Université Paris XI
Orsay, France

Ph. D. Thesis

Type Theory and Rewriting

Frédéric BLANQUI

28 september 2001

Committee:

Mr Thierry COQUAND, referee
Mr Gilles DOWEK, president of the jury
Mr Herman GEUVERS, referee
Mr Jean-Pierre JOUANNAUD, supervisor
Mme Christine PAULIN
Mr Miklos SANTHA

The 19th of March 1998 was an important date for at least two reasons. The first one was personal. The second one was that Jean-Pierre Jouannaud agreed to supervise my master thesis on "extending the Calculus of Constructions with a new version of the General Schema" which he had roughed out with Mitsuhiro Okada. This did not mean much to me then. However, I was very happy with the idea of studying both λ-calculus and rewriting, and their interaction. This work results of this enthusiasm.

This is why I will begin by thanking Jean-Pierre Jouannaud, for the honor he made to me, the trust, the help, the advice and the support that he gave me during these three years. He taught me a lot and I will be always grateful to him.

I also thank Mitsuhiro Okada for the discussions we had together and the support he gave me. It was a great honor to have the opportunity to work with him. I hope we will have other numerous fruitful collaborations.

I also thank Maribel Fernández who helped me at the beginning of my thesis by supervising my work with Jean-Pierre Jouannaud.

I also thank Gilles Dowek who supported me in my work and helped me on several important occasions. His work was (and still is !) an important source of reflexion and inspiration.

I also thank Daria Walukiewicz with whom I had many fruitful discussions. I thank her very much for having read in detail an important part of this thesis and for having helped me to correct errors and lack of precision.

I also thank every person in the DÉMONS team from the LRI and the Coq team (newly baptized LogiCal) from INRIA Rocquencourt, in particular Christine Paulin and Claude Marché who helped me several times. These two teams are a privileged research place and have a pleasant atmosphere.

I also thank the referees of this thesis, Thierry Coquand and Herman Geuvers, for their interest in my work and the remarks they made for improving it.

Finally, I thank the members of the jury and the president of the jury for the honor they made to me by accepting to consider my work.

Contents

1 Introduction **7**
1.1 Some history 7
1.2 Motivations 17
1.3 Previous works 20
1.4 Contributions 22
1.5 Outline of the thesis 23

2 Preliminaries **25**

3 Type Systems Modulo (TSM's) **29**
3.1 Definition 30
3.2 Properties 33
3.3 TSM's stable by substitution 35
3.4 Logical TSM's 37

4 Reduction Type Systems (RTS's) **41**
4.1 Definition 41
4.2 Logical and functional RTS's 43
4.3 Logical and injective RTS's 45
4.4 Confluent RTS's 48

5 Algebraic Type Systems (ATS's) **51**

6 Conditions of Strong Normalization **57**
6.1 Term classes 57
6.2 Inductive types and constructors 58
6.3 General Schema 63
6.3.1 Higher-order rewriting 63
6.3.2 Definition of the schema 65
6.4 Strong normalization conditions 71

7 Examples of CAC's **77**
7.1 Calculus of Inductive Constructions (CIC) 77
7.2 CIC + Rewriting 87
7.3 Natural Deduction Modulo (NDM) 88

8 Correctness of the conditions **91**
8.1 Terms to be interpreted 91
8.2 Reductibility candidates 93
8.3 Interpretation schema 97
8.4 Interpretation of constant predicate symbols 102
8.5 Reductibility ordering 109
8.6 Interpretation of defined predicate symbols 111
8.6.1 Primitive systems 112
8.6.2 Positive, small and simple systems 112
8.6.3 Recursive, small and simple systems 112
8.7 Correctness of the conditions 113

9 Future directions of research **121**

Bibliography **123**

Index **133**

Chapter 1

Introduction

What is good programming? Apart from writing programs which are understandable and reusable by other people, above all it is being able to write programs without errors. But how do you know whether a program has no error? By proving it. In other words, to program well requires doing mathematics.

But how do you know whether a proof that a program has no error itself has no error? By writing a proof which can be checked by a computer. In other words, to program well requires doing formal mathematics.

This is the subject of our thesis : defining a formal system in which one can program and prove that a program is correct.

However, it is not the case that work is duplicated : programming and proving. In fact, from a proof that a program specification is correct, one can extract an error-free program! This is due to the termination of "cut elimination" in intuitionist logic discovered by G. Gentzen in 1933 [57].

More precisely, we will consider a particular class of formal systems, the type systems. We will study their properties when they are extended with definitions by rewriting. Rewriting is a simple and general computation paradigm based on rules like $x + 0 \rightarrow x$, that is, if one has an expression of the form $x + 0$, then one can simplify it to x.

But for such a system to serve in proving the correctness of programs, one must make sure that the system itself is correct, that is, that one cannot prove something which is false. This is why we will give conditions on the rewriting rules and prove that these conditions indeed ensure the correctness of the system.

First, let us see how type systems appeared, what results are already known and what our contributions are (they are summarized in Section 1.4).

1.1 Some history

This section is not intended to provide an absolutely rigorous historical summary. We only want to recall the basic concepts on which our work is based (type theory,

λ-calculus, etc.) and show how our work takes place in the continuation of previous works aiming at introducing more programming into logic, or dually, more logic into programming. We will therefore take some freedom with the formalisms used.

The reader familiar with these notions (in particular the Calculus of Constructions and the Calculus of Inductive Constructions) can directly go to Section 1.2 where we present our motivations for adding rewriting in the Calculus of Constructions both at the object-level and at the predicate-level.

Set theory

One of the first formal system enabling one to describe all mathematics was the *set theory* of E. Zermelo (1908) later extended by A. Fraenkel (1922). It was followed by the *type theory* of A. Whitehead and B. Russell (1911) [120], also called *higher-order logic*. These two formal systems were introduced to avoid the inconsistency of the *set theory* of G. Cantor (1878).

In first-order logic, in which the set theory of E. Zermelo and A. Fraenkel is generally expressed, the objects of the discourse are defined from constants and function symbols $(0, +, \ldots)$. Then, some predicate symbols $(\in, \ldots)$, the logical connectors $(\vee, \wedge, \Rightarrow, \ldots)$ and the universal and existential quantifiers $(\forall, \exists)$ enable one to express propositions in these objects.

One of the axioms of G. Cantor's set theory is the *Comprehension Axiom* which says that every proposition defines a set :

$$(\exists x)(\forall y)\ y \in x \Leftrightarrow P(y)$$

From this axiom, one can express Russell's paradox (1902). By taking $P(x) = x \notin x$, one can define the set R of the x's which do not belong to themselves. Then, $R \in R \Leftrightarrow R \notin R$ and one can deduce that any proposition is true. To avoid this problem, E. Zermelo proposed to restrict the Comprehension Axiom as follows :

$$(\forall z)(\exists x)(\forall y)\ y \in x \Leftrightarrow y \in z \wedge P(y)$$

that is, one can define by comprehension only subsets of previously well-defined sets.

Type theory

In type theory, instead of restricting the Comprehension Axiom, the idea is to forbid expressions like $x \notin x$ or $x \in x$ by restricting the application of a predicate to an object. To this end, one associates to each function symbol and predicate symbol (except $\in$) a *type* as follows :

- to a constant, one associate the type ι,
- to a function symbol taking one argument, one associates the type $\iota \rightarrow \iota$,
- to a function symbol taking two arguments, one associates the type $\iota \rightarrow \iota \rightarrow \iota$,
- ...
- to a proposition, one associates the type o,

– to a predicate symbol taking one argument, one associates the type $\iota \to o$,
– to a predicate symbol taking two arguments, one associates the type $\iota \to \iota \to o$,
– ...

Then, one can apply a function f taking n arguments to n objects $t_1, \dots, t_n$ if the type of f is $\iota \to \dots \to \iota \to \iota$ and every t_i is of type ι. And one can say that n objects $t_1, \dots, t_n$ satisfy a predicate symbol P taking n arguments if the type of P is $\iota \to \dots \to \iota \to o$ and every t_i is of type ι.

Finally, one considers that a set is not an object anymore (that is, an expression of type ι) but a predicate (an expression of type $\iota \to o$). And, for representing $x \in E$, which means that x satisfies E, one writes Ex (application of E to x). Hence, one can easily verify that it is not possible to express Russell's paradox : one cannot write xx for representing $x \in x$ since then x is both of the type $\iota \to o$ and of the type ι which is not allowed. In the following, we write $t : \tau$ for saying that t is of type τ.

Now, to represent natural numbers, there are several possibilities. However it is always necessary to state an axiom of infinity for ι and to be able to express the set of natural numbers as the smallest set containing zero and stable by incrementation. To this end, one must be able to quantify on sets, that is, on expressions of type $\iota \to o$.

Now, one is not restricted to objects and predicate expressions as described before, but can consider all the expressions that can be formed by applications which respect types :

– The set of the *simple types* is the smallest set T containing ι, o and $\sigma \to \tau$ whenever σ and τ belong to T.
– The set of *terms of type* τ is the smallest set containing the constants of type τ and the applications tu whenever t is a term of type $\sigma \to \tau$ and u a term of type σ.

Finally, one introduces an explicit notation for functions and sets, the λ-*abstraction*, and considers logical connectors and quantifiers as predicate symbols by giving them the following respective types : $\vee : o \to o \to o$, $\wedge : o \to o \to o$, $\forall_\tau : (\tau \to o) \to o$, ... For example, if ι denotes the set of natural numbers then one can represent the predicate "is even" (of type $\iota \to o$) by the expression $pair = \lambda x{:}\iota.\exists_\iota(\lambda y{:}\iota.x = 2 \times y)$ that we will abbreviate by $\lambda x{:}\iota.\exists y{:}\iota.x = 2 \times y$. The language we obtain is called the *simply-typed* λ-*calculus* $\lambda^{\to}$.

But what can we say about $(pair\,2)$ and $\exists y{:}\iota.2 = 2 \times y$? The second expression can be obtained from the first one by substituting x by 2 in the body of $pair$. This operation of substitution is called β-*reduction*. More generally, $\lambda x{:}\tau.\,t$ applied to u β-reduces to t where x is substituted by u : $\lambda x{:}\tau.\,t\ u \to_\beta t\{x \mapsto u\}$.

It is quite natural to consider theses two expressions as denoting the same proposition. This is why one adds the following *Conversion Axiom* :

$$P \Leftrightarrow Q \ \text{ if } \ P \to_\beta Q$$

One then gets the *type theory* of A. Church (1940) [30].

In this theory, it is possible to quantify over all propositions : $\forall P : o.P \Rightarrow P$. In other words, a proposition can be defined by quantifying over all propositions, including itself. If one allows such quantifications, the theory is said to be *impredicative*, otherwise it said *predicative*.

Mathematics as a programming language

The β-reduction corresponds to the evaluation process of a function. When one has a function f defined by an expression $f(x)$ and wants its value on 5 for example, one substitutes x by 5 in $f(x)$ and simplify the expression until one gets the value of $f(5)$.

One can wonder which functions are definable in Church's type theory. In fact, very few. With Peano's natural numbers (*i.e.* by taking $0 : \iota$ for zero and $s : \iota \to \iota$ for the successor function), one can express only constant functions or functions adding a constant to one of its arguments. With Church's numerals, where n is represented by $\lambda x\!:\!\iota.\lambda f\!:\!\iota \to \iota.f \ldots fx$ with n occurrences of f, H. Schwichtenberg [105] proved that one can express only extended polynomials (smallest set of functions closed under composition and containing the null function, the successor function, the projections, the addition, the multiplication and the test for zero).

Of course, it is possible to prove the existence of numerous functions, that is, to prove a proposition of the form $(\forall x)(\exists y)\, Pxy$ where P represents the graph of the function. In the intuitionist type theory for example (*i.e.* without using the Excluded-middle Axiom $P \vee \neg P$), it is possible to prove the existence of any primitive recursive function. But there is no term $f : \iota \to \iota$ enabling us to *compute* the powers of 2 for example, that is, such that $fn \to_\beta \ldots \to_\beta 2^n$.

Representation of proofs

G. Frege and D. Hilbert proposed to represent a proof of a proposition Q as a sequence $P_1, \ldots, P_n$ of propositions such that $P_n = Q$ and, for every i, either P_i is an axiom, either P_i is a consequence of the previous propositions by *modus ponens* (from P and $P \Rightarrow Q$ one can deduce Q) or by *generalization* (from $P(x)$ with x arbitrary one can deduce $(\forall x)P(x)$). However, to do such proofs, it is necessary to consider many axioms, independent of any theory, which express the sense of logical connectors.

Later, in 1933, G. Gentzen [57] proposed a new deduction system, called Natural Deduction, where logical axioms are replaced by *introduction rules* and *elimination rules* for the connectors and the quantifiers :

$$\text{(axiom)}\ \frac{}{\Gamma, P, \Gamma' \vdash P}$$

$$(\wedge\text{-intro})\ \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q} \qquad (\wedge\text{-élim1})\ \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P} \qquad (\wedge\text{-élim2})\ \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}$$

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q} \qquad (\Rightarrow\text{-élim})\ \frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}$$

$$(\exists\text{-intro})\ \frac{\Gamma \vdash P(t)}{\Gamma \vdash (\exists x)P(x)} \qquad (\exists\text{-élim})^1\ \frac{\Gamma \vdash (\exists x)P \quad \Gamma, P \vdash Q}{\Gamma \vdash Q} \qquad \ldots$$

where Γ is a set of propositions (the hypothesis). A pair $\Gamma \vdash Q$ made of a set of hypothesis Γ and a proposition Q is called a *sequent*. Then, a proof of a sequent $\Gamma \vdash Q$ is a tree whose root is $\Gamma \vdash Q$, whose nodes are instances of the deduction rules and whose leaves are applications of the rule (axiom).

Cut elimination

G. Gentzen remarked that some proofs can be simplified. For example, this proof of Q :

$$\frac{\dfrac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}\,(\Rightarrow\text{-intro}) \qquad \Gamma \vdash P}{\Gamma \vdash Q}\,(\Rightarrow\text{-élim})$$

does a detour which can be eliminated. It suffices to replace in the proof of $\Gamma, P \vdash Q$ all the leaves (axiom) giving $\Gamma, P, \Gamma' \vdash P$ (Γ' are additional hypothesis that may be introduced for proving Q) by the proof of $\Gamma \vdash P$ where Γ is also replaced by Γ, Γ'. In fact, at every place where there is a *cut*, that is, an introduction rule followed by an elimination rule for the same connector, it is possible to simplify the proof. G. Gentzen proved the following remarkable fact : the cut-elimination process terminates.

Hence, any provable proposition has a cut-free proof. But, in intuitionist logic, any cut-free proof of a proposition $(\exists x)P(x)$ must terminate by an introduction rule whose premise is of the form $P(t)$. Therefore, the cut-elimination process gives us a witness t of an existential proposition. In other words, any function whose existence is provable is computable.

If one can express the proofs themselves as objects of the theory, then it becomes possible to express many more functions than those allowed in the simply-typed λ-calculus.

The isomorphism of Curry-de Bruijn-Howard

In 1958, Curry [41] remarked that there is a correspondence between the types of the simply-typed λ-calculus and the propositions formed from the implication $\Rightarrow$ (one can identify $\rightarrow$ and $\Rightarrow$), and also between the terms of type τ and the proofs of the proposition corresponding to τ. In other words, the simply-typed λ-calculus enables one to represent the proofs of the minimal propositional logic. To this end, one associates to each proposition P a variable x_P of type P. Then, one defines the λ-term associated to a proof by induction on the size of the proof :

[1] If x does not occur neither in Γ nor in Q.

- the proof of $\Gamma \vdash P$ obtained by (axiom) is associated to x_P;
- the proof of $\Gamma \vdash P \Rightarrow Q$ obtained by ($\Rightarrow$-intro) from a proof π of $\Gamma, P \vdash Q$ is associated to the term $\lambda x_P : P.\, t$ where t is the term associated to π;
- the proof of $\Gamma \vdash Q$ obtained by ($\Rightarrow$-elim) from a proof π of $\Gamma \vdash P \Rightarrow Q$ and a proof π' of $\Gamma \vdash P$ is associated to the term tu where t is the term associated to π and u the term associated to π'.

The set of λ-terms that we obtain can be directly defined as follows. We call an *environment* any set Γ of pairs $x : P$ made of a variable x and a type P (representing a proposition). Then, a term t is of type P (a proof of P) in the environment Γ (under the hypothesis Γ) if $\Gamma \vdash t : T$ can be deduced by the following inference rules :

$$\text{(axiom)}\ \frac{}{\Gamma, x : P, \Gamma' \vdash x : P}$$

$$(\Rightarrow\text{-intro})\ \frac{\Gamma, x : P \vdash t : Q}{\Gamma \vdash \lambda x : P.\, t : P \Rightarrow Q}$$

$$(\Rightarrow\text{-elim})\ \frac{\Gamma \vdash t : P \Rightarrow Q \quad \Gamma \vdash u : P}{\Gamma \vdash tu : Q}$$

In 1965, W. W. Tait [110] remarked that β-reduction corresponds to cut-elimination. Indeed, if one annotates the example of cut previously given then one gets :

$$\frac{\dfrac{\Gamma, x : P \vdash t : Q}{\Gamma \vdash \lambda x : P.\, t : P \Rightarrow Q}\,(\Rightarrow\text{-intro}) \qquad \Gamma \vdash u : P}{\Gamma \vdash \lambda x : P.\, t\ u : Q}\,(\Rightarrow\text{-elim})$$

If one β-reduces $\lambda x : P.\, t\ u$ to $t\{x \mapsto u\}$, then one exactly obtains the term corresponding to the cut-free proof of $\Gamma \vdash Q$. Hence, the existence of a cut-free proof corresponds to the *weak normalization* of β-reduction, that is, the existence for any typable λ-term t of a sequence of β-reductions resulting in a β-irreducible term (we also say in *normal form*). This is why normalization has such an important place in the study of type systems.

In 1968, N. de Bruijn [42] proposed a system of *dependent types* extended the simply-typed λ-calculus and in which it was possible to express the propositions and the proofs of intuitionist first-order logic. This system was the basis of one of the first programs for doing formal proofs : AUTOMATH. A dependent type is simply a function which associates a type expression to each object. It enables one to represent predicates and quantifiers. In 1969, W. A. Howard [69] considered a similar system but without considering it as a logical system in its own right.

In a dependent type system, the well-formedness of types depends on the well-formedness of terms. It is then necessary to consider environments with type variables and to add typing rules for types and environments (the order of variables now matters). Finally, it is necessary to add a conversion rule for identifying β-equivalent propositions. One then gets a set of typing rules similar to the ones of Figure 1.1 (this is a modern presentation which emerged at the end of the 80's only).
centering

Figure 1.1: Typing rules of λP

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

(var) $$\frac{\Gamma \vdash T : s \in \{\star, \Box\}}{\Gamma, x\!:\!T \vdash x : T}$$

(weak) $$\frac{\Gamma \vdash t : T \quad \Gamma \vdash U : s \in \{\star, \Box\}}{\Gamma, x\!:\!U \vdash t : T}$$

(prod-$\lambda^{\rightarrow}$) $$\frac{\Gamma \vdash T : \star \quad \Gamma, x\!:\!T \vdash U : \star}{\Gamma \vdash (x\!:\!T)U : \star}$$

(prod-λP) $$\frac{\Gamma \vdash T : \star \quad \Gamma, x\!:\!T \vdash U : \Box}{\Gamma \vdash (x\!:\!T)U : \Box}$$

(abs) $$\frac{\Gamma, x\!:\!T \vdash u : U \quad \Gamma \vdash (x\!:\!T)U : s \in \{\star, \Box\}}{\Gamma \vdash \lambda x\!:\!T.\, u : (x\!:\!T)U}$$

(app) $$\frac{\Gamma \vdash t : (x\!:\!U)V \quad \Gamma \vdash u : U}{\Gamma \vdash tu : V\{x \mapsto u\}}$$

(conv) $$\frac{\Gamma \vdash t : T \quad T \leftrightarrow^*_\beta T' \quad \Gamma \vdash T' : \star}{\Gamma \vdash t : T'}$$

In this system, $\star$ is the type of propositions and of the sets of the discourse (natural numbers, etc.), and $\Box$ is the type of predicate types (of which $\star$ is). For example, the set of natural numbers *nat* has the type $\star$, the predicate *even* has the type $(n : nat)\star$ that we abbreviate by $nat \rightarrow \star$ since n does not occur in $\star$ (non-dependent product) and $nat \rightarrow \star$ has the type $\Box$. Starting from the rule (ax), the rules (var) and (weak) enables one to build environments. The rule (prod-$\lambda^{\rightarrow}$) enables one to build propositions and the rule (prod-λP) enables one to build predicate types. In the case of a proposition, if the product is not dependent (x does not occur in U) then it is an implication, otherwise it is a universal quantification. In other words, without the rule (prod-λP), we get the simply-typed λ-calculus. The rule (abs) enables one to build a function (if $s = \star$) or a predicate (if $s = \Box$). Finally, the rule (app) enables the application of a function or a predicate to an argument. In other words, the rules (abs) and (app) generalize the rules (⇒-intro) and (⇒-elim) of the simply-typed λ-calculus.

From the point of view of programming, dependent types enables one to have more information about data and hence to reduce the risk of error. For example, one can define the type ($list\ n$) of lists of natural numbers of length n by declaring $list : nat \to \star$. Then, the empty list nil has the ($list$ 0) and the function $cons$ which adds a natural number x at the head of a list ℓ of length n has the type $nat \to (n\!:\!nat)(list\ n) \to (list\ (s\ n))$. One can then verify if a list does not exceed some given length.

Inductive definitions

In higher-order logic, the induction principle for natural numbers can be proved only if natural numbers are impredicatively defined. In other words, if one prefers to stay in a predicative framework, it is necessary to state the induction principle for natural numbers as an axiom.

This is why, in 1971, P. Martin-Löf [84] extended the calculus of N. de Bruijn by including expressions for representing inductive types and their induction principles. For example, the type of natural numbers is represented by the symbol $nat : \star$, zero by $0 : nat$, the successor function by $s : nat \to nat$ and a proof by induction for a predicate $P : nat \to o$ by $rec^P : P0 \to (\forall n\!:\!nat.Pn \to Ps(n)) \to \forall n\!:\!nat.Pn$. In the conversion rule (conv), to the β-reduction, P. Martin-Löf adds the *ι-reduction* which corresponds to the elimination of induction cuts :

$$\text{(conv)} \qquad \frac{\Gamma \vdash t : T \quad T \leftrightarrow^*_{\beta\iota} T' \quad \Gamma \vdash T' : \star}{\Gamma \vdash t : T'}$$

In the case of nat, the rules defining the ι-reduction are those of K. Gödel's System T [65] :

$$\begin{array}{rcl} rec^P(p_0, p_s, 0) & \to_\iota & p_0 \\ rec^P(p_0, p_s, s(n)) & \to_\iota & p_s\ n\ rec^P(p_0, p_s, n) \end{array}$$

where p_0 is a proof of $P0$ and p_s a proof of $\forall n\!:\!nat.Pn \to Ps(n)$. From these rules, by taking $P = \lambda x\!:\!nat.nat$, it is possible to define functions on natural numbers like the addition and the multiplication :

$$\begin{array}{rcl} x + y & = & rec^P(y, \lambda u.\lambda v.s(v), x) \\ x \times y & = & rec^P(0, \lambda u.\lambda v.v + y, x) \end{array}$$

To convince oneself, let $f = \lambda u.\lambda v.s(v)$ and let us show that $2+2$ rewrites to 4 : $2 + 2 = rec(2, f, 2) \to_\iota f\,2\ rec(2, f, 1) \to_\beta s(rec(2, f, 1)) \to_\iota s(f\,2\ rec(2, f, 0)) \to_\beta s(s(rec(2, f, 0))) \to_\iota s(s(2)) = 4$.

In fact, in this theory, it is possible to express by a term any function whose existence is provable in predicative intuitionist higher-order arithmetic (and these functions are also those that are expressible in K. Gödel's System T).

Polymorphism

The termination problem of cut-elimination in intuitionist impredicative higher-order arithmetic was solved by J.-Y. Girard [63] in 1971. To this end, he introduced a *polymorphic* type system Fω (J. Reynolds [103] introduced independently a similar system for second-order quantifications only). A polymorphic type is a function which, to a type expression, associates a type expression. And for representing the proofs of impredicative propositions, one also needs the terms themselves to be polymorphic, that is, it must be possible for a term to be applied to a type expression. Formally, for second-order quantifications (*i.e.* on propositions), this requires the replacement of the rule (prod-λP) in Figure 1.1 by the rule :

$$\text{(prod-F)} \qquad \frac{\Gamma \vdash T : \Box \quad \Gamma, x\!:\!T \vdash U : \star}{\Gamma \vdash (x\!:\!T)U : \star}$$

which for example allows one to build the type $(P\!:\!\star)P \to P$ corresponding to the proposition $\forall P : o.P \Rightarrow P$ in higher-order logic. For higher-order quantifications, one must add the following rule :

$$\text{(prod-F}\omega\text{)} \qquad \frac{\Gamma \vdash T : \Box \quad \Gamma, x\!:\!T \vdash U : \Box}{\Gamma \vdash (x\!:\!T)U : \Box}$$

which allows the formation of predicate types like for example $\star \to \star$ which corresponds to $o \Rightarrow o$ in higher-order logic.

In this system, it is then possible to express any function whose existence is provable in impredicative intuitionist higher-order arithmetic.

From the point of view of programming, polymorphism enables one to formalize generic algorithms with respect to data types. For example, one can speak of the type $(list\ A)$ of lists of elements of type A, for any type A, by declaring $list : \star \to \star$.

In 1984, T. Coquand and G. Huet [37] defined a system, the Calculus of Constructions (CC), which makes the synthesis of the systems of N. de Bruijn and J.-Y. Girard (it contains all the product-formation rules we have seen) and in which it is then possible to express all the higher-order logic (but it does not allow to express more functions than Fω). This system served as a basis for the proof assistant Coq [112].

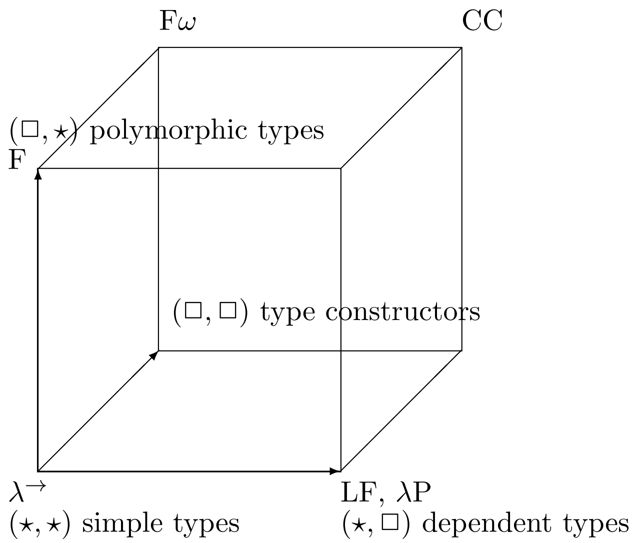
Pure Types Systems (PTS)

At the end of the 80's, H. Barendregt [9] remarked that many type systems ($\lambda^{\to}$, λP, F, Fω, CC, etc.) can be characterized by their product-formation rules. This led to the presentation we adopted here. By considering the following general typing rule parametrized by two *sorts* $s_1, s_2 \in \{\star, \Box\}$:

$$(s_1, s_2) \qquad \frac{\Gamma \vdash T : s_1 \quad \Gamma, x\!:\!T \vdash U : s_2}{\Gamma \vdash (x\!:\!T)U : s_2}$$

it is possible to have 4 different rules (($\star,\star$) corresponds to (prod-$\lambda^{\rightarrow}$), ($\star,\Box$) to (prod-λP), ($\Box,\star$) to (prod-F), and ($\Box,\Box$) to (prod-Fω)) and hence to have, from ($\star,\star$), 8 different systems that one can organize in a cube whose directions correspond to the presence or the absence of dependent types (rule ($\star,\Box$)), polymorphic types (rule ($\Box,\star$)) or type constructors (rule ($\Box,\Box$)) (see Figure 1.2).

Figure 1.2: Barendregt's Cube



- $\lambda^{\rightarrow}$ denotes the simply-typed λ-calculus of A. Church [30],
- LF (Logical Framework) denotes the system of R. Harper, F. Honsell and G. Plotkin [68],
- λP denote the AUTOMATH system of N. de Bruijn [42],
- F and Fω respectively denote the second-order and higher-order polymorphic λ-calculus of J.-Y. Girard [64],
- CC denotes the Calculus of Constructions of T. Coquand and G. Huet [38].

This led S. Berardi [19] and J. Terlouw [113] to a more systematic study of type systems with respect to expressible types until the definition by H. Geuvers and M.-J. Nederhof of the *Pure Type Systems* (PTS) [59] which are type systems parametrized by :

- a set of *sorts* $\mathcal{S}$ representing the different universes of discourse ($\{\star,\Box\}$ in the Cube),
- a set of *axioms* $\mathcal{A} \subseteq \mathcal{S}^2$ representing how these universes are included in one another ($\{(\star,\Box)\}$ in the Cube) and the rule :

$$\text{(ax)} \quad \frac{}{\vdash s_1 : s_2} \quad ((s_1,s_2) \in \mathcal{A})$$

- a set of product-formation *rules* $\mathcal{B} \subseteq \mathcal{S}^3$ representing the possible quantifications ($\{(\star,\star,\star),(\star,\Box,\Box),(\Box,\star,\star),(\Box,\Box,\Box)\}$ in the Cube) and the rule :

$$\text{(prod)} \quad \frac{\Gamma \vdash T : s_1 \quad \Gamma, x\!:\!T \vdash U : s_2}{\Gamma \vdash (x\!:\!T)U : s_3} \quad ((s_1, s_2, s_3) \in \mathcal{B})$$

Calculus of Inductive Constructions (CIC)

We have seen that the Calculus of Constructions is a very powerful system in which it is possible to express many functions. However, these functions cannot be defined as one would like. For example, it does not seem possible to program the predecessor function on natural numbers such that its evaluation takes a constant time [64]. This is not the case in P. Martin-Löf's system where natural numbers and their induction principle are first-class objects while, in the Calculus of Constructions, natural numbers are impredicatively defined.

This is why, in 1988, T. Coquand and C. Paulin proposed the Calculus of Inductive Constructions (CIC) [39] which makes the synthesis between the Calculus of Constructions and P. Martin-Löf's type theory, and hence enable one to write more efficient programs. In 1994, B. Werner [119] proved the termination of cut-elimination in this system. (In 1993, T. Altenkirch [2] proved also this property but for a presentation of the calculus with equality judgments.)

But, even in this system, some algorithms may be inexpressible. L. Colson [32] proved for example that, if one uses a call-by-value evaluation strategy (reduction of the arguments first) then the minimum function of two natural numbers cannot be implemented by a program whose evaluation time is relative to the minimum of the two arguments.

1.2 Motivations

We said at the beginning that rewriting is a simple and general computation paradigm based on *rewrite rules*. This notion is of course very old but it was seriously studied from the 70's with the works of D. Knuth and D. Bendix [18]. They studied rewriting for knowing whether, in a given equational theory, an equation is valid or not. Then, rewriting was quickly used as a programming paradigm [96, 67, 55, 71, 31] since any computable function can be defined by rewrite rules (Turing-completeness).

Let us see the case of addition and multiplication on natural numbers defined from 0 for zero and s for the successor function :

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

These rules completely define these two arithmetic operations : starting from two arbitrary natural numbers p and q (expressed with 0 and s), $p + q$ and $p \times q$ rewrite in a finite number of steps to a term which cannot be further rewritten, that is, to a number representing the value of $p + q$ and $p \times q$ respectively.

Higher-order rewriting

We can also imagine definitions using functional parameters or abstractions : this is *higher-order* rewriting as opposed to *first-order* rewriting which does not allow functional parameters or abstractions. For example, the function *map* which to a function f and a list of natural numbers $(a_1, \ldots, a_n)$ associates the list $(f(a_1), \ldots, f(a_n))$, can be defined by the following rules :

$$\begin{array}{rcl} map(f, nil) & \rightarrow & nil \\ map(f, cons(x, \ell)) & \rightarrow & cons(fx, map(f, \ell)) \end{array}$$

where *nil* stand for the empty list and *cons* for the function which adds an element at the head of a list.

Hence, the rules defining the recursor of an inductive type (the ι-reduction) are a particular case of higher-order rewriting.

Easier definitions

One can see that definitions by rewriting are more natural and easier to write than the ones based on recursors like in P. Martin-Löf type theory or in the Calculus of Inductive Constructions. For example, the definition by recursion of the function $\leq$ on natural numbers requires two levels of recursion :

$$\lambda x.rec(x, \lambda y.true, \lambda nzy.rec(y, false, \lambda n'z'.zn', y))$$

while the definition by rewriting is :

$$\begin{array}{rcl} 0 \leq y & \rightarrow & true \\ s(x) \leq 0 & \rightarrow & false \\ s(x) \leq s(y) & \rightarrow & x \leq y \end{array}$$

More efficient definitions

From a computing point of view, definitions by rewriting can be made more efficient by adding rules. For example, with a definition by recursion on its first argument, $n+0$ requires $n+1$ reduction steps. By simply adding the rule $x+0 \rightarrow x$, this takes only one step.

However, it can become more difficult to ensure that, for any sequence of arguments, the definition always leads, in a finite number of steps (property called *strong normalization*), to a unique result (property called *confluence*) that we call the *normal form* of the starting expression.

Quotient types

Until now we have always spoken of natural numbers but never of integers. Yet they have an important place in mathematics. One way to represent integers is to add a predecessor function p beside 0 and s. Hence, $p(p(0))$ represents -2. Unfortunately, in this case, an integer can have several representations : $p(s(0))$ or $s(p(0))$ represent both 0. In fact, integers are equivalent modulo the equations $p(s(x)) = x$ and $s(p(x)) = x$.

However, it is possible to orient these equations so as to have a confluent and strongly normalizing rewrite system : $p(s(x)) \rightarrow x$ and $s(p(x)) \rightarrow x$. Then, each integer has a unique normal form. Therefore, we see that rewriting enables us to models quotient types without using further extensions [14].

More typable terms

The introduction of rewriting in a dependent type system allows one to type more terms and therefore to formalize more propositions. Let us consider, in the Calculus of Inductive Constructions, the type $listn : nat \rightarrow \star$ of lists of natural numbers of length n with the constructors $niln : listn(0)$ for the empty list and $consn : nat \rightarrow (n : nat)listn(n) \rightarrow listn(s(n))$ for adding an element at the head of a list. Let $appn : (n : nat)listn(n) \rightarrow (n' : nat)listn(n') \rightarrow listn(n + n')$ be the concatenation function. Like $+$, $appn$ can be defined by using the recursor associated to the type $listn$. Assume furthermore that $+$ and $appn$ are defined by induction on their first argument. Then, the following propositions are not typable :

$$\begin{array}{rcl} appn(n, \ell, 0, \ell') & = & \ell \\ appn(n + n', appn(n, \ell, n', \ell'), n'', \ell'') & = & appn(n, \ell, n' + n'', appn(n', \ell', n'', \ell'')) \end{array}$$

In the first rule, the left hand-side is of type $listn(n+0)$ and the right hand-side is of type $listn(n)$. We can prove that $(n : nat)n + 0 = n$ by induction on n but $n+0$ is not $\beta\iota$-convertible to n since $+$ is defined by induction on its first argument. Therefore, we cannot apply the (conv) rule for typing the equality.

In the second rule, the left hand-side is of type $listn((n + n') + n'')$ and the right hand-side is of type $listn(n + (n' + n''))$. Again, although we can prove that $(n+n')+n'' = n+(n'+n'')$ (associativity of $+$), the two terms are not $\beta\iota$-convertible. Therefore, we cannot apply the (conv) rule for typing the equality.

This shows some limitation of the definitions by recursion. The use of rewriting, that is, the replacement in the (conv) rule of the ι-reduction by a reduction relation $\rightarrow_{\mathcal{R}}$ generated from a user-defined set $\mathcal{R}$ of arbitrary rewrite rules :

$$\text{(conv)} \quad \frac{\Gamma \vdash t : T \quad T \leftrightarrow^*_{\beta\mathcal{R}} T' \quad \Gamma \vdash T' : \star}{\Gamma \vdash t : T'},$$

allows us to type the previous propositions which are not typable in the Calculus of Inductive Constructions.

Automatic equational proofs

Another motivation for introducing rewriting in type systems is that it makes equational proofs much easier, which is the reason why rewriting was studied initially. Indeed, in the case of a confluent and strongly normalizing rewrite system, to check whether two terms are equal, it suffices to check whether they have the same normal form.

Moreover, it is not necessary to keep a trace of the rewriting steps since this computation can be done again (if the equality is decidable). This reduces the size of proof-terms and enable us to deal with bigger proofs, which is a critical problem now in proof assistants.

Integration of decision procedures

One can also imagine defining predicates by rewriting or having simplification rules on propositions, hence generalizing the definitions by *strong elimination* of the Calculus of Inductive Constructions [99]. For example, one can consider the set of rules of Figure 1.3 [70] where xor (exclusive "or") and $\wedge$ are commutative and associative symbols, $\bot$ represents the proposition always false and $\top$ the proposition always true (by taking a constant I of type $\top$).

Figure 1.3: Decision procedure for classical propositional tautologies

$$\begin{array}{rcl} P \,\mathrm{xor}\, \bot & \rightarrow & P \\ P \,\mathrm{xor}\, P & \rightarrow & \bot \\ P \wedge \top & \rightarrow & P \\ P \wedge \bot & \rightarrow & \bot \\ P \wedge P & \rightarrow & P \\ P \wedge (Q \,\mathrm{xor}\, R) & \rightarrow & (P \wedge Q) \,\mathrm{xor}\, (P \wedge R) \end{array} \qquad \begin{array}{rcl} \neg P & \rightarrow & P \,\mathrm{xor}\, \top \\ P \vee Q & \rightarrow & (P \wedge Q) \,\mathrm{xor}\, P \,\mathrm{xor}\, Q \\ P \Rightarrow Q & \rightarrow & (P \wedge Q) \,\mathrm{xor}\, P \,\mathrm{xor}\, \top \\ P \Leftrightarrow Q & \rightarrow & (P \,\mathrm{xor}\, Q) \,\mathrm{xor}\, \top \end{array}$$

J. Hsiang [70] showed that this system is confluent and strongly normalizing and that a proposition P is a tautology (*i.e.* is always true) if P reduces to $\top$. This system is therefore a decision procedure for classical propositional tautologies.

Hence, type-level rewriting allows the integration of decision procedures. Indeed, thanks to the conversion rule (conv), if P is a tautology then I, the canonical proof of $\top$, is a proof of P. In other words, if the typing relation is decidable, to know whether a proposition P is a tautology, it is sufficient to propose I to the verification program.

We can also imagine simplification rules for equality like the ones of Figure 1.4 where $+$ and $\times$ are associative and commutative, and $=$ is commutative.

Figure 1.4: Simplification rules for equality

$$\begin{array}{rcl} x+0 & \rightarrow & x \\ x+s(y) & \rightarrow & s(x+y) \\ x \times 0 & \rightarrow & 0 \\ x \times s(y) & \rightarrow & (x \times y)+x \\ x \times (y+z) & \rightarrow & (x \times y)+(x \times z) \end{array} \qquad \begin{array}{rcl} x=x & \rightarrow & \top \\ s(x)=s(y) & \rightarrow & x=y \\ s(x)=0 & \rightarrow & \bot \\ x+y=0 & \rightarrow & x=0 \wedge y=0 \\ x \times y=0 & \rightarrow & x=0 \vee y=0 \end{array}$$

1.3 Previous works

The first works on the combination of typed λ-calculus and (first-order) rewriting were due to V. Breazu-Tannen in 1988 [25]. They showed that the combination of simply-typed λ-calculus and first-order rewriting is confluent if the rewriting is confluent. In 1989, V. Breazu-Tannen and J. Gallier [26], and M. Okada [97] independently, showed that the strong normalization is also preserved. These results were extended by D. Dougherty [47] to any "stable" set of pure λ-terms.

In 1991, J.-P. Jouannaud and M. Okada [74] extended the result of V. Breazu-Tannen and J. Gallier to higher-order rewrite systems satisfying the *General Schema*, a generalization of the primitive recursion schema. With higher-order rewriting, strong normalization becomes more difficult to prove since there is a strong interaction between rewriting and β-reduction, which is not the case with first-order rewriting.

In 1993, M. Fernández [54] extended this method to the Calculus of Constructions with object level rewriting and simply typed symbols. The methods used for first-order rewriting and non dependent systems [26, 47] cannot be applied to this case since rewriting is not just a syntactic addition : since rewriting is included in the type conversion rule (conv), it is a component of typing (in particular, it allows more terms to be typed).

Other methods for proving strong normalization appeared. In 1996, J. van de Pol [116] extended to the simply-typed λ-calculus the use of monotone interpretations. In 1999, J.-P. Jouannaud and A. Rubio [76] extended to the simply-typed λ-calculus the method RPO (Recursive Path Ordering) [100, 44]. This method (HORPO) is more powerful than the General Schema since it is a recursively defined ordering.

In all these works, even the ones on the Calculus of Constructions, function symbols are always simply typed. It was T. Coquand [34] in 1992 who initiated the study of rewriting on dependent and polymorphic symbols. He studied the completeness of definitions on dependent types. For the strong normalization, he proposed a schema more general than the schema of J.-P. Jouannaud and M. Okada since it allows recursive definitions on strictly positive types [39] but it does not necessary imply strong normalization. In 1996, E. Giménez [62] defined a restriction of this schema for which he proved strong normalization. In 1999, J.-P. Jouannaud, M. Okada and I [23, 22] extended the General Schema, keeping simply typed symbols, in order to deal with strictly positive types. Finally, in 2000, D. Walukiewicz [118] extended J.-P. Jouannaud and A. Rubio's HORPO to the Calculus of Constructions with dependent and polymorphic symbols.

But there still is a common point between all these works : rewriting is always confined to the object level.

In 1998, G. Dowek, T. Hardin and C. Kirchner [50] proposed a new approach to deduction for first-order logic : the *Natural Deduction Modulo* (NDM) a congruence $\equiv$ on propositions. This deduction system consists of replacing the rules of usual Natural Deduction by rules equivalent modulo $\equiv$. For example, the elimination rule for $\Rightarrow$ (*modus ponens*)is replaced by :

$$(\Rightarrow\text{-elim-modulo}) \qquad \frac{\Gamma \vdash R \quad \Gamma \vdash P}{\Gamma \vdash Q} \quad \text{if } R \equiv P \wedge Q$$

They proved that the simple theory of types and the skolemized set theory can be seen as first-order theory modulo some congruences using *explicit substitutions*. In [51], G. Dowek and B. Werner gave several conditions ensuring the strong normalization of cut elimination.

1.4 Contributions

Our main contribution was establishing very general conditions for ensuring the strong normalization of the Calculus of Constructions extended with type level rewriting [20]. We showed that our conditions are satisfied by a large subsystem of the Calculus of Inductive Constructions (CIC) and by Natural Deduction Modulo (NDM) a large class of equational theories.

Our work can be seen as an extension of both the Natural Deduction Modulo and the Calculus of Constructions, where the congruence not only includes first-order rewriting but also higher-order rewriting since, in the Calculus of Constructions, functions and predicates can be applied to functions and predicates.

It can therefore serve as the basis of a powerful extension of proof assistants like Coq [112] and LEGO [82] which allow definitions by recursion only. Indeed, strong normalization not only ensures the logical consistency (if the symbols are consistent) but also the decidability of type checking, that is, the verification that a term is the proof of a proposition.

For deciding particular classes of problems, it may be more efficient to use specialized rewriting-based applications like CiME [33], ELAN [24] or Maude [31]. Furthermore, for program extraction [98], we can use rewriting-based languages and hence get more efficient extracted programs.

To consider type-level rewriting is not completely new : a particular case is the "strong elimination" of the Calculus of Inductive Constructions, that is, the ability to define predicates by induction on some inductive data type. The main novelty here is to consider any set of user-defined rewrite rules.

The strong normalization proofs with strong elimination of B. Werner [119] and T. Altenkirch [2] use in an essential way the fact that the definitions are inductive.

Moreover, the methods used in case of first-order rewriting [26, 4, 47] cannot be applied here. Firstly, we consider higher-order rewriting which has a strong interaction with β-reduction. Secondly, rewriting is part of the type conversion rule, which implies that more terms are typable.

For establishing our conditions and proving their correctness, we have adapted the method of reductibility candidates of Tait and Girard [64] also used by F. Barbanera, M. Fernández and H. Geuvers [7, 6, 5] for object-level rewriting and by B. Werner and T. Altenkirch for strong elimination. As candidates, they all use sets of pure (untyped) λ-terms and, except T. Altenkirch, they all use intermediate languages of type systems. By using a work by T. Coquand and J. Gallier [36], we use candidates made of well-typed terms and do not use intermediate languages. We therefore get a simpler and shorter proof for a more general result.

We also mention other contributions.

For allowing quotient types (rules on constructors) and matching on function symbols, which is not possible in the Calculus of Inductive Constructions, we use a notion of "constructor" more general than the usual one (see Subsection 6.2).

For ensuring the subject reduction property, that is, the preservation of typing under reduction, we introduce new conditions more general than the ones previously used. In particular, these conditions allow us to get rid of many non-linearities due to typing, which makes rewriting more efficient and confluence easier to prove.

1.5 Outline of the thesis

Chapter 3 : We study the basic properties of Pure Type Systems whose type conversion relation is abstract. We call such a system a Type System Modulo (TSM).

Chapter 4 : We study the properties of a particular class of TSM's, those whose conversion relation is generated from a reduction relation. We call such a system a Reduction Type System (RTS). An essential problem in these systems is to make sure that the reduction relation preserves typing (subject reduction property).

Chapter 5 : We give sufficient conditions for ensuring the subject reduction property in RTS's whose reduction relation is generated from rewrite rules. We call such a system an Algebraic Type System (ATS).

Chapter 6 : In this chapter and the following ones, we consider a particular ATS, the Calculus of Algebraic Constructions (CAC). We give sufficient conditions for ensuring its strong normalization.

Chapter 7 : We give important examples of type systems satisfying our strong normalization conditions. Among these systems, we find a sub-system with strong elimination of the Calculus of Inductive Constructions (CIC) which is the basis of the proof assistant Coq [112]. We also find Natural Deduction Modulo (NDM) a large class of equational theories.

Chapter 8 : We prove the correctness of our strong normalization conditions and clearly indicate which conditions are used. An index enables one to find where each conditions is used.

Chapter 9 : We finish by enumerating several directions for future research which could improve or extend our strong normalization conditions.

Chapter 2

Preliminaries

In this chapter, we define the syntax of the systems we will study and recall a few elementary notions about λ-calculus, Pure Type Systems (PTS) (see [10] for more details) and relations. This syntax simply extends the syntax of PTS's by adding symbols (nat, 0, $+$, $\geq$, ...) which must be applied to as many arguments as are required by their specified *arity* (see Remark 10 for a discussion about this notion).

Definition 1 (Sorted λ-systems) A *sorted λ-system* is given by :

- a set of *sorts* $\mathcal{S}$,
- a family $\mathcal{F} = (\mathcal{F}_n^s)_{n\geq 0}^{s\in\mathcal{S}}$ of sets of *symbols*,
- a family $\mathcal{X} = (\mathcal{X}^s)^{s\in\mathcal{S}}$ of infinite denumerable sets of *variables*,

such that all sets are disjoint. A symbol $f \in \mathcal{F}_n^s$ is of *arity* $\alpha_f = n$ and of *sort* s. We will denote the set of symbols of sort s by $\mathcal{F}^s$ and the set of symbols of arity n by $\mathcal{F}_n$.

Definition 2 (Terms) The set $\mathcal{T}$ of *terms* is the smallest set such that :

- sorts and variables are terms;
- if x is a variable and t and u are terms then the *dependent product* $(x\!:\!t)u$ and the *abstraction* $[x\!:\!t]u$ are terms;
- if t and u are terms then the *application* tu is a term;
- if f is a symbol of arity n and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ is a term (some binary symbols like $+$, $\times$, ... will sometimes be written infix).

Free and bound variables

A variable x in the scope of an abstraction $[x\!:\!T]$ or a product $(x\!:\!T)$ is *bound*. As usual, it may be replaced by another variable of the same sort. This is *α-equivalence*. A variable which is not bound is *free*. We denote by $\mathrm{FV}(t)$ the set of free variables of a term t and by $\mathrm{FV}^s(t)$ the set of free variables of sort s. A term without free variables is *closed*. We often denote by $U \to V$ a product $(x\!:\!U)V$ such that $x \notin \mathrm{FV}(V)$ (non dependent product).

Vectors

We will often use vectors ($\vec{t}, \vec{u}, \ldots$) for sequences of terms (or anything else). The size of a vector $\vec{t}$ is denoted by $|\vec{t}|$. For example, $[\vec{x} : \vec{T}]u$ denotes the term $[x_1 : T_1] \ldots [x_n : T_n]u$ where $n = |\vec{x}|$.

Definition 3 (Positions) To designate a subterm of a term, we use a system of *positions*. Formally, the set $\mathrm{Pos}(t)$ of the positions in a term t is the smallest set of words over the alphabet of positive integers such that :

- $\mathrm{Pos}(s) = \mathrm{Pos}(x) = \{\varepsilon\}$,
- $\mathrm{Pos}((x:t)u) = \mathrm{Pos}([x:t]u) = \mathrm{Pos}(tu) = 1 \cdot \mathrm{Pos}(t) \cup 2 \cdot \mathrm{Pos}(u)$,
- $\mathrm{Pos}(f(\vec{t})) = \{\varepsilon\} \cup \bigcup \{i \cdot \mathrm{Pos}(t_i) \mid 1 \le i \le \alpha_f\}$,

where ε denotes the empty word and $\cdot$ the concatenation. We denote by $t|_p$ the subterm of t at the position p and by $t[u]_p$ the term obtained by replacing $t|_p$ by u in t. The relation "is subterm of" is denoted by $\lhd$.

Let t be a term and f be a symbol. We denote by $\mathrm{Pos}(f, t)$ the set of the positions p in t where $t|_p$ is of the form $f(\vec{t})$. If x is a variable, we denote by $\mathrm{Pos}(x, t)$ the set of the positions p in t such that $t|_p$ is a free occurrence of x.

Definition 4 (Substitution) A *substitution* θ is an application from $\mathcal{X}$ to $\mathcal{T}$ whose *domain* $\mathrm{dom}(\theta) = \{x \in \mathcal{X} \mid x\theta \ne x\}$ is finite. Applying a substitution θ to a term t consists of replacing all the free variables of t by their image in θ (to avoid variable captures, bound variables must be distinct from free variables). The result is denoted by $t\theta$. We let $\mathrm{dom}^s(\theta) = \mathrm{dom}(\theta) \cap \mathcal{X}^s$. We denote by $\{\vec{x} \mapsto \vec{t}\}$ the substitution which associates t_i to x_i and by $\theta \cup \{x \mapsto t\}$ the substitution which associates t to x and $y\theta$ to $y \ne x$.

Relations

We now recall a few elementary definitions on relations. Let $\rightarrow$ be a relation on terms.

- $\leftarrow$ is the inverse of $\rightarrow$.
- $\rightarrow^+$ is the smallest transitive relation containing $\rightarrow$.
- $\rightarrow^*$ is the smallest reflexive and transitive relation containing $\rightarrow$.
- $\leftrightarrow^*$ is the smallest reflexive, transitive and symmetric relation containing $\rightarrow$.
- $\downarrow$ is the relation $\rightarrow^* {}^*\!\!\leftarrow$ ($t \downarrow u$ if there exists v such that $t \rightarrow^* v$ and $u \rightarrow^* v$).

If $t \rightarrow t'$, we say that t *rewrites* to t'. If $t \rightarrow^* t'$, we say that t *reduces* to t'.

The relation $\rightarrow$ is *stable by context* if $u \rightarrow u'$ implies $t[u]_p \rightarrow t[u']_p$ for all term t and position $p \in \mathrm{Pos}(t)$.

The relation $\rightarrow$ is *stable by substitution* if $t \rightarrow t'$ implies $t\theta \rightarrow t'\theta$ for all substitution θ.

The *β-reduction* (resp. *η-reduction*) relation is the smallest relation stable by context and substitution containing $[x:U]t\ u \rightarrow_\beta t\{x \mapsto u\}$ (resp. $[x:U]tx \rightarrow_\eta t$ if $x \notin \mathrm{FV}(t)$). A term of the form $[x:U]t\ u$ (resp. $[x:U]tx$ with $x \notin \mathrm{FV}(t)$) is a *β-redex* (resp. *η-redex*).

Normalization

The relation $\rightarrow$ is *weakly normalizing* if, for all term t, there exists an irreducible term t' to which t reduces. We say that t' is a *normal form* of t. The relation $\rightarrow$ is *strongly normalizing* (well-founded, nœtherian) if, for all term t, any reduction sequence issued from t is finite.

Confluence

The relation $\rightarrow$ is *locally confluent* if, whenever a term t rewrites to two distinct terms u and v, then $u \downarrow v$. The relation $\rightarrow$ is *confluent* if, whenever a term t reduces to two distinct terms u and v, then $u \downarrow v$.

If $\rightarrow$ is locally confluent and strongly normalizing then $\rightarrow$ is confluent [94]. If $\rightarrow$ is confluent and weakly normalizing then every term t has a normal form denoted by $t \downarrow$.

Lexicographic and multiset orderings

Let $>_1, \ldots, >_n$ be orderings on $E_1, \ldots, E_n$ respectively. We denote by $(>_1, \ldots, >_n)_{\text{lex}}$ the *lexicographic* ordering on $E_1 \times \ldots \times E_n$ from $>_1, \ldots, >_n$. For example, for $n = 2$, $(x, y)(>_1, >_2)_{\text{lex}}(x', y')$ if $x >_1 x'$ or, $x =_1 x'$ and $y >_2 y'$.

Let E be a set. A *multiset* M on E is a function from E to $\mathbb{N}$ ($M(x)$ denotes the number of occurrences of x in M). We denote by $\mathcal{M}(E)$ the set of finite multisets on E. Let $>$ be an ordering on E, the *multiset extension* of $>$ is the ordering $>_{\text{mul}}$ on $\mathcal{M}(E)$ defined as follows : $M >_{\text{mul}} N$ if there exists $P, Q \in \mathcal{M}(E)$ such that $P \neq \emptyset$, $P \subseteq M$, $N = (M \setminus P) \cup Q$ and, for all $y \in Q$, there exists $x \in P$ such that $x > y$.

An important property of these extensions is that they preserve the well-foundedness. For more details on these notions, one can consult [3].

Chapter 3

Type Systems Modulo (TSM's)

In this chapter, we consider an extension of PTS's with function and predicate symbols and a conversion rule (conv) where $\leftrightarrow_\beta^*$ is replaced by an arbitrary conversion relation $\mathcal{C}$.

There has already been different extension of PTS's, in particular :

- In 1989, Z. Luo [81] studied an extension of the Calculus of Constructions with a cumulative hierarchy of sorts ($\star \prec \Box = \Box_0 \prec \Box_1 \prec \ldots$), the Extended Calculus of Constructions (ECC) : $\mathcal{C}$ is the smallest quasi-ordering including $\leftrightarrow_\beta^*$, $\prec$ and which is compatible with the product structure ($U' \;\mathcal{C}\; U$ and $V \;\mathcal{C}\; V'$ implies $(x\!:\!U)V \;\mathcal{C}\; (x\!:\!U')V'$).
- In 1993, H. Geuvers [58] studied the PTS's with η-reduction : $\mathcal{C} = \leftrightarrow_{\beta\eta}^*$.
- In 1993, M. Fernández [54] studied an extension of the Calculus of Constructions with higher-order rewriting *à la* Jouannaud-Okada [74], the λR-cube : $\mathcal{C} = \rightarrow_{\beta\mathcal{R}}^* \cup {}_{\beta R}^*\!\leftarrow$.
- In 1994, E. Poll and P. Severi [101] studied the PTS's with abbreviations (`let x= ... in ...`) : $\mathcal{C} = \leftrightarrow_\beta^* \cup \leftrightarrow_\delta^*$ where $\rightarrow_\delta$ is the replacement of an abbreviation by its definition.
- In 1994, B. Werner [119] studied an extension of the Calculus of Constructions with inductive types, the Calculus of Inductive Constructions (CIC), introduced by T. Coquand and C. Paulin in 1988 [39] : $\mathcal{C} = \leftrightarrow_{\beta\eta\iota}^*$ where $\rightarrow_\iota$ is the reduction relation associated with the elimination schemas of inductive types.
- Between 1995 and 1998, G. Barthe and his co-authors [15, 16, 17, 13] considered different extensions of the Calculus of Constructions or of the PTS's with conversion relations more or less abstract, often based on rewriting *à la* Jouannaud-Okada [74], hence extending the work of M. Fernández [54].

In all this work, basic properties well known in the case of PTS's must be proved again since new constructions or a new conversion rule $\mathcal{C}$ is introduced. This is why it appears useful for us to study the properties of PTS's equipped with an abstract conversion relation $\mathcal{C}$.

Such a need is not new since it has already been undertaken in formal developments :

- In 1994, R. Pollack [102] formally proved in LEGO [82], an implementation of ECC with inductive types, that type checking in ECC (without inductive types) is decidable (by assuming of course that the calculus is strongly normalizing). Type checking is saying if, in some environment Γ, a term t is of type T (*i.e.* is a proof of T). To this end, he showed many properties of PTS's in the case of a conversion relation $\mathcal{C} = \leq$ reflexive, transitive and stable by substitution and context.
- In 1999, B. Barras [11] formally proved in Coq [12], another implementation of ECC with inductive types, that type checking of Coq (so, with inductive types) is decidable (again of course by assuming that the calculus is strongly normalizing). To this end, he also showed some properties of PTS's in the case of a conversion relation $\mathcal{C} = \leq$ also reflexive, transitive and stable by substitution and context. In fact, he considered an extension of PTS's with a schema of typing rules for introducing new constructions in a generic way (abbreviations, inductive types, elimination schemas).

Hence, on the one hand, we make fewer assumptions on the conversion relation $\mathcal{C}$ than R. Pollack or B. Barras. This is justified by the fact that, in the work of M. Fernández [54] for example, to prove that reduction preserves typing, they use the fact that the conversion relation is not transitive. On the other hand, our typing rule for function symbols is not as general as the one of B. Barras.

3.1 Definition

Definition 5 (Environment) An *environment* Γ is a list of pairs $x_i\!:\!T_i$ made of a variable x_i and a term T_i. We denote by $\emptyset$ the empty environment, by $\mathcal{E}$ the set of environments and by $x_i\Gamma$ the term T_i associated to x_i in Γ. The set of *free variables* of an environment Γ is $\mathrm{FV}(\Gamma) = \bigcup\{\mathrm{FV}(x\Gamma) \mid x \in \mathrm{dom}(\Gamma)\}$. The *domain* of an environment Γ is the set of variables to which Γ associates a term. Given two environments Γ and Γ', Γ is *included* in Γ', written $\Gamma \subseteq \Gamma'$, if all the elements of Γ occur in Γ' in the same ordering.

Definition 6 (Type assignment) A *type assignment* is a function τ from $\mathcal{F}$ to $\mathcal{T}$ which, to a symbol f of arity n, associates a closed term τ_f of the form $(\vec{x} : \vec{T})U$ where $|\vec{x}| = n$. We will denote by Γ_f the environment $\vec{x} : \vec{T}$.

Definition 7 (TSM) A *Type System Modulo* (TSM) is a sorted λ-system $(\mathcal{S}, \mathcal{F}, \mathcal{X})$ with :
- a set of *axioms* $\mathcal{A} \subseteq \mathcal{S}^2$,
- a set of product formation *rules* $\mathcal{B} \subseteq \mathcal{S}^3$,
- a type assignment τ,
- a *conversion relation* $\mathcal{C} \subseteq \mathcal{T}^2$.

A βTSM (resp. ηTSM) is a TSM such that $\downarrow_\beta \subseteq \mathcal{C}$ (resp. $\downarrow_\eta \subseteq \mathcal{C}$).

Definition 8 (Typing) The typing relation of a TSM is the smallest ternary relation $\vdash \subseteq \mathcal{E} \times \mathcal{T} \times \mathcal{T}$ defined by the inference rules of Figure 3.1. Compared with PTS's, we have a new rule, (symb), for typing the symbols and, in the conversion rule (conv), instead of the β-conversion, we have an abstract conversion relation $\mathcal{C}$. A term t is *typable* if there exists an environment Γ and a term T such that $\Gamma \vdash t : T$ (T is a *type* of t in Γ). An environment Γ is *valid* if there exists a term typable in Γ. In the rule (symb), the premise "Γ valid" is therefore useful only if f is of null arity ($n = 0$).

- $\mathbb{T} = \{t \in \mathcal{T} \mid \exists \Gamma \in \mathcal{E}, \exists T \in \mathcal{T},\ \Gamma \vdash t : T\}$ is the set of typable terms,
- $\mathbb{T}_0^s = \{T \in \mathcal{T} \mid \exists \Gamma \in \mathcal{E},\ \Gamma \vdash T : s\}$ is the set of *predicates* of sort s,
- $\mathbb{T}_1^s = \{t \in \mathcal{T} \mid \exists \Gamma \in \mathcal{E}, \exists T \in \mathcal{T},\ \Gamma \vdash t : T \text{ and } \Gamma \vdash T : s\}$ is the set of *objects* of sort s,
- $\mathbb{E} = \{\Gamma \in \mathcal{E} \mid \exists t, T \in \mathcal{T},\ \Gamma \vdash t : T\}$ is the set of valid environments.

Figure 3.1: TSM typing rules

$$\text{(ax)} \quad \frac{}{\vdash s_1 : s_2} \quad ((s_1, s_2) \in \mathcal{A})$$

$$\text{(symb)} \quad \frac{\vdash \tau_f : s \quad \Gamma \text{ valid} \qquad \Gamma \vdash t_1 : T_1\gamma \quad \ldots \quad \Gamma \vdash t_n : T_n\gamma}{\Gamma \vdash f(\vec{t}) : U\gamma} \quad \left(f \in \mathcal{F}_n^s,\ \tau_f = (\vec{x} : \vec{T})U,\ \gamma = \{\vec{x} \mapsto \vec{t}\}\right)$$

$$\text{(var)} \quad \frac{\Gamma \vdash T : s}{\Gamma, x{:}T \vdash x : T} \quad (x \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma))$$

$$\text{(weak)} \quad \frac{\Gamma \vdash t : T \quad \Gamma \vdash U : s}{\Gamma, x{:}U \vdash t : T} \quad (x \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma))$$

$$\text{(prod)} \quad \frac{\Gamma \vdash T : s_1 \quad \Gamma, x{:}T \vdash U : s_2}{\Gamma \vdash (x{:}T)U : s_3} \quad ((s_1, s_2, s_3) \in \mathcal{B})$$

$$\text{(abs)} \quad \frac{\Gamma, x{:}T \vdash u : U \quad \Gamma \vdash (x{:}T)U : s}{\Gamma \vdash [x{:}T]u : (x{:}T)U}$$

$$\text{(app)} \quad \frac{\Gamma \vdash t : (x{:}U)V \quad \Gamma \vdash u : U}{\Gamma \vdash tu : V\{x \mapsto u\}}$$

$$\text{(conv)} \quad \frac{\Gamma \vdash t : T \quad \Gamma \vdash T' : s'}{\Gamma \vdash t : T'} \quad (T\ \mathcal{C}\ T')$$

Remark 9 (Conversion)
It might seem more natural to define (conv) in a symmetric way by adding the premise $\Gamma \vdash T : s$ or the premise $\Gamma \vdash T : s'$. We have chosen this definition for two reasons. First, it is defined in this way in the reference papers on PTS's [59, 10]. Second, from a practical point of view, for type checking, this avoids an additional test. However, we will see in Lemma 37 that, for many TSM's, we have $\Gamma \vdash T : s'$. We will denote by $\vdash_s$ the typing relation defined by the same inference rules as for $\vdash$ but with (conv) replaced by :

$$\text{(conv')} \quad \frac{\Gamma \vdash t : T \quad \Gamma \vdash T : s \quad \Gamma \vdash T' : s}{\Gamma \vdash t : T'} \quad (T \; \mathcal{C} \; T')$$

We will show the equivalence between $\vdash_s$ and $\vdash$ in Lemma 43.

In the same way, in *full* TSM's ($\forall s_1, s_2 \in \mathcal{S}, \exists s_3 \in \mathcal{S}, (s_1, s_2, s_3) \in \mathcal{B}$), the rule (abs) can be replaced by :

$$\text{(abs')} \quad \frac{\Gamma, x\!:\!T \vdash u : U}{\Gamma \vdash [x\!:\!T]u : (x\!:\!T)U} \quad (U \notin \mathcal{S} \text{ or } \exists s \!\in\! \mathcal{S}, (U, s) \in \mathcal{A})$$

Finally, we will denote by $\vdash_w$ the typing relation defined by the same inference rules as for $\vdash$ but with (weak) replaced by :

$$\text{(weak')} \quad \frac{\Gamma \vdash t : T \quad \Gamma \vdash U : s}{\Gamma, x\!:\!U \vdash t : T} \quad (x \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma), t \in \mathcal{X} \cup \mathcal{S})$$

that is, where weakening is restricted to variables and sorts ($t \in \mathcal{X} \cup \mathcal{S}$). We will show the equivalence between $\vdash_w$ and $\vdash$ in Lemma 19.

Remark 10 (Arity)
On can wonder why symbols are equipped with a fixed arity since, in general, in λ-calculus, one is used to consider higher-order constants. Of course, having an arity is not a restriction since, to a symbol f of arity n and type $(\vec{x} : \vec{T})U$, one can always associate a curried version f^c of null arity defined by the rewrite rule $f^c \to [\vec{x} : \vec{T}]f(\vec{x})$. Furthermore, in practice, the existence of arities can be masked by doing η-expansions if f is not applied to sufficient arguments, or by doing additional applications if f is applied to too many arguments. However, without arities, we would have a simpler presentation where the rule (symb) would be reduced to :

$$\text{(symb')} \quad \frac{\vdash \tau_f : s}{\vdash f : \tau_f} \quad (f \in \mathcal{F}^s)$$

To our knowledge, except the works of Jouannaud and Okada [75] and of G. Barthe and his co-authors [15, 16, 17, 13], most of the other works on the combination of typed λ-calculi and rewriting [25, 26, 54] do not use arities for typing symbols. We have chosen to use arities for technical reasons. With the method we use for proving strong normalization, we need an application uv not to be a rewriting redex (see Chapter 5 for an explanation of these notions and Lemma 121, case $T = (x : U)V$, (b), (R3) for the use of this property). The introduction of arities enable us to syntactically distinguish between the application of the λ-calculus and the application of a symbol. But one may wonder whether this notion is really necessary.

Definition 11 (Well-typed substitution) Given two valid environments Γ and Δ, a substitution θ is *well typed between* Γ *and* Δ , $\theta : \Gamma \to \Delta$, if, for all $x \in \mathrm{dom}(\Gamma)$, $\Delta \vdash x\theta : x\Gamma\theta$.

For example, in the rule (symb), we have $\gamma : \Gamma_f \to \Gamma$ where $\Gamma_f = \vec{x} : \vec{T}$.

3.2 Properties

In this section and the following one, we prove some properties of TSM's that are well known for PTS's (except Lemma 22). Apart from the new case (symb) that we will detail each time, proofs are identical to the ones for PTS's. The fact that, in (conv), $\leftrightarrow_\beta^*$ is replaced by an arbitrary relation $\mathcal{C}$ is not important. For more details, the reader is invited to look at [59, 10, 58].

Lemma 12 (Free variables) Let $\Gamma = \vec{x} : \vec{T}$ be an environment. If $\Gamma \vdash t : T$ then :
(a) the x_i's are distinct from one another,
(b) $\mathrm{FV}(t) \cup \mathrm{FV}(T) \subseteq \mathrm{dom}(\Gamma)$,
(c) for all i, $\mathrm{FV}(T_i) \subseteq \{x_1, \ldots, x_{i-1}\}$.

Proof. By induction on $\Gamma \vdash t : T$. We only detail the new case (symb). (a) and (c) are true by induction hypothesis. Let us see (b) now. By induction hypothesis, $\mathrm{FV}(\tau_f) = \emptyset$ and, for all i, $\mathrm{FV}(t_i) \subseteq \mathrm{dom}(\Gamma)$. Therefore, $\mathrm{FV}(f(\vec{t})) \subseteq \mathrm{dom}(\Gamma)$. Hence, $\mathrm{FV}(U\gamma) \subseteq \mathrm{dom}(\Gamma)$ since $\mathrm{FV}(U) \subseteq \{\vec{x}\}$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$. ■

Lemma 13 (Subterms) If a term is typable then all its subterms are typable.

Proof. By induction on $\Gamma \vdash t : T$. In the case of (symb), by induction hypothesis, for all i, all the subterms of t_i are typable. Therefore, all the subterms of $f(\vec{t})$ are typable. ■

Lemma 14 (Environment) Let $\Gamma = \vec{x} : \vec{T}$ be a valid environment.
(a) If x_i is of sort s then $x_1 : T_1, \ldots, x_{i-1} : T_{i-1} \vdash T_i : s$.
(b) For all i, $x_1 : T_1, \ldots, x_i : T_i \vdash x_i : T_i$.

Proof. By (var), (b) is an immediate consequence of (a). We prove (a) by induction on $\Gamma \vdash t : T$. In the case (symb), as Γ is valid, there exists v and V such that $\Gamma \vdash v : V$. Therefore, by induction hypothesis, (a) is true. ■

The following lemma is a form of α-equivalence on the variables of an environment.

Lemma 15 (Replacement) If $\Gamma, y : W, \Gamma' \vdash t : T$, $y \in \mathcal{X}^s$ and $z \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma, y : W, \Gamma')$ then $\Gamma, z : W, \Gamma'\{y \mapsto z\} \vdash t\{y \mapsto z\} : T\{y \mapsto z\}$.

Proof. By induction on $\Gamma, y : W, \Gamma' \vdash t : T$. Let $\theta = \{y \mapsto z\}$, $\Delta = \Gamma, y : W, \Gamma'$ and $\Delta' = \Gamma, z : W, \Gamma'\theta$. In the case (symb), by induction hypothesis, we have Δ' valid and, for all i, $\Delta' \vdash t_i\theta : T_i\gamma\theta$. Therefore, by (symb), $\Delta' \vdash f(\vec{t}\theta) : U\gamma\theta$. ■

Lemma 16 (Weakening) If $\Gamma \vdash t : T$ and $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $\Gamma' \vdash t : T$.

Proof. By induction on $\Gamma \vdash t : T$. In the case (symb), by induction hypothesis, we have Γ' valid and, for all i, $\Gamma' \vdash t_i : T_i\gamma$. Therefore, by (symb), $\Gamma' \vdash f(\vec{t}) : U\gamma$. ■

Lemma 17 (Transitivity) Let Γ and Δ be two valid environments. If $\Gamma \vdash t : T$ and, for all $x \in \mathrm{dom}(\Gamma)$, $\Delta \vdash x : x\Gamma$, that we will denote by $\Delta \vdash \Gamma$, then $\Delta \vdash t : T$.

Proof. By induction on $\Gamma \vdash t : T$. In the case (symb), by induction hypothesis, we have Δ valid and, for all i, $\Delta \vdash t_i : T_i\gamma$. Therefore, by (symb), $\Delta \vdash f(\vec{t}) : U\gamma$. ■

Lemma 18 (Weak permutation) If $\Gamma, y\!:\!A, z\!:\!B, \Gamma' \vdash t : T$ and $\Gamma \vdash B : s$ then $\Gamma, z\!:\!B, y\!:\!A, \Gamma' \vdash t : T$.

Proof. Let $\Delta = \Gamma, y : A, z : B, \Gamma'$ and $\Delta' = \Gamma, z : B, y : A, \Gamma'$. By transitivity, it suffices to prove that Δ' is valid and that $\Delta' \vdash \Delta$. And to this end, it suffices to prove that Δ' is valid. By the Environment Lemma, we have $\Gamma \vdash A : s'$ and, by hypothesis, we have $\Gamma \vdash B : s$. Therefore, by weakening, $\Gamma, z : B, y : A$ is valid. Assume that $\Gamma' = \vec{x} : \vec{T}$ and let $\Delta_i = \Gamma, y : A, z : B, x_1 : T_1, \ldots, x_i : T_i$ and $\Delta'_i = \Gamma, z\!:\!B, y\!:\!A, x_1\!:\!T_1, \ldots, x_i\!:\!T_i$. We prove by induction on i that Δ'_i is valid. We have already proved that Δ'_0 is valid. Assume that Δ'_i is valid. By the Environment Lemma, $\Delta_i \vdash T_{i+1} : s_{i+1}$. As $\Delta'_i \vdash \Delta_i$, $\Delta'_i \vdash T_{i+1} : s_{i+1}$ and Δ'_{i+1} is valid. Therefore, Δ' is valid and $\Delta' \vdash t : T$. ■

Lemma 19 (Equivalence of $\vdash_w$ and $\vdash$) $\vdash_w = \vdash$.

Proof. First of all, it is clear that $\vdash_w \subseteq \vdash$. We prove the reverse by induction on $\Gamma \vdash t : T$. The only difficult case is of course (weak) : from $\Gamma \vdash t : T$ and $\Gamma \vdash U : s$, we get $\Gamma, x\!:\!U \vdash t : T$. By induction hypothesis, we have $\Gamma \vdash_w t : T$ and $\Gamma \vdash_w U : s$. One has to modify the proof of $\Gamma \vdash_w t : T$ by adding $x\!:\!U$ at the appropriate places in order to obtain a proof of $\Gamma, x\!:\!U \vdash_w t : T$. See Lemma 4.4.21 page 102 in [58] for more details. ■

Now, let us see what we can do about the derivations of $\Gamma \vdash t : T$ and the form of T with respect to t. To this end, we introduce relations related to the rule (conv).

Definition 20 (Conversion relations)
- $T\ \mathcal{C}_\Gamma\ T'$ iff $T\ \mathcal{C}\ T'$ and there exists t, t' and s' such that $\Gamma \vdash t : T$, $\Gamma \vdash t' : T'$ and $\Gamma \vdash T' : s'$,
- $T\ \mathbb{C}_\Gamma\ T'$ iff $T\ \mathcal{C}_\Gamma\ T'$ and there exists s such that $\Gamma \vdash T : s$,
- $\Gamma\ \mathbb{C}\ \Gamma'$ iff $\Gamma = \vec{x} : \vec{T}$, $\Gamma' = \vec{x} : \vec{T'}$ and, either $|\vec{x}| = 0$, or there exists j such that $T_j\ \mathbb{C}_{x_1:T_1,\ldots,x_{j-1}:T_{j-1}}\ T'_j$ and, for all $i \neq j$, $T_i = T'_i$.

We have $\mathbb{C}_\Gamma \subseteq \mathcal{C}_\Gamma$ but, as opposed to $\mathbb{C}_\Gamma$, $\mathcal{C}_\Gamma$ is not defined in a symmetric way. This comes from the asymmetry of the rule (conv) which requires $\Gamma \vdash T' : s'$ but not $\Gamma \vdash T : s$. However, we will see in Lemma 37 that, for many TSM's, these two relations are equal.

Lemma 21 (Inversion) Assume that $\Gamma \vdash t : T$.

– If $t = s$ then there exists s' such that $(s, s') \in \mathcal{A}$ and $s' \mathcal{C}^*_\Gamma T$.
– If $t = f(\vec{t})$, $f \in \mathcal{F}^s$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$ then $\vdash \tau_f : s$, $\gamma : (\vec{x} : \vec{T}) \to \Gamma$ and $U\gamma \mathcal{C}^*_\Gamma T$.
– If $t = x \in \mathcal{X}^s$ then $\Gamma \vdash x\Gamma : s$ and $x\Gamma \mathbb{C}^*_\Gamma T$.
– If $t = (x:U)V$ then there exists $(s_1, s_2, s_3) \in \mathcal{P}$ such that $\Gamma \vdash U : s_1$, $\Gamma, x:U \vdash V : s_2$ and $s_3 \mathcal{C}^*_\Gamma T$.
– If $t = [x:U]v$ then there exists V such that $\Gamma, x:U \vdash v : V$ and $(x:U)V \mathbb{C}^*_\Gamma T$.
– If $t = uv$ then there exists V and W such that $\Gamma \vdash u : (x:V)W$, $\Gamma \vdash v : V$ and $W\{x \mapsto v\} \mathcal{C}^*_\Gamma T$.

Proof. A typing derivation always finishes by a rule distinct from (weak) and (conv) followed by a possibly empty sequence of (weak)'s and (conv)'s. We hence get the term to which T is convertible. For typing judgments, it suffices to do a weakening to express them in Γ. ■

Lemma 22 (Environment conversion) If $\Gamma \vdash t : T$ and $\Gamma \mathbb{C} \Gamma'$ then $\Gamma' \vdash t : T$.

Proof. We have $\Gamma = \vec{x} : \vec{T}$, $\Gamma' = \vec{x} : \vec{T'}$ and there exists j such that $T_j \mathbb{C}_\Delta T'_j$ with $\Delta = x_1 : T_1, \ldots, x_{j-1} : T_{j-1}$ and, for all $i \neq j$, $T_i = T'_i$. By transitivity, it suffices to prove that, for all i, $\Gamma' \vdash x_i : T_i$. Let $n = |\vec{x}|$, $\Gamma_1 = x_1:T_1, \ldots, x_{j-1}:T_{j-1}$ and $\Gamma_2 = x_{j+1}:T_{j+1}, \ldots, x_n:T_n$. We proceed by induction on the size of Γ_2.

If Γ_2 is empty then $\Gamma = \Gamma_1, x_j:T_j$ and $\Gamma' = \Gamma_1, x_j:T'_j$. Since Γ is valid, Γ_1 is valid and, for all $i < j$, $\Gamma_1 \vdash x_i : T_i$. Since $T_j \mathbb{C}_{\Gamma_1} T'_j$, there exists s and s' such that $\Gamma_1 \vdash T_j : s$ and $\Gamma_1 \vdash T'_j : s'$. By (weak), we get, for all $i < j$, $\Gamma' \vdash x_i : T_i$, and by (var), we get $\Gamma' \vdash x_j : T'_j$. From $\Gamma_1 \vdash T_j : s$, by (weak), we also get $\Gamma' \vdash T_j : s$. Therefore, by (conv), $\Gamma' \vdash x_j : T_j$.

assume now that $\Gamma_2 = \Gamma_3, x_n:T_n$. Let $\Delta = \Gamma_1, x_j:T_j, \Gamma_3$ and $\Delta' = \Gamma_1, x_j:T'_j, \Gamma_3$. By induction hypothesis, for all $i < n$, $\Delta' \vdash x_i : T_i$. Since Γ is valid, there exists s such that $\Delta \vdash T_n : s$. By transitivity, we get $\Delta' \vdash T_n : s$. Therefore, by (var), we get $\Gamma' \vdash x_n : T_n$, and by (weak), $\Gamma' \vdash x_i : T_i$. ■

3.3 TSM's stable by substitution

Definition 23 (TSM stable by substitution) A TSM is stable by substitution if its conversion relation $\mathcal{C}$ is stable by substitution.

Lemma 24 (Substitution) If $\mathcal{C}$ is stable by substitution, $\Gamma \vdash t : T$ and $\theta : \Gamma \to \Delta$ then $\Delta \vdash t\theta : T\theta$.

Proof. By induction on $\Gamma \vdash t : T$. In the case (symb), by induction hypothesis, we have $\Delta \vdash t_i\theta : T_i\gamma\theta$. Therefore, by (symb), $\Delta \vdash f(\vec{t}\theta) : U\gamma\theta$. ■

Corollary 25 If $\mathcal{C}$ is stable by substitution, $\Gamma, x:U, \Gamma' \vdash t : T$ and $\Gamma \vdash u : U$ then $\Gamma, \Gamma'\{x \mapsto u\} \vdash t\{x \mapsto u\} : T\{x \mapsto u\}$.

Proof. We have to prove that $\theta = \{x \mapsto u\}$ is a well-typed substitution from $\Gamma, x:U, \Gamma'$ to $\Gamma, \Gamma'\theta$. We proceed by induction on the size of Γ'. If Γ' is empty, this is immediate since Γ is valid and $\Gamma \vdash u : U$. Assume now that $\Gamma' = \Gamma'', y\!:\!V$. Let $\Delta = \Gamma, x\!:\!U, \Gamma''$ and $\Delta' = \Gamma, \Gamma''\theta$. By induction hypothesis, $\theta : \Delta \to \Delta'$. Since $\Delta \vdash V : s$, by substitution, we get $\Delta' \vdash V\theta : s$. Therefore, by (var), $\Delta', y\!:\!V\theta \vdash y : V\theta$. Now, let $z \in \mathrm{dom}(\Delta)$. As $\Delta \vdash z : z\Delta$, by substitution, $\Delta' \vdash z : z\Delta\theta$. Then, by (weak), $\Delta', y\!:\!V\theta \vdash z : z\Delta\theta$. ■

Corollary 26 If $\mathcal{C}$ is stable by substitution, $\theta_1 : \Gamma_0 \to \Gamma_1$ and $\theta_2 : \Gamma_1 \to \Gamma_2$ then $\theta_1\theta_2 : \Gamma_0 \to \Gamma_2$.

Proof. Let $x \in \mathrm{dom}(\Gamma_0)$. Since $\theta_1 : \Gamma_0 \to \Gamma_1$, by substitution, we get $\Gamma_1 \vdash x\theta_1 : x\Gamma_0\theta_1$, and since $\theta_2 : \Gamma_1 \to \Gamma_2$, by substitution again, we get $\Gamma_2 \vdash x\theta_1\theta_2 : x\Gamma_0\theta_1\theta_2$. ■

Definition 27 (Maximal sort) A sort s is *maximal* if there does not exist any sort s' such that $(s, s') \in \mathcal{A}$.

Lemma 28 (Correctness of types) If $\mathcal{C}$ is stable by substitution and $\Gamma \vdash t : T$ then, either T is a maximal sort, or there exists a sort s such that $\Gamma \vdash T : s$. In other words, $\mathbb{T} = \bigcup\{\mathbb{T}_0^s \cup \mathbb{T}_1^s \mid s \in \mathcal{S}\}$.

Proof. By induction on $\Gamma \vdash t : T$. In the case (symb), we have $\vdash \tau_f : s$. By inversion, there exists s' such that $\Gamma_f \vdash U : s'$. As $\gamma : \Gamma_f \to \Gamma$, by substitution, $\Gamma \vdash U\gamma : s'$. ■

Lemma 29 (Inversion for TSM's stable by substitution) Assume that $\Gamma \vdash t : T$.

- If $t = s$ then there exists s' such that $(s, s') \in \mathcal{A}$ and $s'\ \mathcal{C}_\Gamma^*\ T$.
- If $t = f(\vec{t})$, $f \in \mathcal{F}^s$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$ then $\vdash \tau_f : s$, $\gamma : (\vec{x} : \vec{T}) \to \Gamma$ and $U\gamma\ \mathbb{C}_\Gamma^*\ T$.
- If $t = x \in \mathcal{X}^s$ then $\Gamma \vdash x\Gamma : s$ and $x\Gamma\ \mathbb{C}_\Gamma^*\ T$.
- If $t = (x\!:\!U)V$ then there exists $(s_1, s_2, s_3) \in \mathcal{P}$ such that $\Gamma \vdash U : s_1$, $\Gamma, x\!:\!U \vdash V : s_2$ and $s_3\ \mathcal{C}_\Gamma^*\ T$.
- If $t = [x\!:\!U]v$ then there exists V such that $\Gamma, x\!:\!U \vdash v : V$ and $(x\!:\!U)V\ \mathbb{C}_\Gamma^*\ T$.
- If $t = uv$ then there exists V and W such that $\Gamma \vdash u : (x\!:\!V)W$, $\Gamma \vdash v : V$ and $W\{x \mapsto v\}\ \mathbb{C}_\Gamma^*\ T$.

Proof. Only the cases $t = f(\vec{t})$ and $t = uv$ have been modified.

- $t = f(\vec{t})$. By inversion, $\vdash \tau_f : s$, $\gamma : (\vec{x} : \vec{T}) \to \Gamma$ and $U\gamma\ \mathcal{C}_\Gamma^*\ T$. By inversion again, there exists s' such that $\vec{x} : \vec{T} \vdash U : s'$. Therefore, by substitution, $\Gamma \vdash U\gamma : s'$ and $U\gamma\ \mathbb{C}_\Gamma^*\ T$.
- $t = uv$. By inversion, there exists V and W such that $\Gamma \vdash u : (x : V)W$, $\Gamma \vdash v : V$ and $W\{x \mapsto v\}\ \mathcal{C}_\Gamma^*\ T$. By correctness of types, there exists s such that $\Gamma \vdash (x\!:\!V)W : s$. By inversion, there exists s' such that $\Gamma, x\!:\!V \vdash W : s'$. Therefore, by substitution, $\Gamma \vdash W\{x \mapsto v\} : s'$ and $W\{x \mapsto v\}\ \mathbb{C}_\Gamma^*\ T$. ■

3.4 Logical TSM's

We now introduce an important class of TSM's for which β-reduction preserves typing.

Definition 30 (Logical TSM) A TSM is *logical* if its conversion relation is *product compatible* :

$$(x:T)U \ \mathbb{C}^*_\Gamma \ (x':T')U' \text{ implies } T \ \mathbb{C}^*_\Gamma \ T' \text{ and } U \ \mathbb{C}^*_{\Gamma,x:T} \ U'\{x' \mapsto x\}.$$

$T \ \mathbb{C}^*_\Gamma \ T'$ means that there exists a sequence of terms $\vec{T}$ such that $T_0 = T \ \mathbb{C}_\Gamma \ T_1 \ldots T_{n-1} \ \mathbb{C}_\Gamma \ T_n = T'$. So, there is no reason *a priori* to take $U \ \mathbb{C}^*_{\Gamma,x:T} \ U'\{x' \mapsto x\}$ instead of $U \ \mathbb{C}^*_{\Gamma,x:T_i} \ U'\{x' \mapsto x\}$ with $i \neq 0$. However, as $T \ \mathbb{C}^*_\Gamma \ T'$, by environment conversion, this choice is not important.

Product compatibility is not a new condition and appears in all the previously cited works but, to our knowledge, it has never received any special name.

All the TSM's cited at the beginning of this chapter are logical. In the case where $\mathcal{C} = \leftrightarrow^*$ with $\rightarrow$ a confluent reduction relation, it is clear that $\mathcal{C}$ is product compatible. To prove this property without using confluence is more delicate. This is the case of PTS's with η-reduction [58] or of the λR-cube [54], an extension of the Calculus of Constructions with higher-order rewriting *à la* Jouannaud-Okada at the object-level.

Lemma 31 (Subject reduction for β) In a logical βTSM, if $\Gamma \vdash t : T$ and $t \rightarrow_\beta t'$ then $\Gamma \vdash t' : T$.

Proof. We will say that an environment $\vec{x} : \vec{T}$ β-rewrites to an environment $\vec{x}' : \vec{T}'$, written $\vec{x} : \vec{T} \rightarrow_\beta \vec{x}' : \vec{T}'$, if $\vec{x} = \vec{x}'$ and there exists j such that $T_j \rightarrow_\beta T'_j$ and, for all $i \neq j$, $T_i = T'_i$. We simultaneously prove that :

(a) if $t \rightarrow_\beta t'$ then $\Gamma \vdash t' : T$,

(b) if $\Gamma \rightarrow_\beta \Gamma'$ then $\Gamma' \vdash t : T$,

by induction on $\Gamma \vdash t : T$.

(ax) $\vdash s_1 : s_2 \quad ((s_1, s_2) \in \mathcal{A})$
No β-reduction is possible in s_1 or in $\Gamma = \emptyset$.

(symb) $$\frac{\vdash \tau_f : s \quad \Gamma \text{ valid} \quad \Gamma \vdash t_1 : T_1\gamma \ldots \Gamma \vdash t_n : T_n\gamma}{\Gamma \vdash f(\vec{t}) : U\gamma} \quad \begin{array}{c}(f \in \mathcal{F}^s_n, \\ \tau_f = (\vec{x} : \vec{T})U, \\ \gamma = \{\vec{x} \mapsto \vec{t}\})\end{array}$$

(a) If $f(\vec{t}) \rightarrow_\beta t'$ then $t' = f(\vec{t'})$ with j such that $t_j \rightarrow_\beta t'_j$ and, for all $i \neq j$, $t_i = t'_i$. By induction hypothesis, we have, for all i, $\Gamma \vdash t'_i : T_i\gamma$. Let $\gamma' = \{\vec{x} \mapsto \vec{t'}\}$. We have $U\gamma'^* \leftarrow \beta U\gamma$ and, for all i, $T_i\gamma \rightarrow^*_\beta T_i\gamma'$. As $\downarrow_\beta \subseteq \mathcal{C}$, $U\gamma' \ \mathcal{C} \ U\gamma$ and $T_i\gamma \ \mathcal{C} \ T_i\gamma'$. If we prove that every $T_i\gamma'$ is typable in Γ by a sort then, by (conv), we have $\Gamma \vdash t'_i : T_i\gamma'$ and, by (symb), $\Gamma \vdash t' : U\gamma'$. It then suffices to prove that $U\gamma$ is typable by a sort in Γ to apply again (conv) and conclude that $\Gamma \vdash t' : U\gamma$.

Let us begin by verifying that $U\gamma$ is typable by a sort. We have $\vdash \tau_f : s$. By inversion, $\gamma : (\vec{x} : \vec{T}) \to \Gamma$ and there exists s' such that $\vec{x} : \vec{T} \vdash U : s'$. By substitution, we therefore get $\Gamma \vdash U\gamma : s'$.

Now, we are going to prove that every $T_i\gamma'$ is typable by a sort. To this end, it suffices to prove that $\gamma' : (\vec{x} : \vec{T}) \to \Gamma$. Indeed, since $\vdash \tau_f : s$, by inversion, every T_i is typable by a sort in $\Gamma_{i-1} = x_1 \!:\! T_1, \ldots, x_{i-1} \!:\! T_{i-1}$. Let us prove by induction on i that $\gamma' : \Gamma_i \to \Gamma$.

For $i = 0$, there is nothing to prove. Assume therefore that $\gamma' : \Gamma_i \to \Gamma$. Then $\gamma' : \Gamma_{i+1} \to \Gamma$ if $\Gamma \vdash t'_{i+1} : T_{i+1}\gamma'$. We know that $\Gamma \vdash t'_{i+1} : T_{i+1}\gamma$, $T_{i+1}\gamma \to_\beta^* T_{i+1}\gamma'$ and that there exists s such that $\Gamma_i \vdash T_{i+1} : s$. Therefore, by substitution, $\Gamma \vdash T_{i+1}\gamma' : s$ and, by (conv), $\Gamma \vdash t'_{i+1} : T_{i+1}\gamma'$.

(b) If $\Gamma \to_\beta \Gamma'$ then, by induction hypothesis, Γ' is valid and $\Gamma' \vdash t_i : T_i\gamma$. Therefore, by (symb), $\Gamma' \vdash f(\vec{t}) : U\gamma$.

(var) $\dfrac{\Gamma \vdash T : s}{\Gamma, x \!:\! T \vdash x : T}$

(a) No β-reduction is possible in x.

(b) There is two cases, depending on where takes place the β-reduction :

- $\Gamma \to_\beta \Gamma'$. By induction hypothesis, $\Gamma' \vdash T : s$. Therefore, by (var), $\Gamma', x : T \vdash x : T$.
- $T \to_\beta T'$. By induction hypothesis, $\Gamma \vdash T' : s$. Therefore, by (var), $\Gamma, x \!:\! T' \vdash x : T'$. As $\downarrow_\beta \subseteq \mathcal{C}$, $T' \,\mathcal{C}\, T$. As $\Gamma \vdash T : s$, by (conv), $\Gamma, x \!:\! T' \vdash x : T$.

(weak) $\dfrac{\Gamma \vdash t : T \quad \Gamma \vdash U : s}{\Gamma, x \!:\! U \vdash t : T}$

(a) If $t \to_\beta t'$ then, by induction hypothesis, $\Gamma \vdash t' : T$. As $\Gamma \vdash U : s$, by (weak), $\Gamma, x \!:\! U \vdash t' : T$.

(b) There is two cases, depending on where takes place the β-reduction :

- $\Gamma \to_\beta \Gamma'$. By induction hypothesis, $\Gamma' \vdash t : T$ and $\Gamma' \vdash U : s$. Therefore, by (weak), $\Gamma', x \!:\! U \vdash t : T$.
- $U \to_\beta U'$. By induction hypothesis, $\Gamma \vdash U' : s$. Therefore, by (weak), $\Gamma, x \!:\! U' \vdash t : T$.

(prod) $\dfrac{\Gamma \vdash T : s_1 \quad \Gamma, x \!:\! T \vdash U : s_2}{\Gamma \vdash (x \!:\! T)U : s_3} \quad ((s_1, s_2, s_3) \in \mathcal{B})$

(a) There is two cases, depending on where takes place the β-reduction :

- $T \to_\beta T'$. By induction hypothesis, $\Gamma \vdash T' : s_1$ and $\Gamma, x : T' \vdash U : s_2$. Therefore, by (prod), we get $\Gamma \vdash (x \!:\! T')U : s_3$.
- $U \to_\beta U'$. By induction hypothesis, $\Gamma, x \!:\! T \vdash U' : s_2$. Therefore, by (prod), $\Gamma \vdash (x \!:\! T)U' : s_3$.

(b) If $\Gamma \to_\beta \Gamma'$ then, by induction hypothesis, $\Gamma' \vdash T : s_1$ and $\Gamma', x \!:\! T \vdash U : s_2$. Therefore, by (prod), $\Gamma' \vdash (x \!:\! T)U : s_3$.

(abs) $\dfrac{\Gamma, x \!:\! T \vdash u : U \quad \Gamma \vdash (x \!:\! T)U : s}{\Gamma \vdash [x \!:\! T]u : (x \!:\! T)U}$

(a) There is two cases, depending on where takes place the β-reduction :

- $T \to_\beta T'$. By induction hypothesis, $\Gamma, x:T' \vdash u : U$ and $\Gamma \vdash (x:T')U : s$. By (abs), $\Gamma \vdash [x:T']u : (x:T')U$. As $(x:T')U \leftarrow \beta(x:T)U$ and $\downarrow_\beta \subseteq \mathcal{C}$, $(x:T')U\ \mathcal{C}\ (x:T)U$. As $\Gamma \vdash (x:T)U : s$, by (conv), $\Gamma \vdash [x:T']u : (x:T)U$.
- $u \to_\beta u'$. By induction hypothesis, $\Gamma, x:T \vdash u' : U$. As $\Gamma \vdash (x:T)U : s$, by (abs), $\Gamma \vdash [x:T]u' : (x:T)U$.

(b) If $\Gamma \to_\beta \Gamma'$ then, by induction hypothesis, $\Gamma', x:T \vdash u : U$ and $\Gamma' \vdash (x:T)U : s$. Therefore, by (abs), $\Gamma' \vdash [x:T]u : (x:T)U$.

(app) $$\frac{\Gamma \vdash t : (x:U)V \quad \Gamma \vdash u : U}{\Gamma \vdash tu : V\{x \mapsto u\}}$$

(a) There is three cases, depending on where takes place the β-reduction :

- $t \to_\beta t'$. By induction hypothesis, $\Gamma \vdash t' : (x:U)V$. As $\Gamma \vdash u : U$, by (app), $\Gamma \vdash t'u : V\{x \mapsto u\}$.
- $u \to_\beta u'$. By induction hypothesis, $\Gamma \vdash u' : U$. By (app), $\Gamma \vdash tu' : V\{x \mapsto u'\}$. As $V\{x \mapsto u'\}^* \leftarrow \beta V\{x \mapsto u\}$ and $\downarrow_\beta \subseteq \mathcal{C}$, $V\{x \mapsto u'\}\ \mathcal{C}\ V\{x \mapsto u\}$. By inversion, there exists s such that $\Gamma \vdash V\{x \mapsto u\} : s$. Therefore, by (conv), $\Gamma \vdash tu' : V\{x \mapsto u\}$.
- $t = [x:U']v$ and $tu \to_\beta v\{x \mapsto u\}$. By inversion, there exists V' such that $\Gamma, x:U' \vdash v : V'$ and $(x:U')V'\ \mathbb{C}^*_\Gamma\ (x:U)V$. By product compatibility, $U'\ \mathbb{C}^*_\Gamma\ U$ and $V'\ \mathbb{C}^*_{\Gamma,x:U}\ V$. By environment conversion, $\Gamma, x:U \vdash v : V'$ and, by (conv), $\Gamma, x:U \vdash v : V$.

(b) If $\Gamma \to_\beta \Gamma'$ then, by induction hypothesis, $\Gamma' \vdash t : (x:U)V \quad \Gamma' \vdash u : U$. Therefore, by (app), $\Gamma' \vdash tu : V\{x \mapsto u\}$.

(conv) $$\frac{\Gamma \vdash t : T \quad T\ \mathcal{C}\ T' \quad \Gamma \vdash T' : s'}{\Gamma \vdash t : T'}$$

(a) If $t \to_\beta t'$ then, by induction hypothesis, $\Gamma \vdash t' : T$. As $T\ \mathcal{C}\ T'$ and $\Gamma \vdash T' : s'$, by (conv), $\Gamma \vdash t' : T'$.

(b) If $\Gamma \to_\beta \Gamma'$ then, by induction hypothesis, $\Gamma' \vdash t : T$ and $\Gamma' \vdash T' : s'$. Therefore, by (conv), $\Gamma' \vdash t : T'$. ■

Chapter 4

Reduction Type Systems (RTS's)

Now, we are going to study the case of TSM's whose conversion relation $\mathcal{C}$ is of the form $\downarrow$ with $\rightarrow$ a reduction relation. We shall call such systems Reduction Type Systems (RTS). Except for ECC [81] which uses a notion of subtyping, all the systems previously mentioned at the beginning of Chapter 3 are RTS's, either because they are defined in this way [54, 15], or because they are defined with $\mathcal{C} = \leftrightarrow^*$ and $\rightarrow$ confluent, which is equivalent [99, 58, 119, 101]. The general study of such systems is justified by the fact that, in [54], the proof that reduction preserves typing uses the fact that $\mathcal{C}$ is of the form $\downarrow$.

The proofs of the Lemmas 41, 47, 50 and 52 are widely inspired from the ones given by H. Geuvers and M.-J. Nederhof [59] or H. Geuvers [58].

4.1 Definition

Definition 32 (RTS) A *pre-RTS* is a TSM whose conversion relation $\mathcal{C}$ is of the form $\downarrow$ with $\rightarrow$ a relation stable by substitution and context. The relation $\rightarrow$ is called the *reduction relation* of the pre-RTS. A pre-RTS is *confluent* if its reduction relation is confluent. An *RTS* is a pre-RTS which is *admissible*, that is, whose reduction relation *preserves typing* : $\Gamma \vdash t : T$ and $t \rightarrow t'$ imply $\Gamma \vdash t' : T$, a property often called *subject reduction*.

Any pre-RTS satisfies the following elementary properties :

Lemma 33 The relation $\mathcal{C} =\downarrow$ is :
– symmetric : $T \ \mathcal{C} \ T'$ implies $T' \ \mathcal{C} \ T$.
– stable by substitution : $T \ \mathcal{C} \ T'$ implies $T\theta \ \mathcal{C} \ T'\theta$.
– stable by context : $T \ \mathcal{C} \ T'$ implies $C[T]_p \ \mathcal{C} \ C[T']_p$.
– preserves sorts : $s \ \mathcal{C} \ s'$ implies $s = s'$.

In ECC, the conversion relation $\mathcal{C}$ is not symmetric and does not preserve sorts. It would be interesting to try to formulate some properties below in the more general framework of Cumulative pure Type Systems (CTS) to which belongs ECC. To this end, the reader is refered to the works of Z. Luo [81], R. Pollack [102] and Barras [11].

Subject reduction can be extended to types, environments and substitutions :

Definition 34 A substitution θ *rewrites* to a substitution θ', $\theta \to \theta'$, if there exists x such that $x\theta \to x\theta'$ and, for all $y \neq x$, $y\theta = y\theta'$. An environment $\Gamma = \vec{x} : \vec{T}$ *rewrites* to an environment Γ', $\Gamma \to \Gamma'$, if $\Gamma' = \vec{x} : \vec{T}'$ and there exists i such that $T_i \to T'_i$ and, for all $j \neq i$, $T_j = T'_j$.

Lemma 35 In an RTS :

(a) if $\Gamma \vdash t : T$ and $T \to T'$ then $\Gamma \vdash t : T'$,
(b) if $\theta : \Gamma \to \Delta$ and $\theta \to \theta'$ then $\theta' : \Gamma \to \Delta$,
(c) if $\Gamma \vdash t : T$ and $\Gamma \to \Gamma'$ then $\Gamma' \vdash t : T$.

Proof.

(a) By correctness of types, either $T = s$ or $\Gamma \vdash T : s$. The case $T = s$ is not possible since s is not reducible. Therefore, $\Gamma \vdash T : s$ and, by subject reduction, $\Gamma \vdash T' : s$. Hence, by (conv), $\Gamma \vdash t : T'$.

(b) By induction on the size of Γ. If Γ is empty, this is immediate. Assume then that $\Gamma = \Gamma', x{:}T$. Since $\theta : \Gamma' \to \Delta$, by induction hypothesis, $\theta' : \Gamma' \to \Delta$. Then it suffices to prove that $\Delta \vdash x\theta' : T\theta'$. As $\theta : \Gamma \to \Delta$, we have $\Delta \vdash x\theta : T\theta$. By subject reduction, $\Delta \vdash x\theta' : T\theta$. After the Environment Lemma, there exists s such that $\Gamma \vdash T : s$. By substitution, $\Delta \vdash T\theta : s$. Since $T\theta \to^* T\theta'$, $T\theta\ \mathcal{C}\ T\theta'$ and, by subject reduction, $\Delta \vdash T\theta' : s$. Therefore, by (conv), $\Delta \vdash x\theta' : T\theta'$.

(c) Assume that $\Gamma = \Gamma_1, x : T, \Gamma_2$ and $\Gamma' = \Gamma_1, x : T', \Gamma_2$. By the Environment Lemma, $\Gamma_1 \vdash T : s$. By subject reduction, $\Gamma_1 \vdash T' : s$. Therefore, $\Gamma\ \mathbb{C}\ \Gamma'$ and, by the Environment conversion Lemma, $\Gamma' \vdash t : T$. ∎

Lemma 36 (Inconvertibility of maximal sorts) In an RTS, if $s\ \mathcal{C}^*_\Gamma\ T$ then $s\ \mathbb{C}^*_\Gamma\ T$. Therefore, if s is maximal then $T = s$.

Proof. By case on the number of conversions between s and T. If $s = T$, this is immediate. Assume then that $s\ \mathcal{C}_\Gamma\ T'\ \mathcal{C}^*_\Gamma\ T$. By definition of $\mathcal{C}_\Gamma$, there exists s' such that $\Gamma \vdash T' : s'$. As $\mathcal{C} = \downarrow$ and s is irreducible, $T' \to^* s$. By subject reduction, $\Gamma \vdash s : s'$ and $s\ \mathbb{C}^*_\Gamma\ T$. ∎

Hence we get the equivalence of the two relations $\mathcal{C}_\Gamma$ and $\mathbb{C}_\Gamma$, and a refinement of the Inversion Lemma for RTS's.

Lemma 37 (Equivalence of $\mathcal{C}_\Gamma$ and $\mathbb{C}_\Gamma$) In an RTS, $\mathcal{C}_\Gamma = \mathbb{C}_\Gamma$.

Proof. First of all, we have $\mathbb{C}_\Gamma \subseteq \mathcal{C}_\Gamma$. We prove the reverse. Assume that $T\ \mathcal{C}_\Gamma\ T'$. As there exists t such that $\Gamma \vdash t : T$, by correctness of types, either T is a maximal sort, or there exists s such that $\Gamma \vdash T : s$. After the previous lemma, T cannot be a maximal sort. Therefore, there exists s such that $\Gamma \vdash T : s$ and $T\ \mathbb{C}_\Gamma\ T'$. ∎

Definition 38 (Regular sort) A sort s is *regular* if, for all $(s_1, s_2, s) \in \mathcal{B}$, $s_2 = s$. A TSM is *regular* if all its sorts are regular.

Most of the PTS's that one can find in the literature are regular. For these systems, it is often made use of the abbreviation $(s_1, s_2) \in \mathcal{B}$ for $(s_1, s_2, s_2) \in \mathcal{B}$ [59, 10].

Lemma 39 (Inversion for RTS's) Assume that $\Gamma \vdash t : T$.

- If $t = s$ then there exists s' such that $(s, s') \in \mathcal{A}$ and $s' \ \mathbb{C}_\Gamma^* \ T$.
- If $t = f(\vec{t})$, $f \in \mathcal{F}^s$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$ then $\vdash \tau_f : s$, $\gamma : (\vec{x} : \vec{T}) \to \Gamma$ and $U\gamma \ \mathbb{C}_\Gamma^* \ T$. Moreover, if s is regular then $\Gamma \vdash U\gamma : s$.
- If $t = x \in \mathcal{X}^s$ then $\Gamma \vdash x\Gamma : s$ and $x\Gamma \ \mathbb{C}_\Gamma^* \ T$.
- If $t = (x:U)V$ then there exists $(s_1, s_2, s_3) \in \mathcal{P}$ such that $\Gamma \vdash U : s_1$, $\Gamma, x:U \vdash V : s_2$ and $s_3 \ \mathbb{C}_\Gamma^* \ T$.
- If $t = [x:U]v$ then there exists V such that $\Gamma, x:U \vdash v : V$ and $(x:U)V \ \mathbb{C}_\Gamma^* \ T$.
- If $t = uv$ then there exists V and W such that $\Gamma \vdash u : (x:V)W$, $\Gamma \vdash v : V$ and $W\{x \mapsto v\} \ \mathbb{C}_\Gamma^* \ T$. Moreover, if $\Gamma \vdash (x:V)W : s$ and s is regular then $\Gamma \vdash W\{x \mapsto v\} : s$.

Proof. The modifications of the cases $t = s$ and $t = (x:U)V$ are a consequence of the inconvertibility of maximal sorts. Hence, we are left to prove the additional properties in case of regular sorts. The property for $t = f(\vec{t})$ can be obtained by iteration from the one for $t = uv$.

Assume that $\Gamma \vdash (x:V)W : s$. By inversion, there exists $(s_1, s_2, s_3) \in \mathcal{P}$ such that $\Gamma, x:V \vdash W : s_2$ and $s_3 \ \mathbb{C}_\Gamma^* \ s$. By preservation of sorts, $s_3 = s$. By regularity, $s_2 = s_3$. Therefore, $\Gamma, x:V \vdash W : s$ and, by substitution, $\Gamma \vdash W\{x \mapsto v\} : s$. ∎

4.2 Logical and functional RTS's

Definition 40 (Functional TSM) A set of rules $\mathcal{B}$ is *functional* if $(s_1, s_2, s_3) \in \mathcal{B}$ and $(s_1, s_2, s_3') \in \mathcal{B}$ imply $s_3 = s_3'$. A TSM is *functional* if $\mathcal{A}$ is a functional relation and $\mathcal{B}$ is functional.

Most of the PTS's one can encounter in the literature are functional.

In a regular TSM, $\mathcal{B}$ is functional. Therefore, for a regular TSM to be functional, it suffices that $\mathcal{A}$ is a functional relation.

Lemma 41 (Convertibility of types) In a logical and functional RTS, if $\Gamma \vdash t : T$ and $\Gamma \vdash t : T'$ then $T \ \mathbb{C}_\Gamma^* \ T'$.

Proof. By induction on t. We follow the notations of the Inversion Lemma.

- $t = s$. By inversion, there exists s_1' and s_2' such that $(s, s_1') \in \mathcal{A}$, $(s, s_2') \in \mathcal{A}$, $s_1' \ \mathbb{C}_\Gamma^* \ T$ and $s_2' \ \mathbb{C}_\Gamma^* \ T'$. By functionality, $s_1' = s_2'$. Therefore, by symmetry, $T \ \mathbb{C}_\Gamma^* \ T'$.
- $t = f(\vec{t})$. By inversion, $U\gamma \ \mathbb{C}_\Gamma^* \ T$ and $U\gamma \ \mathbb{C}_\Gamma^* \ T'$. Therefore, by symmetry, $T \ \mathbb{C}_\Gamma^* \ T'$.
- $t = x$. By inversion, $x\Gamma \ \mathbb{C}_\Gamma^* \ T$ and $x\Gamma \ \mathbb{C}_\Gamma^* \ T'$. Therefore, by symmetry, $T \ \mathbb{C}_\Gamma^* \ T'$.

– $t = (x:U)V$. By inversion, there exists $(s_1, s_2, s_3) \in \mathcal{P}$ and $(s'_1, s'_2, s'_3) \in \mathcal{P}$ such that $\Gamma \vdash U : s_1$, $\Gamma \vdash U : s'_1$, $\Gamma, x:U \vdash V : s_2$, $\Gamma, x:U \vdash V : s'_2$, $s_3 \ \mathbb{C}^*_\Gamma \ T$ and $s'_3 \ \mathbb{C}^*_\Gamma \ T'$. By induction hypothesis, $s_1 \ \mathbb{C}^*_\Gamma \ s'_1$ and $s_2 \ \mathbb{C}^*_{\Gamma,x:U} \ s'_2$. By preservation of sorts, $s_1 = s'_1$ and $s_2 = s'_2$. Therefore, by functionality, $s_3 = s'_3$ and, by symmetry, $T \ \mathbb{C}^*_\Gamma \ T'$.
– $t = [x:U]v$. By inversion, there exists V and V' such that $\Gamma, x:U \vdash v : V$, $\Gamma, x:U \vdash v : V'$, $(x:U)V \ \mathbb{C}^*_\Gamma \ T$ and $(x:U)V' \ \mathbb{C}^*_\Gamma \ T'$. By induction hypothesis, $V \ \mathbb{C}^*_{\Gamma,x:U} \ V'$. By stability by context, $(x:U)V \ \mathbb{C}^*_\Gamma \ (x:U)V'$. Therefore, by symmetry, $T \ \mathbb{C}^*_\Gamma \ T'$.
– $t = uv$. By inversion, there exists V, V', W and W' such that $\Gamma \vdash u : (x:V)W$, $\Gamma \vdash u : (x:V')W'$, $W\{x \mapsto v\} \ \mathbb{C}^*_\Gamma \ T$ and $W'\{x \mapsto v\} \ \mathbb{C}^*_\Gamma \ T'$. By induction hypothesis, $(x:V)W \ \mathbb{C}^*_\Gamma \ (x:V')W'$. By product compatibility, $W \ \mathbb{C}^*_{\Gamma,x:V} \ W'$. By substitution and stability by substitution, $W\{x \mapsto v\} \ \mathbb{C}^*_\Gamma \ W'\{x \mapsto v\}$. Therefore, by symmetry, $T \ \mathbb{C}^*_\Gamma \ T'$. ■

Lemma 42 (Conversion correctness) In a logical and functional RTS, if $\Gamma \vdash T : s$ and $T \ \mathbb{C}_\Gamma \ T'$ then $\Gamma \vdash T' : s$.

Proof. By definition of $\mathbb{C}_\Gamma$, there exists s' such that $\Gamma \vdash T' : s'$. As $\mathcal{C} = \downarrow$, there exists U such that $T \to^* U$ and $T' \to^* U$. By subject reduction, $\Gamma \vdash U : s$ and $\Gamma \vdash U : s'$. By convertibility of types and preservation of sorts, $s = s'$ and $\Gamma \vdash T' : s$. ■

Lemma 43 (Equivalence of $\vdash_s$ and $\vdash$) In a logical and functional RTS, $\vdash_s = \vdash$.

Proof. First of all, it is immediate that $\vdash_s \subseteq \vdash$. We show the inverse by induction on $\Gamma \vdash t : T$. The only difficult case is of course (conv). By induction hypothesis, we have $\Gamma \vdash_s t : T$ and $\Gamma \vdash_s T' : s'$. It is easy to verify that the Substitution Lemma and the correctness of types are also valid for $\vdash_s$. Hence, by correctness of types, either T is a maximal sort, or there exists s such that $\Gamma \vdash_s T : s$. If T is a maximal sort s then $T' \to^* s$ and s is typable, which is excluded. Therefore, $\Gamma \vdash_s T : s$. By convertibility of types and preservation of sorts, $s = s'$ and, by (conv'), $\Gamma \vdash_s t : T'$. ■

Lemma 44 (α-equivalence) In a logical and functional RTS, if $(x:T)U \ \mathbb{C}_\Gamma \ (x':T')U'$ then x and x' are of the same sort and $(x':T')U'$ is α-equivalent to $(x:T')U'\{x' \mapsto x\}$.

Proof. Assume that x is of sort s and x' is of sort s'. By definition of $\mathbb{C}_\Gamma$, we have $\Gamma \vdash (x:T)U : s_3$ and $\Gamma \vdash (x':T')U' : s'_3$. By inversion, we have $\Gamma, x:T \vdash U : s_1$ and $\Gamma, x':T' \vdash U' : s'_1$. By the Environment Lemma, we have $\Gamma \vdash T : s$ and $\Gamma \vdash T' : s'$. By conversion correctness and preservation of sorts, $s = s'$. Therefore x and x' are of the same sort and $(x':T')U'$ is α-equivalent to $(x:T')U'\{x' \mapsto x\}$. ■

Lemma 45 (Maximal sort) In a logical and functional RTS, if s is a maximal sort and $\Gamma \vdash t : s$ then t is of the form $(\vec{x} : \vec{t})s'$. Moreover, if $t \ \mathbb{C}^*_\Gamma \ t'$ then t' is of the form $(\vec{y} : \vec{t'})s'$ with $|\vec{y}| = |\vec{x}|$.

Proof. We prove the first assertion by case on t. Note first of all that there is no s' such that $\Gamma \vdash s : s'$. Otherwise, by inversion, there would exist s'' such that $(s, s'') \in \mathcal{A}$ and $s'' \mathbb{C}_\Gamma^* s'$, which is excluded since s is maximal.

- $t = f(\vec{t})$. Let $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$. By inversion, there exists s' such that $\Gamma \vdash U\gamma : s'$ and $U\gamma \mathbb{C}_\Gamma^* s$. By conversion correctness, $\Gamma \vdash s : s'$. This case is therefore impossible.
- $t = x \in \mathcal{X}^{s'}$. By inversion, $\Gamma \vdash x\Gamma : s'$ and $x\Gamma \mathbb{C}_\Gamma^* s$. By conversion correctness, $\Gamma \vdash s : s'$. This case is therefore impossible.
- $t = [x : U]v$. By inversion, there exists V and s' such that $\Gamma, x : U \vdash v : V$, $\Gamma \vdash (x : U)V : s'$ and $(x : U)V \mathbb{C}_\Gamma^* s$. By conversion correctness, $\Gamma \vdash s : s'$. This case is therefore impossible.
- $t = uv$. By inversion, there exists V, W and s' such that $\Gamma \vdash W\{x \mapsto u\} : s'$ and $W\{x \mapsto u\} \mathbb{C}_\Gamma^* s$. By conversion correctness, $\Gamma \vdash s : s'$. This case is therefore impossible.

We are left with the cases $t = (x : U)V$ and $t = s'$. Therefore t must be of the form $(\vec{x} : \vec{t})s'$.

We now show the second assertion. By conversion correctness, $\Gamma \vdash t' : s$. After the first assertion, t' is of the form $(\vec{y} : \vec{t'})s''$. By product compatibility and α-equivalence, and by exchanging the roles of t and t', we can assume that $\vec{y} = \vec{x}\vec{z}$ and $\vec{t'} = \vec{t}\vec{u}$. Hence, $s' \mathbb{C}_{\Gamma'}^* (\vec{z} : \vec{u})s''$ where $\Gamma' = \vec{x} : \vec{t}$. We then prove by induction on the number of conversions between s' and $(\vec{z} : \vec{u})s''$ that $|\vec{z}| = 0$ and $s' = s''$. If $s' = (\vec{z} : \vec{u})s''$, this is immediate. Assume then that $s' \mathbb{C}_\Gamma' v \mathbb{C}_{\Gamma'}^* (\vec{z} : \vec{u})s''$. By conversion correctness, $\Gamma' \vdash v : s$. Therefore, after the first assertion, v is of the form $(\vec{z'} : \vec{u'})s'''$. As $\mathcal{C} = \downarrow$ and s' is irreducible, $v \rightarrow^* s'$. Therefore $|\vec{z'}| = 0$ and $s''' = s'$. So, $v = s'$ and, by induction hypothesis, $|\vec{z}| = 0$ et $s'' = s'$. ■

4.3 Logical and injective RTS's

Definition 46 (Injective TSM) A set of rules $\mathcal{B}$ is *injective* if $(s_1, s_2, s_3) \in \mathcal{B}$ and $(s_1, s_2', s_3) \in \mathcal{B}$ imply $s_2 = s_2'$. A TSM is *injective* if $\mathcal{A}$ is a functional and injective relation and $\mathcal{B}$ is functional and injective.

In a regular TSM, $\mathcal{B}$ is functional and injective. Therefore, for a regular TSM to be injective, it suffices that $\mathcal{A}$ is a functional and injective relation.

Lemma 47 (Separation) In a logical and injective RTS, if $s_1 \neq s_2$ then, for all $i \in \{0, 1\}$, $\mathbb{T}_i^{s_1} \cap \mathbb{T}_i^{s_2} = \emptyset$.

Proof. We show that $t \in \mathbb{T}_i^{s_1} \cap \mathbb{T}_i^{s_2}$ implies $s_1 = s_2$, by induction on t.

Case i = 0. There exists Γ_j such that $\Gamma_j \vdash t : s_j$.

- $t = s$. By inversion, there exists s_j' such that $(s, s_j') \in \mathcal{A}$ and $s_j' \mathbb{C}_{\Gamma_j}^* s_j$. By functionality, $s_1' = s_2'$. Let s' be the sort $s_1' = s_2'$. Then, $s' \mathbb{C}_{\Gamma_j}^* s_j$. Therefore, by preservation of sorts, $s_1 = s_2 = s'$.
- $t = f(\vec{t})$, $f \in \mathcal{F}^s$ and $\tau_f = (\vec{x} : \vec{T})U$. Let $\Gamma = \vec{x} : \vec{T}$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$. By inversion, $\vdash \tau_f : s$, $\gamma : \Gamma \rightarrow \Gamma_j$ and $U\gamma \mathbb{C}_{\Gamma_j}^* s_j$. By inversion again, there exists s'

such that $\Gamma \vdash U : s'$. By substitution, $\Gamma_j \vdash U\gamma : s'$. By conversion correctness, $\Gamma_j \vdash s_j : s'$. By inversion and preservation of sorts, $(s_j, s') \in \mathcal{A}$. Therefore, by injectivity, $s_1 = s_2$.

– $t = x \in \mathcal{X}^s$. By inversion, $\Gamma_j \vdash x\Gamma_j : s$ and $x\Gamma_j \ \mathbb{C}^*_{\Gamma_j} \ s_j$. By conversion correctness, $\Gamma_j \vdash s_j : s$. By inversion and preservation of sorts, $(s_j, s) \in \mathcal{A}$. Therefore, by injectivity, $s_1 = s_2$.

– $t = (x\!:\!U)V$. By inversion, there exists $(s^1_j, s^2_j, s^3_j) \in \mathcal{P}$ such that $\Gamma_j \vdash U : s^1_j$, $\Gamma_j, x\!:\!U \vdash V : s^2_j$ and $s^3_j \ \mathbb{C}^*_{\Gamma_j} \ s_j$. By induction hypothesis, $s^1_1 = s^1_2$ and $s^2_1 = s^2_2$. Therefore, by functionality, $s^3_1 = s^3_2$. Let s be the sort $s^3_1 = s^3_2$. Then, $s \ \mathbb{C}^*_{\Gamma_j} \ s_j$ and, by preservation of sorts, $s_1 = s_2 = s$.

– $t = [x\!:\!U]v$. By inversion, there exists V_j and s^4_j such that $\Gamma_j, x\!:\!U \vdash v : V_j$, $\Gamma_j \vdash (x : U)V_j : s^4_j$ and $(x : U)V_j \ \mathbb{C}^*_{\Gamma_j} \ s_j$. By inversion again, there exists $(s^1_j, s^2_j, s^3_j) \in \mathcal{P}$ such that $\Gamma_j \vdash U : s^1_j$, $\Gamma_j, x : U \vdash V_j : s^2_j$ and $s^3_j \ \mathbb{C}^*_{\Gamma_j} \ s^4_j$. By preservation of sorts, $s^3_j = s^4_j$. By induction hypothesis, $s^1_1 = s^1_2$ and $s^2_1 = s^2_2$. Therefore, by functionality, $s^3_1 = s^3_2$. Let s be the sort $s^3_1 = s^3_2 = s^4_1 = s^4_2$. By conversion correctness, $\Gamma \vdash s_j : s$. By inversion and preservation of sorts, $(s_j, s) \in \mathcal{A}$. Therefore, by injectivity, $s_1 = s_2$.

– $t = uv$. By inversion, there exists V_j and W_j such that $\Gamma_j \vdash u : (x_j : V_j)W_j$, $\Gamma_j \vdash v : V_j$ and $W_j\{x_j \mapsto v\} \ \mathbb{C}^*_{\Gamma_j} \ s_j$. Let $\Gamma'_j = \Gamma_j, x_j : V_j$ and $\theta_j = \{x_j \mapsto v\}$. By correctness of types, there exists s'_j such that $\Gamma_j \vdash (x_j : V_j)W_j : s'_j$. By induction hypothesis on u, $s'_1 = s'_2$. Let $s' = s'_1 = s'_2$. By inversion, there exists $(s^1_j, s^2_j, s^3_j) \in \mathcal{P}$ such that $\Gamma_j \vdash V_j : s^1_j$, $\Gamma'_j \vdash W_j : s^2_j$ and $s^3_j \ \mathbb{C}^*_{\Gamma_j} \ s'$. By preservation of sorts, $s^3_j = s'$. By induction hypothesis on v, $s^1_1 = s^1_2$. Therefore, by injectivity, $s^2_1 = s^2_2$. Let $s'' = s^2_1 = s^2_2$. As $\theta_j : \Gamma'_j \mapsto \Gamma_j$, by substitution, $\Gamma_j \vdash W_j\theta_j : s''$. By conversion correctness, $\Gamma_j \vdash s_j : s''$. By inversion and preservation of sorts, $(s_j, s'') \in \mathcal{A}$. Therefore, by injectivity, $s_1 = s_2$.

Case $\underline{i = 1}$. There exists Γ_j and T_j such that $\Gamma_j \vdash t : T_j$ and $\Gamma_j \vdash T_j : s_j$.

– $t = s$. After case $i = 0$, there exists s' such that $s' \ \mathbb{C}^*_{\Gamma_j} \ T_j$. By conversion correctness, $\Gamma_j \vdash s' : s_j$. By inversion and preservation of sorts, $(s', s_j) \in \mathcal{A}$. Therefore, by functionality, $s_1 = s_2$.

– $t = f(\vec{t})$. After case $i = 0$, there exists s' such that $\Gamma_j \vdash T_j : s'$. By convertibility of types and preservation of sorts, $s_1 = s_2 = s'$.

– $t = x \in \mathcal{X}^s$. After case $i = 0$, $\Gamma_j \vdash T_j : s$. Therefore, by convertibility of types and preservation of sorts, $s_1 = s_2 = s$.

– $t = (x\!:\!U)V$. After case $i = 0$, there exists s such that $s \ \mathbb{C}^*_{\Gamma_j} \ T_j$. By conversion correctness, $\Gamma_j \vdash s : s_j$. By inversion and preservation of sorts, $(s, s_j) \in \mathcal{A}$. Therefore, by functionality, $s_1 = s_2$.

– $t = [x : U]v$. After case $i = 0$, there exists s such that $\Gamma_j \vdash T_j : s$. By convertibility of types and preservation of sorts, $s_1 = s_2 = s$.

– $t = uv$. After case $i = 0$, there exists s'' such that $\Gamma_j \vdash T_j : s''$. By convertibility of types and preservation of sorts, $s_1 = s_2 = s'$. ■

Lemma 48 (Classification) In a logical and injective RTS, either $(s_1, s_2) \in \mathcal{A}$ and $\mathbb{T}_0^{s_1} \subseteq \mathbb{T}_1^{s_2}$, or $(s_1, s_2) \notin \mathcal{A}$ and $\mathbb{T}_0^{s_1} \cap \mathbb{T}_1^{s_2} = \emptyset$.

Proof. If $(s_1, s_2) \in \mathcal{A}$, it is clear that $\mathbb{T}_0^{s_1} \subseteq \mathbb{T}_1^{s_2}$. We then prove that $t \in \mathbb{T}_0^{s_1} \cap \mathbb{T}_1^{s_2}$ implies $(s_1, s_2) \in \mathcal{A}$, by induction on t. Let Γ, Γ' and T such that $\Gamma \vdash t : s_1$, $\Gamma' \vdash t : T$ and $\Gamma' \vdash T : s_2$.

- $t = s$. By inversion, there exists s_1' and s_2' such that $(s, s_1') \in \mathcal{A}$, $s_1' \ \mathbb{C}_\Gamma^* \ s_1$, $(s, s_2') \in \mathcal{A}$ and $s_2' \ \mathbb{C}_{\Gamma'}^* \ T$. By preservation of sorts, $s_1' = s_1$. By functionality, $s_1' = s_2'$. By conversion correctness, $\Gamma \vdash s_2' : s_2$. Therefore, by inversion and preservation of sorts, $(s_2', s_2) \in \mathcal{A}$ and $(s_1, s_2) \in \mathcal{A}$.
- $t = f(\vec{t})$. Let $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{t}\}$. By inversion, there exists s_1' and s_2' such that $\Gamma \vdash U\gamma : s_1'$, $U\gamma \ \mathbb{C}_\Gamma^* \ s_1$, $\Gamma' \vdash U\gamma : s_2'$ and $U\gamma \ \mathbb{C}_{\Gamma'}^* \ T$. By conversion correctness, $\Gamma \vdash s_1 : s_1'$. Therefore, by inversion and preservation of sorts, $(s_1, s_1') \in \mathcal{A}$. By separation, $s_1' = s_2'$. By conversion correctness, $\Gamma' \vdash T : s_2'$. Therefore, by separation, $s_2' = s_2$ and $(s_1, s_2) \in \mathcal{A}$.
- $t = x \in \mathcal{X}^s$. By inversion, $\Gamma \vdash x\Gamma : s$, $x\Gamma \ \mathbb{C}_\Gamma^* \ s_1$, $\Gamma' \vdash x\Gamma' : s$ and $x\Gamma' \ \mathbb{C}_{\Gamma'}^* \ T$. By conversion correctness, $\Gamma \vdash s_1 : s$. By inversion and preservation of sorts, $(s_1, s) \in \mathcal{A}$. By conversion correctness, $\Gamma' \vdash T : s$. Therefore, by separation, $s = s_2$ and $(s_1, s_2) \in \mathcal{A}$.
- $t = (x:U)V$. By inversion, there exists (s_a, s_b, s_c) and $(s_a', s_b', s_c') \in \mathcal{P}$ such that $\Gamma \vdash U : s_a$, $\Gamma, x:U \vdash V : s_b$, $s_c \ \mathbb{C}_\Gamma^* \ s_1$, $\Gamma' \vdash U : s_a'$, $\Gamma', x:U \vdash V : s_b'$ and $s_c' \ \mathbb{C}_{\Gamma'}^* \ T$. By preservation of sorts, $s_c = s_1$. By conversion correctness, $\Gamma' \vdash s_c' : s_2$. By inversion and preservation of sorts, $(s_c', s_2) \in \mathcal{A}$. By separation, $s_a = s_a'$ and $s_b = s_b'$. Therefore, by functionality, $s_c = s_c'$ and $(s_1, s_2) \in \mathcal{A}$.
- $t = [x:U]v$. By inversion, there exists V, s, V' and s' such that $\Gamma, x:U \vdash v : V$, $\Gamma \vdash (x:U)V : s$, $(x:U)V \ \mathbb{C}_\Gamma^* \ s_1$, $\Gamma', x:U \vdash v : V'$, $\Gamma' \vdash (x:U)V' : s'$ and $(x:U)V' \ \mathbb{C}_{\Gamma'}^* \ T$. By conversion correctness, $\Gamma \vdash s_1 : s$ and $\Gamma' \vdash T : s'$. By inversion and preservation of sorts, $(s_1, s) \in \mathcal{A}$. By separation $s' = s_2$. By inversion again, there exists (s_a, s_b, s_c) and $(s_a', s_b', s_c') \in \mathcal{P}$ such that $\Gamma \vdash U : s_a$, $\Gamma, x:U \vdash V : s_b$, $s_c \ \mathbb{C}_\Gamma^* \ s$, $\Gamma' \vdash U : s_a'$ and $\Gamma', x:U \vdash V' : s_b'$ and $s_c' \ \mathbb{C}_{\Gamma'}^* \ s'$. By preservation of sorts, $s_c = s$ and $s_c' = s'$. By separation, $s_a = s_a'$ and $s_b = s_b'$. Therefore, by functionality, $s_c = s_c'$ and $(s_1, s_2) \in \mathcal{A}$.
- $t = uv$. By inversion, there exists V, W, s, V' and W' and s' such that $\Gamma \vdash u : (x:V)W$, $\Gamma \vdash W\{x \mapsto v\} : s$, $W\{x \mapsto v\} \ \mathbb{C}_\Gamma^* \ s_1$, $\Gamma' \vdash u : (x:V')W'$, $\Gamma' \vdash W'\{x \mapsto v\} : s'$ and $W'\{x \mapsto v\} \ \mathbb{C}_{\Gamma'}^* \ T$. By conversion correctness, $\Gamma \vdash s_1 : s$ and $\Gamma' \vdash T : s'$. By inversion and preservation of sorts, $(s_1, s) \in \mathcal{A}$. By separation, $s = s'$ and $s' = s_2$. Therefore, $(s_1, s_2) \in \mathcal{A}$. ∎

Remark 49 (Typing classes)

With the correctness of types, we have seen that a typable term is necessarily in one of the sets $\mathbb{T}_i^s$ where $i \in \{0, 1\}$ and $s \in \mathcal{S}$. With the Separation and Classification Lemmas, we can describe the relations between these sets more precisely.

In an injective TSM, the set of axioms $\mathcal{A}$ is necessarily an union of disjoint maximal "chains", that is, sets $\mathcal{A}'$ such that :

– if $(s_1, s_2) \in \mathcal{A}'$ and $(s_2, s_3) \in \mathcal{A}$ then $(s_2, s_3) \in \mathcal{A}'$,
– if $(s_2, s_3) \in \mathcal{A}'$ and $(s_1, s_2) \in \mathcal{A}$ then $(s_1, s_2) \in \mathcal{A}'$.

For example, in the case where $\mathcal{A}$ is finite, a maximal chain is of the form $\{(s_1, s_2), (s_2, s_3), \ldots, (s_n, s_{n+1})\}$ with $s_1, \ldots, s_n$ distinct from one another. For such a chain, we obtain n classes $\mathbb{T}_1^{s_1}$, $\mathbb{T}_1^{s_2}$, ..., $\mathbb{T}_1^{s_n}$, plus two classes $\mathbb{T}_1^{s_{n+1}}$ and $\mathbb{T}_0^{s_{n+1}}$ if s_{n+1} is distinct from the other s_i's. Finally, a sort s which does not belong to any axiom gives two classes, $\mathbb{T}_1^s$ and $\mathbb{T}_0^s$.

4.4 Confluent RTS's

In the following, we prove results about the dependence of types with respect to variables and symbols. The first result, dependence with respect to variables, is better known under the name of "Strengthening Lemma". We give a proof of this lemma in the case of a functional (and confluent) RTS inspired from the one of H. Geuvers and M.-J. Nederhof [59]. L. S. van Benthem Jutting [115] proved the same lemma for all PTS's. It would be interesting to examine his proof to adapt it to the case of RTS's.

Lemma 50 (Dependence w.r.t. variables) In a confluent and functional RTS, if $\Delta, z\!:\!V, \Delta' \vdash t : T$ and $z \notin \mathrm{FV}(\Delta', t)$ then there exists T' such that $T \to^* T'$ and $\Delta, \Delta' \vdash t : T'$.

Proof. By induction on $\Delta, z\!:\!V, \Delta' \vdash t : T$.

(ax) Impossible.

(symb) Let $\Gamma = \Delta, z : V, \Delta'$. We prove the property for $U\gamma$ itself. By induction hypothesis, for all i, there exists T_i' such that $T_i\gamma \to^* T_i'$ and $\Delta, \Delta' \vdash t_i : T_i'$. We prove that $\gamma : \Gamma_f \to \Delta, \Delta'$. Let $\gamma_i = \{x_1 \mapsto t_1, \ldots, x_i \mapsto t_i\}$ and $\Gamma_i = x_1 : T_1, \ldots, x_i : T_i$. We prove that $\gamma_i : \Gamma_i \to \Delta, \Delta'$ by induction on i. For $i = 0$, there is nothing to prove. Assume then that $\gamma_i : \Gamma_i \to \Delta, \Delta'$ and let us prove that $\gamma_{i+1} : \Gamma_{i+1} \to \Delta, \Delta'$. As $\vdash \tau_f : s$, by inversion, for all j, there exists s_j such that $\Gamma_{j-1} \vdash T_j : s_j$. Hence, by the Environment Lemma, $\mathrm{FV}(T_j) \subseteq \{x_1, \ldots, x_{j-1}\}$. Therefore, for all $j \le i+1$, $T_j\gamma_{i+1} = T_j\gamma_i$. So, $\gamma_{i+1} : \Gamma_i \to \Delta, \Delta'$ and we are left to prove that $\Delta, \Delta' \vdash t_{i+1} : T_{i+1}\gamma_i$. We have $\Delta, \Delta' \vdash t_{i+1} : T'_{i+1}$. As $\Gamma_i \vdash T_{i+1} : s_{i+1}$ and $\gamma_i : \Gamma_i \to \Delta, \Delta'$, by substitution, $\Delta, \Delta' \vdash T_{i+1}\gamma_i : s_{i+1}$. Therefore, by (conv), $\Delta, \Delta' \vdash t_{i+1} : T_{i+1}\gamma_i$ and $\gamma_{i+1} : \Gamma_{i+1} \to \Delta, \Delta'$. Finally, $\gamma = \gamma_n : \Gamma_f \to \Delta, \Delta'$. As $\Gamma_f \vdash U : s$, by substitution, $\Delta, \Delta' \vdash U\gamma : s$.

(var) Let $\Gamma = \Delta, z\!:\!V, \Delta'$. By induction hypothesis, $\Delta, \Delta' \vdash T : s$. Therefore, by (var), $\Delta, \Delta', x : T \vdash x : T$.

(weak) If $z = x$, T itself satisfies the property since $\Gamma \vdash t : T$. Otherwise, let $\Gamma = \Delta, z : V, \Delta'$. By induction hypothesis, there exists T' such that $T \to^* T'$, $\Delta, \Delta' \vdash t : T'$ and $\Delta, \Delta' \vdash U : s$. Therefore, by (weak), $\Delta, \Delta', x\!:\!U \vdash t : T'$.

(prod) Let $\Gamma = \Delta, z\!:\!V, \Delta'$. By induction hypothesis, $\Delta, \Delta' \vdash T : s_1$ and $\Delta, \Delta', x : T \vdash U : s_2$. Therefore, by (prod), $\Delta, \Delta' \vdash (x\!:\!T)U : s_3$.

(abs) Let $\Gamma = \Delta, z : V, \Delta'$. By induction hypothesis, there exists U' such that $U \to^* U'$ and $\Delta, \Delta', x\!:\!T \vdash u : U'$. We prove that $\Delta, \Delta' \vdash (x\!:\!T)U' : s$. Then, by (abs), $\Delta, \Delta' \vdash [x:T]u : (x:T)U'$. As $\Gamma \vdash (x:T)U : s$, by inversion, there exists $(s_1, s_2, s) \in \mathcal{P}$ such that $\Gamma \vdash T : s_1$ and $\Gamma, x:T \vdash U : s_2$. As $z \notin \mathrm{FV}(T)$, by induction hypothesis, $\Delta, \Delta' \vdash T : s_1$. As $\Delta, \Delta', x:T \vdash u : U'$, by correctness of types, either $U' = s'$ or $\Delta, \Delta', x : T \vdash U' : s'$. Assume that $U' = s'$. As $\Gamma, x:T \vdash U : s_2$, by subject reduction, $\Gamma, x:T \vdash U' : s_2$. Therefore $(s', s_2) \in \mathcal{A}$ and $\Delta, \Delta', x:T \vdash U' : s_2$. If now $\Delta, \Delta', x:T \vdash U' : s'$ then, by convertibility of types, $s' = s_2$. Hence, in all cases, $\Gamma, x:T \vdash U' : s_2$. Therefore, by (prod), $\Delta, \Delta' \vdash (x:T)U' : s$.

(app) Let $\Gamma = \Delta, z:V, \Delta'$. By induction hypothesis, there exists U'_1, U'_2 and V such that $U \to^* U'_1$, $U \to^* U'_2$, $V \to^* V'$, $\Delta, \Delta' \vdash t : (x:U'_1)V'$ and $\Delta, \Delta' \vdash u : U'_2$. By confluence, there exists U'' such that $U'_1 \to^* U''$ and $U'_2 \to^* U''$. By subject reduction, $\Delta, \Delta' \vdash t : (x:U'')V'$ and $\Delta, \Delta' \vdash u : U''$. Therefore, by (app), $\Delta, \Delta' \vdash tu : V'\{x \mapsto u\}$.

(conv) By induction hypothesis, there exists T'' such that $T \to^* T''$ and $\Delta, \Delta' \vdash t : T''$. By confluence, there exists T''' such that $T'' \to^* T'''$ and $T' \to^* T'''$. By subject reduction, $\Delta, \Delta' \vdash t : T'''$. ■

Corollary 51 In a confluent and functional RTS, if $\Delta, z:V, \Delta' \vdash t : T$ and $z \notin \mathrm{FV}(\Delta', t, T)$ then $\Delta, \Delta' \vdash t : T$.

Proof. After the lemma, there exists T' such that $T \to^* T'$ and $\Delta, \Delta' \vdash t : T'$. By correctness of types, either T is a maximal sort and $T' = T$, or $\Delta, z:V, \Delta' \vdash T : s$. Then, after the lemma, $\Delta, \Delta' \vdash T : s$. Therefore, by (conv), $\Delta, \Delta' \vdash t : T$. ■

Lemma 52 (Strong permutation) If $\Gamma, y:A, z:B, \Gamma' \vdash t : T$ and $y \notin \mathrm{FV}(B)$ then $\Gamma, z:B, y:A, \Gamma' \vdash t : T$.

Proof. Let $\Delta = \Gamma, y:A, z:B, \Gamma'$ and $\Delta' = \Gamma, z:B, y:A, \Gamma'$. By transitivity, it suffices to prove that Δ' is valid and that $\Delta' \vdash \Delta$. To this end, it suffices to prove that Δ' is valid. By the Environment Lemma, we have $\Gamma \vdash A : s$ and $\Gamma, y:A \vdash B : s'$. After the previous lemma, $\Gamma \vdash B : s'$. Therefore, $\Gamma, z:B$ is valid and, by weakening, $\Gamma, z:B, y:A$ too. Assume that $\Gamma' = \vec{x} : \vec{T}$ and let $\Delta_i = \Gamma, y:A, z:B, x_1:T_1, \ldots, x_i:T_i$ and $\Delta'_i = \Gamma, z:B, y:A, x_1:T_1, \ldots, x_i:T_i$. We prove by induction on i that Δ'_i is valid. We have already proved that Δ'_0 is valid. Assume that Δ'_i is valid. By the Environment Lemma, $\Delta_i \vdash T_{i+1} : s_{i+1}$. As $\Delta'_i \vdash \Delta_i$, $\Delta'_i \vdash T_{i+1} : s_{i+1}$ and Δ'_{i+1} is valid. Therefore, Δ' is valid and $\Delta' \vdash t : T$. ■

Definition 53 (Compatibility w.r.t. a quasi-ordering) Let $\geq$ be a quasi-ordering on $\mathcal{F}$. Given a symbol g, we will denote by $\vdash_g$ the typing relation of the RTS whose symbols are strictly smaller than g.

- $\to$ is *compatible* with $\geq$ if, for all symbol g and all term t, t', if all the symbols in t are strictly smaller than g and $t \to t'$ then the symbols in t' are strictly smaller than g.
- τ is *compatible* with $\geq$ if, for all symbol g, all the symbols in τ_g are smaller than g.

Lemma 54 (Dependence w.r.t. symbols) Consider a confluent and functional RTS and $\geq$ a quasi-ordering on $\mathcal{F}$ such that $\rightarrow$ and τ are compatible with $\geq$. If $\Gamma \vdash t : T$ and the symbols in Γ and t are strictly smaller than g then there exists T' such that $T \rightarrow^* T'$ and $\Gamma \vdash_g t : T'$.

Proof. By induction on $\Gamma \vdash t : T$.

(ax) Immediate.

(symb) We prove that $\Gamma \vdash_g f(\vec{t}) : U\gamma$. By induction hypothesis, for all i, there exists T_i' such that $T_i\gamma \rightarrow^* T_i'$ and $\Gamma \vdash_g t_i : T_i'$. We prove that $\gamma : \Gamma_f \rightarrow \Gamma$ in $\vdash_g$. Let $\gamma_i = \{x_1 \mapsto t_1, \ldots, x_i \mapsto t_i\}$ and $\Gamma_i = x_1 : T_1, \ldots, x_i : T_i$. We prove that $\gamma_i : \Gamma_i \rightarrow \Gamma$ by induction on i. For $i = 0$, there is nothing to prove. Assume then that $\gamma_i : \Gamma_i \rightarrow \Gamma$ and let us prove that $\gamma_{i+1} : \Gamma_{i+1} \rightarrow \Gamma$. As τ is compatible with $\geq$, by induction hypothesis, $\vdash_g \tau_f : s$. By inversion, for all j, there exists s_j such that $\Gamma_{j-1} \vdash_g T_j : s_j$. Hence, by the Free variables Lemma, $\mathrm{FV}(T_j) \subseteq \{x_1, \ldots, x_{j-1}\}$. Therefore, for all $j \leq i+1$, $T_j\gamma_{i+1} = T_j\gamma_i$. So, $\gamma_{i+1} : \Gamma_i \rightarrow \Gamma$ and we are left to prove that $\Gamma \vdash_g t_{i+1} : T_{i+1}\gamma_i$. We have $\Gamma \vdash_g t_{i+1} : T_{i+1}'$. As $\Gamma_i \vdash_g T_{i+1} : s_{i+1}$ and $\gamma_i : \Gamma_i \rightarrow \Gamma$, by substitution, $\Gamma \vdash_g T_{i+1}\gamma_i : s_{i+1}$. Therefore, by (conv), $\Gamma \vdash_g t_{i+1} : T_{i+1}\gamma_i$ and $\gamma_{i+1} : \Gamma_{i+1} \rightarrow \Gamma$. Finally, $\gamma = \gamma_n : \Gamma_f \rightarrow \Gamma$. As $\Gamma_f \vdash_g U : s$, by substitution, $\Gamma \vdash_g U\gamma : s$.

(var) By induction hypothesis, $\Gamma \vdash_g T : s$. Therefore, by (var), $\Gamma, x\!:\!T \vdash_g x : T$.

(weak) By induction hypothesis, there exists T' such that $T \rightarrow^* T'$, $\Gamma \vdash_g t : T'$ and $\Gamma \vdash_g U : s$. Therefore, by (weak), $\Gamma, x\!:\!U \vdash_g t : T'$.

(prod) By induction hypothesis, $\Gamma \vdash_g T : s_1$ and $\Gamma, x\!:\!T \vdash_g U : s_2$. Therefore, by (prod), $\Gamma \vdash_g (x\!:\!T)U : s_3$.

(abs) By induction hypothesis, there exists U' such that $U \rightarrow^* U'$ and $\Gamma, x\!:\!T \vdash_g u : U'$. We prove that $\Gamma \vdash_g (x\!:\!T)U' : s$. Then, by (abs), $\Gamma \vdash_g [x\!:\!T]u : (x\!:\!T)U'$. As $\Gamma \vdash (x\!:\!T)U : s$, by inversion, there exists $(s_1, s_2, s) \in \mathcal{P}$ such that $\Gamma \vdash T : s_1$ and $\Gamma, x\!:\!T \vdash U : s_2$. By induction hypothesis, $\Gamma \vdash_g T : s_1$. As $\Gamma, x\!:\!T \vdash_g u : U'$, by correctness of types, either $U' = s'$ or $\Gamma, x\!:\!T \vdash_g U' : s'$. Assume that $U' = s'$. As $\Gamma, x\!:\!T \vdash U : s_2$, by subject reduction, $\Gamma, x\!:\!T \vdash U' : s_2$. Therefore $(s', s_2) \in \mathcal{A}$ and $\Gamma, x\!:\!T \vdash_g U' : s_2$. If now $\Gamma, x\!:\!T \vdash_g U' : s'$ then, by convertibility of types, $s' = s_2$. Hence, in all cases, $\Gamma, x\!:\!T \vdash_g U' : s_2$. Therefore, by (prod), $\Gamma \vdash_g (x\!:\!T)U' : s$.

(app) By induction hypothesis, there exists U_1', U_2' and V such that $U \rightarrow^* U_1'$, $U \rightarrow^* U_2'$, $V \rightarrow^* V'$, $\Gamma \vdash_g t : (x\!:\!U_1')V'$ and $\Gamma \vdash_g u : U_2'$. By confluence, there exists U'' such that $U_1' \rightarrow^* U''$ and $U_2' \rightarrow^* U''$. By subject reduction, $\Gamma \vdash_g t : (x\!:\!U'')V'$ and $\Gamma \vdash_g u : U''$. Therefore, by (app), $\Gamma \vdash_g tu : V'\{x \mapsto u\}$.

(conv) By induction hypothesis, there exists T'' such that $T \rightarrow^* T''$ and $\Gamma \vdash_g t : T''$. By confluence, there exists T''' such that $T'' \rightarrow^* T'''$ and $T' \rightarrow^* T'''$. By subject reduction, $\Gamma \vdash_g t : T'''$. As $\rightarrow$ is compatible with $\geq$, the symbols in T''' are strictly smaller than g. ∎

Chapter 5

Algebraic Type Systems (ATS's)

Now, we are going to study the case of RTS's whose reduction relation is made of β-reduction and rewrite rules. But, before, we must properly define what rewriting means in a strongly typed higher-order framework.

In first-order frameworks, that is, within a first-order term algebra, a rewrite rule is generally defined as a pair $l \rightarrow r$ of terms such that l is not a variable and the variables occurring in r also occur in l (otherwise, rewriting does not terminate). Then, one says that a term t rewrites to a term t' at position p if there exists a substitution σ such that $t|_p = l\sigma$ (one says that $t|_p$ matches l) and $t' = t[r\sigma]_p$ (the subterm of t at position p, $l\sigma$, is replaced by $r\sigma$). The reader is invited to look at, for example, [46, 3] to get more details on first-order rewriting.

Here, we are going to consider a very similar rewriting mechanism by restricting left-hand sides of rules to belong to the first-order-like term algebra generated from $\mathcal{F}$ and $\mathcal{X}$. On the other hand, right-hand sides can be arbitrary. This is a particular the case of *Combinatory Reduction System* (CRS) [1] of W. Klop [79] for which it is not necessary to use *higher-order pattern matching à la* Klop or *à la* Miller [91, 95].

However, we proved in [21] that a weaker version of the termination criteria that we are going to present in next chapter can be adapted, in case of simply-typed λ-calculus, to rewriting with higher-order matching *à la* Klop or *à la* Miller. It would therefore be interesting to try to define a notion of rewriting with higher-order matching in case of polymorphic and dependent types, and to study if our termination criteria can also be adapted to this notion of rewriting.

Definition 55 (Algebraic terms) A term is *algebraic* if it is a variable or of the form $f(\vec{t})$ with all the t_i's themselves algebraic. We denote by $\mathcal{T}(\mathcal{F}, \mathcal{X})$ the set of algebraic terms built from $\mathcal{F}$ and $\mathcal{X}$, and by $\mathbb{T}(\mathcal{F}, \mathcal{X})$ the set of typable algebraic terms.

Definition 56 (Rewriting) A *rewrite rule* is a pair of terms $l \rightarrow r$ such that l is an algebraic term distinct from a variable and $\mathrm{FV}(r) \subseteq \mathrm{FV}(l)$. A rule $l \rightarrow r$ is *left-linear* if no variable occurs more than once in l.

[1]To see this, it suffices to translate $[x:T]u$ by $\Lambda(T, [x]u)$, $(x:T)U$ by $\Pi(T, [x]U)$ and uv by $@(u, v)$, where Λ, Π and $@$ are symbols of arity 2 and $[\_]\_$ is the abstraction operator of CRS's.

Given a set of rewrite rules $\mathcal{R}$, *$\mathcal{R}$-reduction* $\rightarrow_{\mathcal{R}}$ is the smallest relation containing $\mathcal{R}$ and stable by substitution and context. A term of the form $l\sigma$ with $l \rightarrow r \in \mathcal{R}$ and σ a substitution is an *$\mathcal{R}$-redex*.

Given a set of symbols $\mathcal{G}$, we denote by $\mathcal{R}_{\mathcal{G}}$ the set of rules which *define* a symbol in $\mathcal{G}$, that is, the set of rules such that the head symbol of the left-hand side is a symbol of $\mathcal{G}$.

A symbol f is *constant* if $\mathcal{R}_{\{f\}} = \emptyset$, otherwise it is (partially) *defined*. We denote by $\mathcal{CF}$ the set of constant symbols and by $\mathcal{DF}$ the set of defined symbols.

Definition 57 (ATS) An *ATS* is a pre-RTS whose reduction relation $\rightarrow$ is of the form $\rightarrow_{\mathcal{R}} \cup \rightarrow_{\beta}$ with $\mathcal{R}$ a set of rewrite rules.

Now that we have introduced our notion of rewriting, we can wonder under which conditions it has the subject reduction property.

With first-order rewriting in sorted algebras, for rewriting to preserve the sort of terms, it suffices that, for all rules, both sides of the rule have the same sort.

Carried over to type systems, this condition gives : there exists an environment Γ and a type T such that $\Gamma \vdash l : T$ and $\Gamma \vdash r : T$. This condition is the one which has been taken in all previous work combining typed λ-calculus and rewriting.

However, this condition has an important drawback. With polymorphic or dependent types, it leads to strongly non-left-linear rules. This has two important consequences. First, rewriting is strongly slowed down because of the necessary equality tests. Second, it is more difficult to prove confluence with non-left-linear rules.

Let us take the example of the concatenation of two polymorphic lists in the Calculus of Constructions ($\mathcal{S} = \{\star, \square\}$, $\mathcal{A} = \{(\star, \square)\}$ and $\mathcal{B} = \{(s_1, s_2, s_3) \in \mathcal{S}^3 \mid s_2 = s_3\}$) :

- $list \in \mathcal{F}_1^{\square}$ with $\tau_{list} = \star \rightarrow \star$ the type of polymorphic lists,
- $nil \in \mathcal{F}_1^{\star}$ with $\tau_{nil} = (A\!:\!\star)list(A)$ the empty list,
- $cons \in \mathcal{F}_3^{\star}$ with $\tau_{cons} = (A:\star)A \rightarrow list(A) \rightarrow list(A)$ the function adding an element at the head of a list,
- $app \in \mathcal{F}_3^{\star}$ with $\tau_{app} = (A:\star)list(A) \rightarrow list(A) \rightarrow list(A)$ the concatenation function.

A usual definition for *app* is :

- $app(A, nil(A), \ell') \rightarrow \ell'$
- $app(A, cons(A, x, \ell), \ell') \rightarrow cons(A, x, app(A, \ell, \ell'))$

This definition satisfies the usual condition; it suffices to take $\Gamma = A\!:\!\star, x\!:\!A, \ell : list(A), \ell'\!:\!list(A)$ and $T = list(A)$. But one may wonder whether it is really necessary to do an equality test between the first argument of *app* and the first argument of *cons* when one wants to apply the second rule. Indeed, if $app(A, cons(A', x, \ell), \ell')$ is well typed then, by inversion, $cons(A', x, \ell)$ is of type $list(A)$ and, by inversion again, $list(A')$ is convertible to $list(A)$. Hence, to allow the reduction even though A' is different from A does not seem to be harmful since $list(A')$ is convertible to $list(A)$.

In fact, what is important is not that the left-hand side of a rule is typable, but that, if an instance of the left-hand side of a rule is typable, then the corresponding instance of the right-hand side has the same type. We express this by requiring that there exists an environment Γ in which the right-hand side is typable, and a substitution ρ which replaces the variables of the left-hand side not belonging to Γ by terms typable typable in Γ. Hence, one can consider the following rules :

- $app(A, nil(A'), \ell') \rightarrow \ell'$
- $app(A, cons(A', x, \ell), \ell') \rightarrow cons(A, x, app(A, \ell, \ell'))$

by taking $\Gamma = A : \star, x : A, \ell : list(A), \ell' : list(A)$ and $\rho = \{A' \mapsto A\}$. In [20], we give 5 conditions, (S1) to (S5), which must be satisfied by the rule $l \rightarrow r$, Γ and ρ. Assume that $l = f(\vec{l})$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$. Then, (S1) is $\mathrm{dom}(\rho) \subseteq \mathrm{FV}(l) \setminus \mathrm{dom}(\Gamma)$ and (S2) is $\Gamma \vdash l\rho : U\gamma\rho$. Although these two first conditions are often true, they are not necessary for proving the subject reduction property. This is why, in the following definition, they are not written. However, we will see that (S2) is necessary for proving the strong normalization property (see Definition 81).

Definition 58 (Well-typed rule) A rule $l \rightarrow r$ is *well-typed* if there exists an environment Γ and a substitution ρ such that, if $l = f(\vec{l})$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$ then :

(S3) $\Gamma \vdash r : U\gamma\rho$,
(S4) for all Δ, σ and T, if $\Delta \vdash l\sigma : T$ then $\sigma : \Gamma \rightarrow \Delta$,
(S5) for all Δ, σ and T, if $\Delta \vdash l\sigma : T$ then, for all x, $x\sigma \downarrow x\rho\sigma$.

In the following, we will write $(l \rightarrow r, \Gamma, \rho) \in \mathcal{R}$ when the previous conditions are satisfied.

An example using a dependent type is given by the concatenation of two lists of given length and the function *map* which, to a function f and a list $a_1 \dots a_n$, associates the list $f(a_1) \dots f(a_n)$:

- $T \in \mathcal{F}_0^\square$ with $\tau_T = \star$ a type constant,
- $nat \in \mathcal{F}_0^\square$ with $\tau_{nat} = \star$ the type of natural numbers,
- $0 \in \mathcal{F}_0^\star$ with $\tau_0 = nat$ zero,
- $s \in \mathcal{F}_1^\star$ with $\tau_s = nat \rightarrow nat$ the successor function,
- $+ \in \mathcal{F}_2^\star$ with $\tau_+ = nat \rightarrow nat \rightarrow nat$ the addition on nat,
- $listn \in \mathcal{F}_1^\square$ with $\tau_{listn} = nat \rightarrow \star$ the type of lists of given length,
- $niln \in \mathcal{F}_0^\star$ with $\tau_{niln} = listn(0)$ the empty list,
- $consn \in \mathcal{F}_3^\star$ with $\tau_{consn} = T \rightarrow (n : nat)listn(n) \rightarrow listn(s(n))$ the function adding an element at the head of a list,
- $appn \in \mathcal{F}_4^\star$ with $\tau_{appn} = (n : nat)listn(n) \rightarrow (n' : nat)listn(n') \rightarrow listn(n + n')$ the concatenation function,
- $mapn \in \mathcal{F}_3^\star$ with $\tau_{mapn} = (T \rightarrow T) \rightarrow (n : nat)listn(n) \rightarrow listn(n)$ the function which, to a function $f : T \rightarrow T$ and a list $a_1 \dots a_n$, associates the list $f(a_1) \dots f(a_n)$,

where $+$, *appn* and *mapn* are defined by :

- $+(0,n') \to n'$
- $+(s(n),n') \to s(n+n')$
- $appn(0,\ell,n',\ell') \to \ell'$
- $appn(p,consn(x,n,\ell),n',\ell') \to consn(x,n+n',appn(n,\ell,n',\ell'))$
- $mapn(f,0,\ell) \to \ell$
- $mapn(f,p,consn(x,n,\ell)) \to consn(fx,n,mapn(f,n,\ell))$
- $mapn(f,p,appn(n,\ell,n',\ell')) \to appn(n,mapn(f,n,\ell),n',mapn(f,n',\ell'))$

For the second rule of *appn*, we take $\Gamma = x:T, n:nat, \ell:listn(n), n':nat, \ell':listn(n')$ and $\rho = \{p \mapsto s(n)\}$. This avoids checking that p is convertible to $s(n)$.

For the third rule of *mapn*, we take $\Gamma = f:T \to T, n:nat, \ell:listn(n), n':nat, \ell':listn(n')$ and $\rho = \{p \mapsto n+n'\}$. This avoids checking that p is convertible to $n+n'$.

The reader will find other examples in Section 7.2.

Lemma 59 (Subject reduction for rewriting) If $\mathcal{R}$ is a set of well-typed rules then $\to_{\mathcal{R}}$ preserves typing.

Proof. We proceed as for the correctness of $\to_\beta$ and only consider case (symb) :

$$\textbf{(symb)}\ \frac{\vdash \tau_f : s \quad \Gamma \text{ valid} \quad \Gamma \vdash t_1 : T_1\gamma \ldots \Gamma \vdash t_n : T_n\gamma}{\Gamma \vdash f(\vec{t}) : U\gamma} \qquad \begin{array}{c}(f \in \mathcal{F}^s_n,\\ \tau_f = (\vec{x}:\vec{T})U,\\ \gamma = \{\vec{x} \mapsto \vec{t}\})\end{array}$$

Let $(l \to r, \Gamma_0, \rho) \in \mathcal{R}$ with $l = f(\vec{l})$, $\tau_f = (\vec{x}:\vec{T})U$ and $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$. Assume that $t = l\sigma$. We prove that $\Gamma \vdash r\sigma : U\gamma$. By **(S4)**, $\sigma : \Gamma_0 \to \Gamma$. By **(S3)**, $\Gamma_0 \vdash r : U\gamma_0\rho$. Therefore, by substitution, $\Gamma \vdash r\sigma : U\gamma_0\rho\sigma$. By **(S5)**, for all x, $x\rho\sigma$ and $x\sigma$ have a common reduct that we will call t_x. Therefore, by successively reducing in $U\gamma_0\rho\sigma$ each $x\rho\sigma$ to t_x, and in $U\gamma_0\sigma$ each $x\sigma$ to t_x, we obtain $U\gamma_0\rho\sigma \downarrow U\gamma_0\sigma$. But $U\gamma_0\sigma = U\gamma$ and, by inversion, there exists s' such that $\Gamma \vdash U\gamma : s'$. Therefore, by (conv), $\Gamma \vdash r\sigma : U\gamma_0\sigma$. ∎

Theorem 60 (Admissibility) A logical ATS whose rules are well-typed is an RTS, *i.e.* its reduction relation preserves typing.

Proof. It is true for $\to_\beta$ since we assume that the ATS is logical. For $\to_{\mathcal{R}}$, this comes from the correctness of rewriting. ∎

How to check the conditions (S3), (S4) and (S5) ? In all their generality, they are certainly undecidable. On the one hand, we do not know whether $\vdash$ and $\downarrow$ are decidable and, on the other hand, in (S4) and (S5), we arbitrarily quantify on Δ, σ and T. It is therefore necessary to make additional hypothesis. In the following, we successively consider the three conditions.

Let us look at (S3). In practice, the symbols and their defining rules are often added one after another (or by groups but the following argument can be generalized). Let $(\mathcal{F},\mathcal{R})$ be a system in which $\vdash$ is decidable (for example, a functional, confluent and strongly normalizing system), $f \notin \mathcal{F}$ and $\mathcal{R}_f$ a set of rules defining f

and whose symbols belong to $\mathcal{F}' = \mathcal{F} \cup \{f\}$. Then, in $(\mathcal{F}', \mathcal{R})$, $\vdash$ is still decidable. One can therefore try to check (S3) in this system. This does not seem an important restriction : it would be surprising if the typing of a rule required the use of the rule itself !

We now consider (S4).

Definition 61 (Canonical and derived types) Let t be a term of the form $l\sigma$ with $l = f(\vec{l})$ algebraic, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$. The term $U\gamma\sigma$ will be called the *canonical type* of t.

Let $p \in \mathrm{Pos}(l)$ with $p \neq \varepsilon$. We define the *type of* $t|_p$ *derived from* t, $\tau(t,p)$, as follows :

- if $p = i$ then $\tau(t,p) = T_i\gamma\sigma$,
- if $p = iq$ and $q \neq \varepsilon$ then $\tau(t,p) = \tau(t_i, q)$.

In fact, the type of $t|_p$ derived from t only depends on the term just above $t|_p$ in t.

The following lemma shows that the canonical type of t and the type of $t|_p$ derived from t are indeed types for t and $t|_p$ respectively.

Lemma 62 Let t be a term of the form $l\sigma$ with $l = f(\vec{l})$ algebraic and $\Gamma \vdash t : T$, V the canonical type of t and $p \in \mathrm{Pos}(l)$ with $p \neq \varepsilon$. In any TSM, $\Gamma \vdash t : V$ and $\Gamma \vdash t|_p : \tau(t,p)$.

Proof. From $\Gamma \vdash t : T$, by inversion, we immediately obtain $\Gamma \vdash t : V$. Let us consider $\Gamma \vdash t|_p : \tau(t,p)$ now. As $p \neq \varepsilon$, we have $p = qi$, $t|_q$ of the form $g(\vec{k}\sigma)$ with $g(\vec{k})$ algebraic and $t|_q$ typable in Γ. Assume that $\tau_g = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{k}\}$. Then, $\tau(t,p) = T_i\gamma\sigma$ and, by inversion, $\Gamma \vdash t|_p : T_i\gamma\sigma$. ■

Lemma 63 (S4) Let $l \to r$ be a rule and Γ be an environment. If, for all $x \in \mathrm{dom}(\Gamma)$, there exists $p_x \in \mathrm{Pos}(x,l)$ such that $\tau(l,p_x) = x\Gamma$, then (S4) is satisfied.

Proof. Assume that $\Delta \vdash l\sigma : T$. As l is algebraic, by inversion, $\Delta \vdash x\sigma : \tau(l\sigma, p_x) = \tau(l,p_x)\sigma = x\Gamma\sigma$. ■

On the other hand, for (S5), we have no general result. By inversion, (S5) can be seen as a problem of unification modulo $\downarrow^*$. The confluence of $\to$ (which implies $\downarrow^* = \downarrow$) can therefore be very useful. Unfortunately, there are very few general results on the confluence of $\to_\mathcal{R} \cup \to_\beta$ (see the discussion after Definition 91). On the other hand, one can easily prove that the local confluence is preserved.

Lemma 64 (Local confluence) If $\to_\mathcal{R}$ is locally confluent on $\mathcal{T}(\mathcal{F},\mathcal{X})$ then $\to = \to_\mathcal{R} \cup \to_\beta$ is locally confluent on $\mathcal{T}$.

Proof. We write $t \to^p t'$ when there exists u such that $t|_p \to u$ and $t' = t[u']_p$ (reduction at position p). Assume that $t \to^p t_1$ and $t \to^q t_2$. We prove by induction on t that there exists t' such that $t_1 \to^* t'$ and $t_2 \to^* t'$. There is three cases :

- p and q have no common prefix. The reductions at p and q can be done in parallel : $t_1 \to^q t_1'$, $t_2 \to^p t_2'$ and $t_1' = t_2'$.
- $p = ip'$ and $q = iq'$. We can conclude by induction hypothesis on $t|_i$.
- $p = \varepsilon$ or $q = \varepsilon$. By exchanging the roles of p and q, we can assume that $p = \varepsilon$. Then, there is two cases :
  - $t = [x\!:\!V]u\ v$ and $t_1 = u\{x \mapsto v\}$. One can distinguish three sub-cases :
    - $q = 11q'$ and $V \to^{q'} V'$. Then $t' = t_1$ works.
    - $q = 12q'$ and $u \to^{q'} u'$. Then $t' = u'\{x \mapsto v\}$ works.
    - $q = 2q'$ and $v \to^{q'} v'$. Then $t' = u\{x \mapsto v'\}$ works.
  - $t = l\sigma$, $l \to r \in \mathcal{R}$ and $t_1 = r\sigma$. There exists an algebraic term u of maximal size and a substitution θ such that $t = u\theta$ and $x\theta = y\theta$ implies $x = y$ (u and θ are unique up to the choice of variables and u has the same non-linearities than t). As the left-hand sides of rules are algebraic, $u = l\sigma'$ and $\sigma = \sigma'\theta$. Now, one can distinguish two sub-cases :
    - $q \in \mathrm{Pos}(u)$. As the left-hand sides of rules are algebraic, we have $u \to_{\mathcal{R}} r\sigma'$ and $u \to_{\mathcal{R}} v$. By local confluence of $\to_{\mathcal{R}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, there exists u' such that $r\sigma' \to^* u'$ and $v \to^* u'$. Hence, $t_1 = r\sigma'\theta \to^* u'\theta$ and $t_2 = v\theta \to^* u'\theta$.
    - $q = q_1q'$ and $u|_{q_1} = x$. Let $q_2, \ldots, q_n$ be the positions of the other occurrences of x in u. If one reduces t_2 at each position q_iq', one obtains a term of the form $l\sigma'\theta'$ where θ' is the substitution equal to θ but for x where it is equal to the reduct of $x\theta$. Then , it suffices to take $t' = r\sigma'\theta'$. ■

Chapter 6

Conditions of Strong Normalization

In this chapter, we are going to give strong normalization conditions for ATS's based on the Calculus of Constructions.

Definition 65 (CAC) A *Calculus of Algebraic Constructions* (CAC) is an ATS $(\mathcal{S}, \mathcal{F}, \mathcal{X}, \mathcal{A}, \mathcal{B}, \tau, \mathcal{R})$ such that $\mathcal{S} = \{\star, \Box\}$, $\mathcal{A} = \{(\star, \Box)\}$ and $\mathcal{B} = \{(s_1, s_2, s_3) \in \mathcal{S}^3 \mid s_2 = s_3\}$.

A CAC is injective and regular.

6.1 Term classes

After the Separation and Classification Lemmas, typable terms can be divided into three disjoint classes : $\mathbb{T}_0^\Box$, $\mathbb{T}_1^\Box$ and $\mathbb{T}_1^\star$. For denoting them, we introduce more explicit notations.

Definition 66 (Typing classes)
– Let $\mathbb{K} = \mathbb{T}_0^\Box$ be the class of *predicate types*.
– Let $\mathbb{P} = \mathbb{T}_1^\Box$ be the class of *predicates*.
– Let $\mathbb{O} = \mathbb{T}_1^\star$ be the class of *objects*.

That a well-typed term belongs to one of these classes can be easily decided by introducing the following syntactic classes :

Definition 67 (Syntactic classes)
- The syntactic class $\mathcal{K}$ of *predicate types* :
  – $\star \in \mathcal{K}$,
  – if $x \in \mathcal{X}$, $T \in \mathcal{T}$ and $K \in \mathcal{K}$ then $(x\!:\!T)K \in \mathcal{K}$.
- The syntactic class $\mathcal{P}$ of *predicates* :
  – $\mathcal{X}^\Box \subseteq \mathcal{P}$,
  – if $x \in \mathcal{X}$, $T \in \mathcal{T}$ and $P \in \mathcal{P}$ then $(x\!:\!T)P \in \mathcal{P}$ and $[x\!:\!T]P \in \mathcal{P}$,

  - if $P \in \mathcal{P}$ and $t \in \mathcal{T}$ then $Pt \in \mathcal{P}$,
  - if $F \in \mathcal{F}_n^\square$ and $t_1, \ldots, t_n \in \mathcal{T}$ then $F(\vec{t}) \in \mathcal{P}$.
- The syntactic class $\mathcal{O}$ of *objects* :
  - $\mathcal{X}^\star \subseteq \mathcal{O}$,
  - if $x \in \mathcal{X}$, $T \in \mathcal{T}$ and $u \in \mathcal{O}$ then $[x\!:\!T]u \in \mathcal{O}$,
  - if $u \in \mathcal{O}$ and $t \in \mathcal{T}$ then $ut \in \mathcal{O}$,
  - if $f \in \mathcal{F}_n^\star$ and $t_1, \ldots, t_n \in \mathcal{T}$ then $f(\vec{t}) \in \mathcal{O}$.

Lemma 68 Syntactic classes are disjoint from one another and each typing class is included in its corresponding syntactic class : $\mathbb{K} \subseteq \mathcal{K}$, $\mathbb{P} \subseteq \mathcal{P}$ and $\mathbb{O} \subseteq \mathcal{O}$.

Proof. That the syntactic classes are disjoint from one another comes from their definition. We prove that if $\Gamma \vdash t : T$ then t belongs to the syntactic class corresponding to its typing class by induction on $\Gamma \vdash t : T$. We follow the notations used in the typing rules.

(ax) As $\mathcal{A} = \{(\star, \square)\}$, we necessarily have $s_1 = \star$ and $s_2 = \square$. But, $\star \in \mathbb{K} \cap \mathcal{K}$.

(symb) By inversion and regularity, $\Gamma \vdash U\gamma : s$. Therefore, if $f \in \mathcal{F}^\star$ then $f(\vec{t}) \in \mathbb{O} \cap \mathcal{O}$, and if $f \in \mathcal{F}^\square$ then $f(\vec{t}) \in \mathbb{P} \cap \mathcal{P}$.

(var) If $x \in \mathcal{X}^\star$ then $x \in \mathbb{O} \cap \mathcal{O}$, and if $x \in \mathcal{X}^\square$ then $x \in \mathbb{P} \cap \mathcal{P}$.

(weak) By induction hypothesis.

(prod) By regularity, U and $(x\!:\!T)U$ have the same type. We can therefore conclude by induction hypothesis on U.

(abs) By inversion and regularity, $(x : T)U$ and U have the same type. We can therefore conclude by induction hypothesis on u.

(app) By inversion and regularity, $V\{x \mapsto u\}$ and $(x : U)V$ have the same type. We can therefore conclude by induction hypothesis on t.

(conv) By conversion correctness, T and T' have the same type. We can therefore conclude by induction hypothesis. ■

6.2 Inductive types and constructors

Until now we made few hypothesis on symbols and rewrite rules. However N. P. Mendler [90] showed that the extension of the simply-typed λ-calculus with recursion on inductive types is strongly normalizing if and only if the inductive types satisfy some positivity condition.

A base type T occurs positively in a type U if all the occurrences of T in U are on the left of a even number of $\rightarrow$. A type T is positive if T occurs positively in the type of the arguments of its constructors. Usual inductive types like natural numbers and lists of natural numbers are positive.

Now let us see an example of a non-positive type T. Let U be a base type. Assume that T has constructor c of type $(T \rightarrow U) \rightarrow T$. T is not positive because T occurs at a negative position in $T \rightarrow U$. Consider now the function p of type $T \rightarrow (T \rightarrow U)$ defined by the rule $p(c(x)) \rightarrow x$. Let $\omega = \lambda x.p(x)x$ of type $T \rightarrow U$. Then the term $\omega c(\omega)$ of type U is not normalizable :

$$\omega c(\omega) \rightarrow_\beta p(c(\omega))c(\omega) \rightarrow_\mathcal{R} \omega c(\omega) \rightarrow_\beta \dots$$

In the case where $U = \star$, we can interpret this as Cantor's Theorem : there is no surjection from a set T to the set of its subsets $T \rightarrow \star$. In this interpretation, p is the natural injection between T and $T \rightarrow \star$. Saying that p is surjective is equivalent to saying (with the Axiom of Choice) that there exists c such that $p \circ c$ is the identity, that is, such that $p(c(x)) \rightarrow x$. In [49], G. Dowek shows that such an hypothesis is incoherent. Here, we show that this is related to the non-normalization of non-positive inductive types.

N. P. Mendler also gives a condition, strong positivity, in the case of dependent and polymorphic types. A similar notion, but more restrictive, strict positivity, is used by T. Coquand and C. Paulin in the Calculus of Inductive Constructions [39].

Hereafter we introduce the more general notion of *structure inductive admissible*. In particular, we do not consider that a constructor must be constant : it will be possible to have rewrite rules on constructors. This will allow us to formalize quotient types as the type *int* of integers :

- $int \in \mathcal{F}_0^\Box$ with $\tau_{int} = \star$ the type of integers,
- $0 \in \mathcal{F}_0^\star$ with $\tau_0 = int$ the constant zero,
- $s \in \mathcal{F}_1^\star$ with $\tau_s = int \rightarrow int$ the successor function,
- $p \in \mathcal{F}_1^\star$ with $\tau_p = int \rightarrow int$ the predecessor function,

where s and p are defined by :

- $s(p(x)) \rightarrow x$
- $p(s(x)) \rightarrow x$

Definition 69 (Constructors) Let C be a constant predicate symbol. A symbol f is a *constructor* of C if τ_f is of the form $(\vec{y} : \vec{U})C(\vec{v})$ with $\alpha_f = |\vec{y}|$.

Our notion of constructor not only includes the usual (constant) constructors but also any symbol producing terms of type C. For example :

- $+ \in \mathcal{F}_2^\star$ with $\tau_+ = int \rightarrow int \rightarrow int$ the addition on integers,
- $\times \in \mathcal{F}_2^\star$ with $\tau_\times = int \rightarrow int \rightarrow int$ the multiplication on integers,

or with polymorphic lists :

- $app \in \mathcal{F}_3^\star$ with $\tau_{app} = (A : \star)list(A) \rightarrow list(A) \rightarrow list(A)$ the concatenation function.

A constant predicate symbol having some constructors cannot have any arity :

Definition 70 (Maximal arity) A predicate symbol F is of *maximal arity* if $\tau_F = (\vec{x} : \vec{T})\star$ and $\alpha_F = |\vec{x}|$.

Lemma 71 Let C be a constant predicate symbol and c a constructor of C. In a logical CAC, if $\vdash \tau_C : \Box$ and $\vdash \tau_c : s$ then $s = \star$ and C is of maximal arity.

Proof. Assume that $\tau_C = (\vec{x} : \vec{V})W$ and $\tau_c = (\vec{y} : \vec{U})C(\vec{v})$. Let $\gamma = \{\vec{x} \mapsto \vec{v}\}$. As $\Box$ is a maximal sort and $\vdash \tau_C : \Box$, after the lemma on maximal sorts, W is of the form $(\vec{x}' : \vec{V}')\star$. Now, from $\vdash \tau_c : s$, by inversion and regularity, on deduce that $\Gamma_c \vdash C(\vec{v}) : s$, $\Gamma_c \vdash C(\vec{v}) : W\gamma$ and $W\gamma \ \mathbb{C}^*_{\Gamma_c}\ s$. As $\Gamma_c \vdash W\gamma : \Box$, by conversion correctness, $\Gamma_c \vdash s : \Box$ and, by inversion, $s = \star$. Therefore, after the lemma on maximal sorts, $|\vec{x}'| = 0$ and $W = \star$. ■

Definition 72 (Inductive structure) An *inductive structure* is given by :

- a quasi-ordering $\geq_\mathcal{C}$ on $\mathcal{CF}^\Box$ whose strict part $>_\mathcal{C}$ is well-founded;
- for every constant predicate symbol C of type $(\vec{x} : \vec{T})\star$, a set $\mathrm{Ind}(C) \subseteq \{i \leq \alpha_C \mid x_i \in \mathcal{X}^\Box\}$ for the *inductive positions* of C;
- for every constructor c, a set $\mathrm{Acc}(c) \subseteq \{1, \ldots, \alpha_c\}$ for the *accessible positions* of c.

The accessible positions denote the arguments that one wants to use in the right hand-sides of rules. The inductive positions denote the parameters in which constructors must be monotone.

Definition 73 (Positive and negative positions) Let $T \in \mathcal{T} \setminus \mathcal{O}$. The set of *positive positions* in T, $\mathrm{Pos}^+(T)$, and the set of *negative positions* in T, $\mathrm{Pos}^-(T)$, are simultaneously defined by induction on the structure of T :

- $\mathrm{Pos}^+(s) = \mathrm{Pos}^+(F(\vec{t})) = \mathrm{Pos}^+(X) = \varepsilon$,
- $\mathrm{Pos}^-(s) = \mathrm{Pos}^-(F(\vec{t})) = \mathrm{Pos}^-(X) = \emptyset$,
- $\mathrm{Pos}^\delta((x{:}V)W) = 1.\mathrm{Pos}^{-\delta}(V) \cup 2.\mathrm{Pos}^\delta(W)$,
- $\mathrm{Pos}^\delta([x{:}V]W) = 1.\mathrm{Pos}(V) \cup 2.\mathrm{Pos}^\delta(W)$,
- $\mathrm{Pos}^\delta(Vu) = 1.\mathrm{Pos}^\delta(V) \cup 2.\mathrm{Pos}(u)$,
- $\mathrm{Pos}^\delta(VU) = 1.\mathrm{Pos}^\delta(V)$,
- $\mathrm{Pos}^+(C(\vec{t})) = \{\varepsilon\} \cup \bigcup \{i.\mathrm{Pos}^+(t_i) \mid i \in \mathrm{Ind}(C)\}$,
- $\mathrm{Pos}^-(C(\vec{t})) = \bigcup \{i.\mathrm{Pos}^-(t_i) \mid i \in \mathrm{Ind}(C)\}$,

where $\delta \in \{-,+\}$, $-+ = -$, $-- = +$ (usual rule of signs). The set of *neutral positions* in T is $\mathrm{Pos}^0(T) = \mathrm{Pos}^+(T) \cap \mathrm{Pos}^-(T)$. The set of *non-neutral positions* in T is $\mathrm{Pos}^{\neq 0}(T) = (\mathrm{Pos}^+(T) \cup \mathrm{Pos}^-(T)) \setminus \mathrm{Pos}^0(T)$.

The positive and negative positions do not form two disjoint sets. Their intersection forms the neutral positions. For example, all the positions of u in Vu or all the positions of V in $[x{:}V]W$ are neutral. We will see in Section 8.3 that these subterms are not taken into account into the interpretation of a type.

In [20], we give 6 conditions, (I1) to (I6), for defining what is an admissible inductive structure. But we found that (I1) can be eliminated if we modify (I2) a little bit. That is why, in the following definition, there is no (I1) and (I2) is placed after (I6).

Definition 74 (Admissible inductive structures) An inductive structure is *admissible* if for all constant predicate symbol C, for all constructor c of type $(\vec{y} : \vec{U})\ C(\vec{v})$ and for all $j \in \mathrm{Acc}(c)$:

(I3) $\forall D \in \mathcal{CF}^\Box, D =_\mathcal{C} C \Rightarrow \mathrm{Pos}(D, U_j) \subseteq \mathrm{Pos}^+(U_j)$ (symbols equivalent to C must be at positive positions),

(I4) $\forall D \in \mathcal{CF}^\Box, D >_\mathcal{C} C \Rightarrow \mathrm{Pos}(D, U_j) \subseteq \mathrm{Pos}^0(U_j)$ (symbols greater than C must be at neutral positions),

(I5) $\forall F \in \mathcal{DF}^\Box, \mathrm{Pos}(F, U_j) \subseteq \mathrm{Pos}^0(U_j)$ (defined symbols must be at neutral positions),

(I6) $\forall Y \in \mathrm{FV}^\Box(U_j), \exists\, \iota_Y \leq \alpha_C, v_{\iota_Y} = Y$ (every predicate variable in U_j must be a parameter of C),

(I2) $\forall Y \in \mathrm{FV}^\Box(U_j), \iota_Y \in \mathrm{Ind}(C) \Rightarrow \mathrm{Pos}(Y, U_j) \subseteq \mathrm{Pos}^+(U_j)$ (every predicate variable in U_j which is an inductive parameter of C must be at a positive position).

For example, $\mathrm{Ind}(\mathit{list}) = \{1\}$, $\mathrm{Acc}(\mathit{nil}) = \{1\}$ and $\mathrm{Acc}(\mathit{cons}) = \{1, 2, 3\}$ is an admissible inductive structure. Assume we add :

– $\mathit{tree} \in \mathcal{F}_0^\Box$ with $\tau_{\mathit{tree}} = \star$ the type of finite branching trees,
– $\mathit{node} \in \mathcal{F}_1^\star$ with $\tau_{\mathit{node}} = \mathit{list}(\mathit{tree}) \to \mathit{tree}$ its constructor.

Since $1 \in \mathrm{Ind}(\mathit{list})$, if $\mathrm{Ind}(\mathit{tree}) = \emptyset$ and $\mathrm{Acc}(\mathit{node}) = \{1\}$ then we still have an admissible structure.

To allow greater or defined symbols does not matter if these symbols are at neutral positions since neutral subterms are not taken into account into the interpretation of a type.

The condition (I6) means that the predicate arguments of a constructor must be parameters of their type. A similar condition appears in the works of M. Stefanova [108] ("safeness") and D. Walukiewicz [118] ("$\star$-dependency"). On the other hand, in the Calculus of Inductive Constructions (CIC) [99], there is no such restriction. However, because of the typing rules of the elimination scheme, no very interesting function seems to be definable on a type not satisfying this condition.

For example, let us take the type of heterogeneous non empty lists (in the CIC syntax) :

– $\mathit{listh} = \mathit{Ind}(X : \star)\{C_1 | C_2\}$ where $C_1 = (A\!:\!\star)(x\!:\!A)X$ and $C_2 = (A\!:\!\star)(x\!:\!A)X \to X$,
– $\mathit{endh} = \mathit{Constr}(1, \mathit{listh})$,
– $\mathit{consh} = \mathit{Constr}(2, \mathit{listh})$.

The typing rule of the non dependent elimination scheme ($\mathrm{Nodep}_{\star,\star}$) is :

$$\frac{\Gamma \vdash \ell : \mathit{listh} \quad \Gamma \vdash Q : \star \quad \Gamma \vdash f_1 : C_1\{\mathit{listh}, Q\} \quad \Gamma \vdash f_2 : C_2\{\mathit{listh}, Q\}}{\Gamma \vdash \mathit{Elim}(\ell, Q)\{f_1 | f_2\} : Q}$$

where $C_1\{\mathit{listh}, Q\} = (A\!:\!\star)(x\!:\!A)Q$ and $C_2\{\mathit{listh}, Q\} = (A\!:\!\star)(x\!:\!A)\mathit{listh} \to Q \to Q$. So, Q, f_1 and f_2 must be typable in Γ. The result of f_1 or f_2 cannot depend on A or x. This means that, for example, it is possible to compute the length of such a list but one cannot extract an element of such a list. We think that the length of such a list is an information that can surely be obtained without using such a data type.

We can distinguish several kinds of inductive types.

Definition 75 (Primitive, basic and strictly positive predicates)
A constant predicate symbol C is :

- *primitive* if for all $D =_{\mathcal{C}} C$, for all constructor d of type $(\vec{y} : \vec{U})D(\vec{w})$ and for all $j \in \mathrm{Acc}(d)$, U_j is either of the form $E(\vec{t})$ with $E <_{\mathcal{C}} D$ and E primitive, or of the form $E(\vec{t})$ with $E =_{\mathcal{C}} D$.
- *basic* if for all $D =_{\mathcal{C}} C$, for all constructor d of type $(\vec{y} : \vec{U})D(\vec{w})$ and for all $j \in \mathrm{Acc}(d)$, if $E =_{\mathcal{C}} D$ occurs in U_j then U_j is of the form $E(\vec{t})$.
- *strictly positive* if for all $D =_{\mathcal{C}} C$, for all constructor d of type $(\vec{y} : \vec{U})D(\vec{w})$ and for all $j \in \mathrm{Acc}(d)$, if $E =_{\mathcal{C}} D$ occurs in U_j then U_j is of the form $(\vec{z} : \vec{V})E(\vec{t})$ and no $D' =_{\mathcal{C}} D$ occurs in the V_i's.

It is easy to see that a primitive predicate if basic and that a basic predicate is strictly positive. The type *listn* of lists of length n is primitive. The type *list* of polymorphic lists is basic and not primitive.

The strictly positive predicates are the predicates allowed in the Calculus of Inductive Constructions (CIC). For example, the type of well-founded trees or Brouwer's ordinals :

- $ord \in \mathcal{F}_0^\square$ with $\tau_{ord} = \star$ the type of Brouwer's ordinals,
- $0 \in \mathcal{F}_0^\star$ with $\tau_0 = ord$ the ordinal zero,
- $s \in \mathcal{F}_1^\star$ with $\tau_s = ord \rightarrow ord$ the successor ordinal,
- $lim \in \mathcal{F}_1^\star$ with $\tau_{lim} = (nat \rightarrow ord) \rightarrow ord$ the limit ordinal.[1]

Another example is given by the following process algebra which uses a choice operator Σ other some data type *data* [107] :

- $data \in \mathcal{F}_0^\square$ with $\tau_{data} = \star$ a data type,
- $proc \in \mathcal{F}_0^\square$ with $\tau_{proc} = \star$ the type of processes,
- $\circ \in \mathcal{F}_2^\star$ with $\tau_\circ = proc \rightarrow proc \rightarrow proc$ the sequence,
- $+ \in \mathcal{F}_2^\star$ with $\tau_+ = proc \rightarrow proc \rightarrow proc$ the parallelization,
- $\delta \in \mathcal{F}_0^\star$ with $\tau_\delta = proc$ the deadlock,
- $\Sigma \in \mathcal{F}_1^\star$ with $\tau_\Sigma = (data \rightarrow proc) \rightarrow proc$ the choice operator.

A last example is given by the first-order predicate calculus :

- $term \in \mathcal{F}_0^\square$ with $\tau_{term} = \star$ the type of terms,
- $form \in \mathcal{F}_0^\square$ with $\tau_{form} = \star$ the type of formulas,
- $\vee \in \mathcal{F}_2^\star$ with $\tau_\vee = form \rightarrow form \rightarrow form$ the "or",
- $\neg \in \mathcal{F}_1^\star$ with $\tau_\neg = form \rightarrow form$ the "not",

[1] A term of type *ord* does not necessary correspond to a true ordinal. However, if one carefully chooses the functions f for the limit ordinals then one can represent an initial enumerable segment of the true ordinals.

- $\forall \in \mathcal{F}_1^\star$ with $\tau_\forall = (term \to form) \to form$ the universal quantification.

For the moment, we do not forbid non-strictly positive predicates but the conditions we will describe in the next section do not allow one to define functions by recursion on such predicates.

Yet these predicates can be useful as shown by the following breadth-first label listing function of binary trees defined with the use of continuations [88] :

- $tree \in \mathcal{F}_0^\square$ with $\tau_{tree} = \star$ the type of binary labeled trees,
- $L \in \mathcal{F}_1^\star$ with $\tau_L = nat \to tree$ the leaf constructor,
- $N \in \mathcal{F}_3^\star$ with $\tau_N = nat \to tree \to tree \to tree$ the node constructor,

- $cont \in \mathcal{F}_0^\square$ with $\tau_{cont} = \star$ the type of continuations,
- $D \in \mathcal{F}_0^\star$ with $\tau_D = cont$,
- $C \in \mathcal{F}_1^\star$ with $\tau_C = ((cont \to list) \to list) \to cont$ its constructors,

- $@ \in \mathcal{F}_2^\star$ with $\tau_@ = cont \to (cont \to list) \to list$ the application on continuations defined by :
- $@(D, g) \to g\ D$
- $@(C(f), g) \to f\ g$

- $ex \in \mathcal{F}_1^\star$ with $\tau_{ex} = cont \to list$ the iterator on continuations defined by :
- $ex(D) \to nil$
- $ex(C(f)) \to f\ [k\!:\!cont]ex(k)$

- $br \in \mathcal{F}_2^\star$ with $\tau_{br} = tree \to cont \to cont$,
- $br\_fst \in \mathcal{F}_1^\star$ with $\tau_{br\_fst} = tree \to list$ the breadth-first label listing function defined by :
- $br\_fst(t) \to ex(br(t, D))$
- $br(L(x), k) \to C([g\!:\!cont \to list]cons(x, @(k, g)))$
- $br(N(x, s, t), k) \to C([g\!:\!cont \to list]cons(x, @(k, [m\!:\!cont]g\ br(s, br(t, m)))))$

This function is strongly normalizable since it can be encoded into the polymorphic λ-calculus [86, 87]. However, it is not clear how to define a syntactic condition, a schema, ensuring the strong normalization of such definitions. Indeed, in the right hand-side of the second rule defining ex, ex is explicitly applied to no argument smaller than f. However ex can only be applied to subterms of reducts of f. But not all the subterms of a computable term are *a priori* computable (see Subsection 6.3.2).

6.3 General Schema

6.3.1 Higher-order rewriting

Which conditions on rewrite rules would ensure the strong normalization of $\to = \to_\mathcal{R} \cup \to_\beta$? Since the works of V. Breazu-Tannen and J. Gallier [26] and M. Okada [97] on the simply-typed λ-calculus or the polymorphic λ-calculus, and later the works of F. Barbanera [4] on the Calculus of Constructions and of D. Dougherty [47] on the untyped λ-calculus, it is well known that adding first-order rewriting to typed λ-calculi preserves strong normalization. This comes from the fact that first-order rewriting cannot create new β-redexes. We will prove that this result can be extended to predicate-level rewriting if some conditions are fulfilled.

However, there are also many useful functions whose definition do not enter the first-order framework, either because some arguments are not primitive (the concatenation function *app* on polymorphic lists), or because their definition uses higher-order features like the function *map* which, to a function f and a list $a_1 \dots a_n$ of elements, associates the list $f(a_1) \dots f(a_n)$:

- $map \in \mathcal{F}_4^\star$ with $\tau_{map} = (A{:}\star)(B{:}\star)(A \to B) \to list(A) \to list(B)$
- $map(A, B, f, nil(A')) \to nil(B)$
- $map(A, B, f, cons(A', x, \ell)) \to cons(B, fx, map(A, B, f, \ell))$
- $map(A, B, f, app(A', \ell, \ell')) \to app(B, map(A, f, \ell), map(A, f, \ell'))$

This is also the case of recursors :

- $natrec \in \mathcal{F}_4^\star$ with $\tau_{natrec} = (A{:}\star)A \to (nat \to A \to A) \to nat \to A$ the recursor on natural numbers
- $natrec(A, x, f, 0) \to x$
- $natrec(A, x, f, s(n)) \to f\ n\ natrec(A, x, f, n)$
- $plus \in \mathcal{F}_0^\star$ with $\tau_{plus} = nat \to nat \to nat$ the addition on natural numbers
- $plus \to [p{:}nat][q{:}nat]natrec(nat, p, [q'{:}nat][r{:}nat]s(r), q)$

and of induction principles (recursors are just non-dependent versions of the corresponding induction principles) :

- $natind \in \mathcal{F}_4^\star$ with $\tau_{natind} = (P : nat \to \star)P0 \to ((n : nat)Pn \to Ps(n)) \to (n : nat)Pn$
- $natind(P, h_0, h_s, 0) \to h_0$
- $natrec(P, h_0, h_s, s(n)) \to h_s\, n\ natind(P, h_0, h_s, n)$

The methods used by V. Breazu-Tannen and J. Gallier [26] or D. Dougherty [47] cannot be applied to our calculus since, on the one hand, in contrast with first-order rewriting, higher-order rewriting can create β-redexes and, on the other hand, rewriting is included in the type conversion rule (conv), hence more terms are typable. But there exists other methods, available in the simply-typed λ-calculus only or in richer type systems, for proving the termination of this kind of definitions :

- The *General Schema*, initially introduced by J.-P. Jouannaud and M. Okada [74] for the polymorphic λ-calculus and extended to the Calculus of Constructions by F. Barbanera, M. Fernández and H. Geuvers [7], is basically an extension of the primitive recursion schema : in the right hand-side of a rule $f(\vec{l}) \to r$, the recursive calls to f must be done on strict subterms of $\vec{l}$. It can treat object-level and simply-typed symbols defined on primitive types. It has been reformulated and extended to strictly positive simple types by J.-P. Jouannaud, M. Okada and myself for the simply-typed λ-calculus [23] and the Calculus of Constructions [22].

- The *Higher-Order Recursive Path Ordering* (HORPO) of J.-P. Jouannaud and A. Rubio [76] [2] is an extension of the RPO [100, 45] for first-order terms to the terms of the simply-typed λ-calculus. It has been recently extended by D. Walukiewicz [118] to the Calculus of Constructions with polymorphic and dependent symbols at the object-level and with basic types. The General Schema can be seen as a non-recursive version of HORPO.
- It is also possible to look for an interpretation of the symbols such that the interpretation of a term strictly decreases when a rule is applied. This method, introduced by J. van de Pol for the simply-typed λ-calculus [116] extends to the higher-order the method of the interpretations known for the first-order framework. This is a very powerful method but difficult to use in practice because the interpretations are themselves of higher-order and also because it is not modular : adding new rules or new symbols may require finding new interpretations.

For dealing with higher-order rewriting at the predicate-level together with polymorphic and dependent symbols and strictly positive predicates, we have chosen to extend the method of the General Schema. For first-order symbols, we will use other conditions like in [74, 7].

6.3.2 Definition of the schema

This method is based on Tait and Girard's method of reductibility candidates [111, 64] for proving the strong normalization of the simply-typed and polymorphic λ-calculi. This method consists of defining a subset of the strongly normalizable terms, the *computable* terms, and in proving that each well-typed term is computable. Indeed, a direct proof of strong normalization by induction on the structure of terms does not go through because of the application case : if u and v are strongly normalizable then it is not clear how to prove that uv is also strongly normalizable.

The idea of the General Schema is then, from a left hand-side $f(\vec{l})$ of a rule, to define a set of terms, the *computable closure*, whose elements are computable whenever the l_i's are computable. Hence, to prove the termination of a definition, it suffices to check that, for each rule, the right hand-side belongs to the computable closure of the left hand-side.

To build the computable closure, we first define a subset of the subterms of the l_i's that are computable whenever the l_i's are computable : the *accessible* subterms of the l_i's (*a priori* not all the subterms of a computable term are computable). Then we build the computable closure by closing the set of the accessible variables of the left hand-side with computability-preserving operations.

For most interesting functions, we must be able to accept recursive calls. And to preserve strong normalization, recursive calls must decrease in a well-founded

[2] This generalizes the previous works of C. Loria-Saenz and J. Steinbach [80], O. Lysne and J. Piris [83] and J.-P. Jouannaud and A. Rubio [77].

ordering. The strict subterm relation $\rhd$ (in fact, restricted to accessible subterms for preserving computability) is sufficient for dealing with definition on basic predicates. In the example of *map*, ℓ and ℓ' are strict accessible subterms of $app(A', \ell, \ell')$. But, for non-basic predicates, it is not sufficient as examplified by the following “addition” on Brouwer's ordinals :

- $+ \in \mathcal{F}_2^\star$ with $\tau_+ = ord \to ord \to ord$,
- $+(x, 0) \to x$
- $+(x, s(y)) \to s(+(x, y))$
- $+(x, lim(f)) \to lim([n\!:\!nat] + (x, fn))$

Another example is given the following simplification rules on the process algebra *proc* [107] :

- $+(p, p) \to p$
- $+(p, \delta) \to p$
- $\ldots$
- $\circ(\Sigma(f), p) \to \Sigma([d\!:\!data] \circ (fd, p))$

This is why, in our conditions, we will use two distinct orderings. The first one, $>_1$, will be used for the arguments of basic type and the second one, $>_2$, will be used for the arguments of strictly-positive type.

Finally, for finer control over comparing the arguments, to each symbol we will associate a *status* describing how to compare two sequences of arguments by a simple combination of lexicographic and multiset comparisons [75].

Definition 76 (Accessibility relations) Let c be a constructor of type $(\vec{y} : \vec{U})C(\vec{v})$, $\vec{u}$ be arguments of c, $\gamma = \{\vec{y} \mapsto \vec{u}\}$ and $j \in \mathrm{Acc}(c)$ be an accessible position of c. Then :

- $u_j\!:\!U$ is *weakly accessible modulo* ρ in $c(\vec{u})\!:\!T$, $c(\vec{u})\!:\!T \ \rhd_1^\rho \ u_j\!:\!U$, if $T\rho = C(\vec{v})\gamma\rho$ and $U\rho = U_j\gamma\rho$.
- $u_j\!:\!U$ is *strongly accessible modulo* ρ in $c(\vec{u})\!:\!T$, $c(\vec{u})\!:\!T \rhd_2^\rho u_j\!:\!U$, if $T\rho = C(\vec{v})\gamma\rho$, $U\rho = U_j\gamma\rho$ and U_j is of the form $(\vec{x} : \vec{T})D(\vec{w})$.

We can use these relations to define the orderings $>_1$ and $>_2$. We have $\rhd_2^\rho \subseteq \rhd_1^\rho$. For technical reasons, we take into account not only the terms themselves but also their types. This comes from the fact that we are able to prove that two convertible types have the same interpretation only if these two types are computable. This may imply some restrictions on the types of the symbols.

Indeed, accessibility requires the equality (modulo the application of ρ) between canonical types and derived types (see Definition 61). More precisely, for having $t : T \rhd_1 u : U$, T must be equal (modulo ρ) to the canonical type of t and U must be equal (modulo ρ) to the type of u derived from t. In addition, if $u : U \rhd_1 v : V$ then U must also be equal (modulo ρ) to the canonical type of u.

Definition 77 (Precedence) A *precedence* is a quasi-ordering $\geq_{\mathcal{F}}$ on $\mathcal{F}$ whose strict part $>_{\mathcal{F}}$ is well-founded. We will denote by $=_{\mathcal{F}}$ its associated equivalence relation.

Definition 78 (Status) Let $(x_i)_{i\geq 1}$ be an indexed family of variables.

Status. A *status* is a linear term of the form $lex(m_1, \ldots, m_k)$ with $k \geq 1$ and each m_i of the form $mul(x_{k_1}, \ldots, x_{k_p})$ with $p \geq 1$. The *arity* of a status *stat* is the greatest indice i such that x_i occurs in *stat*.

Status assignment. A *status assignment* is an application *stat* which, to each symbol f of arity $n > 0$ and type $(\vec{x} : \vec{T})U$, associates a status $stat_f = lex(\vec{m})$ of arity smaller than or equal to n such that :

- if $x_i \in \mathrm{FV}(stat_f)$ then T_i is of the form $C^i_f(\vec{u})$ with C^i_f a constant predicate symbol,
- if $m_i = mul(x_{k_1}, \ldots, x_{k_p})$ then $C^{k_1}_f =_{\mathcal{C}} \ldots =_{\mathcal{C}} C^{k_p}_f$.

Strictly positive positions. Let f be a symbol of status $lex(\vec{m})$. We will denote by $SP(f)$ the set of indices i such that if $m_i = mul(x_{k_1}, \ldots, x_{k_p})$ then $C^{k_1}_f$ (therefore also $C^{k_2}_f, \ldots, C^{k_p}_f$) is strictly positive.

Assignment compatibility. A status assignment *stat* is *compatible* with a precedence $\geq_{\mathcal{F}}$ if :

- $f =_{\mathcal{F}} g$ implies $stat_f = stat_g$ and, for all i, $C^i_f =_{\mathcal{F}} C^i_g$.

Status ordering. Let $>$ be an ordering on terms and $stat = lex(\vec{m})$ be a status of arity n. The *extension* by *stat* of $>$ to the sequences of terms of length at least n is the ordering $>_{stat}$ defined as follows :

- $\vec{u} >_{stat} \vec{v}$ if $\vec{m}\{\vec{x} \mapsto \vec{u}\}\ (>^m)_{\mathrm{lex}}\ \vec{m}\{\vec{x} \mapsto \vec{v}\}$,
- $mul(\vec{u}) >^m mul(\vec{v})$ if $\{\vec{u}\} >_{\mathrm{mul}} \{\vec{v}\}$.

For example, if $stat = lex(mul(x_2), mul(x_1, x_3))$ then $\vec{u} >_{stat} \vec{v}$ if $(\{u_2\}, \{u_1, u_3\})\ (>_{\mathrm{mul}})_{\mathrm{lex}}\ (\{v_2\}, \{v_1, v_3\})$. An important property of $>_{stat}$ is that it is a well-founded ordering whenever $>$ so is.

We now define the orderings $>_1$ and $>_2$.

Definition 79 (Ordering on the arguments of a symbol) Let $R = (l \rightarrow r, \Gamma_0, \rho)$ be a rule with $l = f(\vec{l})$, $\tau_f = (\vec{x} : \vec{T})U$, $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$ and $stat_f = lex(\vec{m})$.

- $t : T >_1 u : U$ if $t : T\ (\rhd^\rho_1)^+\ u : U$.
- $t : T >_2 u : U$ if :
  - t is of the form $c(\vec{t})$ with c a constructor of type $(\vec{x} : \vec{T})C(\vec{v})$,
  - u is of the form $x\vec{u}$ with $x \in \mathrm{dom}(\Gamma_0)$, $x\Gamma_0$ of the form $(\vec{y} : \vec{U})D(\vec{w})$ and $D =_{\mathcal{C}} C$,
  - $t : T\ (\rhd^\rho_2)^+\ x : V$ with $V\rho = x\Gamma_0$.

Now, we define the ordering $>$ on the arguments of f (in fact the pairs argument-type $u : U$). This is an adaptation of $>_{stat_f}$ where the ordering $>$ is $>_1$ or $>_2$ depending on the type (basic or strictly positive) of the argument. Assume that $stat_f = lex(m_1, \ldots, m_k)$. Then :

- $\vec{u} > \vec{v}$ if $\vec{m}\{\vec{x} \mapsto \vec{u}\}\ (>^1, \ldots, >^k)_{\mathrm{lex}}\ \vec{m}\{\vec{x} \mapsto \vec{v}\}$,

- $mul(\vec{u}) >^i mul(\vec{v})$ if $\{\vec{u}\}\ (>_{\phi(i)})_{\mathrm{mul}}\ \{\vec{v}\}$ with $>_{\phi(i)} = >_2$ if $i \in SP(f)$ and $>_{\phi(i)} = >_1$ otherwise.

One can easily check that, in the third rule of the addition on ordinals, $lim(f) : ord >_2 fn : ord$. Indeed, for this rule, one can take $\Gamma_0 = x : ord, f : nat \to ord$ and the identity for ρ. Then, $f \in \mathrm{dom}(\Gamma_0)$, $\tau(lim(f), 1)\rho = f\Gamma_0 = nat \to ord$ and $lim(f) : ord \rhd_2^\rho f : nat \to ord$.

Figure 6.1: computable closure of $(f(\vec{l}) \to r, \Gamma_0, \rho)$

(ax) $$\frac{}{\Gamma_0 \vdash_{\mathrm{c}} \star : \Box}$$

(symb$^<$) $$\frac{\Gamma_0 \vdash_{\mathrm{c}} \tau_g : s \quad \Gamma \text{ valid pour } \vdash_{\mathrm{c}} \qquad \Gamma \vdash_{\mathrm{c}} u_1 : U_1\gamma \quad \ldots \quad \Gamma \vdash_{\mathrm{c}} u_n : U_n\gamma}{\Gamma \vdash_{\mathrm{c}} g(\vec{u}) : V\gamma} \quad \left(\begin{array}{c} g \in \mathcal{F}_n^s,\ g <_{\mathcal{F}} f, \\ \tau_g = (\vec{y} : \vec{U})V, \\ \gamma = \{\vec{y} \mapsto \vec{u}\}, \vdash \tau_g : s \end{array}\right)$$

(symb$^=$) $$\frac{\Gamma_0 \vdash_{\mathrm{c}} \tau_g : s \qquad \Gamma \vdash_{\mathrm{c}} u_1 : U_1\gamma \quad \ldots \quad \Gamma \vdash_{\mathrm{c}} u_n : U_n\gamma}{\Gamma \vdash_{\mathrm{c}} g(\vec{u}) : V\gamma} \quad \left(\begin{array}{c} g \in \mathcal{F}_n^s,\ g =_{\mathcal{F}} f, \\ \tau_g = (\vec{y} : \vec{U})V, \\ \gamma = \{\vec{y} \mapsto \vec{u}\}, \vdash \tau_g : s, \\ \vec{l} : \vec{T}\gamma_0 > \vec{u} : \vec{U}\gamma \end{array}\right)$$

(acc) $$\frac{\Gamma_0 \vdash_{\mathrm{c}} x\Gamma_0 : s}{\Gamma_0 \vdash_{\mathrm{c}} x : x\Gamma_0} \quad (x \in \mathrm{dom}^s(\Gamma_0))$$

(var) $$\frac{\Gamma \vdash_{\mathrm{c}} T : s}{\Gamma, x{:}T \vdash_{\mathrm{c}} x : T} \quad \left(\begin{array}{c} x \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma) \\ \cup \mathrm{FV}(l) \end{array}\right)$$

(weak) $$\frac{\Gamma \vdash_{\mathrm{c}} t : T \quad \Gamma \vdash_{\mathrm{c}} U : s}{\Gamma, x{:}U \vdash_{\mathrm{c}} t : T} \quad \left(\begin{array}{c} x \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma) \\ \cup \mathrm{FV}(l) \end{array}\right)$$

(prod) $$\frac{\Gamma, x{:}T \vdash_{\mathrm{c}} U : s}{\Gamma \vdash_{\mathrm{c}} (x{:}T)U : s}$$

(abs) $$\frac{\Gamma, x{:}T \vdash_{\mathrm{c}} u : U \quad \Gamma \vdash_{\mathrm{c}} (x{:}T)U : s}{\Gamma \vdash_{\mathrm{c}} [x{:}T]u : (x{:}T)U}$$

(app) $$\frac{\Gamma \vdash_{\mathrm{c}} t : (x{:}U)V \quad \Gamma \vdash_{\mathrm{c}} u : U}{\Gamma \vdash_{\mathrm{c}} tu : V\{x \mapsto u\}}$$

(conv) $$\frac{\Gamma \vdash_{\mathrm{c}} t : T \quad \Gamma \vdash_{\mathrm{c}} T : s \quad \Gamma \vdash_{\mathrm{c}} T' : s}{\Gamma \vdash_{\mathrm{c}} t : T'} \quad (T \downarrow T')$$

Definition 80 (Computable closure) Let $R = (l \to r, \Gamma_0, \rho)$ be a rule with $l = f(\vec{l})$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$. The *computable closure* of R w.r.t. a precedence $\geq_{\mathcal{F}}$ and a status assignment *stat* compatible with $\geq_{\mathcal{F}}$ is the smallest relation $\vdash_{\mathrm{c}} \subseteq \mathcal{E} \times \mathcal{T} \times \mathcal{T}$ defined by the inference rules of Figure 6.1. We will denote by $\vdash_{\mathrm{c}}^{<}$ the restriction of $\vdash_{\mathrm{c}}$ to the rules distinct from (symb$^=$)

One can easily check that if $\Gamma \vdash_{\mathrm{c}} t : T$ then $\Gamma = \Gamma_0, \Gamma'$. And also that $\vdash_{\mathrm{c}} \subseteq \vdash_s$ and $\vdash_{\mathrm{c}}^{<} \subseteq \vdash_f$.

It is important to note that the computable closure can easily be extended by adding new inference rules. For preserving the strong normalization, it suffices to complete the proof of Theorem 146 where we prove that the rules of the computable closure indeed preserve the computability.

Definition 81 (Well-formed rule) Let $R = (l \to r, \Gamma_0, \rho)$ be a rule with $l = f(\vec{l})$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$. The rule R is *well-formed* if :
– $\Gamma_0 \vdash l\rho : U\gamma_0\rho$,
– for all $x \in \mathrm{dom}(\Gamma_0)$, there exists i such that $l_i : T_i\gamma_0 \ (\rhd_1^\rho)^* \ x : x\Gamma_0$,
– $\mathrm{dom}(\rho) \cap \mathrm{dom}(\Gamma_0) = \emptyset$.

For example, consider the rule :

$$appn(p, consn(x, n, \ell), n', \ell') \to consn(x, n + n', appn(n, \ell, n', \ell'))$$

with $\Gamma_0 = x{:}T, n{:}nat, \ell{:}listn(n), n'{:}nat, \ell'{:}listn(n')$ and $\rho = \{p \mapsto s(n)\}$. We have $\Gamma_0 \vdash l\rho : listn(p + n')\rho$. For x, we have $consn(x, n, \ell) : listn(p) \rhd_1^\rho x : T$. One can easily check that the conditions are also satisfied for the other variables.

Definition 82 (Recursive system) Let $R = (l \to r, \Gamma_0, \rho)$ be a rule with $l = f(\vec{l})$, $\tau_f = (\vec{x} : \vec{T})U$ and $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$. The rule R satisfies the *General Schema* w.r.t. a precedence $\geq_{\mathcal{F}}$ and a status assignment *stat* compatible with $\geq_{\mathcal{F}}$ if R is well-formed and $\Gamma_0 \vdash_{\mathrm{c}} r : U\gamma_0\rho$.

A set of rules $\mathcal{R}$ is *recursive* if there exists a precedence $\geq_{\mathcal{F}}$ and a status assignment *stat* compatible with $\geq_{\mathcal{F}}$ for which every rule of $\mathcal{R}$ satisfies the General Schema w.r.t. $\geq_{\mathcal{F}}$ and *stat*.

Remark 83 (Decidability) *A priori*, because of the rule (conv) and of the condition $\vdash \tau_g : s$ for the rules (symb$^<$) and (symb$^=$), the relation $\vdash_{\mathrm{c}}$ may be undecidable. On the other hand, if we assume $\vdash \tau_g : s$ and restrict the rule (conv) to a confluent and strongly normalizing fragment of $\to$ then $\vdash_{\mathrm{c}}$ becomes decidable (with an algorithm similar to the one for $\vdash$). In practice, the symbols and the rules are often added one after the other (or by groups, but the argument can be generalized).

Let $(\mathcal{F}, \mathcal{R})$ be a confluent and strongly normalizing system, $f \notin \mathcal{F}$ and $\mathcal{R}_f$ a set of rules defining f and whose symbols belong to $\mathcal{F}' = \mathcal{F} \cup \{f\}$. Then $(\mathcal{F}', \mathcal{R})$ is also confluent and strongly normalizing. Thus we can check that the rules of $\mathcal{R}_f$ satisfy the General Schema with the rule (conv) restricted to the case where $T \downarrow_{\beta R} T'$. This does not seem a big restriction : it would be surprising that the typing of a rule requires the use of the rule itself!

Before to give a detailed example, we are going to show several properties of $\vdash_c$. Indeed, to show $\Gamma_0 \vdash_c r : U\gamma_0\rho$, knowing that by (S3) we have $\Gamma_0 \vdash r : U\gamma_0\rho$, one can wonder whether it is possible to transform a derivation of $\Gamma_0 \vdash r : U\gamma_0\rho$ into a derivation of $\Gamma_0 \vdash_c r : U\gamma_0\rho$. The best thing would be that it is sufficient that the symbols in r are smaller than f and that the recursive calls are made on smaller arguments. We prove hereafter that it is sufficient when there is no recursive call.

Definition 84 A rule $l \rightarrow r$ is *compatible* with $\geq_{\mathcal{F}}$ if all the symbols in r are smaller than or equivalent to l. A set of rule $\mathcal{R}$ is *compatible* with $\geq_{\mathcal{F}}$ if every rule of $\mathcal{R}$ is compatible with $\geq_{\mathcal{F}}$.

Lemma 85 Assume that $\mathcal{R}$ and τ are compatible with $\geq_{\mathcal{F}}$. Let Γ be an environment with variables distinct from those of Γ_0. If $\Gamma \vdash_f t : T$ then $\Gamma_0, \Gamma \vdash_c^< t : T$.

Proof. By induction on $\Gamma \vdash_f t : T$. ■

Lemma 86 (Substitution for $\vdash_c$) If $\Gamma_0, \Gamma \vdash_c^< t : T$ and $\theta : \Gamma_0, \Gamma \rightarrow \Gamma_0, \Gamma'$ in $\vdash_c$ (resp. in $\vdash_c^<$) then $\Gamma_0, \Gamma' \vdash_c t\theta : T\theta$ (resp. $\Gamma_0, \Gamma' \vdash_c^< t\theta : T\theta$).

Proof. By induction on $\Gamma_0, \Gamma \vdash_c^< t : T$. ■

Lemma 87 If $\mathcal{R}$ and τ are compatible with $\geq_{\mathcal{F}}$, $\rightarrow$ is confluent and the symbols of Γ_0 are strictly smaller than f then, for all $x \in \mathrm{dom}^s(\Gamma_0)$, $\Gamma_0 \vdash_c^< x\Gamma_0 : s$.

Proof. Assume that $\Gamma_0 = \vec{y} : \vec{U}$ and that y_i is of sort s_i. We show by induction on i that, for all $j \leq i$, $\Gamma_0 \vdash_c^< U_j : s_j$. If $i = 0$, this is immediate. So, assume that $i > 0$. By induction hypothesis, for all $j < i$, $\Gamma_0 \vdash_c^< U_j : s_j$. So, we are left to show that $\Gamma_0 \vdash_c^< U_i : s_i$.

Let $\Gamma = y_1 : U_1, \ldots, y_{i-1} : U_{i-1}$, $\vec{z}$ be $i-1$ fresh variables, $\theta = \{\vec{y} \mapsto \vec{z}\}$, $\theta' = \{\vec{z} \mapsto \vec{y}\}$ and $\Gamma' = \vec{z} : \vec{U}\theta$. By **(S3)**, Γ_0 is valid. So, by the Environment Lemma, $\Gamma \vdash U_i : s_i$. Since $\rightarrow$ is confluent and the symbols in Γ are strictly smaller than f, by Lemma 54, $\Gamma \vdash_f U_i : s_i$. By Replacement, $\Gamma' \vdash_f U_i\theta : s_i$. Therefore, by Lemma 85, $\Gamma_0, \Gamma' \vdash_c^< U_i\theta : s_i$.

Now we show that $\theta' : \Gamma_0, \Gamma' \rightarrow \Gamma_0$ in $\vdash_c^<$. For this, it is sufficient to show that, for all $j < i$, $\Gamma_0 \vdash_c^< z_j\theta' : U_j\theta\theta'$, that is, $\Gamma_0 \vdash_c^< y_j : U_j$. By induction hypothesis, for all $j < i$, $\Gamma_0 \vdash_c^< U_j : s_j$. Thus, by (acc), $\Gamma_0 \vdash_c^< y_j : U_j$. So, $\theta' : \Gamma' \rightarrow \Gamma_0$ in $\vdash_c^<$ and, by Lemma 86, $\Gamma_0 \vdash_c^< U_i : s_i$. ■

Lemma 88 If $\mathcal{R}$ and τ are compatible with $\geq_{\mathcal{F}}$, $\rightarrow$ is confluent and $\Gamma_0, \Gamma \vdash_f t : T$ then $\Gamma_0, \Gamma \vdash_c^< t : T$.

Proof. By induction on the size of Γ. Assume that $\Gamma_0, \Gamma = \vec{y} : \vec{U}$. Let $\vec{z}$ be $|\vec{y}|$ fresh variables, $\theta = \{\vec{y} \mapsto \vec{z}\}$, $\theta' = \{\vec{z} \mapsto \vec{y}\}$ and $\Delta = \vec{z} : \vec{U}\theta$. By Replacement, $\Delta \vdash_f t\theta : T\theta$. By Lemma 85, $\Gamma_0, \Delta \vdash_c^< t\theta : T\theta$. Now we show that $\theta' : \Gamma_0, \Delta \rightarrow \Gamma_0, \Gamma$ in $\vdash_c^<$ in order to conclude with Lemma 86.

We must show that, for all $x \in \mathrm{dom}(\Gamma_0, \Delta)$, $\Gamma_0, \Gamma \vdash_c^< x\theta' : x(\Gamma_0, \Delta)$. If $x \in \mathrm{dom}(\Gamma_0)$ then we have to show that $\Gamma_0, \Gamma \vdash_c^< x : x\Gamma_0$, and if $x = z_i \in \mathrm{dom}(\Delta)$ then we have to show that $\Gamma_0, \Gamma \vdash_c^< y_i : U_i$. So, it is sufficient to show that Γ_0, Γ

is valid in $\vdash_c^<$. If Γ is empty, this is immediate since, by Lemma 87, Γ_0 is valid in $\vdash_c^<$. Assume now that $\Gamma = \Gamma', y:U$. Then, $\Delta = \Delta', z:U\theta$. By the Environment Lemma, $\Gamma_0, \Gamma' \vdash_f U : s$. By induction hypothesis, $\Gamma_0, \Gamma' \vdash_c^< U : s$. Therefore, by (var), $\Gamma_0, \Gamma \vdash_c^< y : U$ and Γ_0, Γ is valid in $\vdash_c^<$. ■

A particular but useful case is :

Corollary 89 If $\mathcal{R}$ and τ are compatible with $\geq_\mathcal{F}$ and $\rightarrow$ is confluent then, for all $g \leq_\mathcal{F} f$, if $\vdash \tau_g : s$ then $\Gamma_0 \vdash_c^< \tau_g : s$.

Proof. Since τ is compatible with $\geq_\mathcal{F}$ and $\rightarrow$ is confluent, by Lemma 54, $\vdash_f \tau_g : s$. Thus, by Lemma 85, $\Gamma_0 \vdash_c^< \tau_g : s$. ■

Now, we can detail an example. Let us consider the rule :

$$appn(p, consn(x, n, \ell), n', \ell') \rightarrow consn(x, n + n', appn(n, \ell, n', \ell'))$$

with $\Gamma_0 = x:T, n:nat, \ell:listn(n), n':nat, \ell':listn(n')$ and $\rho = \{p \mapsto s(n)\}$. We take $stat_{appn} = lex(mul(x_2))$; $appn >_\mathcal{F} consn, +$; $consn >_\mathcal{F} T$ and $+ >_\mathcal{F} s, 0 >_\mathcal{F} nat$. We have already seen that this rule is well formed. Let us show that $\Gamma_0 \vdash_c r : listn(s(n))$. We have $\mathcal{R}$ and τ compatible with $\geq_\mathcal{F}$.

For applying (symb$^<$), we must show $\vdash \tau_{consn} : \star$, $\Gamma_0 \vdash_c \tau_{consn} : \star$, $\Gamma_0 \vdash_c x : T$, $\Gamma_0 \vdash_c n + n' : nat$ and $\Gamma_0 \vdash_c appn(n, \ell, n', \ell') : listn(n + n')$. It is easy to check that $\vdash \tau_{consn} : \star$. Then, by Lemma 89, we deduce that $\Gamma_0 \vdash_c \tau_{consn} : \star$. The assertions $\Gamma_0 \vdash_c x : T$ and $\Gamma_0 \vdash_c n + n' : nat$ come from Lemma 88. We are left to show that $\Gamma_0 \vdash_c appn(n, \ell, n', \ell') : listn(n + n')$.

For applying (symb$^=$), we must show that $\vdash \tau_{appn} : \star$, $\Gamma_0 \vdash_c \tau_{appn} : \star$, $\Gamma_0 \vdash_c n : nat$, $\Gamma_0 \vdash_c \ell : listn(n)$, $\Gamma_0 \vdash_c n' : nat$, $\Gamma_0 \vdash_c \ell' : listn(n')$ and $consn(x, n, \ell) : listn(s(n)) >_1 \ell : listn(n)$. It is easy to check that $\vdash \tau_{appn} : \star$. Then, by Lemma 89, we deduce that $\Gamma_0 \vdash_c \tau_{appn} : \star$. The assertion $consn(x, n, \ell) : listn(p) >_1 \ell : listn(n)$ has already be shown for proving that the rule is well formed. The other assertions come from Lemma 88.

6.4 Strong normalization conditions

Definition 90 (Rewrite systems) Let $\mathcal{G}$ be a set of symbols. The *rewrite system* $(\mathcal{G}, \mathcal{R}_\mathcal{G})$ is :

- *of first-order* if :
  - $\mathcal{G}$ is made of predicate symbols of maximal arity or of constructors of primitive predicates,
  - the rules of $\mathcal{R}_\mathcal{G}$ have an algebraic right hand-side;
- *non-duplicating* if, for all rule of $\mathcal{R}_\mathcal{G}$, no variable has more occurrences in the right hand-side than in the left hand-side;
- *primitive* if all the rules of $\mathcal{R}_\mathcal{G}$ have a right hand-side of the form $[\vec{x} : \vec{T}]g(\vec{u})\vec{v}$ with g a symbol of $\mathcal{G}$ or a primitive predicate;
- *simple* if there is no critical pairs between $cR_\mathcal{G}$ and $\mathcal{R}$:

  – no matching on defined symbols,
  – no ambiguity in the application of rules;
- *small* if, for all rule $g(\vec{l}) \to r \in \mathcal{R}_\mathcal{G}$ and all $X \in \mathrm{FV}^\Box(r)$, there exists κ_X such that $l_{\kappa_X} = X$;
- *positive* if, for all symbol $g \in \mathcal{G}$ and all rule $l \to r \in \mathcal{R}_\mathcal{G}$, $\mathrm{Pos}(g,r) \subseteq \mathrm{Pos}^+(r)$;
- *safe* if, for all rule $(g(\vec{l}) \to r, \Gamma, \rho) \in \mathcal{R}_\mathcal{G}$ with $\tau_g = (\vec{x} : \vec{T})U$ and $\gamma = \{\vec{x} \mapsto \vec{l}\}$:
  – for all $X \in \mathrm{FV}^\Box(\vec{T}U)$, $X\gamma\rho \in \mathrm{dom}^\Box(\Gamma)$,
  – for all $X, X' \in \mathrm{FV}^\Box(\vec{T}U)$, $X\gamma\rho = X'\gamma\rho \Rightarrow X = X'$.

Definition 91 (Strong normalization conditions)
(A0) All rules are well typed.
(A1) The relation $\to = \to_\mathcal{R} \cup \to_\beta$ is confluent on $\mathcal{T}$.
(A2) There exists an admissible inductive structure.
(A3) There exists a precedence $\succeq$ on $\mathcal{DF}^\Box$ with which $\mathcal{R}_{\mathcal{DF}^\Box}$ is compatible and whose equivalence classes form a system which is either :

  (p) primitive,
  (q) positive, small and simple,
  (r) recursive, small and simple.

(A4) There exists a partition $\mathcal{F}_1 \uplus \mathcal{F}_\omega$ of $\mathcal{DF}$ (*first-order* and *higher-order* symbols) such that :

  (a) $(\mathcal{F}_\omega, \mathcal{R}_\omega)$ is recursive,
  (b) $(\mathcal{F}_\omega, \mathcal{R}_\omega)$ is safe,
  (c) no symbol of $\mathcal{F}_\omega$ occurs in the rules of $\mathcal{R}_1$,
  (d) $(\mathcal{F}_1, \mathcal{R}_1)$ is of first-order,
  (e) if $\mathcal{R}_\omega \neq \emptyset$ then $(\mathcal{F}_1, \mathcal{R}_1)$ is non duplicating,
  (f) $\to_{\mathcal{R}_1}$ is strongly normalizing on $\mathbb{T}(\mathcal{F}_1, \mathcal{X})$.

The condition (A1) ensures, among other things, that β-reduction preserves typing while (A0) ensures that rewriting preserves typing. One can wonder whether confluence is necessary for proving that β-reduction preserves typing. H. Geuvers [58] has proved this property for the conversion relation $\mathcal{C} = \leftrightarrow^*_{\beta\eta}$ (instead of $\mathcal{C} = \downarrow_{\beta\mathcal{R}}$ here) while $\to_{\beta\eta}$ is not confluent on not well-typed terms. M. Fernández [54] has also proved this property for $\mathcal{C} = \to^*_{\beta\mathcal{R}} \cup {}^*_{\beta R}\!\leftarrow$ with $\to_\mathcal{R}$ being some rewriting at the object level only, without assuming that $\to_{\beta\mathcal{R}}$ is confluent. But this last result uses in an essential way the fact that rewriting is restricted to the object level. It is not clear how it can be extended to rewriting on types.

One should remember that hypothesis (A1) and (A3) are useful only in case of type-level rewriting.

The condition (A1) can seem difficult to fulfill since one often proves confluence using strong normalization and local confluence of critical pairs (result of D. Knuth and P. Bendix [18] for first order rewriting extended to higher-order rewriting by T. Nipkow [95]).

We know that $\rightarrow_\beta$ is confluent and that there is no critical pair between $\mathcal{R}$ and the β-reduction since the left hand-sides of the rules of $\mathcal{R}$ are algebraic.

F. Müller [92] has shown that, in this case, if $\rightarrow_\mathcal{R}$ is confluent and all the rules of $\mathcal{R}$ are left-linear, then $\rightarrow_\mathcal{R} \cup \rightarrow_\beta$ is confluent. Thus, the possibility we have introduced, of linearizing some rules (substitution ρ) while keeping subject reduction, appears to be very useful.

In the case of left-linear rules, and assuming that $\rightarrow_{\mathcal{R}_1}$ is strongly normalizing as it is required in (f), how can we prove that $\rightarrow$ is confluent? In the case where $\rightarrow_{\mathcal{R}_1}$ is non duplicating if $\mathcal{R}_\omega \neq \emptyset$, we show in Theorem 144 that $\rightarrow_{\mathcal{R}_1} \cup \rightarrow_{\mathcal{R}_\omega}$ is strongly normalizing. Therefore, it suffices to check that the critical pairs of $\mathcal{R}$ are confluent (without using any β-reduction).

In (A4), in the case where $\mathcal{R}_\omega \neq \emptyset$, we require that the rules of $\mathcal{R}_1$ are non-duplicating. Indeed, with first-order rewriting already, strong normalization is not a modular property [114], even with confluent systems [53]. On the other hand, it is modular for disjoint and non duplicating systems [104]. Here, $\mathcal{R}_1$ and $\mathcal{R}_\omega$ are not disjoint but hierarchically defined : by (c), no symbol of $\mathcal{F}_\omega$ occurs in the rules of $\mathcal{R}_1$. In [43], N. Dershowitz gathers many results on the modularity of strong normalization for first-order rewrite systems, especially for hierarchically defined systems. It would be very interesting to study the modularity of strong normalization in the case of higher-order rewriting and, in particular, other conditions than non-duplication which, for example, does not allow us to accept the following definition :

$$\begin{array}{rcl} 0/y & \rightarrow & 0 \\ s(x)/y & \rightarrow & s((x-y)/y) \\[4pt] 0-y & \rightarrow & 0 \\ s(x)-0 & \rightarrow & s(x) \\ s(x)-s(y) & \rightarrow & x-y \end{array}$$

This system is a duplicating first-order system not satisfying the General Schema: it can be put neither in $\mathcal{R}_1$ nor in $\mathcal{R}_\omega$. E. Giménez [61] can deal with this example by using the fact that the result of $x-y$ is smaller than $s(x)$.

In (A3), the smallness condition for recursive and positive systems is equivalent in the Calculus of Inductive Constructions to the restriction of strong elimination to "small" inductive types, that is, the types whose constructors have no other predicate arguments than the ones of their type. For example, the type *list* of polymorphic list is small since, in $(A\!:\!\star)A \rightarrow list(A) \rightarrow list(A)$, the type of its constructor *cons*, A is an argument of *list*. On the other hand, a type T having a constructor c of type $\star \rightarrow T$ is not small. So, we cannot define a function f of type $T \rightarrow \star$ with the rule $f(c(A)) \rightarrow A$. Such a rule is not small and does not form a primitive system either. In some sense, primitive systems can always be considered as small systems since they contain no projection and primitive predicate symbols have no predicate argument.

Finally, in (A4), the safeness condition for higher-order symbols means that one cannot do matching or equality tests on predicate arguments that are necessary for typing other arguments. In her extension of HORPO [76] to the Calculus of Constructions, D. Walukiewicz [118] requires a similar condition. She gives several (pathological) examples of non termination because of non-safeness like $J(A, A, a) \rightarrow a$ with $J : (A\!:\!\star)(B\!:\!\star)B \rightarrow A$ or $J(A, A, a, b) \rightarrow b$ with $J : (A\!:\!\star)(B\!:\!\star)A \rightarrow B \rightarrow A$. On the other hand, the rule $map(A, A, [x : A]x, \ell) \rightarrow \ell$, which does not seem problematic, does not satisfy the safeness condition either (note that the left hand-side if not algebraic).

We can now state our main result :

THEOREM : If a CAC satisfies the conditions of Definition 91 then its reduction relation $\rightarrow = \rightarrow_{\mathcal{R}} \cup \rightarrow_\beta$ preserves typing and is strongly normalizing.

The proof of this theorem is the subject of Chapter 8. This generalizes the results of M. Fernández [54] and of J.-P. Jouannaud, M. Okada and myself [22]. In Chapter 7, we give several important examples of CAC's satisfying these conditions : an important subsystem with strong elimination of the Calculus of Inductive Constructions (CIC) and the Natural Deduction Modulo (NDM) a large class of equational theories.

On the other hand, these conditions do not capture the decision procedure for classical propositional tautologies of Figure 1.3. Let us see why :

- We did not consider rewriting modulo associativity and commutativity.
- Since the system is not left-linear, we do not know how to prove the confluence of its combination with β.
- The system is not primitive since there are projections ($P \,\mathrm{xor}\, \bot \rightarrow P$). It is recursive (and positive also) and small. Unfortunately, it is not simple.

Rewriting modulo AC does not seem to be a difficult extension, except perhaps in the case of type-level rewriting. On the other hand, confluence and simplicity are problems which seem difficult but we expect to solve them in the future. In Section 9, we give other directions for future work but these three problems are certainly the most important ones.

From strong normalization, we can deduce the decidability of the typing relation, which is the essential property on which proof assistants like Coq [112] and LEGO [82] are based.

Theorem 92 (Decidability of $\vdash$) Let Γ be a valid environment and T be $\Box$ or a term typable in Γ. In a CAC satisfying the conditions of Definition 91, checking whether a term t is of type T in Γ is decidable.

Proof. Since Γ is valid, it is possible to say whether t is typable and, if so, it is possible to infer a type T' for t. Since types are convertible, it suffices to check that

T and T' have the same normal form. The reader is invited to look at [35, 11] for more details. ■

Remark 93 (Logical consistency)

In the pure Calculus of Constructions (CC), it is easy to check that, in the empty environment, no normal proof of $\bot = (P:\star)P$ can exist [10]. Therefore, for CC, strong normalization is sufficient for proving the logical consistency.

On the other hand, in a CAC, the situation is not so simple. From a logical point of view, having symbols is equivalent to working in a non empty environment. Therefore, it is possible that symbols and rules allow one to build a normal proof of $\bot$. In [106], J. Seldin shows the logical consistency of the "strongly consistent" environments [3] by syntactical means. However, for proving the consistency of more complex environments, it may be necessary to use semantic methods.

[3] An environment Γ is strongly consistent if, for all $x \in \mathrm{dom}(\Gamma)$, either $x\Gamma$ is a predicate type, or $x\Gamma$ is β-equivalent to a term of the form $y\vec{t}$.

Chapter 7

Examples of CAC's

7.1 Calculus of Inductive Constructions (CIC)

We are going to show that our conditions of strong normalization capture most of the Calculus of Inductive Constructions (CIC) of B. Werner [119] which is the basis of the proof assistant Coq [112]. But, since CIC is expressed in a formalism different from ours, we need to translate CIC into our formalism in order to check our conditions. It is a little bit long and painful but not very difficult.

In order to type the strong elimination schema in a polymorphic way, which is not possible in the usual Calculus of Constructions, B. Werner uses a slightly more general type system with the sorts $\mathcal{S} = \{\star, \Box, \triangle\}$, the axioms $\mathcal{A} = \{(\star, \Box), (\Box, \triangle)\}$ and the rules $\mathcal{B} = \{(s_1, s_2, s_3) \in \mathcal{S}^3 \mid s_1 \in \{\star, \Box\}, s_2 = s_3\}$ (in fact, he denotes $\star$ by Set, $\Box$ by Type and $\triangle$ by Extern).

Then he adds terms for representing inductive types, their constructors and the definitions by recursion on these types :

- **inductive types** : An inductive type is denoted by $I = Ind(X:A)\{\vec{C}\}$ where the C_i's are the types of the constructors of I. For example, $Nat = Ind(X:\star)\{X, X \rightarrow X\}$ represents the type of natural numbers (in fact, any type isomorphic to the type of natural numbers). The term A must be of the form $(\vec{x} : \vec{A})\star$ and the C_i's of the form $(\vec{z} : \vec{B})X\vec{m}$ with $X \notin \mathrm{FV}(\vec{m})$. Furthermore, the inductive types must be strictly positive. In CIC, this means that, if $C_i = (\vec{z} : \vec{B})X\vec{m}$ then, for all j, either X does not appear in B_j, or B_j is of the form $(\vec{y} : \vec{D})X\vec{q}$ and X does not appear neither in $\vec{D}$ nor in $\vec{q}$.
- **constructors** : The i-th constructor of an inductive type I is denoted by $Constr(i, I)$. For example, $Constr(1, Nat)$ represents zero and $Constr(2, Nat)$ represents the successor function.
- **definitions by recursion** : A definition by recursion on an inductive type I is denoted by $Elim(I, Q, \vec{a}, c)$ where Q is the type of the result, $\vec{a}$ the arguments of I and c a term of type $I\vec{a}$. The strong elimination (that is, in the case where Q is a predicate type) is restricted to "small" inductive types, that is, the types whose constructors do not have predicate arguments that their type do not have. More precisely, an inductive type $I = Ind(X : A)\{\vec{C}\}$ is *small* if all the types

of its constructors are small and a constructor type $C = (\vec{z} : \vec{B})X\vec{m}$ is *small* if $\{\vec{z}\} \cap \mathcal{X}^\Box = \emptyset$ (this means that the predicate arguments must be part of the environment in which they are typed; they cannot be part of $\vec{C}$).

For defining the reduction relation associated with *Elim*, called *ι-reduction* and denoted $\rightarrow_\iota$, and the typing rules of *Elim* (see Figure 7.1), it is necessary to introduce a few definitions.

Let C be a constructor type. We define $\Delta\{I, X, C, Q, c\}$ as follows :

- $\Delta\{I, X, X\vec{m}, Q, c\} = Q\vec{m}c$
- $\Delta\{I, X, (z\!:\!B)D, Q, c\} = (z\!:\!B)\Delta\{I, X, D, Q, cz\}$ if $X \notin \mathrm{FV}(B)$
- $\Delta\{I, X, (z\!:\!B)D, Q, c\} = (z\!:\!B\{X \mapsto I\})((\vec{y}\!:\!\vec{D})Q\,\vec{q}\,(z\vec{y})) \rightarrow \Delta\{I, X, D, Q, cz\}$ if $B = (\vec{y}\!:\!\vec{D})X\vec{q}$

The *ι-reduction* is defined by the rule :

$$Elim(I, Q, \vec{x}, Constr(i, I')\,\vec{z})\{\vec{f}\} \;\rightarrow_\iota\; \Delta[I, X, C_i, f_i, FunElim(I, Q, \vec{f})]\,\vec{z}$$

where $I = Ind(X\!:\!A)\{\vec{C}\}$, $FunElim(I, Q, \vec{f}) = [\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}]Elim(I, Q, \vec{x}, y)\{\vec{f}\}$ and $\Delta[I, X, C, f, F]$ is defined as follows :

- $\Delta[I, X, X\vec{m}, f, F] = f$
- $\Delta[I, X, (z\!:\!B)D, f, F] = [z\!:\!B]\Delta[I, X, D, fz, F]$ if $X \notin \mathrm{FV}(B)$
- $\Delta[I, X, (z\!:\!B)D, f, F] = [z\!:\!B\{X \mapsto I\}]\Delta[I, X, D, fz\,[\vec{y}\!:\!\vec{D}](F\vec{q}\,(z\vec{y})), F]$ if $B = (\vec{y}\!:\!\vec{D})X\vec{q}$

Finally, in the type conversion rule (conv), in addition to β-reduction and ι-reduction, B. Werner considers η-reduction : $[x\!:\!T]ux \rightarrow_\eta u$ if $x \notin \mathrm{FV}(u)$. Since $\rightarrow_{\beta\eta}$ is not confluent on badly typed terms[1], to consider η-reduction creates important difficulties [58]. Therefore, since our condition (A1) cannot be satisfied with η-reduction, we cannot consider η-reduction. To find a condition weaker than (A1) that would be satisfied even with η-reduction is a problem that we have temporally left open.

The ι-reduction as defined above introduces many β-redexes and the recursive calls on *Elim* are made on bound variables which must be instanciated by strict subterms (or terms of smaller order in case of a strictly positive inductive type). So, on one hand, from a practical point of view, this is not very efficient since these instanciations could be done immediately after the ι-reduction, and on the other hand, the General Schema cannot directly deal with recursive calls on bound variables, even though these variables must be instanciated with smaller terms.

This is why we are not going to show the strong normalization of the relation $\rightarrow_{\beta\iota}$ but of the relation $\rightarrow_{\beta\iota'}$ where one step of $\rightarrow_{\iota'}$ corresponds to a ι-reduction followed by as many β-reductions as necessary for erasing the β-redexes introduced by the ι-reduction. Note that this is indeed this reduction relation which is actually implemented in the Coq system [112].

Definition 94 (ι'-reduction) The *ι′-reduction* is the reduction relation defined by the rule :

[1] Remark due to R. Nederpelt [93] : $[x\!:\!A]x \;{}_\beta\!\leftarrow [x\!:\!A]([y\!:\!B]y\ x) \rightarrow_\eta [y\!:\!B]y$.

Figure 7.1: Typing rules of CIC

(Ind) $$\frac{A = (\vec{x} : \vec{A})\star \quad \Gamma \vdash A : \Box \quad \forall i, \Gamma, X : A \vdash C_i : \star}{\Gamma \vdash \mathit{Ind}(X\!:\!A)\{\vec{C}\} : A}$$

(Constr) $$\frac{I = \mathit{Ind}(X\!:\!A)\{\vec{C}\} \quad \Gamma \vdash I : T}{\Gamma \vdash \mathit{Constr}(i,I) : C_i\{X \mapsto I\}}$$

($\star$-Elim) $$\frac{\begin{array}{c} A = (\vec{x} : \vec{A})\star \quad I = \mathit{Ind}(X\!:\!A)\{\vec{C}\} \\ \Gamma \vdash Q : (\vec{x} : \vec{A})I\vec{x} \to \star \\ T_i = \Delta\{I, X, C_i, Q, \mathit{Constr}(i,I)\} \\ \gamma = \{\vec{x} \mapsto \vec{a}\} \quad \forall j, \Gamma \vdash a_j : A_j\gamma \quad \Gamma \vdash c : I\vec{a} \quad \forall i, \Gamma \vdash f_i : T_i \end{array}}{\Gamma \vdash \mathit{Elim}(I, Q, \vec{a}, c)\{\vec{f}\} : Q\vec{a}c}$$

($\Box$-Elim) $$\frac{\begin{array}{c} A = (\vec{x} : \vec{A})\star \quad I = \mathit{Ind}(X\!:\!A)\{\vec{C}\} \text{ is small} \\ \Gamma \vdash Q : (\vec{x} : \vec{A})I\vec{x} \to \Box \\ T_i = \Delta\{I, X, C_i, Q, \mathit{Constr}(i,I)\} \\ \gamma = \{\vec{x} \mapsto \vec{a}\} \quad \forall j, \Gamma \vdash a_j : A_j\gamma \quad \Gamma \vdash c : I\vec{a} \quad \forall i, \Gamma \vdash f_i : T_i\} \end{array}}{\Gamma \vdash \mathit{Elim}(I, Q, \vec{a}, c)\{\vec{f}\} : Q\vec{a}c}$$

(Conv) $$\frac{\Gamma \vdash t : T \quad T \leftrightarrow^*_{\beta\eta\iota} T' \quad \Gamma \vdash T' : s}{\Gamma \vdash t : T'}$$

$$\mathit{Elim}(I, Q, \vec{x}, \mathit{Constr}(i, I')\,\vec{z})\{\vec{f}\} \;\to_{\iota'}\; \Delta'[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$$

where $I = \mathit{Ind}(X\!:\!A)\{\vec{C}\}$ and $\Delta'[I, X, C, f, Q, \vec{f}, \vec{z}]$ is defined as follows :

– $\Delta'[I, X, X\vec{m}, f, Q, \vec{f}, \emptyset] = f$
– $\Delta'[I, X, (z\!:\!B)D, f, Q, \vec{f}, z\vec{z}] = \Delta'[I, X, D, fz, Q, \vec{z}]$ if $X \notin \mathrm{FV}(B)$
– $\Delta'[I, X, (z\!:\!B)D, f, Q, \vec{f}, z\vec{z}] = \Delta'[I, X, D, fz\,[\vec{y}\!:\!\vec{D}]\mathit{Elim}(I, Q, \vec{q}, z\vec{y}), Q, \vec{z}]$
  if $B = (\vec{y}\!:\!\vec{D})X\vec{q}$

We think that the strong normalization of $\to_{\beta\iota'}$ implies the strong normalization of $\to_{\beta\iota}$. But, since this problem does not seem very easy to solve and is not directly related to our work, we leave its resolution for the moment.

Conjecture 95 If $\to_{\beta\iota'}$ is strongly normalizing then $\to_{\beta\iota}$ is strongly normalizing.

Definition 96 (Admissible inductive type) An inductive type $I = \mathit{Ind}(X : A)$ $\{\vec{C}\}$ is *admissible* if it satisfies the conditions (I5), (I6) (adapted to the syntax of CIC, a strong elimination being considered as a defined predicate symbol) and the following *safeness* condition : if $A = (\vec{x} : \vec{A})\star$ and $C_i = (\vec{z} : \vec{B})X\vec{m}$ then :

– $\forall x_i \in \mathcal{X}^\Box$, $m_i \in \mathcal{X}^\Box$,
– $\forall x_i, x_j \in \mathcal{X}^\Box$, $m_i = m_j \Rightarrow x_i = x_j$.

Definition 97 (CIC$^-$) The sub-system of CIC that we are going to consider, CIC$^-$, can be obtained by applying the following restrictions :

- We exclude any use of the sort $\triangle$ in order to stay in the Calculus of Constructions.
- In the rule (Ind), instead of requiring $I = Ind(X\!:\!A)\{\vec{C}\}$ to be typable in any environment Γ, we require I to be typable in the empty environment since, in CAC, the types of the symbols must be typable in the empty environment. Moreover, we require I to be admissible and in normal form.

  The restriction to the empty environment is not a real restriction since any type $I = Ind(X : A)\{\vec{C}\}$ typable in an environment $\Gamma = \vec{y} : \vec{U}$ can be replaced by a type $I' = Ind(X' : A')\{\vec{C'}\}$ typable in the empty environment : it suffices to take $A' = (\vec{y} : \vec{U})A$, $C'_i = (\vec{y} : \vec{U})C_i\{X \mapsto X'\vec{y}\}$ and to replace I by $I'\vec{y}$ and $Constr(i, I)$ by $Constr(i, I')\vec{y}$.

  But we need to adapt the definition of *small* constructor type as follows : a constructor type C of an inductive type $I = Ind(X\!:\!A)\{\vec{C}\}$ with $A = (\vec{x} : \vec{A})\star$ is *small* if it is of the form $(\vec{x} : \vec{A})(\vec{z} : \vec{B})X\vec{m}$ with $\{\vec{z}\} \cap \mathcal{X}^\square = \emptyset$.
- In the rule ($\star$-Elim), instead of requiring Q to be typable in any environment Γ, we require Q to be typable in the empty environment. Moreover, we explicitly require I and $T_i = \Delta\{I, X, C_i, Q, Constr(i, I)\}$ to be typable.
- In the rule ($\square$-Elim), instead of requiring $\vdash Q : (\vec{x} : \vec{A})I\vec{x} \to \square$, which is not possible in the Calculus of Constructions, we require Q to be of the form $[\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}]K$ with $\vec{x} : \vec{A}, y\!:\!I\vec{x} \vdash K : \square$ and f_i to be of type $T_i = \Delta'\{I, X, C_i, \vec{x}y, K, Constr(i, I)\}$ where $\Delta'\{I, X, C, \vec{x}y, K, c\}$ is defined as follows :
  - $\Delta'\{I, X, X\vec{m}, \vec{x}y, K, c\} = K\{\vec{x} \mapsto \vec{m}, y \mapsto c\}$,
  - if $B = (\vec{y} : \vec{D})X\vec{q}$ then $\Delta'\{I, X, (z : B)D, \vec{x}y, K, c\} = (z : B\{X \mapsto I\})((\vec{y} : \vec{D})K\{\vec{x} \mapsto \vec{q}, y \mapsto z\vec{y}\}) \to \Delta'\{I, X, D, \vec{x}y, K, cz\}$.

  Moreover, we require Q to be in normal form, T_i to be typable and the inductive types that occur in Q to be subterms of I. Finally, we take $\Gamma \vdash Elim(I, Q, \vec{a}, c)\{\vec{f}\} : K\{\vec{x} \mapsto \vec{a}, y \mapsto c\}$ instead of $\Gamma \vdash Elim(I, Q, \vec{a}, c)\{\vec{f}\} : Q\vec{a}c$.

  Requiring Q to be of the form $[\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}]K$ is not a real restriction since, as shown by B. Werner (Corollary 2.9 page 57 of [119]), if $\Gamma \vdash Q : \square$ then there exists Q' of the form $(\vec{y} : \vec{U})\star$ such that $Q \to_\beta^* Q'$. Hence, if $\vdash Q : (\vec{x} : \vec{A})I\vec{x} \to \square$ then $\vec{x}\!:\!\vec{A}, y\!:\!I\vec{x} \vdash Q\vec{x}y : \square$. Therefore, there exists Q' of the form $(\vec{y} : \vec{U})\star$ such that $Q\vec{x}y \to_\beta^* Q'$. Then, $[\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}]Q\vec{x}y \to_\beta^* [\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}]Q'$ and $[\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}]Q\vec{x}y \to_\eta^* Q$. Therefore, by confluence, there exists Q'' of the form $[\vec{x}\!:\!\vec{A}][y\!:\!I\vec{x}](\vec{y}\!:\!\vec{U'})\star$ such that $Q \to_{\beta\eta}^* Q''$.

  On the other hand, requiring the inductive types occurring in Q to be subterms of I is a more important restriction. But it is only due to the fact that we restrict ourself to the Calculus of Constructions in which it is not possible to type the strong elimination schema in a polymorphic way (that is why B. Werner used a slightly more general PTS).
- In the rule (conv), instead of requiring $T \leftrightarrow_{\beta\eta\iota}^* T'$, we require $T \leftrightarrow_{\beta\iota'}^* T'$ which is equivalent to $T \downarrow_{\beta\iota'} T'$ since $\to_{\beta\iota'}$ is confluent (orthogonal CRS [79]).

We will denote by $\to_{\beta\iota'}$ the reduction relation of CIC$^-$, by $\mathcal{NF}$ the set of CIC$^-$ terms in normal form for $\to_{\beta\iota'}$ (unique by confluence), by $t\downarrow$ the normal form of t, and by $\vdash$ the typing relation of CIC$^-$ (see Figure 7.2).

Figure 7.2: Typing rules of CIC$^-$

$$\text{(Ind)}\quad \frac{A = (\vec{x}:\vec{A})\star \quad \vdash A:\Box \quad \forall i,\ X:A \vdash C_i:\star \qquad I = Ind(X{:}A)\{\vec{C}\} \in \mathcal{NF} \text{ is admissible}}{\vdash I : A}$$

$$\text{(Constr)}\quad \frac{I = Ind(X{:}A)\{\vec{C}\} \quad \Gamma \vdash I : T}{\Gamma \vdash Constr(i,I) : C_i\{X \mapsto I\}}$$

$$(\star\text{-Elim})\quad \frac{\begin{array}{c} A = (\vec{x}:\vec{A})\star \quad I = Ind(X{:}A)\{\vec{C}\} \quad \Gamma \vdash I : T \\ \vdash Q : (\vec{x}:\vec{A})I\vec{x} \to \star \\ T_i = \Delta\{I, X, C_i, Q, Constr(i,I)\} \quad \vdash T_i : \star \\ \gamma = \{\vec{x} \mapsto \vec{a}\} \quad \forall j, \Gamma \vdash a_j : A_j\gamma \quad \Gamma \vdash c : I\vec{a} \quad \forall i, \Gamma \vdash f_i : T_i \end{array}}{\Gamma \vdash Elim(I, Q, \vec{a}, c)\{\vec{f}\} : Q\vec{a}c}$$

$$(\Box\text{-Elim})\quad \frac{\begin{array}{c} A = (\vec{x}:\vec{A})\star \quad I = Ind(X{:}A)\{\vec{C}\} \\ Q = [\vec{x}{:}\vec{A}][y{:}I\vec{x}]K \in \mathcal{NF} \quad \vec{x}:\vec{A}, y : I\vec{x} \vdash K : \Box \\ \text{the inductive types of } Q \text{ are subterms of } I \\ T_i = \Delta'\{I, X, C_i, \vec{x}y, K, Constr(i,I)\} \quad \vdash T_i : \Box \\ \gamma = \{\vec{x} \mapsto \vec{a}\} \quad \forall j, \Gamma \vdash a_j : A_j\gamma \quad \Gamma \vdash c : I\vec{a} \quad \forall i, \Gamma \vdash f_i : T_i \end{array}}{\Gamma \vdash Elim(I, Q, \vec{a}, c)\{\vec{f}\} : K\{\vec{x} \mapsto \vec{a}, y \mapsto c\}}$$

$$\text{(Conv)}\quad \frac{\Gamma \vdash t : T \quad T \leftrightarrow^*_{\beta\iota'} T' \quad \Gamma \vdash T' : s}{\Gamma \vdash t : T'}$$

Theorem 98 There exists a CAC Υ (with typing relation $\vdash_\Upsilon$ and reduction relation $\to$) satisfying the conditions of Definition 91 and a function $\langle \_ \rangle$ which, to a CIC$^-$ term, associates a Υ term such that :

– if $\Gamma \vdash t : T$ then $\langle\Gamma\rangle \vdash_\Upsilon \langle t\rangle : \langle T\rangle$,

– moreover, if $t \to_{\beta\iota'} t'$ then $\langle t\rangle \to^+ \langle t'\rangle$.

Hence, $\to_{\beta\iota'}$ is strongly normalizing in CIC$^-$.

Definition 99 (Translation) We define $\langle t\rangle$ on the well-typed terms, by induction on $\Gamma \vdash t : T$:

- Let $I = Ind(X{:}A)\{\vec{C}\}$ with $A = (\vec{x}:\vec{A})\star$. We take $\langle I\rangle = [\vec{x}{:}\langle\vec{A}\rangle]Ind_I(\vec{x})$ where Ind_I is a symbol of type $\langle A\rangle$.
- By hypothesis, $C_i = (\vec{z}:\vec{B})X\vec{m}$. We take $\langle Constr(i,I)\rangle = [\vec{z}{:}\langle\vec{B}\rangle\theta']Constr^I_i(\vec{z})$ where $\theta' = \{X \mapsto \langle I\rangle\}$, $Constr^I_i$ is a symbol of type $(\vec{z}{:}\vec{B}')Ind_I(\langle\vec{m}\rangle)$, $B'_j = \langle B_j\rangle$ if X does not occur in B_j, and $B'_j = (\vec{y}{:}\langle\vec{D}\rangle)Ind_I(\langle\vec{q}\rangle)$ if $B_j = (\vec{y}{:}\vec{D})X\vec{q}$.
- Let Q be a term not of the form $[\vec{x}:\vec{A}][y:I\vec{x}]K$ with $K = (\vec{y}:\vec{U})\star$. We take $\langle Elim(I, Q, \vec{a}, c)\{\vec{f}\}\rangle = WElim_I(\langle Q\rangle, \langle\vec{a}\rangle, \langle c\rangle, \langle\vec{f}\rangle)$ where $WElim_I$ is a sym-

bol of type $(Q:(\vec{x}:\langle\vec{A}\rangle)\ \langle I\rangle\vec{x}\rightarrow\star)(\vec{x}:\langle\vec{A}\rangle)(y:\langle I\rangle\vec{x})(\vec{f}:\langle\vec{T}\rangle)\langle Q\rangle\vec{x}y$ and $T_i = \Delta\{I, X, C_i, Q, Constr(i, I)\}$.

- Let Q be a term of the form $[\vec{x}:\vec{A}][y:I\vec{x}]K$ with $K = (\vec{y}:\vec{U})\star$. We take $\langle Elim(I, Q, \vec{a}, c)\{\vec{f}\}\rangle = SElim_I^Q(\langle\vec{a}\rangle, \langle c\rangle, \langle\vec{f}\rangle)$ where $SElim_I^Q$ is a symbol of type $(\vec{x}:\langle\vec{A}\rangle)(y:\langle I\rangle\vec{x})(\vec{f}:\langle\vec{T}\rangle)\langle K\rangle$, $T_i = \Delta'\{I, X, C_i, \vec{x}y, K, Constr(i, I)\}$.
- The other terms are defined recursively : $\langle uv\rangle = \langle u\rangle\langle v\rangle, \ldots$

Let Υ be the CAC whose symbols are those just previously defined and whose rules are :

$$\begin{array}{rcl} WElim_I(Q, \vec{x}, Constr_i^I(\vec{z}), \vec{f}) & \rightarrow & \Delta'_W[I, X, C_i, f_i, Q, \vec{f}, \vec{z}] \\ SElim_I^Q(\vec{x}, Constr_i^I(\vec{z}), \vec{f}) & \rightarrow & \Delta'_S[I, X, C_i, f_i, Q, \vec{f}, \vec{z}] \end{array}$$

where $\Delta'_W[I, X, C, f, Q, \vec{f}, \vec{z}]$ and $\Delta'_S[I, X, C, f, Q, \vec{f}, \vec{z}]$ are defined as follows :

- $\Delta'_W[I, X, X\vec{m}, f, Q, \vec{f}, \vec{z}] = \Delta'_S[I, X, X\vec{m}, f, Q, \vec{f}, \vec{z}] = f$,
- $\Delta'_S[I, X, (z:B)D, f, Q, \vec{f}, z\vec{z}] = \Delta'_S[I, X, D, f\,z, Q, \vec{f}, \vec{z}]$ and $\Delta'_W[I, X, (z:B)D, f, Q, \vec{f}, z\vec{z}] = \Delta'_W[I, X, D, f\,z, Q, \vec{f}, \vec{z}]$ if $X \notin \mathrm{FV}(B)$
- $\Delta'_S[I, X, (z:B)D, f, Q, \vec{f}, z\vec{z}] = \Delta'_S[I, X, D, f\,z\,[\vec{y}:\vec{D}]SElim_I^Q(\vec{f}, \vec{q}, z\vec{y}), Q, \vec{f}, \vec{z}]$ and $\Delta'_W[I, X, (z:B)D, f, Q, \vec{f}, z\vec{z}] =$ $\Delta'_W[I, X, D, f\,z\,[\vec{y}:\vec{D}]WElim_I(Q, \vec{f}, \vec{q}, z\vec{y}), Q, \vec{f}, \vec{z}]$ if $B = (\vec{y}:\vec{D})X\vec{q}$

Since $\rightarrow_{\beta\iota'}$ is confluent, the β-reduction has the subject reduction property in Υ. This will be useful for proving that the translation preserves typing :

Lemma 100 If $\Gamma \vdash t : T$ then $\langle\Gamma\rangle \vdash_\Upsilon \langle t\rangle : \langle T\rangle$.

Proof. By induction on $\Gamma \vdash t : T$.

(Ind) We have to prove that $\vdash_\Upsilon \langle I\rangle : \langle A\rangle$. We have $\langle I\rangle = [\vec{x}:\langle\vec{A}\rangle]Ind_I(\vec{x})$ with Ind_I of type $\langle A\rangle = (\vec{x}:\langle\vec{A}\rangle)\star$. Since $\vdash A : \Box$, by induction hypothesis, we have $\vdash_\Upsilon \langle A\rangle : \Box$, that is, $\vdash_\Upsilon \tau_{Ind_I} : \Box$. By inversion, we get $\vec{x}:\langle\vec{A}\rangle \vdash_\Upsilon \star : \Box$. Therefore, $\Gamma = \vec{x}:\langle\vec{A}\rangle$ is valid and, by the Environment Lemma and (weak), for all i, $\Gamma \vdash_\Upsilon x_i : \langle A_i\rangle$. Hence, by (symb), $\Gamma \vdash_\Upsilon Ind_I(\vec{x}) : \star$ and, by (abs), $\Gamma \vdash_\Upsilon \langle I\rangle : \langle A\rangle$.

(Constr) We have to prove that $\langle\Gamma\rangle \vdash_\Upsilon \langle Constr(i, I)\rangle : \langle C_i\theta\rangle$ where $\theta = \{X \mapsto I\}$. We have $C_i = (\vec{z}:\vec{B})\ X\vec{m}$, $\langle Constr(i, I)\rangle = [\vec{z}:\langle\vec{B}\rangle\theta']Constr_i^I(\vec{z})$, $\theta' = \{X \mapsto \langle I\rangle\}$, $Constr_i^I$ of type $(\vec{z}:\vec{B}')Ind_I(\langle\vec{m}\rangle)$, $B'_j = \langle B_j\rangle$ if X does not occur in B_j, $B'_j = (\vec{y}:\langle\vec{D}\rangle)Ind_I(\langle\vec{q}\rangle)$ if $B_j = (\vec{y}:\vec{D})X\vec{q}$, $\langle C_i\theta\rangle = \langle C_i\rangle\theta' = (\vec{z}:\langle\vec{B}\rangle\theta')\langle I\rangle\langle\vec{m}\rangle$ and $\langle I\rangle = (\vec{x}:\langle\vec{A}\rangle)Ind_I(\vec{x})$.

Hence, $\langle I\rangle\langle\vec{m}\rangle \rightarrow_\beta^* Ind_I(\langle\vec{m}\rangle)$. Moreover, if X does not occur in B_j then $B'_j = \langle B_j\rangle = \langle B_j\rangle\theta'$. If $B_j = (\vec{y}:\vec{D})X\vec{q}$ then $B'_j = (\vec{y}:\langle\vec{D}\rangle)Ind_I(\langle\vec{q}\rangle)$ and $\langle B_j\rangle\theta' = (\vec{y}:\langle\vec{D}\rangle)\langle I\rangle\langle\vec{q}\rangle$. Since, $\langle I\rangle\langle\vec{q}\rangle \rightarrow_\beta^* Ind_I(\langle\vec{q}\rangle)$, for all j, $\langle B_j\rangle\theta' \rightarrow_\beta^* B'_j$.

Since $\Gamma \vdash I : T$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle I\rangle : \langle T\rangle$ and $\langle\Gamma\rangle$ is valid. By inversion, $\vdash I : A$ and $X : A \vdash C_i : \star$. By inversion again, $X : A, \vec{z}:\vec{B} \vdash X\vec{m} : \star$. By induction hypothesis, $\vdash_\Upsilon \langle I\rangle : \langle A\rangle$ and $X : \langle A\rangle, \vec{z}:\langle\vec{B}\rangle \vdash_\Upsilon X\langle\vec{m}\rangle : \star$. By substitution, $\vec{z}:\langle\vec{B}\rangle\theta' \vdash_\Upsilon \langle I\rangle\langle\vec{m}\rangle : \star$. Therefore, $\Delta = \vec{z}:\langle\vec{B}\rangle\theta'$ is valid. Since $\langle\vec{B}\rangle\theta' \rightarrow_\beta^* \vec{B}'$, by subject reduction on the environments, $\Delta' = \vec{z}:\vec{B}'$ is also valid.

Therefore, $\vec{z} : \vec{B}' \vdash_\Upsilon \langle I\rangle\langle\vec{m}\rangle : \star$ and, by (prod), $\vdash_\Upsilon (\vec{z} : \vec{B}')Ind_I(\langle\vec{m}\rangle) : \star$, that is, $\vdash_\Upsilon \tau_{Constr_i^I} : \star$.

By the Environment Lemma and (conv), for all j, $\Delta \vdash_\Upsilon z_j : B'_j$. Therefore, by (symb), $\Delta \vdash_\Upsilon Constr_i^I(\vec{z}) : Ind_I(\langle\vec{m}\rangle)$ and, by (abs), $\vdash_\Upsilon \langle Constr(i,I)\rangle : (\vec{z} : \langle\vec{B}\rangle\theta')Ind_I(\langle\vec{m}\rangle)$. Finally, by (conv) and (weak), $\langle\Gamma\rangle \vdash_\Upsilon \langle Constr(i,I)\rangle : \langle C_i\theta\rangle$.

($\star$-Elim) We have to prove that $\langle\Gamma\rangle \vdash_\Upsilon \langle Elim(I,Q,\vec{a},c)\{\vec{f}\}\rangle : \langle Q\vec{a}c\rangle$. We have $\langle Elim(I,Q,\vec{a},c)\{\vec{f}\}\rangle = WElim_I(\langle Q\rangle, \langle\vec{a}\rangle, \langle c\rangle, \langle\vec{f}\rangle)$, $WElim_I$ of type $(Q : \langle B\rangle)(\vec{x} : \langle\vec{A}\rangle)(y : I\vec{x})(\vec{f} : \langle\vec{T}\rangle)\langle Q\rangle\vec{x}y$, $B = (\vec{x} : \vec{A})I\vec{x} \to \star$, $T_i = \Delta\{I, X, C_i, Q, Constr(i,I)\}$ and $\langle Q\vec{a}c\rangle = \langle Q\rangle\langle\vec{a}\rangle\langle c\rangle$. In order to apply (symb), we prove that (1) $\langle\Gamma\rangle \vdash_\Upsilon \langle Q\rangle : \langle B\rangle$, (2) $\langle\Gamma\rangle \vdash_\Upsilon \langle\vec{a}\rangle : \langle\vec{A}\rangle\gamma'$, (3) $\langle\Gamma\rangle \vdash_\Upsilon \langle c\rangle : \langle I\rangle\vec{x}\gamma'$, (4) $\langle\Gamma\rangle \vdash_\Upsilon \langle\vec{f}\rangle : \langle\vec{T}\rangle\gamma'$ and (5) $\vdash_\Upsilon \tau_{WElim_I} : \star$, where $\gamma' = \{Q \mapsto \langle Q\rangle, \vec{x} \mapsto \langle\vec{a}\rangle, y \mapsto \langle c\rangle, \vec{f} \mapsto \langle\vec{f}\rangle\}$. First of all, note that, since $\Gamma \vdash c : I\vec{a}$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle c\rangle : \langle I\vec{a}\rangle$ and $\langle\Gamma\rangle$ is valid.

(1) Since $\vdash Q : B$, by induction hypothesis, $\vdash_\Upsilon \langle Q\rangle : \langle B\rangle$. Therefore, by weakening, $\langle\Gamma\rangle \vdash_\Upsilon \langle Q\rangle : \langle B\rangle$.

(2) Since $\Gamma \vdash a_j : A_j\gamma$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle a_j\rangle : \langle A_j\gamma\rangle$. But $\langle A_j\gamma\rangle = \langle A_j\rangle\gamma'$ since $\mathrm{FV}(A_j) \subseteq \{\vec{x}\}$.

(3) Since $\Gamma \vdash c : I\vec{a}$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle c\rangle : \langle I\vec{a}\rangle$. But $\langle I\vec{a}\rangle = \langle I\rangle\langle\vec{a}\rangle = \langle I\rangle\vec{x}\gamma'$.

(4) Since $\Gamma \vdash f_i : T_i$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle f_i\rangle : \langle T_i\rangle$. But $\langle T_i\rangle\gamma' = \langle T_i\rangle$ since T_i is closed ($\vdash T_i : \star$).

(5) Let $\Delta = Q : \langle B\rangle, \vec{x} : \langle\vec{A}\rangle, (y : \langle I\rangle\vec{x})$ and $\Delta' = \Delta, \vec{f} : \langle\vec{T}\rangle$. We prove that Δ' is valid. Indeed, in this case, $\Delta' \vdash_\Upsilon \langle Q\rangle\vec{x}y : \star$ and, by (prod), $\vdash_\Upsilon \tau_{WElim_I} : \star$. We have $\vdash_\Upsilon \langle Q\rangle : \langle B\rangle$. Therefore, $\vdash_\Upsilon \langle B\rangle : \Box$. By (var), $Q : \langle B\rangle \vdash_\Upsilon Q : \langle B\rangle$. By inversion, $Q : \langle B\rangle, \vec{x} : \langle\vec{A}\rangle, y : \langle I\rangle\vec{x} \vdash_\Upsilon \star : \Box$ and Δ is valid. Let $\Delta_i = \Delta, f_1 : \langle T_1\rangle, \ldots, f_i : \langle T_i\rangle$. We prove by induction on i that Δ_i is valid. If $i = 0$, this is immediate. We now prove that if Δ_i is valid then Δ_{i+1} is valid too. Since $\vdash_\Upsilon \langle T_{i+1}\rangle : \star$, by weakening, $\Delta_i \vdash_\Upsilon \langle T_{i+1}\rangle : \star$. Therefore, by (var), $\Delta_{i+1} \vdash_\Upsilon f_{i+1} : T_{i+1}$ and Δ_{i+1} is valid.

($\Box$-Elim) We have to prove that $\langle\Gamma\rangle \vdash_\Upsilon \langle Elim(I,Q,\vec{a},c)\{\vec{f}\}\rangle : \langle K\rangle$. We have $\langle Elim(I,Q,\vec{a},c)\{\vec{f}\}\rangle = SElim_I^Q(\langle\vec{a}\rangle, \langle c\rangle, \langle\vec{f}\rangle)$, $SElim_I^Q$ of type $(\vec{x} : \langle\vec{A}\rangle)(y : I\vec{x})(\vec{f} : \langle\vec{T}\rangle)\langle K\rangle$ and $T_i = \Delta'\{I, X, C_i, Q, Constr(i,I)\}$. In order to apply (symb), we prove that (1) $\langle\Gamma\rangle \vdash_\Upsilon \langle\vec{a}\rangle : \langle\vec{A}\rangle\gamma'$, (2) $\langle\Gamma\rangle \vdash_\Upsilon \langle c\rangle : \langle I\rangle\vec{x}\gamma'$, (3) $\langle\Gamma\rangle \vdash_\Upsilon \langle\vec{f}\rangle : \langle\vec{T}\rangle\gamma'$ and (4) $\vdash_\Upsilon \tau_{SElim_I} : \Box$, where $\gamma' = \{\vec{x} \mapsto \langle\vec{a}\rangle, y \mapsto \langle c\rangle, \vec{f} \mapsto \langle\vec{f}\rangle\}$. First of all, note that, since $\Gamma \vdash c : I\vec{a}$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle c\rangle : \langle I\vec{a}\rangle$ and $\langle\Gamma\rangle$ is valid.

(1) Since $\Gamma \vdash a_j : A_j\gamma$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle a_j\rangle : \langle A_j\gamma\rangle$. But $\langle A_j\gamma\rangle = \langle A_j\rangle\gamma'$ since $\mathrm{FV}(A_j) \subseteq \{\vec{x}\}$.

(2) Since $\Gamma \vdash c : I\vec{a}$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle c\rangle : \langle I\vec{a}\rangle$. But $\langle I\vec{a}\rangle = \langle I\rangle\langle\vec{a}\rangle = \langle I\rangle\vec{x}\gamma'$.

(3) Since $\Gamma \vdash f_i : T_i$, by induction hypothesis, $\langle\Gamma\rangle \vdash_\Upsilon \langle f_i\rangle : \langle T_i\rangle$. But $\langle T_i\rangle\gamma' = \langle T_i\rangle$ since T_i is closed ($\vdash T_i : \Box$).

(4) Let $\Delta = \vec{x} : \langle\vec{A}\rangle, y : \langle I\rangle\vec{x}$ and $\Delta' = \Delta, \vec{f} : \langle\vec{T}\rangle$. We have $\Delta \vdash_\Upsilon \langle K\rangle : \Box$. We prove that Δ' is valid. Indeed, in this case, by weakening, $\Delta' \vdash_\Upsilon \langle K\rangle : \Box$ and,

by (prod), $\vdash_\Upsilon \tau_{SElim_I} : \star$. Let $\Delta_i = \Delta, f_1 : \langle T_1\rangle, \ldots, f_i : \langle T_i\rangle$. We prove by induction on i that Δ_i is valid. If $i = 0$, this is immediate. We now prove that if Δ_i is valid then Δ_{i+1} is valid too. Since $\vdash_\Upsilon \langle T_{i+1}\rangle : \Box$, by weakening, $\Delta_i \vdash_\Upsilon \langle T_{i+1}\rangle : \Box$. Therefore, by (var), $\Delta_{i+1} \vdash_\Upsilon f_{i+1} : T_{i+1}$ and Δ_{i+1} is valid.

The other cases can be treated without difficulties. ∎

Lemma 101 The rules of Υ are well typed.

Proof. We have to prove that the rules of Υ satisfy the conditions (S3) to (S5). We just see the case of $WElim_I(Q, \vec{x}, Constr_i^I(\vec{z}), \vec{f}) \to \Delta'_W[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$. The case of $SElim_I^Q(\vec{x}, Constr_i^I(\vec{z}), \vec{f}) \to \Delta'_S[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$ is similar. Let $B = (\vec{x} : \vec{A})I\vec{x} \to \star$. We have $\tau_{WElim_I} = (Q : \langle B\rangle)(\vec{x} : \langle\vec{A}\rangle)(y : \langle I\rangle\vec{x})(\vec{f} : \langle\vec{T}\rangle)Q\vec{x}y$, $T_i = \Delta\{I, X, C_i, Q, Constr(i, I)\}$, $C_i = (\vec{z} : \vec{B})X\vec{m}$, $B_j = (\vec{y}^j : \vec{D}^j)X\vec{q}^j$ if $X \in \mathrm{FV}(B_j)$, $\tau_{Constr_i^I} = (\vec{z} : \vec{B}')Ind_I(\langle\vec{m}\rangle)$, $B'_j = \langle B_j\rangle$ if $X \notin \mathrm{FV}(B_j)$, $B'_j = (\vec{y}^j : \langle\vec{D}^j\rangle)Ind_I(\langle\vec{q}^j\rangle)$ otherwise, and $\tau_{Ind_I} = (\vec{x} : \langle\vec{A}\rangle)\star$. Let $l = WElim_I(Q, \vec{x}, c, \vec{f})$, $r = \Delta'_W[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$, $c = Constr_i^I(\vec{z})$ and $\gamma = \{y \mapsto c\}$. We take $\Gamma = Q : \langle B\rangle, \vec{z} : \vec{B}', \vec{f} : \langle\vec{T}\rangle$ and $\rho = \{\vec{x} \mapsto \langle\vec{m}\rangle\}$.

(S2) We have to prove that $\Gamma \vdash_\Upsilon r : Q\langle\vec{m}\rangle c$. We have $r = \Delta'_W[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$ and $T_i = \Delta\{I, X, C_i, Q, Constr(i, I)\}$. There is no difficulty.

(S3) We have to prove that if $\Delta \vdash_\Upsilon l\sigma : T$ then $\sigma : \Gamma \to \Delta$. We have $\Delta \vdash_\Upsilon WElim_I(Q\sigma, \vec{x}\sigma, Constr_i^I(\vec{z}\sigma), \vec{f}\sigma) : T$. Then, by inversion, we deduce that $\Delta \vdash_\Upsilon Q\sigma : \langle B\rangle\sigma$, $\Delta \vdash_\Upsilon \vec{z}\sigma : \vec{B}'\sigma$ and $\Delta \vdash_\Upsilon \vec{f}\sigma : \langle\vec{T}\rangle\sigma$, that is, $\sigma : \Gamma \to \Delta$.

(S4) We have to prove that if $\Delta \vdash_\Upsilon l\sigma : T$ then, for all x, $x\rho\sigma \downarrow x\sigma$. By inversion, we have $\Delta \vdash_\Upsilon c\sigma : \langle I\rangle\vec{x}\sigma$ and $Ind_I(\langle\vec{m}\rangle\sigma)\ \mathcal{C}^*_\Delta\ \langle I\rangle\vec{x}\sigma$. Now, $\langle I\rangle\vec{x}\sigma \to^*_\beta Ind_I(\vec{x}\sigma)$. Therefore, $Ind_I(\langle\vec{m}\rangle\sigma)\ \mathcal{C}^*\ Ind_I(\vec{x}\sigma)$ and, by confluence, $Ind_I(\langle\vec{m}\rangle\sigma) \downarrow Ind_I(\vec{x}\sigma)$. Since Ind_I is constant and $\langle\vec{m}\rangle\sigma = \vec{x}\rho\sigma$, we get $\vec{x}\sigma \downarrow \vec{x}\rho\sigma$. ∎

Lemma 102 The rules of Υ are well formed.

Proof. Let us see the case of $WElim_I(Q, \vec{x}, Constr_i^I(\vec{z}), \vec{f}) \to \Delta'_W[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$. The case of $SElim_I^Q(\vec{x}, Constr_i^I(\vec{z}), \vec{f}) \to \Delta'_S[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$ is dealt with similarly. Let $B = (\vec{x} : \vec{A})I\vec{x} \to \star$. We have $\Gamma = Q : \langle B\rangle, \vec{z} : \vec{B}', \vec{f} : \langle\vec{T}\rangle$ and $\rho = \{\vec{x} \mapsto \langle\vec{m}\rangle\}$. We have to prove that each variable $x \in \mathrm{dom}(\Gamma)$ is weakly accessible in one of the arguments of $WElim_I$, that $x\Gamma$ is equal to $T\rho$ where T is the type of x derived from l and that $\Gamma \vdash_\Upsilon l\rho : (Q\vec{x}y)\gamma\rho$.

The accessibility is immediate for Q and $\vec{f}$. The z_j's are weakly accessible since all the positions of a constructor are accessible (see the definition of $\mathrm{Acc}(Constr_i^I)$). The type of z_j derived from l ia B'_j which does not depend on $\vec{x}$. Therefore, $B'_j\rho = B'_j = z_j\Gamma$.

Let us see $\Gamma \vdash_\Upsilon l\rho : (Q\vec{x}y)\gamma\rho$ now. We have $l\rho = WElim_I(Q, \langle\vec{m}\rangle, Constr_i^I(\vec{z}), \vec{f})$ and $(Q\vec{x}y)\gamma\rho = Q\langle\vec{m}\rangle c$. For applying (symb), we must prove (1) $\vdash_\Upsilon \tau_{WElim_I} : \star$, (2) $\Gamma \vdash_\Upsilon Q : \langle B\rangle$, (3) $\Gamma \vdash_\Upsilon \langle\vec{m}\rangle : \langle\vec{A}\rangle\rho$, (4) $\Gamma \vdash_\Upsilon c : \langle I\rangle\langle\vec{m}\rangle$ and (5) $\Gamma \vdash_\Upsilon \vec{f} : \langle\vec{T}\rangle$.

Let us prove first of all that Γ is valid. Note that $WElim_I$ is defined only if there exists a well-typed term of the form $Elim(I, Q', \vec{a}, c')\{\vec{f}\}$. And, in this case, we have $\vdash Q' : B$ and $\vdash \vec{T} : \star$. Therefore, $\vdash B : \Box$ and $\vdash_\Upsilon \langle B\rangle : \Box$. Hence,

$Q:\langle B\rangle$ is valid. Moreover, if $WElim_I$ is well defined then Ind_I is also well defined, and therefore $Constr_i^I$ too. But, we have proved in the previous lemma that, in this case, $\vdash_\Upsilon \tau_{Constr_i^I} : \star$. By weakening, $Q:\langle B\rangle \vdash_\Upsilon \tau_{Constr_i^I} : \star$. By inversion, $Q:\langle B\rangle, \vec{z}:\vec{B}' \vdash_\Upsilon Ind_I(\langle\vec{m}\rangle) : \star$ and $Q:\langle B\rangle, \vec{z}:\vec{B}'$ is valid. Finally, as $\vdash \vec{T} : \star$, $\vdash_\Upsilon \langle\vec{T}\rangle : \star$ and, by weakening, $Q:\langle B\rangle, \vec{z}:\vec{B}' \vdash_\Upsilon \langle\vec{T}\rangle : \star$. Therefore Γ is valid.

(1) Already proved in the previous lemma.

(2) By the Environment Lemma.

(3) From $\vec{z}:\vec{B}' \vdash_\Upsilon Ind_I(\langle\vec{m}\rangle) : \star$, by inversion, we deduce that $\vec{z}:\vec{B}' \vdash_\Upsilon \langle\vec{m}\rangle : \langle\vec{A}\rangle\rho$. Therefore, by weakening, $\Gamma \vdash_\Upsilon \langle\vec{m}\rangle : \langle\vec{A}\rangle\rho$.

(4) As $\Gamma \vdash_\Upsilon \vec{z}:\vec{B}'$, by (symb), $\Gamma \vdash_\Upsilon c : Ind_I(\langle\vec{m}\rangle)$. Moreover, $\langle I\rangle\langle\vec{m}\rangle \rightarrow_\beta^* Ind_I(\langle\vec{m}\rangle)$. After (3), $\Gamma \vdash_\Upsilon \langle\vec{m}\rangle : \langle\vec{A}\rangle\rho$. Therefore, by (app), $\Gamma \vdash_\Upsilon \langle I\rangle\langle\vec{m}\rangle : \star$ and, by (conv), $\Gamma \vdash_\Upsilon c : \langle I\rangle\langle\vec{m}\rangle$.

(5) By the Environment Lemma. ■

Lemma 103 Υ satisfies the conditions of strong normalization of Definition 91.

Proof.

(A0) After the previous lemma.

(A1) We have already seen that $\rightarrow$ is confluent.

(A2) For the inductive structure, we take :

– $Ind_I >_\mathcal{C} Ind_J$ if J is a strict subterm of I is a well-founded quasi-ordering,

– $\mathrm{Ind}(Ind_I) = \emptyset$,

– $\mathrm{Acc}(Constr_i^I) = \{1, \ldots, n\}$ where n is the arity of $Constr_i^I$.

We prove that this inductive structure is admissible. Let C be a constant predicate symbol. Then $C = Ind_I$ with $I = Ind(X:A)\{\vec{C}\}$ and $A = (\vec{x}:\vec{A})\star$, and C is of type $(\vec{x}:\langle\vec{A}\rangle)\star$. Let $c = Constr_i^I$ be one of the constructors of C. By hypothesis, $C_i = (\vec{z}:\vec{B})X\vec{m}$ and $B_j = (\vec{y}^j:\vec{D}^j)X\vec{q}^j$ if $X \in \mathrm{FV}(B_j)$. Therefore c is of type $(\vec{z}:\vec{B}')$ with $B'_j = \langle B_j\rangle$ if $X \notin \mathrm{FV}(B_j)$, and $B'_j = (\vec{y}^j:\langle\vec{D}^j\rangle)C(\langle\vec{q}^j\rangle)$ with $X \notin \mathrm{FV}(\vec{D}^j\vec{q}^j)$ otherwise. Let $\vec{v} = \vec{m}$, $j \in \mathrm{Acc}(c)$ and $U_j = B'_j$.

(I3) $\forall D \in \mathcal{CF}^\square, D =_\mathcal{C} C \Rightarrow \mathrm{Pos}(D, U_j) \subseteq \mathrm{Pos}^+(U_j)$. Necessary, $D = C$. Either $X \notin \mathrm{FV}(B_j)$ and $\mathrm{Pos}(C, U_j) = \emptyset$, or $B_j = (\vec{y}:\vec{D})X\vec{q}$ and $X \notin \mathrm{FV}(\vec{D}\vec{q})$. In every case, $\mathrm{Pos}(C, U_j) \subseteq \mathrm{Pos}^+(U_j)$.

(I4) $\forall D \in \mathcal{CF}^\square, D >_\mathcal{C} C \Rightarrow \mathrm{Pos}(D, U_j) \subseteq \mathrm{Pos}^0(U_j)$. If $D = Ind_J >_\mathcal{C} C = Ind_I$ then I is a strict subterm of J. Therefore, J cannot occur in I and $\mathrm{Pos}(D, U_j) = \emptyset$.

(I5) $\forall F \in \mathcal{DF}^\square, \mathrm{Pos}(F, U_j) \subseteq \mathrm{Pos}^0(U_j)$. By hypothesis on the types of CIC^-.

(I6) $\forall Y \in \mathrm{FV}^\square(U_j), \exists\, \iota_Y \leq \alpha_C, v_{\iota_Y} = Y$. By hypothesis on the types of CIC^-.

(I2) $\forall Y \in \mathrm{FV}^\square(U_j), \iota_Y \in \mathrm{Ind}(C) \Rightarrow \mathrm{Pos}(Y, U_j) \subseteq \mathrm{Pos}^+(U_j)$. Since $\mathrm{Ind}(C) = \emptyset$.

(A3) For $\succeq$, we take the equality. We prove that the rules defining $SElim_I^Q$ form a system which is :

– **recursive** : We prove this for all the symbols.

– **small** : We have $SElim_I^Q(\vec{x}, Constr_i^I(\vec{z}), \vec{f}) \rightarrow \Delta'_S[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$. We will denote by $\vec{l}$ the arguments of $SElim_I^Q$ and by r the right hand-side of the rule. We have to prove that, for all $X \in \mathrm{FV}^\square(r)$, there exists a unique κ_X such that $l_{\kappa_X} = X$. We have $\mathrm{FV}^\square(r) = \{\vec{f}\} \cup (\{\vec{z}\} \cap \mathcal{X}^\square)$. For f_j, this is immediate. For $z_j \in \mathcal{X}^\square$, this comes from the restriction of the strong elimination to small inductive types : $\vec{z} = \vec{x}\vec{z}'$ with $\{\vec{z}'\} \cap \mathcal{X}^\square = \emptyset$.

– **simple** :

(B1) The symbols occurring in the arguments of $WElim_I$ or $SElim_I^Q$ are constant.

(B2) At most one rule can be applied at the top of a term of the form $WElim_I(Q, \vec{a}, c, \vec{f})$ or $SElim_I^Q(\vec{a}, c, \vec{f})$.

(A4) We have $\mathcal{F}_1 = \emptyset$. Therefore, we just have to check the conditions (a) and (b) :

(a) $(\mathcal{F}, \mathcal{R})$ is recursive : Let us see the case of $WElim_I(Q, \vec{x}, Constr_i^I(\vec{z}), \vec{f}) \rightarrow \Delta'_W[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$. We will denote by l and r the left hand-side and the right hand-side of this rule. The case of $SElim_I^Q(\vec{x}, Constr_i^I(\vec{z}), \vec{f}) \rightarrow \Delta'_S[I, X, C_i, f_i, Q, \vec{f}, \vec{z}]$ is similar.

For the precedence $\geq_\mathcal{F}$, we take $WElim_I >_\mathcal{F} WElim_J$, $WElim_I >_\mathcal{F} SElim_J^Q$, $SElim_I >_\mathcal{F} WElim_J$ and $SElim_I >_\mathcal{F} SElim_J^Q$ if J is a strict subterm of I, and all the defined symbols greater than the constant symbols.

For the status, we take $stat_{WElim_I} = lex(mul(x_k))$ where k is the position of the argument of type $\langle I \rangle \vec{x}$. We do the same for $SElim_I$. Such an assignment is clearly compatible with $\geq_\mathcal{F}$.

We have to check first that the rule is well formed. We have $\Gamma = Q : \langle B \rangle, \vec{z} : \vec{B}', \vec{f} : \langle \vec{T} \rangle$ and $\rho = \{\vec{x} \mapsto \langle \vec{m} \rangle\}$. We have to prove that each $x \in \mathrm{dom}(\Gamma)$ is weakly accessible in one of the arguments of $WElim_I$ and that $x\Gamma$ is equal to $T\rho$ where T is the type of x derived from l. This is immediate for Q and $\vec{f}$. The z_j's are weakly accessible since all the positions of a constructor are accessible (see the definition of $\mathrm{Acc}(Constr_i^I)$). The type of z_j derived from l is B'_j which does not depend on $\vec{x}$. Therefore, $B'_j\rho = B'_j = z_j\Gamma$.

We now show that r belongs to the computable closure of l, that is, $\Gamma \vdash_c r : Q\langle \vec{m} \rangle c$ where $c = Constr_i^I(\vec{z})$. First of all, note that $\mathcal{R}$ and τ are compatible with $\geq_\mathcal{F}$. This is clear for $\mathcal{R}$. For τ, this is due to our restriction on $SElim_I^Q$: the inductive types of Q are subterms of I. Hence, by the Lemmas 89 and 87, we have $\Gamma \vdash_c x\Gamma : s$ for all $x \in \mathrm{dom}^s(\Gamma)$, and $\Gamma \vdash_c \tau_g : s$ for all $g \leq_\mathcal{F} WElim_I$. Hence, we easily check that $\Gamma \vdash_c r : Q\langle \vec{m} \rangle c$.

(b) $(\mathcal{F}, \mathcal{R})$ is safe : Let $\vec{T}U$ the sequence $\langle Q \rangle, \langle \vec{A} \rangle, \langle I \rangle \vec{x}, \langle \vec{T} \rangle, Q\vec{x}y$. We have to prove :

– $\forall X \in \mathrm{FV}^\square(\vec{T}U)$, $X\gamma\rho \in \mathrm{dom}^\square(\Gamma)$,

– $\forall X, X' \in \mathrm{FV}^\square(\vec{T}U)$, $X\gamma\rho = X'\gamma\rho \Rightarrow X = X'$.

We have $\mathrm{FV}^\square(\vec{T}U) = \{Q\} \cup \{\vec{x}\} \cap \mathcal{X}^\square$, $Q\gamma\rho = Q \in \mathrm{dom}^\square(\Gamma)$ and $x_i\gamma\rho = \langle m_i \rangle$. Therefore the previous properties are satisfied thanks to the safety condition on inductive types. ■

We are now left to prove that the translation reflects the strong normalization :

Lemma 104 If $\Gamma \vdash t : T$ and $t \rightarrow_{\beta\iota'} t'$ then $\langle t\rangle \rightarrow^+ \langle t'\rangle$.

Proof. By induction on $\Gamma \vdash t : T$.

(Ind) Since $I \in \mathcal{NF}$, no reduction is possible.

(Constr) Since $\Gamma \vdash I : T$, by inversion, $I \in \mathcal{NF}$ and no reduction is possible.

($\star$-Elim) We have $\langle t\rangle = \mathit{WElim}_I(\langle Q\rangle, \langle\vec{a}\rangle, \langle c\rangle, \langle\vec{f}\rangle)$. Since $\Gamma \vdash I : T$, by inversion, $I \in \mathcal{NF}$ and no reduction is possible in I. If $Q \rightarrow_{\beta\iota'} Q'$ then, since $\vdash Q : (\vec{x} : \vec{A})I\vec{x} \rightarrow \star$, by induction hypothesis, $\langle Q\rangle \rightarrow^+ \langle Q'\rangle$ and $\langle t\rangle \rightarrow^+ \langle t'\rangle$. If $\vec{a} \rightarrow_{\beta\iota'} \vec{a}'$ then, since $\Gamma \vdash \vec{a} : \vec{A}\gamma$, by induction hypothesis, $\langle\vec{a}\rangle \rightarrow^+ \langle\vec{a}'\rangle$ and $\langle t\rangle \rightarrow^+ \langle t'\rangle$. Finally, if $c \rightarrow_{\beta\iota'} c'$ then, since $\Gamma \vdash c : I\vec{a}$, by induction hypothesis, $\langle c\rangle \rightarrow^+ \langle c'\rangle$ and $\langle t\rangle \rightarrow^+ \langle t'\rangle$.

($\Box$-Elim) We have $\langle t\rangle = \mathit{SElim}_I^Q(\langle\vec{a}\rangle, \langle c\rangle, \langle\vec{f}\rangle)$. Since $\Gamma \vdash I : T$, by inversion, $I \in \mathcal{NF}$ and no reduction is possible in I. Since $Q \in \mathcal{NF}$, no reduction is possible in Q. If $\vec{a} \rightarrow_{\beta\iota'} \vec{a}'$ then, since $\Gamma \vdash \vec{a} : \vec{A}\gamma$, by induction hypothesis, $\langle\vec{a}\rangle \rightarrow^+ \langle\vec{a}'\rangle$ and $\langle t\rangle \rightarrow^+ \langle t'\rangle$. Finally, if $c \rightarrow_{\beta\iota'} c'$ then, since $\Gamma \vdash c : I\vec{a}$, by induction hypothesis, $\langle c\rangle \rightarrow^+ \langle c'\rangle$ and $\langle t\rangle \rightarrow^+ \langle t'\rangle$.

The other cases can be treated without difficulties. ■

7.2 CIC + Rewriting

We have just seen that most of the Calculus of Inductive Constructions is formalizable as a CAC. We are going to see that we can add to this CAC rewriting rules that are not formalizable in CIC. Take the symbols $\mathit{nat} : \star$, $0 : \mathit{nat}$, $s : \mathit{nat} \rightarrow \mathit{nat}$, $+, \times : \mathit{nat} \rightarrow \mathit{nat} \rightarrow \mathit{nat}$, $\mathit{list} : \star \rightarrow \star$, $\mathit{nil} : (A:\star)\mathit{list}(A)$, $\mathit{cons} : (A:\star)A \rightarrow \mathit{list}(A) \rightarrow \mathit{list}(A)$, $\mathit{app} : (A:\star)\mathit{list}(A)) \rightarrow \mathit{list}(A) \rightarrow \mathit{list}(A)$, $\mathit{len} : (A:\star)\mathit{list}(A) \rightarrow \mathit{nat}$ the length of a list, $\mathit{in} : (A:\star)A \rightarrow \mathit{list}(A) \rightarrow \star$ the membership predicate, $\mathit{incl} : (A:\star)\mathit{list}(A) \rightarrow \mathit{list}(A) \rightarrow \star$ the inclusion predicate, $\mathit{sub} : (A:\star)\mathit{list}(A) \rightarrow \mathit{list}(A) \rightarrow \star$ the sublist predicate, $\mathit{eq} : (A:\star)A \rightarrow A \rightarrow \star$ the polymorphic Leibniz equality, $\top : \star$ the proposition ever true, $\bot : \star$ the proposition ever false, $\neg : \star \rightarrow \star$, $\vee, \wedge : \star \rightarrow \star \rightarrow \star$, and the following rules :

$$\begin{array}{rcl@{\qquad}rcl}
x+0 & \rightarrow & x & x\times 0 & \rightarrow & 0\\
0+x & \rightarrow & x & 0\times x & \rightarrow & 0\\
x+s(y) & \rightarrow & s(x+y) & x\times s(y) & \rightarrow & (x\times y)+x\\
s(x)+y & \rightarrow & s(x+y) & s(0)\times x & \rightarrow & x\\
(x+y)+z & \rightarrow & x+(y+z) & x\times s(0) & \rightarrow & x
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\neg\top & \rightarrow & \bot & P\vee\top & \rightarrow & \top & P\wedge\top & \rightarrow & P\\
\neg\bot & \rightarrow & \top & P\vee\bot & \rightarrow & P & P\wedge\bot & \rightarrow & \bot
\end{array}$$

$$\begin{array}{rcl}
eq(A,0,0) & \rightarrow & \top \\
eq(A,0,s(x)) & \rightarrow & \bot \\
eq(A,s(x),0) & \rightarrow & \bot \\
eq(A,s(x),s(y)) & \rightarrow & eq(nat,x,y) \\[1ex]
app(A,nil(A'),\ell) & \rightarrow & \ell \\
app(A,cons(A',x,\ell),\ell') & \rightarrow & cons(A,x,app(A,\ell,\ell')) \\
app(A,app(A',\ell,\ell'),\ell'') & \rightarrow & app(A,\ell,app(A,\ell',\ell'')) \\[1ex]
len(A,nil(A')) & \rightarrow & 0 \\
len(A,cons(A',x,\ell)) & \rightarrow & s(len(A,\ell)) \\
len(A,app(A',\ell,\ell')) & \rightarrow & len(A,\ell)+len(A,\ell') \\[1ex]
in(A,x,nil(A')) & \rightarrow & \bot \\
in(A,x,cons(A',y,l)) & \rightarrow & eq(A,x,y) \vee in(A,x,l) \\
sub(A,nil(A'),l) & \rightarrow & \top \\
sub(A,cons(A',x,l),nil(A'')) & \rightarrow & \bot \\
sub(A,cons(A',x,l),cons(A'',x',l')) & \rightarrow & (eq(A,x,x') \wedge sub(A,l,l')) \\
 & & \vee sub(A,cons(A,x,l),l') \\
incl(A,nil(A'),l) & \rightarrow & \top \\
incl(A,cons(A',x,l),l') & \rightarrow & in(A,x,l') \wedge incl(A,l,l') \\[1ex]
eq(L,nil(A),nil(A')) & \rightarrow & \top \\
eq(L,nil(A),cons(A',x,l)) & \rightarrow & \bot \\
eq(L,cons(A',x,l),nil(A)) & \rightarrow & \bot \\
eq(L,cons(A,x,l),cons(A',x',l')) & \rightarrow & eq(A,x,x') \wedge eq(list(A),l,l')
\end{array}$$

This rewriting system is recursive, simple, small, safe and confluent (this can be automatically proved by CiME [33]). Since the rules are left-linear, the combination with $\rightarrow_\beta$ is also confluent. Therefore, the conditions of strong normalization are satisfied.

In particular, we will remark the last rule where $\Gamma = A:\star, x:A, x':A, \ell:list(A), \ell':list(A)$ and $\rho = \{A' \mapsto A, L \mapsto list(A)\}$. It is well formed : for example, $cons(A',x',\ell') : L \rhd_1^\rho x' : A'$. And it satisfies the General Schema : $\{cons(A,x,\ell) : L, cons(A',x',\ell') : L\} \; (\rhd_1^\rho)_{\mathrm{mul}} \; \{x : A, x' : A'\}, \{\ell : list(A), \ell' : list(A)\}$.

However, the system lacks several important rules to get a complete decision procedure for classical propositional tautologies (Figure 1.3) or other simplification rules on the equality (Figure 1.4). To accept these rules, we must deal with rewriting modulo associativity and commutativity and get rid of the simplicity conditions.

7.3 Natural Deduction Modulo (NDM)

Natural Deduction Modulo (NDM) for first-order logic [50] can be presented as an extension of Natural Deduction with the following additional inference rule :

$$\frac{\Gamma \vdash P}{\Gamma \vdash Q} \quad \text{if } P \equiv Q$$

where $\equiv$ is an equivalence relation on propositions stable by substitution and context. This is a very powerful extension of first order logic since higher-order logic and skolemized set theory can both be described as a theories modulo (by using explicit substitutions [1]).

In [51], G. Dowek and B. Werner study the strong normalization of cut elimination in the case where $\equiv$ is generated from a first-order confluent and weakly normalizing rewrite system. In particular, they prove the strong normalization in two general cases : when the system is positive and when it has no quantifier. In [52], they give an example of a confluent and weakly normalizing system for which cut elimination is not normalizing. The problem comes from the fact that the elimination rule for $\forall$ introduces a substitution :

$$\frac{\Gamma \vdash \forall x.P(x)}{\Gamma \vdash P(t)}$$

Hence, when a predicate symbol is defined by a rule whose right hand-side contains quantifiers, its combination with β-reduction may not be normalizing. A normalization criterion for higher-order rewriting like the one we give in this work is therefore necessary.

Now, since NDM is a CAC (logical connectors can be defined as constant predicate symbols), we can compare our conditions with the ones given in [51].

(A1) In [51], only $\rightarrow_{\mathcal{R}}$ is required to be confluent. We do not know whether this always implies the confluence of $\rightarrow_{\mathcal{R}} \cup \rightarrow_{\beta}$. This is true if $\mathcal{R}$ is left-linear since then we have a union of left-linear and confluent CRS's with no critical pair between each other (general result due to V. van Oostrom [117] and proved in the particular case of $\rightarrow_{\beta}$ by F. Müller [92]). But we are not aware of work proving that, in presence of dependent types and rewriting at the type level, $\rightarrow_{\mathcal{R}} \cup \rightarrow_{\beta}$ is confluent even though $\mathcal{R}$ is not left-linear (V. Breazu-Tannen and J. Gallier have shown in [28] the preservation of confluence for the polymorphic λ-calculus with first-order rewriting at the object level).

(A2) The NDM types are primitive and form an admissible inductive structure when we take them as being all equivalent in $=_{\mathcal{C}}$.

(A3) In [51], the strong normalization of cut elimination is proved in two general cases : when the rules of $(\mathcal{DF}^{\Box}, \mathcal{R}_{\mathcal{DF}^{\Box}})$ have no quantifier and when they are positive. The systems without quantifiers are primitive. Therefore, in this case, (A3) is satisfied. On the other hand, in the positive case, we also require the arguments of the left hand-sides to be constant symbols and that at most one rule can be applied at the top of a term (NDM systems are small). But we also provide a new case : $(\mathcal{DF}^{\Box}, \mathcal{R}_{\mathcal{DF}^{\Box}})$ can be recursive, small and simple.

(A4) Rules without quantifiers are of first-order and rules with quantifiers are of higher-order. In [51], these two kinds of rules are treated in the same way. But the counter-example given in [52] shows that they should not be. In our conditions, we require symbols defined by rules with quantifiers to satisfy the General Schema.

Theorem 105 A NDM proof system satisfying the conditions (A1), (A3) and (A4) is strongly normalizing.

Chapter 8

Correctness of the conditions

Our proof of strong normalization is based on the extension to the Calculus of Constructions by T. Coquand and J. Gallier [36] of Tait and Girard's method of reductibility candidates [64]. The idea is to interpret each type T by a set $[\![T]\!]$ of strongly normalizable terms and to prove that every term of type T belongs to $[\![T]\!]$. The reader not familiar with these notions is invited to read the Chapter 3 of the Ph.D. thesis of B. Werner [119] for an introduction to candidates, and the paper of J. Gallier for a more detailed presentation [56]. An important difference between the candidates of T. Coquand and J. Gallier and the candidates of B. Werner for the Calculus of Inductive Constructions is that the former are made of well-typed terms while the later are made of pure (untyped) λ-terms.

8.1 Terms to be interpreted

In order to have the environment in which a term is typable, we use *closures*, that is, environment-term pairs.

Definition 106 (Closure) A *closure* is a pair $\Gamma \vdash t$ made of an environment $\Gamma \in \mathcal{E}$ and a term $t \in \mathcal{T}$. A closure $\Gamma \vdash t$ is *typable* if there exists a term $T \in \mathcal{T}$ such that $\Gamma \vdash t : T$. We will denote by $\overline{\mathbb{T}}$ the set of typable closures.

The set of closures *of type* $\Gamma \vdash T$ is $\overline{\mathbb{T}}_{\Gamma\vdash T} = \{\Gamma' \vdash t \in \overline{\mathbb{T}} \mid \Gamma' \supseteq \Gamma \text{ and } \Gamma' \vdash t : T\}$. The set of closures of type $\Gamma \vdash T$ whose terms are strongly normalizable will be denoted by $\overline{\mathbb{SN}}_{\Gamma\vdash T}$. The *restriction* of a set $S \subseteq \overline{\mathbb{T}}_{\Gamma\vdash T}$ to an environment $\Gamma' \supseteq \Gamma$ is $S|_{\Gamma'} = S \cap \overline{\mathbb{T}}_{\Gamma'\vdash T} = \{\Gamma'' \vdash t \in S \mid \Gamma'' \supseteq \Gamma'\}$.

One can easily check the following basic properties :

Lemma 107

(a) If $\Gamma' \vdash t \in \overline{\mathbb{T}}_{\Gamma\vdash T}$ and $\Gamma' \subseteq \Gamma'' \in \mathbb{E}$ then $\Gamma'' \vdash t \in \overline{\mathbb{T}}_{\Gamma\vdash T}$.
(b) If $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $\overline{\mathbb{T}}_{\Gamma'\vdash T} \subseteq \overline{\mathbb{T}}_{\Gamma\vdash T}$ and $\overline{\mathbb{T}}_{\Gamma\vdash T}|_{\Gamma'} = \overline{\mathbb{T}}_{\Gamma'\vdash T}$.
(c) If $T \; \mathbb{C}_\Gamma \; T'$ then $\overline{\mathbb{T}}_{\Gamma\vdash T} = \overline{\mathbb{T}}_{\Gamma\vdash T'}$.

We have to define an interpretation for all the terms that can be the type of another term, that is, to all the terms T such that there exists Γ and t such that

$\Gamma \vdash t : T$. In this case, by correctness of types, there exists s such that $T = s$ or $\Gamma \vdash T : s$. Thus, we have to define an interpretation for the terms of the following sets :

- $\overline{\mathbb{B}} = \{\Gamma \vdash T \in \mathcal{E} \times \mathcal{T} \mid \Gamma \in \mathbb{E} \ \wedge \ T = \Box\}$,
- $\overline{\mathbb{TY}}^{\star} = \overline{\mathbb{P}}^{0} = \{\Gamma \vdash T \in \mathcal{E} \times \mathcal{T} \mid \Gamma \vdash T : \star\}$,
- $\overline{\mathbb{TY}}^{\Box} = \overline{\mathbb{K}} = \{\Gamma \vdash K \in \mathcal{E} \times \mathcal{T} \mid \Gamma \vdash K : \Box\}$.

A term T such that $\Gamma \vdash T \in \overline{\mathbb{P}}^{0}$ can be obtained by application of a term U to a term v. By inversion, U must have a type of the form $(x\!:\!V)W$. By correctness of types, there exists s such that $\Gamma \vdash (x\!:\!V)W : s$. As T belongs to the same class as U, $T \in \mathbb{P}^0 \subseteq \mathbb{P} = \mathbb{T}_1^{\Box}$ and $U \in \mathbb{T}_1^{s}$. By classification, we obtain $s = \Box$. Therefore, after the Maximal sort Lemma, $\Gamma \vdash U$ cannot belong to $\overline{\mathbb{P}}^{0}$. We therefore have to give an interpretation to the terms of the following sets also :

- $\overline{\mathbb{P}}^{\star} = \{\Gamma \vdash T \in \mathcal{E} \times \mathcal{T} \mid \exists x, U, K, \Gamma \vdash T : (x\!:\!U)K \ \wedge \ \Gamma \vdash U : \star \ \wedge \ \Gamma \vdash K : \Box\}$,
- $\overline{\mathbb{P}}^{\Box} = \{\Gamma \vdash T \in \mathcal{E} \times \mathcal{T} \mid \exists X, K, L, \Gamma \vdash T : (X\!:\!K)L \ \wedge \ \Gamma \vdash K : \Box \ \wedge \ \Gamma \vdash L : \Box\}$,
- $\overline{\mathbb{P}} = \overline{\mathbb{P}}^{0} \cup \overline{\mathbb{P}}^{\star} \cup \overline{\mathbb{P}}^{\Box}$,
- $\overline{\mathbb{TY}} = \overline{\mathbb{B}} \cup \overline{\mathbb{K}} \cup \overline{\mathbb{P}}$.

In order to justify our definition of $\overline{\mathbb{P}}$ and to ensure that all the terms that have to be interpreted are indeed in $\overline{\mathbb{TY}}$, it suffices to see that, after the Maximal sort Lemma, the projection of $\overline{\mathbb{P}}$ on $\mathcal{T}$, that is, the set $\{T \in \mathcal{T} \mid \exists \Gamma \in \mathcal{E}, \Gamma \vdash T \in \overline{\mathbb{P}}\}$, is equal to $\mathbb{P}$.

Lemma 108 The sets $\overline{\mathbb{P}}^{0}$, $\overline{\mathbb{P}}^{\star}$, $\overline{\mathbb{P}}^{\Box}$, $\overline{\mathbb{K}}$ and $\overline{\mathbb{B}}$ are disjoint from one another.

Proof. We have seen that $\overline{\mathbb{P}}^{0}$ is disjoint from $\overline{\mathbb{P}}^{\star}$ and $\overline{\mathbb{P}}^{\Box}$. Since $\Box$ is not typable, $\overline{\mathbb{B}}$ is disjoint from all the other sets. $\overline{\mathbb{P}}$ and $\overline{\mathbb{K}}$ are disjoint since their projections $\mathbb{P}$ and $\mathbb{K}$ are disjoint. We are therefore left to verify that $\overline{\mathbb{P}}^{\star}$ and $\overline{\mathbb{P}}^{\Box}$ are indeed disjoint. Assume that there exists $\Gamma \vdash T \in \overline{\mathbb{P}}^{\star} \cap \overline{\mathbb{P}}^{\Box}$. Then, there exists x, U, K, X, K' and L such that $\Gamma \vdash T : (x\!:\!U)K$, $\Gamma \vdash U : \star$, $\Gamma \vdash K : \Box$, $\Gamma \vdash T : (X\!:\!K')L$, $\Gamma \vdash K' : \Box$ and $\Gamma \vdash L : \Box$. By convertibility of types, $(x : U)K \ \mathbb{C}_{\Gamma}^{*} \ (X : K')L$. By product compatibility, $U \ \mathbb{C}_{\Gamma}^{*} \ K'$. By conversion correctness, $\star = \Box$, which is not possible. ■

We now introduce a measure on $\overline{\mathbb{TY}}$ which will allow us to do recursive definitions.

Definition 109 $\mu(\Gamma \vdash T) = \begin{cases} 0 & \text{if } T = \Box \text{ or } \Gamma \vdash T : \Box \\ \nu(K) & \text{if } \Gamma \vdash T : K \text{ and } \Gamma \vdash K : \Box \end{cases}$
where ν is defined on predicate types as follows :

- $\nu(\star) = 0$
- $\nu((x\!:\!U)K) = 1 + \nu(K)$
- $\nu((X\!:\!K)L) = 1 + \max(\nu(K), \nu(L))$

We must make sure that this definition does not depend on K. As all the types of T are convertible, it suffices to check that ν is invariant by conversion :

Lemma 110 If $K \ \mathbb{C}_{\Gamma}^{*} \ K'$ then $\nu(K) = \nu(K')$.

Proof. By induction on the size of K and K'. After the Maximal sort Lemma, K is of the form $(\vec{x}:\vec{T})\star$, K' is of the form $(\vec{x}:\vec{T'})\star$ and $|\vec{x}| = |\vec{x}'|$. Let $n = |\vec{x}| = |\vec{x}'|$. By product compatibility and α-equivalence, we can assume that $\vec{x}' = \vec{x}$. If $n = 0$ then $K = K'$ and $\nu(K) = \nu(K')$. Assume now that $n > 0$. Let $L = (x_2 : T_2) \dots (x_n : T_n)\star$ and $L' = (x_2 : T'_2) \dots (x_n : T'_n)\star$. By product compatibility, $T_1 \; \mathbb{C}^*_\Gamma \; T'_1$ and $L \; \mathbb{C}^*_{\Gamma, x_1 : T_1} \; L'$. By Conversion correctness, T_1 and T'_1 are typable by the same sort s. By inversion and regularity, $\Gamma, x_1 : T_1 \vdash L : \Box$ and $\Gamma, x_1 : T_1 \vdash L' : \Box$. So, by induction hypothesis, $\nu(L) = \nu(L')$ and, if $s = \Box$, $\nu(T_1) = \nu(T'_1)$. Therefore, $\nu(K) = \nu(K')$. ■

Lemma 111 If $\Gamma \vdash T \in \overline{\mathbb{TY}}$ and $\theta : \Gamma \to \Delta$ then $\Delta \vdash T\theta \in \overline{\mathbb{TY}}$ and $\mu(\Delta \vdash T\theta) = \mu(\Gamma \vdash T)$.

Proof. First of all, one can easily prove by induction on the structure of predicate types that, if K is a predicate type and θ is a substitution, then $K\theta$ is a predicate type and $\nu(K\theta) = \nu(K)$. We now show the lemma by case on T :

- $T = \Box$. $\Delta \vdash T\theta = \Delta \vdash \Box \in \overline{\mathbb{TY}}$ and $\mu(\Delta \vdash T\theta) = 0 = \mu(\Gamma \vdash T)$.
- $\Gamma \vdash T : \Box$. By substitution, $\Delta \vdash T\theta : \Box$, $\Delta \vdash T\theta \in \overline{\mathbb{TY}}$ and $\mu(\Delta \vdash T\theta) = 0 = \mu(\Gamma \vdash T)$.
- $\Gamma \vdash T : K$ and $\Gamma \vdash K : \Box$. By substitution, $\Delta \vdash T\theta : K\theta$ and $\Delta \vdash K\theta : \Box$. Thus $\Delta \vdash T\theta \in \overline{\mathbb{TY}}$. Now, $\mu(\Delta \vdash T\theta) = \nu(K\theta)$ and $\mu(\Gamma \vdash T) = \nu(K)$. But $\nu(K\theta) = \nu(K)$. ■

8.2 Reductibility candidates

We will denote by :

- $\mathcal{SN}$ the set of strongly normalizable terms,
- $\mathcal{WN}$ the set of weakly normalizable terms,
- $\mathcal{CR}$ the set of terms t such that two reduction sequences issued from t are always confluent.

Definition 112 (Neutral term) A term is *neutral* if it is neither an abstraction nor constructor headed.

Definition 113 (Reductibility candidates) For each $\Gamma \vdash T \in \overline{\mathbb{TY}}$, we are going to define by induction on $\mu(\Gamma \vdash T)$:

- the set $\mathcal{R}_{\Gamma \vdash T}$ of *reductibility candidates of type* $\Gamma \vdash T$,
- the *restriction* $R|_{\Gamma'}$ of a candidate $R \in \mathcal{R}_{\Gamma \vdash T}$ w.r.t. an environment $\Gamma' \supseteq \Gamma$,
- the relation $\leq_{\Gamma \vdash T}$ on $\mathcal{R}_{\Gamma \vdash T}$,
- the element $\top_{\Gamma \vdash T}$ of $\mathcal{R}_{\Gamma \vdash T}$,
- the function $\bigwedge_{\Gamma \vdash T}$ from the powerset of $\mathcal{R}_{\Gamma \vdash T}$ to $\mathcal{R}_{\Gamma \vdash T}$.

- $T = \Box$.
  - $\mathcal{R}_{\Gamma \vdash \Box} = \{\overline{\mathbb{SN}}_{\Gamma \vdash \Box}\}$.
  - $R|_{\Gamma'} = R \cap \overline{\mathbb{T}}_{\Gamma' \vdash \Box}$.
  - $R_1 \leq_{\Gamma \vdash \Box} R_2$ if $R_1 \subseteq R_2$.

 - $\top_{\Gamma\vdash\Box} = \overline{\mathbb{SN}}_{\Gamma\vdash\Box}$.
 - $\bigwedge_{\Gamma\vdash\Box}(\Re) = \top_{\Gamma\vdash\Box}$.
- $\Gamma \vdash T : s$.
 - $\mathcal{R}_{\Gamma\vdash T}$ is the set of all the subsets R of $\overline{\mathbb{T}}_{\Gamma\vdash T}$ such that :
 (R1) $R \subseteq \mathcal{SN}$ (strong normalization);
 (R2) if $\Gamma'\vdash t \in R$ and $t \to t'$ then $\Gamma'\vdash t' \in R$ (stability by reduction);
 (R3) if $\Gamma'\vdash t \in \overline{\mathbb{T}}_{\Gamma\vdash T}$, t is neutral and, for all t' such that $t \to t'$, $\Gamma'\vdash t' \in R$, then $\Gamma'\vdash t \in R$ (stability by expansion for neutral terms);
 (R4) if $\Gamma'\vdash t \in R$ and $\Gamma' \subseteq \Gamma'' \in \mathbb{E}$ then $\Gamma''\vdash t \in R$ (stability by weakening).
 - $R|_{\Gamma'} = R \cap \overline{\mathbb{T}}_{\Gamma'\vdash T}$.
 - $R_1 \leq_{\Gamma\vdash T} R_2$ if $R1 \subseteq R_2$.
 - $\top_{\Gamma\vdash T} = \overline{\mathbb{SN}}_{\Gamma\vdash T}$.
 - $\bigwedge_{\Gamma\vdash T}(\Re) = \bigcap \Re$ if $\Re \neq \emptyset$, $\top_{\Gamma\vdash T}$ otherwise.
- $\Gamma \vdash T : (x{:}U)K$.
 - $\mathcal{R}_{\Gamma\vdash T}$ is the set of all the functions R which, to $\Gamma'\vdash u \in \overline{\mathbb{T}}_{\Gamma\vdash U}$, associate an element of $\mathcal{R}_{\Gamma'\vdash Tu}$ and satisfy :
 (P1) if $u \to u'$ then $R(\Gamma'\vdash u) = R(\Gamma'\vdash u')$ (stability by reduction),
 (P2) if $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $R(\Gamma\vdash u)|_{\Gamma'} = R(\Gamma'\vdash u)$ (compatibility with weakening).
 - $R|_{\Gamma'} = R|_{\overline{\mathbb{T}}_{\Gamma'\vdash U}}$.
 - $R_1 \leq_{\Gamma\vdash T} R_2$ if, for all $\Gamma'\vdash u \in \overline{\mathbb{T}}_{\Gamma\vdash U}$, $R_1(\Gamma'\vdash u) \leq_{\Gamma'\vdash Tu} R_2(\Gamma'\vdash u)$.
 - $\top_{\Gamma\vdash T}(\Gamma'\vdash u) = \top_{\Gamma'\vdash Tu}$.
 - $\bigwedge_{\Gamma\vdash T}(\Re)(\Gamma'\vdash u) = \bigwedge_{\Gamma'\vdash Tu}(\{R(\Gamma'\vdash u) \mid R \in \Re\})$.
- $\Gamma \vdash T : (X{:}K)L$. Let $\Sigma_{\Gamma\vdash K}$ be the set of pairs $(\Gamma'\vdash U, S)$ such that $\Gamma'\vdash U \in \overline{\mathbb{T}}_{\Gamma\vdash K}$ and $S \in \mathcal{R}_{\Gamma'\vdash U}$.
 - $\mathcal{R}_{\Gamma\vdash T}$ is the set of all functions R which, to a pair $(\Gamma'\vdash U, S) \in \Sigma_{\Gamma\vdash K}$, associate an element of $\mathcal{R}_{\Gamma'\vdash TU}$ and satisfy :
 (P1) if $U \to U'$ then $R(\Gamma'\vdash U, S) = R(\Gamma'\vdash U', S)$ (stability by reduction),
 (P2) if $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $R(\Gamma\vdash U, S)|_{\Gamma'} = R(\Gamma'\vdash U, S|_{\Gamma'})$ (compatibility with weakening).
 - $R|_{\Gamma'} = R|_{\Sigma_{\Gamma'\vdash K}}$.
 - $R_1 \leq_{\Gamma\vdash T} R_2$ if, for all $(\Gamma'\vdash U, S) \in \Sigma_{\Gamma\vdash K}$, $R_1(\Gamma'\vdash U, S) \leq_{\Gamma'\vdash TU} R_2(\Gamma'\vdash U, S)$.
 - $\top_{\Gamma\vdash T}(\Gamma'\vdash U, S) = \top_{\Gamma'\vdash TU}$.
 - $\bigwedge_{\Gamma\vdash T}(\Re)(\Gamma'\vdash U, S) = \bigwedge_{\Gamma'\vdash TU}(\{R(\Gamma'\vdash U, S) \mid R \in \Re\})$.

The following lemma ensures that all these objects are well defined.

Lemma 114 (Candidates properties)

(a) $\mathcal{R}_{\Gamma\vdash T}$, $\leq_{\Gamma\vdash T}$, $\top_{\Gamma\vdash T}$ and $\bigwedge_{\Gamma\vdash T}$ are well defined.
(b) If $T \to T'$ then $\mathcal{R}_{\Gamma\vdash T} = \mathcal{R}_{\Gamma\vdash T'}$.
(c) If $R \in \mathcal{R}_{\Gamma\vdash T}$ and $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $R|_{\Gamma'} \in \mathcal{R}_{\Gamma'\vdash T}$.
(d) $\top_{\Gamma\vdash T} \in \mathcal{R}_{\Gamma\vdash T}$.
(e) If $T \to T'$ then $\top_{\Gamma\vdash T} = \top_{\Gamma\vdash T'}$.
(f) If $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $\top_{\Gamma\vdash T}|_{\Gamma'} = \top_{\Gamma'\vdash T}$.

(g) If $\Re \subseteq \mathcal{R}_{\Gamma\vdash T}$ then $\bigwedge_{\Gamma\vdash T}(\Re) \in \mathcal{R}_{\Gamma\vdash T}$.
(h) If $T \to T'$ then $\bigwedge_{\Gamma\vdash T} = \bigwedge_{\Gamma\vdash T'}$.
(i) If $\Gamma \subseteq \Gamma' \in \mathbb{E}$ then $\bigwedge_{\Gamma\vdash T}(\Re)|_{\Gamma'} = \bigwedge_{\Gamma'\vdash T}(\{R|_{\Gamma'} \mid R \in \Re\})$.

Proof. By induction on $\mu(\Gamma\vdash T)$.

- $T = \Box$.
  (a) Immediate.
  (b) $\Box$ is not reducible.
  (c) We necessary have $R = \overline{\mathbb{SN}}_{\Gamma\vdash\Box}$. So, $R|_{\Gamma'} = \overline{\mathbb{SN}}_{\Gamma\vdash\Box} \cap \overline{\mathbb{T}}_{\Gamma'\vdash\Box} = \overline{\mathbb{SN}}_{\Gamma'\vdash\Box} \in \mathcal{R}_{\Gamma'\vdash\Box}$.
  (d) Immediate.
  (e) $\Box$ is not reducible.
  (f) $\top_{\Gamma\vdash\Box}|_{\Gamma'} = \overline{\mathbb{SN}}_{\Gamma\vdash\Box} \cap \overline{\mathbb{T}}_{\Gamma'\vdash\Box} = \overline{\mathbb{SN}}_{\Gamma'\vdash\Box} = \top_{\Gamma'\vdash\Box}$.
  (g) $\bigwedge_{\Gamma\vdash\Box}(\Re) = \top_{\Gamma\vdash\Box}$.
  (h) $\Box$ is not reducible.
  (i) $\bigwedge_{\Gamma\vdash\Box}(\Re)|_{\Gamma'} = \top_{\Gamma\vdash\Box}|_{\Gamma'} = \top_{\Gamma'\vdash\Box} = \bigwedge_{\Gamma'\vdash\Box}(\{R|_{\Gamma'} \mid R \in \Re\}$.
- $\Gamma \vdash T : s$.
  (a) Immediate.
  (b) By subject reduction, $\overline{\mathbb{T}}_{\Gamma\vdash T} = \overline{\mathbb{T}}_{\Gamma\vdash T'}$.
  (c) By weakening, $R|_{\Gamma'} \subseteq \overline{\mathbb{T}}_{\Gamma'\vdash T}$. Now we show that $R|_{\Gamma'}$ satisfies (R1) to (R4). For (R1), (R2) and (R4), this is immediate. For (R3), let $\Gamma''\vdash t \in \overline{\mathbb{T}}_{\Gamma'\vdash T}$ such that t is neutral and, for all t' such that $t \to t'$, $\Gamma''\vdash t' \in R|_{\Gamma'}$. Since $R|_{\Gamma'} \subseteq R$, $\Gamma''\vdash t \in R$. But $\Gamma''\vdash t \in \overline{\mathbb{T}}_{\Gamma'\vdash T}$. Therefore, $\Gamma''\vdash t \in R|_{\Gamma'}$.
  (d) By definition, $\top_{\Gamma\vdash T} \subseteq \overline{\mathbb{T}}_{\Gamma\vdash T}$ and it is easy to check that $\top_{\Gamma\vdash T}$ satisfies (R1) to (R4).
  (e) By subject reduction.
  (f) Immediate.
  (g) Since each element of $\Re$ is included in $\overline{\mathbb{T}}_{\Gamma\vdash T}$ and satisfies (R1) to (R4), it is easy to check that $\bigcap \Re$ is included in $\overline{\mathbb{T}}_{\Gamma\vdash T}$ and satisfies (R1) to (R4).
  (h) Immediate.
  (i) $(\bigcap \Re)|_{\Gamma'} = (\bigcap \Re) \cap \overline{\mathbb{T}}_{\Gamma'\vdash T} = \bigcap\{R \cap \overline{\mathbb{T}}_{\Gamma'\vdash T} \mid R \in \Re\} = \bigcap\{R|_{\Gamma'} \mid R \in \Re\}$.
- $\Gamma \vdash T : (x{:}U)K$.
  (a) We have to check that $\mu(\Gamma'\vdash Tu) < \mu(\Gamma\vdash T)$ and that the definitions do not depend on the choice of a type for T.

  By weakening, $\Gamma' \vdash T : (x : U)K$ and $\Gamma' \vdash (x : U)K : \Box$. By (app), $\Gamma' \vdash Tu : K\{x \mapsto u\}$. By inversion and regularity, $\Gamma', x : U \vdash K : \Box$. By substitution, $\Gamma' \vdash K\{x \mapsto u\} : \Box$. Therefore $\Gamma' \vdash Tu \in \overline{\mathbb{TY}}$ and $\mu(\Gamma' \vdash Tu) = \nu(K\{x \mapsto u\})$. By invariance by substitution, $\nu(K\{x \mapsto u\}) = \nu(K)$. Therefore $\mu(\Gamma\vdash T) = \nu((x{:}U)K) = 1 + \nu(K) > \mu(\Gamma\vdash Tu)$.

  We now show that the definitions do not depend on the choice of a type for T. Assume that $\Gamma \vdash T : (x'{:}U')K'$. By Type convertibility and Product compatibility, $U \ \mathbb{C}^*_\Gamma\ U'$. Therefore $\overline{\mathbb{T}}_{\Gamma\vdash U} = \overline{\mathbb{T}}_{\Gamma\vdash U'}$ and $\mathcal{R}_{\Gamma\vdash T}$, $\leq_{\Gamma\vdash T}$, $\top_{\Gamma\vdash T}$ and $\bigwedge_{\Gamma\vdash T}$ are unchanged if we replace U by U'.
  (b) By induction hypothesis, $\mathcal{R}_{\Gamma'\vdash Tu} = \mathcal{R}_{\Gamma'\vdash T'u}$.

(c) Immediate.

(d) We check that $\top_{\Gamma\vdash T}$ satisfies (P1) and (P2).

(P1) By induction hypothesis (e), $\top_{\Gamma'\vdash Tu} = \top_{\Gamma'\vdash Tu'}$.

(P2) By induction hypothesis (f), $\top_{\Gamma'\vdash Tu}|_{\Gamma''} = \top_{\Gamma''\vdash Tu}$.

(e) By subject reduction, $\top_{\Gamma\vdash T}$ and $\top_{\Gamma\vdash T'}$ have the same domain. And they are equal since, by induction hypothesis, $\top_{\Gamma'\vdash Tu}$ satisfies (e).

(f) $\top_{\Gamma\vdash T}|_{\Gamma'}$ and $\top_{\Gamma'\vdash T}$ have the same domain and are equal.

(g) Let $\Re' = \{R(\Gamma'\vdash u) \mid R \in \Re\}$. By definition, if $R \in \mathcal{R}_{\Gamma\vdash T}$ and $\Gamma'\vdash u \in \overline{\mathbb{T}}_{\Gamma\vdash U}$ then $R(\Gamma' \vdash u) \in \mathcal{R}_{\Gamma'\vdash Tu}$. By induction hypothesis, $\bigwedge_{\Gamma'\vdash Tu}$ satisfies (g). Therefore, $\bigwedge_{\Gamma'\vdash Tu}(\Re') \in \mathcal{R}_{\Gamma'\vdash Tu}$. We now check that $\bigwedge_{\Gamma\vdash T}$ satisfies (P1) and (P2).

(P1) Let $\Re'' = \{R(\Gamma'\vdash u') \mid R \in \Re\}$. Since each $R \in \Re$ satisfies (P1), $R(\Gamma'\vdash u') = R(\Gamma'\vdash u)$. By induction hypothesis, $\bigwedge_{\Gamma'\vdash Tu}$ satisfies (h). Therefore $\bigwedge_{\Gamma'\vdash Tu}(\Re') = \bigwedge_{\Gamma'\vdash Tu'}(\Re'')$.

(P2) Let $\Re_1 = \{R(\Gamma\vdash u) \mid R \in \Re\}$ and $\Re_2 = \{R(\Gamma\vdash u)|_{\Gamma'} \mid R \in \Re\}$. Since each $R \in \Re$ satisfies (P2), $R(\Gamma\vdash u)|_{\Gamma'} = R(\Gamma'\vdash u)$. By induction hypothesis, $\bigwedge_{\Gamma\vdash Tu}$ satisfies (i). Therefore $\bigwedge_{\Gamma\vdash Tu}(\Re_1)|_{\Gamma'} = \bigwedge_{\Gamma'\vdash Tu}(\Re_2)$.

(h) After (a), $\mathcal{R}_{\Gamma\vdash T} = \mathcal{R}_{\Gamma\vdash T'}$. Therefore, $\bigwedge_{\Gamma\vdash T}$ and $\bigwedge_{\Gamma\vdash T'}$ have the same domain. Let $\Re \subseteq \mathcal{R}_{\Gamma\vdash T}$. Then, $\bigwedge_{\Gamma\vdash T}(\Re)$ and $\bigwedge_{\Gamma\vdash T'}(\Re)$ have the same domain and are equal since, by induction hypothesis, $\bigwedge_{\Gamma'\vdash Tu}$ satisfies (h).

(i) $\bigwedge_{\Gamma\vdash T}(\Re)|_{\Gamma'}$ and $\bigwedge_{\Gamma'\vdash T}(\{R|_{\Gamma'} \mid R \in \Re\})$ have the same domain and are equal since, if $\Gamma''\vdash u \in \overline{\mathbb{T}}_{\Gamma'\vdash U}$ then $R(\Gamma''\vdash u) = R|_{\Gamma'}(\Gamma''\vdash u)$.

- $\Gamma \vdash T : (X\!:\!K)L$. The proof is similar to the previous case. ∎

Lemma 115 Let $\overline{\mathbb{X}}_{\Gamma\vdash T} = \{\Gamma'\vdash t \in \overline{\mathbb{T}}_{\Gamma\vdash T} \mid t = x\vec{t},\ x \in \mathcal{X},\ \vec{t} \in \mathcal{SN}\}$. If $\Gamma \vdash T : s$ then $\overline{\mathbb{X}}_{\Gamma\vdash T} \neq \emptyset$ and, for all $R \in \mathcal{R}_{\Gamma\vdash T}$, $\overline{\mathbb{X}}_{\Gamma\vdash T} \subseteq R$.

Proof. First of all, $\overline{\mathbb{X}}_{\Gamma\vdash T} \subseteq \overline{\mathbb{T}}_{\Gamma\vdash T}$. Since $\mathcal{X}^s$ is infinite and $\mathrm{dom}(\Gamma)$ is finite, there exists $x \in \mathcal{X}^s \setminus \mathrm{dom}(\Gamma)$. Therefore, by (var), $\Gamma, x\!:\!T \vdash x : T$. So, $\Gamma, x\!:\!T\vdash x \in \overline{\mathbb{T}}_{\Gamma\vdash T}$ and $\overline{\mathbb{X}}_{\Gamma\vdash T} \neq \emptyset$. Now, let $R \in \mathcal{R}_{\Gamma\vdash T}$, $\Gamma' \in \mathcal{E}$, $x \in \mathcal{X}$ and $\vec{t} \in \mathcal{SN}$ such that $\Gamma'\vdash x\vec{t} \in \overline{\mathbb{T}}_{\Gamma\vdash T}$. We show that $\Gamma'\vdash x\vec{t} \in R$ by induction on $\vec{t}$ with $\rightarrow_{\mathrm{lex}}$ as well-founded ordering. Since $x\vec{t}$ is neutral, by **(R3)**, it suffices to show that every immediate reduct of $x\vec{t}$ belongs to R. But this is the induction hypothesis. ∎

Lemma 116 (Completeness of the candidates lattice) For all $\Gamma \vdash T \in \overline{\mathbb{TY}}$, $(\mathcal{R}_{\Gamma\vdash T}, \leq_{\Gamma\vdash T})$ is a complete lattice. The lower bound of a part of $\mathcal{R}_{\Gamma\vdash T}$ is given by $\bigwedge_{\Gamma\vdash T}$.

Proof. It suffices to prove that $(\mathcal{R}_{\Gamma\vdash T}, \leq_{\Gamma\vdash T})$ is a complete inf-semi-lattice and that $\top_{\Gamma\vdash T}$ is its greatest element. One can easily check by induction on $\mu(\Gamma\vdash T)$ that $\leq_{\Gamma\vdash T}$ is an ordering (*i.e.* is reflexive, transitive and anti-symmetric), $\top_{\Gamma\vdash T}$ is the greatest element of $\mathcal{R}_{\Gamma\vdash T}$ and the lower bound of a part of $\mathcal{R}_{\Gamma\vdash T}$ is given by $\bigwedge_{\Gamma\vdash T}$. ∎

8.3 Interpretation schema

We define the interpretation of a type $\Gamma \vdash T$ w.r.t. a substitution $\theta : \Gamma \to \Delta$ by induction on the structure of T. Hence, we have to give an interpretation to the predicate variables and the predicate symbols that occur in T. That is why we first define an *interpretation schema* $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi}$ using a *candidate assignment* ξ for the predicate variables and an interpretation I for the predicate symbols. In the Section 8.4, we define the interpretation of constant predicate symbols and, in Section 8.6, we define the interpretation of defined predicate symbols.

Definition 117 (Candidate assignment) A *candidate assignment* is a function ξ from $\mathcal{X}^\Box$ to $\bigcup\{\,\mathcal{R}_{\Delta\vdash T} \mid \Delta\vdash T \in \overline{\mathbb{TY}}\}$. Let $\theta : \Gamma \to \Delta$ be a substitution. We say that ξ is *compatible with* (θ, Γ, Δ) if, for all $X \in \mathrm{dom}^\Box(\Gamma)$, $X\xi \in \mathcal{R}_{\Delta\vdash X\theta}$. The *restriction* of ξ to an environment Γ' is the assignment $\xi|_{\Gamma'}$ defined by $X(\xi|_{\Gamma'}) = (X\xi)|_{\Gamma'}$.

To any substitution $\theta : \Gamma \to \Delta$, we associate its *canonical candidate assignment* ξ_θ defined by $X\xi_\theta = \top_{\Delta\vdash x\theta}$. After Lemma 114 (d), ξ_θ is compatible with (θ, Γ, Δ).

Lemma 118 Let $\theta : \Gamma \to \Delta$ be a substitution and ξ be a candidate assignment compatible with (θ, Γ, Δ).
(a) If $\theta \to \theta'$ then $\theta' : \Gamma \to \Delta$ and ξ is compatible with $(\theta', \Gamma, \Delta)$.
(b) If $\Delta \subseteq \Delta' \in \mathbb{E}$ then $\theta : \Gamma \to \Delta'$ and $\xi|_{\Delta'}$ is compatible with $(\theta, \Gamma, \Delta')$.

Proof.
(a) After Lemma 35, we know that $\theta' : \Gamma \to \Delta$. Let $X \in \mathrm{dom}^\Box(\Gamma)$. Since ξ is compatible with (θ, Γ, Δ), $X\xi \in \mathcal{R}_{\Delta\vdash X\theta}$. After Lemma 114 (b), $\mathcal{R}_{\Delta\vdash X\theta'} = \mathcal{R}_{\Delta\vdash X\theta}$. Therefore $X\xi \in \mathcal{R}_{\Delta\vdash X\theta'}$.
(b) By weakening, $\theta : \Gamma \to \Delta'$. Let $X \in \mathrm{dom}^\Box(\Gamma)$. By definition, $X(\xi|_{\Delta'}) = (X\xi)|_{\Delta'}$. Since ξ is compatible with (θ, Γ, Δ), $X\xi \in \mathcal{R}_{\Delta\vdash X\theta}$. Therefore, after Lemma 114 (c), $(X\xi)|_{\Delta'} \in \mathcal{R}_{\Delta'\vdash X\theta}$. ∎

Let F be a predicate symbol of type $(\vec{x} : \vec{T})\star$. In the following, we will assume that F, which is not a term if its arity is not null, represents its η-long form $[\vec{x} : \vec{T}]F(\vec{x})$.

Definition 119 (Interpretation of a predicate symbol) An *interpretation* for a predicate symbol F is a function I which, to an environment Δ, associates an element of $\mathcal{R}_{\Delta\vdash F}$ such that :

(P3) if $\Delta \subseteq \Delta' \in \mathbb{E}$ then $I_\Delta|_{\Delta'} = I_{\Delta'}$ (compatibility with weakening).

An *interpretation* for a set $\mathcal{G}$ of predicate symbols is a function which, to a symbol $G \in \mathcal{G}$, associates an interpretation for G.

Definition 120 (Interpretation schema) The *interpretation* of $\Gamma \vdash T \in \overline{\mathbb{TY}}$ w.r.t. an environment $\Delta \in \mathbb{E}$, a substitution $\theta : \Gamma \to \Delta$, a candidate assignment ξ compatible with (θ, Γ, Δ) and an interpretation I_F pour each $F \in \mathcal{F}^\Box$, is an element of $\mathcal{R}_{\Delta\vdash T\theta}$ defined by induction on T :

- $[\![\Gamma \vdash s]\!]^I_{\Delta,\theta,\xi} = \overline{\mathbb{SN}}_{\Delta\vdash s}$,

- $[\![\Gamma\vdash F(\vec{t})]\!]^I_{\Delta,\theta,\xi} = I_{\Delta\vdash F}(\vec{a})$ or, if $\tau_F = (\vec{x}:\vec{T})U$:
  - $a_i = \Delta\vdash t_i\theta$ if $x_i \in \mathcal{X}^\star$,
  - $a_i = (\Delta\vdash t_i\theta, [\![\Gamma\vdash t_i]\!]^I_{\Delta,\theta,\xi})$ if $x_i \in \mathcal{X}^\square$,
- $[\![\Gamma\vdash X]\!]^I_{\Delta,\theta,\xi} = X\xi$,
- $[\![\Gamma\vdash (x:U)V]\!]^I_{\Delta,\theta,\xi} = \{\Delta'\vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta} \mid \forall\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}},\ \Delta''\vdash tu \in [\![\Gamma, x:U\vdash V]\!]^I_{\Delta'',\theta\cup\{x\mapsto u\},\xi|_{\Delta''}}\}$,
- $[\![\Gamma\vdash (X:K)V]\!]^I_{\Delta,\theta,\xi} = \{\Delta'\vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(X:K\theta)V\theta} \mid \forall\Delta''\vdash U \in [\![\Gamma\vdash K]\!]^I_{\Delta',\theta,\xi|_{\Delta'}},\ \Delta''\vdash tU \in \bigcap\{\,[\![\Gamma, X:K\vdash V]\!]^I_{\Delta'',\theta\cup\{X\mapsto U\},\xi|_{\Delta''}\cup\{X\mapsto S\}} \mid S\in\mathcal{R}_{\Delta''\vdash U}\}\}$,
- $[\![\Gamma\vdash [x:U]V]\!]^I_{\Delta,\theta,\xi}(\Delta'\vdash u) = [\![\Gamma, x:U\vdash V]\!]^I_{\Delta',\theta\cup\{x\mapsto u\},\xi|_{\Delta'}}$,
- $[\![\Gamma\vdash [X:K]V]\!]^I_{\Delta,\theta,\xi}(\Delta'\vdash U, S) = [\![\Gamma, X:K\vdash V]\!]^I_{\Delta',\theta\cup\{X\mapsto U\},\xi|_{\Delta'}\cup\{X\mapsto S\}}$,
- $[\![\Gamma\vdash Vu]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Delta\vdash u\theta)$,
- $[\![\Gamma\vdash VU]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Delta\vdash U\theta, [\![\Gamma\vdash U]\!]^I_{\Delta,\theta,\xi})$.

In the case where $\Gamma\vdash T : s$, the elements of $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ are called *computable*. Finally, we will say that (θ,Γ,Δ) is *valid* w.r.t. ξ if, for all $x\in\mathrm{dom}(\Gamma), \Delta\vdash x\theta \in [\![\Gamma\vdash x\Gamma]\!]^I_{\Delta,\theta,\xi}$.

After Lemma 115, the identity substitution is valid w.r.t. any candidate assignment compatible with it.

Lemma 121 (Correctness of the interpretation schema)

(a) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ is well defined.

(b) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \in \mathcal{R}_{\Delta\vdash T\theta}$.

(c) If $\theta\to\theta'$ then $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi}$.

(d) If $\Delta\subseteq\Delta'\in\mathbb{E}$ then $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'} = [\![\Gamma\vdash T]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$.

Proof. Note first of all that, for (c), $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi}$ exists since, after Lemma 118 (a), $\theta' : \Gamma\to\Delta$ and ξ is compatible with (θ',Γ,Δ). For (d), $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'}$ exists since, after (b), $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \in \mathcal{R}_{\Delta\vdash T\theta}$, and $[\![\Gamma\vdash T]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ exists since, after Lemma 118 (b), $\theta : \Gamma\to\Delta'$ and $\xi|_{\Delta'}$ is compatible with (θ,Γ,Δ').

- $T = s$.
  (a) Immediate.
  (b) After Lemma 114 (d).
  (c) Since $[\![\Gamma\vdash s]\!]^I_{\Delta,\theta,\xi}$ does not depend on θ.
  (d) $[\![\Gamma\vdash s]\!]^I_{\Delta,\theta,\xi}|_{\Delta'} = \overline{\mathbb{SN}}_{\Delta\vdash s} \cap \overline{\mathbb{T}}_{\Delta'\vdash s} = \overline{\mathbb{SN}}_{\Delta'\vdash s} = [\![\Gamma\vdash s]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$.
- $T = F(\vec{t})$.
  (a) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = I_{\Delta\vdash F}(\vec{a})$ where $a_i = \Delta\vdash t_i\theta$ if $x_i\in\mathcal{X}^\star$ and $a_i = (\Delta\vdash t_i\theta, [\![\Gamma\vdash t_i]\!]^I_{\Delta,\theta,\xi})$ if $x_i\in\mathcal{X}^\square$. Par induction hypothesis (a) and (b), $[\![\Gamma\vdash t_i]\!]^I_{\Delta,\theta,\xi}$ is well defined and belongs to $\mathcal{R}_{\Delta\vdash t_i\theta}$. Therefore $\vec{a}$ is in the domain of $I_{\Delta\vdash F}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ is well defined.
  (b) By definition of $I_{\Delta\vdash F}$.
  (c) By **(P1)** .

(d) By **(P2)** .

- $T = X$.

(a) Immediate.

(b) Since ξ is compatible with (θ, Γ, Δ).

(c) Since $[\![\Gamma \vdash X]\!]^I_{\Delta,\theta,\xi}$ does not depend on θ.

(d) By definition of $\xi|_{\Delta'}$.

- $T = (x:U)V$. Let $\Gamma' = \Gamma, x:U$.

(a) Assume that $\Delta \subseteq \Delta' \in \mathbb{E}$. After Lemma 118 (b), $\theta : \Gamma \to \Delta'$ and $\xi|_{\Delta'}$ is compatible with $(\theta, \Gamma, \Delta')$. So, by induction hypothesis (a) and (b), $[\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ is well defined and belongs to $\mathcal{R}_{\Delta'\vdash U\theta}$.

By Type correctness, there exists s such that $\Gamma \vdash T : s$. By inversion, $\Gamma' \vdash V : s$. After the Environment Lemma, $\Gamma \vdash U : \star$. Therefore, by substitution, $\Delta' \vdash U\theta : \star$ and $[\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}} \subseteq \overline{\mathbb{T}}_{\Delta'\vdash U\theta}$.

Now, let $\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $\sigma = \theta \cup \{x \mapsto u\}$. Since $\theta : \Gamma \to \Delta$, $\Delta'' \vdash u : U\theta$ and $x \notin \mathrm{FV}(\Gamma')$, we have $\sigma : \Gamma' \to \Delta''$. After Lemma 118 (b), $\xi|_{\Delta''}$ is compatible with $(\theta, \Gamma, \Delta'')$. Since $\mathrm{dom}^\Box(\sigma) = \mathrm{dom}^\Box(\theta)$ and σ and θ are equal on this domain, $\xi|_{\Delta''}$ is compatible with $(\sigma, \Gamma', \Delta'')$. Therefore, by induction hypothesis (a) and (b), $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ is well defined and belongs to $\mathcal{R}_{\Delta''\vdash V\sigma}$.

Finally, by substitution, $\Delta'' \vdash V\sigma : s$. Therefore, $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}} \subseteq \overline{\mathbb{T}}_{\Delta''\vdash V\sigma}$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi}$ is well defined.

(b) By substitution, $\Delta \vdash T\theta : s$. Therefore, we have to prove that $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi}$ is included in $\overline{\mathbb{T}}_{\Delta\vdash T\theta}$ (immediate) and satisfies (R1) to (R4). We have seen in (a) that, if $\Delta \subseteq \Delta' \in \mathbb{E}$, $\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $\sigma = \theta \cup \{x \mapsto u\}$, then $[\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ are included in $\overline{\mathbb{T}}_{\Delta'\vdash U\theta}$ and $\overline{\mathbb{T}}_{\Delta''\vdash V\sigma}$ respectively and satisfy (R1) to (R4).

(R1) After Lemma 115, there exists $y \in \mathcal{X}^\star \setminus \mathrm{dom}(\Delta')$ such that $\Delta', y:U\theta \vdash y \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$. Let $\Delta'' = \Delta', y:U\theta$ and $\sigma = \theta \cup \{x \mapsto y\}$. Then, by definition, $\Delta'' \vdash ty \in [\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ and, since $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ satisfies **(R1)**, $ty \in \mathcal{SN}$ and $t \in \mathcal{SN}$.

(R2) Let $\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $\sigma = \theta \cup \{x \mapsto u\}$. By definition, $\Delta'' \vdash tu \in [\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ and, since $tu \to t'u$ and $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ satisfies **(R2)**, $t'u \in [\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ and $t' \in [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi}$.

(R3) This is in this case that we use the notion of arity which establishes a syntactic distinction between the application of the λ-calculus and the application of a symbol (see Remark 10). Let $\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $\sigma = \theta \cup \{x \mapsto u\}$. By definition, $\Delta'' \vdash tu \in \overline{\mathbb{T}}_{\Delta''\vdash V\sigma}$ and tu is neutral. Since $[\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ satisfies **(R1)**, $u \in \mathcal{SN}$.

We prove that any reduct v' of tu belongs to $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$, by induction on u with $\to$ as well-founded ordering. Since t is not an abstraction, v' is either of the form $t'u$ with t' an immediate reduct of t, or of the form tu' with u' an immediate reduct of u. In the first case, $\Delta'' \vdash t'u \in \overline{\mathbb{T}}_{\Delta''\vdash V\sigma}$

since, by hypothesis, $\Delta''\vdash t' \in [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ and $\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$. In the second case, $\Delta''\vdash tu' \in \overline{\overline{\mathbb{T}}}_{\Delta''\vdash V\sigma}$ by induction hypothesis.

Therefore, since $[\![\Gamma'\vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ satisfies **(R3)**, $\Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ and $\Delta''\vdash t \in [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$.

(R4) Assume that $\Delta' \subseteq \Delta'' \in \mathbb{E}$, $\Delta'''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta'',\theta,\xi|_{\Delta''}}$ and $\sigma = \theta\cup\{x\mapsto u\}$. By induction hypothesis (d), $[\![\Gamma\vdash U]\!]^I_{\Delta'',\theta,\xi|_{\Delta''}} = [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}|_{\Delta''} = [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}} \cap \overline{\overline{\mathbb{T}}}_{\Delta''\vdash U\theta}$. So, $\Delta'''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$, $\Delta'''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta''',\sigma,\xi|_{\Delta'''}}$ and $\Delta''\vdash t \in [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$.

(c) We prove that $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \subseteq [\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi}$. The other way around is similar. Since $\Delta\vdash T\sigma : s$, by conversion, $\overline{\overline{\mathbb{T}}}_{\Delta\vdash T\theta'} = \overline{\overline{\mathbb{T}}}_{\Delta\vdash T\theta}$ and $\Delta'\vdash t \in \overline{\overline{\mathbb{T}}}_{\Delta\vdash T\theta'}$. Now, let $\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta',\xi|_{\Delta'}}$, $\sigma = \theta\cup\{x\mapsto u\}$ and $\sigma' = \sigma\cup\{x\mapsto u\}$. By induction hypothesis (c), $[\![\Gamma\vdash U]\!]^I_{\Delta',\theta',\xi|_{\Delta'}} = [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$. Therefore, since $[\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ satisfies **(R3)**, $\Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta,\sigma,\xi}$. And since $\sigma\to\sigma'$, by induction hypothesis (c), $\Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta,\sigma',\xi}$ and $\Delta'\vdash t \in [\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi}$.

(d) We prove that $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'} \subseteq [\![\Gamma\vdash T]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$. The other way around is similar. By definition, $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \cap \overline{\overline{\mathbb{T}}}_{\Delta'\vdash T\theta}$. Let $\Delta''\vdash t \in [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'}$, $\Delta'''\vdash u \in [\![\Gamma\vdash T]\!]^I_{\Delta'',\theta,\xi|_{\Delta''}}$ and $\sigma = \theta\cup\{x\mapsto u\}$. By definition of $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$, $\Delta'''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta''',\sigma,\xi|_{\Delta'''}}$. Therefore, since $\Delta''\vdash t \in \overline{\overline{\mathbb{T}}}_{\Delta'\vdash T\theta}$, $\Delta''\vdash t \in [\![\Gamma\vdash T]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$.

- $T = (X{:}K)V$. Similar to the previous case.
- $T = [x{:}U]V$. Let $\Gamma' = \Gamma, x{:}U$.

(a) Let $\Delta'\vdash u \in \overline{\overline{\mathbb{T}}}_{\Delta\vdash U\theta}$ and $\sigma = \theta\cup\{x\mapsto u\}$. Since $\Delta\subseteq\Delta'\in\mathbb{E}$, by Lemma 118 (b), $\theta:\Gamma\to\Delta'$ and $\xi|_{\Delta'}$ is compatible with (θ,Γ,Δ'). Since $\Delta'\vdash u : U\theta$ and $x\notin \mathrm{FV}(\Gamma)$, $\sigma:\Gamma'\to\Delta'$. Moreover, $\xi|_{\Delta'}$ is compatible with (σ,Γ',Δ') since $\mathrm{dom}^\square(\sigma) = \mathrm{dom}^\square(\theta)$. So, by induction hypothesis (a), $[\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma,\xi|_{\Delta'}}$ is well defined and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ is well defined.

(b) $\mathcal{R}_{\Delta\vdash T\theta}$ is the set of functions which, to $\Delta'\vdash u \in \overline{\overline{\mathbb{T}}}_{\Delta\vdash U\theta}$, associate an element of $\mathcal{R}_{\Delta'\vdash(T\theta\ u)}$ and satisfy (P1) and (P2). By induction hypothesis (b), $[\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma,\xi|_{\Delta'}} \in \mathcal{R}_{\Delta'\vdash V\sigma}$. Since $(T\theta\ u)\to_\beta V\sigma$, by Lemma 114 (b), $\mathcal{R}_{\Delta'\vdash V\sigma} = \mathcal{R}_{\Delta'\vdash(T\theta\ u)}$.

(P1) Assume that $u\to u'$. $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}(\Delta'\vdash u') = [\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma',\xi|_{\Delta'}}$ where $\sigma' = \theta\cup\{x\mapsto u'\}$. Since $\sigma\to\sigma'$, by induction hypothesis (c), $[\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma',\xi|_{\Delta'}} = [\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma,\xi|_{\Delta'}}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \in \mathcal{R}_{\Delta\vdash T\theta}$.

(P2) Assume that $\Delta\subseteq\Delta'\in\mathbb{E}$. $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}(\Delta\vdash u)|_{\Delta'} = [\![\Gamma'\vdash V]\!]^I_{\Delta,\sigma,\xi}|_{\Delta'}$. By induction hypothesis (d), $[\![\Gamma'\vdash V]\!]^I_{\Delta,\sigma,\xi}|_{\Delta'} = [\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma,\xi'|_{\Delta'}} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}(\Delta'\vdash u)$.

(c) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi}(\Delta'\vdash u) = [\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma',\xi|_{\Delta'}}$ or $\sigma' = \sigma\cup\{x\mapsto u\}$. $\sigma\to\sigma'$ therefore, by induction hypothesis (c), $[\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma',\xi|_{\Delta'}} = [\![\Gamma'\vdash V]\!]^I_{\Delta',\sigma,\xi|_{\Delta'}}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$.

(d) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\overline{\mathbb{T}}_{\Delta'\vdash U\theta}}$ is the function which, to $\Delta''\vdash u \in \overline{\mathbb{T}}_{\Delta'\vdash U\theta}$, associates $[\![\Gamma'\vdash V]\!]^I_{\Delta'',\sigma,\xi|_{\Delta''}}$ where $\sigma = \theta\cup\{x\mapsto u\}$. This is $[\![\Gamma\vdash T]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$.

- $T = [X:K]V$. Similar to the previous case.
- $T = Vu$.

(a) By induction hypothesis (a), $[\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}$ is well defined and belongs to $\mathcal{R}_{\Delta\vdash V\theta}$. Since $T \in \overline{\mathbb{TY}}$ and $T \neq \Box$, T is typable in Γ. By inversion, there exists U and K such that $\Gamma\vdash V : (x:U)K$ and $\Gamma\vdash u : U$. By substitution, $\Delta\vdash V\theta : (x:U\theta)K\theta$ and $\Delta\vdash u\theta : U\theta$. Therefore, $\mathcal{R}_{\Delta\vdash V\theta}$ is the set of functions which, to $\Delta'\vdash u' \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, associate an element of $\mathcal{R}_{\Delta'\vdash V\theta u'}$. So, $\Delta\vdash u\theta \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ is well defined.

(b) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \in \mathcal{R}_{\Delta\vdash(V\theta u\theta)} = \mathcal{R}_{\Delta\vdash T\theta}$.

(c) By induction hypothesis (c), $[\![\Gamma\vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}$. Since $u\theta\rightarrow^* u\sigma$ and $[\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}$ satisfies **(P1)**, $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta',\xi}$.

(d) $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}|_{\Delta'} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Delta\vdash u\theta)|_{\Delta'}$. By **(P2)**, $[\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Delta\vdash u\theta)|_{\Delta'} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Delta'\vdash u\theta) = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}|_{\Delta'}(\Delta'\vdash u\theta)$. By induction hypothesis (d), $[\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}|_{\Delta'}(\Delta'\vdash u\theta) = [\![\Gamma\vdash V]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}(\Delta'\vdash u\theta) = [\![\Gamma\vdash T]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$.

- $T = (V\ U)$. Similar to the previous case. ■

Lemma 122 Let I and I' be two interpretations equal on the predicate symbols occurring in T, and ξ and ξ' be two candidate assignments equal on the predicate variables free in T. Then, $[\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi'} = [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$.

Proof. By induction on T. ■

Lemma 123 (Candidate substitution) Let Γ_0, Γ_1 and Γ_2 be three valid environments, $\Gamma_0\vdash T\in\overline{\mathbb{TY}}$, $\theta_1:\Gamma_0\to\Gamma_1$ and $\theta_2:\Gamma_1\to\Gamma_2$ be two substitutions, and ξ_2 be a candidate assignment compatible with $(\theta_2,\Gamma_1,\Gamma_2)$. Then, the candidate assignment ξ_{12} defined by $X\xi_{12} = [\![\Gamma_1\vdash X\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2}$ is compatible with $(\theta_1\theta_2,\Gamma_0,\Gamma_2)$ and $[\![\Gamma_1\vdash T\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2} = [\![\Gamma_0\vdash T]\!]^I_{\Gamma_2,\theta_1\theta_2,\xi_{12}}$.

Proof. After Lemma 121 (b), $X\xi_{12}\in\mathcal{R}_{\Gamma_2\vdash X\theta_1\theta_2}$. Therefore ξ_{12} is compatible with $(\theta_1\theta_2,\Gamma_0,\Gamma_2)$. Let $R = [\![\Gamma_0\vdash T]\!]^I_{\Gamma_2,\theta_1\theta_2,\xi_{12}}$ and $R' = [\![\Gamma_1\vdash T\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2}$. We show that $R = R'$ by induction on T. After Lemma 121 (b), R and R' both belong to $\mathcal{R}_{\Gamma_2\vdash T\theta_1\theta_2}$.

- $T = s$. $R' = [\![\Gamma_1\vdash s]\!]^I_{\Gamma_2,\theta_2,\xi_2} = \overline{\mathbb{SN}}_{\Gamma_2\vdash s} = R$.
- $T = F(\vec{t})$. Assume that $\tau_F = (\vec{x}:\vec{T})\star$. Then, $R = I_{\Gamma_2\vdash F}(\vec{a})$ where $a_i = \Gamma_2\vdash t_i\theta_1\theta_2$ if $x_i\in\mathcal{X}^\star$, and $a_i = (\Gamma_2\vdash t_i\theta_1\theta_2, [\![\Gamma_0\vdash t_i]\!]^I_{\Gamma_2,\theta_1\theta_2,\xi_{12}})$ if $x_i\in\mathcal{X}^\Box$. Similarly, $R' = I_{\Gamma_2\vdash F}(\vec{a}')$ where $a'_i = \Gamma_2\vdash t_i\theta_1\theta_2$ if $x_i\in\mathcal{X}^\star$, and $a'_i = (\Gamma_2\vdash t_i\theta_1\theta_2, [\![\Gamma_1\vdash t_i\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2})$ if $x_i\in\mathcal{X}^\Box$. By induction hypothesis, for all $x_i\in\mathcal{X}^\Box$, $[\![\Gamma_0\vdash t_i]\!]^I_{\Gamma_2,\theta_1\theta_2,\xi_{12}} = [\![\Gamma_1\vdash t_i\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2}$. Therefore, $\vec{a} = \vec{a}'$ and $R = R'$.
- $T = X$. $R = X\xi_{12} = [\![\Gamma_1\vdash X\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2} = R'$.
- $T = (x:U)V$. Let $\Gamma'_0 = \Gamma_0, x:U$, $\Gamma'_1 = \Gamma_1, x:U\theta_1$ and $\Gamma'_2\vdash t\in R$. Since $R\in\mathcal{R}_{\Gamma_2\vdash T\theta_1\theta_2}$, $\Gamma'_2\vdash t\in\overline{\mathbb{T}}_{\Gamma_2\vdash T\theta_1\theta_2}$. Let $\Gamma''_2\vdash u\in[\![\Gamma_1\vdash U\theta_1]\!]^I_{\Gamma'_2,\theta_2,\xi_2|_{\Gamma'_2}}$. After

Lemma 121 (d), for all $X \in \mathrm{dom}^\Box(\Gamma_0)$, $X\xi_{12}|_{\Gamma'_2} = [\![\Gamma_1 \vdash X\theta_1]\!]^I_{\Gamma'_2,\theta_2,\xi_2|_{\Gamma'_2}}$. By induction hypothesis, $[\![\Gamma_1 \vdash U\theta_1]\!]^I_{\Gamma'_2,\theta_2,\xi_2|_{\Gamma'_2}} = [\![\Gamma_0 \vdash U]\!]^I_{\Gamma'_2,\theta_1\theta_2,\xi_{12}|_{\Gamma'_2}}$. By definition of $\mathcal{R}_{\Gamma_2 \vdash T\theta_1\theta_2}$, $\Gamma''_2 \vdash tu \in [\![\Gamma'_0 \vdash V]\!]^I_{\Gamma''_2,\theta_1\theta_2\cup\{x\mapsto u\},\xi_{12}|_{\Gamma''_2}}$. Moreover, $X\xi_{12}|_{\Gamma''_2} = [\![\Gamma_1 \vdash X\theta_1]\!]^I_{\Gamma''_2,\theta_2,\xi_2|_{\Gamma''_2}}$, $\theta_1 : \Gamma'_0 \to \Gamma'_1$, $\theta_2 \cup \{x \mapsto u\} : \Gamma'_1 \to \Gamma''_2$ and $\theta_1\theta_2 \cup \{x \mapsto u\} = \theta_1(\theta_2 \cup \{x \mapsto u\})$. Therefore, by induction hypothesis, $[\![\Gamma'_0 \vdash V]\!]^I_{\Gamma''_2,\theta_1\theta_2\cup\{x\mapsto u\},\xi_{12}|_{\Gamma''_2}} = [\![\Gamma'_1 \vdash V\theta_1]\!]^I_{\Gamma''_2,\theta_2\cup\{x\mapsto u\},\xi_2|_{\Gamma''_2}}$ and $\Gamma'_2 \vdash t \in R'$. So, $R \subseteq R'$. The other way around is similar.

- $T = (X:K)V$. Similar to the previous case.
- $T = [x:U]V$. Let $\Gamma'_0 = \Gamma_0, x:U$, $\Gamma'_1 = \Gamma_1, x:U\theta_1$ and $\Gamma'_2 \vdash u \in \overline{\overline{\mathbb{T}}}_{\Gamma_2 \vdash U\theta_1\theta_2}$. By definition, $R(\Gamma'_2 \vdash u) = [\![\Gamma'_0 \vdash V]\!]^I_{\Gamma'_2,\theta_1\theta_2\cup\{x\mapsto u\},\xi_{12}|_{\Gamma'_2}}$. By induction hypothesis, $[\![\Gamma'_0 \vdash V]\!]^I_{\Gamma'_2,\theta_1\theta_2\cup\{x\mapsto u\},\xi_{12}|_{\Gamma'_2}} = [\![\Gamma'_1 \vdash V\theta_1]\!]^I_{\Gamma'_2,\theta_2\cup\{x\mapsto u\},\xi_2|_{\Gamma'_2}}$. Therefore, $R(\Gamma'_2 \vdash u) = R'(\Gamma'_2 \vdash u)$ and $R = R'$.
- $T = [X:K]V$. Similar to the previous case.
- $T = Vu$. $R = [\![\Gamma_0 \vdash V]\!]^I_{\Gamma_2,\theta_1\theta_2,\xi_{12}}(\Gamma_2 \vdash u\theta_1\theta_2)$. By induction hypothesis, $[\![\Gamma_0 \vdash V]\!]^I_{\Gamma_2,\theta_1\theta_2,\xi_{12}} = [\![\Gamma_1 \vdash V\theta_1]\!]^I_{\Gamma_2,\theta_2,\xi_2}$. Therefore, $R = R'$.
- $T = VU$. Similar to the previous case. ■

8.4 Interpretation of constant predicate symbols

We define the interpretation I for constant predicate symbols by induction on $>_{\mathcal{C}}$. Let C be a constant predicate symbol and assume that we already have defined an interpretation K for all the symbols smaller than C.

Like N. P. Mendler [90] or B. Werner [119], we define this interpretation as the fixpoint of some monotone function on a complete lattice. The monotonicity is ensured the positivity conditions of an admissible inductive structure (Definition 74). The main difference with these works is that we have a more general notion of constructor since it includes any function symbol whose output type is a constant predicate symbol. This allows us to defined functions or predicates by matching not only on constant constructors but also on defined symbols.

We will denote by :

- $[C]$ the set of constant predicate symbols equivalent to C,
- $\mathcal{I}$ the set of the interpretations for $[C]$,
- $\leq$ the relation on $\mathcal{I}$ defined by $I \leq I'$ if, for all $D \in [C]$ and $\Delta \in \mathbb{E}$, $I_{\Delta\vdash D} \leq_{\Delta\vdash D} I'_{\Delta\vdash D}$.

For simplifying the notations, we will denote $[\![\Gamma \vdash T]\!]^{K\cup I}_{\Delta,\theta,\xi}$ by $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi}$.

Let $D \in [C]$. Assume that D is of arity n and type $(\vec{x} : \vec{T})\star$. Let Δ be an environment. By definition, $\mathcal{R}_{\Delta\vdash D}$ is the set of functions which, to $a_1 \in A_1, \ldots, a_n \in A_n$, associates an element of $\mathcal{R}_{\Delta_n \vdash D(\vec{t})}$ where :

– $a_i = \Delta_i \vdash t_i$ and $A_i = \overline{\mathbb{T}}_{\Delta_{i-1} \vdash T_i\theta}$ if $x_i \in \mathcal{X}^\star$,
– $a_i = (\Delta_i \vdash t_i, S_i)$ and $A_i = \Sigma_{\Delta_{i-1} \vdash T_i\theta}$ if $x_i \in \mathcal{X}^\Box$,
– $\Delta_0 = \Delta$ and $\theta = \{\vec{x} \mapsto \vec{t}\}$.

Definition 124 (Monotone interpretation) Let $I \in \mathcal{I}$, $x_i \in \mathcal{X}^\Box$, $\Delta \in \mathbb{E}$ and $\vec{a}, \vec{a}'$ two sequences of arguments for I such that $a_i = (\Delta_i \vdash t_i, S_i)$, $a'_i = (\Delta_i \vdash t_i, S'_i)$ and, for all $j \neq i$, $a_j = a'_j$. Then, I is *monotone in its i-th argument* if $S_i \leq S'_i$ implies $I_{\Delta \vdash D}(\vec{a}) \leq I_{\Delta \vdash D}(\vec{a}')$. We will denote by $\mathcal{I}^m$ the set of the interpretations that are monotone in all its inductive arguments $i \in \mathrm{Ind}(D)$.

Lemma 125 $(\mathcal{I}^m, \leq)$ is a complete lattice.

Proof. First of all, $\leq$ is an ordering since, for all $D =_{\mathcal{C}} C$ and $\Delta \in \mathbb{E}$, $\leq_{\Delta \vdash D}$ is an ordering.

We show that the function $I^\top$ defined by $I^\top_{\Delta \vdash D} = \top_{\Delta \vdash D}$ is the greatest element of $\mathcal{I}^m$. The function $I^\top_D$ is an interpretation since, after Lemma 114 (d) and (f), $I^\top_{\Delta \vdash D} \in \mathcal{R}_{\Delta \vdash D}$ and if $\Delta \subseteq \Delta' \in \mathbb{E}$ then $I^\top_{\Delta \vdash D}\Delta = \top_{\Delta \vdash D}|_{\Delta'} = \top_{\Delta' \vdash D} = I^\top_{\Delta' \vdash D}$. Moreover, $I^\top$ is the greatest element of $\mathcal{I}$ since $\top_{\Delta \vdash D}$ is the greatest element of $\mathcal{R}_{\Delta \vdash D}$.

We now show that $I^\top$ is monotone in its inductive arguments. Let $i \in \mathrm{Ind}(D)$ and $\vec{a}, \vec{a}'$ two sequences of arguments for $I^\top_{\Delta \vdash D}$ such that $a_i = (\Delta_i \vdash t_i, S_i)$, $a'_i = (\Delta_i \vdash t_i, S'_i)$, $S_i \leq S'_i$ and, for all $j \neq i$, $a_j = a'_j$. Then, $I^\top_{\Delta \vdash D}(\vec{a}) = \top_{\Delta_n \vdash D(\vec{t})} = I^\top_{\Delta \vdash D}(\vec{a}')$.

We now show that every part of $\mathcal{I}^m$ has an inf. Let $\Im \subseteq \mathcal{I}^m$ and $I^\wedge$ be the function defined by $I^\wedge_{\Delta \vdash D} = \bigwedge_{\Delta \vdash D}(\Re_{\Delta \vdash D})$ where $\Re_{\Delta \vdash D} = \{I_{\Delta \vdash D} \mid I \in \Im\}$. The function $I^\wedge$ is an interpretation since, after Lemma 114 (g) and (i), $I^\wedge_{\Delta \vdash D} \in \mathcal{R}_{\Delta \vdash D}$ and if $\Delta \subseteq \Delta' \in \mathbb{E}$ then $I^\wedge_{\Delta \vdash D}|_{\Delta'} = I^\wedge_{\Delta' \vdash D}$.

We now show that $I^\wedge$ is monotone in its inductive arguments. Let $i \in \mathrm{Ind}(D)$ and $\vec{a}, \vec{a}'$ two sequences of arguments for $I^\wedge_{\Delta \vdash D}$ satisfying the conditions of the Definition 124. Then, $I^\wedge_{\Delta \vdash D}(\vec{a}) = \bigcap\{I_{\Delta \vdash D}(\vec{a}) \mid I \in \Im\}$ and $I^\wedge_{\Delta \vdash D}(\vec{a}') = \bigcap\{I_{\Delta \vdash D}(\vec{a}') \mid I \in \Im\}$. Since each I is monotone in its inductive arguments, $I_{\Delta \vdash D}(\vec{a}) \leq I_{\Delta \vdash D}(\vec{a}')$. Therefore, $I^\wedge_{\Delta \vdash D}(\vec{a}) \leq I^\wedge_{\Delta \vdash D}(\vec{a}')$.

We are left to show that $I^\wedge$ is the inf of $\Im$. For all $I \in \Im$, $I^\wedge \leq I$ since $I^\wedge_{\Delta \vdash D}$ is the inf of $\Re_{\Delta \vdash D}$. Assume now that there exists $I' \in \mathcal{I}^m$ such that, for all $I \in \Im$, $I' \leq I$. Then, since $I^\wedge_{\Delta \vdash D}$ is the inf of $\Re_{\Delta \vdash D}$, $I' \leq I^\wedge$.

Definition 126 (Interpretation of constant predicate symbols) Let φ be the function which, to $I \in \mathcal{I}^m$, associates the interpretation φ^I such that $\varphi^I_{\Delta \vdash D}(\vec{a})$ is the set of $\Delta' \vdash u \in \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})}$ such that if u reduces to a term of the form $d(\vec{u})$ with d a constructor of type $(\vec{y} : \vec{U})D(\vec{v})$ then, for all $j \in \mathrm{Acc}(d)$, $\Delta' \vdash u_j \in [\![\vec{y} : \vec{U} \vdash U_j]\!]^I_{\Delta', \theta, \xi|_{\Delta'}}$ where $\theta = \{\vec{y} \mapsto \vec{u}\}$ and, for all $Y \in \mathrm{FV}^\Box(U_j)$, $Y\xi = S_{\iota_Y}$. We show hereafter that φ is monotone. Therefore we can take $I_{\Delta \vdash D} = \mathrm{lfp}(\varphi)_{\Delta \vdash D}$ where $\mathrm{lfp}(\varphi)$ is the least fix point of φ.

The aim of this definition is to ensure the correctness of the accessibility relations (Lemma 134) : if $d(\vec{u})$ is computable then each accessible u_j ($j \in \mathrm{Acc}(d)$) is computable. This will allow us to ensure the computability of the variables of the left hand-side of a rule if the arguments of the left hand-side are computable, and thus the computability of the right hand-sides that belong to the computable closure.

Lemma 127 φ^I is a well defined interpretation.

Proof. We first prove that φ^I is well defined. The existence of ξ is the hypothesis **(I6)**. Let $\Gamma_d = \vec{y} : \vec{U}$. We have to check that $\theta : \Gamma_d \to \Delta'$ and $\xi|_{\Delta'}$ is compatible with $(\theta, \Gamma_d, \Delta')$. By subject reduction, $\Delta' \vdash d(\vec{u}) : D(\vec{t})$. By inversion, $\Delta' \vdash d(\vec{u}) : D(\vec{v}\theta)$, $D(\vec{v}\theta)\ \mathbb{C}^*_{\Delta'}\ D(\vec{t})$ and, for all j, $\Delta' \vdash u_j : U_j\theta$. Therefore, $\theta : \Gamma_d \to \Delta'$. Let $Y \in \mathrm{FV}^\Box(U_j)$. We have $Y\xi = S_{\iota_Y} \in \mathcal{R}_{\Delta_{\iota_Y} \vdash t_{\iota_Y}}$. By Lemma 114 (c), $Y\xi|_{\Delta'} \in \mathcal{R}_{\Delta' \vdash t_{\iota_Y}}$. By **(A1)**, since D is constant, for all i, $v_i\theta \downarrow t_i$. So, by Lemma 114 (b), $\mathcal{R}_{\Delta' \vdash t_{\iota_Y}} = \mathcal{R}_{\Delta' \vdash v_{\iota_Y}\theta}$. By **(I6)**, $v_{\iota_Y} = Y$. Therefore $Y\xi \in \mathcal{R}_{\Delta' \vdash Y\theta}$.

Finally, we must make sure that the interpretations necessary for computing $[\![\vec{y} : \vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ are all well defined. The interpretation of constant predicate symbols smaller than D is K. The interpretation of constant predicate symbols equivalent to D is I. By **(I4)** and **(I5)**, constant predicate symbols greater than D or defined predicate symbols can occur only at neutral positions, but it is easy to check that terms at neutral positions are not interpreted.

We now prove that $\varphi^I_{\Delta \vdash D} \in \mathcal{R}_{\Delta \vdash D}$. To this end, we must prove that $R = \varphi^I_{\Delta \vdash D}(\vec{a})$ is included in $\overline{\mathbb{T}}_{\Delta_n \vdash D(\vec{t})}$ (immediate) and satisfies the properties (R1) to (R4) :

- **(R1)** By definition.
- **(R2)** Let $\Delta' \vdash u \in R$ and u' such that $u \to u'$. By definition, $\Delta' \vdash u \in \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})}$. Therefore, $\Delta' \vdash u' \in \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})}$. Assume that $u' \to^* d(\vec{u})$ with $\tau_d = (\vec{y} : \vec{U})D(\vec{v})$. Then, $u \to^* d(\vec{u})$. Therefore, for all $j \in \mathrm{Acc}(d)$, $\Delta' \vdash u_j \in [\![\vec{y} : \vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $\Delta' \vdash u' \in R$.
- **(R3)** Let $\Delta' \vdash u \in \overline{\mathbb{T}}_{\Delta_n \vdash D(\vec{t})}$ such that u is neutral and, for all u' such that $u \to u'$, $\Delta' \vdash u' \in R$. Then, $\Delta' \vdash u \in \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})}$. Assume now that $u \to^* d(\vec{u})$ with $\tau_d = (\vec{y} : \vec{U})D(\vec{v})$. Since u is neutral, there exists u' such that $u \to u'$ and $u' \to^* d(\vec{u})$. Therefore, for all $j \in \mathrm{Acc}(d)$, $\Delta' \vdash u_j \in [\![\vec{y} : \vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $\Delta' \vdash u \in R$.
- **(R4)** Let $\Delta' \vdash u \in R$ and assume that $\Delta' \subseteq \Delta'' \in \mathbb{E}$. Then, $\Delta'' \vdash u \in \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})}$. Assume now that $u \to^* d(\vec{u})$ with $\tau_d = (\vec{y} : \vec{U})D(\vec{v})$. Then, for all $j \in \mathrm{Acc}(d)$, $\Delta' \vdash u_j \in R_j$ where $R_j = [\![\vec{y} : \vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$. After Lemma 121 (b), R_j belongs to $\mathcal{R}_{\Delta_n \vdash U_j\theta}$ and therefore satisfies **(R4)**. Therefore $\Delta'' \vdash u_j \in R_j$ and $\Delta'' \vdash u \in R$.

Finally, we are left to show the properties (P1) to (P3). For (P1), the stability by reduction, this is immediate since, if $\vec{t} \to \vec{t'}$ then $\overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})} = \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t'})}$. For (P2) and (P3), it is easy to see that the functions have the same domain and are equal.

Lemma 128 φ^I is monotone in its inductive arguments.

Proof. Let $i \in \mathrm{Ind}(D)$. We have to show that $S_i \le S'_i$ implies $\varphi^I_{\Delta \vdash D}(\vec{a}) \subseteq \varphi^I_{\Delta \vdash D}(\vec{a}')$. Let $\Delta' \vdash u \in \varphi^I_{\Delta \vdash D}(\vec{a})$. We prove that $\Delta' \vdash u \in \varphi^I_{\Delta \vdash D}(\vec{a}')$. We have

$\Delta'\vdash u \in \overline{\mathbb{SN}}_{\Delta_n\vdash D(\vec{t})}$. Assume now that u reduces to a term of the form $d(\vec{u})$ with d a constructor of type $(\vec{y}:\vec{U})D(\vec{v})$. Let $j \in \mathrm{Acc}(d)$. We have to prove that $\Delta'\vdash u_j \in [\![\vec{y}:\vec{U}\vdash U_j]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}}$ where $\theta=\{\vec{y}\mapsto\vec{u}\}$ and, for all $Y\in\mathrm{FV}^\Box(U_j)$, $Y\xi'=S'_{\iota_Y}$.

We have $\Delta'\vdash u_j \in [\![\vec{y}:\vec{U}\vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ where, for all $Y\in\mathrm{FV}^\Box(U_j)$, $Y\xi=S_{\iota_Y}$. If, for all $Y\in\mathrm{FV}^\Box(U_j)$, $\iota_Y\neq i$, then $\xi=\xi'$ and $\Delta'\vdash u_j\in[\![\vec{y}:\vec{U}\vdash U_j]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}}$. Assume now that there exists $Y\in\mathrm{FV}^\Box(U_j)$ such that $\iota_Y=i$. Then, $Y\xi=S_i\le Y\xi'=S'_i$. By **(I2)**, $\mathrm{Pos}(Y,U_j)\subseteq\mathrm{Pos}^+(U_j)$. Finally, U_j satisfies **(I3)**, **(I4)** and **(I5)**.

We now prove by induction on T that, for all $\Gamma\vdash T\in\overline{\mathbb{TY}}$, $\Delta\in\mathbb{E}$, $\theta:\Gamma\to\Delta$, ξ,ξ' compatible with (θ,Γ,Δ) such that $Y\xi\le Y\xi'$ and, for all $X\neq Y$, $X\xi=X\xi'$:

– if $\mathrm{Pos}(Y,T)\subseteq\mathrm{Pos}^+(T)$ and T satisfies (I3), (I4) and (I5) then $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}\le[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$,

– if $\mathrm{Pos}(Y,T)\subseteq\mathrm{Pos}^-(T)$ and T satisfies (I3$^-$), (I4) and (I5) then $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}\ge[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$,

where (I3$^-$) is the property $\forall D\in\mathcal{CF}^\Box, D=_\mathcal{C} C\Rightarrow\mathrm{Pos}(D,T)\subseteq\mathrm{Pos}^-(T)$. We detail the first case only; the second is similar.

- $T=s$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}=\overline{\mathbb{SN}}_{\Delta\vdash s}=[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$.
- $T=E(\vec{t})$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}=I_{\Delta\vdash E}(\vec{a})$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}=I_{\Delta\vdash E}(\vec{a}')$ with $a_i=a'_i=\Delta\vdash t_i\theta$ if $x_i\in\mathcal{X}^\star$, and $a_i=(\Delta\vdash t_i\theta,S_i)$, $a'_i=(\Delta\vdash t_i\theta,S'_i)$, $S_i=[\![\Gamma\vdash t_i]\!]^I_{\Delta,\theta,\xi}$ and $S'_i=[\![\Gamma\vdash t_i]\!]^I_{\Delta,\theta,\xi'}$ if $x_i\in\mathcal{X}^\Box$. Since T satisfies (I3), (I4) and (I5), we have $E\in\mathcal{CF}^\Box$ and $E\le_\mathcal{C} D$. Therefore, $I_{\Delta\vdash E}$ is monotone in its inductive arguments. Let $i\le\alpha_E$. We show that t_i satisfies (I3), (I4) and (I5) :

  (I3) Let $D'=_\mathcal{C} D$. We have to show that $\mathrm{Pos}(D',t_i)\subseteq\mathrm{Pos}^+(t_i)$. If $\mathrm{Pos}(D',t_i)=\emptyset$, this is immediate. If there exists $p\in\mathrm{Pos}(D',t_i)$ then $i.p\in\mathrm{Pos}(D',T)$. Since $\mathrm{Pos}(D',T)\subseteq\mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T)=\{\varepsilon\}\cup\bigcup\{i.\mathrm{Pos}^+(t_i)\mid i\in\mathrm{Ind}(E)\}$, we have $i\in\mathrm{Ind}(E)$ and $p\in\mathrm{Pos}^+(t_i)$.

  (I4) and **(I5)** Let $D'>_\mathcal{C} D$ or $D'\in\mathcal{DF}^\Box$. We have to show that $\mathrm{Pos}(D',t_i)\subseteq\mathrm{Pos}^0(t_i)$. If $\mathrm{Pos}(D',t_i)=\emptyset$, this is immediate. If there exists $p\in\mathrm{Pos}(D',t_i)$ then $i.p\in\mathrm{Pos}(D',T)$. Since $\mathrm{Pos}(D',T)\subseteq\mathrm{Pos}^0(T)$ and $\mathrm{Pos}^0(T)=\{\varepsilon\}\cup\bigcup\{i.\mathrm{Pos}^0(t_i)\mid i\in\mathrm{Ind}(E)\}$, we have $i\in\mathrm{Ind}(E)$ and $p\in\mathrm{Pos}^0(t_i)$.

  Let us see now the relations between S_i and S'_i. If $\mathrm{Pos}(Y,t_i)=\emptyset$ then, by induction hypothesis, $S_i=S'_i$. If there exists $p\in\mathrm{Pos}(Y,t_i)$ then $i.p\in\mathrm{Pos}(Y,T)$. Since $\mathrm{Pos}(Y,T)\subseteq\mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T)=\{\varepsilon\}\cup\bigcup\{i.\mathrm{Pos}^+(t_i)\mid i\in\mathrm{Ind}(E)\}$, we necessary have $i\in\mathrm{Ind}(E)$ and $p\in\mathrm{Pos}^+(t_i)$. Therefore, by induction hypothesis, $S_i\le S'_i$. Finally, since $I_{\Delta\vdash E}$ is monotone in its inductive arguments, we can conclude that $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}\le[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$.
- $T=Y$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}=Y\xi\le Y\xi'=[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$. But $\mathrm{Pos}(Y,Y)=\varepsilon\subseteq\mathrm{Pos}^+(Y)$.
- $T=X\neq Y$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}=X\xi=X\xi'=[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$.
- $T=(x:U)V$. Let $\Gamma'=\Gamma,x:U$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}=\{\Delta'\vdash t\in\overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta}\mid\forall\Delta''\vdash u\in[\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}},\ \Delta''\vdash tu\in[\![\Gamma'\vdash V]\!]^I_{\Delta'',\theta',\xi|_{\Delta''}}\}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}=$

$\{\Delta' \vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta} \mid \forall\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}},\ \Delta'' \vdash tu \in [\![\Gamma' \vdash V]\!]^I_{\Delta'',\theta',\xi'|_{\Delta''}}\}$ where $\theta' = \theta \cup \{x \mapsto u\}$. We have to show that $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \subseteq [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$. Let $\Delta' \vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta}$ and $\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}}$. Since $\mathrm{Pos}(Y,T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}^-(U) \cup 2.\mathrm{Pos}^+(V)$, we have $\mathrm{Pos}(Y,U) \subseteq \mathrm{Pos}^-(U)$ and $\mathrm{Pos}(Y,V) \subseteq \mathrm{Pos}^+(V)$. We prove that U satisfies (I3$^-$), (I4) and (I5) :

(I3$^-$) Let $D' =_\mathcal{C} D$. We have to show that $\mathrm{Pos}(D',U) \subseteq \mathrm{Pos}^-(U)$. If $\mathrm{Pos}(D',U) = \emptyset$, this is immediate. If there exists $p \in \mathrm{Pos}(D',U)$ then $1.p \in \mathrm{Pos}(D',T)$. Since $\mathrm{Pos}(D',T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}^-(U) \cup 2.\mathrm{Pos}^+(V)$, we have $p \in \mathrm{Pos}^-(U)$.

(I4) and **(I5)** Let $D' =_\mathcal{C} D$ or $D' \in \mathcal{DF}^\square$. We have to show that $\mathrm{Pos}(D',U) \subseteq \mathrm{Pos}^0(U)$. If $\mathrm{Pos}(D',U) = \emptyset$, this is immediate. If there exists $p \in \mathrm{Pos}(D',U)$ then $1.p \in \mathrm{Pos}(D',T)$. Since $\mathrm{Pos}(D',T) \subseteq \mathrm{Pos}^0(T)$ and $\mathrm{Pos}^0(T) = 1.\mathrm{Pos}^0(U) \cup 2.\mathrm{Pos}^0(V)$, we have $p \in \mathrm{Pos}^0(U)$.

Similarly, V satisfies (I3), (I4) and (I5). Therefore, by induction hypothesis, $[\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}} \subseteq [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $[\![\Gamma' \vdash V]\!]^I_{\Delta'',\theta',\xi|_{\Delta''}} \subseteq [\![\Gamma' \vdash V]\!]^I_{\Delta'',\theta',\xi'|_{\Delta''}}$. Hence, $\Delta'' \vdash u \in [\![\Gamma \vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$, $\Delta'' \vdash tu \in [\![\Gamma' \vdash V]\!]^I_{\Delta'',\theta',\xi|_{\Delta''}}$ and $\Delta'' \vdash tu \in [\![\Gamma' \vdash V]\!]^I_{\Delta'',\theta',\xi'|_{\Delta''}}$. Therefore, $\Delta' \vdash t \in [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \subseteq [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$.

- $T = (X:K)V$. Similar to the previous case.
- $T = [x:U]V$. $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi}$ is the function which, to $\Delta' \vdash u \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, associates $[\![\Gamma' \vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}}$ where $\Gamma' = \Gamma, x:U$ and $\theta' = \theta \cup \{x \mapsto u\}$. $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$ is the function which, to $\Delta' \vdash u \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, associates $[\![\Gamma' \vdash V]\!]^I_{\Delta',\theta',\xi'|_{\Delta'}}$ where $\Gamma' = \Gamma, x:U$ and $\theta' = \theta \cup \{x \mapsto u\}$. We have to show that, for all $\Delta' \vdash u \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, $[\![\Gamma' \vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}} \le [\![\Gamma' \vdash V]\!]^I_{\Delta',\theta',\xi'|_{\Delta'}}$. Since $\mathrm{Pos}(Y,T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}(U) \cup 2.\mathrm{Pos}^+(V)$, we have $\mathrm{Pos}(Y,V) \subseteq \mathrm{Pos}^+(V)$. We now prove that V satisfies (I3), (I4) and (I5).

(I3) Let $D' =_\mathcal{C} D$. We have to show that $\mathrm{Pos}(D',V) \subseteq \mathrm{Pos}^+(V)$. If $\mathrm{Pos}(D',V) = \emptyset$, this is immediate. If there exists $p \in \mathrm{Pos}(D',V)$ then $2.p \in \mathrm{Pos}(D',T)$. Since $\mathrm{Pos}(D',T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}(U) \cup 2.\mathrm{Pos}^+(V)$, we have $p \in \mathrm{Pos}^+(V)$.

(I4) and **(I5)** Let $D' =_\mathcal{C} D$ or $D' \in \mathcal{DF}^\square$. We have to show that $\mathrm{Pos}(D',V) \subseteq \mathrm{Pos}^0(V)$. If $\mathrm{Pos}(D',V) = \emptyset$, this is immediate. If there exists $p \in \mathrm{Pos}(D',V)$ then $2.p \in \mathrm{Pos}(D',T)$. Since $\mathrm{Pos}(D',T) \subseteq \mathrm{Pos}^0(T)$ and $\mathrm{Pos}^0(T) = 1.\mathrm{Pos}(U) \cup 2.\mathrm{Pos}^0(V)$, we have $p \in \mathrm{Pos}^0(V)$.

Therefore, by induction hypothesis, $[\![\Gamma' \vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}} \le [\![\Gamma' \vdash V]\!]^I_{\Delta',\theta',\xi'|_{\Delta'}}$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \le [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$.

- $T = [X:K]V$. Similar to the previous case.
- $T = Vu$. We have $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi}(\Gamma \vdash u)$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'} = [\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi'}(\Gamma \vdash u)$. Since $\mathrm{Pos}(Y,T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}^+(V) \cup 2.\mathrm{Pos}(u)$, we have $\mathrm{Pos}(Y,V) \subseteq \mathrm{Pos}^+(V)$. We now prove that V satisfies (I3), (I4) and (I5).

(I3) Let $D' =_\mathcal{C} D$. We have to show that $\mathrm{Pos}(D',V) \subseteq \mathrm{Pos}^+(V)$. If $\mathrm{Pos}(D',V) = \emptyset$, this is immediate. If there exists $p \in \mathrm{Pos}(D',V)$ then $1.p \in \mathrm{Pos}(D',T)$.

Since $\mathrm{Pos}(D', T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}^+(V) \cup 2.\mathrm{Pos}(u)$, we have $p \in \mathrm{Pos}^+(V)$.

(I4) and **(I5)** Let $D' =_{\mathcal{C}} D$ or $D' \in \mathcal{DF}^\Box$. We have to prove that $\mathrm{Pos}(D', V) \subseteq \mathrm{Pos}^0(V)$. If $\mathrm{Pos}(D', V) = \emptyset$, this is immediate. If there exists $p \in \mathrm{Pos}(D', V)$ then $1.p \in \mathrm{Pos}(D', T)$. Since $\mathrm{Pos}(D', T) \subseteq \mathrm{Pos}^0(T)$ and $\mathrm{Pos}^0(T) = 1.\mathrm{Pos}^0(V) \cup 2.\mathrm{Pos}(u)$, we have $p \in \mathrm{Pos}^0(V)$.

Therefore, by induction hypothesis, $[\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi'}$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$.

- $T = VU$. We have $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi}(\Gamma \vdash U, [\![\Gamma \vdash U]\!]^I_{\Delta,\theta,\xi})$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'} = [\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi'}(\Gamma \vdash U, [\![\Gamma \vdash U]\!]^I_{\Delta,\theta,\xi'})$. Since $\mathrm{Pos}(Y, T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}^+(V)$, we have $\mathrm{Pos}(Y, V) \subseteq \mathrm{Pos}^+(V)$ and $\mathrm{Pos}(Y, U) = \emptyset$. We have seen in the previous case that V satisfies (I3), (I4) and (I5). We now show that U satisfies (I3), (I3$^-$), (I4) and (I5).

  (I3) and **(I3$^-$)** Let $D' =_{\mathcal{C}} D$. We have to prove that $\mathrm{Pos}(D', U) \subseteq \mathrm{Pos}^+(U) \cup \mathrm{Pos}^-(U)$. If there exists $p \in \mathrm{Pos}(D', U)$ then $2.p \in \mathrm{Pos}(D', T)$. Since $\mathrm{Pos}(D', T) \subseteq \mathrm{Pos}^+(T)$ and $\mathrm{Pos}^+(T) = 1.\mathrm{Pos}^+(V)$, this is not possible. Therefore, $\mathrm{Pos}(D', U) = \emptyset \subseteq \mathrm{Pos}^+(U) \cup \mathrm{Pos}^-(U)$.

  (I4) and **(I5)** Let $D' =_{\mathcal{C}} D$ or $D' \in \mathcal{DF}^\Box$. We have to prove that $\mathrm{Pos}(D', V) \subseteq \mathrm{Pos}^0(V)$. If there exists $p \in \mathrm{Pos}(D', U)$ then $2.p \in \mathrm{Pos}(D', T)$. Since $\mathrm{Pos}(D', T) \subseteq \mathrm{Pos}^0(T)$ and $\mathrm{Pos}^0(T) = 1.\mathrm{Pos}^0(V)$, this is not possible. Therefore, $\mathrm{Pos}(D', U) = \emptyset \subseteq \mathrm{Pos}^0(U)$.

  Therefore, by induction hypothesis, $[\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma \vdash V]\!]^I_{\Delta,\theta,\xi'}$, $[\![\Gamma \vdash U]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma \vdash U]\!]^I_{\Delta,\theta,\xi'}$ and $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi'}$.

Lemma 129 φ is monotone.

Proof. Let $I, I' \in \mathcal{I}^m$ such that $I \leq I'$. We have to prove that, for all $D =_{\mathcal{F}} C$, $\Delta \in \mathbb{E}$ and $\vec{a}$, $\varphi^I_{\Delta\vdash D}(\vec{a}) \leq \varphi^{I'}_{\Delta\vdash D}(\vec{a})$. Let $\Delta' \vdash u \in \varphi^I_{\Delta\vdash D}(\vec{a})$. We prove that $\Delta' \vdash u \in \varphi^{I'}_{\Delta\vdash D}(\vec{a})$. We have $\Delta' \vdash u \in \overline{\mathbb{SN}}_{\Delta_n \vdash D(\vec{t})}$. Assume now that u reduces to a term of the form $d(\vec{u})$ with d a constructor of type $(\vec{y} : \vec{U})D(\vec{v})$. Let $j \in \mathrm{Acc}(d)$. We have to prove that $\Delta' \vdash u_j \in [\![\vec{y} : \vec{U} \vdash U_j]\!]^{I'}_{\Delta',\theta,\xi|_{\Delta'}}$ where $\theta = \{\vec{y} \mapsto \vec{u}\}$ and, for all $Y \in \mathrm{FV}^\Box(U_j)$, $Y\xi = S_{\iota_Y}$.

We have $\Delta' \vdash u_j \in [\![\vec{y} : \vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and U_j satisfies **(I3)**, **(I4)** and **(I5)**. We then prove by induction on T that, for all $\Gamma \vdash T \in \overline{\mathbb{TY}}$, $\Delta \in \mathbb{E}$, $\theta : \Gamma \to \Delta$, ξ compatible with (θ, Γ, Δ) :

– if T satisfies (I3), (I4) and (I5) then $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma \vdash T]\!]^{I'}_{\Delta,\theta,\xi}$,

– if T satisfies (I3$^-$), (I4) and (I5) then $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} \geq [\![\Gamma \vdash T]\!]^{I'}_{\Delta,\theta,\xi}$,

where (I3$^-$) is the property $\forall D \in \mathcal{CF}^\Box, D =_{\mathcal{C}} C \Rightarrow \mathrm{Pos}(D, T) \subseteq \mathrm{Pos}^-(T)$. We just detail the first case; the second case is similar.

- $T = s$. We have $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} = \overline{\mathbb{SN}}_{\Delta\vdash s} = [\![\Gamma \vdash T]\!]^{I'}_{\Delta,\theta,\xi}$.
- $T = E(\vec{t})$. We have $[\![\Gamma \vdash T]\!]^I_{\Delta,\theta,\xi} = I_{\Delta\vdash E}(\vec{a})$ and $[\![\Gamma \vdash T]\!]^{I'}_{\Delta,\theta,\xi} = I'_{\Delta\vdash E}(\vec{a}')$ with $a_i = a'_i = \Delta \vdash t_i\theta$ if $x_i \in \mathcal{X}^\star$, and $a_i = (\Delta \vdash t_i\theta, S_i)$, $a'_i = (\Delta \vdash t_i\theta, S'_i)$, $S_i =$

$[\![\Gamma\vdash t_i]\!]^I_{\Delta,\theta,\xi}$ and $S'_i = [\![\Gamma\vdash t_i]\!]^{I'}_{\Delta,\theta,\xi}$ if $x_i \in \mathcal{X}^\Box$. Since T satisfies (I3), (I4) and (I5), we have $E \in \mathcal{CF}^\Box$ and $E \leq_{\mathcal{C}} D$. Therefore, $I_{\Delta\vdash E}$ and $I'_{\Delta\vdash E}$ are monotone in their inductive arguments. We have seen in the previous lemma that t_i satisfies (I3), (I4) and (I5). Therefore, by induction hypothesis, $S_i \leq S'_i$. Finally, since $I_{\Delta\vdash E}$ is monotone in its inductive arguments and $I \leq I'$, we can conclude that $I_{\Delta\vdash E}(\vec{a}) \leq I_{\Delta\vdash E}(\vec{a}') \leq I'_{\Delta\vdash E}(\vec{a}')$ and therefore that $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'}$.

- $T = X$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = Y\xi = [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$.
- $T = (x\!:\!U)V$. Let $\Gamma' = \Gamma, x\!:\!U$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = \{\Delta'\vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta} \mid \forall\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}},\ \Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta'',\theta',\xi|_{\Delta''}}\}$ and $[\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi} = \{\Delta'\vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta} \mid \forall\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^{I'}_{\Delta',\theta,\xi|_{\Delta'}},\ \Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^{I'}_{\Delta'',\theta',\xi|_{\Delta''}}\}$ where $\theta' = \theta \cup \{x \mapsto u\}$. We have to prove that $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \subseteq [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$. Let $\Delta'\vdash t \in \overline{\mathbb{T}}_{\Delta\vdash(x:U\theta)V\theta}$ and $\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}}$. We have seen in the previous lemma that U satisfies (I3$^-$), (I4) and (I5) and that V satisfies (I3), (I4) and (I5). Therefore, by induction hypothesis, $[\![\Gamma\vdash U]\!]^{I'}_{\Delta',\theta,\xi|_{\Delta'}} \subseteq [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ and $[\![\Gamma'\vdash V]\!]^I_{\Delta'',\theta',\xi|_{\Delta''}} \subseteq [\![\Gamma'\vdash V]\!]^{I'}_{\Delta'',\theta',\xi|_{\Delta''}}$. Hence, $\Delta''\vdash u \in [\![\Gamma\vdash U]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$, $\Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^I_{\Delta'',\theta',\xi|_{\Delta''}}$ and $\Delta''\vdash tu \in [\![\Gamma'\vdash V]\!]^{I'}_{\Delta'',\theta',\xi|_{\Delta''}}$. Therefore, $\Delta'\vdash t \in [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \subseteq [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$.
- $T = (X\!:\!K)V$. Similar to the previous case.
- $T = [x\!:\!U]V$. $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$ is the function which, to $\Delta'\vdash u \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, associates $[\![\Gamma'\vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}}$ where $\Gamma' = \Gamma, x:U$ and $\theta' = \theta \cup \{x \mapsto u\}$. $[\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$ is the function which, to $\Delta'\vdash u \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, associates $[\![\Gamma'\vdash V]\!]^{I'}_{\Delta',\theta',\xi|_{\Delta'}}$ where $\Gamma' = \Gamma, x:U$ and $\theta' = \theta \cup \{x \mapsto u\}$. We have to show that, for all $\Delta'\vdash u \in \overline{\mathbb{T}}_{\Delta\vdash U\theta}$, $[\![\Gamma'\vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}} \leq [\![\Gamma'\vdash V]\!]^{I'}_{\Delta',\theta',\xi|_{\Delta'}}$. We have seen in the previous lemma that V satisfies (I3), (I4) and (I5). Therefore, by induction hypothesis, $[\![\Gamma'\vdash V]\!]^I_{\Delta',\theta',\xi|_{\Delta'}} \leq [\![\Gamma'\vdash V]\!]^{I'}_{\Delta',\theta',\xi|_{\Delta'}}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$.
- $T = [X\!:\!K]V$. Similar to the previous case.
- $T = Vu$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Gamma\vdash u)$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi'}(\Gamma\vdash u)$. We have seen in the previous lemma that V satisfies (I3), (I4) and (I5). Therefore, by induction hypothesis, $[\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma\vdash V]\!]^{I'}_{\Delta,\theta,\xi}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$.
- $T = VU$. We have $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi}(\Gamma\vdash U, [\![\Gamma\vdash U]\!]^I_{\Delta,\theta,\xi})$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi'} = [\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi'}(\Gamma\vdash U, [\![\Gamma\vdash U]\!]^I_{\Delta,\theta,\xi'})$. we have seen in the previous lemma that U satisfies (I3), (I3$^-$), (I4) and (I5), and V satisfies (I3), (I4) and (I5). Therefore, by induction hypothesis, $[\![\Gamma\vdash V]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma\vdash V]\!]^{i'}_{\Delta,\theta,\xi}$, $[\![\Gamma\vdash U]\!]^I_{\Delta,\theta,\xi} = [\![\Gamma\vdash U]\!]^{I'}_{\Delta,\theta,\xi}$ and $[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi} \leq [\![\Gamma\vdash T]\!]^{I'}_{\Delta,\theta,\xi}$. ■

Since $(\mathcal{I}^m, \leq)$ is a complete lattice, φ as a least fix point I which is an interpretation for all the constant predicate symbols equivalent to C. Hence, by induction on $>_{\mathcal{C}}$, we obtain an interpretation I for the constant predicate symbols.

In the case of a primitive constant predicate symbol, the interpretation is simply the set of strongly normalizable terms of this type :

Lemma 130 (Interpretation of primitive constant predicate symbols) If C is a primitive constant predicate symbol then $I_{\Delta\vdash C} = \top_{\Delta\vdash C}$.

Proof. Since $I_{\Delta\vdash C} \le \top_{\Delta\vdash C}$, it suffices to prove that, for all $u \in \mathcal{SN}$, C primitive of type $(\vec{x}:\vec{T})\star$, $\vec{a}$ arguments of $I_{\Delta\vdash C}$ with $a_i = \Delta_i \vdash t_i$ if $x_i \in \mathcal{X}^\star$ and $a_i = (\Delta_i \vdash t_i, S_i)$ if $x_i \in \mathcal{X}^\square$, and $\Delta' \supseteq \Delta_n$, if $\Delta' \vdash u : C(\vec{t})$ then $\Delta' \vdash u \in I_{\Delta\vdash C}(\vec{a})$, by induction on u with $\to \cup \rhd$ as well-founded ordering. Assume that $u \to^* c(\vec{u})$ with c a constructor of type $(\vec{y}:\vec{U})C(\vec{v})$. If $u \to^+ c(\vec{u})$, we can conclude by induction hypothesis. So, assume that $u = c(\vec{u})$. In this case, we have to prove that, for all $j \in \mathrm{Acc}(c)$, $\Delta' \vdash u_j \in [\![\vec{y}:\vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}}$ where $\theta = \{\vec{y} \mapsto \vec{u}\}$ and, for all $Y \in \mathrm{FV}^\square(U_j)$, $Y\xi = S_{\iota_Y}$. By definition of primitive constant predicate symbols, for all $j \in \mathrm{Acc}(c)$, U_j is of the form $D(\vec{w})$ with D a primitive constant predicate symbol. Assume that $\tau_D = (\vec{z}:\vec{V})\star$. Let $a'_i = \Delta' \vdash w_i\theta$ if $z_i \in \mathcal{X}^\star$, and $a'_i = (\Delta' \vdash w_i\theta, [\![\vec{y}:\vec{U} \vdash w_i]\!]^I_{\Delta',\theta,\xi'|_{\Delta'}})$ if $z_i \in \mathcal{X}^\square$. Since $u_j \in \mathcal{SN}$ and $\Delta' \vdash u_j : D(\vec{w}\theta)$, by induction hypothesis, $\Delta' \vdash u_j \in I_{\Delta\vdash D}(\vec{a}')$. By **(P3)**, $I_{\Delta\vdash D}(\vec{a}') = I_{\Delta'\vdash D}(\vec{a}')$ and $[\![\vec{y}:\vec{U} \vdash U_j]\!]^I_{\Delta',\theta,\xi|_{\Delta'}} = I_{\Delta'\vdash D}(\vec{a}')$. Therefore, $\Delta' \vdash u \in I_{\Delta\vdash C}(\vec{a})$. ■

8.5 Reductibility ordering

In this subsection, we assume given an interpretation J for the defined predicate symbols and we denote $[\![\Gamma \vdash T]\!]^{I\cup J}_{\Delta,\theta,\xi}$ by $[\![\Gamma \vdash T]\!]_{\Delta,\theta,\xi}$.

The fix point of the function φ defined in the previous subsection can be reached by transfinite iteration from the smallest element of $\mathcal{I}^m$. Let $\varphi^{\mathfrak{a}}$ be the interpretation after $\mathfrak{a}$ iterations.

Definition 131 (Order of a computable term) The *order* of $\Delta' \vdash t \in I_{\Delta\vdash C}(\vec{a})$, $o(\Delta' \vdash t)$, is the smallest ordinal $\mathfrak{a}$ such that $\Delta' \vdash t \in \varphi^{\mathfrak{a}}_{\Delta\vdash C}(\vec{a})$.

This notion of order will enable us to define a well-founded ordering in which recursive definitions on strictly positive predicates strictly decrease. Indeed, in this case, the subterm ordering is not sufficient. In the example of the addition on ordinals, we have the rule :

$$+(x, lim(f)) \quad\to\quad lim([n\!:\!nat] + (x, fn))$$

We have a recursive call with fn as argument, which is not a subterm of $lim(f)$. However, thanks to the definition of the interpretation for constant predicate symbols and products, we can say that, if $lim(f)$ is computable then f is computable and, for all computable n, fn is computable. So, the order of $lim(f)$ is greater than the one of fn : $o(lim(f)) > o(fn)$.

Definition 132 (Reductibility ordering) We assume given a precedence $\ge_{\mathcal{F}}$ on $\mathcal{F}$ and a status assignment *stat* compatible with $\ge_{\mathcal{F}}$. Let f be a symbol of non null arity and status $stat_f = lex(m_1, \ldots, m_k)$. Let Θ_f be the set of (g, Δ, θ, ξ) such that, if $\tau_g = (\vec{x}:\vec{T})U$ and $\Gamma_g = \vec{x}:\vec{T}$ then $g =_{\mathcal{F}} f$, $\Delta \in \mathbb{E}$, $\theta : \Gamma_g \to \Delta$, ξ is compatible with $(\theta, \Gamma_g, \Delta)$, $(\theta, \Gamma_g, \Delta)$ is valid w.r.t. ξ. We equip Θ_f with the ordering $\sqsupset_f$ defined by :

- $(g, \Delta, \theta, \xi) \sqsupset_f (g', \Delta', \theta', \xi')$ if $\vec{m}\theta\ (\sqsupset_f^1, \ldots, \sqsupset_f^k)_{\mathrm{lex}}\ \vec{m}\theta'$,
- $mul(\vec{u}) \sqsupset_f^i mul(\vec{u}')$ if $\{\vec{u}\}\ (\sqsupset^i)_{\mathrm{mul}}\ \{\vec{u}'\}$ with :
  - $\sqsupset^i = \gg$ if $i \in SP(f)$ where $u \gg u'$ if $o(\Delta \vdash u) > o(\Delta' \vdash u')$,
  - $\sqsupset^i = \to \cup \rhd$ otherwise.

We equip $\Theta = \bigcup\{\Theta_f \mid f \in \mathcal{F}\}$ with the *reductibility ordering* $\sqsupset$ defined by $(g, \Delta, \theta, \xi) \sqsupset (g', \Delta', \theta', \xi')$ if $g >_{\mathcal{F}} g'$ or, $g =_{\mathcal{F}} g'$ and $(g, \Delta, \theta, \xi) \sqsupset_f (g', \Delta', \theta', \xi')$.

Lemma 133 The reductibility ordering is well-founded and compatible with $\to$: if $\theta \to \theta'$ then $(g, \Delta, \theta, \xi) \sqsupseteq (g, \Delta, \theta', \xi)$.

Proof. The reductibility ordering is well-founded since the ordinals are well-founded and the lexicographic and multiset extensions preserve the well-foundedness. It is compatible with $\to$ by definition of the interpretation for constant predicate symbols. ■

We check hereafter that the accessibility relation is correct : an accessible subterm of a computable term is computable. Then we check that the ordering on arguments if also correct : if $l_i >_2 u_j$ and l_i is computable then u_j is computable and has an order smaller than the one of l_i.

Lemma 134 (Correctness of accessibility) If $t : T \rhd_1^\rho u : U$, $\Gamma \vdash t\rho : T\rho$, $\sigma : \Gamma \to \Delta$ and $\Delta \vdash t\sigma \in [\![\Gamma \vdash T\rho]\!]_{\Delta,\sigma,\xi}$ then $\Gamma \vdash u\rho : U\rho$ and $\Delta \vdash u\sigma \in [\![\Gamma \vdash U\rho]\!]_{\Delta,\sigma,\xi}$.

Proof. By definition of $\rhd_1^\rho$, we have t of the form $c(\vec{u})$ with c a constructor of type $(\vec{y}:\vec{U})C(\vec{v})$, $T\rho = C(\vec{v})\gamma\rho$ where $\gamma = \{\vec{y} \mapsto \vec{u}\}$, $u = u_j$ with $j \in \mathrm{Acc}(c)$ and $U\rho = U_j\gamma\rho$. Assume that $\tau_C = (\vec{x}:\vec{T})\star$. Then, $[\![\Gamma \vdash C(\vec{v})\gamma\rho]\!]_{\Delta,\sigma,\xi} = I_{\Delta\vdash C}(\vec{a})$ with $a_i = \Delta \vdash v_i\gamma\rho\sigma$ if $x_i \in \mathcal{X}^\star$, and $a_i = (\Delta \vdash v_i\gamma\rho\sigma, S_i)$ where $S_i = [\![\Gamma \vdash v_i\gamma\rho]\!]_{\Delta,\sigma,\xi}$ if $x_i \in \mathcal{X}^\Box$. By definition of $I_{\Delta\vdash C}$, $\Delta \vdash u_j\sigma \in [\![\Gamma_c \vdash U_j]\!]_{\Delta,\gamma\rho\sigma,\xi'}$ with, for all $Y \in \mathrm{FV}^\Box(U_j)$, $Y\xi' = S_{\iota_Y} = [\![\Gamma \vdash v_{\iota_Y}\gamma\rho]\!]_{\Delta,\sigma,\xi}$. By **(I6)**, $v_{\iota_Y} = Y$ and $Y\xi' = [\![\Gamma \vdash Y\gamma\rho]\!]_{\Delta,\sigma,\xi}$. From $\Gamma \vdash t\rho : T\rho$, by inversion, we deduce that, for all i, $\Gamma \vdash u_i\rho : U_i\gamma\rho$. Therefore, $\gamma\rho : \Gamma_c \to \Gamma$. Hence, by candidates substitution, $[\![\Gamma_c \vdash U_j]\!]_{\Delta,\gamma\rho\sigma,\xi'} = [\![\Gamma \vdash U_j\gamma\rho]\!]_{\Delta,\sigma,\xi}$ and $\Delta \vdash u\sigma \in [\![\Gamma \vdash U\rho]\!]_{\Delta,\sigma,\xi}$. ■

Lemma 135 (Correctness of the ordering on arguments) Assume that $t : T >_2 u : U$, that is, that t is of the form $c(\vec{t})$ with c a constructor of type $(\vec{x}:\vec{T})C(\vec{v})$, u of the form $x\vec{u}$ with $x \in \mathrm{dom}(\Gamma_0)$, $x\Gamma_0$ of the form $(\vec{y}:\vec{U})D(\vec{w})$, $D =_{\mathcal{C}} C$ and $t : T\ (\rhd_2^\rho)^+\ x : V$ with $V\rho = x\Gamma_0$.

Let $\theta = \{\vec{y} \mapsto \vec{u}\}$. If $\Gamma \vdash t\rho : T\rho$, $\sigma : \Gamma \to \Delta$, $\Delta \vdash t\sigma \in [\![\Gamma \vdash T\rho]\!]_{\Delta,\sigma,\xi}$ and, for all i, $\Delta \vdash u_i\sigma \in [\![\Gamma \vdash U_i\theta]\!]_{\Delta,\sigma,\xi}$, then $\Delta \vdash x\sigma\vec{u}\sigma \in [\![\Gamma \vdash D(\vec{w})\theta]\!]_{\Delta,\sigma,\xi}$ and $o(\Delta \vdash t\sigma) > o(\Delta \vdash x\sigma\vec{u}\sigma)$.

Proof. Let p be the path from t to x followed in $t : T\ (\rhd_2^\rho)^+\ x : V$. We show that if $p = j_1 \ldots j_n j_{n+1}$ then $c(\vec{t}) : C(\vec{v})\gamma = c_0(\vec{t}^0) : C_0(\vec{v}^0)\gamma_0 \rhd_2^\rho c_1(\vec{t}^1) :$

$C_1(\vec{v}^1)\gamma_1 \rhd_2^\rho \dots \rhd_2^\rho c_n(\vec{t}^n) : C_n(\vec{v}^n)\gamma_n \rhd_2^\rho x : V$ with, if $\tau_{c_i} = (\vec{x}^i : \vec{T}^i)C(\vec{v}^i)$ and $\gamma_i = \{\vec{x}^i \mapsto \vec{t}^i\}$:

– for all $i < n$, $c_{i+1}(\vec{t}^{i+1}) = t^i_{j_{i+1}}$, $T^i_{j_{i+1}} = C_{i+1}(\vec{w}^{i+1})$ and $\vec{w}^{i+1}\gamma_i\rho = \vec{v}^{i+1}\gamma_{i+1}\rho$,

– $T^n_{j_{n+1}} = (\vec{y} : \vec{U})D(\vec{w}')$ and $\vec{w}'\gamma_n = \vec{w}$.

We proceed by induction on n. If $n = 0$, this is immediate. So, assume that $c(\vec{t}) : C(\vec{v})\gamma \rhd_2^\rho t_{j_1} : T_{j_1}\gamma_0 (\rhd_2^\rho)^+ x : V$. Since $t_{j_1} : T_{j_1}\gamma_0 (\rhd_2^\rho)^+ x : V$, $t_{j_1} = c_1(\vec{t}^1)$ with $\tau_{c_1} = (\vec{x}^1 : \vec{T}^1)C_1(\vec{v}^1)$ and $T_{j_1}\gamma_0\rho = C_1(\vec{v}^1)\gamma_1\rho$ where $\gamma_1 = \{\vec{x}^1 \mapsto \vec{t}^1\}$. By definition of $\rhd_2^\rho$, T_{j_1} is of the form $(\vec{z} : \vec{V})C_1'(\vec{w}^1)$. Therefore, $|\vec{z}| = 0$, $C_1' = C_1$ and $\vec{w}^1\gamma_0\rho = \vec{v}^1\gamma_1\rho$. Hence, we can conclude by induction hypothesis on $t_{j_1} : T_{j_1}\gamma (\rhd_2^\rho)^+ x : V$.

Since we are in an admissible inductive structure **(A2)**, we can say that $C \geq_\mathcal{F} C_1 \geq_\mathcal{F} \dots \geq_\mathcal{F} C_n \geq D$. Since $D =_\mathcal{F} C$ and $>_\mathcal{F}$ is well-founded, we get $C =_\mathcal{F} C_1 =_\mathcal{F} \dots =_\mathcal{F} D$.

Let $w_{n+1} = t|_p\vec{u}$, $W_{n+1} = D(\vec{w})\theta$ and, for all $i \leq n$, $w_i = t|_{j_1\dots j_i}$ and $W_i = C_i(\vec{v}^i)\gamma_i$. We show by induction on n that, for all $i \leq n+1$, $\Delta \vdash w_i\sigma \in [\![\Gamma \vdash W_i]\!]_{\Delta,\sigma,\xi}$ and, for all $i \leq n$, $o(\Delta \vdash w_i\sigma) > o(\Delta \vdash w_{i+1}\sigma)$.

- Case $n > 0$. $c(\vec{t}) : C(\vec{v})\gamma \rhd_2^\rho t_{j_1} : T_{j_1}\gamma (\rhd_2^\rho)^+ x : V$. By hypothesis, we have $\Gamma \vdash c(\vec{t})\rho : C(\vec{v})\gamma\rho$ and $\Delta \vdash c(\vec{t})\sigma \in [\![\Gamma \vdash C(\vec{v})\gamma\rho]\!]_{\Delta,\sigma,\xi}$. By correctness of the accessibility relation, $\Gamma \vdash t_{j_1}\rho : T_{j_1}\gamma\rho$ and $\Delta \vdash t_{j_1}\sigma \in [\![\Gamma \vdash T_{j_1}\gamma\rho]\!]_{\Delta,\sigma,\xi}$. Since $T_{j_1}\gamma\rho = C_1(\vec{v}^1)\gamma_1\rho$ and $C_1 =_\mathcal{F} C$, $o(\Delta \vdash t\sigma) > o(\Delta \vdash t_{j_1}\sigma)$. Hence, we can conclude by induction hypothesis on $t_{j_1} : T_{j_1}\gamma$.
- Case $n = 0$. $c(\vec{t}) : C(\vec{v})\gamma \rhd_2^\rho x : V$. Like in the case $n > 0$, we have $\Delta \vdash x\sigma \in [\![\Gamma \vdash V\rho]\!]_{\Delta,\sigma,\xi}$. Moreover, we have $V\rho = (\vec{y} : \vec{U})D(\vec{w})$. Let $m = |\vec{u}|$ and $\theta_k = \{y_1 \mapsto u_1, \dots, y_k \mapsto u_k\}$. We prove by induction on $k \leq m$ that $\Delta \vdash x\sigma\, u_1\sigma \dots u_k\sigma \in [\![\Gamma \vdash (y_{k+1} : U_{k+1}\theta_k) \dots (y_m : U_m\theta_k)D(\vec{w})\theta_k]\!]_{\Delta,\sigma,\xi}$ and therefore that $o(\Delta \vdash t\sigma) > o(\Delta \vdash t'\sigma)$.

  If $k = 0$, this is immediate. So, assume that $k > 0$. By induction hypothesis, $\Delta \vdash x\sigma\, u_1\sigma \dots u_{k-1}\sigma \in [\![\Gamma \vdash (y_k : U_k\theta_{k-1}) \dots (y_m : U_m\theta_{k-1})D(\vec{w})\theta_{k-1}]\!]_{\Delta,\sigma,\xi}$. Since $\Delta \vdash u_k\sigma \in [\![\Gamma \vdash U_k\theta_{k-1}]\!]_{\Delta,\sigma,\xi}$, we have $\Delta \vdash x\sigma\, u_1\sigma \dots u_k\sigma \in [\![\Gamma, y_k : U_k\theta_{k-1} \vdash (y_{k+1} : U_{k+1}\theta_{k-1}) \dots (y_m : U_m\theta_{k-1})D(\vec{w})\theta_{k-1}]\!]_{\Delta,\sigma',\xi'}$ where $\sigma' = \sigma \cup \{y_k \mapsto u_k\sigma\}$, $\xi' = \xi$ if $y_k \in \mathcal{X}^\star$ and $\xi' = \xi \cup \{y_k \mapsto [\![\Gamma \vdash u_k]\!]_{\Delta,\sigma,\xi}\}$ if $y_k \in \mathcal{X}^\Box$. Hence, by candidates substitution, $[\![\Gamma, y_k : U_k\theta_{k-1} \vdash (y_{k+1} : U_{k+1}\theta_{k-1}) \dots (y_m : U_m\theta_{k-1})D(\vec{w})\theta_{k-1}]\!]_{\Delta,\sigma',\xi'} = [\![\Gamma \vdash (y_{k+1} : U_{k+1}\theta_k) \dots (y_m : U_m\theta_k)D(\vec{w})\theta_k]\!]_{\Delta,\sigma,\xi}$. ■

8.6 Interpretation of defined predicate symbols

We define the interpretation J for defined predicate symbols by induction on $\succ$ w.r.t. hypothesis **(A3)**.

Let F be a defined predicate symbol and assume that we already have defined an interpretation K for all the symbols smaller than F. Let $[F]$ be the set of symbols equivalents to F. We defined the interpretation for $[F]$ depending on whether $[F]$ is primitive, positive or recursive.

To make the notations simpler, we will denote $[\![\Gamma \vdash T]\!]^{I \cup K \cup J}_{\Delta,\theta,\xi}$ by $[\![\Gamma \vdash T]\!]^J_{\Delta,\theta,\xi}$. And for the arguments of the interpretation, we follow the notations used in the previous section.

8.6.1 Primitive systems

Definition 136 For every $G \in [F]$, we take $J_{\Delta\vdash G} = \top_{\Delta\vdash G}$.

8.6.2 Positive, small and simple systems

Let $\mathcal{J}$ be the set of the interpretations for $[F]$ and $\leq$ the relation on $\mathcal{J}$ defined by $J \leq J'$ if, for all $G \in [F]$ and $\Delta \in \mathbb{E}$, $J_{\Delta\vdash G} \leq_{\Delta\vdash G} J'_{\Delta\vdash G}$. Since $(\mathcal{R}_{\Delta\vdash G}, \leq_{\Delta\vdash G})$ is a complete lattice, it is easy to see that $(\mathcal{J}, \leq)$ is also a complete lattice.

Definition 137 Let ψ be the function which, to $J \in \mathcal{J}$, associates the interpretation ψ^J defined by :

$$\psi^J_{\Delta\vdash G}(\vec{a}) = \begin{cases} [\![\Gamma\vdash r]\!]^J_{\Delta_n,\sigma,\xi} & \text{if } \vec{t} \in \mathcal{WN} \cap \mathcal{CR},\ \vec{t}{\downarrow} = \vec{l}\sigma \text{ and } (G(\vec{l}) \to r, \Gamma, \rho) \in \mathcal{R} \\ \top_{\Delta_n\vdash G(\vec{t})} & \text{otherwise} \end{cases}$$

where, for all $X \in \mathrm{FV}^{\Box}(r)$, $X\xi = S_{\kappa_X}|_{\Delta_n}$. Hereafter, we show that ψ is monotone. Therefore, we can take $J_{\Delta\vdash G} = \mathrm{lfp}(\psi)_{\Delta\vdash G}$.

Lemma 138 ψ^J is a well defined interpretation.

Proof. By simplicity, at most one rule can be applied at the top of $G(\vec{t}{\downarrow})$. The existence of κ_X is the smallness hypothesis. By **(S4)**, if $\vec{t} = \vec{l}\sigma$ then $\sigma : \Gamma \to \Delta_n$. Moreover, ξ is compatible with $(\sigma, \Gamma, \Delta_n)$ since, for all $X \in \mathrm{FV}^{\Box}(r)$, $X\xi = S_{\kappa_X}|_{\Delta_n} \in \mathcal{R}_{\Delta_n\vdash X\sigma}$ since $S_{\kappa_X} \in \mathcal{R}_{\Delta_{\kappa_X}\vdash t_{\kappa_X}}$ and, by smallness, $t_{\kappa_X} = l_{\kappa_X}\sigma = X\sigma$.

We now show that ψ^J is an interpretation for $[F]$. After Lemma 121 (b), $[\![\Gamma\vdash r]\!]^J_{\Delta_n,\sigma,\xi} \in \mathcal{R}_{\Delta_n\vdash r\sigma} = \mathcal{R}_{\Delta_n\vdash F(\vec{t})}$. Moreover, $\top_{\Delta_n\vdash F(\vec{t})} \in \mathcal{R}_{\Delta_n\vdash F(\vec{t})}$. Thus we are left with proving the properties (P1) to (P3).

- **(P1)** Assume that $\vec{t} \to \vec{t'}$. By **(A1)**, $\to$ is confluent. Therefore, $\{\vec{t}\} \subseteq \mathcal{WN}$ iff $\{\vec{t'}\} \subseteq \mathcal{WN}$, and if $\{\vec{t}\} \subseteq \mathcal{WN}$ then $\vec{t}{\downarrow} = \vec{t'}{\downarrow}$. So, $\psi^J_{\Delta\vdash G}(\vec{a}) = \psi^J_{\Delta\vdash G}(\vec{a'})$.
- **(P2)** Assume that $\Delta_n \subseteq \Delta' \in \mathbb{E}$. After Lemma 121 (d), $[\![\Gamma\vdash r]\!]^J_{\Delta_n,\sigma,\xi}|_{\Delta'} = [\![\Gamma\vdash r]\!]^I_{\Delta',\sigma,\xi|_{\Delta'}}$. Moreover, $\top_{\Delta_n\vdash F(\vec{t})}|_{\Delta'} = \top_{\Delta'\vdash F(\vec{t})}$. Therefore, $\psi^J_{\Delta\vdash G}(\vec{a})|_{\Delta'} = \psi^J_{\Delta'\vdash G}(\vec{a})$. Now, assume that $\Delta_k \subseteq \Delta'_k \in \mathbb{E}$. Let $a'_k = \Delta'_k \vdash t_k$ if $x_k \in \mathcal{X}^\star$, and $a'_k = (\Delta'_k \vdash t_k, S_k|_{\Delta'_k})$ if $x_k \in \mathcal{X}^\Box$. Then, $\psi^J_{\Delta\vdash G}(a_1, \ldots, a_{k-1}, a_k)|_{\Delta'_k}$ and $\psi^J_{\Delta\vdash G}(a_1, \ldots, a_{k-1}, a'_k)$ have the same domain and are equal.
- **(P3)** $\psi_{\Delta\vdash G}|_{\Delta'}$ and $\psi_{\Delta'\vdash G}$ have the same domain and are equal. ■

Lemma 139 ψ is monotone.

Proof. As in Lemma 129. ■

8.6.3 Recursive, small and simple systems

Let $G \in [F]$ and $\Delta \in \mathbb{E}$. To a sequence of arguments $\vec{a}$ for $J_{\Delta\vdash G}$, we associate the substitution $\theta = \{\vec{x} \mapsto \vec{t}\} : \Gamma_G \to \Delta_n$ and the assignment $\xi = \{\vec{x} \mapsto \vec{S}|_{\Delta_n}\}$ compatible with $(\theta, \Gamma_G, \Delta_n)$. Let $\mathcal{D}$ the set of sequences $\vec{a}$ such that $(\theta, \Gamma_G, \Delta_n)$ is valid w.r.t. ξ. We equip $\mathcal{D}$ with the following well-founded ordering : $\vec{a} \sqsupset \vec{a}'$ if $(G, \Delta_n, \theta, \xi) \sqsupset (G, \Delta'_n, \theta', \xi')$.

Definition 140 We define $J_{\Delta\vdash G}(\vec{a})$ by induction on $\sqsupset$:

$$J_{\Delta\vdash G}(\vec{a}) = \begin{cases} [\![\Gamma\vdash r]\!]^J_{\Delta_n,\sigma,\xi} & \text{if } \begin{array}{c}\{\vec{t}\} \subseteq \mathcal{WN}\cap\mathcal{CR},\ \vec{t}\!\downarrow = \vec{l}\sigma,\ \vec{a}\!\downarrow \in \mathcal{D}\\ \text{and } (G(\vec{l}) \to r, \Gamma, \rho) \in \mathcal{R}\end{array} \\ \top_{\Delta_n\vdash G(\vec{t})} & \text{otherwise}\end{cases}$$

where, for all $X \in \mathrm{FV}^{\Box}(r)$, $X\xi = S_{\kappa_X}|_{\Delta_n}$.

Lemma 141 J is a well defined interpretation.

Proof. By simplicity, at most one rule can be applied at the top of $G(\vec{t}\!\downarrow)$. The existence of κ_X is the smallness hypothesis. By **(S4)**, if $\vec{t} = \vec{l}\sigma$ then $\sigma : \Gamma \to \Delta_n$. Moreover, ξ is compatible with $(\sigma, \Gamma, \Delta_n)$ since, for all $X \in \mathrm{FV}^{\Box}(r)$, $X\xi = S_{\kappa_X}|_{\Delta_n} \in \mathcal{R}_{\Delta_n\vdash X\sigma}$ since $S_{\kappa_X} \in \mathcal{R}_{\Delta_{\kappa_X}\vdash t_{\kappa_X}}$ and, by smallness, $t_{\kappa_X} = l_{\kappa_X}\sigma = X\sigma$.

The well-foundedness of the definition comes from Lemma 147 and Theorem 146. In Lemma 147 for the reductibility of higher-order symbols, we show that, starting from a sequence $\vec{a} \in \mathcal{D}$, it is possible to apply Theorem 146 for the correctness of the computable closure. And in this theorem, we show that, in a recursive call $G'(\vec{a}')$ (case (symb$^=$)), we have $\vec{a} \sqsupset \vec{a}'$. Since $\sqsupset$ is compatible with $\to$, $\vec{a} \sqsupset \vec{a}'\!\downarrow$.

Finally, to make sure that J is an interpretation, one can proceed as in the case of a positive system. ■

8.7 Correctness of the conditions

Definition 142 (Cap and aliens) Let ζ be an injection from the classes of terms modulo $\leftrightarrow^*$ to $\mathcal{X}$. The *cap* of a term t w.r.t. a set $\mathcal{G}$ of symbols is the term $cap_{\mathcal{G}}(t) = t[x_1]_{p_1} \ldots [x_n]_{p_n}$ such that, for all i :

- $t|_{p_i}$ is not of the form $f(\vec{t})$ with $f \in \mathcal{G}$,
- $x_i = \zeta(t|_{p_i})$.

The $t|_{p_i}$'s are the *aliens* of t. We will denote by $aliens_{\mathcal{G}}(t)$ the multiset of the aliens of t.

Lemma 143 (Pre-reductibility of first-order symbols) Let $f \in \mathcal{F}_1$ and $\vec{t}$ be terms such that $f(\vec{t})$ is typable. If the t_i's are strongly normalizable then $f(\vec{t})$ is strongly normalizable.

Proof. We prove that every immediate reduct t' of $t = f(\vec{t})$ is strongly normalizable. Hereafter, by *cap*, we mean the cap w.r.t. $\mathcal{F}_1$.

Case $\mathcal{R}_\omega \neq \emptyset$. By induction on $(aliens(t), cap(t))_{\mathrm{lex}}$ with $((\to \cup \rhd)_{\mathrm{mul}}, \to_{\mathcal{R}_1})_{\mathrm{lex}}$ as well-founded ordering (the aliens are strongly normalizable and, by **(f)**, $\to_{\mathcal{R}_1}$ is strongly normalizing on $\mathbb{T}(\mathcal{F}_1, \mathcal{X})$).

If the reduction takes place in $cap(t)$ then this is a $\mathcal{R}_1$-reduction. By **(c)**, no symbol of $\mathcal{F}_\omega$ occurs in the rules of $\mathcal{R}_1$. Therefore, $cap(t) \to_{\mathcal{R}_1} cap(t')$. By **(d)**, the right hand-sides of the rules of $\mathcal{R}_1$ are algebraic and, by **(e)**, the rules of $\mathcal{R}_1$ are non duplicating. Therefore, $aliens(t) \rhd_{\mathrm{mul}} aliens(t')$ and we can conclude by induction hypothesis.

If the reduction takes place in an alien then $aliens(t)\ (\rightarrow \cup \rhd)_{\mathrm{mul}}\ aliens(t')$ and we can conclude by induction hypothesis.

Case $\mathcal{R}_\omega = \emptyset$. Since the t_i's are strongly normalizable and no β-reduction can take place at the top of t, t has a β-normal form. Let $\beta cap(t)$ be the cap of its β-normal form. We prove that every immediate reduct t' of t is strongly normalizable, by induction on $(\beta cap(t), aliens(t))_{\mathrm{lex}}$ with $(\rightarrow_{\mathcal{R}_1}, (\rightarrow \cup \rhd)_{\mathrm{mul}})_{\mathrm{lex}}$ as well-founded ordering (the aliens are strongly normalizable and, by **(f)**, $\rightarrow_{\mathcal{R}_1}$ is strongly normalizing on $\mathbb{T}(\mathcal{F}_1, \mathcal{X})$).

If the reduction takes place in $cap(t)$ then this is a $\mathcal{R}_1$-reduction. By **(d)**, the right hand-sides of the rules of $\mathcal{R}_1$ are algebraic. Therefore, t' has a β-normal form and $\beta cap(t) \rightarrow_{\mathcal{R}_1} \beta cap(t')$. Hence, we can conclude by induction hypothesis.

If the reduction is a β-reduction in an alien then $\beta cap(t) = \beta cap(t')$ and $aliens(t)$ $(\rightarrow \cup \rhd)_{\mathrm{mul}}$ $aliens(t')$. Hence, we can conclude by induction hypothesis.

We are left with the case where the reduction is a $\mathcal{R}_1$-reduction talking place in an alien u. Then, $aliens(t) \rightarrow_{\mathrm{mul}} aliens(t')$, $\beta cap(t) \rightarrow^*_{\mathcal{R}_1} \beta cap(t')$ and we can conclude by induction hypothesis. To see that $\beta cap(t) \rightarrow^*_{\mathcal{R}_1} \beta cap(t')$, it suffices to remark the following : if we β-normalize u, all the residuals of the $\mathcal{R}_1$-redex are still reductible since, by **(c)**, no symbol of $\mathcal{F}_\omega$ occurs in the rules of $\mathcal{R}_1$. ■

Theorem 144 (Strong normalization of $\rightarrow_{\mathcal{R}}$) The relation $\rightarrow_{\mathcal{R}_1} \cup \rightarrow_{\mathcal{R}_\omega}$ is strongly normalizing on typable terms.

Proof. By induction on the structure of terms. The only difficult case is $f(\vec{t})$. If f is first-order, we use the Lemma of pre-reductibility of first-order symbols. If f is higher-order, we have to show that if $\vec{t} \in \mathcal{SN}$ and $f(\vec{t}) \in \mathbb{T}$ then $t = f(\vec{t}) \in \mathcal{SN}$ where $\mathcal{SN}$ means here the strong normalization w.r.t. $\rightarrow_{\mathcal{R}} = \rightarrow_{\mathcal{R}_1} \cup \rightarrow_{\mathcal{R}_\omega}$.

We prove that every immediate reduct t' of t is strongly normalizable by induction on $(f, \varpi(\vec{t}), \vec{t}, \vec{t})$ with $(>_{\mathcal{F}}, (>_{\mathbb{N}})_{stat_f}, (\rhd \cup \rightarrow_{\mathcal{R}})_{stat_f}, (\rightarrow_{\mathcal{R}})_{\mathrm{lex}})_{\mathrm{lex}}$ as well-founded ordering where $\varpi(t) = 0$ if t is not of the form $g(\vec{u})$ and $\varpi(t) = 1$ otherwise. Assume that $t' = f(\vec{t'})$ with $t_i \rightarrow_{\mathcal{R}} t'_i$ and, for all $j \neq i$, $t_j = t'_j$. Then, $\vec{t}\ (\rightarrow_{\mathcal{R}})_{\mathrm{lex}}\ \vec{t'}$ and $\varpi(t_i) \geq \varpi(t'_i)$ since if t_i is not of the form $g(\vec{u})$ then t'_i is not of the form $g(\vec{u})$ either.

Assume now that there exists $f(\vec{l}) \rightarrow r \in \mathcal{R}_\omega$ such that $\vec{t} = \vec{l}\sigma$ and $t' = r\sigma$. By **(a)**, r belongs to the computable closure of l. It is then easy to prove that $r\sigma$ is strongly normalizable by induction on the structure of r. Again, the only difficult case is $g(\vec{u})$. But, either g is smaller than f, or g is equivalent to f and its arguments are smaller than $\vec{l}$. If $l_i >_1 u_j$ then $l_i \rhd u_j$ and $\mathrm{FV}(u_j) \subseteq \mathrm{FV}(l_i)$. Therefore $l_i\sigma \rhd u_j\sigma$ and $\varpi(l_i\sigma) = 1 \geq \varpi(u_j\sigma)$. If now $l_i >_2 u_j$ then u_j is of the form $x\vec{v}$ and $\varpi(l_i\sigma) = 1 > \varpi(u_j\sigma) = 0$. ■

Lemma 145 (Reductibility of first-order symbols) Let $f \in \mathcal{F}_1$, $\tau_f = (\vec{x} : \vec{T})U$, $\Delta \in \mathbb{E}$, $\theta : \Gamma_f \rightarrow \Delta$ and ξ compatible with $(\theta, \Gamma_f, \Delta)$. If $(\theta, \Gamma_f, \Delta)$ is valid w.r.t. ξ then $\Delta \vdash f(\vec{x}\theta) \in [\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi}$.

Proof. Let $t_i = x_i\theta$ and $t = f(\vec{t})$. If f is not a constructor then t is neutral and it suffices to prove that, for every immediate reduct t' of t, $\Delta \vdash t' \in [\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi}$.

If f is a constructor then $U = C(\vec{v})$ and $[\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi} = I_{\Delta \vdash C}(\vec{a})$ where $a_i = \Delta \vdash v_i\theta$ if $x_i \in \mathcal{X}^\star$ and $a_i = (\Delta \vdash v_i\theta, S'_i)$ with $S'_i = [\![\Gamma_f \vdash v_i]\!]_{\Delta,\theta,\xi}$ if $x_i \in \mathcal{X}^\Box$. Let $j \in \mathrm{Acc}(f)$. Since θ is valid w.r.t. ξ, $\Delta \vdash t_j \in [\![\Gamma_f \vdash T_j]\!]_{\Delta,\theta,\xi}$. Therefore, in this case too, it suffices to prove that, for every immediate reduct t' of t, $\Delta \vdash t' \in [\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi}$.

For first-order symbols, $U = \star$ or $U = C(\vec{v})$ with C a primitive constant predicate symbol. Therefore $[\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi} = \overline{\mathbb{SN}}_{\Delta \vdash U\theta}$ and it suffices to prove that every immediate reduct t' of t is strongly normalizable. This is the Lemma of pre-reductibility of first-order symbols. ■

Theorem 146 (Correctness of the computable closure) Let $f \in \mathcal{F}$, $\tau_f = (\vec{x} : \vec{T})U$, $R = (f(\vec{l}) \to r, \Gamma_0, \rho) \in \mathcal{R}$, $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$, $\Delta \in \mathbb{E}$, $\sigma : \Gamma_0 \to \Delta$ and ξ compatible with $(\sigma, \Gamma_0, \Delta)$ such that :

- $(\sigma, \Gamma_0, \Delta)$ is valid w.r.t. ξ;
- for all i, $\Delta \vdash l_i\sigma \in [\![\Gamma_0 \vdash T_i\gamma_0\rho]\!]_{\Delta,\sigma,\xi}$;
- for all $g \leq_{\mathcal{F}} f$, if $\tau_g = (\vec{y} : \vec{U})V$, $\Delta' \in \mathbb{E}$, $\theta' : \Gamma \to \Delta$, ξ' is compatible with $(\theta', \Gamma_g, \Delta')$ and $(\theta', \Gamma_g, \Delta')$ is valid w.r.t. ξ', then $\Delta' \vdash g(\vec{y})\theta' \in [\![\Gamma_g \vdash V]\!]_{\Delta',\theta',\xi|_{\Delta'}}$ whenever $(f, \Delta, \gamma_0\sigma, \xi) \sqsupset (g, \Delta', \theta', \xi')$.

If $\Gamma_0, \Gamma \vdash_c t : T$, $\Delta \subseteq \Delta' \in \mathbb{E}$, $\sigma' : \Gamma \to \Delta'$, ξ' is compatible with $(\sigma', \Gamma, \Delta')$ and $(\sigma', \Gamma, \Delta')$ is valid w.r.t. ξ', then $\Delta' \vdash t\sigma\sigma' \in [\![\Gamma_0, \Gamma \vdash T]\!]_{\Delta',\sigma\sigma',\xi''}$ where $\xi'' = \xi|_{\Delta'} \cup \xi'$.

Proof. By induction on $\Gamma' \vdash_c t : T$ ($\Gamma' = \Gamma_0, \Gamma$), we prove :

(a) $\Delta' \vdash t\sigma\sigma' \in [\![\Gamma' \vdash T]\!]_{\Delta',\sigma\sigma',\xi''}$,
(b) if $t \notin \mathcal{O}$ and $t \to t'$ then $[\![\Gamma' \vdash t]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash t']\!]_{\Delta',\sigma\sigma',\xi''}$.

(ax) $\Gamma_0 \vdash_c \star : \Box$

(a) $\Delta' \vdash \star \in [\![\Gamma_0 \vdash \Box]\!]_{\Delta',\sigma,\xi|_{\Delta'}} = \overline{\mathbb{SN}}_{\Delta' \vdash \Box}$.
(b) $\star$ is not reductible.

(symb$^<$) $\dfrac{\Gamma_0 \vdash_c \tau_g : s \quad \Gamma' \vdash_c u_1 : U_1\gamma \ldots \Gamma' \vdash_c u_n : U_n\gamma}{\Gamma' \vdash_c g(\vec{u}) : V\gamma}$

(a) By induction hypothesis, $\Delta' \vdash u_i\sigma\sigma' \in [\![\Gamma' \vdash U_i\gamma]\!]_{\Delta',\sigma\sigma',\xi''}$. By candidates substitution, there exists ξ''' such that $[\![\Gamma' \vdash U_i\gamma]\!]_{\Delta,\theta,\xi} = [\![\Gamma_g \vdash U_i]\!]_{\Delta',\gamma\sigma\sigma',\xi'''}$ and $[\![\Gamma' \vdash V\gamma]\!]_{\Delta,\theta,\xi} = [\![\Gamma_g \vdash V]\!]_{\Delta',\gamma\sigma\sigma',\xi'''}$. Hence, $(\gamma\sigma\sigma', \Gamma_g, \Delta')$ is valid w.r.t. ξ''' and, by hypothesis on g, $\Delta' \vdash g(\vec{y})\gamma\sigma\sigma' \in [\![\Gamma_g \vdash V]\!]_{\Delta',\gamma\sigma\sigma',\xi'''}$.

(b) We have $[\![\Gamma' \vdash g(\vec{u})]\!]_{\Delta',\sigma\sigma',\xi''} = I_{\Delta' \vdash g}(\vec{a})$ with $a_i = \Delta' \vdash u_i\sigma\sigma'$ if $y_i \in \mathcal{X}^\star$, et $a_i = (\Delta' \vdash u_i\sigma\sigma', S_i)$ where $S_i = [\![\Gamma' \vdash u_i]\!]_{\Delta',\sigma\sigma',\xi''}$ if $y_i \in \mathcal{X}^\Box$. There is two cases :

- $u_i \to u'_i$. We conclude by **(P1)**.
- $\vec{u} = \vec{l}'\sigma''$ and $(g(\vec{l}') \to r', \Gamma'', \rho') \in \mathcal{R}$. Let $\sigma''' = \sigma''\sigma\sigma'$. By **(A3)**, there is two sub-cases :
  - g **belongs to a primitive system.** Then $I_{\Delta' \vdash g} = \top_{\Delta' \vdash g}$ and $[\![\Gamma' \vdash g(\vec{u})]\!]_{\Delta',\sigma\sigma',\xi''} = \top_{\Delta \vdash g(\vec{l}'\sigma''')} = \top_{\Delta \vdash r'\sigma'''}$. By candidates substitution, there exists ξ''' such that $[\![\Gamma' \vdash r'\sigma'']\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma'' \vdash r']\!]_{\Delta',\sigma''',\xi'''}$. Moreover, r' is of the form $[\vec{x} : \vec{T}]g'(\vec{u})\vec{v}$ with $g' \simeq g$ or g' a primitive constant predicate

symbol. If $g' \simeq g$ then $I_{\Delta'\vdash g'} = \top_{\Delta'\vdash g'}$. If g' is a primitive constant predicate symbol then, after Lemma 130, $I_{\Delta'\vdash g'} = \top_{\Delta'\vdash g'}$. Therefore, $[\![\Gamma''\vdash r']\!]_{\Delta',\sigma''',\xi'''} = \top_{\Delta'\vdash r'\sigma'''}$ and $[\![\Gamma'\vdash g(\vec{l'}\sigma'')]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma'\vdash r'\sigma'']\!]_{\Delta',\sigma\sigma',\xi''}$.

- g **belongs to a positive or recursive, small and simple system.** Since $\Delta'\vdash u_i\sigma\sigma' \in [\![\Gamma'\vdash U_i\gamma]\!]_{\Delta',\sigma\sigma',\xi''} \subseteq \mathcal{SN}$, by **(A1)**, $u_i\sigma\sigma'$ has a normal form. By simplicity the symbols in $\vec{l'}$ are constant. Therefore $u_i\sigma\sigma'$ is of the form $\vec{l'}\sigma'''$ with $\sigma''\sigma\sigma' \to^* \sigma'''$. By simplicity again, at most one rule can be applied at the top of a term. Therefore $I_{\Delta'\vdash g}(\vec{a}) = [\![\Gamma''\vdash r']\!]_{\Delta',\sigma''',\xi'''}$ with, for all $X \in \mathrm{FV}^\square(r')$, $X\xi''' = S_{\kappa_X} = [\![\Gamma'\vdash l_{\kappa_X}\sigma'']\!]_{\Delta',\sigma\sigma',\xi''}$. By smallness, $l_{\kappa_X} = X$. Therefore $X\xi''' = [\![\Gamma'\vdash X\sigma'']\!]_{\Delta',\sigma\sigma',\xi''}$ and, by candidate substitution, $[\![\Gamma''\vdash r']\!]_{\Delta',\sigma''',\xi'''} = [\![\Gamma'\vdash r'\sigma'']\!]_{\Delta',\sigma\sigma',\xi''}$.

$$\textbf{(symb}^=\textbf{)}\ \frac{\Gamma_0 \vdash_{\mathrm{c}} \tau_g : s \quad \Gamma' \vdash_{\mathrm{c}} u_1 : U_1\gamma \ldots \Gamma' \vdash_{\mathrm{c}} u_n : U_n\gamma}{\Gamma' \vdash_{\mathrm{c}} g(\vec{u}) : V\gamma} \quad (\vec{l} : \vec{T}\gamma_0 > \vec{u} : \vec{U}\gamma)$$

(a) Like in the previous case, we have $(\gamma\sigma\sigma', \Gamma_g, \Delta')$ valid w.r.t. ξ'''. Let us prove that $(f, \Delta, \gamma_0\sigma, \xi) \sqsupset_f (g, \Delta', \gamma\sigma\sigma', \xi''')$. To this end, it suffices to prove that, if $l_i : T_i\gamma_0 >_1 u_j : U_j\gamma$ then $l_i\sigma \rhd u_j\sigma\sigma'$, and if $l_i : T_i\gamma_0 >_2 u_j : U_j\gamma$ then $o(\Delta\vdash l_i\sigma) > o(\Delta'\vdash u_j\sigma\sigma')$.

Assume that $l_i : T_i\gamma_0 >_1 u_j : U_j\gamma$. Then, $l_i \rhd u_j$ and $\mathrm{FV}(u_j) \subseteq \mathrm{FV}(l_i)$. Therefore, $l_i\sigma\sigma' = l_i\sigma \rhd u_j\sigma\sigma'$.

Assume now that $l_i : T_i\gamma_0 >_2 u_j : U_j\gamma$. By definition of $>_2$, we have $u_j = x\vec{v}$, $x \in \mathrm{dom}(\Gamma_0)$, $x\Gamma_0 = (\vec{z} : \vec{V})W$. Let $\theta = \{\vec{z} \mapsto \vec{v}\}$. By correctness of the ordering on the arguments, it suffices to check that :

(1) $\Gamma' \vdash l_i\rho : T_i\gamma_0\rho$,
(2) $\Delta'\vdash l_i\sigma\sigma' \in [\![\Gamma'\vdash T_i\gamma_0\rho]\!]_{\Delta',\sigma\sigma',\xi''}$,
(3) for all k, $\Delta'\vdash v_k\sigma\sigma' \in [\![\Gamma'\vdash V_k\theta]\!]_{\Delta',\sigma\sigma',\xi''}$.

Indeed, from this, we can deduce that $o(\Delta'\vdash l_i\sigma\sigma') > o(\Delta'\vdash u_j\sigma\sigma')$. But $l_i\sigma = l_i\sigma\sigma'$ and $o(\Delta\vdash l_i\sigma) \geq o(\Delta'\vdash l_i\sigma)$.

(1) By definition of a well-formed rule, $\Gamma_0 \vdash f(\vec{l})\rho : U\gamma_0\rho$. Therefore, by inversion, $\Gamma_0 \vdash l_i\rho : T_i\gamma_0\rho$. Since $\Gamma_0 \subseteq \Gamma' \in \mathbb{E}$, by weakening, $\Gamma' \vdash l_i\rho : T_i\gamma_0\rho$.

(2) By hypothesis, $\Delta \vdash l_i\sigma \in [\![\Gamma_0 \vdash T_i\gamma_0\rho]\!]_{\Delta,\sigma,\xi}$. But $l_i\sigma\sigma' = l_i\sigma$ and $[\![\Gamma_0\vdash T_i\gamma_0\rho]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma_0\vdash T_i\gamma_0\rho]\!]_{\Delta',\sigma,\xi|_{\Delta'}} = [\![\Gamma_0\vdash T_i\gamma_0\rho]\!]_{\Delta,\sigma,\xi}|_{\Delta'}$.

(3) By hypothesis, we have $\Gamma' \vdash_{\mathrm{c}} x\vec{v} : U_j\gamma$. Let $q = |\vec{v}|$. By inversion, there exists $\vec{V'}$ and $\vec{W'}$ such that $\Gamma' \vdash_{\mathrm{c}} xv_1\ldots v_{q-1} : (x_q : V'_q)W'_q$, $\Gamma' \vdash_{\mathrm{c}} v_q : V'_q$ and $W'_q\{x_q \mapsto v_q\}\ \mathbb{C}^*_{\Gamma'}\ U_j\gamma$ and, for all $k < q-1$, $\Gamma' \vdash_{\mathrm{c}} xv_1\ldots v_k : (x_{k+1} : V'_{k+1})W'_{k+1}$, $\Gamma' \vdash_{\mathrm{c}} v_{k+1} : V'_{k+1}$ and $W'_{k+1}\{x_{k+1} \mapsto v_{k+1}\}\ \mathbb{C}^*_{\Gamma'}\ (x_{k+2} : V'_{k+2})W'_{k+2}$. Hence, by induction hypothesis, for all k, $\Delta'\vdash v_k\sigma\sigma' \in [\![\Gamma'\vdash V'_k]\!]_{\Delta',\sigma\sigma',\xi''}$. We now show that $V'_k\ \mathbb{C}^*_{\Gamma'}\ V_k\theta$. Let $W_k = (z_{k+1} : V_{k+1})\ldots(z_q : V_q)W$ and $\theta_k = \{z_1 \mapsto v_1, \ldots, z_{k-1} \mapsto v_{k-1}\}$. We prove by induction on k that $V'_k\ \mathbb{C}^*_{\Gamma'}\ V_k\theta_{k-1}$ and $W'_k\{x_k \mapsto v_k\}\ \mathbb{C}^*_{\Gamma'}\ W_k\theta_k$.

– Case $k = 1$. Since $(\vec{z} : \vec{V})W\ \mathbb{C}^*_{\Gamma'}\ (x_1 : V'_1)W'_1$, by product compatibility and α-equivalence (we take $x_1 = z_1$), $V'_1\ \mathbb{C}^*_{\Gamma'}\ V_1$ and $W'_1\ \mathbb{C}^*_{\Gamma',z_1:V'_1}\ W_1$. Therefore, $W'_1\{x_1 \mapsto v_1\}\ \mathbb{C}^*_{\Gamma'}\ W_1\theta_1$.

– Case $k > 1$. By induction hypothesis, we have $W'_{k-1}\{x_{k-1} \mapsto v_{k-1}\}\ \mathbb{C}^*_{\Gamma'}$

$W_{k-1}\theta_{k-1}$. But $W'_{k-1}\{x_{k-1} \mapsto v_{k-1}\} \ \mathbb{C}^*_{\Gamma'} \ (x_k : V'_k)W'_k$ and $W_{k-1}\theta_{k-1} = (z_k : V_k\theta_{k-1})W_k\theta_{k-1}$. By product compatibility and α-equivalence (we take $x_k = z_k$), $V'_k \ \mathbb{C}^*_{\Gamma'} \ V_k\theta_{k-1}$ and $W'_k \ \mathbb{C}^*_{\Gamma',z_k:V'_k} \ W_k\theta_{k-1}$. Therefore, $W'_k\{x_k \mapsto v_k\} \ \mathbb{C}^*_{\Gamma'} \ W_k\theta_k$.

In conclusion, $\Delta' \vdash v_k\sigma\sigma' \in [\![\Gamma' \vdash V'_k]\!]_{\Delta',\sigma\sigma',\xi''}$ and $V'_k \ \mathbb{C}^*_{\Gamma'} \ V_k\theta_k = V_k\theta$. By (conv'), the types used in a conversion are typable. Therefore, we can apply the induction hypothesis (b). Hence, $[\![\Gamma' \vdash V'_k]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash V_k\theta]\!]_{\Delta',\sigma\sigma',\xi''}$ and $\Delta' \vdash v_k\sigma\sigma' \in [\![\Gamma' \vdash V_k\theta]\!]_{\Delta',\sigma\sigma',\xi''}$.

(b) Like for (symb$^<$).

(acc) $\dfrac{\Gamma_0 \vdash_c x\Gamma_0 : s}{\Gamma_0 \vdash_c x : x\Gamma_0}$

(a) Since $(\sigma, \Gamma_0, \Delta')$ is valid w.r.t. $\xi|_{\Delta'}$.

(b) x is irreductible.

(var) $\dfrac{\Gamma' \vdash_c T : s}{\Gamma', x : T \vdash_c x : T}$

(a) Car $(\sigma\sigma', \Gamma', \Delta')$ is valid w.r.t. ξ''.

(b) x is irreductible.

(weak) $\dfrac{\Gamma' \vdash_c t : T \quad \Gamma' \vdash_c U : s}{\Gamma', x : U \vdash_c t : T}$

(a) By induction hypothesis, $\Delta' \vdash t\sigma\sigma' \in [\![\Gamma' \vdash T]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma_0, \Gamma, x : U \vdash T]\!]_{\Delta',\sigma\sigma',\xi''}$.

(b) Since $x \notin \mathrm{FV}(t)$, $[\![\Gamma', x : U \vdash t]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash t]\!]_{\Delta',\sigma\sigma',\xi''}$. But, by induction hypothesis, $[\![\Gamma' \vdash t]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash t']\!]_{\Delta',\sigma\sigma',\xi''}$.

(prod) $\dfrac{\Gamma', x : T \vdash_c U : s}{\Gamma' \vdash_c (x : T)U : s}$

(a) We have to prove that $\Delta' \vdash (x : T\sigma\sigma')U\sigma\sigma' \in [\![\Gamma' \vdash s]\!]_{\Delta',\sigma\sigma',\xi''} = \overline{\mathbb{SN}}_{\Delta' \vdash s}$. By inversion, $\Gamma' \vdash T : s'$. By induction hypothesis, $\Delta' \vdash T\sigma\sigma' \in [\![\Gamma' \vdash s']\!]_{\Delta',\sigma\sigma',\xi''}$. We now show that $U\sigma\sigma' \in \mathcal{SN}$. We can always assume that $x \notin \mathrm{dom}(\Delta')$. Then, $\sigma\sigma' : \Gamma', x : T \to \Delta''$ where $\Delta'' = \Delta, x : T\sigma\sigma'$. Let $\xi''' = \xi''|_{\Delta''}$ if $x \in \mathcal{X}^\star$, and $\xi''' = \xi''|_{\Delta''} \cup \{x \mapsto \top_{\Delta'' \vdash x}\}$ if $x \in \mathcal{X}^\Box$. Then, ξ''' is compatible with $(\sigma\sigma', \Gamma', x : T, \Delta'')$ since, if $x \in \mathcal{X}^\Box$ then $x\xi''' \in \mathcal{R}_{\Delta'' \vdash x} = \mathcal{R}_{\Delta' \vdash x\sigma\sigma'}$ and, for all $X \in \mathrm{dom}^\Box(\Gamma')$, $X\xi''' = X\xi'' \in \mathcal{R}_{\Delta' \vdash X\sigma\sigma'}$ since ξ'' is compatible with $(\sigma\sigma', \Gamma', \Delta')$. Moreover, $(\sigma\sigma', \Gamma', x : T, \Delta'')$ is valid w.r.t. ξ''' since, by Lemma 115, $\Delta'' \vdash x\sigma\sigma' = \Delta'' \vdash x \in [\![\Gamma', x : T \vdash T]\!]_{\Delta'',\sigma\sigma',\xi'''}$. Therefore, by induction hypothesis, $\Delta'' \vdash U\sigma\sigma' \in [\![\Gamma', x : T \vdash s']\!]_{\Delta'',\sigma\sigma',\xi'''} = \overline{\mathbb{SN}}_{\Delta'' \vdash s'}$.

(b) There is two cases :

– $T \to T'$. We prove that $[\![\Gamma' \vdash (x : T)U]\!]_{\Delta',\sigma\sigma',\xi''} \subseteq [\![\Gamma' \vdash (x : T')U]\!]$. The other way around is similar. Let $\Delta'' \vdash u \in [\![\Gamma' \vdash (x : T)U]\!]_{\Delta',\sigma\sigma',\xi''}$, $\Delta''' \vdash t \in [\![\Gamma' \vdash T']\!]_{\Delta'',\sigma\sigma',\xi''|_{\Delta''}}$ and $\sigma'' = \sigma\sigma' \cup \{x \mapsto t\}$. By induction hypothesis, $[\![\Gamma' \vdash T]\!]_{\Delta'',\sigma\sigma',\xi''|_{\Delta''}} = [\![\Gamma' \vdash T']\!]_{\Delta'',\sigma\sigma',\xi''|_{\Delta''}}$. Therefore, $\Delta''' \vdash t \in [\![\Gamma' \vdash T]\!]_{\Delta'',\sigma\sigma',\xi''|_{\Delta''}}$, $\Delta''' \vdash ut \in [\![\Gamma', x : T \vdash U]\!]_{\Delta''',\sigma'',\xi''|_{\Delta'''}}$ and $\Delta'' \vdash u \in [\![\Gamma' \vdash (x : T')U]\!]_{\Delta',\sigma\sigma',\xi''}$.

– $U \to U'$. We prove that $[\![\Gamma' \vdash (x:T)U]\!]_{\Delta',\sigma\sigma',\xi''} \subseteq [\![\Gamma' \vdash (x:T)U']\!]$. The other way around is similar. Let $\Delta'' \vdash u \in [\![\Gamma' \vdash (x:T)U]\!]_{\Delta',\sigma\sigma',\xi''}$, $\Delta''' \vdash t \in [\![\Gamma' \vdash T]\!]_{\Delta'',\sigma\sigma',\xi''|_{\Delta''}}$ and $\sigma'' = \sigma\sigma' \cup \{x \mapsto t\}$. Then, $\Delta''' \vdash ut \in [\![\Gamma', x:T \vdash U]\!]_{\Delta''',\sigma'',\xi''|_{\Delta'''}}$. By induction hypothesis, $[\![\Gamma', x:T \vdash U]\!]_{\Delta''',\sigma'',\xi''|_{\Delta'''}} = [\![\Gamma', x:T \vdash U']\!]_{\Delta''',\sigma'',\xi''|_{\Delta'''}}$. Therefore, $\Delta'' \vdash ut \in [\![\Gamma', x:T \vdash U']\!]_{\Delta''',\sigma'',\xi''|_{\Delta'''}}$ and $\Delta'' \vdash u \in [\![\Gamma' \vdash (x:T)U']\!]_{\Delta',\sigma\sigma',\xi''}$.

(abs) $\dfrac{\Gamma', x:T \vdash_c u : U \quad \Gamma' \vdash_c (x:T)U : s}{\Gamma' \vdash_c [x:T]u : (x:T)U}$

(a) Let $\Gamma'' = \Gamma', x:T$. We have to prove that $\Delta' \vdash [x:T\sigma\sigma']u\sigma\sigma' \in [\![\Gamma' \vdash (x:T)U]\!]_{\Delta',\sigma\sigma',\xi''}$. Let $\Delta'' \vdash t \in [\![\Gamma' \vdash T]\!]_{\Delta',\sigma\sigma',\xi''}$ and $S \in \mathcal{R}_{\Delta'' \vdash t}$ if $x \in \mathcal{X}^\Box$. Let $\sigma'' = \sigma\sigma' \cup \{x \mapsto t\}$, $\xi''' = \xi''|_{\Delta''}$ if $x \in \mathcal{X}^\star$ and $\xi''' = \xi''|_{\Delta''} \cup \{x \mapsto S\}$ if $x \in \mathcal{X}^\Box$. We have $\sigma'' : \Gamma'' \to \Delta''$, ξ''' compatible with $(\sigma'', \Gamma'', \Delta'')$ and $(\sigma'', \Gamma'', \Delta'')$ valid w.r.t. ξ''. Therefore, by induction hypothesis, $\Delta'' \vdash u\sigma'' \in R = [\![\Gamma'' \vdash U]\!]_{\Delta'',\sigma'',\xi'''}$. Then, for proving that $v = [x:T\sigma\sigma']u\sigma\sigma'\ t \in R$, it suffices to prove that $T\sigma\sigma', u\sigma\sigma' \in \mathcal{SN}$. Indeed, in this case, it is easy to prove that every immediate reduct of the neutral term v belongs to R by induction on $(T\sigma\sigma', u\sigma\sigma', t)$ with $\to_{\mathrm{lex}}$ with well-founded ordering. By induction hypothesis, we have $\Delta' \vdash (x:T\sigma\sigma')U\sigma\sigma' \in [\![\Gamma' \vdash s]\!]_{\Delta',\sigma\sigma',\xi''}$. Therefore, $T\sigma\sigma' \in \mathcal{SN}$. Finally, if we take $t = x$, which is possible after Lemma 115, we have seen that, by induction hypothesis, $u\sigma'' = u\sigma\sigma' \in \mathcal{SN}$.

(b) There is two cases :

– $T \to T'$. Since $\overline{\mathbb{T}}_{\Delta' \vdash T\sigma\sigma'} = \overline{\mathbb{T}}_{\Delta' \vdash T'\sigma\sigma'}$, $[\![\Gamma' \vdash [x:T]u]\!]_{\Delta',\sigma\sigma',\xi''}$ and $[\![\Gamma' \vdash [x:T']u]\!]$ have the same domain and are equal.

– $u \to u'$. Let $\Delta'' \vdash t \in \overline{\mathbb{T}}_{\Delta' \vdash T\sigma\sigma'}$, $S \in \mathcal{R}_{\Delta'' \vdash t}$ if $x \in \mathcal{X}^\Box$, $\sigma'' = \sigma\sigma' \cup \{x \mapsto t\}$, $\xi''' = \xi''|_{\Delta''}$ if $x \in \mathcal{X}^\star$ and $\xi''' = \xi''|_{\Delta''} \cup \{x \mapsto S\}$ if $x \in \mathcal{X}^\Box$. By induction hypothesis, $[\![\Gamma', x:T \vdash u]\!]_{\Delta'',\sigma'',\xi'''} = [\![\Gamma', x:T \vdash u']\!]_{\Delta'',\sigma'',\xi'''}$. Therefore, $[\![\Gamma' \vdash [x:T]u]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash [x:T]u']\!]_{\Delta',\sigma\sigma',\xi''}$.

(app) $\dfrac{\Gamma' \vdash_c t : (x:U)V \quad \Gamma' \vdash_c u : U}{\Gamma' \vdash_c tu : V\{x \mapsto u\}}$

(a) By induction hypothesis, $\Delta' \vdash t\sigma\sigma' \in [\![\Gamma' \vdash (x:U)V]\!]_{\Delta',\sigma\sigma',\xi''}$ and $\Delta' \vdash u\sigma\sigma' \in [\![\Gamma' \vdash U]\!]_{\Delta',\sigma\sigma',\xi''}$. Let $S = [\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''}$ if $x \in \mathcal{X}^\Box$. Then, by definition of $[\![\Gamma' \vdash (x:U)V]\!]_{\Delta',\sigma\sigma',\xi''}$, $\Delta' \vdash t\sigma\sigma' u\sigma\sigma' \in R = [\![\Gamma', x:U \vdash V]\!]_{\Delta',\sigma'',\xi'''}$ where $\sigma'' = \sigma\sigma' \cup \{x \mapsto u\sigma\sigma'\}$, $\xi''' = \xi''$ if $x \in \mathcal{X}^\star$, and $\xi''' = \xi'' \cup \{x \mapsto S\}$ if $x \in \mathcal{X}^\Box$. By candidates substitution, $R = [\![\Gamma' \vdash V\{x \mapsto u\}]\!]_{\Delta',\sigma\sigma',\xi''}$.

(b) There is three cases :

– $t \to t'$. By induction hypothesis, $[\![\Gamma' \vdash t]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash t']\!]_{\Delta',\sigma\sigma',\xi''}$. Therefore, $[\![\Gamma' \vdash tu]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash t]\!]_{\Delta',\sigma\sigma',\xi''}(\Delta' \vdash u\sigma\sigma', [\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''}) = [\![\Gamma' \vdash t']\!](\Delta' \vdash u\sigma\sigma', [\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''}) = [\![\Gamma' \vdash t'u]\!]_{\Delta',\sigma\sigma',\xi''}$.

– $u \to u'$. By induction hypothesis, $[\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash u']\!]_{\Delta',\sigma\sigma',\xi''}$. By **(P1)**, $[\![\Gamma' \vdash t]\!](\Delta' \vdash u\sigma\sigma', [\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''}) = [\![\Gamma' \vdash t]\!](\Delta' \vdash u'\sigma\sigma', [\![\Gamma' \vdash u']\!]_{\Delta',\sigma\sigma',\xi''})$. Therefore, $[\![\Gamma' \vdash tu]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash tu']\!]_{\Delta',\sigma\sigma',\xi''}$.

– $t = [x:U']v$ and $t' = v\{x \mapsto u\}$. Let $\sigma'' = \{x \mapsto u\}\sigma\sigma'$ and $\xi''' = \xi'' \cup \{x \mapsto [\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''}\}$. By candidates substitution, $[\![\Gamma' \vdash t']\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma', x:U' \vdash v]\!]_{\Delta',\sigma'',\xi'''}$. On the other hand, $[\![\Gamma' \vdash tu]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash$

$t]\!]_{\Delta',\sigma\sigma',\xi''}(\Delta' \vdash u\sigma\sigma', [\![\Gamma' \vdash u]\!]_{\Delta',\sigma\sigma',\xi''}) = [\![\Gamma', x:U' \vdash v]\!]_{\Delta',\sigma\sigma'\cup\{x\mapsto u\sigma\sigma'\},\xi'''} = J\Gamma', x\!:\!U'v_{\Delta',\sigma'',\xi'''}$.

$$\textbf{(conv)}\ \frac{\Gamma' \vdash_c t : T \quad \Gamma' \vdash_c T : s \quad T \downarrow T' \quad \Gamma' \vdash_c T' : s}{\Gamma' \vdash_c t : T'}$$

(a) Let U be the common reduct of T and T'. By induction hypothesis, $\Delta' \vdash t\sigma\sigma' \in [\![\Gamma' \vdash T]\!]_{\Delta',\sigma\sigma',\xi''}$, $[\![\Gamma' \vdash T]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash U]\!]_{\Delta',\sigma\sigma',\xi''}$ and $[\![\Gamma' \vdash T']\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash U]\!]_{\Delta',\sigma\sigma',\xi''}$. Therefore, $[\![\Gamma' \vdash T]\!]_{\Delta',\sigma\sigma',\xi''} = [\![\Gamma' \vdash T']\!]_{\Delta',\sigma\sigma',\xi''}$ and $\Delta' \vdash t\sigma\sigma' \in [\![\Gamma' \vdash T']\!]_{\Delta',\sigma\sigma',\xi''}$.

(b) By induction hypothesis. ■

Lemma 147 (Reductibility of higher-order symbols) Let $f \in \mathcal{F}_\omega$, $\tau_f = (\vec{x} : \vec{T})U$, $\Delta \in \mathbb{E}$, $\theta : \Gamma_f \to \Delta$ and ξ compatible with $(\theta, \Gamma_f, \Delta)$. If $(\theta, \Gamma_f, \Delta)$ is valid w.r.t. ξ then $\Delta \vdash f(\vec{x})\theta \in [\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi}$.

Proof. By induction on $((f, \Delta, \theta, \xi), \vec{x}\theta)$ with $(\sqsupset, \to)_{\text{lex}}$ as well-founded ordering.

Let $t_i = x_i\theta$ and $t = f(\vec{t})$. Like in the Lemma of reductibility of first-order symbols, it suffices to prove that, for every immediate reduct t' of t, $\Delta \vdash t' \in [\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi}$.

If the reduction takes place one t_i then we can conclude by induction hypothesis since reductibility candidates are stable by reduction **(R2)** and $\sqsupset$ is compatible with the reduction.

Assume now that there exists a rule $(l \to r, \Gamma_0, \rho)$ and a substitution σ such that $t = l\sigma$. Assume also that $l = f(\vec{l})$ and $\gamma_0 = \{\vec{x} \mapsto \vec{l}\}$. Then, $\theta = \gamma_0\sigma$. By **(S5)**, for all $x_i \in \mathrm{FV}^\square(\vec{T}U)$, $x_i\gamma_0\sigma \downarrow x_i\gamma_0\rho\sigma$.

We prove that $[\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi} = [\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi}$ and $[\![\Gamma_f \vdash T_i]\!]_{\Delta,\theta,\xi} = [\![\Gamma_f \vdash T_i]\!]_{\Delta,\gamma_0\rho\sigma,\xi}$. By **(S4)**, $\sigma : \Gamma_0 \to \Delta$. By definition of a well-formed rule, $\Gamma_0 \vdash l\rho : U\gamma_0\rho$. Therefore, by inversion, $\gamma_0\rho : \Gamma_f \to \Gamma_0$ and $\gamma_0\rho\sigma : \Gamma_f \to \Delta$. We now prove that ξ is compatible with $(\gamma_0\rho\sigma, \Gamma_f, \Delta)$. Then, after Lemma 121 (c), $[\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi} = [\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi}$ and $[\![\Gamma_f \vdash T_i]\!]_{\Delta,\theta,\xi} = [\![\Gamma_f \vdash T_i]\!]_{\Delta,\gamma_0\rho\sigma,\xi}$. Let $x_i \in \mathrm{FV}^\square(\vec{T}U)$. Since ξ is compatible with $(\gamma_0\sigma, \Gamma_f, \Delta)$, $x_i\xi \in \mathcal{R}_{\Delta \vdash x_i\gamma_0\sigma}$. Since $x_i\gamma_0\sigma \downarrow x_i\gamma_0\rho\sigma$, after Lemma 114 (b), $x_i\xi \in \mathcal{R}_{\Delta \vdash x_i\gamma_0\rho\sigma}$.

We now define ξ' such that $[\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi} = [\![\Gamma_0 \vdash U\gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$ and $[\![\Gamma_f \vdash T_i]\!]_{\Delta,\gamma_0\rho\sigma,\xi} = [\![\Gamma_0 \vdash T_i\gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$. Let $Y \in \mathrm{dom}^\square(\Gamma_0)$. By **(b)**, for all $X \in \mathrm{FV}^\square(\vec{T}U)$, $X\gamma_0\rho \in \mathrm{dom}^\square(\Gamma_0)$ and, for all $X, X' \in \mathrm{FV}^\square(\vec{T}U)$, $X\gamma_0\rho = X'\gamma_0\rho$ implies $X = X'$. Then, if there exists X (unique) such that $Y = X\gamma_0\rho$, we take $Y\xi' = X\xi$. Otherwise, we take $Y\xi' = \top_{\Delta \vdash Y\sigma}$. We check that ξ' is compatible with $(\sigma, \Gamma_0, \Delta)$. Let $Y \in \mathrm{dom}^\square(\Gamma_0)$. If $Y = X\gamma_0\rho$ then $Y\xi' = X\xi$. Since ξ is compatible with $(\gamma_0\rho\sigma, \Gamma_0, \Delta)$, $X\xi \in \mathcal{R}_{\Delta \vdash X\gamma_0\rho\sigma}$. Since $X\gamma_0\rho = Y$, $Y\xi' \in \mathcal{R}_{\Delta \vdash Y\sigma}$. Finally, if $Y \neq X\gamma_0\rho$, $Y\xi' = \top_{\Delta \vdash Y\sigma} \in \mathcal{R}_{\Delta \vdash Y\sigma}$. Therefore, ξ' is compatible with $(\sigma, \Gamma_0, \Delta)$. By candidates substitution, there exists ξ'' such that $[\![\Gamma_0 \vdash U\gamma_0\rho]\!]_{\Delta,\sigma,\xi'} = [\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi''}$ and, for all $X \in \mathrm{dom}^\square(\Gamma_f)$, $X\xi'' = [\![\Gamma_0 \vdash X\gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$. If $X \in \mathrm{FV}^\square(\vec{T}U)$ then, by **(b)**, $X\gamma_0\rho = Y \in \mathrm{dom}^\square(\Gamma_0)$ and $X\xi'' = Y\xi' = X\xi$. Since ξ'' and ξ are equal on $\mathrm{FV}^\square(U)$, $[\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi''} = [\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi}$. Hence, $[\![\Gamma_f \vdash U]\!]_{\Delta,\gamma_0\rho\sigma,\xi} = [\![\Gamma_0 \vdash U\gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$. The proof that $[\![\Gamma_f \vdash T_i]\!]_{\Delta,\gamma_0\rho\sigma,\xi} = [\![\Gamma_0 \vdash T_i\gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$ is similar.

We now prove that $(\sigma, \Gamma_0, \Delta)$ is valid w.r.t. ξ'. By definition of the General Schema, $(l \rightarrow r, \Gamma_0, \rho)$ is well-formed : $\Gamma_0 \vdash f(\vec{l})\rho : U\gamma_0\rho$, $\mathrm{dom}(\rho) \cap \mathrm{dom}(\Gamma_0) = \emptyset$ and, for all $x \in \mathrm{dom}(\Gamma_0)$, there exists i such that $l_i : T_i\gamma_0 \ (\rhd_1^\rho)^* \ x : x\Gamma_0$. By inversion, $\Gamma_0 \vdash l_i\rho : T_i\gamma_0\rho$. Since $\Delta \vdash l_i\sigma \in [\![\Gamma_0 \vdash T_i\gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$, by correctness of the accessibility relation, $\Delta \vdash x\sigma \in [\![\Gamma_0 \vdash x\Gamma_0\rho]\!]_{\Delta,\sigma,\xi'}$. Since $\mathrm{dom}(\rho) \cap \mathrm{dom}(\Gamma_0) = \emptyset$, $x\Gamma_0\rho = x\Gamma_0$ and $\Delta \vdash x\sigma \in [\![\Gamma_0 \vdash x\Gamma_0]\!]_{\Delta,\sigma,\xi'}$.

Hence, by correctness of the computable closure, $\Delta \vdash r\sigma \in [\![\Gamma_0 \vdash U\gamma_0\rho]\!]_{\Delta,\sigma,\xi'} = [\![\Gamma_f \vdash U]\!]_{\Delta,\theta,\xi}$. ■

Lemma 148 (Reductibility of typable terms) For all $\Gamma \vdash t : T$, $\Delta \in \mathbb{E}$, $\theta : \Gamma \rightarrow \Delta$, ξ compatible with (θ, Γ, Δ), if (θ, Γ, Δ) is valid w.r.t. ξ then $\Delta \vdash t\theta \in [\![\Gamma \vdash T]\!]_{\Delta,\theta,\xi}$.

Proof. By induction on $\Gamma \vdash t : T$. We proceed like in the Lemma of correctness of the computable closure but in the case (symb) where we use the Lemmas of reductibility of first-order and higher-order symbols. ■

Theorem 149 (Strong normalization) Every typable term is strongly normalizable.

Proof. Assume that $\Gamma \vdash t : T$. Let θ be the identity substitution. This is a well-typed substitution from Γ to Γ. Let ξ be the candidate assignment defined by $X\xi = \top_{\Gamma \vdash X}$. It is compatible with (θ, Γ, Γ) since, for all $X \in \mathrm{dom}^\Box(\Gamma)$, $X\xi \in \mathcal{R}_{\Gamma \vdash X}$. Finally, (θ, Γ, Γ) is valid w.r.t. ξ since, for all $x \in \mathrm{dom}(\Gamma)$, $\Gamma \vdash x \in [\![\Gamma \vdash x\Gamma]\!]_{\Gamma,\theta,\xi}$. Therefore, after the Lemma of reductibility of typable terms, $\Gamma \vdash t \in [\![\Gamma \vdash T]\!]_{\Delta,\theta,\xi}$ and, since $[\![\Gamma \vdash T]\!]_{\Delta,\theta,\xi} \subseteq \mathcal{SN}$, t is strongly normalizable. ■

Chapter 9

Future directions of research

In this section, we enumerate, more or less by order of importance, some of our strong normalization conditions that we should weaken or some extensions that we should study in order to get more general conditions. All these problems seem difficult.

- **Rewriting modulo.** In our work, we have not considered rewriting modulo some very useful equational theories like associativity and commutativity. It is important to be able to extend our results to this kind of rewriting. But, while this does not seem to create important difficulties for rewriting at the object level and we already have preliminary results in this direction, it is less clear for rewriting at the type level.

- **Quotient types.** We have seen that rewriting enables one to formalize quotient types by allowing rules on "constructors". However, to prove properties by "induction" on such types requires knowing what the normal forms are [72] and may also require a particular reduction strategy [40] or conditional rewriting.

- **Confluence.** Among our strong normalization conditions, we not only require rewriting to be confluent but also its combination with β-reduction. This is a strong condition since we cannot rely on strong normalization for proving confluence [95, 21]. Except for first-order rewriting systems without dependent types [28] or left-linear higher-order rewrite systems [92, 117], few results are known on modularity of confluence for the combination higher-order rewriting and β-reduction. It would therefore be interesting to study this very general question.

- **Local confluence.** We believe that, perhaps, local confluence is sufficient for establishing our result. Indeed, local confluence and strong normalization imply confluence. However, in this case, it seems necessary to prove many properties simultaneously like strong normalization and correctness of β-reduction (*"subject reduction"*), which seems difficult since many definitions rely on this last property.

- **Simplicity.** For non-primitive predicate symbols, we require that their defining rules have no critical pairs between them or with the other rules. These strong conditions allow us to define a valid interpretation in a simple way. It is important to be able to weaken these conditions in order to capture more decision procedures.

- **Logical consistency.** In the Calculus of Algebraic Constructions, in contrast with the pure Calculus of Constructions, it is possible that symbols and rules enable one to write a normal proof of $\bot = (P:\star)P$. Strong normalization does not suffice to ensure logical consistency. We should look for syntactic conditions like the "strongly consistent" environments of J. Seldin [106]) and, more generally, we should study the models of the Calculus of Algebraic Constructions.

- **Conservativity.**[1] We have seen that, in the Calculus of Constructions, adding rewriting allows one to type more terms. It is then important to study whether a proposition provable by using rewrite rules $l_1 \rightarrow r_1, \ldots$ can also be proved by using hypothesis $l_1 = r_1, \ldots$. This is another way to establish logical consistency and better understand the impact of rewriting on typing and provability.

- **Local definitions.** In our work, we have considered only globally defined symbols, that is, symbols typable in the empty environment. However, in practice, during a formal proof in a system like Coq [112], it may be very useful to introduce symbols and rules using some hypothesis. We should study the problems arising from local definitions and how our results can be used to solve them. Local abbreviations have already been studied by E. Poll and P. Severi [101]. Local definitions by rewriting is considered by J. Chrzaszcz [29].

- **HORPO.** For higher-order definitions, we have chosen to extend the General Schema of J.-P. Jouannaud and M. Okada [75]. But the Higher-Order Recursive Path Ordering (HORPO) of J.-P. Jouannaud and A. Rubio [76], which is an extension N. Dershowitz's RPO to the terms of the simply typed λ-calculus, is naturally more powerful. D. Walukiewicz has recently extended this ordering to the terms of the Calculus of Constructions with symbols at the object level [118]. The combination of the two works should allow us to extend RPO to the terms of the Calculus of Constructions with symbols at the type level also.

- **η-Reduction.** Among our conditions, we require the confluence of $\rightarrow = \rightarrow_{\mathcal{R}} \cup \rightarrow_\beta$. Hence, our results cannot be directly extended to η-reduction, which is well known to create important difficulties [58] since $\rightarrow_\beta \cup \rightarrow_\eta$ is not confluent on badly typed terms.

- **Non-strictly positive predicates.** The ordering used in the General Schema for comparing the arguments of the function symbols can capture recursive definitions on basic and strictly positive types but cannot capture recursive definitions on non-strictly positive types [88]. However, N. P. Mendler [90] has shown that such definitions are strongly normalizing. It would be interesting to extend our work to such definitions.

[1] We thank Henk Barendregt for having suggested us to study this question deeper.

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] T. Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, Edinburgh University (UK), 1993.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[4] F. Barbanera. Adding algebraic rewriting to the Calculus of Constructions : strong normalization preserved. In *Proc. of the 2nd Int. Work. on Conditional and Typed Rewriting Systems*, LNCS 516, 1990.

[5] F. Barbanera and M. Fernández. Combining first and higher order rewrite systems with type assignment systems. In *Proc. of the 1st Int. Conf. on Typed Lambda Calculi and Applications*, LNCS 664, 1993.

[6] F. Barbanera and M. Fernández. Modularity of termination and confluence in combinations of rewrite systems with λ_ω. In *Proc. of the 20th Int. Colloq. on Automata, Languages and Programming*, LNCS 700, 1993.

[7] F. Barbanera, M. Fernández, and H. Geuvers. Modularity of strong normalization and confluence in the algebraic-λ-cube. In *Proc. of the 9th IEEE Symp. on Logic in Computer Science*, 1994. Extended version in [8].

[8] F. Barbanera, M. Fernández, and H. Geuvers. Modularity of strong normalization in the algebraic-λ-cube. *Journal of Functional Programming*, 7(6):613–660, 1997.

[9] H. Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[10] H. Barendregt. Lambda calculi with types. In S. Abramski, D. Gabbay, and T. Maibaum, editors, *Handbook of logic in computer science*, volume 2. Oxford University Press, 1992.

[11] B. Barras. *Auto-validation d'un système de preuves avec familles inductives*. PhD thesis, Université Paris VII (France), 1999.

[12] B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.C. Filliâtre, E. Giménez, H. Herbelin, G. Huet, H. Laulhère, P. Loiseleur, C. Mu noz, C. Murthy, C. Parent, C. Paulin, A. Saïbi, and B. Werner. *The Coq Proof Assistant Reference Manual – Version 6.3*. INRIA Rocquencourt (France), 1999. Available at `http://coq.inria.fr/`.

[13] G. Barthe. The relevance of proof-irrelevance. In *Proc. of the 25th Int. Colloq. on Automata, Languages and Programming*, LNCS 1443, 1998.

[14] G. Barthe and H. Geuvers. Congruence types. In *Proc. of the 9th Int. Work. on Computer Science Logic*, LNCS 1092, 1995.

[15] G. Barthe and H. Geuvers. Modular properties of algebraic type systems. In *Proc. of the 2nd Int. Work. on Higher-Order Algebra, Logic and Term Rewriting*, LNCS 1074, 1995.

[16] G. Barthe and P.-A. Melliès. On the subject reduction property for algebraic type systems. In *Proc. of the 10th Int. Work. on Computer Science Logic*, LNCS 1258, 1996.

[17] G. Barthe and F. van Raamsdonk. Termination of algebraic type systems : the syntactic approach. In *Proc. of the 6th Int. Conf. on Algebraic and Logic Programming*, LNCS 1298, 1997.

[18] P. Bendix and D. Knuth. *Computational problems in abstract algebra*, chapter Simple word problems in universal algebra. Pergamon Press, 1970.

[19] S. Berardi. Towards a mathematical analysis of the Coquand-Huet Calculus of Constructions and the other systems in Barendregt's Cube. Technical report, Carnegie-Mellon University (USA) and Universita di Torino (Italy), 1988.

[20] F. Blanqui. Definitions by rewriting in the Calculus of Constructions. In *Proc. of the 16th IEEE Symp. on Logic in Computer Science*, 2001.

[21] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Proc. of the 11th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1833, 2000.

[22] F. Blanqui, J.-P. Jouannaud, and M. Okada. The Calculus of Algebraic Constructions. In *Proc. of the 10th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1631, 1999.

[23] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive-data-type Systems. *Theoretical Computer Science*, 277, 2002. To appear. Submitted in 1999, revised in 2001. Available at `http://www.lri.fr/~blanqui/`.

[24] P. Borovanský, H. Cirstea, H. Dubois, C. Kirchner, H. Kirchner, P.-E. Moreau, C. Ringeissen, and M. Vittek. *ELAN User Manual*. INRIA Nancy (France), 2000. Available at `http://elan.loria.fr/`.

[25] V. Breazu-Tannen. Combining algebra and higher-order types. In *Proc. of the 3rd IEEE Symp. on Logic in Computer Science*, 1988.

[26] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization. In *Proc. of the 16th Int. Colloq. on Automata, Languages and Programming*, LNCS 372, 1989. Extended version in [27].

[27] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization. *Theoretical Computer Science*, 83(1):3–28, 1991.

[28] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic confluence. *Information and Computation*, 114(1):1–29, 1994.

[29] J. Chrzaszcz. Modular rewriting in the Calculus of Constructions, 2000. Presented at the Int. Work. on Types for Proofs and Programs.

[30] A. Church. A simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[31] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. *Maude: Specification and Programming in Rewriting Logic*. SRI International's Computer Science Laboratory (USA), 1999. Available at `http://maude.csl.sri.com/`.

[32] L. Colson and D. Fredholm. System T, call-by-value and the minimum problem. *Theoretical Computer Science*, 206:301–315, 1998.

[33] E. Contejean, C. Marché, B. Monate, and X. Urbain. CiME, 2000. Available at `http://www.lri.fr/~demons/cime.html`.

[34] T. Coquand. Pattern matching with dependent types. In *Proc. of the 1992 Int. Work. on Types for Proofs and Programs. See* `http://www.lfcs.informatics.ed.ac.uk/research/types-bra/proc/`.

[35] T. Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.

[36] T. Coquand and J. Gallier. A proof of strong normalization for the Theory of Constructions using a Kripke-like interpretation, 1990. Paper presented at the 1st Int. Work. on Logical Frameworks but not published in the proceedings. Available at `ftp://ftp.cis.upenn.edu/pub/papers/gallier/sntoc.dvi.Z`.

[37] T. Coquand and G. Huet. A theory of constructions, 1984. Paper presented at the Int. Symp. on Semantics of Data Types but not published in the proceedings. Extended version in [38].

[38] T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2–3):95–120, 1988.

[39] T. Coquand and C. Paulin-Mohring. Inductively defined types. In *Proc. of the 1988 Int. Conf. on Computer Logic*, LNCS 417.

[40] P. Courtieu. Normalized types. In *Proc. of the 15th Int. Work. on Computer Science Logic*, LNCS 2142, 2001.

[41] H. B. Curry and R. Feys. *Combinatory Logic*. North-Holland, 1958.

[42] N. de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In *Proc. of the Symp. on Automatic Demonstration*, Lecture Notes in Mathematics. Springer, 1968. Reprinted in [60].

[43] N. Dershowitz. Hierarchical termination. In *Proc. of the 4th Int. Work. on Conditional and Typed Rewriting Systems*, LNCS 968, 1994.

[44] N. Dershowitz. Orderings for term rewriting systems. In *20th IEEE Symposium on Foundations of Computer Science*, 1979. Extended version in [45].

[45] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[46] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6. North-Holland, 1990.

[47] D. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. In *Proc. of the 4th Int. Conf. on Rewriting Techniques and Applications*, LNCS 488, 1991. Extended version in [48].

[48] D. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. *Information and Computation*, 101(2):251–267, 1992.

[49] G. Dowek. La part du calcul, 1999. Mémoire d'habilitation. Available at `http://logical.inria.fr/~dowek/`.

[50] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. Technical Report 3400, INRIA Rocquencourt (France), 1998.

[51] G. Dowek and B. Werner. Proof normalization modulo. In *Proc. of the 1998 Int. Work. on Types for Proofs and Programs*, LNCS 1657.

[52] G. Dowek and B. Werner. An inconsistent theory modulo defined by a confluent and terminating rewrite system, 2000. Available at `http://logical.inria.fr/~dowek/`.

[53] K. Drosten. *Termersetzungssysteme*. PhD thesis, Passau University (Germany), 1989.

[54] M. Fernández. *Modèles de calculs multiparadigmes fondés sur la réécriture*. PhD thesis, Université Paris-Sud (France), 1993. See [8].

[55] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ-2. In *Proc. of the 12th ACM Symp. on Principles of Programming Languages, 1985.*

[56] J. Gallier. On Girard's "Candidats de Réductibilité". In P.-G. Odifreddi, editor, *Logic and Computer Science*. North-Holland, 1990.

[57] G. Gentzen. *Investigations into logical deduction.* PhD thesis, Göttingen, Germany, 1933. See [109].

[58] H. Geuvers. *Logics and Type Systems.* PhD thesis, Nijmegen University (Netherlands), 1993.

[59] H. Geuvers and M.-J. Nederhof. A modular proof of strong normalization for the Calculus of Constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

[60] H. Geuvers, R. Nederpelt, and R. de Vrijer, editors. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, 1994.

[61] E. Giménez. Structural recursive definitions in type theory. In *Proc. of the 25th Int. Colloq. on Automata, Languages and Programming*, LNCS 1443, 1998.

[62] E. Giménez. *Un Calcul de Constructions infinies et son application à la vérification de systèmes communiquants.* PhD thesis, ENS Lyon (France), 1996.

[63] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. Fenstad, editor, *Proc. of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, 1971. See [64].

[64] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambridge University Press, 1988.

[65] K. Gödel. Über einer bisher noch nicht benützte Erweiterung des finiten standpunktes. *Dialectica*, 12, 1958. Reprinted in [66].

[66] K. Gödel. *Collected works – vol. 2 : publications 1938-1974.* Oxford University Press, 1990.

[67] J. A. Goguen and J. J. Tardo. An introduction to OBJ : a language for writing and testing formal algebraic specifications. In *Proc. of the Specification and Reliable Software Conference*, 1979.

[68] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Proc. of the 2nd IEEE Symp. on Logic in Computer Science*, 1987.

[69] W. A. Howard. The formulae-as-types notion of construction (1969). In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[70] J. Hsiang. *Topics in Automated Theorem Proving and Program Generation*. PhD thesis, University of Illinois (USA), 1982.

[71] J.-P. Jouannaud, C. Kirchner, H. Kirchner, and A. Megreliss. Programming with equalities, subsorts, overloading and parametrization in OBJ. *Journal of Logic Programming*, 12(1 and 2):257–279, 1992.

[72] J.-P. Jouannaud and E. Kounalis. Proof by induction in equational theories without constructors. In *Proc. of the 1st IEEE Symp. on Logic in Computer Science*, 1986. Extended version in [73].

[73] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1–33, 1989.

[74] J.-P. Jouannaud and M. Okada. Executable higher-order algebraic specification languages. In *Proc. of the 6th IEEE Symp. on Logic in Computer Science*, 1991. Extended version in [75].

[75] J.-P. Jouannaud and M. Okada. Abstract Data Type Systems. *Theoretical Computer Science*, 173(2):349–391, 1997.

[76] J.-P. Jouannaud and A. Rubio. The Higher-Order Recursive Path Ordering. In *Proc. of the 14th IEEE Symp. on Logic in Computer Science*, 1999. Extended version in [78].

[77] J.-P. Jouannaud and A. Rubio. A recursive path ordering for higher-order terms in eta-long beta-normal form. In *Proc. of the 7th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1103, 1996.

[78] J.-P. Jouannaud and A. Rubio. Higher-order recursive path orderings *"à la carte"*, 2001. Journal submission. Available at `http://www.lri.fr/~jouannau/`.

[79] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems : introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[80] C. Loria-Saenz and J. Steinbach. Termination of combined (rewrite and λ-calculus) systems. In *Proc. of the 3rd Int. Work. on Conditional and Typed Rewriting Systems*, LNCS 656, 1992.

[81] Z. Luo. *An Extended Calculus of Constructions*. PhD thesis, Edinburgh University (UK), 1990.

[82] Z. Luo and R. Pollack. *LEGO Proof Development System : User's manual*. Edinburgh University (UK), 1992. Available at `http://www.dcs.ed.ac.uk/home/lego/`.

[83] O. Lysne and J. Piris. A termination ordering for higher order rewrite systems. In *Proc. of the 6th Int. Conf. on Rewriting Techniques and Applications*, LNCS 914, 1995.

[84] P. Martin-Löf. Haupsatz for the intuitionnistic theory of iterated inductive definitions. In J. Fenstad, editor, *Proc. of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1971. See [85].

[85] P. Martin-Löf. *Intuitionistic type theory*. Bibliopolis, Napoli, Italy, 1984.

[86] R. Matthes. Monotone fixed-point types and strong normalization. In *Proc. of the 12th Int. Work. on Computer Science Logic*, LNCS 1584, 1998.

[87] R. Matthes. *Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types*. PhD thesis, München University (Germany), 1998.

[88] R. Matthes. Lambda calculus: A case for inductive definitions. Available at `http://www.tcs.informatik.uni-muenchen.de/~matthes/`, 2000.

[89] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(2):3–29, 1998.

[90] N. P. Mendler. *Inductive Definition in Type Theory*. PhD thesis, Cornell University (USA), 1987.

[91] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In *Proc. of the 1989 Int. Work. on Extensions of Logic Programming*, LNCS 475.

[92] F. Müller. Confluence of the lambda calculus with left-linear algebraic rewriting. *Information Processing Letters*, 41(6):293–299, 1992.

[93] R. Nederpelt. *Strong normalization in a typed lambda calculus with lambda structured types*. PhD thesis, Eindhoven University (Netherlands), 1973.

[94] M. Newman. On theories with a combinatorial definition of equivalence. *Annals of Mathematics*, 43(2):223–243, 1942.

[95] T. Nipkow. Higher-order critical pairs. In *Proc. of the 6th IEEE Symp. on Logic in Computer Science*, 1991. Extended version in [89].

[96] M. J. O'Donnell. Computing in systems described by equations. In *LNCS 58*. Springer, 1977.

[97] M. Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent term rewrite system. In *Proc. of the 1989 Int. Symp. on Symbolic and Algebraic Computation*, ACM Press.

[98] C. Paulin-Mohring. Extracting Fω's programs from proofs in the Calculus of Constructions. In *Proc. of the 16th ACM Symp. on Principles of Programming Languages, 1989*.

[99] C. Paulin-Mohring. Inductive definitions in the system Coq - rules and properties. In *Proc. of the 1st Int. Conf. on Typed Lambda Calculi and Applications*, LNCS 664, 1993.

[100] D. A. Plaisted. A recursively defined ordering for proving termination of term rewriting systems. Technical report, University of Illinois (USA), 1978.

[101] E. Poll and P. Severi. Pure Types Systems with definitions. In *Proc. of the 3rd Int. Symp. on Logical Foundations of Computer Science*, LNCS 813, 1994.

[102] R. Pollack. *The theory of LEGO, a proof checker for the Extended Calculus of Constructions*. PhD thesis, Edinburgh University (UK), 1994.

[103] J. Reynolds. Towards a theory of type structure. In *Programming Symposium, LNCS 19*, 1974.

[104] M. Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 1987.

[105] H. Schwichtenberg. Definierbare Funktionen im λ–Kalkül mit Typen. *Archive for Mathematical Logic*, 17:113–114, 1976.

[106] J. Seldin. Excluded middle without definite descriptions in the theory of constructions. In *Proc. of the 1st Montreal Workshop on Programming Language Theory*, 1991.

[107] M. P. A. Sellink. Verifying process algebra proofs in type theory. In *Proc. of the Int. Work. on Semantics of Specification Languages*, Workshops in Computing, 1993.

[108] M. Stefanova. *Properties of Typing Systems*. PhD thesis, Nijmegen University (Netherlands), 1998.

[109] M. E. Szabo, editor. *Collected papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1969.

[110] W. W. Tait. Infinitely long terms of transfinite type. In J. Crossley and M. Dummet, editors, *Formal Systems and Recursive Functions*, Studies in Logic and the Foundations of Mathematics. North-Holland, 1965.

[111] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967.

[112] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version 7.1*. INRIA Rocquencourt (France), 2001. Available at `http://coq.inria.fr/`.

[113] J. Terlouw. Een nadera bewijstheoretische analyse van GSTT's. Technical report, Nijmegen University (Netherlands), 1989.

[114] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3):141–143, 1987.

[115] L. S. van Benthem Jutting. Typing in pure type systems. *Information and Computation*, 105(1):30–41, 1993.

[116] J. van de Pol. *Termination of higher-order rewrite systems.* PhD thesis, Utrecht University (Nederlands), 1996.

[117] V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting.* PhD thesis, Vrije University (Netherlands), 1994.

[118] D. Walukiewicz. Termination of rewriting in the Calculus of Constructions. In *Proc. of the 2000 Workshop on Logical Frameworks and Meta-languages.*

[119] B. Werner. *Une Théorie des Constructions Inductives.* PhD thesis, Université Paris VII (France), 1994.

[120] A. Whitehead and B. Russell. *Principia Mathematica.* Cambridge University Press, 1911.

Index

$(>_1,\ldots,>_n)_{\mathrm{lex}}$, Chap. 2
$(\mathcal{G},\mathcal{R}_{\mathcal{G}})$, Def. 90
$(l\rightarrow r,\Gamma,\rho)$, Def. 58
$>_{\mathrm{mul}}$, Chap. 2
$>_{stat_f}$, Def. 78
$>_1,>_2,>$, Def. 79
$>_{\mathcal{C}},\geq_{\mathcal{C}},=_{\mathcal{C}}$, Def. 72
$>_{\mathcal{F}},\geq_{\mathcal{F}},=_{\mathcal{F}}$, Def. 77
$T\rightarrow U$, Chap. 2
$[C]$, page 102
$[F]$, Sec. 8.6
$\Gamma\vdash t:T$, Def. 8
$[\![\Gamma\vdash T]\!]^I_{\Delta,\theta,\xi}$, Def. 120
$\rightarrow^*,\rightarrow^+,\leftarrow,\leftrightarrow^*,\downarrow$, Chap. 2
$\rightarrow_\beta,\rightarrow_\eta$, Chap. 2
$\rightarrow_\iota$, page 14, page 78
$\rightarrow_{\iota'}$, Def. 94, Conj. 95, Lem. 104
$\bigwedge_{\Gamma\vdash T}$, Def. 113
$\succ,\succeq,\simeq$, Def. 91
$\equiv$, Sec. 7.3
$\leq$, page 102, Sec. 8.6.2
$\leq_{\Gamma\vdash T}$, Def. 113
$[x\!:\!T]u$, Def. 2
$\langle t\rangle,\langle\Gamma\rangle$, Def. 99, Thm. 98, Lem. 100
$(x\!:\!T)U$, Def. 2
$\sqsupset,\sqsupset_f$, Def. 132
$\subseteq$, Def. 5
$\star,\Box$, page 13, Def. 65
$\rhd$, Def. 3
$\rhd_1,\rhd_2$, Def. 76
$\vdash$, Def. 8
$\vdash_f$, Def. 53
$\vdash_c$, Def. 80
$\vdash_s$, Rem. 9, Lem. 43
$\vdash_\Upsilon$, Def. 99
$\vdash_w$, Rem. 9, Lem. 19
$\top_{\Gamma\vdash T}$, Def. 113
$\triangle$, page 77
$\emptyset$, Def. 5
$\vec{t}$, Chap. 2
$f(\vec{t})$, Def. 2
$l\rightarrow r$, Def. 56
$t|_p, t[u]_p$, 26
uv, Def. 2
(conv'), Rem. 9
(weak'), Rem. 9

$\mathcal{A}$, Def. 7
$\vec{a}$, page 102
a, Def. 91, Thm. 144, Lem. 147
A0, Def. 91, page 72
A1, Def. 91, page 72, Lem. 127, Lem. 138, Lem. 146
A2, Def. 91, Lem. 135
A3, Def. 91, Sec. 8.6, Lem. 146
A4, Def. 91, Lem. 148
abstraction, Def. 2
Acc(c), Def. 72, Def. 76
accessibility, Lem. 134, Lem. 135
accessible
 position, Def. 72
 term, Def. 76
admissible
 ATS, Thm. 60
 inductive structure, Def. 74
 inductive type, Def. 96
 pre-RTS, Def. 32
A_i, page 102
a_i, page 102
algebraic
 term, Def. 55
 type system, Def. 57
$aliens, aliens_{\mathcal{G}}$, Def. 142, Lem. 143, Thm. 144
α-equivalence, Chap. 2, Lem. 44
application, Def. 2
arity

maximal, Def. 70, Lem. 71
status, Def. 78
symbol, Def. 1, Rem. 10, Lem. 121 (R3)
assignment
candidate, Def. 117
canonical, Def. 117, Thm. 149
status, Def. 78
type, Def. 6
ATS, Def. 57
axiom, Def. 7

$\mathcal{B}$, Def. 7
$\overline{\mathbb{B}}$, page 92
b, Def. 91, Lem. 147
Barendregt's cube, Fig. 1.2
basic predicate, Def. 75
β-reduction, page 9, Chap. 2

C, page 102
$\mathbb{C}$, Def. 20
$\mathcal{C}$, Def. 7
$\mathcal{C}_\Gamma, \mathbb{C}_\Gamma$, Rem. 9, Def. 20, Lem. 37
c, Def. 91, Lem. 143
CAC, Def. 65
Calculus of Constructions, page 15
Algebraic, Def. 65
Inductive, page 17, Sec. 7.1
candidate, Def. 113, Lem. 114, Lem. 116, Lem. 115
assignment, Def. 117
restriction, Def. 113
substitution, Lem. 123
canonical
candidate assignment, Def. 117
type, Def. 61, Lem. 62
$cap, cap_{\mathcal{G}}$, Def. 142, Lem. 143, Thm. 144
CC, page 15
$\mathcal{CF}, \mathcal{CF}^s$, Def. 56
$C_f^i(\vec{u})$, Def. 78
CIC, page 17, Sec. 7.1, Fig. 7.1
CIC$^-$, Def. 97, Fig. 7.2, Lem. 104
class
syntactic, Def. 67
typing, Def. 66
classification, Lem. 48
closure, Def. 106, Lem. 107
compatible with
ordering
reduction, Def. 53
rule, Rem. 84
status assignment, Def. 78
type assignment, Def. 53
substitution
candidate assignment, Def. 53
computable closure, Fig. 6.1, Def. 80, Rem. 83, Lem. 87, Lem. 88, Lem. 85, Rem. 86, Cor. 89, Lem. 146
confluence, Chap. 2, Chap. 9
local, Chap. 2, Lem. 64, Chap. 9
conjecture, Conj. 95
conservativity, Chap. 9
consistency, Rem. 93, Chap. 9
constant symbol, Def. 56
$Constr_i^I$, Def. 99
$Constr(i, I)$, page 77
constructor, Def. 69, page 77
conversion correctness, Lem. 42
convertibility of types, Lem. 41
correctness of types, Lem. 28
$\mathcal{CR}$, page 93
cut elimination, page 11

D, page 102
$\mathcal{D}$, Def. 140
d, Def. 91, Lem. 143, Lem. 145
decidability
computable closure, Rem. 83
typing, Thm. 92
well-typed rule, page 54
decision procedure, page 20
defined symbol, Def. 56
Δ_i, page 103
$\Delta[I, X, C, f, F]$, page 78
$\Delta\{I, X, C, Q, c\}$, page 78
$\Delta'[I, X, C, f, Q, \vec{f}, \vec{z}]$, Def. 94
$\Delta'\{I, X, C, \vec{x}y, K, c\}$, Def. 97
$\Delta'_S[I, X, C, f, Q, \vec{f}, \vec{z}]$, Def. 99
$\Delta'_W[I, X, C, f, Q, \vec{f}, \vec{z}]$, Def. 99
dependence w.r.t.
symbol, Lem. 54, Lem. 87
variable, Lem. 50
dependent

product, Def. 2
type, page 12
derived type, Def. 61, Lem. 62
$\mathcal{DF}, \mathcal{DF}^s$, Def. 56
dom(Γ), Def. 5
domain
environment, Def. 5
substitution, Def. 4
$\mathrm{dom}(\theta), \mathrm{dom}^s(\theta)$, Def. 4

$\mathbb{E}$, Def. 8
$\mathcal{E}$, Def. 5
e, Def. 91, Lem. 143
elimination rule, page 10
$Elim(I, Q, \vec{a}, c)$, page 77
environment, Def. 5, Lem. 14
environment conversion, Lem. 22
η-reduction, Chap. 2

F, Sec. 8.6
$\mathcal{F}, \mathcal{F}^s, \mathcal{F}_n, \mathcal{F}_n^s$, Def. 1
f, Def. 91, Lem. 143
first-order
rewrite system, Def. 90
symbol, Def. 91
$\mathcal{F}_1, \mathcal{F}_\omega$, Def. 91
functional TSM, Def. 40
$FunElim(I, Q, \vec{f})$, page 78
FV(Γ), Def. 5
$\mathrm{FV}(t), \mathrm{FV}^s(t)$, Chap. 2

Γ_f, Def. 6
$\Gamma \vdash T$, Def. 106
General Schema, page 64, Def. 82

higher-order
rewriting, page 18
symbol, Def. 91
HORPO, page 65, Chap. 9

I, Def. 119
$\mathcal{I}$, page 102
I2, Def. 74, Lem. 128
I3, Def. 74, Lem. 128, Lem. 129
I4, Def. 74, Lem. 127, Lem. 128, Lem. 129
I5, Def. 74, Lem. 127, Lem. 128, Lem. 129
I6, Def. 74, Def. 126, Lem. 127, Lem. 134
$\mathcal{I}^m$, Def. 124
inconvertibility of maximal sorts, Lem. 36
Ind(C), Def. 72
Ind_I, Def. 99
inductive
position, Def. 72, Def. 124
structure, Def. 72
admissible, Def. 74
type, page 14, Sec. 6.2, page 77
admissible, Def. 96
small, page 73, page 77
$Ind(X:A)\{\vec{C}\}$, page 77
injective TSM, Def. 46
interpretation
constant, Def. 126, Lem. 127, Lem. 128
primitive, Lem. 130
defined
positive, Def. 137, Lem. 138, Lem. 139
primitive, Def. 136
recursive, Def. 140, Lem. 141
monotone, Def. 124, Lem. 125
predicate symbol, Def. 119
schema, Def. 120, Lem. 121, Lem. 122
introduction rule, page 10
inversion
RTS, Lem. 39
TSM, Lem. 21
TSM stable by substitution, Lem. 29
ι, page 8
ι-reduction, page 14, page 78
ι_Y, Def. 74
isomorphim of Curry-Howard, page 11

J, Sec. 8.6
$\mathcal{J}$, Sec. 8.6.2
$J_{\Delta \vdash G}(\vec{a})$, Def. 140

K, page 102, Sec. 8.6
$\mathbb{K}$, Def. 66
$\mathcal{K}$, Def. 67
κ_X, Def. 90

λP, Fig. 1.1
left-linear rule, Def. 56, page 73
lexicographic ordering, Chap. 2
$lex(\vec{m})$, Def. 78
lfp, Def. 126, Def. 137

local confluence, Chap. 2, Lem. 64, Chap. 9
local definition, Chap. 9
logical TSM, Def. 30

maximal
  arity, Def. 70, Lem. 71
  sort, Def. 27, Lem. 36, Lem. 45
monotone interpretation, Def. 124, Lem. 125
μ, Def. 109
multiset, Chap. 2
$mul(\vec{x})$, Def. 78

nœtherian, Chap. 2
Natural Deduction, page 10
  Modulo, page 21, Sec. 7.3
NDM, page 21, Sec. 7.3
negative position, Def. 73
neutral
  position, Def. 73
  term, Def. 112
$\mathcal{NF}$, Def. 97
non-duplicating rewrite system, Def. 90, page 73, Lem. 143
normal form, Chap. 2
normalization
  strong, Chap. 2
    $\to_{\mathcal{R}}$, Thm. 144
    CAC, Thm. 149
    $\mathrm{CIC}^{-}, \Upsilon$, Thm. 98, Lem. 103, Lem. 104
  weak, Chap. 2
ν, Def. 109

$\mathbb{O}$, Def. 66
$\mathcal{O}$, Def. 67
object, Def. 8, Def. 66
o, page 8
$o(\Gamma \vdash t)$, Def. 131
order, Def. 131
ordering
  $(>_1, \ldots, >_n)_{\mathrm{lex}}$, Chap. 2
  $>_{\mathrm{mul}}$, Chap. 2
  $>_{stat_f}$, Def. 78
  $>_1, >_2, >$, Def. 79
  $>_{\mathcal{C}}$, Def. 72
  $>_{\mathcal{F}}$, Def. 77
  $\succ$, Def. 91
  $\leq$, Sec. 8.6.2
  $\sqsupset$, Def. 140
  $\sqsupset, \sqsupset_f$, Def. 132
  $\rhd$, Def. 3
  $\rhd_1, \rhd_2$, Def. 76
  accessibility, Def. 76
  arguments, Def. 79, Lem. 135
  lexicographic, Chap. 2
  multiset, Chap. 2
  precedence, Def. 77
  reductibility, Def. 132, Def. 140
  status, Def. 78
  subterm, Def. 3

$\mathbb{P}$, Def. 66
$\mathcal{P}$, Def. 67
p, Def. 91, Def. 136
P1, Def. 113, Lem. 121, Lem. 146
P2, Def. 113, Lem. 121
P3, Def. 119, Lem. 130
permutation
  strong, Lem. 50
  weak, Lem. 18
$\varphi^I_{\Delta \vdash D}(\vec{a})$, Def. 126
polymorphism, page 15
position
  accessible, Def. 72
  inductive, Def. 72
  negative, Def. 73
  neutral, Def. 73
  positive, Def. 73
  strictly positive, Def. 78
  term, Def. 3
positive
  position, Def. 73
  rewrite system, Def. 90
$\mathrm{Pos}^+, \mathrm{Pos}^-, \mathrm{Pos}^\delta, \mathrm{Pos}^0, \mathrm{Pos}^{\neq 0}$, Def. 73
$\mathrm{Pos}(t), \mathrm{Pos}(f,t), \mathrm{Pos}(x,t)$, Def. 3
pre-RTS, Def. 32
  admissible, Def. 32
precedence, Def. 77
predicate, Def. 8, Def. 66
  admissible, Def. 96
  basic, Def. 75
  non-strictly positive, Chap. 9
  primitive, Def. 75
  strictly positive, Def. 75

type, Def. 66
preservation of sorts, Lem. 33
primitive
predicate, Def. 75
rewrite system, Def. 90
product
dependent, Def. 2
non-dependent, Chap. 2
$\psi^{J}_{\Delta\vdash G}(\vec{a})$, Def. 137
PTS, page 15
$\overline{\mathbb{P}}^0, \overline{\mathbb{P}}^\star, \overline{\mathbb{P}}^\Box$, page 92

q, Def. 91, Def. 137
quotient type, page 18

$\mathcal{R}, \mathcal{R}_{\mathcal{G}}$, Def. 56
$\mathcal{R}_{\Gamma\vdash T}$, Def. 113
r, Def. 91, Def. 140
$\mathcal{R}$-reduction, Def. 56
R1, Def. 113, Lem. 114, Lem. 121
R2, Def. 113, Lem. 114, Lem. 121, Lem. 147
R3, Def. 113, Lem. 114, Lem. 115, Lem. 121
R4, Def. 113, Lem. 114, Lem. 127
recursive rewrite system, Def. 82
reductibility
accessibility, Lem. 134
candidate, Def. 113, Lem. 114, Lem. 116, Lem. 115
assignment, Thm. 149
ordering, Def. 132, Lem. 133, Def. 140
symbol
first-order, Lem. 143, Lem. 145
higher-order, Lem. 147
reduction
β, page 9, Chap. 2
η, Chap. 2, Chap. 9
ι, 14, page 78
ι', Def. 94, Conj. 95, Lem. 104
$\mathcal{R}$, Def. 56
regular
sort, Def. 38
TSM, Def. 38
replacement, Lem. 15
restriction
candidate, Def. 113
candidate assignment, Def. 117
set of closures, Def. 106
rewrite rule, Def. 56
rewrite system, Def. 90
first-order, Def. 90
non-duplicating, Def. 90
positive, Def. 90
primitive, Def. 90
recursive, Def. 82
safe, Def. 90
simple, Def. 90
small, Def. 90, page 73, page 77
rewriting, Def. 56
modulo, page 74, Chap. 9
$R|_{\Gamma'}$, Def. 113
ρ, Def. 58
RTS, Def. 32, Thm. 60
rule
compatible, Rem. 84
elimination, page 10
introduction, page 10
product formation, Def. 7
rewrite, Def. 56
well-formed, Def. 81, Lem. 102, Thm. 146, Lem. 147
well-typed, Def. 58, Lem. 101

$\mathcal{S}$, Def. 1
S3, Def. 58, page 54, Thm. 60, Lem. 87
S4, Def. 58, Thm. 60, Lem. 63, Lem. 138, Lem. 141, Lem. 147
S5, Def. 58, Thm. 60, page 55, Lem. 147
safe rewrite system, Def. 90
$SElim^{Q}_{I}$, Def. 99
separation, Lem. 47
sequent, page 11
set theory, page 8
S_i, page 103
$\Sigma_{\Gamma\vdash K}$, Def. 113
simple rewrite system, Def. 90
simplicity, Def. 90, Def. 137, Lem. 138, Def. 140, Lem. 141, Lem. 146, Chap. 9
small rewrite system, Def. 90
smallness, Def. 90, Def. 137, Lem. 138, Def. 140, Lem. 141, Lem. 146
$\mathcal{SN}$, page 93

$\overline{\mathbb{SN}}_{\Gamma \vdash T}$, Def. 106
sort, Def. 1
  maximal, Lem. 36, Lem. 45
  preservation, Lem. 33
  regular, Def. 38
sorted λ-system, Def. 1
$SP(f)$, Def. 78
stable by
  context, Chap. 2, Lem. 33
  substitution, Chap. 2, Lem. 33
$stat, stat_f$, Def. 78
status, Def. 78
  assignment, Def. 78
strengthening, Lem. 50
strictly positive
  position, Def. 78
  predicate, Def. 75
subject reduction, Def. 32, Lem. 35, Thm. 60
  β, Lem. 31
  rewriting, Lem. 59
substitution, Lem. 24
  reductibility candidate, Lem. 123
  term, Def. 4
  valid, Def. 120
  well-typed, Def. 11
subterm, Def. 3, Lem. 13
symbol, Def. 1
  constant, Def. 56
  defined, Def. 56
  first-order, Def. 91, Lem. 143, Lem. 145
  higher-order, Def. 91, Lem. 147
symmetry, Lem. 33
syntactic class, Def. 67
System T, page 14

$\mathbb{T}, \mathbb{T}_0^s, \mathbb{T}_1^s$, Def. 8
$\mathcal{T}$, Def. 2
$\overline{\mathbb{T}}$, Def. 106
$\overline{\mathbb{T}}_{\Gamma \vdash T}$, Def. 106
τ, τ_f, Def. 6
$\tau(t, p)$, Def. 61
term, Def. 2
  algebraic, Def. 55
  closed, Chap. 2
  computable, Def. 120
  neutral, Def. 112
  order, Def. 131
$\mathcal{T}(\mathcal{F}, \mathcal{X}), \mathbb{T}(\mathcal{F}, \mathcal{X})$, Def. 55
t_i, page 103
transitivity, Lem. 17
TSM, Def. 7, Fig. 3.1
  β, Def. 7
  η, Def. 7
  functional, Def. 40
  injective, Def. 46
  logical, Def. 30
  regular, Def. 38
  stable by substitution, Def. 23
$\overline{\mathbb{TY}}, \overline{\mathbb{TY}}^\star, \overline{\mathbb{TY}}^\Box$, page 92
type, Def. 8
  assignment, Def. 6
  canonical, Def. 61, Lem. 62
  dependent, page 12
  derived, Def. 61, Lem. 62
  quotient, page 18, Chap. 9
  theory, page 8
Type System
  Algebraic, Def. 57
  Modulo, Def. 7
  Pure, page 15
  Reduction, Def. 32
typing
  CAC, Def. 65
  CIC, Fig. 7.1
  CIC^-, Fig. 7.2
  class, Rem. 49, Def. 66
  decidability, Thm. 92
  λP, Fig. 1.1
  TSM, Def. 8, Fig. 3.1

Υ, Def. 99, Thm. 98, Lem. 102, Lem. 101, Lem. 103

valid
  environment, Def. 8
  substitution, Def. 120
variable, Def. 1
  bound, Chap. 2
  free, Chap. 2, Lem. 12
vector, Chap. 2

weakening, Lem. 16
$WElim_I$, Def. 99

well-formed rule, Def. 81, Lem. 102, Thm. 146,
Lem. 147
well-founded, Chap. 2
well-typed
rule, Def. 58, Lem. 101
substitution, Def. 11
$\mathcal{WN}$, page 93

$\mathcal{X}, \mathcal{X}^s$, Def. 1
$\overline{\mathbb{X}}_{\Gamma \vdash T}$, Lem. 115
$\xi, \xi|_{\Gamma'}, \xi_\theta$, Def. 117
x_i, Def. 78