Q, c\vec{p}Z_i\}[s(\vec{p})/\overline{s}] \ : \star$$

If $i = k$ then $\Delta\{\overline{s}\,\vec{t}, Q, c\vec{p}\vec{z}\}[s(\vec{p})/\overline{s}] = Q\,\vec{t}\,c\vec{p}\vec{z}$ which is obviously of type $\star$. Otherwise $\Delta\{C^{i-1}, Q, c\vec{p}Z_{i-1}\} = (z_i : d_i)\Delta\{C^i, Q, c\vec{p}Z_i\}$ if $\overline{s} \notin d_i$ or $(z_i : d_i)(\overrightarrow{(y:V)}.(Q\,\vec{q}\,(z_i\vec{y}))) \to \Delta\{C^i, Q, c\vec{p}Z_i\}$ if $d_i = \overrightarrow{(y:V)}.\overline{s}\,\vec{q}$. By induction hypothesis $\Sigma, \overrightarrow{p:P}, Q : (\overrightarrow{(x:b)}(s(\vec{p})\vec{x}) \to \star), \Delta_i \vdash_{CC_0} \Delta\{C^i, Q, c\vec{p}Z_i\}[s(\vec{p})/\overline{s}] : \star$. Using the product formation rule, one gets the desired conclusion. □

Definition 7.3.5 (Elimination rules for parametric inductive types) *Let* $s : (p_1 : P_1) \dots (p_r : P_r)(x_1 : b_1) \dots (x_n : b_n).\star$, $C_i = \overrightarrow{(z:d)}.\overline{s}\vec{w}$ *and* $c_i : \overrightarrow{(p:P)}C_i[s(\vec{p})/\overline{s}]$ *and let* Γ_{s,c_i} *denote the environment*

$$\overrightarrow{p:P}, Q : (\overrightarrow{(x:b)}(s(\vec{p})\vec{x}) \to \star), \overrightarrow{f : \Delta\{C, Q, c\vec{p}\}[s(\vec{p})/\overline{s}]}, \overrightarrow{z : d[s(\vec{p})/\overline{s}]}$$

Then the rewrite rule for eliminating terms headed by c_i *is:*

$$\Gamma_{s,c_i} \vdash Elim_s(\vec{p}, Q, \vec{f}, \vec{w}, c_i(\vec{p}, \vec{z})) \to r\{\vec{p}, Q, \vec{f}, C_i, f_i\vec{p}, \emptyset, \vec{z}\}$$

where $r\{\vec{p}, Q, \vec{f}, C, v, \vec{b}, \vec{e}\}$ *is defined by induction on the proof that* $C \in Pos^k(\overline{s})$ *(*C *is a type of constructor) and under the assumption that* $|\vec{e}| = k$.

$$\begin{array}{rcl}
r\{\vec{p}, Q, \vec{f}, \overline{s}\vec{t}, v, \vec{a}, \emptyset\} & = & v \\
r\{\vec{p}, Q, \vec{f}, (z{:}B).D, v, \vec{a}, (e', \vec{e})\} & = & r\{\vec{p}, Q, \vec{f}, D, v\,e', (\vec{a}, e'), \vec{e}\} \qquad \textit{if } \overline{s} \notin B \\
r\{\vec{p}, Q, \vec{f}, (z{:}B).D, v, \vec{a}, (e', \vec{e})\} & = & r\{\vec{p}, Q, \vec{f}, D, \\
 & & v\,e'\,(\lambda\overrightarrow{y:V}.Elim_s(\vec{p}, Q, \vec{f}, \vec{q}, e'\vec{y})), (\vec{a}, e'), \vec{e}\} \\
 & & \qquad\qquad \textit{if } B = \overrightarrow{(y:V)}.\overline{s}\vec{q}
\end{array}$$

Lemma 7.3.6 *For every plain inductive type $s \in \Sigma$, all rewrite rules for $Elim_s$ are parametric.*

Proof

From the definition of the rules it follows that the arguments at parametric positions are different variables. Hence, the only nontrivial part in checking parametricity is to show that the right-hand side has an appropriate type.

By the definition of $Elim_s$ we have $\Gamma_{s,c_j} \vdash Elim_s(\vec{p}, Q, \vec{f}, \vec{w}, c_j(\vec{p}, \vec{z}))$: $Q\vec{w}c_j(\vec{p}, \vec{z})$. We need to show that $\Gamma_{s,c_j} \vdash r\{\vec{p}, Q, \vec{f}, C_j, f_j\vec{p}, \emptyset, \vec{z}\} : Q\vec{w}c_j(\vec{p}, \vec{z})$.

To this end, suppose that $C = \overrightarrow{(z:d)}.\overline{s}\,\vec{t} \in Pos^k(\overline{s})$ and $c : \overrightarrow{(p:P)}C[s(\vec{p})/\overline{s}]$. Let us denote the type $(z_{i+1} : d_{i+1}) \ldots (z_n : d_n).\overline{s}\,\vec{t}$ by C^i, the vector $z_1 \ldots z_i$ by Z_i and $z_{i+1} \ldots z_k$ by X_i.

We will show by induction on i decreasing from k to 0 that for every v satisfying $\Gamma_{s,c_j} \vdash v : \Delta\{C^i, Q, c\vec{p}\,Z_i\}[s(\vec{p})/\overline{s}]$ we have

$$\Gamma_{s,c_j} \vdash r\{\vec{p}, Q, \vec{f}, C^i, v, Z_i, X_i\} : \Box\{C^i, Q, (c\,\vec{p}\,Z_i), X_i\}$$

where

$$\begin{array}{rcl} \Box\{\overline{s}\,\vec{t}, Q, u, \emptyset\} & = & Q\vec{t}u \\ \Box\{(z:B).D, Q, u, (e', \vec{e})\} & = & \Box\{D, Q, u\,e', \vec{e}\} \end{array}$$

If $i = k$ then $r\{\vec{p}, Q, \vec{f}, C^i, v, Z_i, X_i\} = v$, and by assumption $\Gamma_{s,c_j} \vdash v : \Delta\{C^k, Q, c\vec{p}\,Z_k\}[s(\vec{p})/\overline{s}]$. Since $\Delta\{C^k, Q, c\vec{p}\,Z_k\}[s(\vec{p})/\overline{s}] = Q\,\vec{t}\,(c\vec{p}\,\vec{z})$ and $Q\,\vec{t}\,(c\vec{p}\,\vec{z}) = \Box\{C^k, Q, (c\,\vec{p}\,Z_k), X_k\}$, we are done.

Otherwise, $r\{\vec{p}, Q, \vec{f}, C^{i-1}, v, Z_{i-1}, X_{i-1}\}$ equals $r\{\vec{p}, Q, \vec{f}, C^i, (v\,z_i), Z_i, X_i\}$ if $\overline{s} \notin \mathcal{FV}(d_i)$ or $r\{\vec{p}, Q, \vec{f}, C^i, (v\,z_i\,(\lambda\overrightarrow{y:V}.Elim_s(\vec{p}, Q, \vec{f}, \vec{q}, z_i\vec{y}))), Z_i, X_i\}$ if $d_i = \overrightarrow{(y:V)}.\overline{s}\,\vec{q}$. Let us denote by v' the term $(v\,z_i)$ if $\overline{s} \notin \mathcal{FV}(d_i)$ and $(v\,z_i\,(\lambda\overrightarrow{y:V}.Elim_s(\vec{p}, Q, \vec{f}, \vec{q}, z_i\vec{y})))$ if $d_i = \overrightarrow{(y:V)}.\overline{s}\,\vec{q}$. Since $\Gamma_{s,c_j} \vdash v$: $\Delta\{C^{i-1}, Q, c\vec{p}\,Z_{i-1}\}[s(\vec{p})/\overline{s}]$, we have also $\Gamma_{s,c_j} \vdash v' : \Delta\{C^i, Q, c\vec{p}\,Z_i\}[s(\vec{p})/\overline{s}]$. Hence, by induction hypothesis we get $\Gamma_{s,c_j} \vdash r\{\vec{p}, Q, \vec{f}, C^i, v', Z_i, X_i\}$: $\Box\{C^i, Q, (c\,\vec{p}\,Z_i), X_i\}$. We conclude by noting that $\Box\{C^i, Q, (c\,\vec{p}\,Z_i), X_i\} = \Box\{C^{i-1}, Q, (c\,\vec{p}\,Z_{i-1}), X_{i-1}\}$. $\Box$

Below, we define a subclass of inductive types for which eliminations rules can be accepted using the status *Lex*, which is a lexicographic comparison from left to right. For other types one would have to use a little more complex status, namely a permutation of positions.

Definition 7.3.7 (Inductive type compatible with HORPO) *Inductive type* $s : (p_1 : P_1) \dots (p_r : P_r)(x_1 : b_1) \dots (x_n : b_n).\star$ *is* compatible with HORPO *if for every type of its constructor* $C_i = \overrightarrow{(z : d)}.\overline{s}\vec{w}$ *with* $d_i = \overrightarrow{(y : V)}.\overline{s}\vec{q}$ *we have*

$$\Gamma_{s,c_i} \vdash \vec{w}(\succ^{\mathcal{RCS}}_{\overrightarrow{y:V},l})_{lex}\vec{q} \quad \textit{or} \quad \vec{w} = \vec{q}$$

where Γ_{s,c_i} *is the environment from Definition 7.3.5 and* $l = Elim_s(\vec{p}, Q, \vec{f}, \vec{w}, c_i(\vec{p}, \vec{z}))$.

Lemma 7.3.8 *For every plain and compatible with HORPO inductive type* $s \in \Sigma$*, all rewrite rules for* $Elim_s$ *are accepted by HORPO.*

Proof
Recall that the rules are of the form

$$\Gamma_{s,c_i} \vdash Elim_s(\vec{p}, Q, \vec{f}, \vec{w}, c_i(\vec{p}, \vec{z})) \to r\{\vec{p}, Q, \vec{f}, C_i, f_i\vec{p}, \emptyset, \vec{z}\}$$

Let us abbreviate $Elim_s(\vec{p}, Q, \vec{f}, \vec{w}, c_i(\vec{p}, \vec{z}))$ by l and $r\{\vec{p}, Q, \vec{f}, C_i, f_i\vec{p}, \emptyset, \vec{z}\}$ by r. By the definition of r, the right-hand side is always an application of the variable f_i to the parameters $\vec{p}$, other arguments of constructor $\vec{z}$ and terms of the form $(\lambda\overrightarrow{y : V}.Elim_s(\vec{p}, Q, \vec{f}, \vec{q}, z_i\vec{y}))$ for $d_i = \overrightarrow{(y : V)}.\overline{s}\,\vec{q}$. If we show that these terms belong to the computable closure of $Elim_s$ then by case I.6 of HORPO the rule is accepted.

Let us concentrate on the proof that $\lambda\overrightarrow{y : V}.Elim_s(\vec{p}, Q, \vec{f}, \vec{q}, z_i\vec{y}) \in CCl(\Gamma_{s,c_i} \vdash l)$. To this end it is sufficient that $(\overrightarrow{y : V},\ Elim_s(\vec{p}, Q, \vec{f}, \vec{q}, z_i\vec{y}),\ Q\,\vec{q}\,z_i\vec{y}) \in CCl(\Gamma_{s,c_i} \vdash l)$. This can be done by **Recursive call** provided $\vec{p}$, Q, $\vec{f}$, $\vec{q}$, $z_i\vec{y}$ belong to $CCl(\Gamma_{s,c_i} \vdash l)$ and

$$\Gamma_{s,c_i} \vdash (\vec{p}, Q, \vec{f}, \vec{w}, c_i(\vec{p}, \vec{z}))(\succ^{\mathcal{RCS}}_{\overrightarrow{y:V},l})_{lex}(\vec{p}, Q, \vec{f}, \vec{q}, z_i\vec{y})$$

The latter condition is satisfied, because $\Gamma_{s,c_i} \vdash \vec{w}(\succ^{\mathcal{RCS}}_{\overrightarrow{y:V},l})_{lex}\vec{q}$ or $\vec{w} = \vec{q}$ and $\Gamma_{s,c_i}, \overrightarrow{y : V} \vdash c_i(\vec{p}, \vec{z})(\succ^{\mathcal{RCS}}_{\overrightarrow{y:V},l})z_i\vec{y}$, as z_i is a constructor subterm of $c_i(\vec{p}, \vec{z})$ and $(\overrightarrow{y : V},\ y_i,\ V_i) \in CCl(\Gamma_{s,c_i} \vdash l)$ for all i.

It remains to show that $(\overrightarrow{y : V},\ t,\ T) \in CCl(\Gamma_{s,c_i} \vdash l)$ for t ranging over $\vec{p}$, Q, $\vec{f}$, $\vec{q}$, $z_i\vec{y}$ and T being the type of t. Most of these terms ($\vec{p}$, Q, $\vec{f}$) belong to the initial set. Term $z_i\vec{y}$ belongs to the computable closure since z_i is in the initial set and $\vec{y}$ are BV variables. We are left with $\vec{q}$. These may be arbitrary complex terms, but recall that they come from $d_i = \overrightarrow{(y : V)}.\overline{s}\,\vec{q}$ and by the definition of well-formedness of the signature they are well-typed in the environment $\overrightarrow{z:d}, \overrightarrow{y:V}$ in the calculus of constructions with constants from Σ. Note that $\overrightarrow{z:d}$ is the last part of Γ_{s,c_i} and $\overrightarrow{y:V}$ is the environment of a tuple. Fortunately, the following result is easy to prove:

> Let $f : \overrightarrow{(p:P)}\overrightarrow{(x:b)}.c \in \mathcal{F}^{r+n}$ and $\Gamma_1, \Gamma_2 \vdash_{CC} f(\vec{p}, \vec{u}) \,:\, c[\vec{u}/\vec{x}]$.
> If $\Gamma_2, \Delta \vdash_{CC_\Sigma} a \,:\, b$ then $(\Delta,\ a,\ b) \in CCl(\Gamma_1, \Gamma_2 \vdash f(\vec{p}, \vec{u}))$.

The proof can be done by induction on the derivation of $\Gamma_2, \Delta \vdash_{CC_\Sigma} a \,:\, b$ and uses the fact that for every typing rule in CC_Σ there is a corresponding rule in the computable closure definition. It is important to note that there are no function symbols in Σ and consequently in the derivation of $\Gamma_2, \Delta \vdash_{CC_\Sigma} a \,:\, b$. $\square$

Chapter 8

Comparison with the general schema

The purpose of this chapter is to make a detailed comparison of our work with another method for proving termination of higher-order rewriting in the calculus of constructions: the general schema from the thesis of Frédéric Blanqui [17]. As already mentioned in the introduction, Blanqui's work is one of the few that is concerned with dependent rewriting in the calculus of constructions and conceptually the closest one to the present thesis.

The main importance of Blanqui's work is the possibility to define rewriting rules at the level of types and not only at the level of objects. This kind of rewriting enables to write, for example large elimination rules for inductive types.

If we restrict our attention to rewriting on objects then, roughly speaking, the general schema is a particular case of HORPO. Indeed, the general schema consists essentially of using the computable closure and the use of the computable closure is a subcase (I.1) of HORPO. Nevertheless, it is not the case that all object level rules accepted by Blanqui's criterion are accepted by our HORPO. Apart from some technical conditions, the reason is mainly hidden in different approaches to inductive types and constructors.

In [17] every function symbol whose output type is an instance of an inductive type I is considered as a constructor of I. Moreover, it is possible to define interpretations of inductive types in such a way that reaching under a constructor (at accessible positions) is safe from the computability point of view. This approach turns out to be very useful for accepting many practical examples.

But treating function symbols as constructors does not overcome the non-recursive character of the general schema. Indeed the computable closure from [17] in the rule **(symb$^=$)**, corresponding to our **(Recursive call)**, uses β-reduction and subterm relation and there is no way to do another call to the general schema.

Let us now discuss in more details the differences between our approaches. To this end we will have to introduce and explain the main definitions from [17]. For a better understanding, sometimes we will present very simple rewrite rules that are rejected because of some technical restrictions in these definitions. Removing these restrictions may unfortunately result in great complication in proofs. Moreover, we are aware, that there are many rules of comparable simplicity that cannot be accepted by HORPO for some other technical reasons.

Inductive types Inductive types considered in [17] are strictly positive and functional. Moreover they should satisfy a certain condition ((I6) from Definition 74 in [17]) which is very close to the safeness condition from [86]. This condition plays a role similar to star dependency; it serves to define interpretations of inductive types in such a way that reaching under a constructor is "safe" from the computability point of view. Syntactically, both star dependency and safeness force all $z \in \mathcal{V}ar^{\Box}$ from the type of constructor $\overrightarrow{(z:d)}.I\vec{w}$ to appear as variables in $\vec{w}$. Since safeness is only a little bit more general than star dependency, we do not consider this difference as significant.

The real difference in our approaches lies in the choice of constructors. In CC+H, the constructors of an inductive type I are given with I in its inductive declaration. In [17] every function symbol whose output type is an instance of an inductive type I is considered as a constructor of I. Reaching under a constructor is allowed only on accessible positions. They are essentially the positions whose type would be acceptable as a type of an argument of a classical constructor.

Example 17 *Suppose that plus has the type $Nat \to Nat \to Nat$. Then the positions $\{1,2\}$ corresponding to both arguments of plus are accessible.*

Now, suppose that map has the type $(A:\star)(B:\star)(ListA) \to (A \to B) \to (ListB)$. Only the positions $\{1,2\}$ are accessible, because the types of these positions (and only these) do not depend on other big variables than those appearing in the output type. Consequently it is not possible to reach *under map and deduce the computability of l or f from the computability of the term $map(A,B,l,f)$.*

This approach has two consequences. On the one hand, in the proofs and in the definition of the computable closure, it is possible to reach under a function symbol at its accessible positions. For some function symbols, like the polymorphic *map* above, it buys almost nothing since the set of accessible positions is not interesting. But for others, like *plus* or the version of *map* restricted to one type $(mapM : (A : \star)(A \to A) \to (List\ A) \to (List\ A))$ it is very useful. As a consequence more rules are easily accepted without recursive comparisons.

But on the other hand, treating function symbols as constructors changes in some sense the classical meaning of an inductive type. Traditionally it represents the smallest set build from a given set of constructors. Consequently all (ground) elements of an inductive type are built from constructors and they are all different. Without some additional conditions about completeness of definitions, this is no longer true in Blanqui's work. However, this approach seems to be an elegant way to consider quotient types.

It is worth noting that, in our framework, we could easily adopt the convention described above and treat function symbols as constructors. The changes would have to be done in the definition of interpretation of inductive types and in the proofs of the inductive and main lemmas where constructor-headed terms could rewrite.

Rewrite rules In our framework, a rule is a quadruple $\Gamma \vdash l \to r : T$ satisfying the parametricity condition (see Section 3.1.3).

In [17] a rewrite rule is a pair consisting of a left-hand side l and a right-hand side r. A rule is well-typed if there exists an environment Γ and a substitution ρ such that (among other conditions) $l\rho$ and r have equal types in environment Γ (in the calculus of constructions with rewriting already present in the conversion). This condition seems complicated and difficult to decide, but it is motivated by the fact that the left-hand side must be linear. Linearity is necessary to prove confluence (no other general result is known for proving confluence of rewriting and β-reduction) and confluence is needed for rewriting on types. Having l linear and only $l\rho$ typable is a solution to allow rules where nonlinearities are due to typing, which is quite common in definitions of polymorphic functions. Note however, that many interesting 'shortcut' rules (that is, equalities proved by induction in the calculus of inductive constructions that are not present in conversion) are naturally nonlinear, like in

$$\begin{array}{rcl} minus(x,x) & \to & 0 \\ nth\_tail(length(l1),\, append(l1,l2)) & \to & l2 \end{array}$$

Let us stress that these rules can be accepted in [17], if we restrict ourselves to rewriting on objects.

In [17] a well-typed rule (l, r, Γ, ρ) is well-formed if:

- $l = f(\vec{l})$ is algebraic (built from function symbols and variables),
- every variable from Γ is accessible in one of $l_1, \dots l_n$.

The first condition is again needed for confluence and it may be considered as inoffensive, due to the lack of higher order matching. Indeed, most interesting non-algebraic rules cannot be effectively used in this setting when only first-order matching is considered. Nevertheless, one can imagine a sensible rule that is rejected due to this requirement:

$$mapM(A, \lambda x\!:\!A.x, l) \to l$$

This rule is acceptable in our framework.

The second requirement is motivated by the definition of the computable closure which we discuss later.

Apart from the two above conditions, it is required that all rewrite rules must be *safe* ((A4)(b) from Definition 91 in [17]). A rewrite rule is *safe* if all big meaningful arguments of the left-hand are different variables. Up to the order of big arguments this corresponds exactly to our parametricity condition, which uses star dependency to find out which big arguments are meaningful.

Acceptance condition In [17] a well-formed rule $(f(\vec{l}), r, \Gamma, \rho)$ is acceptable, if it follows the general schema, i.e. if r belongs to the computable closure of $\vec{l}$ with respect to f. The main idea of this computable closure is the same as in our work, since they both follow the lines of the definition originally given in [18]. But despite these common roots there are also some differences.

Most computable closure rules describe how to get a new computable term by putting together already computable ones. The base case in this definition is constituted from terms accessible from $\vec{l}$.

In our framework the accessible set contains all terms that can be obtained by reaching under constructors in $\vec{l}$. In [17] the author demands that all variables from the environment are accessible, and only if this is true, he allows to use other computable closure rules to build other terms starting from these variables. Since function symbols are considered as constructors it is not that difficult to have all the variables accessible. Still, there are simple rules which seem to be absolutely correct from the strong normalization and computability point of view, which do not satisfy this condition. Consider for example two functions $\mathit{classify\_ord} : Ord \rightarrow Nat$ and $\mathit{is\_bigger\_than\_three} : Nat \rightarrow Bool$ defined by the rewriting system containing (among others) the following rules:

$$\begin{aligned} \mathit{classify\_ord}(zero) &\rightarrow O \\ \mathit{classify\_ord}(succ(x)) &\rightarrow s(O) \\ \mathit{classify\_ord}(lim(f)) &\rightarrow s(s(O)) \\ \mathit{is\_bigger\_than\_three}(\mathit{classify\_ord}(y)) &\rightarrow false \end{aligned}$$

Only one of the last two rules can be accepted by the general schema, because each of them requires different precedence between Nat and Ord in order to prove f or y accessible (since lim has the type $(Nat \rightarrow Ord) \rightarrow Ord$ the type constructor Nat must be smaller than Ord, and since $\mathit{classify\_ord}$ has the type $Ord \rightarrow Nat$, Ord must be smaller or equal to Nat).

On the other hand, if all variables from the environment are accessible then a nice lemma holds: if the types of function symbols are compatible with the precedence then every term well-typed in the calculus of constructions using only constants smaller than f belongs to the computable closure (compatibility means that function symbols appearing in the type assigned to a function symbol g have smaller precedence than g). Of course in our framework this lemma does not hold in general.

The main difference in the criterion for accepting rules lays in its recursive/non-recursive aspect. In our framework, the computable closure is a part of HORPO and it uses HORPO (or rather its constructor extensions) in the **(Recursive call)** rule. Apart from the computable closure, in the first part of HORPO there are six cases that can deal with different recursive calls according to the head symbol of the right-hand side.

In [17] the computable closure is the only available mechanism to accept rules and it is non-recursive because in the rule **(symb$^=$)**, corresponding to our **(Recursive call)**, the ordering used to compare arguments of f is either $>_1$ which is essentially $\rhd^{\mathcal{CS}}$ or $>_2$ corresponding more or less to $\rhd^{\mathcal{RCS}}$.

Before to go on with examples showing that the recursive aspect of the acceptance condition may be necessary, let us make a little comment on $>_2$. Unlike $>_1$, the comparison $c(\vec{a}) >_2 a_i\vec{u}$ holds only if a_i is a variable from the environment. Consequently, supposing that g is a 0-ary constant function symbol one cannot accept a simple rule like

$$plus(x, lim(g)) \rightarrow plus(x, @(g, s(0)))$$

(because g is a constant), even though a complicated rule like

$$plus(x, lim(f)) \rightarrow lim(\lambda n{:}Nat.plus(x, @(f, n)))$$

is accepted (f is a variable).

Let us now see some examples where the recursive aspect of HORPO is needed. Consider three polymorphic functions $map : (A : \star)(B : \star)(A \rightarrow B) \rightarrow (List\ A) \rightarrow (List\ B)$, $rev : (A : \star)(List\ A) \rightarrow (List\ A)$ and $rev\_map : (A : \star)(B : \star)(A \rightarrow B) \rightarrow (List\ A) \rightarrow (List\ B)$. Suppose that map has a standard definition (as in Example 6) and let rev and rev_map have tail-recursive definitions:

$$\begin{array}{rcl}
rev(A, l) & \rightarrow & rev_aux(A, l, nil) \\
rev_aux(A, nil, l2) & \rightarrow & l2 \\
rev_aux(A, cons(A, a, l1), l2) & \rightarrow & rev_aux(A, l1, cons(A, a, l2))
\end{array}$$

$$\begin{array}{rcl}
rev_map(A, B, f, l) & \rightarrow & rev_map_aux(A, B, f, l, nil) \\
rev_map_aux(A, B, f, nil, l2) & \rightarrow & l2 \\
rev_map_aux(A, B, f, cons(A, a, l1), l2) & \rightarrow & rev_map_aux(A, B, f, l1, \\
 & & \qquad cons(B, @(f, a), l2))
\end{array}$$

The following rule:

$$rev(A, map(A, B, f, l)) \rightarrow rev\_map(A, B, f, l)$$

is not accepted by the general schema, because f and l are not accessible variables. In HORPO, given $map > rev\_map$, using case I.1 we proceed to the trivial recursive comparison $map(A, B, f, l) \succ rev\_map(A, B, f, l)$. The monomorphic version of the above rule is accepted by both methods.

Let us consider another example where the recursive aspect of HORPO is needed. Let $nth : (A : \star)A \rightarrow Nat \rightarrow (List\ A) \rightarrow A$ represent a function such that $nth(A, default, n, l)$ computes the n-th element of the list l. The default argument is needed in case the list is too short. Suppose also that map has a tail-recursive definition like $map(A, B, f, l) \rightarrow rev(B, rev\_map(A, B, f, l))$ and nth has the following rules:

$$\begin{array}{rcl}
nth(B, d, n, nilB) & \rightarrow & d \\
nth(B, d, 0, cons(B, b, l)) & \rightarrow & b \\
nth(B, d, s(n), cons(B, b, l)) & \rightarrow & nth(B, d, n, l) \\
nth(B, d, s(n), map(A, B, f, cons(A, a, l))) & \rightarrow & nth(B, d, n, map(A, B, f, l))
\end{array}$$

As before, the last rule for nth is not accepted by the general schema since f and l are not accessible. In HORPO it is sufficient to suppose that nth has a lexicographic status and to use case I.4. This leads to the recursive comparisons $s(n) \succ^{\mathcal{RCS}} n$ which obviously holds and $map(A, B, f, cons(A, a, l)) \succ^{\mathcal{RCS}} map(A, B, f, l)$ which can be shown using once again case I.4. together with $cons(A, a, l) \succ^{\mathcal{RCS}} l$.

Before we conclude, let us consider an example that shows how the extended accessibility from [17] can sometimes replace a recursive comparison in HORPO.

Let $append$ have a type $(A : \star)(List\ A) \to (List\ A) \to (List\ A)$ and let us consider the rule

$$map(A, B, f, append(A, l1, l2)) \to append(B, map(A, B, f, l1), map(A, B, f, l2))$$

Assuming that $map >_{\mathcal{F}} append$ in the precedence, this rule is accepted by the general schema since all variables are accessible and $append(A, l1, l2) >_1 l1$ and $append(A, l1, l2) >_1 l2$ (as $>_1$ corresponds to constructor subterm and $append$ is a constructor).

It is also accepted by HORPO because, after using case I.2, one is left with $map(A, B, f, append(A, l1, l2)) \succ map(A, B, f, l1)$ and $map(A, B, f, append(A, l1, l2)) \succ map(A, B, f, l2)$ which can be shown by case I.4 and because $append(A, l1, l2) \succ l1$ and $append(A, l1, l2) \succ l1$.

Let us also present an example accepted by general schema and not accepted by HORPO.

Consider two functions: $mapM$ (that is map function restricted to one type) and $compM : (A : \star)(A \to A) \to (A \to A) \to (A \to A)$, which stands for a composition of functions. The following rule:

$$mapM(A, f, mapM(A, g, l)) \to mapM(A, compM(f, g), l)$$

is accepted by the general schema (if $mapM > compM$ and $mapM$ is lexicographic right), because all variables of the left-hand side are accessible and the term $compM(f, g)$ can be constructed by the computable closure. Our definition of HORPO does not permit to accept this rule. The intuitive reason is that there are no means to get from the left-hand side both f and g within the same recursive comparison. More precisely, $compM(f, g)$ cannot be dominated because it is not smaller than $mapM(A, g, l)$ and it does not belong to the computable closure of the whole left-hand side (because g is not accessible).

Note however, that the more general polymorphic version of the above rule is neither accepted by HORPO nor by the general schema:

$$map(A, B, f, map(B, C, g, l)) \to map(A, C, comp(A, B, C, f, g), l)$$

where $comp : (A : \star)(B : \star)(C : \star)(A \to B) \to (B \to C) \to (A \to C)$. In this rule the variables g and l are not accessible and the rule is rejected by general schema.

Summarizing, HORPO and the general schema do not subsume each other. On the one hand, the general schema from [17] may be seen as a non-recursive version of HORPO, but on the other hand the possibility to reach under a function symbol, and not only a constructor, reveals to be a very useful and strong mechanism. Another great advantage of [17] is the possibility to have rewriting on types. Combining HORPO and general schema may result in a very powerful tool allowing to extend the calculus of constructions in a consistent way with many interesting rewrite rules. This is left for the future work.

Chapter 9

Practical issues

The long term practical motivation of our work is the incorporation of rewriting in interactive theorem provers such as Coq [87]. In order to be suitable for implementation, 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 and define successive rewrite systems on the top of the previous ones. The modular termination criterion would then ensure him(her) that the entire system is strongly normalizing.

9.1 Decidability

In this section we show that HORPO is decidable, provided that we remove the **Reduction** rule from the computable closure.

Since HORPO, the computable closure and the constructor subterm extensions of HORPO are mutually dependent, we have to consider all these components simultaneously. We will show how their definitions can be transformed into four mutually recursive terminating procedures, called $\mathcal{HORPO}$ algorithms:

- $\mathcal{H}[\Gamma \vdash l \succ r \ : \ A]$,
- $\mathcal{CS}[\Gamma \vdash l \succ^{\mathcal{CS}} r]$,
- $\mathcal{RCS}[\Gamma \vdash l_i (\succ^{\mathcal{RCS}}_{BV, f(\vec{l})}) r]$, where l_i is an immediate subterm of $f(\vec{l})$,
- $\mathcal{CCL}_{\Gamma \vdash f(\vec{l})}(BV; r)$

The first three are decision procedures (returning true or false), saying whether the appropriate judgment holds.

The fourth algorithm, $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; r)$, returns either a type T such that $(BV, r, T) \in CCl(\Gamma \vdash f(\vec{l}))$ or $\perp$ if no such T exists. The algorithm is very similar in spirit to the type inference algorithm for the calculus of constructions.

All four definitions use the calculus of constructions for typing. Therefore, the corresponding algorithms will use the following auxiliary functions:

- $\mathcal{B}\text{eta}\mathcal{N}\text{orm}(t)$ calculating the β-normal form of a well typed term t,
- $\mathcal{B}\text{eta}\mathcal{E}\text{q}(t_1, t_2)$ returning true iff $t_1 =_\beta t_2$ for well typed terms t_1 and t_2,
- $\mathcal{C}[\Gamma \vdash_{CC} t : T]$ returning true iff $\Gamma \vdash_{CC} t : T$

The existence of the above functions is well known and follows from the confluence and strong normalization properties of β-reduction on well-typed terms in the calculus of constructions.

As the definitions of HORPO and its constructor extensions are in almost algorithmic form, we do not write the details of the algorithms $\mathcal{H}[\cdot]$, $\mathcal{CS}[\cdot]$ and $\mathcal{RCS}[\cdot]$. To get the procedures from the definitions it is enough to replace the judgments by the corresponding procedure calls:

$\Gamma \vdash l \succ r : A$	*by*	$\mathcal{H}[\Gamma \vdash l \succ r : A]$
$\Gamma \vdash l \succ^{\mathcal{CS}} r$	*by*	$\mathcal{CS}[\Gamma \vdash l \succ^{\mathcal{CS}} r]$
$\Gamma \vdash \vec{l} \succ^{\mathcal{CS}}_{mul} \vec{r}$	*by*	$\mathcal{CS}[\Gamma \vdash \vec{l} \succ^{\mathcal{CS}}_{mul} \vec{r}]$
$\Gamma \vdash l_i (\succ^{\mathcal{RCS}}_{BV, f(\vec{l})}) r$	*by*	$\mathcal{RCS}[\Gamma \vdash l_i (\succ^{\mathcal{RCS}}_{BV, f(\vec{l})}) r]$
$\Gamma \vdash \vec{l} (\succ^{\mathcal{RCS}}_{BV, f(\vec{l})})_{lex} \vec{r}$	*by*	$\mathcal{RCS}[\Gamma \vdash \vec{l} (\succ^{\mathcal{RCS}}_{BV, f(\vec{l})})_{lex} \vec{r}]$
$r \in CCl(\Gamma \vdash f(\vec{l}))$	*by*	$\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(\emptyset; r) \neq \perp$
$(BV, r, T) \in CCl(\Gamma \vdash f(\vec{l}))$	*by*	$\mathcal{B}\text{eta}\mathcal{E}\text{q}(\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; r), T)$
$\Gamma \vdash_{CC} t : T$	*by*	$\mathcal{C}[\Gamma \vdash_{CC} t : T]$

To avoid further proliferation of symbols, we keep the names $\mathcal{CS}[\cdot]$ and $\mathcal{RCS}[\cdot]$ for (obvious) algorithms deciding the corresponding lexicographic or multiset extensions.

The last procedure, $\mathcal{CCL}(\cdot)$, is given below in details:

Definition 9.1.1 (Type inference algorithm for the computable closure) *Let $f : \overrightarrow{(x : b)}.c \in \mathcal{F}$ and $\vec{l}$ be terms satisfying $\Gamma \vdash_{CC} f(\vec{l}) : c[\vec{l}/\vec{x}]$. Given an environment BV and a term M, the algorithm $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M)$ returns either a term or $\perp$ according to the following cases:*

- If $(BV, M, T) \in Acc(\Gamma \vdash f(\vec{l}))$ for some T then we check the well-formedness of BV with $\mathcal{C}[(\Gamma, BV) \vdash_{CC} \star : \Box]$ and if the latter succeeds $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = T$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = \star$ then again, we check BV with $\mathcal{C}[(\Gamma, BV) \vdash_{CC} \star : \Box]$ and if the latter succeeds, $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \Box$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = x$ then we check BV with $\mathcal{C}[(\Gamma, BV) \vdash_{CC} \star : \Box]$. If the latter succeeds then we look for an i such that $x = x_i \in dom(BV)$. If such i exists then $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = A_i$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = g(\vec{M})$, $g : \overrightarrow{(z : d)}.A \in \mathcal{F}$ and $f >_{\mathcal{F}} g$ or $M = c(\vec{M})$, $c : \overrightarrow{(z : d)}.A \in \mathcal{CS}$ or $M = s(\vec{M})$, $s : \overrightarrow{(z : d)}.A \in \mathcal{TC}$ then for all i we call $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M_i)$ and get T_i. If for all i, $\mathcal{B}\text{eta}\mathcal{E}\text{q}(T_i, d_i[\vec{M}/\vec{z}])$ then $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = A[\vec{M}/\vec{z}]$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = f(\vec{M})$, then for all i we call $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M_i)$ and get T_i. If for all i, $\mathcal{B}\text{eta}\mathcal{E}\text{q}(T_i, b_i[\vec{M}/\vec{z}])$ and $\mathcal{RCS}[\Gamma \vdash \vec{l}(\succ^{\mathcal{RCS}}_{BV, f(\vec{l})})_{lex}\vec{M}]$ or $\mathcal{CS}[\Gamma \vdash \vec{l} \succ^{\mathcal{CS}}_{mul} \vec{M}]$ holds, depending on the status of f, then $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = c[\vec{M}/\vec{z}]$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = \lambda x : M_1.M_2$ then we call $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M_1)$ and $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV, x : M_1; M_2)$ and get T_1 and T_2 respectively. If they are both different from $\perp$ and $\mathcal{C}[(\Gamma, BV, x : M_1) \vdash_{CC} T_2 : p]$ for $p = \star$ or $p = \Box$ then $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = (x : M_1).T_2$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = M_1\, M_2$ then we call $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M_1)$ and $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M_2)$ and get T_1 and T_2 respectively. If $\mathcal{B}\text{eta}\mathcal{N}\text{orm}(T_1) = (x : A).B$ and $\mathcal{B}\text{eta}\mathcal{N}\text{orm}(T_2) = A$ then $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = B[M_2/x]$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

- If $M = (x : M_1).M_2$ then we call $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M_1)$ and $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV, x : M_1; M_2)$ and get T_1 and T_2 respectively. If $T_1, T_2 \in \{\star, \Box\}$ then $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = T_2$; otherwise $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; M) = \perp$.

In order to analyze our algorithms, we first provide a measure showing the well-foundedness of their mutual recursive calls.

Definition 9.1.2 *Let I be an instance of a $\mathcal{HORPO}$ algorithm. We define its measure $|I|$ by:*

$$
\begin{array}{rcl}
|\mathcal{H}[\Gamma \vdash l \succ r \; : \; A]| & = & 2|l| + |r| + 1 \\
|\mathcal{CS}[\Gamma \vdash l \succ^{\mathcal{CS}} r]| & = & 2|l| + |r| + 2 \\
|\mathcal{RCS}[\Gamma \vdash l_i(\succ^{\mathcal{RCS}}_{BV, f(\vec{l})})r]| & = & 2|f(\vec{l})| + |r| \\
|\mathcal{CCL}_{\Gamma \vdash f(\vec{l})}(BV; r)| & = & 2|f(\vec{l})| + |r|
\end{array}
$$

where $|a|$ is the size of the term a.

Lemma 9.1.3 *Let I be an instance of a $\mathcal{HORPO}$ algorithm and suppose that all instances of the $\mathcal{HORPO}$ algorithms with measure smaller than $|I|$ terminate. Then I also terminates.*

Proof
The proof is based on the observation that for any instance I of the $\mathcal{HORPO}$ algorithms all recursive calls have a strictly smaller measure than $|I|$. Let us discuss the algorithms one by one.

Calculating $\mathcal{H}[\Gamma \vdash l \succ r \; : \; A]$ requires calling $\mathcal{C}[\Gamma \vdash_{CC} l \; : \; A]$ and $\mathcal{C}[\Gamma \vdash_{CC} r \; : \; A]$ and then checking whether one of the ten cases from the definition of HORPO holds. In all these cases, apart from finitely many calls to the $\mathcal{HORPO}$ algorithms, all one has to do is to choose a subterm of l or a left-flattening of r, to check precedence inequalities and status of function symbols and to do some convertibility checks in the calculus of constructions. Since there are only finitely many subterms of l and r, it remains to show that the recursive calls used in the ten HORPO cases have a smaller measure.

For $I = \mathcal{H}[\Gamma \vdash l \succ r \; : \; A]$ the measure is $|I| = 2|l| + |r| + 1$. The recursive calls we refer to in the definition of $\mathcal{H}[\Gamma \vdash l \succ r \; : \; A]$ are of the following form:

- $\mathcal{H}[\Gamma' \vdash l' \succ r' \; : \; A']$ where l' is a subterm of l and r' a proper subterm of r (for example in Part II),
- $\mathcal{CS}[\Gamma \vdash l' \succ^{\mathcal{CS}} r']$ where l' is a proper subterm of l and r' a subterm of r (for example in $\mathfrak{P}$),
- $\mathcal{RCS}[\Gamma \vdash l'(\succ^{\mathcal{RCS}}_{\emptyset, l})r']$ where $l = f(\vec{l})$, r' is a subterm of r and l' is an immediate subterm of l,

- $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(\emptyset; r)$, where $l = f(\vec{l})$

The measure of $\mathcal{H}[\Gamma' \vdash l' \succ r' : A']$ equals $2|l'| + |r'| + 1$ which is smaller than $2|l| + |r| + 1$ because r' is a proper subterm of r and l' is a subterm of l. The measure of $\mathcal{CS}[\Gamma \vdash l' \succ^{\mathcal{CS}} r']$ is $2|l'| + |r'| + 2$ which is smaller than $2|l| + |r| + 1$ because l' is a proper subterm of l. The measures of the other two kinds of calls are smaller or equal to $2|l| + |r|$ which is strictly smaller than $2|l| + |r| + 1$.

To calculate $\mathcal{CS}[\Gamma \vdash l \succ^{\mathcal{CS}} r]$, we first have to find a constructor subterm l' of l such that $\mathcal{H}[\Gamma \vdash l' \succ r : T]$. For $I = \mathcal{CS}[\Gamma \vdash l \succ^{\mathcal{CS}} r]$, the measure is $|I| = 2|l| + |r| + 2$. The measure of $\mathcal{H}[\Gamma \vdash l' \succ r : T]$ equals $2|l'| + |r| + 1$ and is always smaller than $|I|$. Consequently $\mathcal{CS}[\Gamma \vdash l \succ^{\mathcal{CS}} r]$ terminates, as clearly there are only finitely many subterms of l.

To calculate $\mathcal{RCS}[\Gamma \vdash l_i(\succ^{\mathcal{RCS}}_{BV,f(\vec{l})})r]$ where l_i is an immediate subterm of $f(\vec{l})$, one has to find a recursive constructor subterm l' of l_i, a left-flattening $@(r_0, r_1, \ldots r_k)$ of r and then call some $\mathcal{H}[\cdot]$ and $\mathcal{CCL}(\cdot)$ instances of the form:

- $\mathcal{H}[\Gamma \vdash l' \succ r' : A']$, where r' is a subterm of r,

- $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; r_i)$ for $i = 1, \ldots k$.

The measure of $\mathcal{H}[\Gamma \vdash l' \succ r' : A']$ equals $2|l'| + |r'| + 1$ and is smaller than $2|f(\vec{l})| + |r|$, because l' is a subterm of $f(\vec{l})$. Obviously, $|\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; r_i)| = 2|f(\vec{l})| + |r_i| < 2|f(\vec{l})| + |r|$.

Since there are only finitely many subterms of l_i and only finitely many left-flattenings of r, $\mathcal{RCS}[\Gamma \vdash l_i(\succ^{\mathcal{RCS}}_{BV,f(\vec{l})})r]$ is terminating.

We are left with the computable closure and all instances of the form $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(BV; r)$. In that case $|I| = 2|f(\vec{l})| + |r|$. All recursive calls to $\mathcal{CCL}_{\Gamma\vdash f(\vec{l})}(\cdot;\cdot)$ are made with a smaller r, so the measure decreases. Other recursive calls are made only if $r = f(\vec{r})$. In that case both possible calls, $\mathcal{RCS}[\Gamma \vdash \vec{l}(\succ^{\mathcal{RCS}}_{BV,f(\vec{l})})_{lex}\vec{r}]$ and $\mathcal{CS}[\Gamma \vdash \vec{l} \succ^{\mathcal{CS}}_{mul} \vec{r}]$ reduce to several basic comparisons of the form $\mathcal{RCS}[\Gamma \vdash l'(\succ^{\mathcal{RCS}}_{BV,f(\vec{l})})r']$ or $\mathcal{CS}[\Gamma \vdash l' \succ^{\mathcal{CS}} r']$, where r' is an immediate subterm of r and l' is an immediate subterm of $f(\vec{l})$. The measures of those basic comparisons are $2|f(\vec{l})| + |r'|$ and $2|l'| + |r'| + 2$ respectively, hence always smaller than $2|f(\vec{l})| + |r|$. $\square$

Before giving the final theorem about the decidability of HORPO, we have to prove the correctness of $\mathcal{HORPO}$ algorithms and especially $\mathcal{CCL}(\cdot)$.

Lemma 9.1.4 (Correctness of type inference for the computable closure) *Let $f : \overrightarrow{(x:b)}.c \in \mathcal{F}$, $\vec{l}$ be terms satisfying $\Gamma \vdash_{CC} f(\vec{l}) : c[\vec{l}/\vec{x}]$, BV be an environment and M be a term. Suppose also that the $\mathcal{HORPO}$ algorithms are correct for all instances with a measure smaller than $2|f(\vec{l})| + |M|$.*

- *If there exists a term T such that $(BV,\ M,\ T) \in CCl(\Gamma \vdash f(\vec{l}))$ then $T \leftrightarrow_\beta \mathcal{CCL}_{\Gamma \vdash f(\vec{l})}(BV; M)$.*

- *If there is no term T such that $(BV,\ M,\ T) \in CCl(\Gamma \vdash f(\vec{l}))$ then $\mathcal{CCL}_{\Gamma \vdash f(\vec{l})}(BV; M) = \bot$.*

Proof
By the Computable Closure Lemma, given BV and M, for all T such that $(BV,\ M,\ T) \in CCl(\Gamma \vdash f(\vec{l}))$, we have $\Gamma, BV \vdash_{CC} M : T$. By uniqueness of typing in the calculus of constructions and by the **(Conversion)** rule of the computable closure, either there is no T such that $(BV,\ M,\ T) \in CCl(\Gamma \vdash f(\vec{l}))$ or the latter holds for all terms T from one abstraction class modulo $\leftrightarrow_\beta$.

Given that remark, the rest of the proof follows by case analysis on the structure of M. $\square$

Theorem 3 (Decidability of HORPO) *If we remove the rule **(Reduction)** from the definition of computable closure, then given an environment Γ, terms l, r and A it is decidable whether $\Gamma \vdash l \succ r : A$.*

Proof
It follows easily from the correctness of $\mathcal{HORPO}$ algorithms and decreasingness of their recursive calls. $\square$

9.1.1 Complexity

The worst-case complexity of HORPO is non-elementary, because one has to perform β convertibility tests during the verification of a HORPO judgment. Since types to be compared may contain arbitrary terms, there is no limit on the complexity of HORPO, other than the complexity of β convertibility itself [85, 64].

Fortunately, this worst-case scenario is very rare in practice, as types of real-world rewriting rules and constant symbols are usually in "almost" normal form. Therefore, it is interesting to consider the relative complexity of HORPO, that is with the assumption of an oracle deciding β convertibility.

It is obvious that the relative complexity of the type-checking in the calculus of constructions is linear. Similarly checking whether a tuple is in the computable closure is linear. Consequently from the measure argument in the previous section it follows easily that the relative complexity of a HORPO judgment $\Gamma \vdash s \succ t : A$ is polynomial in the size of the two terms s and t.

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 can be used in a modular way. Modularity means that rules may be given step by step, possibly using the rules given at a previous step for typing the new ones. Consider for example the rules for *plus* and *append* (inductive type *Listl* corresponds to integer lists with length).

$$\begin{array}{rcl} plus & : & Nat \to Nat \to Nat \\ & & plus(0,\ n) \to n \\ append & : & (n,m : Nat)\ (Listl\ n) \to (Listl\ m) \to (Listl\ plus(n,\ m)) \\ & & append(0,\ m,\ nill,\ lm) \to lm \end{array}$$

Unless the rule for *plus* is used 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 for proving termination can be used 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 $n \geq 1$, 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 first three 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. Two points are important: that CC is defined beforehand and independently from $CC + H$ and that the definition of CC makes $CC + H$ have the nice properties described in Section 4.

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 7.1.3).

Chapter 10

Conclusions and future work

10.1 Conclusions

We have defined an extension of the calculus of constructions by higher-order and dependent rewriting and presented a decidable criterion (HORPO) which guarantees the strong normalization of the resulting calculus.

Our assumptions are threefold:

1. We consider rewriting at the object-level only.
2. Types of function symbols have to satisfy the star dependency condition, and the rewrite rules have to be parametric.
3. Types of 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 [92, 40, 41, 17].

It is not clear how important the second assumption is. At present, it is crucial for the proof of normalization and we believe that it should be difficult to weaken it significantly.

The third assumption was introduced in order to define the interpretation of inductive types in such a way that reaching under a constructor is safe from the computability point of view. This condition can be relaxed by choosing a set of accessible positions, as discussed in Section 6.2. On the other hand it seems that star dependency on constructor types is not a severe restriction (see comments to Example 15).

10.2 Future work

In this section we discuss further research topics and directions.

Star dependency and parametricity condition on rewrite rules. Star dependency and parametricity condition imposed on rewrite rules are two major concepts of our work. They follow from the proof of the Fun Lemma 5.4.5 and prevent the acceptance of non-terminating rules, like

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

for $J : (A : \star)(B : \star)A \to B \to A$

Nevertheless, there are still examples of rewrite rules that terminate but do not satisfy parametricity condition. These are, for example, elimination rules of inductive types that are not plain (see Example 16). One may also conjecture that the rules like

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

where $map : (A : \star)(B : \star)(A \to B)List(A) \to List(B)$, do not lead to non-termination.

It is very interesting to find out whether the above rule for *map* is terminating and whether this follows from the fact that this rule is confluent with two standard rules completely defining *map* or that this rule is an inductive theorem on lists.

More generally, it is important to better understand the problem of type arguments of function symbols and to find a better criterion on the form of rewrite rules that guarantees normalization.

HORPO with decreasing types. The current version of HORPO can compare only terms of equal types. Since function symbols are dependently typed, sometimes one needs to recursively compare two terms of different types. Typically, it concerns the "shortcut" rules. Let us consider the functions $OR : (n : Nat)(LList\ Bool\ n) \to Bool$ that computes the disjunction of the elements on the list and $putF : (n : Nat)(LList\ Bool\ n) \to LList\ Bool\ s(n)$ that puts one $false$ value at the end of a given list. The type $LList$ is defined in Example 12 and the rules defining OR and $putF$

are:

$$OR(0, Lnil(Bool)) \to false$$
$$OR(s(n), l) \to or(a, OR(n, l))$$
$$putF(0, Lnil(Bool)) \to Lcons(Bool, 0, false, Lnil(Bool))$$
$$putF(s(n), Lcons(Bool, n, a, l)) \to Lcons(Bool, s(n), a, putF(n, l))$$

Then, one can imagine a "shortcut" rule

$$OR(s(n), putF(n, l)) \to OR(n, l)$$

which says that we can omit the application of *putF* while calculating *OR*. To accept this rule in HORPO, one needs to compare recursively $putF(n, l)$ with l. But it cannot be done as they have different types.

The solution, coming from [54] is to allow HORPO to compare terms whose types decrease in a certain ordering on types $\geq_T$. We define HORPO$_T$ like in definition 3.1.2 except that now we compare couples composed of a term and its type:

Definition 10.2.1 (HORPO$_T$) *The higher order recursive path ordering with decreasing types is defined as follows:*

$$\Gamma \vdash l : A \succ_T r : B \quad \textit{iff}$$
$$\Gamma \vdash_{CC} l \,:\, A \qquad \Gamma \vdash_{CC} r \,:\, B \qquad \Gamma \vdash A \geq_T B \quad \textit{and}$$

... (the rest of the definition is the same as in 3.1.2)

Of course, correct rewrite rules will have the left-hand and right-hand sides of the same type, and only during checking these rules in HORPO, the recursive calls will possibly use comparison on types. The new version of the calculus of constructions with rewriting, CC+H$_T$, will have the following typing rule introducing rewriting:

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

We conjecture that the extended version of HORPO does not compromise the termination of CC+H$_T$, under some conditions on the ordering $\geq_T$.

Conjecture 1 *Every well typed term of CC+H$_T$ is strongly normalizing, provided that the typing ordering $\geq_T$ satisfies the following conditions:*

1. $\geq_T \cup \rhd$ *is well-founded,*

2. *for every* $s \in \mathcal{TC}$, $\Gamma \vdash s(\vec{a})\vec{t} \geq_T A$ *implies that* $A = s(\vec{a})\vec{u}$ *for some* $\vec{u}$.

3. $\Gamma \vdash (x:B).D \geq_T A$ *implies* $A = (x:B).D'$ *with* $\Gamma, x : B \vdash D \geq_T D'$

Introducing $\geq_T$ comparisons to HORPO, makes it necessary to consider the closure of $\geq_T$ in the definition of the strongly normalizing terms, the candidates and the definition of interpretation. This closure must be transitive, closed by substitution and context and compatible with β-convertibility.

Like in [54] one can imagine also that the ordering $\geq_T$ is itself given in the RPO like form, or even that $\geq_T$ and HORPO are given simultaneously by the same definition.

Note that having types comparison in HORPO is something different than rewriting on types. In the former case rewrite rules must still be on the level of objects.

Logical consistency. The consistency problems of CC+H are difficult, since CC+H is not a closed system, like the calculus of constructions or the calculus of inductive constructions from [92]. CC+H is open, in the sense that there is an initial environment built from constants whose types can be almost arbitrary (even uninhabited). For this reason we cannot repeat the standard syntactical proof inspecting the normal form of the hypothetical proof of $(x:\star).x$ (or the inductive type $False$) in the empty environment. Neither we can try to use the standard translation from the calculus of constructions to F_ω, because during this translation a consistent environment may become contradictory (for example, $f : ((P : Bool \to \star)P\,true \to P\,false) \to (x:\star).x$ gets translated to $f : ((P:\star)P \to P) \to (x:\star).x$ and $f(\lambda A{:}\star.\lambda x{:}A.x)$ is of type $(x:\star).x$).

One may rather be interested in a conservativity result over the calculus of inductive constructions, like

$$\text{If } \ \Gamma, \epsilon \vdash_{CIC} A : \star \text{ and } \ \Gamma \vdash_\epsilon t : A \text{ then } \ \text{there exists } t' \\ \text{such that } \ \Gamma, \epsilon \vdash_{CIC} t' : A,$$

where Γ represents hypotheses (function symbols and their types), $\vdash_\epsilon$ corresponds to the calculus with conversion extended by relation ϵ (rewriting) and Γ, ϵ corresponds to the environment where the relation ϵ is somehow presented in an equational form.

Another interesting consistency problem, in the presence of rewriting, is whether the term structure does not collapse.

Rewriting on types. As already said rewriting at the level of types is a difficult question. In his thesis [17] Blanqui extended the general schema to the rewriting on types, under the assumption of confluence. Adapting ideas from his paper to our framework may result in a very powerful tool allowing to extend the calculus of constructions in a consistent way with many interesting rewrite rules.

Confluence. Of course neither HORPO, nor $\rightarrow$ are confluent. Nevertheless, confluence is a desired property.

First, it is probably necessary for rewriting on types (see [17]). Next, it is needed to show the decidability of type-checking of CC+R for a given user-defined rewrite system R. Finally, it is also possible that confluence is useful for consistency reasons.

Bibliography

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[2] Andreas Abel. Termination checking with types. Technical Report 0201, Institut für Informatik, Ludwig-Maximilians-Universität München, 2002.

[3] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, 1998.

[4] Franco Barbanera and Maribel Fernández. Combining first and higher order rewrite systems with type assignment systems. In M. Bezem and J. F. Groote, editors, *Proceedings of 1st International Conference on Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 60–74, Utrecht, The Netherlands, March 1993.

[5] Franco Barbanera and Maribel Fernández. Modularity of termination and confluence in combinations of rewrite systems with lambda omega. In A. Lingas, R. Karlsson, and S. Carlsson, editors, *20th International Colloquium on Automata, Languages and Programming*, volume 700 of *LNCS*, pages 657–668, Lund, Sweden, July 1993. Springer-Verlag.

[6] Franco Barbanera, Maribel Fernández, and Herman Geuvers. Modularity of strong normalization and confluence in the λ-algebraic-cube. In *Proceedings of the Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 406–415, Paris, France, July 1994. IEEE Comp. Soc. Press.

[7] Franco Barbanera, Maribel Fernández, and Herman Geuvers. Modularity of strong normalization in the algebraic-λ-cube. *Journal of Functional Programming*, 7(6):613–660, 1997.

[8] Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[9] Henk Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 2, pages 117–309. Oxford University Press, 1992.

[10] Henk Barendregt and Herman Geuvers. Proof-assistants using dependent type systems. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 18, pages 1149–1238. Elsevier Science, 2001.

[11] G. Barthe, M. J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, To appear.

[12] Gilles Barthe and Herman Geuvers. Modular properties of algebraic type systems. In *Proceedings of HOA'95 (Higher Order Algebra)*, volume 1074 of *LNCS*, pages 37–56. Springer-Verlag, 1996.

[13] Gilles Barthe and Femke van Raamsdonk. Termination of algebraic type systems: the syntactic approach. In M. Hanus, J. Heering, and K. Meinke, editors, *Proceedings of Algebraic and Logic Programming, 6th International Joint Conference, ALP '97 - HOA '97*, volume 1298 of *LNCS*, pages 174–193. Springer-Verlag, 1997.

[14] Stefano Berardi. *Type dependence and constructive mathematics.* PhD thesis, University of Torino, Italy, 1990.

[15] Frédéric Blanqui. Inductive types in the calculus of algebraic constructions. Draft, to be submitted.

[16] Frédéric Blanqui. Definitions by rewriting in the calculus of constructions. In *Sixteenth Annual IEEE Symposium on Logic in Computer Science*, pages 9–18, Los Alamitos, USA, June 2001. IEEE Comp. Soc. Press.

[17] Frédéric Blanqui. *Théorie des Types et Récriture.* PhD thesis, Université Paris-Sud, 2001.

[18] Frédéric Blanqui, Jean-Pierre Jouannaud, and Mitsuhiro Okada. The Calculus of Algebraic Constructions. In Narendran and Rusinowitch [71].

[19] Freédéric Blanqui, Jean-Pierre Jouannaud, and Mitsuhiro Okada. Inductive data type systems. *Theoretical Computer Science*, 272(1–2):41–68, 2002.

[20] Eduardo Bonelli, Delia Kesner, and Alejandro Ríos. From higher-order to first-order rewriting. In Middeldorp [70], pages 47–62.

[21] P. Borovansky, C. Kirchner, H. Kirchner, P. E. Moreau, and M. Vittek. ELAN: A logical framework based on computational systems. In *Proceedings of the First International Workshop on Rewriting Logic*, volume 4 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1996.

[22] Val Breazu-Tannen. Combining algebra and higher-order types. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 82–90, Edinburgh, Scotland, July 1988. IEEE Comp. Soc. Press.

[23] Val Breazu-Tannen and Jean H. Gallier. Polymorphic rewriting conserves algebraic strong normalization. *Theoretical Computer Science*, 83(1):3–28, June 1991.

[24] Val Breazu-Tannen and Jean H. Gallier. Polymorphic rewriting conserves algebraic confluence. *Information and Computation*, 114(1):1–29, October 1994.

[25] Alonzo Church. A set of postulates for the foundations of logic. *Annals of Mathematics*, 33:346–366, 1932.

[26] Alonzo Church. A set of postulates for the foundations of logic. *Annals of Mathematics*, 34:839–864, 1933.

[27] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[28] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. The Maude system. In Narendran and Rusinowitch [71], pages 240–243.

[29] Thierry Coquand. Pattern matching with dependent types. In *Proceedings of the Workshop on Types for Proofs and Programs*, pages 71–83, Båstad, Sweden, 1992.

[30] Thierry Coquand and Jean H. Gallier. A proof of strong normalization for the Theory of Constructions using a Kripke-like interpretation.

In G. Huet and G. Plotkin, editors, *Preliminary Proceedings 1st International Workshop on Logical Frameworks*, pages 479–497, Antibes, France, 1990.

[31] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, February 1988.

[32] Thierry Coquand and Christine Paulin-Mohring. Inductively defined types. In P. Martin-Löf and G. Mints, editors, *Proceedings of Colog'88*, volume 417 of *LNCS*. Springer-Verlag, 1990.

[33] Haskell B. Curry and Robert Feys. *Combinatory Logic I.* North-Holland, Amsterdam, 1958.

[34] N.G. de Bruijn. The mathematical language AUTOMATH, its usage and some of its extensions. In *Symposium on automatic demonstration (IRIA, Versailles)*, number 125 in Lecture Notes in Mathematics, pages 29–61. Springer, 1968.

[35] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, March 1982.

[36] Nachum Dershowitz. Equations as programming language. In *Proceedings of the Fourth Jerusalem Conference on Information Technology*, pages 114–124, Jerusalem, Israel, May 1984. IEEE Comp. Soc. Press.

[37] Nachum Dershowitz. Computing with rewrite systems. *Information and Control*, 64(2/3):122–157, May/June 1985.

[38] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 6, pages 243–309. North-Holland, 1990.

[39] Daniel J. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. *Information and Computation*, 101(2):251–267, December 1992.

[40] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo, 1998. Rapport de recherche INRIA 3400.

[41] Gilles Dowek and Benjamin Werner. Proof normalization modulo. In T. Altenkirch, W. Naraschewski, and B. Rues, editors, *Proceedings of the Workshop on Types for Proofs and Programs*, volume 1657 of *LNCS*, pages 62–77. Springer-Verlag, 1999.

[42] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ-2. In B. Reid, editor, *Proceedings 12th ACM Symposium on Principles of Programming Languages*, pages 52–66. ACM Press, 1985.

[43] Eduardo Giménez. Codifying guarded definitions with recursive schemes. In P. Dybjer, B. Nordström, and J. Smith, editors, *Selected Papers 2nd International Workshop on Types for Proofs and Programs*, volume 996 of *LNCS*, pages 39–59. Springer-Verlag, 1995.

[44] Eduardo Giménez. *Un Calcul de Constructions Infinies et son Application à la Vérification des Systèmes Communicants*. PhD thesis, Ecole Normale Supérieure de Lyon, 1996.

[45] Eduardo Giménez. Structural recursive definitions in type theory. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *LNCS*, pages 397–408, Aalborg, Denmark, July 1998. Springer-Verlag.

[46] Jean-Yves Girard. 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 J.E. Fenstad, editor, *Proceedings of the 2nd Scandinavian Logic Symposium*, pages 63–92. North-Holland, 1971.

[47] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris 7, 1972.

[48] William A. Howard. To H.B. Curry: The formulae-as-types notion of construction. In J. Hindley and J. Seldin, editors, *Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Academic Press, 1969.

[49] IEEE Comp. Soc. Press. *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science*, Amsterdam, The Netherlands, July 1991.

[50] Jean-Pierre Jouannaud and Mitsuhiro Okada. Executable higher-order algebraic specification languages. In *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science* [49], pages 350–361.

[51] Jean-Pierre Jouannaud and Mitsuhiro Okada. Abstract data type systems. *Theoretical Computer Science*, 173(2):349–391, February 1997.

[52] Jean-Pierre Jouannaud and Albert Rubio. A recursive path ordering for higher-order terms in η-long β-normal form. In H. Ganzinger, editor, *7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *LNCS*, pages 108–122, New Brunswick, NJ, USA, July 1996. Springer-Verlag.

[53] Jean-Pierre Jouannaud and Albert Rubio. The higher-order recursive path ordering. In Giuseppe Longo, editor, *Fourteenth Annual IEEE Symposium on Logic in Computer Science*, Trento, Italy, July 1999. IEEE Comp. Soc. Press.

[54] Jean-Pierre Jouannaud and Albert Rubio. Higher-order recursive path orderings "à la carte". Invited lecture in Workshop on Termination, Netherland, april 2001.

[55] Fairouz Kamareddine and Alejandro Ríos. A lambda-calculus à la De Bruijn with explicit substitutions. In S. Doaitse Swierstra and M. Hermenegildo, editors, *7th International Symposium on Programming Languages: Implementations, Logics and Programs*, volume 982 of *LNCS*, pages 45–62, Utrecht, The Netherlands, September 1995. Springer-Verlag.

[56] Delia Kesner. Confluence of extensional and non-extensional λ-calculi with explicit substitutions. *Theoretical Computer Science*, 238(1–2):183–220, May 2000.

[57] Jan Willem Klop. *Combinatory Reduction Systems*. PhD thesis, Utrecht University, 1980. Published as Mathematical Center Tract 129.

[58] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–117. Oxford University Press, 1992.

[59] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–298. Pergamon Press, Oxford, 1967. Appeared 1970.

[60] Jean-Louis Krivine. Un algorithme non typable dans le système F. In *Comptes Rendus de L'Académie des Sciences Paris t. 304*, Série I, pages 123–126. 1987.

[61] Dallas S. Lankford. Canonical inference. Memo ATP-32, Automatic Theorem Proving Project, University of Texas, Austin, TX, December 1975.

[62] Pierre Lescanne. From lambda-sigma to lambda-upsilon a journey through calculi of explicit substitutions. In *Proceedings 21st ACM Symposium on Principles of Programming Languages*, pages 60–69, Portland, Oregon, January 1994. ACM Press.

[63] Carlos Loría-Sáenz and J. Steinbach. Termination of combined (rewrite and λ-calculus) systems. In *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems*, volume 656 of *LNCS*, pages 143–147, Pont-à-Mousson, France, 1992. Springer-Verlag.

[64] Harry G. Mairson. A simple proof of a theorem of Statman. *Theoretical Computer Science*, 103(2):387–394, 1992.

[65] Per Martin-Löf. An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Sheperdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic*, pages 73–118. North-Holland, 1975.

[66] Per Martin-Löf. *Intuitionistic Type Theory.* Biblioplois, Napoli, 1984. Notes of Giowanni Sambin on a series of lectues given in Padova.

[67] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(1):3–29, February 1998.

[68] Nax Paul Mendler. First- and second-order lambda calculi with recursive types. Technical report TR 86-764, Dept. of Computer Science, Cornell University, Ithaca, NY, July 1986.

[69] Nax Paul Mendler. Recursive types and type constraints in second-order lambda-calculus. In *Proceedings of the Second Symposium on Logic in Computer Science*, pages 30–36, Ithaca, USA, June 1987. IEEE Comp. Soc. Press.

[70] A. Middeldorp, editor. *12th International Conference on Rewriting Techniques and Applications*, volume 2051 of *LNCS*, Utrecht, The Netherlands, May 2001. Springer-Verlag.

[71] P. Narendran and M. Rusinowitch, editors. *10th International Conference on Rewriting Techniques and Applications*, volume 1631 of *LNCS*, Trento, Italy, July 1999. Springer-Verlag.

[72] Tobias Nipkow. Higher-order critical pairs. In *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science* [49], pages 342–349.

[73] Michael J. O'Donnell. *Computing in systems described by equations*, volume 58 of *LNCS*. Springer-Verlag, 1977.

[74] Mitsuhiro Okada. Strong normalizability for the combined system of the typed λ calculus and an arbitrary convergent term rewrite system. In G. H. Gonnet, editor, *Proceedings of the 1989 International Symposium on Symbolic and Algebraic Computation*, pages 357–363, Portland, Oregon, July 1989. ACM Press.

[75] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: The higher-order case. In *Logical Foundations of Computer Science*, volume 813 of *LNCS*, pages 379–392. Springer-Verlag, 1994.

[76] Christine Paulin-Mohring. Inductive definitions in the system Coq: Rules and properties. Research report RR 92-49, Laboratoire de l'Informatique du Parallélisme, Ecole normale supérieure de Lyon, December 1992.

[77] Christine Paulin-Mohring. Inductive definitions in the system Coq: Rules and properties. In M. Bezem and J. F. Groote, editors, *Proceedings of 1st International Conference on Typed Lambda Calculi and Applications*, volume 664 of *LNCS*, pages 328–345, Utrecht, The Netherlands, March 1993. Springer-Verlag.

[78] Christine Paulin-Mohring. *Définitions inductives en théorie des types d'ordre supérieur*. Thèse d'habilitation, Ecole Normale Supérieure de Lyon, 1996.

[79] Frank Pfenning and Christine Paulin-Mohring. Inductively defined types in the Calculus of Constructions. In *Proceedings of Mathematical Foundations of Programming Semantics*, volume 442 of *LNCS*. Springer-Verlag, 1990.

[80] Jaco van de Pol. Termination proofs for higher-order rewrite systems. In *Proceedings of HOA'93 (Higher Order Algebra)*, volume 816 of *LNCS*, pages 305–325, Amsterdam, Holland, 1994. Springer-Verlag.

[81] Jaco van de Pol. *Termination of Higher-Order Rewrite Systems.* PhD thesis, Utrecht University, 1996.

[82] Femke van Raamsdonk. *Confluence and Normalization for Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1996.

[83] Femke van Raamsdonk. On termination of higher-order rewriting. In Middeldorp [70], pages 261–275.

[84] George A. Robinson and Larry Wos. Paramodulation and theorem-proving in first-order theories with equality. In B. Meltzer, D. Michie, and M. Swann, editors, *Machine Intelligence 4*, pages 135–150. Edinburgh University Press, 1969.

[85] Richard Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9(9):73–81, 1979.

[86] Milena Stefanova. *Properties of Typing Systems*. PhD thesis, University of Nijmegen, 1999.

[87] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V7.3*, 2002.

[88] Jan Terlouw. Strong normalization in type systems: a model theoretic approach. *Annals of Pure and Applied Logic*, 73(2):53–78, 1995.

[89] Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1994.

[90] Daria Walukiewicz-Chrząszcz. Termination of rewriting in the calculus of constructions. *Journal of Functional Programming*. To appear.

[91] Daria Walukiewicz-Chrząszcz. Termination of rewriting in the calculus of constructions. In J. Despeyroux, editor, *Proceedings of the 2nd Workshop on Logical Frameworks and Meta-Languages*, Santa Barbara, California, 2000.

[92] Benjamin Werner. *Méta-théorie du Calcul des Constructions Inductives*. PhD thesis, Université Paris 7, 1994.