Modèles et normalisation des preuves

Denis Cousineau

Thèse de doctorat

Spécialité Informatique

Models and proof normalization

Préseetée et souteeue publiquemeet par

Denis Cousineau

le 1er décembre 2009

Devaet la comissioe d'examee composée de

Gilles	DOWEK	*Directeur de thèse*
Herman	GEUVERS	*Rapporteurs*
Claude	KIRCHNER	
Roberto	DI COSMO	*Examieateurs*
Thomas	EHRHARD	
Dale	MILLER	
Alexandre	MIQUEL	

Composed with LaTeX.

Si c'était à refaire, recommenceriez-vous ? dit la chanson.
Jamais on ne recommencerait, à moins d'être gâteux ou d'ignorer le goût de l'expérience.

Boris Vian

Remerciements

Contents

1

Introduction

Representation of reasoning

For a long time, mathematics was purely about computation, long before we even had machines to compute. Mathematics was at first essentially confined to book-keeping. It is believed that humans have been able to count and compute since at least 3500 b.c. in Mesopotamia. The mesopotamians known how to divide a number of grains into equal parts, to compute the area of a field, and so on... But it took approximately 3100 more years for humans to be interested in the study of the part of mathematics which analyses human reasoning: logics.

The greek philosopher Aristotle, which lived in the 4th century b.c. is considered to be the inventor of formal logics. He introduced the notion of syllogism to represent formally how a human argumentation can be built. He realized that the human reasoning ought to be describable as a formal deduction system, but he did not succeed in building a powerful enough system to represent all legitimate argumentations.

2300 years later, at the end of the 19th century, such mathematicians as David Hilbert, Gottlob Frege and Bertrand Russell reinvented the discipline of logics by improving Aristotle's method in order to define formal systems that, in effect, constituted the first predicate logics (also called first-order logics). With these formal systems, they developed an analysis of quantified statements and formalized the notion of a proof in terms that logicians still use today.

Finally, in the 1930's Gehrard Gentzen and Stanisław Jaśkowski defined, independently, what is considered today as the realization at last of the goal of Aristotle to provide a formal system to represent logical reasoning, and is still studied today. The systems they defined, Natural Deduction, Sequent

Calculus, and the Method of Suppositions are slightly different but provide the same expressiveness for representing proofs and proof-building. These systems are much more expressive to represent proofs and propositions than sentential axiomatizations that preceded, common to the systems of Hilbert, Frege, and Russell. For example, Natural Deduction considers a language in predicate logic which contains symbols of functions and predicates. Terms are built up from term-variables and functions. Propositions are built up from atomic propositions, which are predicates applied to terms, with the usual connectives we use to represent propositions: the implication, the universal quantifier, the disjunction, and so on... Inference rules are provided to distinguish between provable and unprovable propositions. A mathematical theory can be expressed in Natural Deduction, by the use of axioms, which are propositions that are always considered true. For example, given a language in predicate logic in which we represent integers, we represent the addition by such axioms as "for all integers x, $x + 0 = x$", "for all integers x and y, $x + (y + 1) = (x + 1) + y$"', and so on... Given these axioms, we are able, via the inference rules, to build proofs of propositions of arithmetic as "$2012 + 2012 = 4024$".

Confidence in those representations

How can we be sure that these formal systems actually represent human reasoning ? We consider that a system is reliable if one cannot prove an assertion and its opposite using this system, or, equivalently, that there exists assertions which are not provable in this system. We say, in this case that the system is *consistent*. Two different methods have been developed to prove consistency of such a system. These two methods seem to be very different as one is syntactic and the other is semantic. We shall see that they are actually connected.

The syntactic method, called cut elimination, stems fom the question of what kind of proofs can be built for a provable assertion. The notion of cut represents the introduction of an intermediate lemma in a proof, in order to use a particular case of this lemma. The cut elimination property expresses the fact that we could have avoided to introduce the intermediate lemma and proved the particular case we need directly. If a system allows to build a cut-free proof for each provable assertion, then it is consistent, since we can verify that there is no possibility to build a cut-free proof for some assertions, which are therefore unprovable. Cut elimination has been proved by Gentzen for the sequent calculus and by Prawitz for natural deduction, hence both of them are consistent. In both sequent calculus and natural deduction, cuts have a unique representation: as a specific rule in sequent calculus and as a specific form of the proof, in natural deduction. But as soon as we want to extend natural deduction to express more powerful theories by adding

axioms, we have to introduce different notions of cut and cut elimination quickly becomes more complicated to prove.

The semantic method consists in building a model for such a system. A model is a mathematical object, which represents the considered system, and which proves the consistency of the system by its existence: the system is reliable since it can be modeled in the "real" world of mathematical objects. For example, let us consider a system in which implication $\Rightarrow$ is represented by the two rules: "if B is provable given a proof of A, then $A \Rightarrow B$ is provable" and "if $A \Rightarrow B$ and A are both provable, then B is also provable. We are able to build a model for this part of such a system by considering the $\{\top, \bot\}$-algebra: a set with two distinct elements $\top$ (that is "true") and $\bot$ (that is "false"), and, as a model of the implication, a mathematical function $\tilde{\Rightarrow}$ which maps ordered pairs of elements of $\{\top, \bot\}$ to elements of $\{\top, \bot\}$. This function corresponds to the well known truth table for the implication: $\top \tilde{\Rightarrow} \top = \top$, $\bot \tilde{\Rightarrow} \top = \top$, $\bot \tilde{\Rightarrow} \bot = \top$ (if A is false or B is true, then $A \Rightarrow B$ is true), and $\top \tilde{\Rightarrow} \bot = \bot$ (if A is true and B is false, then $A \Rightarrow B$ is false). This algebra has been used by George Boole, in the 19th century, to provide a model for propositional logic, and to conclude that it is consistent. This algebra has been extended into the so-called "Boolean algebra" on which we can build models for proving consistency of theories expressed in classical first-order logic. In this sense, having a model valued in this boolean algebra is a sound criterion for consistency of theories expressed in classical first-order logic. In 1929, Kurt Gödel proved that this criterion is also complete: all consistent theories expressed in classical first-order logic have such a boolean model. This completeness property expresses the fact that this criterion is general enough to discriminate exactly the theories which are consistent. Finally, this notion of algebra has been extended by Arend Heyting into the so-called Heyting algebras. He proved that having a model valued in such an algebra is a sound criterion for consistency of theories expressed in intuitionistic first-order logic.

Proofs as programs

The study of computation swept back into mathematics when Alan Turing and Alonzo Church defined, in the 1930's, Turing machines and the λ-calculus, respectively. These systems both define the same notion of computable functions. Functions that are computable are the functions whose results can be computed, in finite time, of an algorithm composed by a finite number of computation rules. They correspond to functions a computer can compute, given a program implementing the algorithm.

The λ-calculus is a very simple and powerful language which represents mathematical functions. There are three kinds of terms in this language:

variables, functions and application of a term to another one. The computation of functions is modeled by one single rule called the β-reduction. This rule expresses the fact that the computation of a function f applied to a term t is $f(t)$, i.e. the value of the function which associates $f(x)$ to x when replacing all the occurences of x in $f(x)$ by the term t. A term is considered computable if β-reduction of this term terminates (and therefore provides a *value* for this term). There exists terms which are not computable in the λ-calculus. The simply-typed λ-calculus was introduced to discriminate which of those terms are computable. The idea is to consider those functions together with their domains (called types), which are the sets of terms on which those functions are allowed to be applied. Whereas in the pure λ-calculus, functions can be applied to any term, typed λ-terms only contain functions applied to terms in their domains. The fact that a λ-term is well-typed can be verified via an algorithm given by a set of typing rules.

The Brouwer-Heyting-Kolmogorov correspondence, relates, via the Curry-de Bruijn-Howard isomorphism those typing rules and the deduction rules of natural deduction. Indeed, the typing rules for functions are the following ones: if f is of type $A \Rightarrow B$ (which represents the type of functions from A to B) and x is of type A, then $f(x)$ is of type B; and if $f(x)$ is of type B when x is of type A, then f is of type $A \Rightarrow B$. We obtain in this way the two deduction rules for the implication. This isomorphism identifies proving and typing, and a proof of a proposition can be seen as a function of the corresponding type, from a mathematical point of view, and therefore as a program of the corresponding data type, from a programming point of view. Via this isomorphism, the elimination of a cut is modeled via a β-reduction.

The first proof that the simply-typed λ-calculus actually capture only computable terms was given by William Tait. He proved that all β-reductions sequences from a well-typed term terminate. Therefore all well-typed terms are computable. We call this property "strong normalization". Since the elimination of a cut is modeled as a β-reduction, the strong normalization property entails the cut elimination property for the associated theory. Therefore it is not surprising that Tait reused for this proof the technique developed by Prawitz for proving cut elimination of natural deduction. This technique has been generalized, as the "reducibility candidates" by Jean-Yves Girard for strong normalization of system F.

This paradigm of "proofs as programs" is the foundation of several proof-assistants that can check proofs built by a human user. A problem of proof-assistants that use deductions rules to check those proofs is that expressing computations is inconvenient and often inefficient. For example, given the axioms of arithmetic $x + 0 = x$ and $x + (y + 1) = (x + 1) + y$, we need 2013 steps of deduction to prove that $2012 + 2012 = 4024$. The 2012 first steps use the second axiom: $2012 + 2012$ is equal to $2013 + 2011$ which is equal to $2014 + 2010$, ... which is equal to $4024 + 0$, which is finally equal to

4024, using the first axiom. Representing computation by deduction rules is therefore a liability the efficiency of proof-assistants, whereas there exists reliable processors which compute operations on integers very fast.

Deduction modulo

Deduction modulo was introduced by Gilles Dowek, Thérèse Hardin and Claude Kirchner in order to provide a logical framework where proving and computing are separated, in order to discriminate, in a proof, which parts are more efficiently performed by a processor: the computation; and which parts are more efficiently performed by a human: the (logical part of the) proof. The idea is to consider as equal, propositions which are computed equal by an external tool. This way, we only need one step of deduction to prove the proposition $2012 + 2012 = 4024$, since an external processor can compute the term $2012 + 2012$ into the term 4024. This formalism was introduced to build improved proof-assistants, which allow the human user to focus only on the parts of proofs which need human reasoning, and which can be faster in checking proofs, by using a external dedicated reliable program to compute on integers (or on other data types).

Deduction modulo is a powerful logical framework, since it allows to express proofs of theories like arithmetic, simple type theory, some variants of set theory, and so on... The counter-part of this power, is that it allows to express theories which do not satisfy the cut elimination property or the strong normalization property. Hence we have to prove cut elimination or strong normalization for each theory that we want to prove consistent. Semantic methods were developed by Olivier Hermant to prove cut elimination for theories expressed in deduction modulo. On the other hand, Gilles Dowek and Benjamin Werner adapted the technique of reducibility candidates to deduction modulo, by defining the notion of pre-model which provides a sound criterion for strong normalization of theories expressed in deduction modulo. Later, Gilles Dowek exhibited the connection between semantic methods of building models and the apparently syntactic technique of reducibility candidates, by defining an extension of Heyting algebras, called Truth Values Algebras, pointing out one of these algebras corresponding to reducibility candidates, and deducing from his work with Benjamin Werner, that having a model valued in this algebra is a sound criterion for strong normalization of theories expressed in deduction modulo. However this criterion is not known to be complete. It could be that there exists strongly normalizing theories which do not satisfy this criterion.

Complete criteria for strong normalization

In the present work, we shall define a model-theoretic criterion, based on a refinement of reducibility candidates, which is a sound and complete semantics for strong normalization of theories expressed in a fragment of deduction modulo called minimal deduction modulo, and in the $\lambda\Pi$-calculus modulo which is the adaptation of deduction modulo to the $\lambda\Pi$-calculus (the extension of the simply-typed λ-calculus with dependent types).

This thesis confirms the idea that the difference between strongly normalizing theories and non-strongly normalizing theories has also a semantical dimension and that model-theory methods can be very useful in proof-theory.

Outline

In chapter 2, we define formally natural deduction and how we can use the syntax of the λ-calculus to represent proofs of (minimal) natural deduction. We then define (minimal) deduction modulo and present some examples of theories expressed in minimal deduction modulo. We present the technique of reducibility candidates, and its adaptation for deduction modulo (pre-models). We finally define the notion of Truth Values Algebras, and show how pre-models can be defined as models valued in a specific Truth Values Algebra.

In chapter 3, we first analyze the technique of reducibility candidates and show why it would be difficult to prove that they provide a complete criterion for strong normalization. We then adapt them into a sound and complete criterion, by defining a refinement of Truth Values Algebras, called Language-dependent Truth Values Algebras, and exhibiting two of those Language-dependent Truth Values Algebras such that having a model valued in one of them is a sound and complete criterion for strong normalization in minimal deduction modulo. While the first Language-dependent Truth Values Algebra depends on the studied theory, the second one does not and provides therefore a theory-independent criterion.

In chapter 4, we first define the $\lambda\Pi$-calculus modulo, and then extend the techniques developed in the previous chapter, to provide a sound and complete criterion for strong normalization of theories expressed in the $\lambda\Pi$-calculus modulo. For that purpose, we adapt the notion of pre-models for deduction modulo to dependent types, and provide an innovative method for building interpretations of dependent types, expressive enough to reuse the techniques of the previous chapter, for proving completeness.

In chapter 5, we propose an embedding of another logical framework, called functional Pure Type Systems, into the $\lambda\Pi$-calculus modulo. An example of functional Pure Type System is the Calculus of Constructions which provides the logical foundation of the proof-assistant Coq. This embedding, which we prove sound and conservative, shows that the $\lambda\Pi$-calculus modulo is a powerful logical framework, since it subsumes all the theories expressed in functional Pure Type Systems. This embedding also provides an approach to a sound and complete criterion for strongly normalizing functional Pure Type Systems, by using the notion of pre-models for $\lambda\Pi$-calculus modulo.

2

Proof normalization as a model-theoretic notion

Outline

The aim of this chapter is to provide a partial view of the state of the art concerning proof normalization, in particular, in the logical framework of deduction modulo. We shall, as a first step, present minimal natural deduction, in order to define minimal deduction modulo thereafter. We shall describe the main technique for proving proof normalization, introduced by Girard, following the work of Tait, called *reducibility candidates*. We shall see how we can adapt this technique to deduction modulo by defining the so-called *pre-models*, expliciting the fact that this apparently syntactical method can be viewed as the construction of a specific model, hence is also a semantical method. Finally, we shall define *truth values algebras* which are the kinds of algebras on which we can build such models.

2.1 Natural deduction

Natural deduction was independently proposed by Jaśkowski [37] ("the method of suppositions") and Gentzen [24] in 1935. It is a system much more expressive to represent proofs and propositions than sentential axiomatizations common to the systems of Hilbert, Frege, and Russell that preceded. This system considers a first-order language which can be many-sorted and which contains symbols of functions and predicates. Terms are built up from term-variables and functions. We shall consider minimal natural deduction, where propositions are built up from atomic propositions, which are predicates applied to terms, with the connective $\Rightarrow$ and the universal quantifier $\forall$. Inference rules are provided to distinguish between provable and unprovable propositions by giving the rules for building a proof judgement. We shall see

that, using the Curry-de Bruijn-Howard isomorphism, we can use the syntax of λ-calculus [9] to represent proofs of this system and their behaviour. This isomorphism relates typing to being a proof of a proposition. How to build a proof of a proposition can therefore be defined by the typing rules of the simply-typed lambda-calculus. Notice that we use two different binders λ to represent abstracting on proofs and abstracting on terms. We shall see in chapter 4 an extension of the simply-typed λ-calculus that uses a single binder for both those abstractions.

2.1.1 Terms, propositions and inference rules

In this work, we present a version of Natural deduction based on a many-sorted first-order language. We shall focus on minimal natural deduction, that is natural deduction with only the connective representing the implication and the universal quantifier.

Definition 2.1 ((Many-sorted first-order) language)**.**
A language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *is the combination of a set* $\mathbb{T}$ *of* sorts*, a set* $\mathbb{F}$ *of function symbols which come with their* ranks*, and a set* $\mathbb{P}$ *of predicate symbols which also come with their* ranks*.*

Given an infinite set of *variables* of each sort, we define the set of terms associated to this language, as follows:

Definition 2.2 (Terms)**.**
Given a language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$*, the set of terms is defined inductively from variables.*

- *Variables of sort* T *are terms of sort* T*.*
- *If* f *is a function symbol of rank* $\langle T_1, \ldots, T_n, U \rangle$ *and* $t_1, \ldots, t_n$ *are respectively terms of sort* $T_1, \ldots, T_n$*, then* $f\ t_1 \ldots t_n$ *is a term of sort* U*.*

Propositions are built-up from predicates and terms with the connective $\Rightarrow$ and the universal quantifier $\forall$. Notice that the quantification in $\forall x.A$ is implicitly restricted over the sort of the variable x.

Definition 2.3 (Propositions)**.**
Given a language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$*, the set of propositions of minimal natural deduction based on this language is defined as follows:*

- *If* P *is a predicate symbol of rank* $\langle T_1, \ldots, T_n \rangle$ *and* $t_1, \ldots, t_n$ *are respectively terms of sort* $T_1, \ldots, T_n$*, then* $P\ t_1 \ldots t_n$ *is an* atomic proposition*.*
- *If* A *and* B *are two propositions, then* $A \Rightarrow B$ *is a proposition.*

- *If A is a proposition and x is a variable, then $\forall x.A$ is a proposition (we say that x is bound in A by the universal quantifier $\forall$ and we consider propositions equal modulo renaming of bound variables of the same sort).*

Definition 2.4 (Free variables)**.**
The set $FV(t)$ of free variables of a term t is the set of variables which appear in t. For all propositions A, we define its set free variables $FV(A)$ as the set of occurrences of variables which appear in A and which are not bound by a universal quantifier:

- $FV(P\ t_1 \dots t_n) = FV(t_1) \cup \dots \cup FV(t_n)$
- $FV(A \Rightarrow B) = FV(A) \cup FV(B)$
- $FV(\forall x.A) = FV(A) - \{x\}$

A proposition (resp. a term) is closed *if it does not contain free variables.*

Definition 2.5 (Substitution)**.**
The substitution $(t/x)t'$ of a variable x by a term t of the same sort (such that x does not appear free in t) in a term t' is defined by induction on the structure of t' as:

- $(t/x)y = t$, *if* $x = y$ *and* y *otherwise*
- $(t/x)(f\ t_1 \dots t_n) = f\ (t/x)t_1 \dots (t/x)t_n$

The substitution $(t/x)A$ of a variable x by a term t of the same sort (such that x does not appear free in t) in a proposition A is defined by induction on the structure of A as:

- $(t/x)(P\ t_1 \dots t_n) = P\ (t/x)t_1 \dots (t/x)t_n$
- $(t/x)(A \Rightarrow B) = (t/x)A \Rightarrow (t/x)B$
- $(t/x)(\forall y.A) = \forall y.\ (t/x)A$
  Notice that we suppose here that $x \neq y$ and $y \notin FV(t)$ (y has to be first renamed otherwise).

Remark 2.1. *With this last assumption we can prove that if a variable x does not appear free in a term t' (resp. a proposition A), then for all terms t of the same sort as x, we have $(t/x)t' = t'$ (resp. $(t/x)A = A$).*

In order to build proofs of propositions of minimal natural deduction, we need to define the notion of *context* which corresponds to the propositions we assume in order to prove another one.

Definition 2.6 (Contexts)**.**
Contexts are finite lists of propositions $A_1, \ldots, A_n$ *with* $n \in \mathbb{N}$*.*
We extend the notion of free variables to contexts as follows:
if $\Gamma = A_1, \ldots, A_n$*, then* $FV(\Gamma) = FV(A_1) \cup \ldots \cup FV(A_n)$*.*

Definition 2.7 ($\subseteq$)**.** *For all contexts* Γ, Γ'*, we write* $\Gamma \subseteq \Gamma'$ *if and only if each declaration in* Γ *also appears in* Γ'*.*

Then we define the notion of proof judgement which allows to express which propositions are true (i.e. provable) in minimal natural deduction based on a predicates language.

Definition 2.8 (Proof judgement)**.**
A proof judgement is given by a context Γ *and a proposition* A*. The proof judgement* $\Gamma \vdash A$ *expresses that* A *is true assuming that all propositions in* Γ *are true.*

In order to build such proof judgements, we define inference rules allowing to build proof judgements from other proof judgements or from the propositions which are assumed in the considered context.

Definition 2.9 (Inference rules)**.**
In minimal natural deduction, we define five different inference rules. The proof judgements on top of such a rule are called premises*. The proof judgement at the bottom is called* goal*.*

- *The axiom rule expresses the fact that if a proposition is assumed then it is true.*

$$\frac{A \in \Gamma}{\Gamma \vdash A} \quad (\text{AXIOM})$$

- *The* $\Rightarrow$*-introduction rule expresses how to build a proof of* $A \Rightarrow B$ *from a proof of* B *when* A *is assumed.*

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \quad (\Rightarrow \text{-INTRO})$$

- *The* $\Rightarrow$*-elimination rule expresses how to build a proof of* B *from proofs of* A *and* $A \Rightarrow B$*.*

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash A \Rightarrow B}{\Gamma \vdash B} \quad (\Rightarrow \text{-ELIM})$$

- *The $\forall$-introduction rule expresses how to build a proof of $\forall x.A$ from a proof of A with x chosen abritrarily.*

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\, x \notin FV(\Gamma) \quad (\forall\text{-INTRO})$$

- *The $\forall$-elimination rule expresses how to instantiate a proof of $\forall x.A$ into a proof of $(t/x)A$ for all terms t of same sort as x.*

$$\frac{\Gamma \vdash \forall x.A}{\Gamma \vdash (t/x)A}\, t \text{ has the same sort as } x \quad (\forall\text{-ELIM})$$

We call *proof derivations* trees formed with these rules.

Example 2.1. *Here is a proof derivation that for all propositions A, $A \Rightarrow A$ is true in any context (and, in particular, in the empty context):*

$$\dfrac{\dfrac{}{A \vdash A}\,\text{axiom}}{\vdash A \Rightarrow A}\,\Rightarrow\text{-intro}$$

Definition 2.10 (Theory)**.**
A theory *expressed in minimal natural deduction is given by a language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ and a context. We call* axioms *the elements of this context.*

2.1.2 Proof-terms and typing rules

Following the Brouwer-Heyting-Kolmogorov correspondence, we use the syntax of λ-calculus to define proof constructors, which we shall call proof-terms. For example, we represent the constructor which builds a proof of $A \Rightarrow B$ as the function which forms a proof of B when applied to a proof of A.

In this language, proof-terms can contain both term variables (written $x, y, \ldots$) and proof variables (written $\alpha, \beta, \ldots$). We call $\mathcal{X}$ the set of proof variables and $\mathcal{Y}$ the set of term variables. Notice that $\mathcal{X}$ and $\mathcal{Y}$ have no common element. Terms are written $t, u, \ldots$ while proof-terms are written $\pi, \rho, \ldots$

Definition 2.11 (Proof-terms)**.**
We call $\mathcal{T}$, the set of proof-terms: $\pi := \alpha \mid \lambda\alpha.\pi \mid \pi\ \pi' \mid \lambda x.\pi \mid \pi\ t$

Each proof-term construction corresponds to a natural deduction inference rule: proof-terms of the form α express proofs built with the axiom rule, proof-terms of the form $\lambda\alpha.\pi$ and $(\pi\ \pi')$ express proofs built respectively with the introduction and elimination rules of the implication and proof-terms of the form $\lambda x.\pi$ and $(\pi\ t)$ express proofs built with the introduction and elimination rules of the universal quantifier. This correspondence will be revealed when defining typing rules of minimal natural deduction in the following.

Notice that variables α and x are bound in the constructions $\lambda\alpha.\pi$, and $\lambda x.\pi$. We therefore extend the notion of free-variables to proof-terms in the following way.

Definition 2.12 (Free variables in proof-terms)**.**
The set of free term-variables of a proof-term is defined inductively as follows:

- $FV(\alpha) = \emptyset$
- $FV(\lambda\alpha.\pi) = FV(\pi)$
- $FV(\pi\pi') = FV(\pi) \cup FV(\pi')$
- $FV(\lambda x.\pi) = FV(\pi) - \{x\}$
- $FV(\pi t) = FV(\pi) \cup FV(t)$

And the set of free proof-variables of a proof-term as follows:

- $FV_p(\alpha) = \{\alpha\}$
- $FV_p(\lambda\alpha.\pi) = FV_p(\pi) - \{\alpha\}$
- $FV_p(\pi\pi') = FV_p(\pi) \cup FV_p(\pi')$
- $FV_p(\lambda x.\pi) = FV_p(\pi)$
- $FV_p(\pi t) = FV_p(\pi)$

Notice that we shall also write $FV(\pi)$ for the set of free proof-variables of the proof-term π, when it is not ambiguous.

We also extend the notion of substitution on proof-terms as follows. Notice that we can substitute term-variables by terms of the same sort in proof-terms, as previously, and that we can also substitute proof-variables by proof-terms in proof-terms. We shall use the same notation for those two notions of substitution.

Definition 2.13 (Substitution)**.**
For all proof-terms π, term-variables x and terms t of the same sort as x, we define $(t/x)\pi$, the substitution of x by t in π inductively as follows:

- $(t/x)\alpha = \alpha$
- $(t/x)(\lambda\alpha.\pi) = \lambda\alpha.(t/x)\pi$
- $(t/x)(\pi\pi') = (t/x)\pi\ (t/x)\pi'$
- $(t/x)(\pi t') = (t/x)\pi\ (t/x)t'$
- $(t/x)(\lambda y.\pi) = \lambda y.\ (t/x)\pi$
  Notice that we suppose here that $x \neq y$
  (and we have to rename y otherwise).

For all proof-terms π, ρ and proof-variables α, we define $(\rho/\alpha)\pi$, the substitution of α by ρ in π inductively as follows:

- $(\rho/\alpha)\beta = \rho$ *if* $\alpha = \beta$ *and* β *otherwise*
- $(\rho/\alpha)(\lambda\beta.\pi) = \lambda\beta.(\rho/\alpha)\pi$
  Notice that we suppose here that $\alpha \neq \beta$
  (and we have to rename β otherwise).
- $(\rho/\alpha)(\pi\pi') = (\rho/\alpha)\pi\ (\rho/\alpha)\pi'$
- $(\rho/\alpha)(\pi t') = (\rho/\alpha)\pi\ t'$
- $(\rho/\alpha)(\lambda x.\pi) = \lambda x.\ (\rho/\alpha)\pi$
  Notice that we suppose here that $x \notin FV(\rho)$
  (and we have to rename x otherwise).

Given this notion of proof-term, we can view the inference rules we defined previously as typing rules on proof-terms, and proof judgements as typing judgements. We have therefore to refine our definition of contexts as follows.

Definition 2.14 (Contexts)**.**
A context is a list of declarations $\alpha : A$ where α is a proof-variable and A is a proposition, such that a proof-variable cannot be declared twice or more in such a list (we consider, this way, only well-formed *contexts).*

Definition 2.15 (Typing judgements)**.**
A typing judgement is given by a context Γ, a proof-term π and a proposition A. The typing judgement $\Gamma \vdash \pi : A$ expresses that π is of type A in the context Γ (or in other words, that π is a proof of the proposition A in the context Γ).

Definition 2.16 (Typing rules)**.**
We refine the inference rules previously defined in order to obtain typing rules on proof-terms:

- *The axiom rule expresses the fact that if a proof-variable is assumed to be a proof of a proposition then it is a proof of this proposition.*

$$\frac{\alpha : A \in \Gamma}{\Gamma \vdash \alpha : A} \quad (\text{AXIOM})$$

- *The $\Rightarrow$-introduction rule expresses that if π is a proof of a proposition B when assuming that α is a proof of A, then $\lambda\alpha.\pi$ is a proof of $A \Rightarrow B$.*

$$\frac{\Gamma, \alpha : A \vdash \pi : B}{\Gamma \vdash \lambda\alpha.\pi : A \Rightarrow B} \quad (\Rightarrow \text{-INTRO})$$

- *The $\Rightarrow$-elimination rule expresses that if, in the same context, π is a proof of $A \Rightarrow B$ and π' is a proof of A, then $\pi\pi'$ is a proof of B.*

$$\frac{\Gamma \vdash \pi' : A \qquad \Gamma \vdash \pi : A \Rightarrow B}{\Gamma \vdash \pi\pi' : B} \quad (\Rightarrow \text{-ELIM})$$

- *The $\forall$-introduction rule expresses that if π is a proof of A then $\lambda x.\pi$ is a proof of $\forall x.A$.*

$$\frac{\Gamma \vdash \pi : A}{\Gamma \vdash \lambda x.\pi : \forall x.A}\, x \notin FV(\Gamma) \quad (\forall\text{-INTRO})$$

- *The $\forall$-elimination rule expresses that if π is a proof of $\forall x.A$ and t is a term of the same sort as x, then πt is a proof of $(t/x)A$.*

$$\frac{\Gamma \vdash \pi : \forall x.A}{\Gamma \vdash \pi t : (t/x)A}\, t \text{ has the same sort as } x \quad (\forall\text{-ELIM})$$

Trees formed with these rules are called *typing derivations* .

Example 2.2. *Here is a typing derivation that for all propositions A, $\lambda\alpha.\alpha$ is a proof of $A \Rightarrow A$ in any context (and, in particular, in the empty context):*

$$\dfrac{\dfrac{}{\alpha : A \vdash \alpha : A}\,\text{axiom}}{\vdash \lambda\alpha.\alpha : A \Rightarrow A}\,\Rightarrow\text{-intro}$$

Definition 2.17 (Well-typed proof-terms)**.**
A proof-term π a proof *of a proposition A if there exists a context Γ such that $\Gamma \vdash \pi : A$ is a derivable typing judgement (i.e. there exists a well-formed typing derivation such that its root is this typing jugement). In this case π is said to be well-typed.*

2.1.3 Cut elimination and normalization

Given the typing rules of minimal natural deduction, one can build different typing derivations which correspond to the same typing judgement. Hence one might wonder how to build a minimal tree giving the derivation of a typing judgement (for automated theorem proving, for example). And how to eliminate detours which may appear during the building of such typing derivations, detours which we call *cuts*. The notion of *cut* represents the introduction of an intermediate lemma in a proof, in order to use an instantiation of this lemma. Formally, a cut in a proof derivation is an elimination rule, whose main premise is the introduction rule of the same connective or quantifier.

Example 2.3.
In natural deduction, a cut on the connective $\Rightarrow$ *is of the form:*

$$\dfrac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\,\Rightarrow\text{-intro} \qquad \Gamma \vdash A}{\Gamma \vdash B}\,\Rightarrow\text{-elim}$$

And a cut on the universal quantifier $\forall$ *is of the form:*

$$\dfrac{\dfrac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\,\forall\text{-intro}}{\Gamma \vdash (t/x)A}\,\forall\text{-elim}$$

With t *a term of same sort as* x*.*

The cut elimination property expresses the fact that we could have avoided to introduce the intermediate lemma and proved the particular case we need directly.

Definition 2.18 (Cut-free proofs)**.**
We call cut-free *those proof derivations which do not contain a cut.*

The main property, concerning cut-free proofs is that such derivations have a specific form:

Lemma 2.1.
A cut-free proof derivation of a proof judgement $\vdash A$ *in an empty context always ends with an introduction inference rule.*

Remark 2.2. *A proof derivation* ends with an introduction rule *if the last (top down) inference rule used in this proof derivation is an introduction rule.*

Proof. By induction on the length of the derivation.
This proof cannot be of length 0 because in this case it would end with the rule AXIOM and it is not possible since it is a proof derivation with an empty context. Otherwise the premises of this proof derivation are also cut-free and therefore end with an introduction rule. If this proof derivation ends with an elimination rule then it has to be the elimination rule of the same connective as the introduction rule of the main premise. It therefore forms a cut and this is contradictory.

□

Definition 2.19 (Cut elimination)**.**
A logical framework has the cut elimination property iff each proposition which has a proof derivation, also has a cut-free proof derivation.

The cut elimination property has been proved for natural deduction by Prawitz in [45], following the work of Gentzen for sequent calculus [24]. The cut elimination property induces a lot of properties for a logical framework. In particular, in minimal natural deduction, it entails its consistency and completeness of automated theorem proving methods like tableau method. In deduction modulo, it also entails the disjunction property, the witness property, and so on...

As for many logical frameworks, we consider minimal natural deduction *consistent* if there exists propositions that are not provable. Notice that the minimal natural deduction on a language whose set of predicates is empty, is not consistent since its set of propositions is also empty.

Definition 2.20 (Consistency)**.**
Minimal natural deduction based on a language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *is* consistent *iff there exists a proposition built on this language which is not provable in the empty context (i.e. there exists a proposition* A *such that we cannot construct a proof derivation of* $\vdash A$*).*

Lemma 2.2 (Cut elimination and consistency)**.**
For all languages $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *such that* $\mathbb{P} \neq \emptyset$*, if minimal natural deduction based on* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *has the cut elimination property, then it is consistent.*

Proof. Since $\mathbb{P}$ is non-empty, there exists $P \in \mathbb{P}$ and a atomic proposition $P\ t_1 \dots t_n$. There is no cut-free proof derivation of $\vdash P\ t_1 \dots t_n$ since such a proof derivation cannot end with an introduction rule (using lemma 2.1). Since minimal natural deduction based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ has the cut elimination property, if there is no cut-free proof derivation of $\vdash P\ t_1 \dots t_n$, there is no proof at all of $\vdash P\ t_1 \dots t_n$, hence minimal natural deduction based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ is consistent.

□

Cut elimination can also be viewed as a proof-transformation process via the well-known Curry-Howard isomorphism. Indeed, a cut in a proof derivation correspond to a β-redex for the correspondent proof-terms. And the elimination of this cut is simulated by the reduction of the associated β-redex.

Definition 2.21 (β-reduction)**.**
A β-redex is a proof-term of the form $(\lambda\alpha.\pi)\pi'$ or of the fo rm $(\lambda x.\pi)t$.
The β-reduction is the relation on proof-terms defined by the rules:

$$(\lambda\alpha.\pi)\pi' \to (\pi'/\alpha)\pi$$
$$(\lambda x.\pi)t \to (t/x)\pi$$

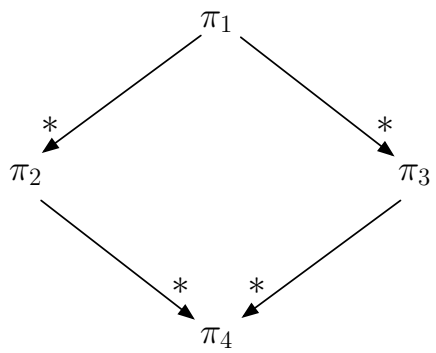
and the contextual closure:

$$\text{if } \pi \to \pi' \text{ then } \lambda\alpha.\pi \to \lambda\alpha.\pi'$$
$$\text{if } \pi_1 \to \pi_1' \text{ then } \pi_1\pi_2 \to \pi_1'\pi_2$$
$$\text{if } \pi_2 \to \pi_2' \text{ then } \pi_1\pi_2 \to \pi_1\pi_2'$$
$$\text{if } \pi \to \pi' \text{ then } \lambda x.\pi \to \lambda x.\pi'$$
$$\text{if } \pi \to \pi' \text{ then } \pi t \to \pi_1' t$$

We write $\pi \to^+ \pi'$ if π β-reduces to π' in one or more reduction steps.
We write $\pi \to^* \pi'$ if π β-reduces to π' in an arbitrary number of reduction steps.

As the elimination of a cut is simulated by the reduction of the associated β-redex, the cut elimination is entailed by the fact that all reduction-sequences from all well-typed proof-terms are finite. This property is called *strong normalization.*

Definition 2.22 (Confluence)**.**
We say that a logical framework is confluent *if and only if for all proof-terms π_1, π_2, π_3, if $\pi_1 \to^* \pi_2$ and $\pi_1 \to^* \pi_3$, then there exists a proof-term π_4 such that $\pi_2 \to^* \pi_4$ and $\pi_3 \to^* \pi_4$.*



Definition 2.23 (Normalization)**.**

- *We say that a proof-term is* normal *if it does not contain a redex.*
- *We say that a proof-term is* weakly normalizing *if there exists a reductions sequence from it which reaches a normal proof-term (hence is finite). We write WN for the set of weakly normalizing proof-terms.*
- *We say that a proof-term is* strongly normalizing *if all reductions sequences from it reach a normal proof-term. We write SN for the set of strongly normalizing proof-terms.*
- *We say that a logical framework is* weakly normalizing *if all well-typed proof-terms are weakly normalizing.*
- *We say that a logical framework is* strongly normalizing *if all well-typed proof-terms are strongly normalizing.*

Example 2.4. *We define the proof-term $\delta = \lambda\alpha.\alpha\alpha$. The proof-term $\delta\delta$ is not weakly normalizing since its only β-reduct is itself. The proof-term $(\lambda\beta.\lambda\gamma.\gamma)(\delta\delta)$ is weakly normalizing since it reduces to $\lambda\gamma.\gamma$ in one step of β-reduction, but it is not strongly normalizing since it also reduces to itself in one step of β-reduction.*

Finally we distinguish some of those proof-terms called *neutral* proof-terms.

Definition 2.24 (Neutral proof-terms)**.**
We call neutral those proof-terms that are not abstractions i.e. *that are proof-terms of the form α, $(\pi\pi')$ or (πt).*

Remark 2.3. *If π is a neutral proof-term, then for all proof-terms π', $\pi\pi'$ is not a β-redex. This property will be useful in the proofs of strong normalization we shall study.*

Definition 2.25 (Isolated proof-terms)**.**
A proof term is called isolated *if it is neutral and only β-reduces to neutral proof-terms.*

Example 2.5. *For all proof-variables α, the proof-terms α, $\delta\delta$ and $(\lambda\beta.\beta)\alpha$ are examples of isolated proof-terms.*

Remark 2.4. *If π is an isolated proof-term then for all proof-terms π', $\pi\pi'$ is an isolated proof-term.*

2.2 Minimal deduction modulo

There exists other methods than the one of axioms in order to express theories in minimal natural deduction. A first one, coming from Prawitz, consists in replacing these axioms by new inference rules. This method is used, for example, in the method of Minimal Generic Quantification [2] and in the logical frameworks of Logic with Definitions [33] and Superdeduction [52]. Another method consists in replacing these axioms by rewrite rules. This method is used, for example, in the logical framework of Deduction modulo.

Deduction modulo [17] is a logical framework which allows to express theories via both axioms and rewrite rules, and therefore gives a formal account of the difference between deduction and computation in mathematical proofs. It allows to express proofs of many theories like arithmetic [21], simple type theory [18], some variants of set theory [19], and so on...

2.2.1 Rewrite rules versus axioms

In deduction modulo a theory is formed with a set of axioms and a congruence relation, often defined by a set of rewrite rules. For instance, when defining arithmetic in deduction modulo, we can either take the usual axiom $\forall x.\ x + 0 = x$ or orient this axiom as a rewrite rule $x+0 \rightarrow x$, preserving provability. In this case the rule rewrites the term $x+0$ into the term x. Another solution is to consider a rewrite rule on propositions as: $x + 0 = x \rightarrow A \Rightarrow A$ (where A is a proposition, since $A \Rightarrow A$ is derivable in any context as we have seen in section 2.1.1).

Axioms and rewrite rules play different roles in proofs and the structure of proofs is deeply affected when theories are expressed as rewrite rules. In particular, the cut elimination property may hold for one formulation of an axiom as a rewrite rule, but not for another one. Cut elimination is a much more powerful result (and hence more difficult to prove) when theories are expressed via rewrite rules: G. Burel has recently proved that cut elimination is an undecidable property in deduction modulo [8]. In particular, for axiom-free theories, cut elimination implies consistency as in minimal natural deduction,. Indeed, in the case of theories expressed with axioms, we cannot prove anymore that a cut-free proof always ends with an introduction rule, since it can use one of the axioms of the considered context. Whereas in axiom-free theories, we can still prove that all cut-free proofs in an empty context end with an introduction rule. Notice that when theories are expressed via inference rules, as in Superdeduction, we can neither prove that a cut-free proof always ends with an introduction rule, since it can use one of the introduced inference rules.

Another important point is that deduction modulo permits to define a uniform notion of cut for all theories that can be presented by rewrite rules only. This notion of cut is the one we have defined in section 2.1.3

and subsumes that of Church's simple type theory (also-called higher-order logic) since simple type theory can be defined as an axiom free theory modulo. More generally, it subsumes the notion of cut introduced by Prawitz [45] and used for various theories, in particular for set theory [10, 29, 3, 11, 22].

In chapter 3, we shall focus on rewriting and will consider theories expressed without axioms turned into inference rules. Moreover, we shall only consider theories expressed with rewrite rules on propositions (and not on terms). We shall see in section 2.2.3 how (minimal) simple type theory and arithmetic can be expressed this way in (minimal) deduction modulo.

2.2.2 Definition

We shall study, in this work (minimal) deduction modulo which is based on (minimal) natural deduction, but there also exists modulo versions of Gentzen's sequent calculus (see [7] for example), $\lambda\Pi$-calculus ([14], we shall study $\lambda\Pi$-calculus modulo in chapters 4 and 5) and other logical frameworks.

We shall focus on theories expressed via rewrite rules on propositions only and not on terms. We shall not consider precisely the rewrite rules used to express theories, but directly the congruence relation generated by those rewrite rules.

Definition 2.26 (Minimal deduction modulo)**.**
Given a language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$, and a congruence $\equiv$ relation on propositions of minimal natural deduction based on this language, we define the minimal deduction modulo $\equiv$ based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ as follows:

- *the syntax of terms is the one of minimal natural deduction based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$:*
$$t \;=\; x \mid f\ t \dots t$$
- *the syntax of propositions is the one of minimal natural deduction based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$:*
$$A \;=\; P\ t \dots t \mid A \Rightarrow A \mid \forall x.A$$
- *the syntax of proof-terms is the one of minimal natural deduction based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$:*
$$\pi \;=\; \alpha \mid \lambda\alpha.\pi \mid \pi\pi \mid \lambda x.\pi \mid \pi t$$
- *the typing rules are the ones of minimal natural deduction based on $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ where $\equiv$-equivalent propositions are considered equal:*

$$\frac{\alpha : A \in \Gamma}{\Gamma \vdash \alpha : B}\ {\scriptstyle A \equiv B}\ \ (\text{AXIOM})$$

$$\frac{\Gamma, \alpha : A \vdash \pi : B}{\Gamma \vdash \lambda\alpha.\pi : C} \quad C \equiv A \Rightarrow B \quad (\Rightarrow\text{-INTRO})$$

$$\frac{\Gamma \vdash \pi' : A \qquad \Gamma \vdash \pi : C}{\Gamma \vdash \pi\pi' : B} \quad C \equiv A \Rightarrow B \quad (\Rightarrow\text{-ELIM})$$

$$\frac{\Gamma \vdash \pi : A}{\Gamma \vdash \lambda x.\pi : B} \quad x \notin FV(\Gamma), \quad B \equiv \forall x.A \quad (\forall\text{-INTRO})$$

$$\frac{\Gamma \vdash \pi : B}{\Gamma \vdash \pi t : C} \quad t \text{ has the same sort as } x, \quad B \equiv \forall x.A, \quad C \equiv (t/x)A \quad (\forall\text{-ELIM})$$

The main point is that we can replace a proposition by a equivalent one at any place in a typing derivation. That is how we can use rewrite rules to express axioms instead of inference rules.

Definition 2.27 (Theory)**.**
A theory *expressed in minimal deduction modulo is given by a language* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *and a congruence relation* $\equiv$ *on propositions of minimal natural deduction based on this language. We write* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ *this theory.*

Let us now prove the so-called *subject-reduction* property: all β-reducts of a proof of a proposition A, are also proofs of A. We need, for this purpose, a first lemma about (well-typed) substitution in typing judgements.

Lemma 2.3 (Substitution in typing jugements)**.**
For all contexts Γ*, propositions* A, B*, term-variables* x*, terms* t *of same sort as* x*, proof-terms* π, π' *and proof-variables* α*,*

1. *if* $\Gamma, \alpha : B \vdash \pi : A$ *and* $\Gamma \vdash \pi' : B$ *then* $\Gamma \vdash (\pi'/\alpha)\pi : A$*,*
2. *if* $\Gamma \vdash \pi : A$ *then* $\Gamma \vdash (t/x)\pi : (t/x)A$*.*

Proof. 1. By induction of the structure of π. If π is a proof-variable β then either $(\pi'/\alpha)\beta = \beta$ if $\alpha \neq \beta$ or $(\pi'/\alpha)\beta = \pi'$ if $\alpha = \beta$. In both cases we have $\Gamma \vdash (\pi'/\alpha)\pi : A$. And if π is a (proof- or term-) application or a (proof- or term-) abstraction, we conclude by induction hypothesis.

2. By induction of the structure of π, remarking that if x is free in A then it is also free in π. If π is a proof-variable β then $(t/x)\pi = \pi$. And if π is a (proof- or term-) application or a (proof- or term-) abstraction, we conclude by induction hypothesis.

□

Lemma 2.4 (Subject-reduction)**.**
For all contexts Γ*, propositions* A *and proof-terms* π, π'*,*
if $\Gamma \vdash \pi : A$ *and* $\pi \to \pi'$ *then* $\Gamma \vdash \pi' : A$*.*

Proof. By induction on the place in π of the β-redex reduced in $\pi \to \pi'$.

- If $\pi = \lambda\alpha.\pi_1$ and $\pi' = \lambda\alpha.\pi'_1$ with $\pi_1 \to \pi'_1$, we conclude by induction hypothesis.
- If $\pi = \pi_1\pi_2$ and $\pi' = \pi'_1\pi_2$ with $\pi_1 \to \pi'_1$ or $\pi' = \pi_1\pi'_2$ with $\pi_2 \to \pi'_2$, we conclude by induction hypothesis.
- If $\pi = \lambda x.\pi_1$ and $\pi' = \lambda x.\pi'_1$ with $\pi_1 \to \pi'_1$, we conclude by induction hypothesis.
- If $\pi = \pi_1 t$ with $\pi' = \pi'_1 t$ and $\pi_1 \to \pi'$, we conclude by induction hypothesis.
- Otherwise,
  - either $\pi = (\lambda\alpha.\pi_1)\pi_2$ and $\pi' = (\pi_2/\alpha)\pi_1$. By case on the last rule used in $\Gamma \vdash \pi : A$, there exists propositions B and C such that $\Gamma \vdash \lambda\alpha.\pi_1 : B \Rightarrow C$, $\Gamma \vdash \pi_2 : B$ and $A \equiv C$. Hence $\Gamma \vdash \pi' = (\pi_2/\alpha)\pi_1 : C \equiv A$ by lemma 2.3.
  - or $\pi = (\lambda x.\pi_1)t$ and $\pi' = (t/x)\pi_1$. By case on the last rule used in $\Gamma \vdash \pi : A$, there exists a proposition B such that $\Gamma \vdash \lambda x.\pi_1 : \forall x.B$ and $A \equiv (t/x)B$t.
    Hence $\Gamma \vdash \pi' = (t/x)\pi_1 : (t/x)B \equiv A$ by lemma 2.3.

□

2.2.3 Theories expressed in minimal deduction modulo

In this subsection, we propose two examples of theories expressed only via rewrite rules in (minimal) deduction modulo. We focus here on minimal simple type theory and arithmetic.

Minimal simple type theory

Intentional simple type theory [9] can be expressed in deduction modulo this way (see [18]). We show, in the following, a subset of this embedding: how to express minimal (intentional) simple type theory in minimal deduction modulo.

The language is composed of:

the *sorts* which are are *simple types* inductively defined by:

- ι and o are sorts,
- if T and U are sorts then $T \to U$ is a sort.

the individual symbols

- $S_{T,U,V}$ of sort $(T \to U \to V) \to (T \to U) \to T \to V$,
- $K_{T,U}$ of sort $T \to U \to T$,
- $\dot{\Rightarrow}$, of sort $o \to o \to o$,
- $\dot{\forall}_T$ of sort $(T \to o) \to o$,

the function symbols $\alpha_{T,U}$ of rank $\langle T \to U, T, U\rangle$,

and the predicate symbol ε of rank $\langle o\rangle$.

The combinators $S_{T,U,V}$ and $K_{T,U}$ are used to express functions. The terms $\dot{\Rightarrow}$, and $\dot{\forall}_T$ enable the representation of propositions as terms of sort o. Finally, the predicate ε allows to transform such an object t of type o into the actual corresponding proposition $\varepsilon(t)$.

$$\begin{aligned}
\alpha(\alpha(\alpha(S_{T,U,V}, x), y), z) &\to \alpha(\alpha(x, z), \alpha(y, z)) \\
\alpha(\alpha(K_{T,U}, x), y) &\to x \\
\varepsilon(\alpha(\alpha(\dot{\Rightarrow}, x), y)) &\to \varepsilon(x) \Rightarrow \varepsilon(y) \\
\varepsilon(\alpha(\dot{\forall}, x)) &\to \forall y\ \varepsilon(\alpha(x, y))
\end{aligned}$$

Arithmetic

Arithmetic can also be expressed in minimal deduction modulo (with two additional connectives $\top$ and $\bot$ which represent "True" and "False") as follows. See [21] for details.

The language is composed of:

the *sorts* ι and κ,

the constant 0 of sort ι,

the function symbols S and *Pred* of rank $\langle \iota, \iota\rangle$ and $+$ and $\times$ of rank $\langle \iota, \iota, \iota\rangle$,

the predicate symbols $=$ of rank $\langle \iota, \iota\rangle$, *Null* and N of rank $\langle \iota\rangle$ and $\in$ of rank $\langle \iota, \kappa\rangle$ and for each formula P in the language 0, S, *Pred*, $+$, $\times$, $=$, *Null* and N and whose free variables are among x, $y_1, \ldots, y_n$ of sort ι, the function symbol $f_{x,y_1,\ldots,y_n,P}$ of rank $\langle \iota, \ldots, \iota, \kappa\rangle$.

The rewrite rules are

$$x \in f_{x,y_1,\ldots,y_n,P}(y_1,\ldots,y_n) \longrightarrow P$$

$$y = z \longrightarrow \forall p\ (y \in p \Rightarrow z \in p)$$

$$N(n) \longrightarrow \forall p\ (0 \in p \Rightarrow \forall y\ (N(y) \Rightarrow y \in p \Rightarrow S(y) \in p) \Rightarrow n \in p)$$

$$\mathit{Pred}(0) \longrightarrow 0$$

$$\mathit{Pred}(S(x)) \longrightarrow x$$

$$\mathit{Null}(0) \longrightarrow \top$$

$$\mathit{Null}(S(x)) \longrightarrow \bot$$

$$0 + y \longrightarrow y$$

$$S(x) + y \longrightarrow S(x + y)$$

$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow x \times y + y$$

2.3 Reducibility candidates

Two main techniques have been developed to prove the cut elimination property of a logical framework. The first one is (apparently) purely syntactical and follows the work of Gentzen for its sequents calculus. This work has been extended by Prawitz, Tait (realisability) and Girard (reducibility candidates) to obtain a method to prove normalization of proof-terms. The second one is a semantical method developed by Beth, Hintikka, Kanger and Schutte, using the notion of model and proving that if a proposition is true in all models then it has a cut-free proof. This method has then been used by Tait [47], Prawitz [45], Takahashi [49] and Andrews [1] to prove cut elimination for simple type theory. It has been generalized, more recently, by De Marco and Lipton [15] to prove cut elimination for an intuitionistic variant of simple type theory, by Hermant [34, 36] to prove cut elimination for classical and intuitionistic theories in deduction modulo and by Okada [43] to prove cut elimination for intuitionistic linear logic.

In this section, we shall study the first method and we shall see, in the next sections, how those two methods can be unified, by introducing the notions of *pre-models* and then *truth values algebras*.

2.3.1 About reducibility candidates

The main idea of reducibility candidates is to associate to each proposition A a set of proof-terms called $\mathcal{R}_A$ containing only strongly normalizing terms and then prove the so-called *adequacy lemma*: if a proof-term π is a proof of A (there exists a context Γ such that $\Gamma \vdash \pi : A$) then it is in $\mathcal{R}_A$ and therefore strongly normalizing. In order to prove this adequacy lemma, we need that the sets $\mathcal{R}_A$ satisfy other properties than containing only strongly normalizing proof-terms, as we shall see in the following.

The proof of this adequacy lemma is done by induction on the length of the typing derivation $\Gamma \vdash \pi : A$. Let us describe how it works for simply-typed λ-calculus (we consider natural deduction with only one connective : $\Rightarrow$).

We actually need a more precise formulation of the adequacy lemma as: if $\Gamma \vdash \pi : A$ then if σ is a substitution such that for all proof-variables α declared of type B in Γ, $\sigma\alpha \in \mathcal{R}_B$, then $\sigma\pi \in \mathcal{R}_A$ (notice that in this case, if such a substitution σ exists, then $\pi \in SN$ since $\sigma\pi \in SN$). In order to prove this statement, we reason by case on the last typing rule used in the typing derivation $\Gamma \vdash \pi : A$.

- If this last rule is the rule AXIOM then π is a proof variable declared of type A in Γ therefore $\sigma\pi \in \mathcal{R}_A$ by hypothesis.

- If this last rule is the rule $\Rightarrow$-ELIM then π has the form $\pi_1\pi_2$ and we know, by induction hypothesis that there exists a proposition B such that $\sigma\pi_1 \in \mathcal{R}_{B\Rightarrow A}$ and $\sigma\pi_2 \in \mathcal{R}_B$. In order to prove that $\sigma(\pi_1\pi_2)$ is therefore in $\mathcal{R}_A$ we suppose another property on the sets $\mathcal{R}_C$, for all propositions C: we suppose that for all propositions D and E, $\mathcal{R}_{D\Rightarrow E}$ is exactly the set of proof-terms ρ such that for all ρ' in $\mathcal{R}_D$, $\rho\rho'$ is in $\mathcal{R}_E$. Provided that the sets $\mathcal{R}_C$ satisfy this property, we can conclude, in this case, that $\sigma(\pi_1\pi_2) = \sigma\pi_1\ \sigma\pi_2 \in \mathcal{R}_A$.

- If this last rule is the rule $\Rightarrow$-INTRO then π has the form $\lambda\alpha.\pi_1$, A has the form $B \Rightarrow C$ and we know, by induction hypothesis that for all $\rho \in \mathcal{R}_B$, $(\rho/\alpha)\sigma\pi_1 \in \mathcal{R}_C$. But we want that for all $\rho \in \mathcal{R}_B$, $\sigma(\lambda\alpha.\pi_1)\ \rho \in \mathcal{R}_C$. We can notice that $(\rho/\alpha)\sigma\pi_1$ is a β-reduct of $\sigma(\lambda\alpha.\pi_1)\ \rho$. This leads to make another assumption on the sets $\mathcal{R}_D$, for all propositions D: we suppose that if a proof-term is neutral and all its β-reducts are in $\mathcal{R}_D$ then it is also in $\mathcal{R}_D$ (notice that we make an assumption on *all* β-reducts of a neutral proof-term, not on only one β-reduct, since we want this property to be adequate for strong normalization and not for weak normalization). Then, in order to prove that $\sigma(\lambda\alpha.\pi_1)\ \rho$ is in $\mathcal{R}_C$ we reason on the sum of maximal lengths of reductions sequences from $\sigma(\lambda\alpha.\pi_1)$ and ρ which are both strongly normalizing since they belong to $\mathcal{R}_{B\Rightarrow C}$ and $\mathcal{R}_B$ respectively. If the

considered β-reduct of $\sigma(\lambda\alpha.\pi_1)\ \rho$ is $(\rho/\alpha)\sigma\pi_1$, we can conclude by induction hypothesis on the length of the typing derivation. Otherwise, in order to be able to conclude by induction hypothesis on the maximal lenghts of reductions sequences if the β-redex reduced in $\sigma(\lambda\alpha.\pi_1)\ \rho$ appears either in $\sigma(\lambda\alpha.\pi_1)$ or ρ, we have to make a last assumption on the sets $\mathcal{R}_D$, for all propositions D: they are stable by β-reduction.

In sum, the additional properties we want the sets $\mathcal{R}_A$ are the following ones, in the case of the simply-typed λ-calculus: for all propositions A, $\mathcal{R}_A$ is a set of proof-terms satisfying the so-called (CR_1), (CR_2) and (CR_3) properties:

(CR_1) $\mathcal{R}_A \subseteq SN$

(CR_2) if $\pi \in \mathcal{R}_A$ and $\pi \to \pi'$ then $\pi' \in \mathcal{R}_A$

(CR_3) if π is a neutral proof-termfor whom all one-step β-reducts are in $\mathcal{R}_A$, then π is also in $\mathcal{R}_A$.

And for all propositions A, B,

$$\mathcal{R}_{A\Rightarrow B} = \{\pi \text{ such that for all } \pi' \in \mathcal{R}_A,\ \pi\pi' \in \mathcal{R}_B\}$$

In our particular case, since we are able to define such a set of reducibility candidates for all propositions (by taking the set SN for atomic propositions and the sets $\{\pi$ such that for all $\pi' \in \mathcal{R}_A,\ \pi\pi' \in \mathcal{R}_B\}$ for propositions of the form $A \Rightarrow B$), we can conclude, via the adequacy lemma, that the simply-typed λ-calculus is strongly normalizing.

Let us notice a last property on reducibility candidates: they are non-empty since they contain all neutral normal proof-terms (in particular all variables) because of the (CR_3) property. A normal neutral proof-term has no reduct therefore is in all reducibility candidates. We shall see in chapter 3 that this property can raise difficulties when trying to prove that we can build such a set of reducibility candidates for all strongly normalizing logical frameworks.

2.3.2 Pre-models for deduction modulo

As we have said, the strength of deduction modulo (with axioms turned into rewrite rules and not into inference rules) is that it allows to express powerful theories with only one notion of cut. Those theories can have the cut elimination property or not. But the fact that we can express all sorts of cuts with a single notion allows us to provide a general method to prove cut elimination, independent from the theory expressed.

In [20], Dowek and Werner have extended the notion of reducibility candidates to deduction modulo in order to provide a sufficient condition for a theory to be strongly normalizing (and therefore to have the cut elimination property) in deduction modulo. They renamed the set of reducibility candidates obtained as *pre-model* in order to explicit the convergence between the construction of such an interpretation of propositions and the construction of a model. We shall see in the next section how, in a second step, Dowek characterized those pre-models as models on some kind of algebras: *truth values algebras*.

Let us detail the particular definition of pre-models for minimal deduction modulo with axioms turned into rewite rules on propositions of minimal natural deduction. Notice that we propose, in the following, a sligthly different version of the definition of pre-model. Whereas in [20], interpretations of propositions are explicitely defined from the interpretations of atomic propositions, we only focus here on which properties interpretations of propositions have to satisfy to be a pre-model.

Definition 2.28 (Valuations)**.**
Given a set $\hat{T}$ for all sorts T of the language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$,
a valuation *is a function φ which maps variables of sort T to elements of $\hat{T}$.*
For all valuations φ, we write $\text{DOM}(\varphi)$ the set of term-variables x such that φx is defined.
For all propositions A, we write $\text{VAL}(A)$ for the set of valuations φ such that all term-variables x free in A, are in $\text{DOM}(\varphi)$.
We write $\langle x, m \rangle$ the valuation which associates m to x and we write $\varphi + \varphi'$ the concatenation of two valuations φ and φ' such that $\text{DOM}(\varphi) \cap \text{DOM}(\varphi') = \emptyset$.

Definition 2.29 (pre-models)**.**
Given a set $\hat{T}$ for all sorts T of the language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$,
a function $[\![.]\!]_.$ which associates to an ordered pair of a proposition A and a valuation φ in $\text{VAL}(A)$, a set of proof-terms is a pre-model *if and only if:*

- *for all propositions A and valuations φ in $\text{VAL}(A)$,*
  $[\![A]\!]_\varphi$ satisfies (CR_1), (CR_2) and (CR_3),
- *for all propositions A, B and valuations φ in $\text{VAL}(A) \cap \text{VAL}(B)$,*
  $[\![A \Rightarrow B]\!]_\varphi = \{\pi$ such that for all $\pi' \in [\![A]\!]_\varphi$, $\pi\pi' \in [\![B]\!]_\varphi\}$,
- *for all propositions A, term-variables x and valuations φ in $\text{VAL}(\forall x.A)$*
  $[\![\forall x.A]\!]_\varphi = \{\pi$ such that for all t of same sort T as x, $\pi t \in \cap_{m \in \hat{T}} [\![A]\!]_{\varphi + \langle x,m \rangle}\}$,
- *for all propositions A, B and valuations φ in $\text{VAL}(A) \cap \text{VAL}(B)$,*
  if $A \equiv B$ then $[\![A]\!]_\varphi = [\![B]\!]_\varphi$

Apart from the property that interpretations of universally quantified propositions have to satisfy (we shall discuss this property later), the main difference with the usual method of reducibility candidates is that the sets of proof-terms associated to two $\equiv$-equivalent propositions have to be equal. Indeed, if one can replace a proposition by an $\equiv$-equivalent one in a typing judgement, this property is necessary to prove the adequacy lemma (which contains now a quantification on valuation and becomes: if $\Gamma \vdash \pi : A$ then for all valuations $\varphi \in \textsc{Val}(A)$, substitutions θ mapping term-variables to closed terms of the same sort, and substitutions σ mapping proof-variables of type B to elements of $[\![B]\!]_\varphi$, we have $\sigma\theta\pi \in [\![A]\!]_\varphi$). In fact, if $\Gamma \vdash \pi : A$ and $A \equiv B$ then we have also $\Gamma \vdash \pi : B$, hence the adequacy lemma implies that $[\![A]\!]_\varphi \subseteq [\![B]\!]_\varphi$ and, conversely that $[\![B]\!]_\varphi \subseteq [\![A]\!]_\varphi$ since $\equiv$ is a congruence relation and is therefore symmetric. This property on $\equiv$-equivalent propositions is therefore necessary to prove the adequacy lemma and to deduce that having a pre-model is a sufficient condition for theories expressed in minimal deduction modulo to be strongly normalizing.

As a corollary, Dowek and Werner proved the cut elimination property for many theories, in particular for theories presented by a quantifier free confluent and terminating rewrite system, for theories presented by a confluent and terminating positive rewrite system and for simple type theory. They proved that we can construct a pre-model for all these theories hence they are strongly normalizing and therefore have the cut elimination property. In this work, we shall not focus on this question of which theories satisfy this sound criterion for strong normalization, but on how we can improve this criterion to make it more precise (i.e. also complete). Even so, we can notice that whereas the sets $\hat{T}$ chosen to interpret the sorts T are not always the same ones (for example we can choose $\hat{T}$ as a set of of sets of terms for simple type theory), we shall focus in this work on the cases where we can choose $\hat{T}$ directly as the set of closed terms of sort T.

Let us first, in the next section, present how Dowek gave a more algebraic view of his joint work with Werner, by defining the notion of truth values algebras.

2.4 Truth values algebras

In [16], Dowek defined a generalization of Heyting algebras which enable the distinction, in models valued in these algebras, between the computational equivalence of formulae (the congruence of deduction modulo) and the provable equivalence of formulae. We shall see how this definition of algebras also allows to see reducibility candidates as models valued in one of these algebras.

2.4.1 Definition

We shall present here, the subpart of truth values algebras (TVAs), adapted to minimal deduction modulo. We can notice that some elements of the domain of a TVA are called *positive* and represent propositions which can be deduced from the axioms only. Of course, for axiom-free theories, this subset of positive truth values becomes useless, as we shall see in our definition of LDTVAs of section 3.1.4, and we shall only consider the so-called *trivial* TVAs .

Definition 2.30 (Truth values algebra)**.**
Notice that since we consider many-sorted languages, we have to define a different interpretation of the universal quantifier for each sort of the language. We then consider, for the following, a set $\mathbb{T}$ *of sorts.*

Let $\mathcal{B}$ *be a set, whose elements are called* truth values*,*
$\mathcal{B}^+$ *be a subset of* $\mathcal{B}$*, whose elements are called* positive *truth values,*
for all $T \in \mathbb{T}$*,* $\mathcal{A}_T$ *be a subset of* $\mathbb{P}(\mathcal{B})$*,*
$\tilde{\Rightarrow}$ *be a function from* $\mathcal{B} \times \mathcal{B}$ *to* $\mathcal{B}$*,*
for all $T \in \mathbb{T}$*,* $\tilde{\forall}_T$ *be a function from* $\mathcal{A}_T$ *to* $\mathcal{B}$ *and*
The structure $\mathcal{B} = \langle \mathcal{B}, \mathcal{B}^+, \mathcal{A}_T, \tilde{\Rightarrow}, \tilde{\forall}_T \rangle$ *is said to be* a truth values algebra *if the set* $\mathcal{B}^+$ *is* closed by the intuitionistic deduction rules i.e. *if for all* a*,* b*,* c *in* $\mathcal{B}$*, sorts* $T \in \mathbb{T}$ *and* A *in* $\mathcal{A}_T$*,*

1. *if* $a \tilde{\Rightarrow} b \in \mathcal{B}^+$ *and* $a \in \mathcal{B}^+$ *then* $b \in \mathcal{B}^+$*,*
2. $a \tilde{\Rightarrow} b \tilde{\Rightarrow} a \in \mathcal{B}^+$*,*
3. $(a \tilde{\Rightarrow} b \tilde{\Rightarrow} c) \tilde{\Rightarrow} (a \tilde{\Rightarrow} b) \tilde{\Rightarrow} a \tilde{\Rightarrow} c \in \mathcal{B}^+$*,*
4. *the set* $a \tilde{\Rightarrow} A = \{a \tilde{\Rightarrow} e \mid e \in A\}$ *is in* $\mathcal{A}_T$*,*
5. *if all elements of* A *are in* $\mathcal{B}^+$ *then* $\tilde{\forall}_T\ A \in \mathcal{B}^+$*,*
6. $\tilde{\forall}_T\ (a \tilde{\Rightarrow} A) \tilde{\Rightarrow} a \tilde{\Rightarrow} (\tilde{\forall}_T\ A) \in \mathcal{B}^+$*,*
7. *if* $a \in A$*, then* $(\tilde{\forall}_T\ A) \tilde{\Rightarrow} a \in \mathcal{B}^+$*,*

Definition 2.31 (Full)**.**
A truth values algebra is said to be full *if for all sorts* $T \in \mathbb{T}$*,* $\mathcal{A}_T = \mathbb{P}(\mathcal{B})$*,* i.e. *if* $\tilde{\forall}_T$ *A exists for all subsets* A *of* $\mathcal{B}$ *ands sorts* $T \in \mathbb{T}$*.*

Definition 2.32 (Trivial)**.**
A truth values algebra is said to be trivial *if* $\mathcal{B}^+ = \mathcal{B}$*.*

Example 2.6. *Let* $\mathcal{B} = \{0, 1\}$*. Let* $\mathcal{B}^+ = \{1\}$*,* $\mathcal{A}_T = \mathbb{P}(\mathcal{B})$ *for all* $T \in \mathbb{T}$*,* $\tilde{\Rightarrow}$ *be the usual boolean operation,* $\tilde{\forall}_T$ *be the function mapping the sets* $\{0\}$ *and* $\{0, 1\}$ *to* 0 *and* $\varnothing$ *and* $\{1\}$ *to* 1 *Then* $\langle \mathcal{B}, \mathcal{B}^+, \mathcal{A}_T, \tilde{\Rightarrow}, \tilde{\forall}_T \rangle$ *is a truth values algebra.*

Example 2.7. *Let $\mathcal{B}$ be an arbitrary set, $\mathcal{B}^+ = \mathcal{B}$, $\mathcal{A}_T = \mathbb{P}(\mathcal{B})$ for all $T \in \mathbb{T}$, and $\tilde{\Rightarrow}$, $\tilde{\forall}_T$ be arbitrary operations. Then $\langle \mathcal{B}, \mathcal{B}^+, \mathcal{A}_T, \tilde{\Rightarrow}, \tilde{\forall}_T \rangle$ is a trivial and full truth values algebra. Notice that we shall only consider this kind of truth values algebras in section 3.1.4*

Notice that truth values algebras can alternatively be characterized as pseudo-Heyting algebras (i.e. Heyting algebras where the usual order of the algebra is only a pre-order (a transitive and reflexive relation on elements of the domain)). See [16] for details.

2.4.2 Models valued in truth values algebras

We detail in this subsection how we define models valued in those kinds of algebras. Unlike what we have done in section 2.3.2, we shall present the original way to build such models: inductively from interpretations of the language.

Definition 2.33 ($\mathcal{B}$-valued structure)**.**
Let $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ be a many-sorted language in predicate logic and $\mathcal{B}$ be a truth values algebra, $\langle \{\hat{T}\}_{T \in \mathbb{T}}, \{\hat{f}\}_{f \in \mathbb{F}}, \{\hat{P}\}_{P \in \mathbb{P}} \rangle$ is a $\mathcal{B}$-valued structure *for the language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$, if and only if*

- *for all $T \in \mathbb{T}$, $\hat{T}$ is a set,*
- *for all $f \in \mathbb{F}$ of rank $\langle T_1, \ldots, T_n, U \rangle$,*
  $\hat{f}$ is a function from $\hat{T}_1 \times \ldots \times \hat{T}_n$ to $\hat{U}$,
- *for all $P \in \mathbb{P}$ of rank $\langle T_1, \ldots, T_n \rangle$,*
  $\hat{P}$ is a function from $\hat{T}_1 \times \ldots \times \hat{T}_n$ to $\mathcal{B}$.

A valuation associates to term-variables of sort T, elements of $\hat{T}$.

Definition 2.34 (Valuation)**.**
Given a $\mathcal{B}$-valued structure, a valuation φ *is a function such that for all term-variables of sort T, $\varphi(x) \in \hat{T}$.*
We write $\textsc{Dom}(\varphi)$ *for the set of term-variables x such that $\varphi(x)$ is defined. For all propositions A, we write* $\textsc{Val}(A)$ *for the set of valuations such that $FV(A) \subseteq \textsc{Dom}(\varphi)$.*

Now we can define $\mathcal{B}$-valued interpretations, inductively from a $\mathcal{B}$-valued structure using the so-called *denotation* defined as follows.

Definition 2.35 (Denotation)**.**
Let $\mathcal{B}$ be a truth values algebra, $\langle \{\hat{T}\}_{T \in \mathbb{T}}, \{\hat{f}\}_{f \in \mathbb{F}}, \{\hat{P}\}_{P \in \mathbb{P}} \rangle$ be a $\mathcal{B}$-valued structure and φ be a valuation. The denotation $[\![A]\!]_\varphi$ of a formula A in $\langle \{\hat{T}\}_{T \in \mathbb{T}}, \{\hat{f}\}_{f \in \mathbb{F}}, \{\hat{P}\}_{P \in \mathbb{P}} \rangle$ is defined as follows

- $[\![x]\!]_\varphi = \varphi(x)$,
- $[\![f(t_1, ..., t_n)]\!]_\varphi = \hat{f}([\![t_1]\!]_\varphi, ..., [\![t_n]\!]_\varphi)$,
- $[\![P(t_1, ..., t_n)]\!]_\varphi = \hat{P}([\![t_1]\!]_\varphi, ..., [\![t_n]\!]_\varphi)$,
- $[\![A \Rightarrow B]\!]_\varphi = [\![A]\!]_\varphi \tilde{\Rightarrow} [\![B]\!]_\varphi$,
- $[\![\forall x.\ A]\!]_\varphi = \tilde{\forall}_T\ \{[\![A]\!]_{\varphi+\langle x,e\rangle} \mid e \in \hat{T}\}$ *(with T the sort of x)*,

Notice that the denotation of a formula containing quantifiers may be undefined, but it is always defined if the truth values algebra is full.

We can then distinguish some of these $\mathcal{B}$-valued structures which are *adapted to the congruence* (i.e. two equivalent propositions have the same denotation). We call those particular $\mathcal{B}$-valued interpretations "*models*".

Definition 2.36 (Model)**.**
Let $\equiv$ be (a congruence relation defining) a theory in minimal deduction modulo. The $\mathcal{B}$-valued structure $\langle \{\hat{T}\}_{T\in\mathbb{T}}, \{\hat{f}\}_{f\in\mathbb{F}}, \{\hat{P}\}_{P\in\mathbb{P}}\rangle$ is said to be a model of *the theory $\equiv$ if for all propositions A and B such that $A \equiv B$ and valuations φ, $[\![A]\!]_\varphi$ and $[\![B]\!]_\varphi$ are defined and $[\![A]\!]_\varphi = [\![B]\!]_\varphi$.*

Remark 2.5. *The condition that $[\![A]\!]_\varphi$ and $[\![B]\!]_\varphi$ are defined, is not necessary when considering full truth values algebras.*

2.4.3 $\mathcal{C}$, the TVA of reducibility candidates

Given this notion of truth values algebras, we are now able to define $\mathcal{C}$, the TVA of reducibility candidates, and prove, using the work done for pre-models by Dowek and Werner, that having a $\mathcal{C}$-valued model is a sound semantics for strongly normalizing theories in minimal deduction modulo.

Definition 2.37 ($\mathcal{C}$)**.**
We define $\langle \mathcal{C}, \mathcal{C}^+, \mathcal{A}, \tilde{\Rightarrow}, \tilde{\forall}\rangle$, the TVA *of reducibility candidates, as follows:*

- *$\mathcal{C}$ is the set of sets of proof-terms which satisfy the properties (CR_1), (CR_2) and (CR_3),*
- *$\mathcal{C}^+ = \mathcal{C}$*
  (or any other set of proof-terms closed by intuitionistic deduction rules),
- *$\mathcal{A} = \mathbb{P}(\mathcal{B})$,*
- *For all $E, F \in \mathcal{C}$, $E \tilde{\Rightarrow} F = \{\pi$ such that for all $\pi' \in E$, $\pi\pi' \in F\}$*
- *For all $A \in \mathcal{A}_T$,*
  $\tilde{\forall}_T A = \{\pi$ such that for all t of sort T and $a \in A$, $\pi t \in a\}$.

Given this TVA of reducibility candidates, we are able to prove, using the work done with pre-models, that if a theory in minimal deduction modulo has a model valued in $\mathcal{C}$ then it is strongly normalizing. In the next chapter, we shall study how to refine those notions of TVAs and reducibility candidates in order to provide a semantics that is, moreover, complete for strong normalization (i.e. a notion of model such that if a theory is strongly normalizing, then it has such a model).

3

Sound and complete semantics for strong normalization in minimal deduction modulo

Context

We have detailed in the previous chapter the main method for proving strong normalization in different logical frameworks, using the set of so-called reducibility candidates. This method has been adapted for the logical framework of deduction modulo by defining the notion of pre-model, which is a set of reducibility candidates such that two equivalent propositions are interpreted by the same set. Then we have seen how this technique can be viewed as the building of a model on some sort of algebras called truth values algebras. And we have exhibited the particular truth values algebras of reducibility candidates $\mathcal{C}$, concluding that having a $\mathcal{C}$-valued model is a sound semantics for strong normalization in (minimal) deduction modulo.

Contributions

The aim of this chapter is to analyse how we can refine this notion of reducibility candidates in order to provide a sound and also complete semantics for strong normalization. We focus in this chapter on the logical framework of minimal deduction modulo. We define a refinement of truth values algebras (TVA), called language-dependent truth values algebras (LDTVA). We call them *language-dependent* since this new notion of algebra depends on the sets of closed terms of the sorts of the predicate language on which we build a theory in minimal deduction modulo. Then we exhibit two of these languages-dependent truth valuas algebras such that both provide a sound and complete semantics for strong normalization via models valued in them. The method for proving completeness uses a specificity of the first LDTVA we

define: the fact that we consider only proofs of a proposition in its interpretation. We therefore define, given a theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$, the LDTVA of $\equiv$-well-typed reducibility candidates. It allows us to capture exactly the proofs of a proposition, via sets of proof-terms which satisfy analog properties as the well-known ones of reducibility candidates. We then define a more general LDTVA $\mathcal{C}'$ which is independent of the congruence relation $\equiv$ of the theory we consider . This LDTVA provides a general sound and complete semantics for strong normalization for all theories expressed in minimal deduction modulo (based on a same predicate language).

Outline

As a first step, we analyze precisely the role of the (CR_3) property of reducibility candidates, and of the interpretation of the universal quantifier in the TVA of reducibility candidates, in proofs of strong normalization. This analysis suggests that both are responsible of the difficulties involved when trying to prove that having a model valued in reducibility candidates is a complete semantics for strong normalization, when trying to prove the converse of the adequacy lemma for that purpose. We first adapt the latter by defining language dependent truth values algebras which provide a functional interpretation of the universal quantifier, and then propose two adaptations of the (CR_3) property which define two new definitions of reducibility candidates as some particular LDTVAs . While the first depends on typing and therefore on the congruence relation which defines the theory we consider, the second one is theory-independent. We prove models valued in both these LDTVAs provide two different sound and complete semantics for strong normalization in minimal deduction modulo. Completeness of models valued in the second one is entailed by completeness of models valued in the first one, by building a morphism of LDTVAs from the first one to the second one, inducing a mapping from models valued in the first one to models valued in the second one.

3.1 Are usual reducibility candidates complete ?

We saw, in the previous chapter that reducibility candidates are sound for strong normalization, in the sense that if we can build a set of reducibility candidates adapted to a logical framework then this logical framework is strongly normalizing. From a TVA point of view it means that having a $\mathcal{C}$-valued model is a sufficient condition for a theory to be strongly normalizing in minimal deduction modulo. Then we may wonder whether reducibility candidates also are complete, i.e. whether having a $\mathcal{C}$-valued model is also a necessary condition for a theory to be strongly normalizing.

In 1929, Gödel proved his completeness theorem concerning consistency of theories expressed in natural deduction. Gödel proved in [28] that besides being a sound semantics, having a $\{0,1\}$-valued model is a complete semantics for the property of consistency of theories expressed in first order logic. In other words, he exhibited a correspondence, in first order logic, between truth an provability, between model theory and proof theory. Other completeness theorems have been proved: for higher order logic by Henkin [32] and for modal logic, by Kripke [38]. They both concern sound and complete semantics for consistency.

As we saw in 2.1.3, when we express theories in minimal deduction modulo, only within the congruence relation and not with axioms, the strong normalization of the theory we consider entails its cut elimination and therefore its consistency. Hence strong normalization implies consistency. The converse is not true, since there exists theories which have the cut elimination property, but which do not strongly normalize [35]. Hence strong normalization is a strictly more powerful property than consistency. In this way, we can see the work presented in this section, as the natural following of the previous completeness theorems cited above.

As we said in 2.3.1, the main point of proofs of normalization using reducibility candidates is the adequacy lemma, which states that each proof-term which is a proof of a proposition belongs to the reducibility candidate associated to this proposition, and is therefore strongly normalizing. The first naïve idea to prove completeness of $\mathcal{C}$-valued models for strong normalization is to try to prove the converse of the adequacy lemma: each reducibility candidate associated to a proposition contains only proofs of this proposition, in order to obtain what we shall call *well-typed reducibility candidates*. We shall see, in the following that it is not possible with usual reducibility candidates, because of the definition of the interpretation of the universal quantifier in TVAs , and because of the (CR_3) property. We will therefore propose a new definition of algebras, and a modified version of the (CR_3) property in order to be able to prove the converse of the adequacy lemma.

3.1.1 About the (CR_3) property

Let us first analyze the role of the (CR_3) property in soundness of reducibility candidates. As we can notice in chapter 2, this property is used twice in proofs of normalization. First to prove that $\mathcal{C}$-models are non empty since they contain all proof-variables, and moreover, all neutral normal proof-terms. And in a second step, to prove that abstractions of type $A \Rightarrow B$ are in the interpretation of $A \Rightarrow B$. We first recall the definition of (CR_3): we say that a set E of proof-terms satisfies (CR_3) if and only if all neutral proof-terms whose all β-reducts are in E, are also in E.

Variables

As soon as a set of proof-terms satisfies (CR_3), it contains all neutral normal proof-terms, since they have no reduct. Therefore all reducibility candidates contain all neutral normal proof-terms, and, in particular, all proof-variables. We shall see in 3.1.2 that this is one of the reasons why we cannot prove the converse of the adequacy lemma with usual reducibility candidates.

Abstractions

In order to prove that an abstraction $\lambda\alpha.\pi$ of type $A \Rightarrow B$ is in the interpretation of $A \Rightarrow B$, the method is to prove that for all elements π' of the interpretation of A, $(\lambda\alpha.\pi)\pi'$ is in the interpretation of B, as you can notice in 2.3.1. The main argument is that $(\lambda\alpha.\pi)\pi'$ is neutral and all its reducts are in the interpretation of B (by induction hypothesis). We can notice that in this case, $(\lambda\alpha.\pi)\pi'$ is well-typed (of type B namely) and non-normal. Those facts will be useful when we shall modify the (CR_3) property in the following.

Those two properties are essential in the proof of the adequacy lemma. We shall have to verify that the new definitions of reducibility candidates we shall propose in the following, still satisfy those properties, in order to prove that they are still sound for strong normalization.

3.1.2 The problem of neutral normal proof-terms

The usual (CR_3) property prevents from proving the converse of the adequacy lemma: reducibility candidates do not contain only well-typed proof-terms. We say that they are *not adapted to typing*. In fact, all reducibility candidates contain all neutral normal proof-terms: as we have seen for variables, those proof-terms are neutral and do not have β-reducts. In particular, if α is a proof-variable, the proof-term $\alpha\alpha$ belongs to all reducibility candidates, and yet it cannot be well-typed in a strongly normalizing theory. Indeed, if there exists a context Γ and a proposition A such that $\Gamma \vdash \alpha\alpha : A$, then there exists propositions B and C such that $\Gamma \vdash \alpha : B$ and $\Gamma \vdash \alpha : B \Rightarrow A$, by case analysis on the last rule used in the derivation of $\Gamma \vdash \alpha\alpha : A$. Hence we also have $B \equiv\equiv B \Rightarrow A$. If we write $\delta = \lambda\alpha.\alpha\alpha$, we can then obtain $\Gamma \vdash \delta : B \Rightarrow A$, since $\Gamma \vdash \alpha : B$ and $\Gamma \vdash \alpha\alpha : A$. Moreover, since $B \Rightarrow A \equiv B$, we also have $\Gamma \vdash \delta : B$, and we obtain $\Gamma \vdash \delta\delta : A$.

If we note Π_0 the following typing judgement:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha : B \vdash \alpha : B \Rightarrow A}\,\text{axiom} \quad \dfrac{}{\Gamma, \alpha : B \vdash \alpha : B}\,\text{axiom}}{\Gamma, \alpha : B \vdash \alpha\alpha : A}\,\Rightarrow\text{-elim}}{\Gamma \vdash \lambda\alpha.\alpha\alpha : B \Rightarrow A}\,\Rightarrow\text{-intro}$$

and Π_1 this one:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha : B \vdash \alpha : B \Rightarrow A}\,\text{axiom} \quad \dfrac{}{\Gamma, \alpha : B \vdash \alpha : B}\,\text{axiom}}{\Gamma, \alpha : B \vdash \alpha\alpha : A}\,\Rightarrow\text{-elim}}{\Gamma \vdash \lambda\alpha.\alpha\alpha : B}\,\Rightarrow\text{-intro} \quad B \equiv B \Rightarrow A$$

then we obtain:

$$\dfrac{\dfrac{\Pi_0}{\Gamma \vdash \lambda\alpha.\alpha\alpha : B \Rightarrow A} \quad \dfrac{\Pi_1}{\Gamma \vdash \lambda\alpha.\alpha\alpha : B}}{\Gamma \vdash (\lambda\alpha.\alpha\alpha)(\lambda\alpha.\alpha\alpha) : A}\,\Rightarrow\text{-elim}$$

Finally, $\alpha\alpha$ cannot be well-typed in a strongly normalizing theory, otherwise $\delta\delta$ would be also well-typed, whereas it is not even weakly normalizing since the only β-reduct of $\delta\delta$ is itself.

We therefore have to modify this (CR_3) property in order to obtain a definition of well-typed reducibility candidates, but we also have to modify the interpretation of the universal quantifier as we shall see in the next subsection.

3.1.3 How to interpret the universal quantifier

In the usual Truth Values Algebra $\mathcal{C}$ of reducibility candidates, the interpretation of the universal quantifier is defined as an intersection of a set of sets of proof-terms:

$$\hat{\forall}\ \mathcal{E} = \cap\ \mathcal{E}$$

For all propositions A, term-variables x of sort T, and valuations $\varphi \in \text{Val}(\forall x.A)$, the interpretation of $\forall x.A$ and the valuation φ is obtained from all interpretations of A with the different valuations $\varphi + \langle x, t\rangle$, for $t \in \hat{T}$:

$$\begin{array}{rcl} [\![\forall x.A]\!]_\varphi & = & \hat{\forall}\{[\![A]\!]_{\varphi+\langle x,t\rangle},\ t \in \hat{T}\} \\ & = & \{\pi \text{ such that for all terms } t_1 \text{ and } t_2 \in \hat{T},\ \pi t_1 \in [\![A]\!]_{\varphi+\langle x,t_2\rangle}\} \end{array}$$

In the particular case where $\hat{T}$ is chosen as the set of closed terms of sort T, we can notice that this definition of $\hat{\forall}$ is not adapted to the notion of typing, because of the fact that t_1 and t_2 are not synchronized. If $\Gamma \vdash \pi : \forall x.A$, and t_1, t_2 are two diferent closed terms in $\hat{T}$, then we do not have necessarily $\Gamma \vdash \pi t_1 : (t_2/x)A$. For example, in arithmetic, if we consider the proposition $\forall x.\ 1 + x = x + 1$, we can build a proof of this proposition which will, applied to a term z, use $(z - 1)$ times the axiom

$y + (t + 1) = (y + t) + 1$. Of course, when we apply this proof to 2, we do not obtain a proof of $3 + 1 = 1 + 3$.

In order to make interpretation of the universal quantifier adapted to the notion of typing, we have therefore to modify the notion of Truth Values Algebras. In order to synchronize t_1 and t_2 in the previous definition, we have first to consider a different interpretation of the universal quantifier for each sort of the language, and, secondly, we have to provide a interpretation of the universal quantifier which takes in argument a function an not a set of sets, in order to define a dependent intersection:

$$[\![\forall x.A]\!]_\varphi = \{\pi \text{ such that for all } t \in \hat{T},\ \pi t \in [\![A]\!]_{\varphi+\langle x,t\rangle}\}$$

We then adapt the notion of truth values algebras by defining *language-dependent truth values algebras*, which use $\hat{\forall}$ as a dependent intersection. Those LDTVAs depend on the language as we have to know the sets of closed terms $\hat{T}$, to define such a synchronized interpretation of the universal quantifier.

3.1.4 Language-dependent truth values algebras

We define in this subsection *language-dependent truth values algebras* (LDTVA), which differ from usual truth values algebras on two points. First, as we already said in 2.2 and 2.4, we only consider theories in deduction modulo where axioms are expressed only within the congruence relation and not with axioms. Therefore we do not distinguish *positive* truth values in the domain of a LDTVA .

Secondly, we always choose $\hat{T}$ as the set of closed terms of sort T and we define the interpretation of the universal quantifier $\hat{\forall}$ as a function which maps to elements of the domain, not subsets of the domain anymore but functions from $\hat{T}$ to the domain. We therefore define a family of interpretations of the universal quantifier $(\hat{\forall}_T)_{T\in\mathbb{T}}$, which are indexed by the different sorts of the language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P}\rangle$ we consider and more precisely, and which depend on the sets of closed terms of each of these sorts. We then have to define LDTVAs , given a language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P}\rangle$, this is why we call them *language-dependent*. Notice that the definition of a LDTVA does not depend on the whole language $\langle \mathbb{T}, \mathbb{F}, \mathbb{P}\rangle$, but only on the set of closed terms of each sort in $\mathbb{T}$, hence it only depends on $\mathbb{T}$ and $\mathbb{F}$. Notice, finally, that those interpretations $\hat{\forall}_T$ cannot always be defined on the whole set of functions from $\hat{T}$ to the domain, but only on one of its subsets. We then define, for all sorts T, $\hat{\mathcal{A}}_T$, which is the subset of the set of functions from $\hat{T}$ to the domain, on which $\hat{\forall}_T$ is defined. This gives the following definition.

Definition 3.1 (Language-dependent TVAs)**.**
Let $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *be a many-sorted language in predicate logic.*
$\langle \mathcal{B}, \hat{\Rightarrow}, (\hat{\mathcal{A}}_T), (\hat{\forall}_T) \rangle$ *is a* LDTVA *for* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ *if and only if:*

- $\mathcal{B}$ *is a set (called the* domain*),*
- $\hat{\Rightarrow}$ *is a function from* $\mathcal{B} \times \mathcal{B}$ *to* $\mathcal{B}$*,*
- *for all sorts* $T \in \mathbb{T}$*,* $\hat{\mathcal{A}}_T$ *is a set of functions from* $\hat{T}$ *to* $\mathcal{B}$*,*
- *for all sorts* $T \in \mathbb{T}$*,* $\hat{\forall}_T$ *is a function from* $\hat{\mathcal{A}}_T$ *to* $\mathcal{B}$*.*

Notice that we shall write $\mathcal{B}$ both for denominating the LDTVA $\mathcal{B}$ and its domain.

In LDTVAs , we define models in a different way than in TVAs : we do not define anymore interpretations of propositions from interpretations of atomic propositions, using interpretations of connectives, and then call them models if and only if they are *adapted to the congruence* (all equivalent propositions have the same interpretation). As in our presentation of pre models of section 2.3.2, given a LDTVA $\mathcal{B}$, we define interpretations of propositions simply as functions which map every ordered pair of a proposition A and a valuation in $\text{VAL}(A)$ to an element of $\mathcal{B}$.

Definition 3.2 (Interpretations)**.** *We call* $\mathcal{B}$-valued interpretations *those functions which map every ordered pair of a proposition* A *and a valuation in* $\text{VAL}(A)$ *to an element of the domain of a* LDTVA $\mathcal{B}$*.*

Definition 3.3. *We say that an interpretation of propositions is* non empty *if and only if each interpretation of a proposition and a valuation in* $\text{VAL}(A)$ *is a non empty set.*

And then we distinguish two properties of those interpretations. First, we say that an interpretation is a *model* if it is *adapted to the connectives* (with this definition, all TVA -interpretations are models since they are built inductively from the interpretation of connectives). And then we say that such a model is a model of a theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ if and only if all $\equiv$-equivalent propositions have the same interpretation (we say that the interpretation is *adapted to the congruence*). With these new definitions, we are able to define interpretations of propositions adapted to the congruence, while not necessarily adapted to the connectives, what we could not do with the original definition of models valued in TVAs. We shall see in section 3.2 how this new technique has been useful in order to prove completeness of such models (even if a simpler proof has been exhibited later).

Definition 3.4 (Models)**.**
Let $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ *be a theory expressed in minimal deduction modulo and* $\mathcal{B} = \langle \mathcal{B}, \hat{\Rightarrow}, (\hat{\mathcal{A}}_T), (\hat{\forall}_T) \rangle$ *be a* LDTVA *for* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$.

- *A* $\mathcal{B}$*-valued interpretation* $[\![.]\!]_.$ *is a* $\mathcal{B}$-valued model *if and only if:*
  - *for all* $A, B \in \mathcal{P}$ *and* $\varphi \in \text{VAL}(A \Rightarrow B)$, $[\![A \Rightarrow B]\!]_\varphi = [\![A]\!]_\varphi \hat{\Rightarrow} [\![B]\!]_\varphi$
  - *for all* $A \in \mathcal{P}$, x *of sort* T *and* $\varphi \in \text{VAL}(\forall x.A)$ *such that* $x \notin \text{DOM}(\varphi)$, $(t \mapsto [\![\forall x.A]\!]_{\varphi + \langle x,t \rangle}) \in \hat{\mathcal{A}}_T$ *and* $[\![\forall x.A]\!]_\varphi = \hat{\forall}_T(t \mapsto [\![A]\!]_{\varphi + \langle x,t \rangle})$
  - *for all* $A \in \mathcal{P}$, x *of sort* T, $t \in \hat{T}$ *and* $\varphi \in \text{VAL}(\forall x.A)$ *such that* $x \notin \text{DOM}(\varphi)$, $[\![(t/x)A]\!]_\varphi = [\![A]\!]_{\varphi + \langle x,t \rangle}$.
- *A* $\mathcal{B}$*-valued model* $[\![.]\!]_.$ *is a* model of the theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ *if and only if: for all* $A, A' \in \mathcal{P}$, $\varphi \in \text{VAL}(A)$ *and* $\psi \in \text{VAL}(A')$, *if* $\varphi A \equiv \psi A'$, *then* $[\![A]\!]_\varphi = [\![A']\!]_\psi$

Remark 3.1. *The previous conditions can be reformulated as:*

1. *Interpretations of propositions have to be* adapted to the connectives $\Rightarrow$ *and* $\forall$ *to be a model.*
2. *Models have to be* adapted to the congruence *to be a model of the associated theory.*

We finally define a notion of morphism on LDTVAs such that a morphism between two LDTVAs $\mathcal{B}_1$ and $\mathcal{B}_2$ based on a same language, maps $\mathcal{B}_1$-valued models of a theory to $\mathcal{B}_2$-valued models of the same theory.

Definition 3.5 (Morphism)**.**
Let $\mathcal{B}^1 = \langle \mathcal{B}^1, \hat{\Rightarrow}^1, (\hat{\mathcal{A}}^1_T), (\hat{\forall}^1_T) \rangle$ *and* $\mathcal{B}^2 = \langle \mathcal{B}^2, \hat{\Rightarrow}^2, (\hat{\mathcal{A}}^2_T), (\hat{\forall}^2_T) \rangle$ *be two* LDTVA*s.* *A* morphism *from* $\mathcal{B}^1$ *to* $\mathcal{B}^2$ *is a function* F *from* $\mathcal{B}^1$ *to* $\mathcal{B}^2$ *such that:*

- *for all* $E, G \in \mathcal{B}^1$, $F(E \hat{\Rightarrow}^1 G) = F(E) \hat{\Rightarrow}^2 F(G)$,
- *for all sorts* T *and* $f \in \hat{\mathcal{A}}^1_T$, $F \circ f \in \hat{\mathcal{A}}^2_T$,
- *for all sorts* T *and* $f \in \hat{\mathcal{A}}^1_T$, $F(\hat{\forall}^1_T f) = \hat{\forall}^2_T (F \circ f)$.

Lemma 3.1. *For all* LDTVA*s* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *and morphisms* F *from* $\mathcal{B}_1$ *to* $\mathcal{B}_2$*, if* $[\![.]\!]_.$ *is a* $\mathcal{B}_1$*-valued model of a theory* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$*, then* $F \circ [\![.]\!]_.$ *is a* $\mathcal{B}_2$*-valued model of* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$*.*

Proof. We can first notice, that if $[\![.]\!]_.$ is a $\mathcal{B}_1$-valued interpretation and F is a function from $\mathcal{B}_1$ to $\mathcal{B}_2$ then $F \circ [\![.]\!]_.$ is a $\mathcal{B}_2$-valued interpretation. If $[\![.]\!]_.$ is adapted to the congruence then so does $F \circ [\![.]\!]_.$ since F is a function. Finally, if $[\![.]\!]_.$ is adapted to the connectives and F is a morphism then $F \circ [\![.]\!]_.$ is also adapted to the connectives through the definition of morphisms.

□

3.2 Well-typed reducibility candidates

We show in this section, given a theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ expressed in minimal deduction modulo, how to define the LDTVA $\mathcal{C}_{\equiv}$ of $\equiv$-well-typed reducibility candidates: reducibility candidates such that the reducibility candidate associated to a proposition A only contains proof-terms which are proofs of A. We set, for the following, a theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ expressed in minimal deduction modulo and we shall define the LDTVA $\mathcal{C}_{\equiv}$ associated to this theory.

3.2.1 $\mathcal{C}_{\equiv}$, the LDTVA of $\equiv$-well-typed reducibility candidates

As a typing judgement $\Gamma \vdash \pi : A$ associates an ordered pair formed by a context Γ and a proof π , to a proposition A, we shall consider, for the domain of $\mathcal{C}_{\equiv}$, the set of sets of such pairs, and interpret propositions by the pairs that are associated to. We write $\mathcal{U}$ the set of such pairs.

Definition 3.6 ($\mathcal{U}$)**.**
$\mathcal{U} = \{(\Gamma, \pi)$ *such that* Γ *is a context and* π *is a proof-term* $\}$*.*

We shall consider, for the domain of $\mathcal{C}_{\equiv}$ (which we shall also call $\mathcal{C}_{\equiv}$), the set of subsets of $\mathcal{U}$ which verify adapted versions of the usual properties (CR_1) and (CR_2) of reducibility candidates, a modified version of (CR_3) and another property ($\mathrm{CR}_{\equiv}$), which both express well-typing.

($\mathrm{CR}_{\equiv}$) expresses the fact that each reducibility candidate E is associated to a proposition A_E, and that all elements (Γ, π) of E satisfy $\Gamma \vdash \pi : A_E$.

In order to avoid proof-terms that are not well-typed in those new reducibility candidates, we propose a modified version of (CR_3): ($\mathrm{CR}_{3_{\equiv}}$), which excludes explicitely proof-terms that are not well-typed, *i.e.* a pair (Γ, π) such that π is neutral and all its one-step reducts are in a reducibility candidate E, can be added to E only if we have $\Gamma \vdash \pi : A_E$.

Definition 3.7.
For all $E \subseteq \mathcal{U}$, we define the following properties :

($CR_{\equiv}$) *There exists A_E such that $\forall(\Gamma, \pi) \in E,\ \Gamma \vdash \pi : A_E$*

($CR_{1_{\equiv}}$) *For all $(\Gamma, \pi) \in E$, $\pi \in SN$*

($CR_{2_{\equiv}}$) *For all $(\Gamma, \pi) \in E$, and $\pi' \in \mathcal{T}$ such that $\pi \to \pi'$, $(\Gamma, \pi') \in E$*

($CR_{3_{\equiv}}$) *For all $(\Gamma, \pi) \in \mathcal{U}$ such that π is neutral and $\Gamma \vdash \pi : A_E$, if for all one-step reducts τ of π, $(\Gamma, \tau) \in E$, then $(\Gamma, \pi) \in E$.*

Remark 3.2. *We could have merged the definitions of ($CR_{\equiv}$) and ($CR_{3_{\equiv}}$) as satisfying ($CR_{3_{\equiv}}$) has no sense for a set if it does not satisfy ($CR_{\equiv}$): if we want to restrict the (CR_3) expansion to well-typed proof-terms, we have to associate to each reducibility candidate a proposition, and use our new ($CR_{3_{\equiv}}$) property only on neutral proof-terms which are proofs of this proposition.*

As explained in the following lemma, these properties define non-empty sets: those new reducibility candidates still contain all ordered pairs of a context and a proof-variable such that the proof-variable is well-typed in the context. But they do not contain ordered pairs of a context and an ill-typed normal neutral proof-term unlike in the usual candidates (where sets of candidates contain ill-typed terms).

Lemma 3.2. *For all $E \subseteq \mathcal{U}$, if E satisfies ($CR_{\equiv}$) and ($CR_{3_{\equiv}}$), then for all proof-variables α, $(\alpha : A_E, \alpha) \in E$, but $(\Gamma, \alpha\alpha) \notin E$, for any context Γ, if $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing.*

Proof. If E satisfies ($\mathrm{CR}_{\equiv}$) and ($\mathrm{CR}_{3_{\equiv}}$), and $\Gamma \vdash \alpha : A_E$, then $(\Gamma, \alpha) \in E$ since α is neutral and normal. Moreover, we saw in 3.1.2 that the proof-term $\alpha\alpha$ cannot be well-typed in a strongly normalizing theory, therefore it cannot belong to a set which satisfies ($\mathrm{CR}_{\equiv}$).

□

We then define the domain of $\mathcal{C}_{\equiv}$ as the set of sets of ordered pairs (Γ, π) satisfying the four properties we defined previously.

Definition 3.8 (domain $\mathcal{C}_{\equiv}$)**.**
$\mathcal{C}_{\equiv}$ is the set that contains all (and only) subsets of $\mathcal{U}$ satisfying ($CR_{\equiv}$), ($CR_{1_{\equiv}}$), ($CR_{2_{\equiv}}$) and ($CR_{3_{\equiv}}$).

Now that we have defined the domain of the LDTVA $\mathcal{C}_{\equiv}$, let us define the interpretations of connectives. We first adapt the usual interpretation of $\Rightarrow$ to elements of $\mathcal{U}$.

Definition 3.9 ($\dot{\Rightarrow}$)**.** *For all $E, F \subseteq \mathcal{U}$,*
$E \dot{\Rightarrow} F = \{(\Gamma, \pi) \in \mathcal{U}$ *such that for all* $(\Gamma', \pi') \in E$, $(\Gamma\Gamma', \pi\pi') \in F\}$

Remark 3.3. *We recall the fact that we only consider well-formed contexts, therefore the only variables* Γ *and* Γ' *can share have to be declared proofs of equivalent propositions, otherwise we have to rename variables in* π' *and* Γ' *when concatening* Γ *and* Γ' *into* $\Gamma\Gamma'$.

Let us prove now that this actually defines a function from $\mathcal{C}_{\equiv} \times \mathcal{C}_{\equiv}$ to $\mathcal{C}_{\equiv}$.

Lemma 3.3. $\mathring{\Rightarrow}$ *is a function from* $\mathcal{C}_{\equiv} \times \mathcal{C}_{\equiv}$ *to* $\mathcal{C}_{\equiv}$.

Proof. Let $E, F \in \mathcal{C}_{\equiv}$, let us prove that $E \mathring{\Rightarrow} F \in \mathcal{C}_{\equiv}$. Let $(\Gamma, \pi) \in E \mathring{\Rightarrow} F$,

- ($\mathrm{CR}_{\equiv}$) Let α be a proof-variable. Since E satisfies ($\mathrm{CR}_{\equiv}$) and ($\mathrm{CR}_{3_{\equiv}}$), $(\alpha : A_E, \alpha) \in E$, therefore $(\Gamma, \alpha : A_E, \pi\alpha) \in F$, since $(\Gamma, \pi) \in E \mathring{\Rightarrow} F$. since F satisfies ($\mathrm{CR}_{\equiv}$), we have $\Gamma, \alpha : A_E \vdash \pi\alpha : A_F$. Therefore $\Gamma \vdash \pi : A_E \Rightarrow A_F$ (by case analysis on the last rule used in the derivation of $\Gamma, \alpha : A_E \vdash \pi\alpha : A_F$). Finally, $A_{E \mathring{\Rightarrow} F} \equiv A_E \Rightarrow A_F$.

- ($\mathrm{CR}_{1_{\equiv}}$) Let α be a proof-variable. Since E satisfies ($\mathrm{CR}_{\equiv}$) and ($\mathrm{CR}_{3_{\equiv}}$), $(\alpha : A_E, \alpha) \in E$, therefore $(\Gamma, \alpha : A_E, \pi\alpha) \in F$, since $(\Gamma, \pi) \in E \mathring{\Rightarrow} F$. Since F satisfies ($\mathrm{CR}_{1_{\equiv}}$), we have $\pi\alpha \in SN$, therefore $\pi \in SN$.

- ($\mathrm{CR}_{2_{\equiv}}$) Let τ be such that $\pi \to \tau$. Then, for all $(\Gamma', \pi') \in E$, $(\Gamma\Gamma', \pi\pi') \in F$ and $\pi\pi' \to \tau\pi'$, therefore $(\Gamma\Gamma', \tau\pi') \in F$ since F satisfies ($\mathrm{CR}_{2_{\equiv}}$). Finally, $(\Gamma, \tau) \in E \mathring{\Rightarrow} F$.

- ($\mathrm{CR}_{3_{\equiv}}$) Let $(\Gamma, \mu) \in \mathcal{U}$ such that $\Gamma \vdash \mu : A_{E \mathring{\Rightarrow} F}$, μ is neutral and all its one-step reducts are in $E \mathring{\Rightarrow} F$. Then, $\Gamma \vdash \mu : A_E \Rightarrow A_F$. For all $(\Gamma', \pi') \in E$, $\mu\pi'$ is neutral and we have $\Gamma\Gamma' \vdash \mu\pi' : A_F$. Let τ be a one-step reduct of $\mu\pi'$. We prove, by induction on the maximal length of a reduction sequence from π' ($\in SN$), that $(\Gamma\Gamma', \tau) \in F$. As μ is neutral, either $\tau = \mu\pi''$ with $\pi' \to \pi''$, and we conclude by induction hypothesis. Either $\tau = \mu'\pi'$, and in this case, $(\Gamma\Gamma', \tau) \in F$, by hypothesis on μ. Finally, for all $(\Gamma', \pi') \in E$, $(\Gamma\Gamma', \mu\pi') \in F$, since F satisfies ($\mathrm{CR}_{3_{\equiv}}$), and finally $(\Gamma, \mu) \in E \mathring{\Rightarrow} F$.

□

Let us now focus on how to define the interpretation of the universal quantifier in $\mathcal{C}_{\equiv}$. We first define, for all sorts $T \in \mathbb{T}$, $\mathring{\mathcal{A}}_T$ as the set of functions f from $\hat{T}$ to $\mathcal{C}_{\equiv}$ such that there exists a term-variable x_f and a proposition A_f such that for all $t \in \hat{T}$, all ordered pairs (Γ, π) in $f(t)$ satisfy $\Gamma \vdash \pi : (t/x_f)A_f$. This will help us to know, when interpreting a family of functions from $\mathring{\mathcal{A}}_T$ to $\mathcal{C}_{\equiv}$, which is the universally quantified proposition we are interpreting.

Definition 3.10 ($\mathring{\mathcal{A}}_T$)**.** *For all sorts T,*
$\mathring{\mathcal{A}}_T = \{f : \hat{T} \mapsto \mathcal{C}_{\equiv}$, *such that there exists $A_f \in \mathcal{P}$ and $x_f \in \mathcal{X}$ such that for all $t \in \hat{T}$ and $(\Gamma, \pi) \in f(t)$, $\Gamma \vdash \pi : (t/x_f)A_f\}$*

Remark 3.4. *In other words, $f \in \mathring{\mathcal{A}}_T$ iff for all $t \in \hat{T}$, $A_{f(t)} = (t/x_f)A_f$ (with the notation we used for (CR$_{\equiv}$)).*

Definition 3.11 ($\mathring{\forall}_T$)**.** *For all sorts T and functions $f \in \mathring{\mathcal{A}}_T$,*
$\mathring{\forall}_T f = \{(\Gamma, \pi) \in \mathcal{U}$ *s.t. $\pi \in SN$, $\Gamma \vdash \pi : \forall x_f.A_f$ and for all $t \in \hat{T}$, $(\Gamma, \pi t) \in f(t)\}$*

Let us now prove that $\mathring{\forall}_T$ actually defines a function from $\mathring{\mathcal{A}}_T$ to $\mathcal{C}_{\equiv}$.

Lemma 3.4. *For all sorts T, $\mathring{\forall}_T$ is a function from $\mathring{\mathcal{A}}_T$ to $\mathcal{C}_{\equiv}$.*

Proof. Let $f \in \mathring{\mathcal{A}}_T$, and $(\Gamma, \pi) \in \mathring{\forall}_T f$

- (CR$_{\equiv}$) By definition: if $(\Gamma, \pi) \in \mathring{\forall}_T f$ then $\Gamma \vdash \pi : \forall x_f.A_f$.
- (CR$_{1_{\equiv}}$) By definiton.
- (CR$_{2_{\equiv}}$) Let π' be such that $\pi \to \pi'$. Then, for all $t \in \hat{T}$, $\pi t \to \pi' t$, therefore $\pi' t \in f(t) \in \mathcal{C}_{\equiv}$.
- (CR$_{3_{\equiv}}$) Let $(\Gamma, \mu) \in \mathcal{U}$ such that μ is neutral, $\Gamma \vdash \mu : \forall x_f.A_f$, and for all one-step reducts μ' of μ, $(\Gamma, \mu') \in \mathring{\forall}_T f$. Let $t \in \hat{T}$, then μt is neutral, $\Gamma \vdash \mu t : (t/x_f)A_f$, and, since μ is neutral, all one-step reducts of μt are of the form $\mu' t$, with $\mu \to \mu'$, hence $(\Gamma, \mu t) \in f(t)$ since $f(t)$ satisfies (CR$_{3_{\equiv}}$). Finally, $(\Gamma, \mu) \in \mathring{\forall}_T f$.

□

Remark 3.5. *We could have defined $\mathring{\forall}_T$ in a simpler way without assuming explicitely that all $(\Gamma, \pi) \in \mathring{\forall}_T f$ satisfy $\pi \in SN$ and $\Gamma \vdash \pi : \forall x_f.A_f$. This would have led to the following definition:*

$$\mathring{\forall}_T f = \{(\Gamma, \pi) \in \mathcal{U} \text{ s.t. for all } t \in \hat{T},\ (\Gamma, \pi t) \in f(t)\}$$

But in order to prove that $\mathring{\forall}_T$ is a function from $\mathring{\mathcal{A}}_T$ to $\mathcal{C}_{\equiv}$, and in particular that for all f in $\mathring{\mathcal{A}}_T$, $\mathring{\forall}_T f$ satisfies (CR$_{\equiv}$) and (CR$_{1_{\equiv}}$), we would have to assume, in this case, that all sets $\hat{T}$ contain at least two different elements, for all sorts $T \in \mathbb{T}$. As soon as $\hat{T}$ contains at least one element t and $(\Gamma, \pi t) \in f(t) \in \mathcal{C}_{\equiv}$, we can conclude that $\pi t \in SN$ and so does π. And in order to prove that if $(\Gamma, \pi) \in \mathring{\forall}_T f$ then $\Gamma \vdash \pi : \forall x_f.A_f$, since we only know that for all $t \in \hat{T}$, $\Gamma \vdash \pi t : (t/x_f)A_f$, we have to suppose that $\hat{T}$ contains two different elements, otherwise, we can only conclude that $\Gamma \vdash \pi : \forall x_f.A'$, with $(t/x_f)A' = (t/x_f)A_f$.

We finally define $\mathcal{C}_{\equiv}$ with the domain and the interpretations of connectives we previously defined.

Definition 3.12 ($\mathcal{C}_{\equiv}$)**.** $\mathcal{C}_{\equiv}$ *is the* LDTVA $\langle \mathcal{C}_{\equiv}, \mathring{\Rightarrow}, (\mathring{\mathcal{A}}_T), (\mathring{\forall}_T) \rangle$.

Let us prove in the next section that this definition of ($\equiv$-)well-typed reducibility candidates still provide a sound criterion for strong normalization in minimal deduction modulo.

3.2.2 $\mathcal{C}_{\equiv}$-models as a sound semantics for strong normalization

It is not difficult to adapt the usual adequacy lemma to prove that if a theory has a $\mathcal{C}_{\equiv}$-valued model adapted to typing (i.e. the interpretation of a proposition and a valuation is the value of this valuation on this proposition), then it is strongly normalizing.

In fact, as explained in 3.1.1, (CR_3) is only required twice: to prove that interpretations are non-empty since they contain all normal neutral proof-terms, and to prove that abstractions of type $A \Rightarrow B$ are in the interpretation of $A \Rightarrow B$. We have seen in lemma 3.2 that $\mathcal{C}_{\equiv}$-interpretations are also non-empty since the interpretation of a proposition A contains all ordered pairs (Γ, π) such that π is neutral and normal, and $\Gamma \vdash \pi : A$. Moreover, when proving that abstractions $\lambda\alpha.\pi$ of type $A \Rightarrow B$ (in a context Γ) are (when paired with Γ) in the interpretation of $A \Rightarrow B$, the fact that $\Gamma \vdash \lambda\alpha.\pi : A \Rightarrow B$ is assumed, and we can therefore use ($\mathrm{CR}_{3_{\equiv}}$) instead of (CR_3) without any problem.

The only difference with the usual soundness lemma (like the one of section 2.3.2) is the fact that since we use a *synchronized* version of the interpretation of the universal quantifier, the statement of this lemma reflects this synchronization. This statement is of the form: if $\Gamma \vdash \pi : A$ then for all (adapted) substitutions σ and valuations φ, we have $\sigma\varphi\pi \in [\![A]\!]_\varphi$ whereas in the usual statement of the soundness lemma, there is a quantification on two different valuations: for all valuations θ and φ, $\sigma\theta\pi \in [\![A]\!]_\varphi$. Let us see precisely this difference in the following lemma.

Lemma 3.5. *For all theories* $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ *expressed in minimal deduction modulo, if* $[\![.]\!]$. *is a* $\mathcal{C}_{\equiv}$*-model adapted to typing, then for all typing judgements* $\Gamma \vdash \pi : A$*, valuations* $\varphi \in \textsc{Val}(A)$ *and substitutions* σ *such that for all proof-variables* α *declared of type* B *in* Γ*,* $(\Gamma, \sigma\alpha) \in [\![B]\!]_\varphi$*, we have* $(\Gamma, \sigma\varphi\pi) \in [\![A]\!]_\varphi$.

Proof. By induction on the length of the derivation of $\Gamma \vdash \pi : A$. by case analysis on the last rule used. If the last rule used is :

- axiom: in this case, π is a variable α, and Γ contains a declaration $\alpha : B$ with $A \equiv B$ (therefore $\varphi A \equiv \varphi B$). Then $(\Gamma, \sigma\varphi\pi) = (\Gamma, \sigma\alpha) \in [\![B]\!]_\varphi = [\![A]\!]_\varphi$.

- $\Rightarrow$-intro: in this case, π is an abstraction $\lambda\alpha.\tau$, and we have $\Gamma, \alpha : B \vdash \tau : C$ with $A \equiv B \Rightarrow C$. Let σ' be such that for all variables β declared in Γ, $\sigma'\beta = \sigma\beta$ and $(\varphi(\Gamma, \alpha : B), \sigma'\alpha)$ is an element of $[\![B]\!]_\varphi$. Then $(\varphi(\Gamma, \alpha : B), \sigma'\varphi\tau) \in [\![C]\!]_\varphi$ by induction hypothesis (and $\sigma'\varphi\tau$ is in SN, therefore $\sigma\varphi\pi$ is also in SN). Let $(\Gamma', \pi') \in [\![B]\!]_\varphi$, we prove by induction on the sum of both maximal lengths of a reductions sequence from $\sigma\varphi(\lambda\alpha.\tau)$ and π' (each in SN) that every one-step reduct μ of the neutral well-typed (by A in $\Gamma\Gamma'$) proof-term $\sigma\varphi(\lambda\alpha.\tau)\ \pi'$ satisfies the fact that $(\Gamma\Gamma', \mu) \in [\![C]\!]_\varphi$. If the one-step reduct is $\sigma\varphi(\pi'/\alpha)\tau$, we conclude by induction hypothesis (on the length of the derivation) as $(\Gamma', \pi') \in [\![B]\!]_\varphi$. Otherwise, the reduction takes place either in $\sigma\varphi(\lambda\alpha.\tau)$, either in π'. We conclude by induction hypothesis on the sum of the maximal lengths of reductions sequence from $\sigma\varphi(\lambda\alpha.\tau)$ and π'. And the fact that both $[\![B]\!]_\varphi$ and $[\![B \Rightarrow C]\!]_\varphi$ satisfy $(\mathrm{CR}_{2_\equiv})$. Finally, $(\Gamma\Gamma', \sigma\varphi(\lambda\alpha.\tau)\ \pi') \in [\![C]\!]_\varphi$, since it satisfies $(\mathrm{CR}_{3_\equiv})$ and $\sigma\varphi(\lambda\alpha.\tau)\ \pi'$ is neutral and well-typed. Hence $(\Gamma, \sigma\varphi(\lambda\alpha.\tau)) \in [\![B]\!]_\varphi \tilde{\Rightarrow} [\![C]\!]_\varphi = [\![B \Rightarrow C]\!]_\varphi = [\![A]\!]_\varphi$

- $\Rightarrow$-elim: in this case, π is an application $\rho\tau$, and we have $\Gamma \vdash \rho : C \equiv B \Rightarrow A$ and $\Gamma \vdash \tau : B$. Therefore, by induction hypothesis, $(\Gamma, \sigma\varphi\rho) \in [\![B \Rightarrow A]\!]_\varphi = [\![B]\!]_\varphi \tilde{\Rightarrow} [\![A]\!]_\varphi$ and $(\Gamma, \sigma\varphi\tau) \in [\![B]\!]_\varphi$. Therefore $(\Gamma, \sigma\varphi(\rho\tau)) \in [\![A]\!]_\varphi$.

- $\forall$-intro: in this case, π is a term abstraction $\lambda x.\pi'$ and we have $\Gamma \vdash \pi' : B$ with $A \equiv \forall x.B$. Let $t \in \hat{T}$ (where T is the sort of x), and $\varphi' = \varphi + \langle x, t\rangle$. Then $(\Gamma, \sigma\varphi'\pi') = (\Gamma, \sigma\varphi(t/x)\pi') \in [\![B]\!]_{\varphi'}$, by induction hypothesis. Therefore $(\Gamma, \sigma\varphi(\lambda x.\pi')) \in \tilde{\forall}_T(t \mapsto [\![B]\!]_{\varphi+\langle x,t\rangle}) = [\![A]\!]_\varphi$ (by induction on the maximal length of a reductions sequence from πt, with $t \in \hat{T}$, using the fact that for all $t \in \hat{T}$, $[\![B]\!]_{\varphi+\langle x,t\rangle}$ satisfies $(\mathrm{CR}_{2_\equiv})$ and $(\mathrm{CR}_{3_\equiv})$).

- $\forall$-elim: in this case, π is an application ρt, and we have $\Gamma \vdash \rho : \forall x.B$ with $A = (t/x)B$ and $x \notin FV(\Gamma)$. By induction hypothesis, we have $(\Gamma, \sigma\varphi\rho) \in [\![\forall x.B, \varphi]\!] = \tilde{\forall}_T(t \mapsto [\![B]\!]_{\varphi+\langle x,t\rangle})$. Therefore we have $(\Gamma, \sigma\varphi(\rho t)) = (\Gamma, \sigma\varphi\rho\ (\varphi t)) \in [\![B]\!]_{\varphi+\langle x,\varphi t\rangle} = [\![(t/x)B]\!]_\varphi = [\![A]\!]_\varphi$

□

Proposition 3.1 (Soundness)**.**
For all theories $\langle \mathbb{T}, \mathbb{F}, \mathbb{P}\rangle_\equiv$ expressed in minimal deduction modulo, if it has a $\mathcal{C}_\equiv$-model adapted to typing $[\![.]\!]$. (i.e. for all propositions A, $\varphi \in \textsc{Val}(A)$ and $(\Gamma, \pi) \in [\![A]\!]_\varphi$, $\Gamma \vdash \pi : \varphi A$) then it is strongly normalizing.

Proof. As usual, we deduce from the previous lemma that if a proof-term π is well-typed then there exists a substitution σ (as $\mathcal{C}_\equiv$-interpretations are non-empty) such that $\sigma\pi \in SN$ therefore π is also in SN. □

3.2.3 $\mathcal{C}_{\equiv}$-models as a complete semantics for strong normalization: the long way

We detail in this subsection the first proof of completeness of $\mathcal{C}_{\equiv}$-models for strong normalization we obtained. Thanks to an idea of Stéphane Lengrand, we obtained afterwards a simpler proof of completeness of $\mathcal{C}_{\equiv}$-models for strong normalization. This simpler proof is detailed in the next subsection.

An interpretation of propositions adapted to typing

Now that we have defined the LDTVA $\mathcal{C}_{\equiv}$ of well-typed reducibility candidates, our goal is to build a $\mathcal{C}_{\equiv}$-valued interpretation which is a model of the theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ if it is strongly normalizing. For that purpose, we define a first $\mathcal{C}_{\equiv}$-valued interpretation $[.]_{.}$ in a natural way: we interpret atomic propositions by proofs of A which are strongly normalizing, with their associated contexts, and we then deduce inductively interpretations of all propositions by using the definitions of $\overset{\circ}{\Rightarrow}$ and $\overset{\circ}{\forall}_T$ (as usually done in the construction of models valued in original truth values algebras).

Definition 3.13. *Let A be a proposition and $\varphi \in \text{VAL}(A)$.*
We define the subset of $\mathcal{U}$, $[A]_\varphi$ by induction over the structure of A.

- $[P\ t_1 \dots t_n]_\varphi = \{(\Gamma, \pi) \in \mathcal{U}$ *such that* $\pi \in SN$ *and* $\Gamma \vdash \pi : \varphi(P\ t_1 \dots t_n)\}$
- $[B \Rightarrow C]_\varphi = [B]_\varphi \overset{\circ}{\Rightarrow} [C]_\varphi$
- $[\forall x.B]_\varphi = \overset{\circ}{\forall}_T\ (t \mapsto [B]_{\varphi + \langle x,t \rangle})$

Let us first prove that this actually defines a $\mathcal{C}_{\equiv}$-valued interpretation, with the particularity that for all propositions A and $\varphi \in \text{VAL}(A)$, their interpretation $[A]_\varphi$ only contains proofs of φA. In other words, $[.]_{.}$ is a $\mathcal{C}_{\equiv}$-valued interpretation adapted to typing..

Lemma 3.6. *For all $A \in \mathcal{P}$, and $\varphi \in \text{VAL}(A)$,*
$[A]_\varphi \in \mathcal{C}_{\equiv}$ *with* $A_{[A]_\varphi} = \varphi A$ *(i.e, for all $(\Gamma, \pi) \in [A]_\varphi$, $\Gamma \vdash \pi : \varphi A$).*

Proof. By induction on A.

- If A is an atomic proposition $P\ t_1 \dots t_n$,

  ($\text{CR}_{\equiv}$) By definition. (where $A_{[P\ t_1 \dots t_n]_\varphi} \equiv P\ \varphi(t_1) \dots \varphi(t_n)$).

  ($\text{CR}_{1_{\equiv}}$) By definition.

  ($\text{CR}_{2_{\equiv}}$) By subject-reduction (lemma 2.4) and the fact that each reduct of a proof-term in SN is also in SN.

  ($\text{CR}_{3_{\equiv}}$) If all one-step reducts of a proof-term are in SN, then it is also in SN.

- If $A = B \Rightarrow C$, as $\mathring{\Rightarrow} : \mathcal{C}_{\equiv} \times \mathcal{C}_{\equiv} \mapsto \mathcal{C}_{\equiv}$, we conclude by induction hypothesis
(where $A_{[B\Rightarrow C]_\varphi} = A_{[B]_\varphi \mathring{\Rightarrow} [C]_\varphi} \equiv A_{[B]_\varphi} \Rightarrow A_{[C]_\varphi} \equiv \varphi B \Rightarrow \varphi C = \varphi(B \Rightarrow C)$).

- If $A = \forall x.B$, let T be the sort of x and $f = t \mapsto [B]_{\varphi+\langle x,t\rangle}$.
Then f is a function from $\hat{T}$ to $\mathcal{C}_{\equiv}$, by induction hypothesis. Moreover, for all $t \in \hat{T}$, $A_{f(t)} = (t/x)\varphi B$, by induction hypothesis. Therefore $f \in \mathring{\mathcal{A}}_T$ and $\mathring{\forall}_T f \in \mathcal{C}_{\equiv}$ (where $A_{[\forall x.B]_\varphi} = \forall x.A_f = \varphi(\forall x.B)$).

□

By construction, since we define $[.]_{\_}$ inductively using $\mathring{\Rightarrow}$ and $\mathring{\forall}$, there remains only one property to prove, to show that this interpretation is adapted to the connectives and therefore a $\mathcal{C}_{\equiv}$-valued model.

Lemma 3.7. *For all $A \in \mathcal{P}$, x of sort T, $t \in \hat{T}$ and $\varphi \in \text{VAL}(\forall x.A)$ such that $x \notin \text{DOM}(\varphi)$, we have $[(t/x)A]_\varphi = [A]_{\varphi+\langle x,t\rangle}$.*

Proof. By induction on A. Let us write $\varphi' = \varphi + \langle x, t\rangle$.

- If A is an atomic proposition $P\ t_1 \dots t_n$,

$$\begin{aligned}
[(t/x)A]_\varphi &= \{(\Gamma,\pi) \text{ s.t. } \pi \in SN \text{ and } \Gamma \vdash \pi : P\ (t/x)\varphi(t_1) \dots (t/x)\varphi(t_n)\} \\
&= \{(\Gamma,\pi) \text{ s.t. } \pi \in SN \text{ and } \Gamma \vdash \pi : P\ \varphi'(t_1) \dots \varphi'(t_n)\} \\
&= [A]_{\varphi+\langle x,t\rangle}
\end{aligned}$$

  as t is term-closed.

- If $A = B \Rightarrow C$,

$$\begin{aligned}
[(t/x)A]_\varphi &= [(t/x)B \Rightarrow (t/x)C]_\varphi \\
&= [(t/x)B]_\varphi \mathring{\Rightarrow} [(t/x)C]_\varphi \\
&= [B]_{\varphi+\langle x,t\rangle} \mathring{\Rightarrow} [C]_{\varphi+\langle x,t\rangle} \quad \text{by induction hypothesis} \\
&= [B \Rightarrow C]_{\varphi+\langle x,t\rangle} \\
&= [A]_{\varphi+\langle x,t\rangle}
\end{aligned}$$

- If $A = \forall y.B$, let T be the sort of x

$$\begin{aligned}
[(t/x)A]_\varphi &= [(t/x)\forall y.B]_\varphi \\
&= [\forall y.(t/x)B]_\varphi \quad \text{as } t \text{ is term-closed} \\
&= \mathring{\forall}_T(t' \mapsto [(t/x)B]_{\varphi+\langle y,t'\rangle}) \\
&= \mathring{\forall}_T(t' \mapsto [(B]_{\varphi+\langle y,t'\rangle+\langle x,t\rangle}) \quad \text{by induction hypothesis} \\
&= \mathring{\forall}_T(t' \mapsto [(B]_{\varphi+\langle x,t\rangle+\langle y,t'\rangle}) \quad \text{as } t \text{ is term-closed} \\
&= [\forall y.B]_{\varphi+\langle x,t\rangle} \\
&= [A]_{\varphi+\langle x,t\rangle}
\end{aligned}$$

□

We have proved that $[.]_.$ is a $\mathcal{C}_{\equiv}$-valued model: it is a $\mathcal{C}_{\equiv}$-valued interpretation adapted to the connectives. We have still to prove that it is a $\mathcal{C}_{\equiv}$-valued model of $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ (*i.e.* also adapted to the congruence $\equiv$) when this theory is strongly normalizing. Although we conjecture that it is the case, we did not find a way to prove it. It is not easy to prove that all $\equiv$-equivalent propositions have the same interpretation when the theory is strongly normalizing (we shall see in the next subsection that it is not so hard when choosing the short way). We therefore chose another way to get this result: first define a second $\mathcal{C}_{\equiv}$-valued interpretation which is adapted the congruence, and then show that it is also adapted to the connectives when the theory is strongly normalizing.

Adapting this interpretation to the congruence

Changing our first interpretation to make it adapted to the congruence is not difficult: we simply take the intersection of all the previous interpretations of equivalent propositions. We then define a second interpretation $\lfloor . \rfloor_.$ as follows:

Definition 3.14 ($\lfloor . \rfloor_.$)**.** *For all $A \in \mathcal{P}$ and $\varphi \in \text{VAL}(A)$,*

$$\lfloor A \rfloor_\varphi = \bigcap_{\varphi A \equiv \psi A'} [A']_\psi$$

Remark 3.6. *For all $A, A' \in \mathcal{P}$, $\varphi \in \text{VAL}(A)$ and $\psi \in \text{VAL}(A')$ such that $\varphi A \equiv \psi A'$, we have $\lfloor A \rfloor_\varphi = \lfloor A' \rfloor_\psi$, by construction: $\lfloor . \rfloor_.$ is always adapted to the congruence whereas the theory is strongly normalizing or not.*

Then we prove that $\lfloor . \rfloor_.$ is also a $\mathcal{C}_{\equiv}$-valued interpretation adapted to typing, and that it satisfies the property of models concerning term-substitution.

Lemma 3.8. *For all $A \in \mathcal{P}$, and $\varphi \in \text{VAL}(A)$,*
$\lfloor A \rfloor_\varphi \in \mathcal{C}_{\equiv}$ with $A_{\lfloor A \rfloor_\varphi} = \varphi A$ (i.e, $\forall (\Gamma, \pi) \in \lfloor A \rfloor_\varphi$, $\Gamma \vdash \pi : \varphi A$).

Proof. By lemma 3.6, since $\lfloor A \rfloor_\varphi \subseteq [A]_\varphi$. □

Lemma 3.9. *For all $A \in \mathcal{P}$, x of sort T, $t \in \hat{T}$ and $\varphi \in \text{VAL}(\forall x.A)$ such that $x \notin \text{DOM}(\varphi)$, we have $\lfloor (t/x)A \rfloor_\varphi = \lfloor A \rfloor_{\varphi + \langle x,t \rangle}$.*

Proof. As $(\varphi + \langle x, t \rangle)A = (t/x)\varphi(A_\varphi)$. □

Finally, we proved, that $\lfloor . \rfloor_.$ is a $\mathcal{C}_{\equiv}$-valued interpretation of propositions adapted to typing and to the congruence relation $\equiv$. Let us now show that if the theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing, then $\lfloor . \rfloor_.$ is a $\mathcal{C}_{\equiv}$-valued model

of $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$, *i.e.* it is also adapted to connectives. In order to prove that $\lfloor . \rfloor_.$ is a $\mathcal{C}_{\equiv}$-valued model of $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$, if it is strongly normalizing, we proceed by contraposition, showing that if $\lfloor . \rfloor_.$ is not connectives-adapted, then we can exhibit a typing judgement $\Gamma \vdash \pi : A$ such that $\pi \notin SN$.

We first show that if $\lfloor . \rfloor_.$ is not adapted to the connective $\Rightarrow$, then we can exhibit a proposition C, a valuation $\psi \in \textsc{Val}(C)$ and $(\Gamma, \pi) \in \mathcal{U}$ such that $\Gamma \vdash \pi : \psi C$ but $(\Gamma, \pi) \notin \lfloor C \rfloor_\psi$

Lemma 3.10.
If there exists $A, B \in \mathcal{P}$ and $\varphi \in \textsc{Val}(A \Rightarrow B)$, such that

$$\lfloor A \Rightarrow B \rfloor_\varphi \neq \lfloor A \rfloor_\varphi \mathring{\Rightarrow} \lfloor B \rfloor_\varphi$$

then there exists $\pi \in \mathcal{T}$, $C \in \mathcal{P}$, $\psi \in \textsc{Val}(C)$ such that

$$\Gamma \vdash \pi : \psi C \text{ and } (\Gamma, \pi) \notin \lfloor C \rfloor_\psi .$$

Proof.

- If there exists $(\Gamma, \pi) \in \mathcal{U}$ such that $(\Gamma, \pi) \notin \lfloor A \Rightarrow B \rfloor_\varphi$ and $(\Gamma, \pi) \in \lfloor A \rfloor_\varphi \mathring{\Rightarrow} \lfloor B \rfloor_\varphi$. Then $\Gamma \vdash \pi : \varphi A \Rightarrow \varphi B = \varphi(A \Rightarrow B)$.
  We take $C = A \Rightarrow B$ and $\psi = \varphi$.

- If there exists $(\Gamma, \pi) \in \mathcal{U}$ such that $(\Gamma, \pi) \in \lfloor A \Rightarrow B \rfloor_\varphi$ and $(\Gamma, \pi) \notin \lfloor A \rfloor_\varphi \mathring{\Rightarrow} \lfloor B \rfloor_\varphi$. Then there exists $(\Gamma', \pi') \in \lfloor A \rfloor_\varphi$ such that $(\Gamma\Gamma', \pi\pi') \notin \lfloor B \rfloor_\varphi$. Since $(\Gamma, \pi) \in \lfloor A \Rightarrow B \rfloor_\varphi$, and $(\Gamma', \pi') \in \lfloor A \rfloor_\varphi$, we have $\Gamma \vdash \pi : \varphi(A \Rightarrow B) = \varphi A \Rightarrow \varphi B$ and $\Gamma' \vdash \pi' : \varphi A$. Therefore $\Gamma\Gamma' \vdash \pi\pi' : \varphi B$. We take $C = A \Rightarrow B$ and $\psi = \varphi$.

□

Then we show the same for the universal quantifier $\forall$: if $\lfloor . \rfloor_.$ is not adapted to $\forall$, then we can exhibit a proposition C, a valuation $\psi \in \textsc{Val}(C)$ and $(\Gamma, \pi) \in \mathcal{U}$ such that $\Gamma \vdash \pi : \psi C$ but $(\Gamma, \pi) \notin \lfloor C \rfloor_\psi$

Lemma 3.11.
If there exists $A \in \mathcal{P}$, $\varphi \in \textsc{Val}(A)$, and x of sort T not in $\textsc{dom}(\varphi)$, s.t.

$$\lfloor \forall x.A \rfloor_\varphi \neq \mathring{\forall}_T(t \mapsto \lfloor A \rfloor_{\varphi + \langle x,t \rangle})$$

then there exists $\pi \in \mathcal{T}$, $C \in \mathcal{P}$, $\psi \in \textsc{Val}(C)$ such that

$$\Gamma \vdash \pi : \psi C \text{ and } (\Gamma, \pi) \notin \lfloor C \rfloor_\psi .$$

Proof.

- If there exists $(\Gamma, \pi) \in \mathcal{U}$ such that $(\Gamma, \pi) \notin \lfloor \forall x.A \rfloor_\varphi$ and $(\Gamma, \pi) \in \mathring{\forall}_T(t \mapsto \lfloor A \rfloor_{\varphi + \langle x,t \rangle})$. Then $\Gamma \vdash \pi : \varphi(\forall x.A)$.
  We take $C = \forall x.A$ and $\psi = \varphi$.

- If there exists $(\Gamma, \pi) \in \mathcal{U}$ such that $(\Gamma, \pi) \in \lfloor \forall x.A \rfloor_\varphi$ and $(\Gamma, \pi) \notin \mathring{\forall}_T(t \mapsto \lfloor A \rfloor_{\varphi + \langle x,t \rangle})$. Then there exists $t \in \hat{T}$ such that $(\Gamma, \pi t) \notin \lfloor A \rfloor_{\varphi + \langle x,t \rangle})$. As $(\Gamma, \pi) \in \lfloor \forall x.A \rfloor_\varphi$, we have $\Gamma \vdash \pi : \varphi(\forall x.A)$, therefore $\Gamma \vdash \pi t : (t/x)\varphi A$. We take $C = A$ and $\psi = \varphi + \langle x, t \rangle$

□

Given these lemmas, we can now deduce that if $\lfloor . \rfloor$ is not adapted to the connectives, then we can exhibit a proposition D, a valuation $\psi \in \text{VAL}(D)$ and $(\Gamma, \pi) \in \mathcal{U}$ such that $\Gamma \vdash \pi : \psi D$ but $(\Gamma, \pi) \notin [D]_\psi$ (the first interpretation).

Lemma 3.12.
If there exists $A, B \in \mathcal{P}$, $\varphi \in \text{VAL}(A \Rightarrow B)$ *or* $\varphi' \in \text{VAL}(\forall x.A)$ *with* x *of sort* T, $x \notin \text{DOM}(\varphi')$ *and*

$$\lfloor A \Rightarrow B \rfloor_\varphi \neq \lfloor A \rfloor_\varphi \tilde{\Rightarrow} \lfloor B \rfloor_\varphi \quad \text{or} \quad \lfloor \forall x.A \rfloor_{\varphi'} \neq \mathring{\forall}_T (t \mapsto \lfloor A \rfloor_{\varphi' + \langle x,t \rangle}$$

then there exists $D \in \mathcal{P}$, $\pi \in \mathcal{T}$, $\psi \in \text{VAL}(D)$ *such that*

$$\Gamma \vdash \pi : \psi D \text{ and } (\Gamma, \pi) \notin [D]_\psi.$$

Proof. By lemmas 3.10 and 3.11, there exists C, Γ, π and ψ such that $\Gamma \vdash \pi : \psi C$ and $(\Gamma, \pi) \notin \lfloor C \rfloor_\psi$. Therefore, there exists a proposition D and $\psi' \in \text{VAL}(D)$ such that $\psi' D \equiv \psi C i$ and $(\Gamma, \pi) \notin [D]_{\psi'}$. And $\Gamma \vdash \pi : D_{\psi'}$, by equivalence of ψC and $\psi' D$. □

And finally, this allows us to prove that if $\lfloor . \rfloor$ is not adapted to the connectives, then there exists a well-typed proof-term which is not strongly normalizing.

Lemma 3.13.
If there exists $A, B \in \mathcal{P}$, $\varphi \in \text{VAL}(A \Rightarrow B)$ *or* $\varphi' \in \text{VAL}(\forall x.A)$ *with* x *of sort* T, $x \notin \text{DOM}(\varphi')$ *and*

$$\lfloor A \Rightarrow B \rfloor_\varphi \neq \lfloor A \rfloor_\varphi \tilde{\Rightarrow} \lfloor B \rfloor_\varphi \quad \text{or} \quad \lfloor \forall x.A \rfloor_{\varphi'} \neq \mathring{\forall}_T (t \mapsto \lfloor A \rfloor_{\varphi' + \langle x,t \rangle})$$

then there exists a (term-closed) proposition E, $\pi \in \mathcal{T}$ *and a context* Γ *such that*

$$\Gamma \vdash \pi : E \text{ and } \pi \notin SN.$$

Proof. By lemma 3.12, there exists a proposition D, a context Γ, a proof π and $\varphi \in \mathcal{V}(D)$ such that $\Gamma \vdash \pi : \varphi D$ and $(\Gamma, \pi) \notin [D]_\varphi$. By induction on D.

- if D is atomic, then since $\Gamma \vdash \pi : \varphi D$, we have $\pi \notin SN$.
- if $D = F \Rightarrow G$,
  then $\Gamma \vdash \pi : \varphi(F \Rightarrow G)$ and $(\Gamma, \pi) \notin [F \Rightarrow G]_\varphi = [F]_\varphi \tilde{\Rightarrow} [G]_\varphi$. Then there exists $(\Gamma', \pi') \in [F]_\varphi$ such that $(\Gamma\Gamma', \pi\pi') \notin [G]_\varphi$. Since $(\Gamma, \pi) \in [F \Rightarrow G]_\varphi$, and $(\Gamma', \pi') \in [F]_\varphi$, we therefore have $\Gamma \vdash \pi : \varphi(F \Rightarrow G) = \varphi \Rightarrow \varphi G$ and $\Gamma' \vdash \pi' : \varphi F$. Therefore $\Gamma\Gamma' \vdash \pi\pi' : \varphi G$. We conclude by induction hypothesis.
- if $D = \forall x.F$,
  then $\Gamma \vdash \pi : \varphi(\forall x.F)$ and $(\Gamma, \pi) \notin [\forall x.F]_\varphi$. Then there exists $t \in \hat{T}$ such that $(\Gamma, \pi t) \notin [F]_{\varphi + \langle x,t \rangle})$. As $(\Gamma, \pi) \in [\forall x.F]_\varphi$, we have $\Gamma \vdash \pi : \varphi(\forall x.F)$, therefore $\Gamma \vdash \pi t : (t/x)\varphi F$. We conclude by induction hypothesis.

□

We can then conclude this subsection by the following proposition: if $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is a strongly normalizing theory, then $\lfloor . \rfloor_.$ is a $\mathcal{C}_{\equiv}$-valued model of $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$. In other words, having a $\mathcal{C}_{\equiv}$-valued model is a complete condition for a theory to be strongly normalizing.

Proposition 3.2 (Completeness)**.**
If the theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing,
then $\lfloor . \rfloor_. = \langle A, \varphi \rangle \mapsto \lfloor A \rfloor_\varphi$ is a $\mathcal{C}_{\equiv}$-model of this theory.

Proof. By remark 3.6 and lemmas 3.8 and 3.13. □

3.2.4 $\mathcal{C}_{\equiv}$-models as a complete semantics for strong normalization: the short way

Thanks to an idea of Stéphane Lengrand, we were able to provide a simpler proof of completeness of $\mathcal{C}_{\equiv}$-models for strong normalization. We propose for the interpretation of a proposition A and $\varphi \in \textsc{Val}(A)$ directly the set of ordered pairs (Γ, π) such that $\Gamma \vdash \pi : \varphi A$. This interpretation is adapted to connectives as interpretations of connectives follow the definitions of typing rules. It is adapted to the congruence since a proposition can be replaced by an $\equiv$-equivalent one in a typing judgement. It satisfies ($\mathrm{CR}_{2_{\equiv}}$) by subject-reduction and obviously ($\mathrm{CR}_{3_{\equiv}}$). And finally, it satisfies ($\mathrm{CR1}_{\equiv}$) if (and only if) the theory is strongly normalizing.

We shall also write $\lfloor . \rfloor_.$ for this interpretation in order to muddle it with the former one in the following of this work.

Definition 3.15 ((Second) $\lfloor . \rfloor_.$)**.**
For all propositions A and $\varphi \in \textsc{Val}(A)$,

$$\lfloor A \rfloor_\varphi = \{(\Gamma, \pi) \text{ such that } \Gamma \vdash \pi : \varphi A\}.$$

Proposition 3.3 (Completeness)**.**
If the theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing,
then $\lfloor . \rfloor_. = \langle A, \varphi \rangle \mapsto \lfloor A \rfloor_\varphi$ is a $\mathcal{C}_{\equiv}$-model of this theory.

Proof. If the theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing,

- let us first prove that $\lfloor . \rfloor_.$ is a $\mathcal{C}_{\equiv}$-valued interpretation. For all propositions A, if $(\Gamma, \pi) \in \lfloor A \rfloor_\varphi$ then $\Gamma \vdash \pi : \varphi A$ and therefore $\pi \in SN$ and $\lfloor A \rfloor_\varphi$ satisfies ($\mathrm{CR}_{1_{\equiv}}$), since the theory is strongly normalizing. Moreover if $\pi \to \pi'$ then $\Gamma \vdash \pi' : \varphi A$ by subject-reduction hence $\lfloor A \rfloor_\varphi$ satisfies ($\mathrm{CR}_{2_{\equiv}}$). Finally $\lfloor A \rfloor_\varphi$ satisfies obviously ($\mathrm{CR}_{\equiv}$) and ($\mathrm{CR}_{3_{\equiv}}$).

- $\lfloor . \rfloor_.$ is adapted to the congruence since a proposition can be replaced by an $\equiv$-equivalent one in a typing judgement.
- Let us finally prove that $\lfloor . \rfloor_.$ is adapted to the conncetives.
  - for all propositions A and $\varphi + \langle x, t \rangle \in \text{VAL}(A)$, $\lfloor (t/x)A \rfloor_\varphi$ is obviously equal to $\lfloor A \rfloor_{\varphi + \langle x,t \rangle}$.
  - Let A, B be propositions and $\varphi \in \text{VAL}(A \Rightarrow B)$.
    - Let $(\Gamma, \pi) \in \lfloor A \Rightarrow B \rfloor_\varphi$ hence $\Gamma \vdash \varphi A \Rightarrow \varphi B$. And for all $(\Gamma', \pi') \in \lfloor A \rfloor_\varphi$, we have $\Gamma' \vdash \pi' : \varphi A$ hence $\Gamma\Gamma' \vdash \pi\pi' : \varphi B$ and $(\Gamma\Gamma', \pi, \pi') \in \lfloor B \rfloor_\varphi$. Finally, $(\Gamma, \pi) \in \lfloor A \rfloor_\varphi \mathring{\Rightarrow} \lfloor B \rfloor_\varphi$.
    - Let $(\Gamma, \pi) \in \lfloor A \rfloor_\varphi \mathring{\Rightarrow} \lfloor B \rfloor_\varphi$, for all proof-variables α not free in π, since $(\alpha : \varphi A) \in \lfloor A \rfloor_\varphi$, we have $(\Gamma, \alpha : \varphi A,\ \pi\alpha) \in \lfloor B \rfloor_\varphi$ hence $\Gamma, \alpha : \varphi A \vdash \pi\alpha : \varphi B$ therefore $\Gamma \vdash \pi : \varphi A \Rightarrow \varphi B$ by case analysis on the last rule used in $\Gamma, \alpha : \varphi A \vdash \pi\alpha : \varphi B$, and finally
      $(\Gamma, \pi) \in \lfloor A \Rightarrow B \rfloor_\varphi$.
  - Let A be a proposition and $\varphi \in \text{VAL}(\forall x.A)$ such that $x \notin \text{DOM}(\varphi)$,
    - Let $(\Gamma, \pi) \in \lfloor \forall x.A \rfloor_\varphi$ hence $\Gamma \vdash \pi : \varphi(\forall x.A) = \forall x.\varphi A$ since $x \notin \text{DOM}(\varphi)$. Then $\pi \in SN$ since the theory is strongly normalizing. Let $t \in \hat{T}$, then $\Gamma \vdash \pi t : (t/x)\varphi A$ hence $(\Gamma, \pi t) \in \lfloor A \rfloor_{\varphi + \langle x,t \rangle}$. Finally, $(\Gamma, \pi) \in \mathring{\forall}_T(t \mapsto \lfloor A \rfloor_{\varphi + \langle x,t \rangle}$.
    - If $(\Gamma, \pi) \in \mathring{\forall}_T(t \mapsto \lfloor A \rfloor_{\varphi + \langle x,t \rangle})$, then $\Gamma \vdash \pi : \varphi(\forall x.A)$ by definition.

□

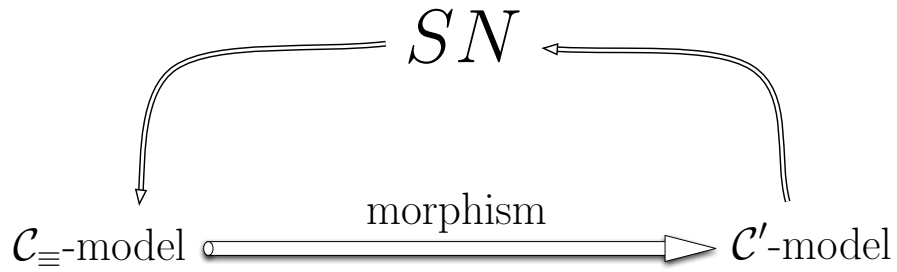
3.3 Theory-independent sound and complete reducibility candidates

We have introduced, for each theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_\equiv$ a LDTVA $\mathcal{C}_\equiv$ such that having a $\mathcal{C}_\equiv$-valued model is a sound and complete semantic condition for strong normalization property. But each LDTVA $\mathcal{C}_\equiv$ depends on the congruence relation $\equiv$, while we want to define a sufficient and necessary semantics for *all* strongly normalizing theories expressed in minimal deduction modulo. Notice that, as previously, since LDTVAs depend on the sets of closed terms of the studied theory, we define a class of LDTVAs , each corresponding to a class of theories in minimal deduction modulo based on many-sorted predicate languages which share the same sets of sorts and functions.

In this section, we introduce another LDTVA which we call $\mathcal{C}'$, independent of the theory studied, and we prove that having a $\mathcal{C}'$-valued model is also a sound and complete semantics for strongly normalizing theories in minimal deduction modulo.

3.3.1 The main idea

As we want $\mathcal{C}'$ to be theory-independent, we cannot assume properties of typing in its definition. Therefore we are not able to reuse the techniques developed in 3.2.3, since they depend explicitely on those assumptions on typing. In particular, we shall not be able to prove the converse of the adequacy lemma directly. But we can reuse the results of 3.2.3 in the following way: in order to prove completeness of $\mathcal{C}'$-models, we build a morphism of LDTVAs from each LDTVA $\mathcal{C}_{\equiv}$ corresponding to a strongly normalizing theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$, to $\mathcal{C}'$. Actually, we build a more general morphism: from each sub-LDTVA satisfying the substitution property (defined in 3.3.5) of each $\mathcal{C}_{\equiv}$ to $\mathcal{C}'$.



This way, as soon as a theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing, it has a $\mathcal{C}_{\equiv}$-valued model, as proved in 3.2, and this model is mapped to a $\mathcal{C}'$-valued model, by the morphism we are to build. This will prove that having a $\mathcal{C}'$-valued model is also a complete semantics for strongly normalizing theories, and we shall prove separately that it is still a sound semantics. This technique of building models by using morphisms of algebras is, up to our knowledge, innovative and could lead to other new results about normalization. We shall discuss on that point in section 6.2.2.

3.3.2 $\mathcal{C}'$, yet another algebra of reducibility candidates

Let us define, in this subsection the LDTVA $\mathcal{C}'$. We shall use, for this purpose, a notion of substitution with capture which we define below.

Definition 3.16. *For all proof-variables α, proof-terms π and π', we define $[\pi'/\alpha]\pi$ by induction on the structure of π:*

- *if π is a proof-variable β,*
  then $[\pi'/\alpha]\pi = \pi'$ if $\alpha = \beta$ and $[\pi'/\alpha]\pi = \pi$ otherwise,
- *if π is an abstraction $\lambda\gamma.\rho$, then $[\pi'/\alpha]\pi = \lambda\gamma.[\pi'/\alpha]\rho$*
- *if π is an abstraction $\lambda x.\rho$, then $[\pi'/\alpha]\pi = \lambda x.[\pi'/\alpha]\rho'$,*
- *if π is an application $\rho_1\rho_2$ then $[\pi'/\alpha]\pi = [\pi'/\alpha]\rho_1\ [\pi'/\alpha]\rho_2$,*
- *if π is an application $\rho_1 t$ then $[\pi'/\alpha]\pi = [\pi'/\alpha]\rho_1\ t$,*

Notice that renaming in π is not allowed in this case.

Example 3.1. *For all proof-variables α and proof-terms π,*
$[\pi/\alpha](\lambda\alpha.\alpha) = \lambda\alpha.\pi$ whereas $(\pi/\alpha)(\lambda\alpha.\alpha) = \lambda\alpha.\alpha$.

First, we naturally consider in this LDTVA sets of proof-terms again, and we do not mention contexts in the LDTVA we build, since we do not want to depend on typing anymore. And we define the domain of $\mathcal{C}'$ as the set of sets E of proof-terms that satisfy the usual (CR_1), (CR_2) and another modified version of (CR_3) below.

Definition 3.17.
For all sets E of proof-terms, we define (or recall) the following properties :

(CR_1) *For all $\pi \in E$, $\pi \in SN$.*

(CR_2) *For all $\pi \in E$, for all $\pi' \in \mathcal{T}$ such that $\pi \to \pi'$, then $\pi' \in E$.*

(CR_3') *For all $n \in \mathbb{N}$, for all $\nu, \mu_1, \ldots, \mu_n \in \mathcal{T}$, if*

- *for all $i \leq n$, μ_i is neutral and non-normal,*
- *$\forall \rho_1, \ldots, \rho_n \in \mathcal{T}$ such that for all $i \leq n$, $\mu_i \to \rho_i$, $[\rho_i/\alpha_i]_{i\leq n}\nu \in E$*

then $[\mu_i/\alpha_i]_{i\leq n}\nu \in E$.

This (CR_3') property is different from the usual (CR_3) on two points.

First, as we said in 3.1.2, the problem with usual reducibility candidates for being complete comes from the fact that all neutral normal proof-terms belong to all candidates because of the (CR_3) property. With this new definition, we do not allow neutral normal proof-terms but only neutral non-normal proof-terms whose all one-step reducts are in the set. For example, unlike with the usual (CR_3) property, $\alpha\alpha$ can't be added to a set using this new property since it is normal.

As a second step, those neutral non-normal proof-terms are allowed not only at the root of a proof-term but at the root of different subtrees of this proof-term. For example, if we write $\delta = \lambda\alpha.\alpha\alpha$, $K = \lambda\alpha.\lambda\beta.\alpha$ and if $\lambda\gamma.\gamma$ is

in a set E, then $\lambda\gamma.(K\gamma\ \delta)$ can be added by (CR_3') to E: we take $\nu = \lambda\gamma.\alpha_1$ and $\mu_1 = K\gamma\ \delta$. μ_1 is neutral, non-normal and its only β-reduct is γ, with $[\gamma/\alpha_1]\nu = \lambda\gamma.\gamma$ which is in E. We can notice that $\lambda\gamma.(K\gamma\ \delta)$ is not neutral therefore it could not have been added with the usual (CR_3) property.

Finally, (CR_3') allows to add non-neutral proof-terms to a set E, unlike (CR_3), but these proof-terms will always reduce to an element of E (still unlike (CR_3)). We shall see in 3.3.5 how those two differences allow us to build morphisms from each LDTVA $\mathcal{C}_{\equiv}$ of a strongly normalizing theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ to $\mathcal{C}'$.

Notice that a specific case of this (CR_3') property is the (CR_3) property with an additional condition on the neutral proof-term considered: to be non-normal (take $n = 1$ in the definition of (CR_3')).

For the interpretation of the connective $\Rightarrow$, we use, in $\mathcal{C}'$, the usual definition of reducibilty candidates.

Definition 3.18 ($\tilde{\Rightarrow}$)**.**
For all $E, F \subseteq \mathcal{T}$, $E \tilde{\Rightarrow} F = \{\pi \in SN$ such that for all $\pi' \in E$, $\pi\pi' \in F\}$.

Then we verify that this defines a function from $\mathcal{C}' \times \mathcal{C}'$ to $\mathcal{C}'$. As for the usual (CR_3) property, if E and F are sets of proof-terms, which satisfy (CR_3') then $E \tilde{\Rightarrow} F$ also satisfy (CR_3').

Lemma 3.14. *$\tilde{\Rightarrow}$ is a function from $\mathcal{C}' \times \mathcal{C}'$ to $\mathcal{C}'$.*

Proof. Let $E, F \in \mathcal{C}'$ and $\pi \in E \tilde{\Rightarrow} F$,

(CR_1) $\pi \in SN$, by definition.

(CR_2) If ρ is a one-step reduct of π, then for all $\pi' \in E$, $\rho\pi'$ is a one-step reduct of $\pi\pi'$.

(CR_3') If there exists $\nu, \mu_1, \ldots, \mu_n \in \mathcal{T}$, such that each μ_i is neutral non-normal, $\tau = [\mu_i/\alpha_i]_{i \leq n}\ \nu$ and for all $(\rho_i)_{i \leq n} \subseteq \mathcal{T}$, such that for all $i \leq n$,
$\mu_i \rightarrow \rho_i$, then $[\rho_i/\alpha_i]_{i \leq n}\ \nu \in E \tilde{\Rightarrow} F$. Then, for all $\pi' \in E$, $\tau\pi' = [\mu_i/\alpha_i]_{i \leq n}\ \nu\pi' = [\mu_i/\alpha_i]_{i \leq n}\ \nu'$ with $\nu' = \nu\pi'$. And for all $(\rho_i)_{i \leq n} \subseteq \mathcal{T}$, such that for all $i \leq n$, $\mu_i \rightarrow \rho_i$, we have $[\rho_i/\alpha_i]_{i \leq n}\ \nu' = [\rho_i/\alpha_i]_{i \leq n}\ \nu\ \pi' \in F$ by hypothesis, therefore $\tau\pi' \in F$ as it satifies (CR_3'). And finally, $\tau \in E \tilde{\Rightarrow} F$.

□

And we define the interpretation of the universal quantifier in a functional way, as in 3.2.1.

Definition 3.19 ($\tilde{\mathcal{A}}_T$)**.**
For all sorts T, $\tilde{\mathcal{A}}_T$ is the set of all function from $\hat{T}$ to $\mathcal{C}'$.

Definition 3.20 ($\tilde{\forall}_T$). *For all sorts T and function $f \in \tilde{\mathcal{A}}_T$,*
$\tilde{\forall}_T.f = \{\pi \in SN$ *such that for all* $t \in \hat{T}$, $\pi t \in f(t)\}$

Then we verify that this defines a function from $\tilde{\mathcal{A}}_T$ to $\mathcal{C}'$.

Lemma 3.15. *For all sorts T and $f \in \tilde{\mathcal{A}}_T$, $\tilde{\forall}_T f \in \mathcal{C}'$.*

Proof. Let T be a sort, $f \in \tilde{\mathcal{A}}_T$ and $\pi \in \tilde{\forall}_T.f$.

(CR_1) By definition.

(CR_2) Let π' be such that $\pi \to \pi'$. Then for all $t \in \hat{T}$, $\pi' t$ is a one-step reduct of πt.

(CR_3') If there exists $\nu, \mu_1, \ldots, \mu_n \in \mathcal{T}$, such that each μ_i is neutral non-normal, $\tau = [\mu_i/\alpha_i]_{i\leq n}\ \nu$ and for all $(\rho_i)_{i\leq n} \subseteq \mathcal{T}$, such that for all $i \leq n$, $\mu_i \to \rho_i$, then $[\rho_i/\alpha_i]_{i\leq n}\ \nu \in \tilde{\forall}_T.f$. Then, for all $t \in \hat{T}$, $\tau t = [\mu_i/\alpha_i]_{i\leq n}\ \nu t = [\mu_i/\alpha_i]_{i\leq n}\ \nu'$ with $\nu' = \nu t$. And for all $(\rho_i)_{i\leq n} \subseteq \mathcal{T}$, such that for all $i \leq n$, $\mu_i \to \rho_i$, we have $[\rho_i/\alpha_i]_{i\leq n}\ \nu' = [\rho_i/\alpha_i]_{i\leq n}\ \nu\ t \in f(t)$ by hypothesis, therefore $\tau t \in f(t)$ as it satifies (CR_3'). And finally, $\tau \in \tilde{\forall}_T.f$.

□

Remark 3.7. *As in 3.2.1, we could have defined $\tilde{\forall}_T$ in a simpler way, without assuming explicitely that proof-terms in $\tilde{\forall}_T.f$ are strongly normalizing:* $\tilde{\forall}_T.f = \{\pi$ *such that for all* $t \in \hat{T}$, $\pi t \in f(t)\}$. *But this would have led us, as previously, to consider only theories such that there exists at least one closed term of each sort.*

We finally define formally $\mathcal{C}'$ with the previous definitions.

Definition 3.21 ($\mathcal{C}'$). *$\mathcal{C}'$ is the* LDTVA $\langle \mathcal{C}', \tilde{\Rightarrow}, (\tilde{\mathcal{A}}_T), (\tilde{\forall}_T)\rangle$.

3.3.3 Soundness of non-empty $\mathcal{C}'$-models

As explained in 3.1.1, we can notice that in the usual adequacy lemma, (CR_3) is only used on proof-variables and on non-normal proof-terms. For non-normal proof-terms, adapting the proof of this lemma to the new property (CR_3') can be done without any difficulty as explained below. But with the new (CR_3') property, we cannot deduce that $\mathcal{C}'$ models are non-empty from the fact that they contain all proof-variables. We have therefore to suppose explicitely that the models we consider are non-empty. Another difference with the original adequacy lemma for pre models of section 2.3.2 comes from our synchronized version of the interpretation of the universal quantifier. Due to this synchronization, our new form of adequacy lemma quantifies on only one valuation (and not two as previously) as we shall see below.

Lemma 3.16. *If $[\![.]\!]_.$ is a non-empty $\mathcal{C}'$-valued model of a theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$, then for all $A \in \mathcal{P}$, contexts Γ, $\varphi \in \text{VAL}(A) \cap \text{VAL}(\Gamma)$, $\pi \in \mathcal{T}$ and σ substitutions such that for all declarations $\alpha : B$ in Γ, $\sigma\alpha \in [\![B]\!]_\varphi$, we have:*

$$\text{if } \Gamma \vdash \pi : A \text{ then } \sigma\varphi\pi \in [\![A]\!]_\varphi.$$

Proof. By induction on the length of the derivation of $\Gamma \vdash \pi : A$. by case analysis on the last rule used. If the last rule used is :

- axiom: in this case, π is a variable α, and Γ contains a declaration $\alpha : B$ with $A \equiv B$ (therefore $\varphi A \equiv \varphi B$). Then $\sigma\varphi\pi = \sigma\alpha \in [\![B]\!]_\varphi = [\![A]\!]_\varphi$.

- $\Rightarrow$-intro: in this case, π is an abstraction $\lambda\alpha.\tau$, and we have $\Gamma, \alpha : B \vdash \tau : C$ with $A \equiv B \Rightarrow C$. Let σ' such that for all variables β declared in Γ, $\sigma'\beta = \sigma\beta$ and $\sigma'\alpha$ is an element of $[\![B]\!]_\varphi$. Then $\sigma'\varphi\tau \in [\![C]\!]_\varphi$ by induction hypothesis (and $\sigma'\varphi\tau$ is in SN, therefore $\sigma\varphi\pi$ is also in SN). Let $\pi' \in [\![B]\!]_\varphi$, we prove by induction on the sum of both maximal lengths of a reductions sequence from $\sigma\varphi(\lambda\alpha.\tau)$ and π' (each in SN) that every one-step reduct of the neutral non-normal proof-term $\sigma\varphi(\lambda\alpha.\tau)\ \pi'$ is in $[\![C]\!]_\varphi$. If the one-step reduct is $\sigma\varphi(\pi'/\alpha)\tau$, we conclude by induction hypothesis (on the length of the derivation) as $\pi' \in [\![B]\!]_\varphi$. Otherwise, the reduction takes place either in $\sigma\varphi(\lambda\alpha.\tau)$, either in π'. We conclude by induction hypothesis on the sum of the maximal lengths of reductions sequence from $\sigma\varphi(\lambda\alpha.\tau)$ and π'. And the fact that both $[\![B]\!]_\varphi$ and $[\![B \Rightarrow C]\!]_\varphi$ satisfy (CR_2). Finally, $\sigma\varphi(\lambda\alpha.\tau)\ \pi' \in [\![C]\!]_\varphi$, since it satisfies ($\text{CR}_3$') and $\sigma\varphi(\lambda\alpha.\tau)\ \pi'$ is neutral, non-normal. Hence $\sigma\varphi(\lambda\alpha.\tau) \in [\![B]\!]_\varphi \tilde{\Rightarrow} [\![C]\!]_\varphi = [\![B \Rightarrow C]\!]_\varphi = [\![A]\!]_\varphi$

- $\Rightarrow$-elim: in this case, π is an application $\rho\tau$, and we have $\Gamma \vdash \rho : C \equiv B \Rightarrow A$ and $\Gamma \vdash \tau : B$. Therefore, by induction hypothesis, $\sigma\varphi\rho \in [\![B \Rightarrow A]\!]_\varphi = [\![B]\!]_\varphi \tilde{\Rightarrow} [\![A]\!]_\varphi$ and $\sigma\varphi\tau \in [\![B]\!]_\varphi$. Therefore $\sigma\varphi(\rho\tau) \in [\![A]\!]_\varphi$.

- $\forall$-intro: in this case, π is a term abstraction $\lambda x.\pi'$ and we have $\Gamma \vdash \pi' : B$ with $A \equiv \forall x.B$. Let $t \in \hat{T}$ (where T is the sort of x), and $\varphi' = \varphi + \langle x, t \rangle$. Then $\sigma\varphi'\pi' = \sigma\varphi(t/x)\pi' \in [\![B]\!]_{\varphi'}$, by induction hypothesis. Therefore, $\sigma\varphi(\lambda x.\pi') \in \tilde{\forall}_T(t \mapsto [\![B]\!]_{\varphi+\langle x,t\rangle} = [\![A]\!]_\varphi$ (by induction on the maximal length of a reductions sequence from πt, with $t \in \hat{T}$, using the fact that for all $t \in \hat{T}$, $[\![B]\!]_{\varphi+\langle x,t\rangle}$ satisfies (CR_2) and (CR_3').

- $\forall$-elim: in this case, π is an application ρt, and we have $\Gamma \vdash \rho : \forall x.B$ with $A = (t/x)B$ and $x \notin FV(\Gamma)$. By induction hypothesis, we have $\sigma\varphi\rho \in [\![\forall x.B, \varphi]\!] = \tilde{\forall}_T(t \mapsto [\![B]\!]_{\varphi+\langle x,t\rangle})$. And therefore $\sigma\varphi(\rho t) = \sigma\varphi\rho\ (\varphi t) \in [\![B]\!]_{\varphi+\langle x,\varphi t\rangle} = [\![(t/x)B]\!]_\varphi = [\![A]\!]_\varphi$

□

Proposition 3.4 (Soundness)**.**
If $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ has a non-empty $\mathcal{C}'$-valued model, then $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing.

Proof. If $[\![.]\!]_{\cdot}$ is a non-empty $\mathcal{C}'$-valued model of $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ then for all typing judgement $\Gamma \vdash \pi : A$ and φ as in the previous proposition, there exists a substitution σ as in the previous proposition (as $[\![.]\!]_{\cdot}$ is non-empty) and we have $\sigma\varphi\pi \in [\![A]\!]_{\varphi}$ hence $\sigma\varphi\pi \in SN$, therefore $\pi \in SN$.

□

Finally, we proved that this (CR_3') property, slightly different from the usual (CR_3) is sufficient to prove the adequacy lemma, provided that we suppose explicitely the non-emptiness of our models. Hence having a non-empty $\mathcal{C}'$-valued model is still a sound semantics for strongly normalizing theories in minimal deduction modulo. Let us now prove, in the following, that this definition of $\mathcal{C}'$ is also adequate to prove completeness of (non-empty) $\mathcal{C}'$-valued models.

3.3.4 Defining a function from $\mathcal{C}_{\equiv}$ to $\mathcal{C}'$

In order to build a function from $\mathcal{C}_{\equiv}$ to $\mathcal{C}'$, the first idea is to filter sets in $\mathcal{C}_{\equiv}$, by a same context (i.e. only consider proof-terms that are in a element of $\mathcal{C}_{\equiv}$ associated to a same context). This means that from each set in $\mathcal{C}_{\equiv}$ we shall only consider proof-terms that are associated to a same context in this set. In order to obtain big enough sets in $\mathcal{C}'$, we need to consider a context Δ which contains, for each proposition, an infinity of variables declared proofs of this proposition. But since we can only consider finite contexts in typing judgements, we shall filter sets in $\mathcal{C}_{\equiv}$ by finite subcontexts of Δ.

Definition 3.22 (Δ)**.** *We consider a context which contains an infinite number of variables for each proposition.* $\Delta = (\beta_i^A : A)_{A \in \mathcal{P}, i \in \mathbb{N}}$.

Let us define now the notion of *leaf* of a proof-term. The leaves of a proof-term are its first β-reducts which are normal or non-neutral. Intuitively, when proving that a proof-term π belongs to a set E satisfying (CR_3'), the leaves of π are the first reducts of π on which we cannot use (CR_3') to prove that they are in E. In other words, it is sufficient that all leaves of π belong to E, in order to prove that π is in E (as soon as E satisfies (CR_3')).

Definition 3.23 (Leaves)**.**
For all proof-terms ρ, π, ρ is a leaf *of π if and only if ρ is normal or non-neutral and there exists $n \geq 0$ and $\pi_1 \dots \pi_{n-1}$ neutral non-normal terms such that $\pi = \pi_1 \to \dots \to \pi_{n-1} \to \rho$). We call $\mathcal{L}(\pi)$ the set of leaves of π.*

Remark 3.8. *The only leaf of a normal or non-neutral proof-term is itself. If π is a neutral non-normal proof-term, then $\rho \in \mathcal{L}(\pi)$ if and only if there exists a one-step reduct π' of π such that $\rho \in \mathcal{L}(\pi')$.*

Then we define our function $Cl(.)$ as follows. First we filter sets in $\mathcal{C}_{\equiv}$ by our context Δ. But we can notice that those filtered sets cannot satisfy (CR_3'). Indeed, ill-typed proof-terms can be added to a set using this (CR_3') property as we have seen previously, whereas those filtered sets only contain well-typed proof-terms in Δ. Therefore we have to saturate those sets by (CR_3') in order to obtain sets in $\mathcal{C}'$. Notice, that in the following definition, we use the notion of leaf in order to obtain directly a (CR_3') big-step expansion whereas (CR_3') is originally defined as a small-step expansion.

Definition 3.24 (Cl)**.** *For all $E \subseteq \mathcal{U}$, we define $Cl(E)$ as follows :*
for all $k \in \mathbb{N}$,

- $Cl^0(E) = \{\pi,\ \exists \Delta' \subseteq \Delta$ *finite such that* $(\Delta', \pi) \in E\}$
- $Cl^{k+1}(E) = \{\pi \in \mathcal{T}$, *such that* $\exists n \in \mathbb{N}$*:*
  $\exists \nu_\pi \in \mathcal{T}, \exists (\mu_i)_{i\leq n} \subseteq SN$, *each neutral non-normal s.t.*
  $\pi = [\mu_i/\alpha_i]_{i\leq n}\ \nu_\pi$ *and* $\forall (\rho_i)_{i\leq n} \subseteq \mathcal{T}$, *s.t.* $\forall i \leq n, \rho_i \in \mathcal{L}(\mu_i)$,
  we have $[\rho_i/\alpha_i]_{i\leq n}\ \nu_\pi \in Cl^k(E)\}$
- $Cl(E) = \cup_{j\in\mathbb{N}} Cl^j(E)$

Remark 3.9. *Notice that for all $E \in \mathcal{C}_{\equiv}$ and $k \in \mathbb{N}$, $Cl^k(E) \subseteq Cl^{k+1}(E)$ (take $n = 0$ in the definition).*

Lemma 3.17. *For all $E \in \mathcal{C}_{\equiv}$, if $E \neq \emptyset$, then $Cl(E) \neq \emptyset$.*

Proof. If $E \neq \emptyset$, there exists $(\Gamma, \pi) \in E$, let us write Γ' and π' for the context Γ and the proof-term π where proof-variables are renamed in order to obtain $\Gamma' \subseteq \Delta$. Then (Γ', π') is also in E and therefore in $Cl^0(E) \subseteq Cl(E)$, hence $Cl(E) \neq \emptyset$. □

Let us now prove that $Cl(.)$ maps each element of $\mathcal{C}_{\equiv}$ to an element of $\mathcal{C}'$. Although this statement may seem quite intuitive, we shall see that it is not so simple to prove.

Lemma 3.18.
For all $\pi \in SN$, if π is not isolated then there exists an abstraction τ such that for all abstractions τ' such that $\pi \to^* \tau'$, we have $\tau \to^* \tau'$.

We call τ the primary leaf *of π.*

Remark 3.10. *This definition of* primary leaf *is very closed to C. Riba's definition of the notion of* princpal reduct *[46]. The only difference is that in our case, the primary leaf has to be an abstraction.*

Proof. If π is not isolated, then all reductions sequences from π reach a non-neutral proof-term, by confluency. Notice that the not-neutral reducts of π are either all proof-abstractions, either all term-abstractions, by confluency. In particular, the head-reduction sequence from π reach a not-neutral proof-term. Let τ be the first non-neutral proof-term reached in this reductions sequence. If π is not neutral, then $\pi = \tau$, therefore all non-neutral reducts of π are obviously reducts of $\pi = \tau$. Otherwise, π is neutral and has the form $\rho\ \theta_1 \dots \theta_n$, with each θ_i a term or a proof-term, $n > 0$ (as π is neutral), and ρ is not neutral (as π is not isolated). We suppose, in the following that ρ is a proof-abstraction $\lambda\alpha.\rho'$, then θ_1 is a proof-term (the proof is the same in the case of a term-abstraction). Let us prove by induction on the maximal length of a reductions sequence from π ($\in SN$), that each non-neutral reduct of π is a reduct of τ. If this maximal length is equal to zero, then π cannot be neutral. Otherwise, let $\pi', \pi'' \in \mathcal{T}$ such that π'' is not neutral and $\pi = (\lambda\alpha.\rho')\ \theta_1 \dots \theta_n \to \pi' \to^* \pi''$.

- If $\pi' = (\theta_1/\alpha)\rho'\ \theta_2 \dots \theta_n$, then π' is the head-one-step-reduct of π, therefore π'' is a reduct of τ, by induction hypothesis.

- If $\pi' = (\lambda\alpha.\rho'')\ \theta_1 \dots \theta_n$, with $\rho' \to \rho''$, let τ' be the first non-neutral term reached in the head-reduction of $(\theta_1/\alpha)\rho''\ \theta_2 \dots \theta_n$ (then τ' is also the first not neutral term reached in the head-reduction of π'). By induction hypothesis, τ' is a reduct of τ. Moreover, π'' is a reduct of τ' by induction hypothesis (as τ' is the first non-neutral head-reduct of $(\theta_1/\alpha)\rho'\ \theta_2 \dots \theta_n$). Finally, π'' is a reduct of τ.

- If $\pi' = (\lambda\alpha.\rho')\ \theta_1 \dots \theta_{i-1}\ \theta_i'\ \theta_{i+1} \dots \theta_n$, we use the same sort of argument than in the previous point.

□

Definition 3.25 ($\pi \Vdash \alpha$)**.** *For all $\alpha \in \mathcal{X}$ and $\pi \in \mathcal{T}$, we write $\pi \Vdash \alpha$ the number of occurrences of α in π.*

Definition 3.26 ($\mathcal{K}$)**.**
$\mathcal{K} = \{\ \langle \nu, n, (\mu_1, \dots, \mu_n)\rangle$ *such that*
. $n \in \mathbb{N},\ \nu \in \mathcal{T},\ \mu_1 \dots \mu_n \in SN$
. *for all* $i \leq n,\ \nu \Vdash \alpha_i \leq 1$
. *for all* $(\rho_i)_{i\leq n}$ *each respectively in* $\mathcal{L}(\mu_i),\ [\rho_i/\alpha_i]_{i\leq n}\nu \in SN\ \}$

Definition 3.27 ($\rightharpoonup$)**.**
Let $\eta = \langle \nu, n, (\mu_1, \dots, \mu_n)\rangle$ *and* $\eta' = \langle \nu', n', (\mu_1', \dots, \mu_n')\rangle$ *in* $\mathcal{K}$.
We say that $\eta \rightharpoonup \eta'$ *if and only if one of the following conditions occurs:*

(a) $\nu = \nu'$ *and there exists* $i_0 \leq n$ *such that for all* $i \neq i_0$, $\mu_i = \mu_i'$, μ_{i_0} *is neutral and* $\mu_{i_0} \to \mu_{i_0}'$

(b) *there exists* $i_0 \leq n$ *such that* μ_{i_0} *is not neutral,* $\nu' = (\mu_{i_0}/\alpha_{i_0})\nu$, $n' = n - 1$ *and* $\mu'_1 \dots \mu'_{n'} = \mu_1 \dots \mu_{i_0 - 1}\ \mu_{i_0+1} \dots \mu_n$

(c) $\nu \to \nu'$ *and the* μ'_i *are copies of the* μ_i *resulting of the linearization of the occurrences of the variables* α_i *in* ν'.

Definition 3.28 ($\ell(\pi)$)**.**
For all $\pi \in SN$, *we define* $\ell(\pi)$ *as follows: if* π *is isolated,* $\ell(\pi) = \alpha_0$ *(a special variable), otherwise,* $\ell(\pi)$ *is the primary leaf of* π.

Definition 3.29 ($||\eta||$)**.**
For all $\eta = \langle \nu, n, (\mu_1, \dots, \mu_n) \rangle \in \mathcal{K}$, $||\eta|| = (\ell(\mu_i)/\alpha_i)_{i \leq n}\nu$.

Remark 3.11. *For all* $\eta \in \mathcal{K}$, $||\eta|| \in SN$, *by definition.*

Lemma 3.19. *For all* $\eta, \eta' \in \mathcal{K}$, *if* $\eta \rightharpoonup \eta'$ *then* $||\eta|| \to^* ||\eta'||$.

Proof. by case analysis on $\eta \rightharpoonup \eta'$.

(a) If the μ_{i_0} which is reduced (on μ'_{i_0}) is isolated then $\ell(\mu_{i_0}) = \ell(\mu'_{i_0}) = \alpha_0$, therefore $||\eta|| = ||\eta'||$. Otherwise $\ell(\mu_{i_0}) \to^* \ell(\mu'_{i_0})$, by lemma 3.18, therefore $||\eta|| \to^* ||\eta'||$.

(b) In this case, $||\eta|| = ||\eta'||$.

(c) In this case, $||\eta|| \to ||\eta'||$ (as $\nu \to \nu'$).

□

Lemma 3.20. *All* $\rightharpoonup$*-reductions sequences from an element* η *of* $\mathcal{K}$ *are finite.*

Proof. As all the μ_i are in SN (and n is finite), there can only be a finite number of consecutive (a) and (b) reductions. As $||\eta|| \in SN$, there can only be a finite number of (c) reductions from η, by lemma 3.19. Hence there cannot be an infinite $\rightharpoonup$-reductions sequence from η. □

Then we can use the previous lemmas in order to prove that $Cl(.)$ maps elements of $\mathcal{C}_{\equiv}$ to elements of $\mathcal{C}'$.

Proposition 3.5.
For all $E \in \mathcal{C}_{\equiv}$, $Cl(E) \in \mathcal{C}'$.

Proof. Let $E \in \mathcal{C}_{\equiv}$.

(CR_2) Let $\pi \in Cl(E)$ and $\pi' \in \mathcal{T}$ such that $\pi \to \pi'$. Then there exists (a minimal) $k \in \mathbb{N}$ such that $\pi \in Cl^k(E)$. By induction on k.

- If $k = 0$, then there exists $\Delta' \subseteq \Delta$, finite, such that $(\Delta', \pi) \in E$, therefore $(\Delta', \pi') \in E$ since it satisfies ($\mathrm{CR}_{2_{\equiv}}$).

- If $k > 0$, then $\pi = [\mu_i/\alpha_i]_{i\leq n}\nu$ with each μ_i in SN, neutral and non-normal, and such that $\forall i \leq n$, and for all $(\rho_i)_{i\leq n}$ such that for all $i \leq n$, $\rho_i \in \mathcal{L}(\mu_i)$, we have $[\rho_i/\alpha_i]_{i\leq n}\nu \in Cl^{k-1}(E)$. We suppose that for all $i \leq n$, $\nu \shortparallel \alpha_i \leq 1$. As each μ_i is neutral:
  - Either $\pi' = [\mu'_{i_0}/\alpha_{i_0}][\mu_i/\alpha_i]_{i\neq i_0}\nu$, with $\mu_{i_0} \to \mu'_{i_0}$. In this case,
    - if $\mu'_{i_0} \in \mathcal{L}(\mu_{i_0})$ then $\pi' = [\mu_i/\alpha_i]_{i\neq i_0}\nu''$, with $\nu'' = (\mu'_{i_0}/\alpha_{i_0})\nu$, , and for all $(\rho_i)_{i\leq n}$ such that $\forall i \neq i_0$, $\rho_i \in \mathcal{L}(\mu_i)$, we have $[\rho_i/\alpha_i]_{i\leq n}\nu'' \in Cl^{k-1}(E)$, hence $\pi' \in Cl^k(E)$.
    - Otherwise, μ'_{i_0} is neutral, non-normal and all its leaves are leaves of μ_{i_0}, hence $\pi' \in Cl^k(E)$.
  - Either $\pi' = [\mu_i/\alpha_i]_{i\leq n}\nu$ with $\nu \to \nu'$. In this case, we conclude by the fact that $Cl^{k-1}(E)$ satisfies (CR_2) by induction hypothesis.

(CR_1) Let $\pi \in Cl(E)$, then there exists (a minimal) $k \in \mathbb{N}$ such that $\pi \in Cl^k(E)$. By induction on k.

- If $k = 0$, then there exists $\Delta' \subseteq \Delta$, finite, such that $(\Delta', \pi) \in E$, therefore $\pi \in SN$ since E satisfies ($\mathrm{CR}_{1_\equiv}$).
- If $k > 0$, then $\pi = [\mu_i/\alpha_i]_{i\leq n}\nu$ with each μ_i in SN, neutral and non-normal, and such that $\forall i \leq n$, and $\forall(\rho_i)_{i\leq n}$ such that for all $i \leq n$, $\rho_i \in \mathcal{L}(\mu_i)$, we have $[\rho_i/\alpha_i]_{i\leq n}\nu \in Cl^{k-1}(E) \subseteq SN$, by induction hypothesis. Then, if we suppose that for all $i \leq n$, $\nu||\alpha_i \leq 1$, if $\pi \to \pi'$, and if we write $\eta_\pi = \langle \nu, n, (\mu_1, \ldots, \mu_n)\rangle$ (and the same for $\eta_{\pi'}$, since $\pi' \in Cl^k(E)$ as explained in the previous point), then $\eta_\pi \rightharpoonup \eta_{\pi'}$. Hence all reductions sequences from π are finite, by lemma 3.20.

(CR_3') Let $n \in \mathbb{N}$, $\nu, \mu_1, \ldots, \mu_n \in \mathcal{T}$, such that for all $i \leq n$, μ_i is neutral and non-normal, and for all $\mu'_1, \ldots, \mu'_n \in \mathcal{T}$ such that $\forall i \leq n$, $\mu_i \to \mu'_i$, $[\mu'_i/\alpha_i]_{i\leq n}\nu \in Cl(E)$. Since the number of one-step reducts of a term is finite, there exists $k \in \mathbb{N}$, such that for all $\mu'_1, \ldots, \mu'_n \in \mathcal{T}$ such that $\forall i \leq n$, $\mu_i \to \mu'_i$, we have $[\mu'_i/\alpha_i]_{i\leq n}\nu \in Cl^k(E)$. Therefore, for all $\rho_1 \ldots \rho_n$ each respectively a leaf of $\mu_1 \ldots \mu_n$, $[\rho_i/\alpha_i]_{i\leq n}\nu \in Cl^k(E)$ since it satisfies (CR_2) and each μ_i is neutral non-normal. Finally, $[\mu_i/\alpha_i]_{i\leq n}\nu \in Cl^{k+1}(E)$.

□

3.3.5 Proving that this function is a morphism

Now that we have defined our function $Cl(.)$ and have proved that it maps elements of $\mathcal{C}_{\equiv}$ to elements of $\mathcal{C}'$, we still need to prove that it is a morphism of LDTVAs , in order to induce a mapping from $\mathcal{C}_{\equiv}$-models to $\mathcal{C}'$-models. We shall prove that $Cl(.)$ is a morphism from each LDTVA $\mathcal{C}_{\equiv}$ which corresponds to a strongly normalizing theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$, to $\mathcal{C}'$. More generally, we shall prove that $Cl(.)$ is a morphism from each LDTVA $\mathcal{C}_{\equiv}$ satisfying the substitution property, to $\mathcal{C}'$.

$\Rightarrow$-morphism

Let us first focus on the $\Rightarrow$-property of morphisms. As we shall see, $Cl(.)$ is not a morphism on the entire LDTVA $\mathcal{C}_{\equiv}$, but on each subLDTVA of $\mathcal{C}_{\equiv}$ which satisfies the substitution property. As explained below, satisfying the substitution property for a subLDTVA of $\mathcal{C}_{\equiv}$ means that each set of this subLDTVA is stable by well-typed substitution.

Definition 3.30. *For all $E \in \mathcal{C}_{\equiv}$,*
we say that E satisfies the substitution property if and only if
for all $F \in \mathcal{C}_{\equiv}$, proof-variables α, $\pi, \pi' \in \mathcal{T}$ and contexts Γ, Γ',
if $(\Gamma, \alpha) \in E$, $(\Gamma', \pi') \in E$ and $(\Gamma, \pi) \in F$ then $(\Gamma\Gamma', (\pi'/\alpha)\pi) \in F$.

*We say that a sub*LDTVA *of $\mathcal{C}_{\equiv}$ satisfies the substitution property if and only if each of its elements satisfies the substitution property.*

We point out, in the following lemma, the main property of subLDTVAs which satisfy the substitution property. If $E, F \in \mathcal{C}_{\equiv}$ such that F satisfies the substitution property, then it is sufficient to prove that a proof-term maps proof-variables in E to F, to prove that it is in $E \mathring{\Rightarrow} F$. This property will be very useful to prove that $Cl(.)$ is a $\Rightarrow$-morphism from $\mathcal{C}_{\equiv}$ to $\mathcal{C}'$.

Lemma 3.21. *For all $E, F \in \mathcal{C}_{\equiv}$ such that F satisfies the substitution property, for all $(\Gamma, \pi) \in \mathcal{U}$, if for all $(\Gamma', \alpha) \in E$, $(\Gamma\Gamma', \pi\alpha) \in F$, then $(\Gamma, \pi) \in E \mathring{\Rightarrow} F$.*

Proof. In this case, for all $(\Gamma', \pi') \in E$, $(\Gamma\Gamma', \pi\pi') = (\Gamma\Gamma', (\pi'/\alpha)\pi\alpha) \in F$ since F satisfies the substitution property, if α is not free in π (we can rename α in π if it is not the case).

□

Let us now prove some lemmas, which will be useful for the proof that $Cl(.)$ is a $\Rightarrow$-morphism.

Lemma 3.22. *For all $E \subseteq \mathcal{T}$ and $\pi \in \mathcal{T}$,*
If $\pi \in SN$, π is neutral non-normal and $\forall \rho \in \mathcal{L}(\pi)$, $\rho \in Cl(E)$, then $\pi \in Cl(E)$

Proof. As $\pi \in SN$, $\mathcal{L}(\pi)$ is defined and finite.
And, if we call $k_m = \max\{\min\{k, \rho \in Cl^k(E)\},\ \rho \in \mathcal{L}(\pi)\}$,
then $\pi \in Cl^{k_m+1}(E) \subseteq Cl(E)$.

□

Remark 3.12. *In the same way, if there exists $\nu_\pi \in \mathcal{T}$, and $(\mu_i)_{i\leq n} \subseteq SN$, each neutral non-normal such that $\pi = [\mu_i/\alpha_i]_{i\leq n}\ \nu_\pi$ and $\forall (\rho_i)_{i\leq n} \subseteq \mathcal{T}$, such that for all $i \leq n, \rho_i \in \mathcal{L}(\mu_i)$, then $[\rho_i/\alpha_i]_{i\leq n}\ \nu_\pi \in Cl(E)$, then we have $\pi \in Cl(E)$.*

Lemma 3.23. *For all $E, F \in \mathcal{C}_{\equiv}$,*
if F satisfies the substitution property, $(\Delta, \alpha) \in E$, and $(\alpha/\beta)\pi \in Cl(F)$
then $\lambda\beta.\pi \in Cl(E \mathring{\Rightarrow} F)$.

Proof. There exists a minimal k such that $(\alpha/\beta)\pi \in Cl^k(F)$.
By induction on k.

- if $k = 0$ then $(\Delta, \pi) \in F$ and π is normal. Since $(\Delta, \alpha) \in E$, we have $(\Delta, (\lambda\beta.\pi)\alpha) \in F$, by $(\mathrm{CR}_{3_{\equiv}})$. Therefore $(\Delta, \lambda\beta.\pi) \in E \mathring{\Rightarrow} F$, by substitution property. And $\lambda\beta.\pi \in Cl^0(E \mathring{\Rightarrow} F)$, since it is normal.
- if $k > 0$, then $(\alpha/\beta)\pi = [\mu_i/\alpha_i]_{i\leq n}\nu$ with each μ_i in SN, neutral and non-normal, and such that $\forall i \leq n$, and $\forall (\rho_i)_{i\leq n}$ such that for all $i \leq n$, $\rho_i \in \mathcal{L}(\mu_i)$, we have $[\rho_i/\alpha_i]_{i\leq n}\nu \in Cl^{k-1}(F)$, therefore $\lambda\beta.[\rho_i/\alpha_i]_{i\leq n}\nu = [\rho_i/\alpha_i]_{i\leq n}\lambda\alpha.\nu \in Cl(E \mathring{\Rightarrow} F)$, by induction hypothesis. Therefore $\lambda\beta.\pi \in Cl(E \mathring{\Rightarrow} F)$ by remark 3.12.

□

Lemma 3.24. *For all $E, F \in \mathcal{C}_{\equiv}$ and $\pi \in Cl(E) \tilde{\Rightarrow} Cl(F)$,*
π cannot reduce to a term-abstraction.

Proof. Let $\pi \in Cl(E) \tilde{\Rightarrow} Cl(F)$, then $\pi \in SN$. If π reduces to a term-abstraction, then its normal form is a term-abstraction $\lambda x.\rho$, by confluency. Let $\alpha \in \mathcal{X}$ such that $(\Delta, \alpha) \in E$, then $\alpha \in Cl(E)$ and $\pi\alpha \in Cl(F)$, therefore $(\lambda x.\rho)\alpha \in Cl(F)$, since F satisfies (CR_2). Moreover, $(\lambda x.\rho)\alpha$ is normal, therefore $(\lambda x.\rho)\alpha \in Cl^0(F)$. Hence $(\Delta, (\lambda x.\rho)\alpha) \in F$ and $\Delta \vdash (\lambda x.\rho)\alpha : A_F$. That's absurd.

□

We prove now that for all $E, F \in \mathcal{C}_{\equiv}$ such that F satisfies the substitution property, we have $Cl(E \mathring{\Rightarrow} F) = Cl(E) \tilde{\Rightarrow} Cl(F)$.

Proposition 3.6. *For all $E, F \in \mathcal{C}_{\equiv}$,*
if F satisfies the substitution property, then $Cl(E \mathring{\Rightarrow} F) = Cl(E) \tilde{\Rightarrow} Cl(F)$.

Proof. $\subseteq$ Let $\pi \in Cl(E \mathring{\Rightarrow} F)$,
then $\pi \in SN$ by (CR_1). Moreover there exists (a minimal) $k \in \mathbb{N}$, such that $\pi \in Cl^k(E \mathring{\Rightarrow} F)$. Let $\pi' \in Cl(E)$, then there exists (a minimal) $j \in \mathbb{N}$, such that $\pi' \in Cl^j(E)$. Let us show that $\pi\pi' \in Cl(F)$ by induction on $k + j$.

- If $k + j = 0$ then $(\Delta, \pi) \in E \mathring{\Rightarrow} F$ and $(\Delta, \pi') \in E$ therefore $(\Delta, \pi\pi') \in F$ and $\pi\pi' \in Cl^0(F)$.
- If $k > 0$, then there exists $\nu_\pi \in \mathcal{T}$, and $(\mu_i)_{i \leq n} \subseteq SN$, each neutral non-normal such that $\pi = [\mu_i/\alpha_i]_{i \leq n}\ \nu_\pi$ and $\forall (\rho_i)_{i \leq n} \subseteq \mathcal{T}$, such that for all $i \leq n, \rho_i \in \mathcal{L}(\mu_i)$, then $[\rho_i/\alpha_i]_{i \leq n}\ \nu_\pi \in Cl^{k-1}(E \mathring{\Rightarrow} F)$. Therefore $[\rho_i/\alpha_i]_{i \leq n}\ (\nu_\pi\ \pi') = [\rho_i/\alpha_i]_{i \leq n}\ \nu_\pi\ \pi' \in Cl(F)$ by induction hypothesis. Hence $\pi\pi' \in Cl(F)$, since it satisfies (CR_3').
- If $j > 0$, then there exists $\nu_{\pi'} \in \mathcal{T}$, and $(\mu_i)_{i \leq n} \subseteq SN$, each neutral non-normal such that $\pi' = [\mu_i/\alpha_i]_{i \leq n}\ \nu_{\pi'}$ and $\forall (\rho_i)_{i \leq n} \subseteq \mathcal{T}$, such that for all $i \leq n$, $\rho_i \in \mathcal{L}(\mu_i)$, then $[\rho_i/\alpha_i]_{i \leq n}\ \nu_{\pi'} \in Cl^{j-1}(E)$. Therefore $[\rho_i/\alpha_i]_{i \leq n}\ (\pi\ \nu_{\pi'}) = [\rho_i/\alpha_i]_{i \leq n}\ \pi\ \nu_{\pi'} \in Cl(F)$ by induction hypothesis. Hence $\pi\pi' \in Cl(F)$, since it satisfies (CR_3').

$\supseteq$ Let $\pi \in Cl(E) \mathring{\Rightarrow} Cl(F)$. Then $\pi \in SN$ and for all $\pi' \in Cl(E)$, $\pi\pi' \in Cl(F)$. By lemma 3.24, π cannot reduce to a term-abstraction.

- If π is a proof-abstraction $\lambda\alpha.\pi'$, let $\beta \in \mathcal{X}$ such that $\Delta \vdash \beta : A_E$, then $(\lambda\alpha.\pi')\beta \in Cl(F)$ and so does $(\beta/\alpha)\pi'$, by (CR_2). Therefore $\pi \in Cl(E \mathring{\Rightarrow} F)$ by lemma 3.23.
- If π is neutral and normal, let $\alpha \in \mathcal{X}$ such that $\Delta \vdash \alpha : A_E$, then $\pi\alpha \in Cl(F)$. Moreover π is neutral and normal, therefore $\pi\alpha$ is normal, hence $\pi\alpha \in Cl^0(F)$, *i.e.* $(\Delta, \pi\alpha) \in F$, with $(\Delta, \alpha) \in E$, therefore $(\Delta, \pi) \in E \mathring{\Rightarrow} F$, since F satisfies the substitution property.
  Finally, $\pi \in Cl^0(E \mathring{\Rightarrow} F)$, since it is normal.
- Otherwise, $\pi \in SN$, is neutral and non-normal. All its leaves are either neutral, either proof-abstractions, by lemma 3.24. And all these leaves are in $Cl(E) \mathring{\Rightarrow} Cl(F)$, since it satisfies ($\mathrm{CR}_2$), therefore they also are in $Cl(E \mathring{\Rightarrow} F)$, as we saw in the previous points. Finally, $\pi \in Cl(E \mathring{\Rightarrow} F)$, by lemma 3.22.

□

Let us now explain why we could not build a morphism from $\mathcal{C}_{\equiv}$ to the usual $\mathcal{C}$, and why the new property (CR_3') is adequate since it allows to add non-neutral proof-terms to a set. If we take π in $Cl(F)$ such that $(\Delta, \pi) \notin F$ and α not free in π, then $\lambda\alpha.\pi$ is in $Cl(E) \mathring{\Rightarrow} Cl(F)$ because for

all $\pi' \in Cl(E)$, $(\lambda\alpha.\pi)\pi'$ is neutral non-normal and all its reducts are in $Cl(F)$ (by induction on the maximal length of a reductions sequence from π': if π' is normal, then the only reduct of $(\lambda\alpha.\pi)\pi'$ is π which is in $Cl(F)$ by hypothesis). But $\lambda\alpha.\pi \notin Cl^0(E \mathring{\Rightarrow} F)$, since $(\Delta, \pi) \notin F$ and $\lambda\alpha.\pi$ could not had been added to $Cl(E \mathring{\Rightarrow} F)$ by the usual (CR_3) since it is not neutral.

$\forall$-morphism

We prove now that for all sorts T and $f \in \mathring{\mathcal{A}}_T$, $Cl(\mathring{\forall}_T f) = \tilde{\forall}_T\, Cl \circ f$. Notice that for all functions $f \in \mathring{\mathcal{A}}_T$, $Cl \circ f \in \tilde{\mathcal{A}}_T$.

Lemma 3.25. *For all $E \in \mathcal{C}_{\equiv}$, $k \in \mathbb{N}$, terms t, term-variables x, proof-terms π', if $(t/x)\pi \in Cl^k(E)$, then $(\lambda x.\pi)t \in Cl^k(E)$.*

Proof. By induction on k.

- If $k = 0$, by ($\mathrm{CR}_{3_{\equiv}}$) since $(\lambda x.\pi)t$ is neutral, non-normal, well-typed and we can prove by induciton on the maximal length of a reductions sequence from π ($\in SN$) that all its β-reducts are in $Cl^0(E)$.
- If $k > 0$, by induction hypothesis.

□

Lemma 3.26. *For all $\pi \in \mathcal{T}$ and $f \in \mathring{\mathcal{A}}_T$, if $\pi \in \tilde{\forall}_T\, Cl \circ f$ then there exists $k \in \mathbb{N}$ such that $\pi \in \tilde{\forall}_T\, Cl^k \circ f$.*

Proof. For all $E \in \mathcal{C}_{\equiv}$, if $\pi \in Cl(E)$ then $\pi \in SN$ and if k is the maximal length of a reductions sequence from π then $\pi \in Cl^k(E)$. Then, since the maximal length of reductions sequence from πt is the same for all $t \in \hat{T}$, if we note l this maximal length, we have, for all $t \in \hat{T}$, $\pi t \in Cl^l \circ f(t)$, therefore $\pi \in \tilde{\forall}_T Cl^l \circ f$.

□

Proposition 3.7. *For all sorts T and $f \in \mathring{\mathcal{A}}_T$, $Cl(\mathring{\forall}_T f) = \tilde{\forall}_T\, Cl \circ f$.*

Proof. $\subseteq$ Let $\pi \in Cl(\mathring{\forall}_T f)$, then there exists (a minimal) $k \in \mathbb{N}$ such that $\pi \in Cl^k(\mathring{\forall}_T f)$. By induction on k.

- If $k = 0$, $(\Delta, \pi) \in \mathring{\forall}_T f$ and $\pi \in SN$, then for all $t \in \hat{T}$, $(\Delta, \pi t) \in f(t)$, hence $\pi t \in Cl^0 \circ f(t)$. And $\pi \in \tilde{\forall}_T\, Cl \circ f$.
- If $k > 0$, then $\pi = [\mu_i/\alpha_i]_{i \leq n}\nu$, with each μ_i neutral non-normal and such that for all $(\rho_i)_{i \leq n}$ each respectively a leaf of μ_i, we have $[\rho_i/\alpha_i]_{i \leq n}\nu \in Cl^{k-1}(\mathring{\forall}_T f) \subseteq \tilde{\forall}_T\, Cl \circ f$, by induction hypothesis. Let $t \in \hat{T}$, then if we write $\nu' = \nu t$, we have $\pi t = [\mu_i/\alpha_i]_{i \leq n}\nu'$ and for all $(\rho_i)_{i \leq n}$ each respectively a leaf of μ_i, $[\rho_i/\alpha_i]_{i \leq n}\nu' = [\rho_i/\alpha_i]_{i \leq n}\nu\ t \in Cl \circ f(t)$. Therefore $\pi t \in Cl \circ f(t)$ by remark 3.12. Finally, $\pi \in \tilde{\forall}_T\, Cl \circ f$.

$\supseteq$ Let $\pi \in \tilde{\forall}_T\, Cl \circ f$, then, by lemma 3.26, there exists $k \in \mathbb{N}$ such that $\pi \in \tilde{\forall}_T Cl^k \circ f$. By induction on k.

- If $k = 0$, then there exists $t \in \hat{T}$ such that $\pi t \in Cl^0 \circ f(t)$. Hence $(\Delta, \pi t) \in f(t)$ and πt is normal. Hence π is normal and for all $t' \in \hat{T}$, $\pi t'$ is also normal, therefore, since $\pi t' \in Cl \circ f(t)$, we have, in particular, $\pi t' \in Cl^0 \circ f(t)$. Finally, for all $t' \in \hat{T}$, $(\Delta, \pi t') \in f(t)$, therefore $(\Delta, \pi) \in \mathring{\forall}_T f$, and $\pi \in Cl^0(\mathring{\forall}_T f)$, since it is normal.
- If $k > 0$, let $t \in \hat{T}$ such that $\pi t \in Cl^k \circ f(t)$. Therefore $\pi t = [\mu_i/\alpha_i]_{i\leq n}\nu$, with each μ_i neutral non-normal and such that for all $(\rho_i)_{i\leq n}$ each respectively a leaf of μ_i, we have $[\rho_i/\alpha_i]_{i\leq n}\nu \in Cl^{k-1} \circ f(t)$.
  - If $\nu \neq \alpha_1$, then $\nu = \nu' t$, with $\pi = [\mu_i/\alpha_i]_{i\leq n}\nu'$, and for all $(\rho_i)_{i\leq n}$ each respectively a leaf of μ_i, we have $[\rho_i/\alpha_i]_{i\leq n}\nu' \in Cl(\mathring{\forall}_T f)$, by induction hypothesis. We conclude by lemma 3.22.
  - Otherwise, every leaf of πt is in $Cl^{k-1} \circ f(t)$. If π is isolated, then all its leaves ρ are neutral and normal, hence ρt is a leaf of πt, therefore $\rho \in Cl(\mathring{\forall}_T f)$, by induction hypothesis, and we conclude by lemma 3.22. If π reduces to $\lambda x.\pi'$ then all leaves of $(t/x)\pi'$ are in $Cl^{k-1} \circ f(t)$, therefore, for all leaves ρ of π', we have $(\lambda x.\rho)t \in Cl^{k-1} \circ f(t)$, by lemma 3.25, hence $\lambda x.\rho \in Cl(\mathring{\forall}_T f)$, by induction hypothesis. And finally, $\lambda x.\pi' \in Cl(\mathring{\forall}_T f)$, and so does π.

□

Finally, we proved that $Cl(.)$ maps non-empty sets in $\mathcal{C}_{\equiv}$ to non-empty sets in $\mathcal{C}'$, and that it is a morphism from each subLDTVA of $\mathcal{C}_{\equiv}$ satisfying the substitution property to $\mathcal{C}'$. Hence $Cl(.)$ induce a mapping from non-empty $\mathcal{C}_{\equiv}$-valued models satisfying the substitution property (*i.e.* such that each interpretation of a proposition satisfies the substitution property) to non-empty $\mathcal{C}'$-models. Therefore the last thing to prove, in order to prove that having a non-empty $\mathcal{C}'$-model is a complete semantics for strong normalization in minimal deduction modulo, is that for all propositions A and $\varphi \in \textsc{Val}(A)$, $\lfloor A \rfloor_\varphi$ satisfies the substitution property, if $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing.

3.3.6 Completeness of non-empty $\mathcal{C}'$-models

Let us prove that for all propositions B and $\varphi \in \textsc{Val}(B)$, $\lfloor B \rfloor_\varphi$ satisfies the subtitution property when $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing. Let us first notice that it is a corollary of lemma 2.3 in the case of the version of

$\lfloor . \rfloor$. of the "short way" and that we do not need the hypothesis of strong normalization in this case. We prove, in the following lemma, that when the theory is strongly normalizing then the version of $\lfloor . \rfloor$. of the "long way" also satisfies the substitution property.

Lemma 3.27. *If $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalising, then for all $E \in \mathcal{C}_{\equiv}$, proof-variables α, $\pi, \pi' \in \mathcal{T}$, $B \in \mathcal{P}$, $\varphi \in \text{VAL}(B)$, and Γ, Γ' contexts such that $(\Gamma, \alpha) \in E$, $(\Gamma', \pi') \in E$ and $(\Gamma, \pi) \in [B]_\varphi$ (resp. $\lfloor B \rfloor_\varphi$) then $(\Gamma\Gamma', (\pi'/\alpha)\pi) \in [B]_\varphi$ (resp. $\lfloor B \rfloor_\varphi$).*

Proof. Notice that if the $[.]$. version of this lemma is true, then also is the $\lfloor . \rfloor$. version. Let us prove the $[.]$. version by induction on A.

- If A is atomic, we have $\Gamma \vdash \pi : \varphi B$, $\Gamma \vdash \alpha : A_E$ and $\Gamma' \vdash \pi' : A_E$, therefore, $\Gamma\Gamma' \vdash (\pi'/\alpha)\pi : \varphi B$. Moreover $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing, so $(\pi'/\alpha)\pi \in SN$ and $(\Gamma\Gamma', (\pi'/\alpha)\pi) \in [B]_\varphi$.

- If $B = C \Rightarrow D$, then $(\Gamma, \pi) \in [C]_\varphi \mathring{\Rightarrow} [D]_\varphi$. Let $(\Gamma'', \tau) \in [C]_\varphi$ such that τ doesn't contain α (by α-conversion). Then $(\Gamma\Gamma'', \pi\tau) \in [D]_\varphi$ therefore $(\Gamma\Gamma'\Gamma'', (\pi'/\alpha)(\pi\tau)) = (\Gamma\Gamma'\Gamma'', (\pi'/\alpha)\pi\ \tau) \in [D]_\varphi$, by induction hypothesis. Finally, $(\Gamma\Gamma', (\pi'/\alpha)\pi) \in [C]_\varphi \mathring{\Rightarrow} [D]_\varphi = [B]_\varphi$.

- If $B = \forall x.C$, then $(\Gamma, \pi) \in \mathring{\forall}_T(t \mapsto \lfloor C \rfloor_{\varphi + <x,t>})$. Let $t \in \hat{T}$, then $(\Gamma, \pi t) \in [C]_{\varphi + <x,t>}$. Hence $(\Gamma\Gamma', (\pi'/\alpha)(\pi t)) = (\Gamma\Gamma', (\pi'/\alpha)\pi t) \in [C]_{\varphi + <x,t>}$, by induction hypothesis. Finally $(\Gamma\Gamma', (\pi'/\alpha)\pi) \in [B]_\varphi$.

□

We finally get the following (second) completeness result:

Proposition 3.8. *If $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ is strongly normalizing, then $Cl \circ \lfloor . \rfloor$. is a (non-empty) $\mathcal{C}'$-valued model of $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$.*

Proof. By lemma 3.1, 3.27 and propositions 3.2 (or 3.3), 3.5, 3.6, 3.7.

□

Theorem 3.1. *A theory $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle_{\equiv}$ expressed in minimal deduction modulo is strongly normalizing iff it has a non-empty $\mathcal{C}'$-valued model.*

Proof. By propositions 3.4 and 3.8.

□

3.4 Conclusion

In this chapter, we have given a sound and complete semantics, based on a refinement of Girard's reducibility candidates, for strong normalization of theories expressed in the logical framework of minimal deduction modulo. The soundness part is usual since many proofs of strong normalization of various logical frameworks use the definition of reducibility candidates we used or alternative versions. But the completeness part is innovative and there is, up to our knowledge, no other work on complete semantics for strong normalization based on the notion of reducibility candidates.

For this purpose, we have first defined a refinement of Dowek's truth values algebras called language-dependent truth values algebras. Those LDTVAs are not anymore completely independent of the studied theory, since they depend on the language (more precisely the set of sorts and function symbols of this language) on which we built minimal natural deduction. A following of this work could be to analyse how we can change this definition of LDTVA in order to obtain a totally theory-independent notion of algebra which also allows to build a sound and complete semantics for strong normalization. This does not seem obvious since we explicitely use the fact that we know the set of closed terms of each sort of the language in our definition of LDTVAs .

We have exhibited two of these language-dependent truth values algebras such that having a model valued in one of those two LDTVAs provides a sound and complete semantics for strong normalization. The first one called $\mathcal{C}_{\equiv}$ also depends on the congruence relation $\equiv$ of the theory we study. It allows us to build this LDTVA of $\equiv$-well-typed reducibility candidates. And although the proof of soundness of $\mathcal{C}_{\equiv}$-valued models does not differ from usual proofs of strong normalization using reducibility candidates, the proof of completeness of $\mathcal{C}_{\equiv}$-valued models use explicitely the fact that the interpretation of a proposition only contains proofs of this proposition (and we think we could not have obtained this property without being able to characterize $\equiv$-typing into the algebra).

The second LDTVA we have built is called $\mathcal{C}'$ and is theory-independent since it is independent of a specific congruence relation. It is also a refinement of reducibility candidates: it modifies the well-known property (CR_3) such that each proof-term we add to the interpretation of a proposition via this modified (CR_3') property, is not necessarily a proof of this proposition but it will always reduce to a proof of this proposition: each reductions sequence from such a proof-term reaches a proof of this proposition. It is quite easy to obtain this property by supposing that the neutral proof-terms usually considered in (CR_3) have, this time, to be non-normal. The second difference of this modified (CR_3') property is that we allow those neutral non-normal proof-terms not only at the top of the considered proof-term, but also at

different places in this proof-term. This allows us to build a morphism of LDTVAs from each $\mathcal{C}_{\equiv}$ of a strongly normalizing theory to $\mathcal{C}'$ and therefore prove the completeness of $\mathcal{C}'$-valued models (the proof of soundness also follows the original proof of Girard).

This technique of building models by using morphisms of LDTVAs is, up to our knowledge, also innovative. And we think that it might provide other theorems about normalization in deduction modulo and other logical frameworks. In particular, we hope to be able to prove that weak normalization implies strong normalization in minimal deduction modulo, by building a morphism from a LDTVA complete for weak normalization to a LDTVA sound for strong normalization ($\mathcal{C}_{\equiv}$ or $\mathcal{C}'$ for example). Unfortunately, we didn't succeed in managing the technical difficulties involved before the writing of this manuscript.

In the next chapter, we shall adapt this work on minimal deduction modulo in order to provide a sound and complete semantics for $\lambda\Pi$-calculus modulo, a logical framework based on the $\lambda\Pi$-calculus (the simply-typed λ-calculus with dependent types) with a congruence relation on propositions.

4

Sound and complete semantics for strong normalization in $\lambda\Pi$-calculus modulo

Context

The $\lambda\Pi$-calculus (also known as λP-calculus), is an extension of the simply-typed λ-calculus with dependent types. Different variants have been introduced in [39] (Intuitionistic Type Theory), [51] (Automath) and [31] (Logical Framework). The precise definition we shall study is that of [4]. The $\lambda\Pi$-calculus can be used as a system for interpreting proofs of minimal natural deduction, but it can also express proofs of stronger theories, by applying the idea of deduction modulo, defining the $\lambda\Pi$-calculus modulo [14]. There exists a proof of strong normalization of the $\lambda\Pi$-calculus by defining a translation from terms of $\lambda\Pi$ to terms of the simply typed λ-calculus ([31]), but this method for proving strong normalization cannot apparently be extended for theories expressed in $\lambda\Pi$-calculus modulo. There also exists proofs of strong normalization of the $\lambda\Pi$-calculus using the method of reducibility candidates ([25]). We can therefore wonder if we can adapt our notion of sound and complete reducibility candidates for theories expressed in the $\lambda\Pi$-calculus modulo.

Contributions

We exhibit a notion of pre-models for $\lambda\Pi$-calculus modulo, which provide a sound and complete semantics for strong normalization of theories expressed in the $\lambda\Pi$-calculus modulo. For that purpose, we adapt the method used for minimal deduction modulo and use the notion of sound and complete reducibility candidates we have defined in section 3.3. The first difference with the work done in chapter 3 comes from the presence of dependent types.

We refine the usual method for building interpretations of dependent types by using valuations, in order to use the method for proving completeness of those reducibility candidates shown in sections 3.2.4 and 3.3.4. The second main difference is that in λΠ-calculus modulo, β-reduction and rewrite rules can be applied on a same kind of term (whereas in deduction modulo, β-reduction is only applied on proof-terms, and rewrite rules only on propositions). For example, we shall see that the theory defined by a single rewrite rule $P \to P$, with P an atomic proposition, is strongly normalizing in deduction modulo but not in λΠ-calculus modulo.
This work can be seen as a first step in finding the notion of algebra, on which we should build models for theories expressed in the λΠ-calculus modulo, as it has been done for deduction modulo.

Outline

We first introduce the λΠ-calculus and define its syntax and typing rules. We then introduce the λΠ-calculus modulo which is the application of the ideas of deduction modulo to the λΠ-calculus, and prove some properties of theories expressed in the λΠ-calculus modulo, like subject-reduction, uniqueness of types (modulo the congruence relation) and weakening. We finally define the notion of pre-model for theories expressed in the λΠ-calculus modulo and prove that it provides a sound and complete semantics for strong normalization of those theories.

4.1 The λΠ-calculus

The λΠ-*calculus* is a dependently typed lambda-calculus that permits to construct types depending on terms, for instance a type $array\ n$, of arrays of size n, that depends on a term n of type nat. It also permits to construct a function f taking a natural number n as an argument and returning an array of size n. Thus, the arrow type $nat \Rightarrow array$ of simply typed lambda-calculus must be extended to a dependent product type $\Pi x : nat.\ (array\ x)$ where, in the expression $\Pi x : A.\ B$, the occurrences of the variable x are bound in B by the symbol Π (the expression $A \Rightarrow B$ is used as a shorter notation for the expression $\Pi x : A.\ B$ when x has no free occurrence in B). When we apply the function f to a term n, we do not get a term of type $array\ x$ but of type $array\ n$. Thus, the application rule must include a substitution of the term n for the variable x. The symbol $array$ itself takes a natural number as an argument and returns a type. Thus, its type is $nat \Rightarrow Type$, *i.e.* $\Pi x : nat.\ Type$. The terms $Type$, $nat \Rightarrow Type$, ... cannot have type $Type$, because Girard's paradox [27] could then be expressed in the system, thus we introduce a new symbol $Kind$ to type such terms. To form terms, like $\Pi x : nat.\ Type$, whose type is $Kind$, we need a rule expressing that

the symbol $Type$ has type $Kind$ and a new product rule allowing to form the type $\Pi x : nat.\ Type$, whose type is $Kind$. Besides the variables such as x whose type has type $Type$, we must permit the declaration of variables such as nat of type $Type$, and more generally, variables such as $array$ whose type has type $Kind$. This leads to introduce the following syntax and typing rules.

4.1.1 Syntax of the λΠ-calculus

Notice that, unlike in deduction modulo, we use a single syntax to represent proof-terms and propositions (terms and types) in λΠ-calculus, as a type (proposition) can depend on a term (proof-term) as explained above.

Definition 4.1 (The syntax of λΠ)**.** *The syntax of the λΠ-calculus is*

$$t = x \mid Type \mid Kind \mid \Pi x : t.\ t \mid \lambda x : t.\ t \mid t\ t$$

Notice that in the constructions $\Pi x : t_1.\ t_2$ and $\lambda x : t_1.\ t_2$, the variable x is bound in the term t_2.

Then we redefine free variables and substitution for λΠ-calculus.

Definition 4.2 (Free variables)**.**
We call free variables *of a term t those occurrences of variables which appear in t and are not bound in t. We define the set $FV(t)$ of free variables of a term t by induction on the structure of t:*

- $FV(x) = \{x\}$
- $FV(Type) = FV(Kind) = \emptyset$
- $FV(\Pi x : t_1.\ t_2) = FV(t_1) \cup (FV(t_2) - \{x\})$
- $FV(\lambda x : t_1.\ t_2) = FV(t_1) \cup (FV(t_2) - \{x\})$
- $FV(t_1\ t_2) = FV(t_1) \cup FV(t_2)$

We say that a term is closed *if it does not contain free variables.*

Definition 4.3 (Substitution)**.**
We define the substitution $(t/x)t'$ of a variable x by a term t (such that x is not free in t) in a term t' by induction of the structure of t' as:

- $(t/x)y = t$ *if* $x = y$ *and* y *otherwise*
- $(t/x)Type = Type$
- $(t/x)Kind = Kind$

- $(t/x)(\Pi y : t_1.t_2) = \Pi y : (t/x)t_1.\ (t/x)t_2$
  Notice that we suppose here that $x \neq y$ *and* $y \notin FV(t_1) \cup FV(t_2)$
  (we have to rename y *otherwise).*
- $(t/x)(\lambda y : t_1.t_2) = \lambda y : (t/x)t_1.\ (t/x)t_2$
  Notice that we suppose here that $x \neq y$ *and* $y \notin FV(t_1) \cup FV(t_2)$
  (we have to rename y *otherwise).*
- $(t/x)(t_1\ t_2) = (t/x)t_1\ (t/x)t_2$

Remark 4.1. *As usual, if a variable* x *is not free in a term* t' *then for all terms* t*, we have* $(t/x)t' = t'$*.*

Definition 4.4 (β-reduction)**.**
A β*-redex is a proof-term of the form* $(\lambda x.t)\ t'$*.*
The β*-reduction is the relation defined by the following rule:*

$$(\lambda x.\pi)t \to (t/x)\pi$$

and the contextual closure:

$$\begin{array}{c} \textit{if } t_1 \to t_1' \textit{ then } \Pi x : t_1.t_2 \to \Pi x : t_1'.t_2, \\ \textit{if } t_2 \to t_2' \textit{ then } \Pi x : t_1.t_2 \to \Pi x : t_1.t_2', \\ \textit{if } t_1 \to t_1' \textit{ then } \lambda x : t_1.t_2 \to \lambda x : t_1'.t_2, \\ \textit{if } t_2 \to t_2' \textit{ then } \lambda x : t_1.t_2 \to \lambda x : t_1.t_2', \end{array}$$

We write $t \to^+ t'$ *if* t β*-reduces to* t' *in one or more reduction steps.*
We write $t \to^* t'$ *if* t β*-reduces to* t' *in an arbitrary number of reduction steps.*
We write $\equiv_\beta$ *for the congruence relation induced by the* β*-reduction.*

4.1.2 Typing rules of the λΠ-calculus

A particularity of the λΠ-calculus is that the formation of contexts is also defined by inference rules. Since we have a single syntax for terms and types, we have to specify which of these terms can considered as types. As we shall see in the following, terms which can be considered as types are $Kind$ and all terms typed by $Type$ or $Kind$ (see rules Declaration, Declaration2 and Sort).

Another particularity about contexts in λΠ-calculus is that, unlike in natural deduction, the order of the declarations does matter: for example, a type in a context can depend on a variable which has been previously declared in this context. Hence concatenation of contexts can be more complicated to define than in natural deduction (we shall avoid using those concatenations in the present work).

In the following, we detail the typing rules of a part of the λΠ-calculus called the $\lambda\Pi^-$-calculus (notice that we write here [ ] for the empty context).

Definition 4.5 (The typing rules of $\lambda\Pi^-$)**.**

$$\frac{}{[\,]\ \textit{well-formed}}\ \textbf{Empty}$$

$$\frac{\Gamma \vdash A : Type}{\Gamma[x : A]\quad \textit{well-formed}}\ \textbf{Declaration}\ x\ \textit{not in}\ \Gamma$$

$$\frac{\Gamma \vdash A : Kind}{\Gamma[x : A]\quad \textit{well-formed}}\ \textbf{Declaration2}\ x\ \textit{not in}\ \Gamma$$

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma \vdash Type : Kind}\ \textbf{Sort}$$

$$\frac{\Gamma\ \textit{well-formed}\quad x : A \in \Gamma}{\Gamma \vdash x : A}\ \textbf{Variable}$$

$$\frac{\Gamma \vdash A : Type\quad \Gamma[x : A] \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}\ \textbf{Product}$$

$$\frac{\Gamma \vdash A : Type\quad \Gamma[x : A] \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}\ \textbf{Product2}$$

$$\frac{\Gamma \vdash A : Type\quad \Gamma[x : A] \vdash B : Type\quad \Gamma[x : A] \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction}$$

$$\frac{\Gamma \vdash t : \Pi x : A\ B\quad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : (u/x)B}\ \textbf{Application}$$

It is useful, in some situations, to add a rule allowing to build type families by abstraction, for instance $\lambda x : nat\ (array\ (2 \times x))$ and rules asserting that a term of type $(\lambda x : nat\ (array\ (2 \times x))\ n)$ also has type $array\ (2 \times n)$. This leads to introduce the following extra typing rules.

Definition 4.6 (The typing rules of λΠ)**.**
The typing rules of λΠ *are those of* $\lambda\Pi^-$ *plus the following ones:*

$$\frac{\Gamma \vdash A : Type\quad \Gamma[x : A] \vdash B : Kind\quad \Gamma[x : A] \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction2}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}\ \textbf{Conversion}\ A \equiv_\beta B$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma \vdash B : Kind \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}\ \textbf{Conversion2}\ A \equiv_\beta B$$

where $\equiv_\beta$ is the β-equivalence relation.

As we have said previously, the order in which variables are declared does matter in λΠ-calculus modulo, unlike in deduction modulo. Hence we have to redefine sub-contexts in this logical framework, as initial sublists of a context. Notice that we shall only consider well-formed contexts in the following of this chapter.

Definition 4.7 ($\subseteq$)**.**
For all (well-formed) contexts Γ, Γ', Γ is said to be a sub-context *of Γ' if and only if there exists $n \in \mathbb{N}$, variables $x_1, \ldots, x_n$ and terms $A_1, \ldots, A_n$ such that $\Gamma' = \Gamma[x_1 : A_1] \ldots [x_n : A_n]$. We write $\Gamma \subseteq \Gamma'$.*

It can be proved that types are preserved by β-reduction, that β-reduction is confluent and strongly normalizing and that each term has a unique type modulo β-equivalence. We shall not detail those proofs in the present work but will focus on the proofs of subject-reduction and uniqueness of types for λΠ-calculus modulo detailed in section 4.2 (the proofs for λΠ-calculus modulo can easily be adapted for λΠ-calculus).

The λΠ-calculus, and even the λΠ$^-$-calculus, can be used to express proofs of minimal natural deduction, following the Brouwer-Heyting-Kolmogorov interpretation and the Curry-de Bruijn-Howard correspondence. Let $\langle \mathbb{T}, \mathbb{F}, \mathbb{P} \rangle$ be a language, we consider a context Γ formed with variables $\iota_1, ..., \iota_n$ of type $Type$ (corresponding to the differents sorts of $\mathbb{T}$, for each function symbol $f \in \mathbb{F}$, a variable f of type $\iota \Rightarrow ... \Rightarrow \iota \Rightarrow \iota$ and for each predicate symbol $P \in \mathbb{P}$, a variable P of type $\iota \Rightarrow ... \Rightarrow \iota \Rightarrow Type$.

To each formula P containing free variables $x_1, ..., x_p$ we associate a term P° of type $Type$ in the context $\Gamma, x_1 : \iota, ..., x_p : \iota$ translating each variable, function symbol and predicate symbol by itself and the implication symbol and the universal quantifier by a product.

To each proof π, in minimal natural deduction, of a sequent $A_1, ..., A_n \vdash B$ with free variables $x_1, ..., x_p$, we can associate a term π° of type B° in the context $\Gamma[x_1 : \iota] \ldots [x_p : \iota][\alpha_1 : A_1^\circ] \ldots [\alpha_n : A_n^\circ]$. From the strong normalization of the λΠ-calculus, we get cut elimination for minimal natural deduction and therefore its consistency.

4.2 The λΠ-calculus modulo

The λΠ-calculus modulo is the adaptation of the principle of deduction modulo to λΠ-calculus. As we shall see in example 4.1 (and also in chapter 5), we can express, via the Brouwer-Heyting-Kolmogorov interpretation and the Curry-de Bruijn-Howard correspondence, stronger theories than minimal natural deduction by adding typing rules to λΠ-calculus (as we can extend theories expressed in minimal natural deduction by adding axioms as inference rules). But another solution is to add rewrite rules on types, as done in deduction modulo, this leads to the definition of λΠ-calculus modulo. By the way, we shall see that the former method can be simulated by the latter one in chapter 5.

4.2.1 Syntax of the λΠ-calculus modulo

If Σ, Γ_1 and Γ_2 are contexts, a substitution θ, binding the variables declared in Γ_1, is said to be *of type* $\Gamma_1 \rightsquigarrow \Gamma_2$ *in* Σ if for all x declared of type T in Γ_1, we have $\Sigma\Gamma_2 \vdash \theta x : \theta T$, and that, in this case, if $\Sigma\Gamma_1 \vdash u : U$, then $\Sigma\Gamma_2 \vdash \theta u : \theta U$.

A *rewrite rule* is a quadruple $l \longrightarrow^{\Gamma_1, T} r$ where Γ_1 is a context and l, r and T are β-normal terms. Such a rule is said to be *well-typed* in the context Σ if, in the λΠ-calculus, the context $\Sigma\Gamma_1$ is well-formed and the terms l and r have type T in this context.

If Σ is a context, $l \longrightarrow^{\Gamma_1, T} r$ is a rewrite rule well-typed in Σ and θ is a substitution of type $\Gamma_1 \rightsquigarrow \Gamma_2$ in Σ then the terms θl and θr both have type θT in the context $\Sigma\Gamma_2$. We say that the term θl *rewrites* to the term θr.

If Σ is a context and $\mathcal{R}$ a set of rewrite rules well-typed in the λΠ-calculus in Σ, then the *congruence generated by* $\mathcal{R}$, $\equiv_{\mathcal{R}}$, is the smallest congruence relation such that if t rewrites to u then $t \equiv_{\mathcal{R}} u$.

Definition 4.8 (Theory)**.**
A theory *expressed in λΠ-calculus is given by a context and a set of set of well-typed rewrite rules on terms.*

Given such a context Σ (which we shall call signature) and such a set of rewrite rules $\mathcal{R}$, we define the λΠ-calculus on Σ modulo $\mathcal{R}$, as follows:

Definition 4.9 (The syntax of λΠ modulo)**.**
Given a signature Σ *and a rewrite system* $\mathcal{R}$*, the syntax of the λΠ-calculus on* Σ *modulo* $\mathcal{R}$ *is the same as the one of λΠ-calculus:*

$$t = \ x \mid Type \mid Kind \mid \Pi x : t.\ t \mid \lambda x : t.\ t \mid t\ t$$

Notice that variables declared in Σ *have not the same behaviour as other variables since they cannot be bound or substituted.*
We write Λ *the set of terms of λΠ-calculus on* Σ *modulo* $\mathcal{R}$*.*

Definition 4.10 (Neutral terms)**.**
We call neutral those proof-terms of Λ *that are not abstractions* i.e. *that are terms of the form* x*,* $Type$*,* $Kind$*,* $\Pi x : t.t'$ *or* tt' *are the only neutral proof-terms.*

Then we define the notions of *isolated* terms and *leaves* of a term, as we have done for minimal deduction modulo.

Definition 4.11 (Isolated proof-terms)**.**
A term is called isolated *if it is neutral and only* $\beta\mathcal{R}$*-reduces to neutral proof-terms.*

Definition 4.12 (Leaves)**.**
The leaves *of a term* t *are its first* $\beta\mathcal{R}$*-reducts which are normal or not neutral. We call* $\mathcal{L}(t)$ *the set of leaves of* t*.*

Definition 4.13 (Normalization)**.**

- *We say that a term is* normal *if it does not contain neither a* β*-redex nor a* $\mathcal{R}$*-redex.*
- *We say that a proof-term is* weakly normalizing *if there exists a* $\beta\mathcal{R}$*-reductions sequence from it which reaches a normal term (hence is finite).*
- *We write* WN *for the set of weakly normalizing terms.*
- *We say that a term is* strongly normalizing *if all* $\beta\mathcal{R}$*-reductions sequences from it reach a normal proof-term.*
- *We write* SN *for the set of strongly normalizing proof-terms.*
- *We say that a theory expressed in* λΠ*-calculus modulo is* weakly normalizing *if all well-typed proof-terms are weakly normalizing.*
- *We say that a theory expressed in* λΠ*-calculus modulo is* strongly normalizing *if all well-typed proof-terms are strongly normalizing.*

4.2.2 Typing rules of the λΠ-calculus modulo

As for deduction modulo, the main particularity of λΠ-calculus modulo is that we can replace a type by an $\equiv_{\mathcal{R}}$-equivalent one in a typing judgement. As we were yet able to replace β-equivalent types in λΠ-calculus, we shall consider $\equiv_{\beta\mathcal{R}}$-equivalence here, as you can see in the following rules Conversion and Conversion2.

Definition 4.14 (The typing rules of λΠ modulo)**.**

$$\frac{}{[\,]\ \textit{well-formed}}\ \textbf{Empty}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Type}{\Sigma\Gamma[x : A]\ \ \textit{well-formed}}\ \textbf{Declaration}\ x\ \textit{not in}\ \Sigma\Gamma$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Kind}{\Sigma\Gamma[x : A]\ \ \textit{well-formed}}\ \textbf{Declaration2}\ x\ \textit{not in}\ \Sigma\Gamma$$

$$\frac{\Gamma\ \textit{well-formed}}{\Sigma\Gamma \vdash_{\mathcal{R}} Type : Kind}\ \textbf{Sort}$$

$$\frac{\Gamma\ \textit{well-formed}\ \ x : A \in \Sigma\Gamma}{\Sigma\Gamma \vdash_{\mathcal{R}} x : A}\ \textbf{Variable}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Type\ \ \Sigma\Gamma[x : A] \vdash_{\mathcal{R}} B : Type}{\Sigma\Gamma \vdash_{\mathcal{R}} \Pi x : A\ B : Type}\ \textbf{Product}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Type\ \ \Sigma\Gamma[x : A] \vdash_{\mathcal{R}} B : Kind}{\Sigma\Gamma \vdash_{\mathcal{R}} \Pi x : A\ B : Kind}\ \textbf{Product2}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Type\ \ \Sigma\Gamma[x : A] \vdash_{\mathcal{R}} B : Type\ \ \Sigma\Gamma[x : A] \vdash_{\mathcal{R}} t : B}{\Sigma\Gamma \vdash_{\mathcal{R}} \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Type\ \ \Sigma\Gamma[x : A] \vdash_{\mathcal{R}} B : Kind\ \ \Sigma\Gamma[x : A] \vdash_{\mathcal{R}} t : B}{\Sigma\Gamma \vdash_{\mathcal{R}} \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction2}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} t : \Pi x : A\ B\ \ \Sigma\Gamma \vdash_{\mathcal{R}} u : A}{\Sigma\Gamma \vdash_{\mathcal{R}} (t\ u) : (u/x)B}\ \textbf{Application}$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Type\ \ \Sigma\Gamma \vdash_{\mathcal{R}} B : Type\ \ \Sigma\Gamma \vdash_{\mathcal{R}} t : A}{\Sigma\Gamma \vdash_{\mathcal{R}} t : B}\ \textbf{Conversion}\ A \equiv_{\beta\mathcal{R}} B$$

$$\frac{\Sigma\Gamma \vdash_{\mathcal{R}} A : Kind\ \ \Sigma\Gamma \vdash_{\mathcal{R}} B : Kind\ \ \Sigma\Gamma \vdash_{\mathcal{R}} t : A}{\Sigma\Gamma \vdash_{\mathcal{R}} t : B}\ \textbf{Conversion2}\ A \equiv_{\beta\mathcal{R}} B$$

Where $\equiv_{\beta\mathcal{R}}$ is the congruence of terms generated by the rules of $\mathcal{R}$ and the rule β.

Example 4.1. *Consider the signature* $\Sigma_0 = [P : Type, Q : Type]$*, then* $\Sigma_0\Gamma \vdash P : Type$ *and* $\Sigma_0\Gamma \vdash \Pi x : Q.Q : Type$*. We can therefore define the single rewrite rule* $\mathcal{R}_0 = P \longrightarrow^{[],Type} \Pi x : Q.Q$*. In the* λΠ *calculus on* Σ_0 *modulo* $\mathcal{R}_0$*, we have* $\Sigma_0[y : Q] \vdash_{\mathcal{R}_0} (\lambda x : P.x)y : Q$*.*

4.2.3 Technical lemmas

Let us first detail some useful technical lemmas about theories expressed in λΠ-calculus modulo. Let us prove a property of λΠ-calculus modulo called *weakening*: the fact that if a term is well-typed in a context then it is also well-typed by the same type in all well-formed contexts which contain the first one.

Lemma 4.1 (Weakening)**.**
For all (well-formed) contexts Γ, Γ' *and* $t, A \in \Lambda$*,*
if $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$ *and* $\Gamma \subseteq \Gamma'$*, then* $\Sigma\Gamma' \vdash_{\mathcal{R}} t : A$*.*

Proof. By induction on the length of the derivation $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$. By case on the last rule used in this derivation. If the last rule is:

- **Sort**: we have $t = Type$ and $A = Kind$, and $Type$ is of type $Kind$ in any well-formed context.

- **Variable**: then t is a variable x such that either $x : A \in \Sigma$ or $x : A \in \Gamma$. Therefore $x : A \in \Sigma$ or $x : A \in \Gamma'$ hence $\Sigma\Gamma' \vdash_{\mathcal{R}} x : A$

- **Product**: then t is a product $\Pi x : B.C$ and A is $Type$ (resp. $Kind$). Notice that we can choose x such that x is not declared in Γ'. By induction hypothesis, $\Sigma\Gamma' \vdash_{\mathcal{R}} B : Type$ and $\Sigma\Gamma'[x : B] \vdash_{\mathcal{R}} C : Type$ (resp. $Kind$). Notice that since x is not declared in Γ', $\Gamma'[x : B]$ is well-formed and we may do not have $\Gamma[x : B] \subseteq \Gamma'[x : B]$, but $\Gamma'[x : B]$ is equivalent to a sup-context of $\Gamma[x : B]$ in the sense that it defines exactly the same set of ordered pairs of two terms such that the former one is typed by the latter one in this context. Finally, we have $\Sigma\Gamma' \vdash_{\mathcal{R}} \Pi x : B.C : Type$ (resp. $Kind$).

- **Abstraction**: then $t = \lambda x : B.u$ and $A = \Pi x : B.C$. As previously we can choose x such that x is not declared in Γ', in this case $\Gamma'[x : B]$ is equivalent to a sup-context of $\Gamma[x : B]$. Hence, by induction hypothesis, we have $\Sigma\Gamma' \vdash_{\mathcal{R}} B : Type$, $\Sigma\Gamma'[x : B] \vdash_{\mathcal{R}} C : Type$ (resp. $Kind$) and $\Sigma\Gamma'[x : B] \vdash_{\mathcal{R}} u : C$. Hence $\Sigma\Gamma' \vdash_{\mathcal{R}} \lambda x : B.u : \Pi x : B.C$.

- **Application**: then $t = uv$ and $A = (v/x)C$, with $\Sigma\Gamma' \vdash_{\mathcal{R}} u : \Pi x : B.C$ and $\Sigma\Gamma' \vdash_{\mathcal{R}} v : B$, by induction hypothesis, hence $\Sigma\Gamma' \vdash_{\mathcal{R}} uv : (v/x)C$.

- **Conversion**: by induction hypothesis.

□

The next lemma is about well-typed substitution in typing judgements. The main difference with what we have seen in chapter 3 for minimal deduction modulo is that the substitution applies also on contexts besides applying on terms and types (but not on the signature Σ since its variables cannot be substituted).

Lemma 4.2 (Substitution in typing judgements)**.**
For all variables x, terms $t, t', A, B \in \Lambda$ and well-formed contexts $\Gamma_1[x : B]\Gamma_2$,
If $\Sigma\Gamma_1[x : B]\Gamma_2 \vdash_{\mathcal{R}} t : A$ and $\Sigma\Gamma_1 \vdash_{\mathcal{R}} t' : B$ then $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t : (t'/x)A$.

Proof. Notice, that in order to prove that $\Gamma_1(t'/x)\Gamma_2$ is a well-formed context, we need to use this lemma on types of which variables are declared in Γ_2. We therefore have to reason by induction on the length of Γ_2. We present, in the following, the reasoning which can be used in both atomic and induction cases of this induction. We reason by induction on the length of the derivation of $\Sigma\Gamma_1[x : B]\Gamma_2 \vdash_{\mathcal{R}} t : A$. By case on the last rule used in this derivation. If the last rule is:

- **Sort**: we have $t = (t'/x)t = Type$ and $A = (t'/x)A = Kind$, hence $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t : (t'/x)A$.
- **Variable**: then t is a variable y.
  - If $x = y$, then $A = B$ and $x \notin FV(B) = FV(A)$ since $\Gamma_1[x : B]$ is well-formed. Moreover $\Sigma\Gamma_1 \vdash_{\mathcal{R}} (t'/x)t = t' : B = (t'/x)B = (t'/x)A$. Therefore $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t : (t'/x)A$ by lemma 4.1.
  - If $x \neq y$,
    - if $x \in FV(A)$, then the declaration $y : A$ appears in Γ_2 (since $\Gamma_1[x : B]\Gamma_2$ is well-formed, types containing free occurrences of x can only appear in Γ_2). Hence $y : (t'/x)A$ appears in $(t'/x)\Gamma_2$. Therefore $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t = y : (t'/x)A$.
    - if $x \notin FV(A)$, then $(t'/x)A = A$ and $y : A$ appears in $\Gamma_1(t'/x)\Gamma_2$ therefore $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t = y : A = (t'/x)A$.
- **Product**: then t is a product $\Pi y : C.D$ and A is $Type$ (resp. $Kind$). By induction hypothesis, we have $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)C : Type$ and $\Sigma\Gamma_1(t'/x)\Gamma_2[y : (t'/x)C] \vdash_{\mathcal{R}} (t'/x)D : Type$ (resp. $Kind$). Hence $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t = \Pi y : (t'/x)C.(t'/x)D : Type$ (resp. $Kind$).

- **Abstraction**: then $t = \lambda y : C.u$ and $A = \Pi x : C.D$.
  By induction hypothesis, we have $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)C : Type$, $\Sigma\Gamma_1(t'/x)\Gamma_2[y : (t'/x)C] \vdash_{\mathcal{R}} (t'/x)D : Type$ (resp. $Kind$) and $\Sigma\Gamma_1(t'/x)\Gamma_2[y : (t'/x)C] \vdash_{\mathcal{R}} (t'/x)u : (t'/x)D$. Hence we have $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)t = \lambda y : (t'/x)C.(t'/x)u : (t'/x)(\Pi y : C.D)$.

- **Application**: then $t = uv$ and $A = (v/y)D$, with $\Sigma\Gamma_1[x : B]\Gamma_2 \vdash_{\mathcal{R}} u : \Pi y : C.D$ and $\Sigma\Gamma_1[x : B]\Gamma_2 \vdash_{\mathcal{R}} v : B$. By induction hypothesis, we have $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)u : \Pi y : (t'/x)C.(t'/x)D$ and also $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)v : (t'/x)C$.
  Hence $\Sigma\Gamma_1(t'/x)\Gamma_2 \vdash_{\mathcal{R}} (t'/x)(uv) : ((t'/x)v/y)((t'/x)D)$.

- **Conversion**: If $A \equiv_{\beta\mathcal{R}} C$, then $(t'/x)A \equiv_{\beta\mathcal{R}} (t'/x)C$, because (t'/x) is a substitution of type $\Gamma_1[x : B]\Gamma_2 \rightsquigarrow \Gamma_1(t'/x)\Gamma_2$ in Σ.

□

The following lemma states that each well-typed proof-term in λΠ-calculus modulo on Σ modulo $\mathcal{R}$ has a unique type modulo $\beta\mathcal{R}$-equivalence.

Lemma 4.3 (Uniqueness of types)**.**
For all well-formed contexts Γ *and* $t, A, B \in \Lambda$,
if $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$ *and* $\Sigma\Gamma \vdash_{\mathcal{R}} t : B$, *then* $A \equiv_{\beta\mathcal{R}} B$.

Proof. By induction on the structure of t. For each case, the last rule used in the derivation of $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$ can only be the rule which corresponds to the syntactic form of t or the rule Conversion (resp. Conversion2).

□

The following lemma states that types are preserved under $\beta\mathcal{R}$-reduction in λΠ-calculus modulo on Σ modulo $\mathcal{R}$.

Lemma 4.4 (Subject-reduction)**.**
For all $t, t', A \in \Lambda_\Sigma$ *and well-formed contexts* Γ,
if $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$ *and* $t \rightarrow_{\beta\mathcal{R}} t'$ *then* $\Sigma\Gamma \vdash_{\mathcal{R}} t' : A$

Proof.
- If $t \rightarrow_{\mathcal{R}} t'$, by lemma 4.3 and the fact that rewrite rules in $\mathcal{R}$ are well-typed.

- If $t \rightarrow_\beta t'$, by induction on the position of the β-redex reduced in $t \rightarrow_\beta t'$. If this redex is not at the root of t, we conclude by induction hypothesis. Otherwise $t = (\lambda x : B.u)v$ and $t' = (v/x)u$ with $\Sigma\Gamma \vdash_{\mathcal{R}} \lambda x : B.u : \Pi x : B.C$, $\Sigma\Gamma[x : B] \vdash_{\mathcal{R}} v : B$ and $A \equiv_{\beta\mathcal{R}} (v/x)C$ by lemma 4.3. Hence $\Sigma\Gamma[x : B] \vdash_{\mathcal{R}} u : C$ therefore $\Sigma(v/x)\Gamma \vdash_{\mathcal{R}} (v/x)u : (v/x)C$ by lemma 4.2. We conclude by the fact that x is not declared in Γ and therefore cannot be free in Γ, hence $(v/x)\Gamma = \Gamma$.

□

Finally we define formally what we have been calling *types* since the beginning of this chapter. Given a signature Σ, a set of rewrite rules $\mathcal{R}$, and a specific context Γ, we discriminate the elements of Λ which can be used as types of variables in Γ, as the term $Kind$ and those which are of type $Type$ or $Kind$ in Γ.

Definition 4.15 ($\mathcal{I}_\Gamma$)**.** *Given a signature Σ and a set of rewrite rules $\mathcal{R}$,*

$$\mathcal{I}_{\Sigma,\mathcal{R},\Gamma} = \{Kind\} \cup \{A \text{ s.t. } \Sigma\Gamma \vdash_{\mathcal{R}} A : Type \text{ or } \Sigma\Gamma \vdash_{\mathcal{R}} A : Kind\}$$

We shall write $\mathcal{I}_\Gamma$ for $\mathcal{I}_{\Sigma,\mathcal{R},\Gamma}$ when there is no ambiguity.

We can notice that the terms A which are of type $Type$ or $Kind$ in a context Γ are the ones which can be used to extend Γ into a context $\Gamma[x : A]$, via the rules Declaration and Declaration2, hence the ones which can be considered as types in Γ. We verify, in the following lemma, that $\mathcal{I}_\Gamma$ contains all inhabited types in Γ.

Lemma 4.5. *For all $A \in \Lambda$, $t \in \Lambda$ and (well-formed) contexts Γ,*
if $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$, then $A \in \mathcal{I}_\Gamma$.

Proof. If $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$, we reason by case on the last rule used in the derivation of $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$. If the last rule is:

- **Sort**: we have $t = Type$ and $A = Kind$.
- **Variable**: then t is a variable x and we have $x : A \in \Gamma$. Therefore $\Sigma\Gamma \vdash_{\mathcal{R}} A : Type$ or $\Sigma\Gamma \vdash_{\mathcal{R}} A : Kind$ (rules Declaration and Declaration2).
- **Product**: then t is a product $\Pi y : C.D$ and A is $Type$ or $Kind$.
- **Abstraction**: then $t = \lambda y : C.u$ and $A = \Pi x : C.D$. Therefore $\Sigma\Gamma \vdash_{\mathcal{R}} A : Type$ or $\Sigma\Gamma \vdash_{\mathcal{R}} A : Kind$ (rules Product and Product2).
- **Application**: then $t = uv$ and $A = (v/y)D$, with $\Sigma\Gamma \vdash_{\mathcal{R}} u : \Pi y : C.D$ and $\Sigma\Gamma \vdash_{\mathcal{R}} v : B$. By rules Product or Product2, we have $\Sigma\Gamma[y : C] \vdash_{\mathcal{R}} D : Type$ or $\Sigma\Gamma[y : C] \vdash_{\mathcal{R}} D : Kind$ hence, by lemma 4.2, we have $\Sigma\Gamma \vdash_{\mathcal{R}} (v/y)D : Type$ or $\Sigma\Gamma \vdash_{\mathcal{R}} (v/y)D : Kind$.
- **Conversion**: This condition is necessary to use the rules Conversion and Conversion2.

□

Notice that the converse is not true: if $\Sigma\Gamma \vdash_{\mathcal{R}} A : Type$ (resp. $Kind$), we only know that for all variables x not declared in Γ, A is inhabited in $\Gamma[x : A]$ (which is well-formed), but it may be not in Γ.

4.3 Pre-models for λΠ-modulo

In this section, we provide a definition of pre-models for λΠ-calculus modulo. As it has been for deduction modulo, this work can be seen as a first step in the search of a definition of algebras for λΠ-calculus modulo on which we could build models sound (and complete) for consistency and for strong normalization. This definition of algebras may emerge from our definition of LDTVAs of section 3.1.4, because as we shall see the interpretation of products we propose can be seen as a mixing of interpretations of the implication and the universal quantifier of LDTVAs. It will be studied in future work.

4.3.1 Definition of pre-models

Let us first redefine the notion of substitution with capture and the definition of properties (CR_1), (CR_2) and (CR_3') we have already seen in section 3.3.2.

Definition 4.16. *We define $[u/x]t$ the substitution with capture of x by u (such that $x \notin FV(u)$) in t by induction on the structure of t as follows:*

- $[u/x]y = u$ *if* $x = y$ *and* y *otherwise*
  (notice that x cannot be a variable declared in the signature Σ).
- $[u/x]Type = Type$
- $[u/x]Kind = Kind$
- $[u/x](\lambda y : t_1.\ t_2) = \lambda y : [u/x]t_1.\ [u/x]t_2$
- $[u/x](\Pi y : t_1.\ t_2) = \Pi y : [u/x]t_1.\ [u/x]t_2$
- $[u/x](t_1\ t_2) = [u/x]t_1\ [u/x]t_2$

Definition 4.17.
For all $E \subseteq \Lambda$, we define the following properties :

(CR_1) *For all $t \in E$, $t \in SN$*

(CR_2) *For all $t \in E$, and t' such that $t \rightarrow_{\beta\mathcal{R}} t'$, $t' \in E$*

(CR_3') *For all $n \in \mathbb{N}$, for all $v, u_1, \ldots, u_n \in \Lambda$, if*
- for all $i \leq n$, u_i is neutral and non-normal,
- $\forall r_1, \ldots, r_n$ such that for all $i \leq n$, $u_i \rightarrow_{\beta\mathcal{R}} r_i$, $[r_i/x_i]_{i\leq n}v \in E$
then $[u_i/x_i]_{i\leq n}v \in E$.

Notice that the real difference with the properties defined in section 3.3.2 is that we consider now $\beta\mathcal{R}$-reduction in (CR_1), (CR_2) and (CR_3'), and not only β-reduction anymore.

Let us now think about how to define interpretations and (pre-)models for λΠ-calculus modulo. First, we want to interpret types (propositions, as in deduction modulo) as sets of terms. Since we do not have distinguished syntaxes for terms and types, we shall interpret terms which can be considered as types in a specific (well-formed) context Γ and we shall use, for that purpose, the set $\mathcal{I}_\Gamma$ we have defined in section 4.2.
Now what about the property we want interpretations of products to satisfy ? As we have seen for deduction modulo, in order to prove completeness of our definition of pre-models, we want interpretations of a connective to be modeled on the elimination rule of this connective (notice that there exists also works on normalization in which we use the introduction rule of the connective, as in [20]). For example, we want the interpretation of $A \Rightarrow B$ to be equal to the set of (proof-)terms which map elements in the interpretation of A to elements in the interpretation of B. Following this method, we would want the interpretation of a product $\Pi x : A.B$ to be equal to the set of terms which map elements u of the interpretation of A to elements of the interpretation of $(u/x)B$, using a kind of dependent intersection, as we have done for the interpretation of the universal quantifier in LDTVAs of section 3.1.4. The problem is that, since we consider interpretations of types which can contain ill-typed terms, the term $(u/x)B$ may not be a type (i.e. in a specific $\mathcal{I}_\Gamma$). The usual solution to this problem is to use a notion of valuation (here functions which map variables to terms). The property about interpretations of products becomes: the interpretation of a product $\Pi x : A.B$ and a valuation φ is equal to the set of terms which map elements u of the interpretation of A and φ to elements of the interpretation of B and $\varphi + \langle x, u\rangle$. This solution has been used (in slightly different forms) in [26] or [41]. The main difference is that, we consider here a kind of dependent intersection, whereas in [26], the interpretation of the product $\Pi x : A.B$ and a valuation φ is defined as the intersection of all the interpretations of B in $\varphi + \langle x, u\rangle$, for u in the interpretation of A and φ. This way, we adapt to products of λΠ the work done in section 3.1.3 for the interpretation of the universal quantifier in LDTVAs , by refining the considered intersection as a dependent intersection.

Let us formally define the notions of valuation and interpretations.

Definition 4.18 (Valuations)**.**
For all well-formed contexts Γ*, a* valuation *on* Γ *is a substitution which maps none to all the variables declared in* Γ *to terms in* Λ*. We shall call them* total *valuations on* Γ *if they map all the variables in* Γ*.*

Remark 4.2. *With this definition, for all (well-formed) contexts* Γ, Γ' *such that* $\Gamma \subseteq \Gamma'$*, a valuation on* Γ *is also a valuation on* Γ'*. And we can also notice that the only valuation on the empty context is the empty substitution.*

Definition 4.19 (Interpretations)**.**
An interpretation *is a collection* $[\![.]\!]_{\cdot}$ *such that for all well-formed contexts* Γ*,* $[\![.]\!]_{\cdot}^{\Gamma}$ *is a function which maps ordered pairs of a term* A *in* $\mathcal{I}_\Gamma$ *and a valuation* φ *on* Γ*, to a set* $[\![A]\!]_{\varphi}^{\Gamma}$ *of terms of* Λ*, such that:*

- *for all contexts* Γ, Γ'*,* $A \in \mathcal{I}_\Gamma$ *and valuations* φ *on* Γ*, if* $\Gamma \subseteq \Gamma'$*, then* $[\![A]\!]_{\varphi}^{\Gamma} \subseteq [\![A]\!]_{\varphi}^{\Gamma'}$*,*
- *for all contexts* Γ*,* $A \in \mathcal{I}_\Gamma$ *and valuations* $\varphi + \langle x, u\rangle$ *on* Γ*, if* $t \in [\![A]\!]_{\varphi}^{\Gamma}$ *and* $x \notin FV(t)$*, then* $t \in [\![A]\!]_{\varphi+\langle x,u\rangle}^{\Gamma}$*.*

This second property is, up to our knowledge, unusual. This property is necessary in order to build what we call *adequate valuations* (given an interpretation $[\![.]\!]_{\cdot}$ and a context Γ). An adequate valuation for $[\![.]\!]_{\cdot}^{\Gamma}$, is either empty or a valuation $\varphi + \langle x, u\rangle$ such that x is declared of a type B in Γ and $u \in [\![B]\!]_{\varphi}^{\Gamma}$. We shall see that we can only consider this kind of valuations in our definition of pre-models since we only use this kind of valuations in the adequacy lemma. And we shall see that it is very convenient for proving completeness of those pre-models for strong normalization.

Definition 4.20 (Adequate valuations)**.**
Given an interpretation $[\![.]\!]_{\cdot}$ *and a (well-formed) context* Γ*, an* adequate valuation *on* Γ *for* $[\![.]\!]_{\cdot}$ *(we shall also say "adequate for* $[\![.]\!]_{\cdot}^{\Gamma}$*"), is defined inductively as follows:*

- *the empty valuation is adequate on* Γ *for* $[\![.]\!]_{\cdot}$*,*
- *if* φ *is adequate on* Γ *for* $[\![.]\!]_{\cdot}$*,* $x : B \in \Gamma$ *(with* $x \notin \textsc{Dom}(\varphi)$*), and* $u \in [\![B]\!]_{\varphi}^{\Gamma}$*, then* $\varphi + \langle x, u\rangle$ *is adequate on* Γ *for* $[\![.]\!]_{\cdot}$*.*

We shall say "adequate valuations on a context Γ*" when there is no ambiguity about the considered interpretation.*

Definition 4.21 (Length of a valuation)**.**
The length *of a valuation is the number of variables it substitutes. Proofs* by induction on a valuation φ *are done by induction on the length of* φ*.*

Let us now define pre-models for λΠ-calculus modulo. They are interpretations such that each associated interpretation of a type in $\mathcal{I}_\Gamma$ and an adequate valuation on Γ satisfies (CR_1), (CR_2) and (CR'_3), and that are non-empty if there exists a variable declared of this type in the considered context or in

Σ. Such that interpretations of $Kind$ are modeled on the associated typing rule (i.e. contain the term $Type$). Such that interpretations of products are modeled on the associated elimination typing rules. Such that interpretations are adapted to the congruence (interpretations of $\equiv_{\beta\mathcal{R}}$-equivalent types are equal). And finally such that interpretations are adapted to well-typed substitution (see behind for a formal definition).

Definition 4.22.
A variable x is said to be fresh *for a (well-formed) context Γ is x is not declared in Γ, nor in Σ (in this case it cannot appear free in a type of Γ).*

Definition 4.23 (pre-model)**.**
An interpretation $[\![.]\!]^{\cdot}$ is a pre-model *iff for all (well-formed) contexts Γ,*

(a) interpretations of types which appear in $\Sigma\Gamma$ are non-empty and for all declarations $x : A$ in Γ, x is in the interpretation of A:

1. *for all adequate valuations φ on Γ, $A \in \mathcal{I}_\Gamma$ and $x : A \in \Sigma\Gamma$ such that $x \notin \textsc{Dom}(\varphi)$, then $x \in [\![A]\!]^\Gamma_\varphi$*
2. *for all adequate valuations φ on Γ, $A \in \mathcal{I}_\Gamma$ and $x : A \in \Sigma\Gamma$, $[\![A]\!]^\Gamma_\varphi$ is non-empty and contains a term whose free variables are all declared in Γ,*

(b) interpretations satisfy (CR_1), (CR_2) and (CR'_3):

1. *for all adequate valuations φ on Γ and $A \in \mathcal{I}_\Gamma$, $[\![A]\!]^\Gamma_\varphi$ satisfies (CR_1), (CR_2) and (CR'_3),*

(c) interpretations of $Type$, $Kind$ and products are modeled on the associated typing rules:

1. *for all adequate valuations φ on Γ, $Type \in [\![Kind]\!]^\Gamma_\varphi$*
2. *for all adequate valuations φ on Γ, variables x and propositions A, B such that $\Gamma[x : A]$ is a well-formed context, if $A \in [\![Type]\!]^\Gamma_\varphi$ and $B \in [\![Type]\!]^{\Gamma[x:A]}_\varphi$, then $\Pi x : A.B \in [\![Type]\!]^\Gamma_\varphi$,*
3. *for all adequate valuations φ on Γ, variables x and propositions A, B such that $\Gamma[x : A]$ is a well-formed context, if $A \in [\![Type]\!]^\Gamma_\varphi$ and $B \in [\![Kind]\!]^{\Gamma[x:A]}_\varphi$, then $\Pi x : A.B \in [\![Kind]\!]^\Gamma_\varphi$,*
4. *for all adequate valuations φ on Γ and $\Pi x : A.B \in \mathcal{I}_\Gamma$ with x fresh for Γ,*
   $[\![\Pi x : A.B]\!]^\Gamma_\varphi = \{t$ s.t. for all $u \in [\![A]\!]^{\Gamma[x:A]}_\varphi$, $tu \in [\![B]\!]^{\Gamma[x:A]}_{\varphi+\langle x,u\rangle}\}$,

(d) interpretations are adapted to the $\beta\mathcal{R}$-congruence:

1. *for all adequate valuations φ on Γ and $A, B \in \mathcal{I}_\Gamma$, if $A \equiv_{\beta\mathcal{R}} B$ in Γ, then $[\![A]\!]^\Gamma_\varphi = [\![B]\!]^\Gamma_\varphi$.*

(e) interpretations are adapted to well-typed substitution:

1. *for all adequate valuations φ on Γ, $t, B \in \Lambda$, variables x not declared and not free in Γ and $A \in \mathcal{I}_{\Gamma[x:B]}$, if $\Sigma\Gamma \vdash_\mathcal{R} t : B$, then $[\![A]\!]^{\Gamma[x:B]}_{\varphi+\langle x,\varphi t\rangle} \subseteq [\![(t/x)A]\!]^\Gamma_\varphi$ (notice that in this case, $(t/x)A \in \mathcal{I}_\Gamma$).*

4.3.2 Soundness of pre-models for strong normalization

Let us now prove that this definition of pre-models provides a sound semantics for strong normalization in λΠ-calculus modulo (that is: if a theory is λΠ-calculus modulo has a pre-model, then it is strongly normalizing). We have to refine again the statement of our adequacy lemma by syncronizing one more time the different valuations/substitutions which appear in this statement. As previously for deduction modulo, this synchronization reflects the fact that our definition of pre-models is modeled on the typing rules of the λΠ-calculus modulo, in order to provide an also complete semantics for strong normalization. There was three different valuations/substitutions in the adequacy lemma for pre-models for deduction modulo, two for models valued in LDTVAs , and now only one for pre-models for λΠ-calculus modulo: we prove that for specific valuations if $\Sigma\Gamma \vdash_\mathcal{R} t : A$ then $\sigma t \in [\![A]\!]^\Gamma_\sigma$. In the usual adequacy lemma, those specific valuations are the ones which map variables declared of type B in Γ to elements of the interpretation of B. In this synchronized version, we should therefore consider valuations σ on Γ such that for all $x : B \in \Gamma$, $\sigma x \in [\![B]\!]^\Gamma_\sigma$. We shall actually consider, more generally, total (in the sense that they map all the variables in Γ) adequate valuations. These total adequate valuations satisfy this property: if Γ is empty, the empty valuation satisfy this property, and if $\Gamma = \Gamma'[x : B]$, an adequate valuation $\sigma + \langle x, u\rangle$ satisfies the fact that $u \in [\![B]\!]^\Gamma_\varphi$ and also $u \in [\![B]\!]^\Gamma_{\varphi+\langle x,u\rangle}$, as x cannot be free in u and $[\![.]\!]^{\cdot}_{\cdot}$ is an interpretation.

Proposition 4.1. *For all pre-models $[\![.]\!]^{\cdot}_{\cdot}$ and well-formed contexts Γ, if $\Sigma\Gamma \vdash_\mathcal{R} t : A$ then for all total adequate valuations σ on $[\![.]\!]^\Gamma_{\cdot}$, we have $\sigma t \in [\![A]\!]^\Gamma_\sigma$.*

Proof. By induction on the length of the derivation $\Sigma\Gamma \vdash_\mathcal{R} t : A$. By case on the last rule used in this derivation. If the last rule is:

- **Sort**: we have $\sigma Type = Type \in [\![Kind]\!]^{\Gamma}_{\sigma}$, since $[\![.]\!]$ is a pre-model.

- **Variable**: then t is a variable x such that either $x : A \in \Sigma$ or $x : A \in \Gamma$. Therefore $\sigma x \in [\![A]\!]^{\Gamma}_{\sigma}$, by hypothesis on σ and since $[\![.]\!]$ is a pre-model (remind that if x is declared in Σ, then $\sigma x = x$).

- **Product**: then t is a product $\Pi x : B.C$ and A is $Type$ (resp. $Kind$). Notice that we can (and have to) choose x such that x is not declared in Γ therefore σ does not substitute x, and we have $x \in [\![B]\!]^{\Gamma[x:B]}_{\sigma}$. By induction hypothesis, we have $\sigma B \in [\![Type]\!]^{\Gamma}_{\sigma}$ and $(x/x)\sigma C \in [\![Type]\!]^{\Gamma[x:B]}_{\sigma+\langle x,x\rangle} = [\![Type]\!]^{\Gamma[x:B]}_{\sigma}$ (resp. $[\![Kind]\!]^{\Gamma[x:B]}_{\sigma}$). Hence $\sigma(\Pi x : B.C) = \Pi x : \sigma B.\sigma C \in [\![Type]\!]^{\Gamma}_{\sigma}$ (resp. $[\![Kind]\!]^{\Gamma}_{\sigma}$), since $[\![.]\!]$ is a pre-model.

- **Abstraction**: then $t = \lambda x : B.u$ and $A = \Pi x : B.C$. As previously, notice that we have to choose x such that x is not declared in Γ therefore σ does not substitute x, and we have $x \in [\![B]\!]^{\Gamma[x:B]}_{\sigma}$. By induction hypothesis, $\sigma B \in [\![Type]\!]^{\Gamma}_{\sigma}$ and for all $v \in [\![B]\!]^{\Gamma[x:B]}_{\sigma}$, $(v/x)\sigma u \in [\![C]\!]^{\Gamma[x:B]}_{\sigma+\langle x,v\rangle}$. Hence we can prove by induction on the sum of maximal lengths of reductions sequences from σB, σu and v (all in SN by induction hypothesis and (CR_1)), that $(\lambda x : \sigma B.\sigma u)\ v \in [\![C]\!]^{\Gamma[x:B]}_{\sigma+\langle x,v\rangle}$ using (CR_2) and (CR_3'), as usual. Finally $\sigma t = \lambda x : \sigma B.\sigma u \in [\![\Pi x : B.C]\!]^{\Gamma}_{\sigma}$.

- **Application**: then $t = uv$ and $A = (v/x)C$, with $\sigma u \in [\![\Pi x : B.C]\!]^{\Gamma}_{\sigma}$ and $\sigma v \in [\![B]\!]^{\Gamma}_{\sigma}$, by induction hypothesis. Hence $\sigma(uv) \in [\![C]\!]^{\Gamma[x:B]}_{\sigma+\langle x,\sigma v\rangle}$ since $[\![.]\!]$ is a pre-model. Moreover, since $\Sigma\Gamma \vdash_{\mathcal{R}} v : B$, we have $[\![C]\!]^{\Gamma[x:B]}_{\sigma+\langle x,\sigma v\rangle} \subseteq [\![(v/x)C]\!]^{\Gamma}_{\sigma}$ and therefore $\sigma(uv) \in [\![((v/x)C)]\!]^{\Gamma}_{\sigma}$.

- **Conversion**: then, by induction hypothesis, we have $\sigma t \in [\![B]\!]^{\Gamma}_{\sigma}$, with $B \equiv_{\beta\mathcal{R}} A$ in Γ. Hence $[\![A]\!]^{\Gamma}_{\sigma} = [\![B]\!]^{\Gamma}_{\sigma}$, as $[\![.]\!]$ is a pre-model. Finally, $\sigma t \in [\![A]\!]^{\Gamma}_{\sigma}$.

□

We can therefore prove that having a pre-model is a sound semantics for theories expressed in λΠ-calculus modulo.

Proposition 4.2 (Soundness)**.**
If a theory in λΠ-modulo has a pre-model, then it is strongly normalizing.

Proof. If $\Sigma\Gamma \vdash_{\mathcal{R}} t : A$, then either Γ is empty, and in this case, $t \in SN$ by proposition 4.1, or Γ is non-empty and therefore there exists a substitution σ as in proposition 4.1: since the theory has a pre-model $[\![.]\!]$, if we write $\Gamma = [x_1 : A_1] \dots [x_n : A_n]$, we know that $[\![A_1]\!]^{[x_1:A_1]}_{\bullet} \neq \emptyset$ (with $\bullet$ the empty

substitution) and therefore contains an element t_1 (whose all free variables are declared in $[x_1 : A_1]$). And so on, $[\![A_2]\!]^{[x_1:A_1][x_2:A_2]}_{\langle x_1,t_1\rangle} \neq \emptyset$ and therefore contains an element t_2 (whose all free variables are declared in $[x_1 : A_1][x_2 : A_2]$). This way we are able to build σ as $\langle x_1, t_1\rangle + \ldots + \langle x_n, t_n\rangle$, and it satisfies, for all $i \leq n$, $\sigma x_i \in [\![A_i]\!]^{\Gamma}_{\sigma}$, since $[x_1 : A_1] \ldots [x_i : A_i] \subseteq \Gamma$, and $x_{i+1}, \ldots, x_n$ are not free in t_i. Finally, since $\sigma t \in SN$, we also have $t \in SN$.

□

We shall see, in the following subsection that we can prove the strong normalization of the λΠ-calculus by building such a pre-model for the λΠ-calculus based on an empty signature modulo an empty set of rewrite rules.

4.3.3 An example of pre-model

We present, in this subsection, the simplest corollary of proposition 4.2 about soundness of pre-models for strong normalization: the λΠ-calculus is strongly normalizing. By λΠ-calculus, we mean here the empty theory in λΠ-calculus modulo, i.e. the λΠ-calculus based on an empty signature modulo an empty set of rewrite rules.
Let us first remark that for all (well-formed) contexts Γ, $\mathcal{I}_\Gamma$ can be defined as the union of the singleton $\{Kind\}$ and the set which is composed by (well-typed) β-expansions of what we call *atomic types* and (well-typed) β-expansions of products inductively built on those (β-expansions of) atomic types.

Definition 4.24 (Atomic types)**.**
Given a (well-formed) context Γ, we call atomic types*, terms of the form $x\ t_1 \ldots t_n$ with x a variable, $n \in \mathbb{N}$ and $t_1 \ldots t_n$ terms such that $\Sigma\Gamma \vdash_{\mathcal{R}} x\ t_1 \ldots t_n : Type$ or $\Sigma\Gamma \vdash_{\mathcal{R}} x\ t_1 \ldots t_n : Kind$.*
In the following, we shall also call atomic*, well-typed β-expansions of atomic types (notice that such a β-expansion cannot be a product).*

Remark 4.3. *We can therefore define an interpretation by giving its value on $Kind$, $Type$ and atomic types, and by defining inductively the interpretation of (β-expansions of) a product $\Pi x : A.B$ a context Γ and a valuation φ as the set of proof terms which map elements u of the interpretation of A, $\Gamma[x : A]$ and φ to elements of the interpretation of B, $\Gamma[x : A]$ and $\varphi + \langle x, u\rangle$. The interpretations defined this way automatically satisfy the property (c_4) of pre-models. Moreover, by defining interpretations of $Kind$, $Type$ and atomic types as the set SN, we automatically obtain the properties (a_1), (a_2), (b_1), (c_1), (c_2) and (c_3). And finally we obtain the property (e) by the fact that the interpretations of types we propose do not depend on the considered valuation and the fact that for all well-typed substitutions (t/x), A is an atomic type if and only if $(t/x)A$ is also an atomic type.*

Let us define such an interpretation for the empty theory in λΠ-calculus modulo and verify that we automatically obtain all these properties.

Definition 4.25 ($[.]_{\cdot}$)**.**
For all contexts Γ *and valuations* φ *on* Γ*,*

- $[Kind]^{\Gamma}_{\varphi} = SN$,
- $[Type]^{\Gamma}_{\varphi} = SN$,
- *for all* $A \in \mathcal{I}_\Gamma$,

  - if A *is (a well-typed* β*-expansion of) an atomic type, then* $[A]^{\Gamma}_{\varphi} = SN$,
  - if A *is (a well-typed* β*-expansion of) a product* $\Pi x : B.C \in \mathcal{I}_\Gamma$*,*
  then $[A]^{\Gamma}_{\varphi} = \{t$ *s.t. for all* $u \in [B]^{\Gamma[x:B]}_{\varphi}$, $tu \in [C]^{\Gamma[x:B]}_{\varphi+\langle x,u\rangle}\}$

We then verify that this provides a pre-model for λΠ-calculus modulo the empty theory.

Lemma 4.6.
$[.]_{\cdot}$ *is a pre-model for the* λΠ*-calculus based on an empty signature modulo an empty set of rewrite rules.*

Proof. Let us first notice, that the interpretation of a type is the same in any context Γ and valuations φ (as soon as it is a type in Γ and φ is a valuation on Γ). Hence $[.]_{\cdot}$ is a well-defined interpretation: for all contexts Γ, Γ', $A \in \mathcal{I}_\Gamma$ and valuations φ on Γ, if $\Gamma \subseteq \Gamma'$, then $[A]^{\Gamma}_{\varphi} \subseteq [A]^{\Gamma'}_{\varphi}$. Moreover for all contexts Γ, $A \in \mathcal{I}_\Gamma$ and valuations $\varphi + \langle x, u\rangle$ on Γ, if $t \in [A]^{\Gamma}_{\varphi}$ and $x \notin FV(t)$, then $t \in [A]^{\Gamma}_{\varphi+\langle x,u\rangle}$, since $[A]^{\Gamma}_{\varphi} = [A]^{\Gamma}_{\varphi+\langle x,u\rangle}$. Let us now prove that it is a pre-model.

(a)
1. for all adequate valuations φ on Γ, $A \in \mathcal{I}_\Gamma$ and $x : A \in \Gamma$ such that $x \notin \textsc{Dom}(\varphi)$, then $x \in [A]^{\Gamma}_{\varphi}$. Indeed, we can prove, by induction on A that $[A]^{\Gamma}_{\varphi}$ contains all isolated strongly normalizing terms (remind that isolated terms are neutral terms which reduce only to neutral terms). If A is $Kind$ or (a well-typed β-expansion of) an atomic type, it is obvious. If A is (a well-typed β-expansion of) a product, we can notice that if t is isolated and strongly normalizing and u is strongly normalizing, then tu is isolated and strongly normalizing.
2. for all adequate valuations φ on Γ, $A \in \mathcal{I}_\Gamma$ and $x : A \in \Gamma$, $[A]^{\Gamma}_{\varphi}$ is non-empty and contains a term whose free variables are all declared in Γ. We can use the same argument as in the previous point: if $x : A \in \Gamma$, then $x \in [A]^{\Gamma}_{\varphi}$.

(b) 1. for all adequate valuations φ on Γ and $A \in \mathcal{I}_\Gamma$, $\lfloor A \rfloor^\Gamma_\varphi$ satisfies (CR_1). Indeed, for $Kind$ and (well-typed β-expansions of) atomic types on Γ, SN satisfies obviously (CR_1). And for (well-typed β-expansions of) product types $\Pi x : B.C$, since there exists a term u in $[B]^{\Gamma[x:B]}_\varphi$, $tu \in [C]^{\Gamma[x:B]}_{\varphi+\langle x,u\rangle}$, hence $tu \in SN$ by induction hypothesis and so does t.

2. for all adequate valuations φ on Γ and $A \in \mathcal{I}_\Gamma$, $[A]^\Gamma_\varphi$ satisfies (CR_2). Indeed, SN is stable by β-reduction. And for product types $\Pi x : B.C$, if $t \in [\Pi x : B.C]^\Gamma_\varphi$ and $t \longrightarrow_\beta t'$ then for all $u \in [B]^{\Gamma[x:B]}_\varphi$, $t'u$ is a $\beta\mathcal{R}$-reduct of tu and is therefore in $[C]^{\Gamma[x:B]}_{\varphi+\langle x,u\rangle}$ since it satisfies (CR_2) by induction hypothesis.

3. for all adequate valuations φ on Γ and $A \in \mathcal{I}_\Gamma$, $\lfloor A \rfloor^\Gamma_\varphi$ satisfies (CR_3'). Indeed SN satisfies (CR_3') and if $[C]^{\Gamma[x:B]}_{\varphi+\langle x,u\rangle}$ satisfies (CR_3') (with $u \in [B]^{\Gamma[x:B]}_\varphi$) then $[\Pi x : B.C]^\Gamma_\varphi$ also satisfies (CR_3').

(c) 1. for all adequate valuations φ on Γ, $Type \in [Kind]^\Gamma_\varphi$, since $Type$ is strongly normalizing.

2. for all adequate valuations φ on Γ, variables x and propositions A, B such that $\Gamma[x : A]$ is a well-formed context, if $A \in [Type]^\Gamma_\varphi$ and $B \in [Type]^{\Gamma[x:A]}_\varphi$, then $\Pi x : A.B \in [Type]^\Gamma_\varphi$. Since $[Type]^\Gamma_\varphi = SN$ for all well-formed contexts Γ and valuations φ on Γ. And if $A, B \in SN$ then so does $\Pi x : A.B$.

3. for all adequate valuations φ on Γ, variables x and propositions A, B such that $\Gamma[x : A]$ is a well-formed context, if $A \in [Type]^\Gamma_\varphi$ and $B \in [Kind]^{\Gamma[x:A]}_\varphi$, then $\Pi x : A.B \in [Kind]^\Gamma_\varphi$, by the same argument as in the previous point.

4. for all adequate valuations φ on Γ and $\Pi x : A.B \in \mathcal{I}_\Gamma$ with x fresh for Γ, $[\Pi x : A.B]^\Gamma_\varphi = \{t \text{ s.t. for all } u \in [A]^{\Gamma[x:A]}_\varphi,\ tu \in [B]^{\Gamma[x:A]}_{\varphi+\langle x,u\rangle}\}$, by construction.

(d) 1. for all adequate valuations φ on Γ and $A, B \in \mathcal{I}_\Gamma$, if $A \equiv_\beta B$ in Γ, then $[A]^\Gamma_\varphi = [B]^\Gamma_\varphi$, since the interpretations of a type and one of its (well-typed) β-expansions are defined equal.

(e) 1. for all adequate valuations φ on Γ, $t, B \in \Lambda$, variables x not declared and not free in Γ and $A \in \mathcal{I}_{\Gamma[x:B]}$, if $\Gamma \vdash_\mathcal{R} t : B$, then $[A]^{\Gamma[x:B]}_{\varphi+\langle x,\varphi t\rangle} \subseteq [(t/x)A]^\Gamma_\varphi$. Indeed we can prove by induction on A, that $[(t/x)A]^\Gamma_\varphi = [A]^\Gamma_\varphi$ since in this case; if A is atomic then so does $(t/x)A$. Moreover, $[A]^\Gamma_\varphi = [A]^\Gamma_{\varphi+\langle x,\varphi t\rangle}$ as the interpretation of A is the same for all valuations ψ on Γ.

□

Corollary 4.1.
The λΠ-calculus based on an empty signature modulo an empty set of rewrite rules is strongly normalizing.

We have seen that if we define the interpretation of a type A, a context Γ and a valuation φ by induction on A as the set SN for (β-expansions of) atomic types and using the interpretation of products of pre-models for (β-expansions of) products, the only property we do not obtain automatically is the property (d_1): the fact that it is adapted to the congruence. But, as we have identified the interpretations of types and their β-expansions, we can do the same for the rewrite rules of the theory studied, as explained in the following lemma in which we prove the strong normalization of the theory defined in example 4.1.

Lemma 4.7.
Let Σ_0 be the signature $[P : Type, Q : Type]$
and $\mathcal{R}_0$ be the single rewrite rule $P \longrightarrow^{[],Type} \Pi x : Q.Q$. For all contexts Γ and valuations φ on Γ, we define:

- $[Kind]^{\Gamma}_{\varphi} = SN$,
- $[Type]^{\Gamma}_{\varphi} = SN$,
- $[P]^{\Gamma}_{\varphi} = \{t$ *such that for all* $u \in SN$, $tu \in SN\}$
  (and is equal to the interpretations of β-expansions of P),
- *Otherwise, for all $A \in \mathcal{I}_\Gamma$,*
  - if A is (a well-typed β-expansion of) an atomic type, then $[A]^{\Gamma}_{\varphi} = SN$,
  - if A is (a well-typed β-expansion of) a product $\Pi x : B.C \in \mathcal{I}_\Gamma$,
  then $[A]^{\Gamma}_{\varphi} = \{t$ *s.t. for all* $u \in [B]^{\Gamma[x:B]}_{\varphi}$, $tu \in [C]^{\Gamma[x:B]}_{\varphi+\langle x,u\rangle}\}$

$[.]$: is a pre-model hence the λΠ-calculus based on Σ_0 modulo $\mathcal{R}_0$ is strongly normalizing.

Proof. The only difference with the interpretation for the empty theory we have previously defined, is that we interpret P and its (well-typed) β-expansions directly as the interpretation of $\Pi x : Q.Q$ in order to obtain the property (d_1).

□

Remark 4.4. *Notice that we cannot use this method for building a pre-model for the theory defined by the signature $[P : Type]$ and the single rewrite rule $P \longrightarrow^{[],Type} P$, since P is not strongly normalizing in this case and is therefore not in the interpretation of $Type$. An interpretation built using this method cannot therefore satisfy the property (a_1) and it is not a pre-model (fortunately because this theory is not strongly normalizing in λΠ-calculus*

modulo). This theory reveals one of the differences between expressing a theory in deduction modulo and expressing it in λΠ-calculus modulo, as it is strongly normalizing in deduction modulo but not strongly normalizing in λΠ-calculus modulo.

We have proposed a method for building a pre-model for a given theory in λΠ-calculus modulo, obtaining, as a corollary of soundness of those pre-models for strong normalization in λΠ-calculus modulo, the strong normalization of this given theory. We shall see, in the following subsection that this definition of pre-models also provide a complete semantics for strong normalization in λΠ-calculus modulo.

4.3.4 Completeness of pre-models for strong normalization

In order to prove that this definition of pre-models also provides a complete semantics for strong normalization in λΠ-calculus modulo, we use the same method as in sections 3.2.4 and 3.3.4. Since we have not defined a notion of algebras for λΠ-calculus modulo, we cannot use morphisms on those not defined algebras, but we shall define directly our candidate to be a pre-model, as the sets of (CR_3') expansions of sets of *well-typed* terms. We can therefore see the following proof as an adapted version of the one of sections 3.2.4 and 3.3.4 with an eliminated cut.

In order to use our technique for proving completeness, we would want to define the equivalent of our $\mathcal{C}_{\equiv}$-interpretation (which we shall call 0-interpretation in the following) as the function which, given a context Γ, associates to a type A in $\mathcal{I}_\Gamma$ and a valuation φ the set of terms typed by φA in Γ, however φA is not necessarily in $\mathcal{I}_\Gamma$. If we build this equivalent of $\mathcal{C}_{\equiv}$-interpretation as the function which associates, given a context Γ, to a type A and a valuation φ the set of terms φt with t of type A in Γ, we shall be able to deduce from the fact that the theory is strongly normalizing, the fact that all the elements of this set are strongly normalizing, but only for an adequate valuation. We shall therefore define our interpretation only for adequate valuations, by first defining a function $Cl(.)^\Gamma_\bullet$ from $\mathcal{I}_\Gamma$ to $\mathbb{P}(\Lambda_\Sigma)$ as the (CR_3') expansions of sets of *well-typed* terms. And then we shall define, inductively adequate valuations and $Cl(A)^\Gamma_\varphi$, with φ an adequate valuation, as the set which contains all φt with $t \in Cl(A)^\Gamma_\bullet$. We can therefore build our interpretation only on adequate valuations. This is sufficient since we only need this kind of valuations in the definition of pre-models.

Let us first define the function $Cl(.)^\Gamma_\bullet$ from $\mathcal{I}_\Gamma$ to $\mathbb{P}(\Lambda_\Sigma)$ as the (CR_3') expansion of sets of well-typed terms in Γ.

Definition 4.26 ($Cl(.)^\Gamma_\bullet$).
For all (well-formed) contexts Γ *and* $A \in \mathcal{I}_\Gamma$,

$Cl^0(A)^\Gamma_\bullet = \{t \text{ such that } \Sigma\Gamma \vdash_\mathcal{R} t : A\}$
$Cl^{k+1}(A)^\Gamma_\bullet = \{t, \text{ such that } \exists n \in \mathbb{N}:$
$\exists v, \exists (u_i)_{i\leq n}, \text{ each neutral non-normal such that}$
$t = [u_i/x_i]_{i\leq n}\ v \text{ and } \forall (r_i)_{i\leq n}, \text{ s.t. } \forall i \leq n,\ u_i \to_{\beta\mathcal{R}} r_i,$
$\text{we have } [r_i/x_i]_{i\leq n}\ v \in Cl^k(A)^\Gamma_\bullet\}$
$Cl(A)^\Gamma_\bullet = \cup_{j\in\mathbb{N}}\, Cl^j(A)^\Gamma_\bullet$

We then define inductively $Cl(.)^\Gamma_\varphi$, with φ an adequate valuation, as the set which contains all φt with $t \in Cl(A)^\Gamma_\bullet$.

Definition 4.27 ($Cl(.)^\Gamma_\varphi$)**.**
For all (well-formed) contexts Γ*, valuations* φ *on* Γ *and* $A \in \mathcal{I}_\Gamma$*,*

- *if* φ *is the empty valuation, then* $Cl(A)^\Gamma_\varphi = Cl(A)^\Gamma_\bullet$*,*
- *otherwise, if* $\varphi = \psi + < x, u >$*,* $x : B \in \Gamma$ *and* $u \in Cl(B)^\Gamma_\psi$*, then* $Cl(A)^\Gamma_\varphi = \{(u/x)t,\ t \in Cl(A)^\Gamma_\psi\}$

This definition provides an interpretation, as it easy to verify that for all contexts Γ, Γ', $A \in \mathcal{I}_\Gamma$ and valuations φ on Γ, if $\Gamma \subseteq \Gamma'$, we have $Cl(A)^\Gamma_\varphi \subseteq Cl(A)^{\Gamma'}_\varphi$, by weakening, in the case of an adequate valuation. And that for all contexts Γ, $A \in \mathcal{I}_\Gamma$ and adequate valuations $\varphi + \langle x, u\rangle$ on Γ, if $t \in Cl(A)^\Gamma_\varphi$ and $x \notin FV(t)$, then $t \in Cl(A)^\Gamma_{\varphi+\langle x,u\rangle}$ since $t \in Cl(A)^\Gamma_{\varphi+\langle x,u\rangle}$ by construction. For non-adequate valuations φ, we can define $Cl(A)^\Gamma_\varphi$ as the set SN. In this case, both previous properties are satisfied.

Now let us prove in the following lemmas, that $Cl(.)$: forms a pre-model when the considered theory is strongly normalizing.

Lemma 4.8. *For all (well-formed) contexts* Γ*, adequate valuations* φ *on* Γ*,* $A \in \mathcal{I}_\Gamma$ *and* $x : A \in \Sigma\Gamma$ *such that* $x \notin \textsc{Dom}(\varphi)$*, then* $x \in Cl(A)^\Gamma_\varphi$*.*

Proof. For all $x : A \in \Sigma\Gamma$, $\Sigma\Gamma \vdash_\mathcal{R} x : A$, therefore $x \in Cl^0(A)^\Gamma_\bullet \subseteq Cl(A)^\Gamma_\bullet$, and $x = \varphi x \in Cl(A)^\Gamma_\varphi$ if $x \notin \textsc{Dom}(\varphi)$.

□

Lemma 4.9. *For all (well-formed) contexts* Γ*, adequate valuations* φ *on* Γ*,* $A \in \mathcal{I}_\Gamma$ *and* $x : A \in \Sigma\Gamma$*,* $Cl(A)^\Gamma_\varphi$ *is non-empty and contains a term whose free variables are all declared in* Γ*.*

Proof. For all $x : A \in \Sigma\Gamma$, $\Sigma\Gamma \vdash_\mathcal{R} x : A$, therefore $x \in Cl^0(A)^\Gamma_\bullet \subseteq Cl(A)^\Gamma_\bullet$, and $\varphi x \in Cl(A)^\Gamma_\varphi$, notice that free variables of φx are all declared in Γ.

□

Lemma 4.10. *If the theory is stronly normalizing, then for all (well-formed) contexts* Γ*,* $A \in \mathcal{I}_\Gamma$ *and adequate valuations* φ *on* Γ*,* $Cl(A)^\Gamma_\varphi$ *satisfies (*CR_1*), (*CR_2*) and (*CR_3*').*

Proof. The proofs that $Cl(A)^\Gamma_\varphi$ satisfies (CR_1), (CR_2) and (CR_3') follow the ones we have seen in section 3.3.4.

- If the theory is strongly normalizing, then $Cl^0(A)^\Gamma_\bullet \subseteq SN$ and so does $Cl(A)^\Gamma_\bullet$, as done for deduction modulo in section 3.3.4 (all reductions sequences from an element of $Cl(A)^\Gamma_\bullet$ reach a well-typed term). Then we can prove by induction on the length of an adequate valuation φ that $Cl(A)^\Gamma_\varphi \subseteq SN$ (as all reductions sequences from an element of $Cl(A)^\Gamma_\varphi$ reach a well-typed term). Indeed, if φ is the empty valuation then $Cl(A)^\Gamma_\varphi = Cl(A)^\Gamma_\bullet$. Otherwise, if $\varphi = \psi + \langle x, u\rangle$, $Cl(A)^\Gamma_\psi \subseteq SN$ by induction hypothesis and all terms in $Cl(A)^\Gamma_\varphi$ are of the form $(u/x)t$ with $t \in Cl(A)^\Gamma_\psi$. Therefore we can prove that all reduction sequences from such a term $(u/x)t$ will also reach a well-typed term (as done in section 3.3.4 for deduction modulo) since all reduction sequences from u will reach a term of the same type as x in Γ, and therefore $(u/x)t$ is strongly normalizing.
- $Cl^0(A)^\Gamma_\bullet$ is stable by $\beta\mathcal{R}$-reduction, by subject-reduction property, hence its (CR_3') expansion $Cl(A)^\Gamma_\bullet$ is stable by $\beta\mathcal{R}$-reduction, as done for deduciton modulo in section 3.3.4. Finally, $Cl(A)^\Gamma_\varphi$ is also stable by $\beta\mathcal{R}$-reduction, since terms substituted in φ are (CR_3') expansions of terms of the same type in Γ as the associated variable.
- $Cl(A)^\Gamma_\bullet$ satisfies (CR'_3) by construction. And so does $Cl(A)^\Gamma_\varphi$ are terms substituted in φ are yet (CR_3') expansions of terms of the same type in Γ as the associated variable.

□

Lemma 4.11.
For all (well-formed) contexts Γ *and adequate valuations on* Γ*, we have* $Type \in Cl(Kind)^\Gamma_\varphi$*.*

Proof. For all adequate valuations φ on Γ, $Type \in Cl^0(Kind)^\Gamma \subseteq Cl(Kind)^\Gamma$, therefore $\varphi Type = Type \in Cl(Kind)^\Gamma_\varphi$.

□

Lemma 4.12.
For all (well-formed) contexts Γ*, adequate valuations* φ *on* Γ*, variables* x *fresh for* Γ*,* $A \in Cl(Type)^\Gamma_\varphi$ *and* $B \in Cl(Type)^{\Gamma[x:A]}_\varphi$*, (resp.* $Cl(Kind)^{\Gamma[x:A]}_\varphi$*),* $\Pi x : A.B \in Cl(Type)^\Gamma_\varphi$ *(resp.* $Cl(Kind)^\Gamma_\varphi$*).*

Proof. By induction on φ.

- If φ is the empty substitution, there exists $k, j \in \mathbb{N}$ such that $A \in Cl^k(Type)^\Gamma_\bullet$ and $B \in Cl^j(Type)^{\Gamma[x:A]}_\bullet$ (resp. $Cl^j(Kind)^{\Gamma[x:A]}_\bullet$). By induction on $k+j$.
  - If $k+j=0$, then $\Sigma\Gamma \vdash_\mathcal{R} A : Type$ and $\Sigma\Gamma[x:A] \vdash_\mathcal{R} B : Type$ (resp. $Kind$). Hence we have $\Sigma\Gamma \vdash_\mathcal{R} \Pi x:A.B : Type$ (resp. $Kind$) therefore $\Pi x:A.B \in Cl^0(Type)^\Gamma_\bullet \subseteq Cl(Type)^\Gamma_\bullet$ (resp. $Cl^0(Kind)^\Gamma_\bullet \subseteq Cl(Kind)^\Gamma_\bullet$).
  - If $k > 0$, then A is a (one-step) (CR_3') expansion of a term $A' \in Cl^{k-1}(Type)^\Gamma_\bullet$. Hence $\Pi x : A'.B \in Cl(Type)^\Gamma_\bullet$ (resp. $Cl(Kind)^\Gamma_\bullet$) by induction hypothesis. And $\Pi x:A.B$ is a (CR_3') expansion of $\Pi x:A'.B$ therefore $\Pi x:A.B \in Cl(Type)^\Gamma_\bullet$ (resp. $Cl(Kind)^\Gamma_\bullet$) since it satisfies (CR_3').
  - If $j > 0$, then B is a (one-step) (CR_3') expansion of a term $B' \in Cl^{j-1}(Type)^\Gamma_\bullet$ (resp. $Cl^{j-1}(Kind)^\Gamma_\bullet$). Hence we have $\Pi x:A.B' \in Cl(Type)^\Gamma_\bullet$ (resp. $Cl(Kind)^\Gamma_\bullet$) by induction hypothesis. And $\Pi x:A.B$ is a (CR_3') expansion of $\Pi x:A.B'$ therefore $\Pi x:A.B \in Cl(Type)^\Gamma_\bullet$ (resp. $Cl(Kind)^\Gamma_\bullet$) since it satisfies (CR_3').

- If $\varphi = \psi + \langle y, v\rangle$, then $A = (v/y)A'$ with $A' \in Cl(Type)^\Gamma_\psi$ and $B = (v/y)B'$ with $B' \in Cl(Type)^\Gamma_\psi$. By induction hypothesis, we have $\Pi x:A'.B' \in Cl(Type)^\Gamma_\psi$ (resp. $Cl(Kind)^\Gamma_\psi$). Finally, we have $\Pi x:A.B = (v/y)(\Pi x:A'.B') \in Cl(Type)^\Gamma_\varphi$ (resp. $Cl(Kind)^\Gamma_\varphi$).

□

Lemma 4.13. *for all (well-formed) contexts* Γ*, adequate valuations* φ *on* $Cl(.)^\Gamma$ *and* $\Pi x:A.B \in \mathcal{I}_\Gamma$ *with* x *fresh for* Γ*,*
$Cl(\Pi x:A.B)^\Gamma_\varphi = \{t$ *s.t. for all* $u \in Cl(A)^{\Gamma[x:A]}_\varphi,\ tu \in Cl(B)^{\Gamma[x:A]}_{\varphi+\langle x,u\rangle}\}$*,*

Proof. By induction on φ.

- If φ is the empty substitution,
  - $\subseteq$ Let $t \in Cl(\Pi x:A.B)^\Gamma_\bullet$ and $u \in Cl(A)^{\Gamma[x:A]}_\bullet$. Then there exists $k \in \mathbb{N}$ such that $t \in Cl^k(\Pi x:A.B)^\Gamma_\bullet$. By induction on k.
    - If $k=0$, then $\Sigma\Gamma \vdash_\mathcal{R} t : \Pi x:A.B$ therefore $\Sigma\Gamma[x:A] \vdash_\mathcal{R} tx : B$ hence $tx \in Cl^0(B)^{\Gamma[x:A]}_\bullet$ and, since x is not free in t, $tu = (u/x)(tx) \in Cl^0(B)^{\Gamma[x:A]}_{\langle x,u\rangle} \subseteq Cl(B)^{\Gamma[x:A]}_{\langle x,u\rangle}$.

- If $k > 0$ then t is a (CR_3') expansion of a term t' in $Cl^{k-1}(\Pi x : A.B)^{\Gamma}_{\bullet}$ hence $t'u \in Cl(B)^{\Gamma[x:A]}_{\langle x,u\rangle}$ by induction hypothesis, and so does tu since it is a (CR_3') expansion of $t'u$ and $Cl(B)^{\Gamma[x:A]}_{\langle x,u\rangle}$ satisfies (CR_3') .

$\supseteq$ Let t be a term such that for all $u \in Cl(A)^{\Gamma[x:A]}_{\bullet}$, $tu \in Cl(B)^{\Gamma[x:A]}_{\langle x,u\rangle}$.

- If t is an abstraction $\lambda y : A.t'$, since $x \in Cl(A)^{\Gamma[x:A]}_{\bullet}$, there exists $k \in \mathbb{N}$ such that $(\lambda y : A.t')x \in Cl^k(B)^{\Gamma[x:A]}_{\bullet}$ and so does $(x/y)t'$ by (CR_2). Hence we can prove that $t \in Cl(\Pi x : A.B)^{\Gamma}_{\bullet}$ by induction on k: if $k = 0$, then $\Sigma\Gamma[x : A] \vdash_{\mathcal{R}} (x/y)t' : B$, hence $\Sigma\Gamma \vdash_{\mathcal{R}} \lambda x : A.(x/y)t' : \Pi x : A.B$ (by case on the last rule used in the former typing judgement) and $\lambda x : A.(x/y)t' \in Cl^0(\Pi x : A.B)^{\Gamma}_{\bullet}$ and so does $\lambda y : A.t'$ by renaming. If $k > 0$, we conclude by induction hypothesis.
- If t is neutral and normal, then so does tx. And since tx is in $Cl(B)^{\Gamma[x:A]}_{\bullet}$, it is in $Cl^0(B)^{\Gamma[x:A]}_{\bullet}$, because ($\mathrm{CR}_3$') expansions only add non-normal proof-terms to a set. Finally, since $\Sigma\Gamma[x : A] \vdash_{\mathcal{R}} tx : B$, we also have $\Sigma\Gamma \vdash_{\mathcal{R}} t : \Pi x : A.B$, and $t \in Cl^0(\Pi x : A.B)^{\Gamma}_{\bullet}$.
- If t is neutral and non-normal, since it is in SN (because $tx \in Cl(B)^{\Gamma[x:A]}_{\langle x,u\rangle} \subseteq SN$), and all its leaves are in the set $\{u \text{ s.t. for all } v \in Cl(A)^{\Gamma[x:A]}_{\bullet},\ uv \in Cl(B)^{\Gamma[x:A]}_{\langle x,v\rangle}\}$ (because this set is stable by $\beta\mathcal{R}$-reduction), we can conclude that $t \in Cl(\Pi x : A.B)^{\Gamma}_{\bullet}$ since $Cl(\Pi x : A.B)^{\Gamma}_{\bullet}$ satisfies (CR_3').

- If $\varphi = \psi + \langle y, v\rangle$,

$\subseteq$ Let $t \in Cl(\Pi x : A.B)^{\Gamma}_{\varphi}$ then there exists $t' \in Cl(\Pi x : A.B)^{\Gamma}_{\psi}$ such that $t = (v/y)t'$. By induction hypothesis, we have $t'x \in Cl(B)^{\Gamma[x:A]}_{\psi}$, hence $tx = (v/y)(t'x) \in Cl(B)^{\Gamma[x:A]}_{\varphi}$ (as x cannot be equal to y) and therefore, for all $u \in Cl(A)^{\Gamma[x:A]}_{\varphi}$, $tu = (u/x)(tx) \in Cl(B)^{\Gamma[x:A]}_{\varphi+\langle x,u\rangle}$ (as x is not free in t).

$\supseteq$ Let t be a term such that for all $u \in Cl(A)^{\Gamma[x:A]}_{\varphi}$, $tu \in Cl(B)^{\Gamma[x:A]}_{\varphi+\langle x,u\rangle}$. Then $tx \in Cl(B)^{\Gamma[x:A]}_{\varphi}$ hence there exists $t'' \in Cl(B)^{\Gamma[x:A]}_{\psi}$ such that $tx = (v/y)t''$. If $t'' = y$ then $tx = v \in Cl(B)^{\Gamma[x:A]}_{\psi}$ therefore for all $u \in Cl(A)^{\Gamma[x:A]}_{\psi}$, $tu = (u/x)v \in Cl(B)^{\Gamma[x:A]}_{\psi+\langle x,u\rangle}$ (as $x \notin FV(t)$). Therefore we have $t \in Cl(\Pi x : A.B)^{\Gamma}_{\psi}$, by induction hypothesis and $t \in Cl(\Pi x : A.B)^{\Gamma}_{\varphi}$ since y is not free in v and

therefore neither in t. Otherwise there exists a term t' such that $t'' = t'x$, then $t = (v/y)t'$ and $t' \in Cl(\Pi x : A.B)^\Gamma_\psi$ by induction hypothesis, hence we have $t \in Cl(\Pi x : A.B)^\Gamma_\varphi$.

□

Lemma 4.14.
For all (well-formed) contexts Γ*, adequate valuations* φ *on* Γ *and* $A, B \in \mathcal{I}_\Gamma$*, if* $A \equiv_{\beta\mathcal{R}} B$*, then* $Cl(A)^\Gamma_\varphi = Cl(B)^\Gamma_\varphi$.

Proof. If $A \equiv_{\beta\mathcal{R}} B$ then $Cl^0(A)^\Gamma = Cl^0(B)^\Gamma$ by rules Conversion and Conversion2. Hence $Cl(A)^\Gamma_\bullet = Cl(B)^\Gamma_\bullet$ and therefore $Cl(A)^\Gamma_\varphi = Cl(B)^\Gamma_\varphi$.

□

Lemma 4.15.
For all contexts Γ*, adequate valuations* φ *on* Γ*,* $t, B \in \Lambda$*, variables* x *fresh for* Γ *and* $A \in \mathcal{I}_{\Gamma[x:B]}$*, if* $\Sigma\Gamma \vdash_\mathcal{R} t : B$*, then* $Cl(A)^{\Gamma[x:B]}_{\varphi+\langle x,\varphi t\rangle} \subseteq Cl(t/x)A)^\Gamma_\varphi$.

Proof. By induction on φ.

- If φ is the empty substitution. Let $u \in Cl(A)^{\Gamma[x:B]}_{\langle x,t\rangle}$ then there exists $u' \in Cl(A)^{\Gamma[x:B]}_\bullet$ such that $u = (t/x)u'$. And there exists $k \in \mathbb{N}$ such that $u' \in Cl^k(A)^{\Gamma[x:B]}_\bullet$. By induction on k.
  - If $k = 0$, then $\Sigma\Gamma[x : B] \vdash_\mathcal{R} u' : A$. Therefore we have $\Sigma\Gamma[x : B] \vdash_\mathcal{R} (t/x)u' : (t/x)A$ by lemma 4.2. Moreover we have $\Sigma\Gamma \vdash_\mathcal{R} (t/x)u' : (t/x)A$ since x is not free neither in $(t/x)u'$, nor in $(t/x)A$. Finally $u = (t/x)u' \in Cl^0((t/x)A)^\Gamma_\bullet$.
  - If $k > 0$ then u' is a (CR_3') expansion of a term u'' in $Cl^{k-1}(A)^{\Gamma[x:B]}_\bullet$. Therefore $(t/x)u'' \in Cl((t/x)A)^\Gamma_\bullet$ by induction hypothesis. And since $(t/x)u'$ is also a (CR_3') expansion of $(t/x)u''$ (because the neutral terms considered in a (CR_3') expansion are non-normal), we finally have $u = (t/x)u' \in Cl((t/x)A)^\Gamma_\bullet$ since it satisfies (CR_3').

- If $\varphi = \psi+\langle y, v\rangle$, let $u \in Cl(A)^{\Gamma[x:B]}_{\varphi+\langle x,\varphi t\rangle}$, then there exists $u' \in Cl(A)^{\Gamma[x:B]}_\psi$ such that $u = (\varphi t/x)(v/y)u'$ then $u = (v/y)(\varphi t/x)u'$ since x is not free in v and y is yet substituted by v in φ. Since $\varphi = \psi + \langle y, v\rangle$, we have $u = (v/y)(\psi t/x)u'$, with $(\psi t/x)u' \in Cl((t/x)A)^\Gamma_\psi$ by induction hypothesis, hence $u = (v/y)(\psi t/x)u' \in Cl((t/x)A)^\Gamma_{\psi+\langle y,v\rangle} = Cl((t/x)A)^\Gamma_\varphi$.

□

Using those previous lemmas, we are finally able to prove that $Cl(.)$ actually defines a pre-model when the theory considered in λΠ-calculus modulo is strongly normalizing.

Proposition 4.3 (Completeness)**.**
If a theory is strongly normalizing, then $Cl(.)^{\cdot}$ *is a pre-model of this theory.*

Proof. By lemmas 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 and 4.15.

□

And we finally can assert the following theorem: having a pre-model is a sound and complete semantics for strong normalization of theories expressed in λΠ-calculus modulo.

Theorem 4.1 (Soundness and Completeness)**.**
A theory in λΠ*-modulo is strongly normalizing iff it has a pre-model.*

Proof. By propositions 4.2 and 4.3.

□

4.4 Conclusion

In this chapter, we have given a sound and complete semantics, based on our work of chapter 3, for strong normalization of theories expressed in the logical framework of λΠ-calculus modulo. The soundness part is usual as many proofs of strong normalization of various logical frameworks (with or without dependent types) use reducibility candidates (with slightly different definitions from the one we used). But the completeness part is innovative and there is, up to our knowledge, no other work on complete semantics for strong normalization based on the notion of reducibility candidates in logical frameworks with dependent types.

As it has been done for deduction modulo, we have defined a notion of pre-model such that having a pre-model is a sound semantics for strongly normalizing theories in λΠ-calculus modulo. Moreover, we have reused the ideas of chapter 3, in order to obtain an also complete semantics for strong normalization. The completeness part comes from the use of our modified property (CR_3'), instead of the usual (CR_3) property and from a refinement of the method for interpreting dependent types using valuations. This work can be seen as a first step in the search of a definition of algebras for λΠ-calculus modulo on which we could build models sound (and complete) for consistency and for strong normalization (as it has been done for deduction modulo). This definition of algebras may emerge from our definition of

LDTVAs of section 3.1.4, but it has not been studied in the present work. Notice that this definition of algebras for $\lambda\Pi$-calculus modulo would not depend on the language anymore, since there is no universal quantifier in $\lambda\Pi$-calculus modulo. But we could reuse the idea of dependent intersection of the interpretation of the universal quantifier to propose a dependent interpretation of the product Π, as we have done in our definition of pre-models.

Apart from the algebraic point of view, we can see the completeness proof of this chapter as the one of chapter 3 where a cut has been eliminated: instead of defining a notion of well-typed pre-model (the equivalent of the notion of $\mathcal{C}_{\equiv}$-valued model) using an equivalent of properties ($\mathrm{CR}_{1_{\equiv}}$),($\mathrm{CR}_{2_{\equiv}}$) and ($\mathrm{CR}_{3_{\equiv}}$) and defining a function which maps all well-typed pre-models to a pre-model, we have only defined this function, when the theory is strongly normalizing, on a precise well-typed pre-model (based on well-typed terms, as in section 3.2.4), and only proved that this precise well-typed pre-model was led to a pre-model by our function.

As a last remark on this chapter, we have seen that the theories which are strongly normalizing when expressed in deduction modulo and in $\lambda\Pi$-calculus modulo, are not the same ones. For example the theory defined by the rewrite rule $P \to P$, with P an atomic proposition or a variable declared of type $Type$ in the signature, is strongly normalizing when expressed in deduction modulo but not when expressed in $\lambda\Pi$-calculus modulo. This induces an interesting question: how to define a criterion for strongly normalizing theories expressed in deduction modulo that are also strongly normalizing when expressed in in $\lambda\Pi$-calculus modulo. The termination of the rewrite system is a necessary condition because of the previous example, but it may be not sufficient.

In the next chapter, we shall introduce another extension of $\lambda\Pi$-calculus called (functional) Pure Type Systems, and we shall see how those functional Pure Type Systems can be embedded as a theory in $\lambda\Pi$-calculus modulo.

5

Embedding functional Pure Type Systems in $\lambda\Pi$-calculus modulo

Context

We have detailed in the previous chapters the technique of expressing theories via a set of rewrite rules concerning propositions in natural deduction, defining the deduction modulo, or via a set of rewrite rules concerning terms in the $\lambda\Pi$-calculus, defining the $\lambda\Pi$-calculus modulo. In order to extend the $\lambda\Pi$-calculus to express stronger theories than minimal predicate logic, another solution is to add typing rules to the $\lambda\Pi$-calculus. For example, by adding the possibility to type polymorphic products and type constructors products to the $\lambda\Pi$-calculus, we obtain the *calculus of constructions* ([12]). More generally, by authorizing an arbitrary set of sorts (not only $Type$ and $Kind$), different typing rules concerning sorts and products allowed, we obtain the notion of *Pure Types Systems* (PTS) [5],[50]. In [14] G. Dowek has defined an embedding as a theory in $\lambda\Pi$-calculus modulo of those PTS in which uniqueness of types is satisfied, called *functional* PTS. This embedding is inspired both by the expression of simple type theory in Deduction modulo and by the mechanisms of universes *à la* Tarski [40] of Intuitionistic type theory.

Contributions

We first provide a discussion on how to prove strong normalization of those theories expressed in $\lambda\Pi$-calculus modulo, and obtain, as a corollary of soundness of this embedding, a proof of strong normalization of the embedded functional PTS. We then prove in this chapter, the conservativity of this embedding. This property was not so simple to prove. Indeed, using the $\lambda\Pi$-calculus to simulate PTS is very convenient since we can use the depen-

dent types of λΠ and simulate therefore easily the behaviour of terms in PTS. The obverse of that convenience is that the theory obtained in λΠ-modulo can be more expressive than P, in the sense that there exists terms in this theory which do not correspond to translations of terms of the embedded PTS. We had therefore to be very precise concerning the form of statement of conservativity we wanted to prove. This conservativity property we have proved is similar to that of the Curry-de Bruijn-Howard correspondence, relating terms in normal form in the theory expressed in λΠ-modulo to terms in the embedded PTS. In order to prove this conservativity property, we need to prove confluence of the theories in λΠ modulo which correspond to embeddings of functional PTS.

Outline

We first define formally PTS, and functional Pure Type Systems. We then define, for each functional PTS P, a theory in λΠ modulo called $\lambda\Pi_P$ given by specific signature and set of rewrite rules. We then define two translations from terms of P to terms of $\lambda\Pi_p$. One *as a term* and one *as a type*. We point out the soundness of our embedding by proving that the translation as a term of a well-typed term of P is typed by the translation as a type of the former type. In order to prove the conservativity of this embedding, we first prove the confluence of all theories $\lambda\Pi_P$ for P a functional PTS. We then prove a first conservativity property: the fact that all terms in a certain form called weak η-long normal form and typed by the translation as a type of a type of P, are equivalent to a term of P well-typed by this same type. We finally get rid of this condition about weak η-long forms by proving that the weak η-long form of a well-typed term has the same type as this term.

5.1 The Pure Type Systems

As we have seen, the λΠ-*calculus* is a dependently typed lambda-calculus that allows to express proofs of minimal predicate logic through the Brouwer-Heyting-Kolmogorov interpretation and the Curry-de Bruijn-Howard correspondence. It can be extended in several ways to express proofs of some theory. A first solution is to express the theory in Deduction modulo [17, 20], *i.e.* to orient the axioms as rewrite rules and to extend the λΠ-calculus to express proofs in Deduction modulo [6]. We get this way the λΠ-*calculus modulo* we have studied in the previous chapter. This idea of extending the dependently typed lambda-calculus with rewrite rules is also that of Intuitionistic type theory used as a logical framework [42].

A second way to extend the λΠ-calculus is to add typing rules, in particular to allow polymorphic typing and type constructors. We get this way the *Calculus of Constructions* [12] that allows to express proofs of simple type

theory and more generally the *Pure Type Systems* [5, 50, 4]. These two kinds of extensions of the λΠ-calculus are somewhat redundant. For instance, simple type theory can be expressed in deduction modulo [18], hence the proofs of this theory can be expressed in the λΠ-calculus modulo. But they can also be expressed in the Calculus of Constructions. This suggests to relate and compare these two ways to extend the λΠ-calculus.

We show in this chapter that all functional Pure Type Systems can be embedded in the λΠ-calculus modulo using an appropriate rewrite system. This rewrite system is inspired both by the expression of simple type theory in Deduction modulo and by the mechanisms of universes *à la* Tarski [40] of Intuitionistic type theory. In particular, this work extends Palmgren's construction of an impredicative universe in type theory [44].

There are several ways to extend the functional interpretation of proofs to simple type theory. The first is to use the fact that simple type theory can be expressed in Deduction modulo with rewrite rules only [18]. Thus, the proofs of simple type theory can be expressed in the λΠ-calculus modulo, and even in the $\lambda\Pi^{-}$-calculus modulo. The second is to extend the λΠ-calculus by adding the following typing rules, allowing for instance the construction of the type $\Pi P : Type\ (P \Rightarrow P)$.

$$\frac{\Gamma \vdash A : Kind \quad \Gamma[x : A] \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}\ \textbf{Product3}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma[x : A] \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}\ \textbf{Product4}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma[x : A] \vdash B : Type \quad \Gamma[x : A] \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction3}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma[x : A] \vdash B : Kind \quad \Gamma[x : A] \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction4}$$

We obtain the *Calculus of Constructions* [12].

The rules of the simply typed λ-calculus, the λΠ-calculus and of the Calculus of Constructions can be presented in a schematic way as follows.

Definition 5.1 (Pure type system)**.**
A Pure Type System *P is defined by a set S, whose elements are called* sorts*, a subset A of $S \times S$, whose elements are called* axioms *and a subset R of $S \times S \times S$, whose elements are called* rules*. The typing rules of P are*

$$\frac{}{[\,]\ \textit{well-formed}}\ \textbf{Empty}$$

$$\frac{\Gamma \vdash A : s}{\Gamma[x : A]\ \ \textit{well-formed}}\ \textbf{Declaration}\ s \in S\ \textit{and}\ x\ \textit{not in}\ \Gamma$$

$$\frac{\Gamma \ \textit{well-formed}}{\Gamma \vdash s_1 : s_2}\ \textbf{Sort}\ \langle s_1, s_2\rangle \in A$$

$$\frac{\Gamma \ \textit{well-formed} \quad x : A \in \Gamma}{\Gamma \vdash x : A}\ \textbf{Variable}$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma[x : A] \vdash B : s_2}{\Gamma \vdash \Pi x : A\ B : s_3}\ \textbf{Product}\ \langle s_1, s_2, s_3\rangle \in R$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma[x : A] \vdash B : s_2 \quad \Gamma[x : A] \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction}\ \langle s_1, s_2, s_3\rangle \in R$$

$$\frac{\Gamma \vdash t : \Pi x : A\ B \quad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : (u/x)B}\ \textbf{Application}$$

$$\frac{\Gamma \vdash A : s \quad \Gamma \vdash B : s \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}\ \textbf{Conversion}\ s \in S \quad A \equiv_\beta B$$

Example 5.1. *The simply typed λ-calculus is the system defined by the sorts $Type$ and $Kind$, the axiom $\langle Type, Kind\rangle$ and the rule $\langle Type, Type, Type\rangle$. The λΠ-calculus is the system defined by the same sorts and axiom and the rules $\langle Type, Type, Type\rangle$ and $\langle Type, Kind, Kind\rangle$. The Calculus of Constructions is the system defined by the same sorts and axiom and the rules $\langle Type, Type, Type\rangle$, $\langle Type, Kind, Kind\rangle$, $\langle Kind, Type, Type\rangle$ and $\langle Kind, Kind, Kind\rangle$. Other examples of Pure Type Systems are Girard's systems F and Fω.*

In all Pure Type Systems, types are preserved under reduction and the β-reduction relation is confluent. It terminates in some systems, such as the λΠ-calculus, the Calculus of Constructions, the system F and the system Fω. Uniqueness of types is lost in general, but it holds for the λΠ-calculus, the Calculus of Constructions, the system F and the system Fω, and more generally for all functional Pure Type Systems.

Definition 5.2 (Functional Pure Type Systems)**.**
A type system is said to be functional *if*

$$\langle s_1, s_2\rangle \in A \textit{ and } \langle s_1, s_3\rangle \in A \textit{ implies } s_2 = s_3$$

$$\langle s_1, s_2, s_3\rangle \in R \textit{ and } \langle s_1, s_2, s_4\rangle \in R \textit{ implies } s_3 = s_4$$

5.2 Embedding functional Pure Type Systems in the λΠ-calculus modulo

We have seen that the λΠ-calculus modulo and the Pure Type Systems are two extensions of the λΠ-calculus. At a first glance, they seem quite different since the latter adds more typing rules to the λΠ-calculus, while the former

adds more computation rules. But they both allow to express proofs of simple type theory.

We show in this section that functional Pure Type Systems can, in fact, be embedded in the $\lambda\Pi$-calculus modulo with an appropriate rewrite system.

5.2.1 Definition

Consider a functional Pure Type System $P = \langle S, A, R\rangle$. We build the following context and rewrite system.
The context Σ_P contains, for each sort s, two variables

$$U_s : Type \quad \textit{and} \quad \varepsilon_s : U_s \Rightarrow Type$$

for each axiom $\langle s_1, s_2\rangle$, a variable

$$\dot{s_1} : U_{s_2}$$

and for each rule $\langle s_1, s_2, s_3\rangle$, a variable

$$\dot{\Pi}_{\langle s_1,s_2,s_3\rangle} : \Pi X : U_{s_1}\ (((\varepsilon_{s_1}\ X) \Rightarrow U_{s_2}) \Rightarrow U_{s_3})$$

The type U_s is called the *universe* of s and the symbol ε_s the *decoding function* of s.
The rewrite rules are

$$\varepsilon_{s_2}(\dot{s}_1) \longrightarrow U_{s_1}$$

in the empty context and with the type $Type$, and

$$\varepsilon_{s_3}(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ X\ Y) \longrightarrow \Pi x : (\varepsilon_{s_1}\ X)\ (\varepsilon_{s_2}\ (Y\ x))$$

in the context $X : U_{s_1}, Y : (\varepsilon_{s_1}\ X) \Rightarrow U_{s_2}$ and with the type $Type$.

These rules are called the *universe-reduction rules*, we write $\equiv_P$ for the equivalence relation generated by these rules and the rule β and we call the $\lambda\Pi_P$-calculus the $\lambda\Pi$-calculus modulo these rewrite rules and the rule β. To ease notations, in the $\lambda\Pi_P$-calculus, we do not recall the context Σ_P in each sequent and write $\Gamma \vdash t : T$ for $\Sigma_P\Gamma \vdash t : T$, and we note $\equiv$ for $\equiv_P$ when there is no ambiguity.

Example 5.2. *The embedding of the Calculus of Constructions is defined by the context*

$$\dot{Type} : U_{Kind} \qquad U_{Type} : Type \qquad U_{Kind} : Type$$

$$\varepsilon_{Type} : U_{Type} \Rightarrow Type \qquad \varepsilon_{Kind} : U_{Kind} \Rightarrow Type$$

$$\dot{\Pi}_{\langle Type,Type,Type\rangle} : \Pi X : U_{Type}\ (((\varepsilon_{Type}\ X) \Rightarrow U_{Type}) \Rightarrow U_{Type})$$

$$\dot{\Pi}_{\langle Type,Kind,Kind\rangle} : \Pi X : U_{Type}\ (((\varepsilon_{Type}\ X) \Rightarrow U_{Kind}) \Rightarrow U_{Kind})$$
$$\dot{\Pi}_{\langle Kind,Type,Type\rangle} : \Pi X : U_{Kind}\ (((\varepsilon_{Kind}\ X) \Rightarrow U_{Type}) \Rightarrow U_{Type})$$
$$\dot{\Pi}_{\langle Kind,Kind,Kind\rangle} : \Pi X : U_{Kind}\ (((\varepsilon_{Kind}\ X) \Rightarrow U_{Kind}) \Rightarrow U_{Kind})$$

and the rules

$$\varepsilon_{Kind}(\dot{Type}) \longrightarrow U_{Type}$$
$$\varepsilon_{Type}(\dot{\Pi}_{\langle Type,Type,Type\rangle}\ X\ Y) \longrightarrow \Pi x : (\varepsilon_{Type}\ X)\ (\varepsilon_{Type}\ (Y\ x))$$
$$\varepsilon_{Kind}(\dot{\Pi}_{\langle Type,Kind,Kind\rangle}\ X\ Y) \longrightarrow \Pi x : (\varepsilon_{Type}\ X)\ (\varepsilon_{Kind}\ (Y\ x))$$
$$\varepsilon_{Type}(\dot{\Pi}_{\langle Kind,Type,Type\rangle}\ X\ Y) \longrightarrow \Pi x : (\varepsilon_{Kind}\ X)\ (\varepsilon_{Type}\ (Y\ x))$$
$$\varepsilon_{Kind}(\dot{\Pi}_{\langle Kind,Kind,Kind\rangle}\ X\ Y) \longrightarrow \Pi x : (\varepsilon_{Kind}\ X)\ (\varepsilon_{Kind}\ (Y\ x))$$

We shall use two translations from terms of a functionnal pure type system to the associated $\lambda\Pi$-calculus modulo. As we have seen for the variables used to represent a sort of the functionnal pure type system, we distinguish whether this sort is considered as a term or as a type. This is useful not to confound those two different cases. Hence we shall define a translation *as a term* and a translation *as a type* of terms of the functionnal pure type system. We can notice that the translation *as a type* use the so-called *decoding functions* and the translation *as a term*.

Definition 5.3 (Translation as a term)**.**
Let Γ be a context in a functional Pure Type System P and t a term well-typed in Γ, we defined the translation $|t|$ of t in Γ, that is a term in $\lambda\Pi_P$, as follows

- $|x| = x$,
- $|s| = \dot{s}$,
- $|\Pi x : A\ B| = \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ |A|\ (\lambda x : (\varepsilon_{s_1}\ |A|)\ |B|)$, *where s_1 is the type of A, s_2 is the type of B and s_3 the type of $\Pi x : A\ B$,*
- $|\lambda x : A\ t| = \lambda x : (\varepsilon_s\ |A|)\ |t|$,
- $|t\ u| = |t|\ |u|$.

Definition 5.4 (Translation as a type)**.**
Consider a term A of type s for some sort s. The translation of A as a type *is*

$$\|A\| = \varepsilon_s\ |A|.$$

Note that if A is a well-typed sort s' then

$$\|s'\| = \varepsilon_s\ \dot{s}' \equiv_P U_{s'}.$$

We extend this definition to non well-typed sorts, such as the sort Kind in the Calculus of Constructions, by

$$\|s'\| = U_{s'}$$

The translation of a well formed context is defined by

$$\|[\,]\| = [\,] \quad \textit{and} \quad \|\Gamma[x:A]\| = \|\Gamma\|[x:\|A\|]$$

5.2.2 Soundness of the embedding

Let us first prove some useful lemmas for proving the soundness of the embedding we have defined in the previous subsection. The first one expresses the fact that the translation (as a term or as a type) of a substitution is the substitution of the associated translated terms.

Lemma 5.1.
For all variables x and terms t, u such that t and $(u/x)t$ are well-typed, we have $|(u/x)t| = (|u|/x)|t|$ and $\|(u/x)t\| = (|u|/x)\|t\|$.

Proof. • Let us first prove that $|(u/x)t| = (|u|/x)|t|$ by induction of the structure of t.

– If t is a variable y, then either $y = x$ and in this case, $|(u/x)t| = |u| = (|u|/x)|t|$ since $|t| = |x| = x$. Or $y \neq x$, and in this case, $|(u/x)t| = |y| = y = (|u|/x)|y|$.

– If t is a sort s, then $(u/x)s = s$ therefore $|(u/x)s| = |s| = \dot{s} = (|u|/x)\dot{s}$ since $\dot{s} \neq x$.

– If t is a product $\Pi y : A.B$ corresponding to a rule $\langle s_1, s_2, s_3\rangle$, then

$$\begin{aligned}
|(u/y)(\Pi x : A.B)| &= |\Pi x : (u/y)A.(u/y)B| \\
&= \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ |(u/y)A|\ (\lambda x : (\varepsilon_{s_1}\ |(u/y)A|)\ |(u/y)B|) \\
&= \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ (|u|/y)|A|\ (\lambda x : (\varepsilon_{s_1}\ (|u|/y)|A|)\ (|u|/y)|B|) \\
&\qquad \text{(by induction hypothesis)} \\
&= (|u|/y)(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ (|A|\ (\lambda x : (\varepsilon_{s_1}\ |A|)\ (|B|)) \\
&= (|u|/y)|\Pi x : A.B|
\end{aligned}$$

– If t is an abstraction $\lambda y : A.v$ with s the sort of A, then

$$\begin{aligned}
|(u/y)(\lambda x : A.v)| &= |\lambda x : (u/y)A.(u/y)v| \\
&= \lambda x : (\varepsilon_s|(u/y)A|).|(u/y)v| \\
&= \lambda x : (\varepsilon_s(|u|/y)|A|).(|u|/y)|v| \\
&\qquad \text{(by induction hypothesis)} \\
&= (|u|/y)\lambda x : (\varepsilon_s|A|).|v| \\
&= (|u|/y)|\lambda x : A.v|
\end{aligned}$$

(notice that A and $(u/y)A$ have the same sort using an analogous property as in lemma 4.2)

– If t is an application $t_1\ t_2$, then $(|u|/x)|t| = (|u|/x)(|t_1|\ |t_2|) = (|u|/x)|t_1|\ (|u|/x)|t_2| = |(u/x)t_1|\ |(u/x)t_2|$ by induction hypothesis and it is therefore equal to $|(u/x)t_1\ (u/x)t_2| = |(u/x)(t_1\ t_2)|$.

- If $\|t\|$ is defined then either t is an ill-typed sort s and in this case $\|(u/x)s\| = \|s\| = U_s = (|u|/x)U_s$. Or $\|t\| = \varepsilon_s t$ with s the sort of t. And we have $(|u|/x)\|t\| = (|u|/x)(\varepsilon|t|) = \varepsilon\ (|u|/x)|t| = \varepsilon\ |(u/x)t| = \|(u/x)t\|$, using the first point.

□

The following lemma states that if a term β-reduces to another one then so do their translations as a term.

Lemma 5.2.
For all (well-typed) terms t, u, if $t \longrightarrow_\beta u$ then $|t| \longrightarrow_\beta |u|$.

Proof. We can notice that an abstraction of a PTS P is translated (as a term) as an abstraction of the associated $\lambda\Pi_P$-calculus modulo, and that the translation as a term of an application is the application of the translations as a term of the terms of this application, therefore a β-redex of a PTS P is translated as a β-redex of the associated $\lambda\Pi_P$-calculus modulo.

□

The following lemma states that the translation as a type of a product is equivalent to the product of the translations of the arguments of the former product. This reflects the fact that we exploit the similarities of the $\lambda\Pi$-calculus modulo and Pure Type Systems, and use the actual product of $\lambda\Pi$-calculus modulo to represent the products of the embedded Pure Type System.

Lemma 5.3.
For all well-formed products $\Pi x : A\ B$ of a PTS P,
we have $\|\Pi x : A.\ B\| \equiv_P \Pi x : \|A\|.\ \|B\|$

Proof. Let s_1 be the type of A, s_2 that of B and s_3 that of $\Pi x : A\ B$. We have

$$\begin{array}{rcl}
\|\Pi x : A.\ B\| & = & \varepsilon_{s_3}\ |\Pi x : A.\ B| \\
& = & \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ |A|\ (\lambda x : (\varepsilon_{s_1}\ |A|)\ |B|)) \\
& \equiv_P & \Pi x : (\varepsilon_{s_1}\ |A|).\ (\varepsilon_{s_2}\ ((\lambda x : (\varepsilon_{s_1}\ |A|)\ |B|)\ x)) \\
& \equiv_P & \Pi x : (\varepsilon_{s_1}\ |A|).\ (\varepsilon_{s_2}\ |B|) \\
& \equiv_P & \Pi x : (\varepsilon_{s_1}\ |A|).\ (\varepsilon_{s_2}\ |B|) \\
& = & \Pi x : \|A\|.\ \|B\|
\end{array}$$

□

Example 5.3. *In the Calculus of Constructions, the translation as a type of* $\Pi X : Type\ (X \Rightarrow X)$ *is* $\Pi X : U_{Type}\ ((\varepsilon_{Type}\ X) \Rightarrow (\varepsilon_{Type}\ X))$. *The translation as a term of* $\lambda X : Type\ \lambda x : X\ x$ *is the term* $\lambda X : U_{Type}\ \lambda x : (\varepsilon_{Type} X)\ x$. *Notice that the former is the type of the latter. The generalization of this remark is the following proposition.*

Proposition 5.1 (Soundness)**.**
For all contexts Γ*, and terms* t, B *of a functional PTS* P*,*
If $\Gamma \vdash t : B$ *in* P *then* $\|\Gamma\| \vdash |t| : \|B\|$ *in* $\lambda\Pi_P$.

Proof. By induction on the structure of t.

- If t is a variable x, then $|x| = x$ and x is declared of type $\|B\|$ in $\|\Gamma\|$.
- If $t = s_1$ then $B = s_2$ (where $\langle s_1, s_2\rangle$ is an axiom), we have $\dot{s_1} : U_{s_2} = \|s_2\|$.
- If $t = \Pi x : C\ D$, let s_1 be the type of C, s_2 that of D and s_3 that of t. By induction hypothesis, we have

  $$\|\Gamma\| \vdash |C| : U_{s_1} \quad and \quad \|\Gamma\|, x : \|C\| \vdash |D| : U_{s_2}$$

  i.e.

  $$\|\Gamma\|, x : (\varepsilon_{s_1}\ |C|) \vdash |D| : U_{s_2}$$

  Thus

  $$\|\Gamma\| \vdash (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ |C|\ \lambda x : (\varepsilon_{s_1}\ |C|)\ |D|) : U_{s_3}$$

  i.e.

  $$\|\Gamma\| \vdash |\Pi x : C\ D| : \|s_3\|$$

- If $t = \lambda x : C\ u$, then we have

  $$\Gamma, x : C \vdash u : D$$

  and $B = \Pi x : C\ D$. By induction hypothesis, we have

  $$\|\Gamma\|, x : \|C\| \vdash |u| : \|D\|$$

  i.e.

  $$\|\Gamma\|, x : (\varepsilon_{s_1}\ |C|) \vdash |u| : \|D\| \ then\ \|\Gamma\| \vdash \lambda x : (\varepsilon_{s_1}\ |C|)\ |u| : \Pi x : \|C\|\ \|D\|$$

  i.e.

  $$\|\Gamma\| \vdash |t| : \|\Pi x : C\ D\|$$

- If $t = u\ v$, then we have

$$\Gamma \vdash u : \Pi x : C\ D, \quad \Gamma \vdash v : C$$

and $B = (v/x)D$. By induction hypothesis, we get

$$\|\Gamma\| \vdash |u| : \|\Pi x : C\ D\| = \Pi x : \|C\|\ \|D\| \quad \textit{and} \quad \|\Gamma\| \vdash |v| : \|C\|$$

Thus

$$\|\Gamma\| \vdash |t| : (|v|/x)\|D\| = \|(v/x)D\|$$

□

This soundness property about typing of our embedding allows us to prove another soundness property about strong normalization: for all functional PTS P, the strong normalization of P is entailed by the strong normalization of $\lambda\Pi_P$.

Proposition 5.2.
If $\lambda\Pi_P$ is strongly normalizing then P is strongly normalizing.

Proof. Let t_1 be a well-typed term in P and $t_1, t_2, ...$ be a reduction sequence of t_1 in P. By Proposition 5.1, the term $|t_1|$ is well-typed in $\lambda\Pi_P$ and, by lemma 5.2, $|t_1|, |t_2|, ...$ is a reduction sequence of $|t_1|$ in $\lambda\Pi_P$. Hence it is finite.

□

5.3 How to prove strong normalization of the embedded theories in $\lambda\Pi$-calculus modulo

We propose in this section a discussion on how to prove strong normalization of such a theory obtained as the embedding, defined in section 5.2.1, of a functional PTS in $\lambda\Pi$-calculus modulo. As a corollary of soundness of this embedding, proving strong normalization of such a theory in $\lambda\Pi$-calculus modulo also provides a proof of strong normalization of the embedded Pure Types System. For example, proving strong normalization of the theory in $\lambda\Pi$-calculus modulo defined in example 5.2, would provide another proof of strong normalization of the Calculus of Constructions. Now raises the question of which method should we use to prove strong normalization of such theories. Can we use the notion of pre-model we have defined in section 4.3, or should we use a more usual technique as the one defined in [26] ?

We have seen in section 4.3.3 how to build a pre-model for two very simple theories, the empty theory and the theory defined by the single rewrite rule

$P \longrightarrow \Pi x : Q.Q$ with signature $[P : Type,\ Q : Type]$. But building a pre-model for stronger theories can be more difficult. Let us analyze the case of the embedding in $\lambda\Pi$ modulo of the Calculus of Constructions: $\lambda\Pi_{CC}$.

Expressing polymorphic types and type constructors by the meaning of rewrite rules in $\lambda\Pi$-calculus modulo instead of rules of the Calculus of Constructions does not seem to simplify the method to prove strong normalization of this theory. We conjecture that we can prove strong normalization of $\lambda\Pi_{CC}$ by adapting the method defined in [26] in order to interpret $\mathcal{R}$-equivalent types by the same set of reducibility candidates. But it seems difficult, because of the presence of polymorphic types and type constructors, to avoid using a technique which is not directly reusable with our definition of pre-model of section 4.3.

This technique consists in interpreting products which correspond to polymorphism (rule $\langle Kind, Type, Type\rangle$) and types construction (rule $\langle Kind, Kind, Kind\rangle$) by the intersection of a set of reducibility candidates. In [26], a valuation ξ is a function which associates sets of terms to variables, and the interpretation is a function which associates to a type and a valuation, a set of proof-terms. In order to interpret polymorphic and type constructors products, Geuvers first defines for each term A of type $Kind$, the set-interpretation of A as the set of sets of proof-terms $\mathcal{V}(A)$. It allows to define, when A is a term of type $Kind$, the interpretation $[\![\Pi x : A.B]\!]_\xi$ of a product $\Pi x : A.B$ and a valuation ξ as the set of terms t which map each element of $[\![A]\!]_\xi$ to a term which belongs to every $[\![B]\!]_{\xi+\langle x,a\rangle}$ for $a \in \mathcal{V}(A)$. This technique is similar to the interpretation of universal quantifiers in the definition of pre-models for deduction modulo of section 2.3.2. It allows to avoid the circularity which appears when trying to interpret polymorphic and types constructor products using the dependent intersection we have defined in property (c_4) of pre-models for $\lambda\Pi$-calculus modulo in section 4.3: we cannot define inductively the interpretation of $\Pi x : A.B$ from all interpretations of $(u/x)B$ for each u in the interpretation of A since the size of $(u/x)B$ is strictly greater than the size of $\Pi x : A.B$ when u contains products and x appears free in B. Notice that this problem does not appear in deduction modulo, in the case of the interpretation of the universal quantifier as a dependent intersection, since terms cannot contain types in deduction modulo therefore for all variables x, terms t and propositions A, the size of $\forall x.A$ is strictly greater than the size of $(t/x)A$.

But this interpretation we would obtain by adapting the method of [26] does not seem, at a first glance, to satisfy the property (c_4) of our definition of pre-models for $\lambda\Pi$-calculus modulo. We might be able to use this technique of (non-dependent) intersection anyway, by using the fact that polymorphic and types constructor products are represented by variables of the signature in $\lambda\Pi$-calculus modulo, which may not bring as many problems if they

are not reduced to the products associated by the rewrite rules. It will be investigated in future work.

To summarize, proofs of strong normalization for $\lambda\Pi_{CC}$ cannot reuse directly neither the technique defined in [26], nor the one defined in section 4.3.3. But if we suppose that we can adapt the technique of [26] for $\lambda\Pi_{CC}$, then this system is strongly normalizing and, using the completeness theorem, it has a pre-model. The link between the model built using the method of [25] and the pre-model given by the completeness theorem still needs to be investigated.

5.4 Conservativity of the embedding

In this section, we prove conservativity of our embbeding of functional PTS in $\lambda\Pi$-calculus modulo. We shall see that this conservativity is not as strong as the soundness we have proved: there exists well-typed terms in $\lambda\Pi_P$-calculus modulo which are not translations of well-typed terms of P (think about the embedding of the simply-typed λ-calculus which is not polymorphic, whereas $\lambda\Pi$-calculus modulo is). We shall see that the notion of conservativity property we shall prove is similar to that of the Curry-de Bruijn-Howard correspondence: if a type is inhabited by a normal term in $\lambda\Pi_P$, then this normal term is the translation of a well-typed term of P. We need, for that purpose, to prove first confluence in $\lambda\Pi_P$-calculus modulo, for any functional PTS P.

5.4.1 Confluence of $\lambda\Pi_P$-calculus modulo

We prove in this subsection that for any functional Pure Type System P, the system $\lambda\Pi_P$ is confluent. Like that of pure λ-calculus, the reduction relation of $\lambda\Pi_P$ is not strongly confluent: the term $M = (\lambda x\ (x\ x))\ ((\lambda y\ y)\ z)$ has two one-step reducts: $N_1 = (\lambda x\ (x\ x))\ z$ and $N_2 = ((\lambda y\ y)\ z)\ ((\lambda y\ y)\ z)$ and these two terms have no common one-step reduct. Thus, we introduce another reduction relation ($\twoheadrightarrow\!\!\!\!\!+$) that can reduce, in one step, none to all the $\beta\mathcal{R}$-redices that appears in a term, that is strongly confluent and such that $\twoheadrightarrow\!\!\!\!\!+^*\ = \longrightarrow^*$. Then, from the confluence of the relation $\twoheadrightarrow\!\!\!\!\!+$, we shall deduce that of the relation $\longrightarrow$.

Definition 5.5 (Parallel reduction)**.**
The parallel reduction ($\twoheadrightarrow\!\!\!\!\!+$) *in* $\lambda\Pi_P$*, is the smallest relation on terms which verifies the following rules:*

$$\frac{}{M \twoheadrightarrow\!\!\!\!\!+ M}\ (\alpha)$$

$$\frac{}{\varepsilon_{s_2}\ \dot{s_1} \twoheadrightarrow U_{s_1}}\ (\beta) \quad \langle s_1, s_2\rangle \in \mathcal{A}$$

$$\frac{A \twoheadrightarrow A' \quad M \twoheadrightarrow M'}{\lambda x : A\ M \twoheadrightarrow \lambda x : A'\ M'}\ (\gamma)$$

$$\frac{A \twoheadrightarrow A' \quad B \twoheadrightarrow B'}{\Pi x : A\ B \twoheadrightarrow \Pi x : A'\ B'}\ (\delta)$$

$$\frac{M \twoheadrightarrow M' \quad N \twoheadrightarrow N'}{M\ N \twoheadrightarrow M'\ N'}\ (\theta_1)$$

$$\frac{M \twoheadrightarrow M' \quad N \twoheadrightarrow N'}{(\lambda x : A\ M)\ N \twoheadrightarrow (N'/x)M'}\ (\theta_2)$$

$$\frac{A \twoheadrightarrow A' \qquad B \twoheadrightarrow B'}{\varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B) \twoheadrightarrow \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B')}\ (\eta_1) \quad \langle s_1, s_2, s_3\rangle \in \mathcal{R}$$

$$\frac{A \twoheadrightarrow A' \qquad B \twoheadrightarrow B'}{\varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B) \twoheadrightarrow \Pi x : (\varepsilon_{s_1}\ A')\ (\varepsilon_{s_2}(B'\ x))}\ (\eta_2) \quad \langle s_1, s_2, s_3\rangle \in \mathcal{R}$$

Then we prove that if two terms are parallel reducts of two other terms, then the substitution of one parallel reduct by the other parallel reduct is also a parallel reduct of the substitution on the associated term by the other associated one.

Lemma 5.4. *For all terms M, M', N, N' of $\lambda\Pi_P$,*
if $M \twoheadrightarrow M'$ and $N \twoheadrightarrow N'$, then $(N/x)M \twoheadrightarrow (N'/x)M'$.

Proof. By induction on M.

- if M is a variable,
  - ⋆ if $M = x$, then $M' = M = x$
    (because $M \twoheadrightarrow M'$ and the only rule we can apply is (α)).
    Therefore, $(N/x)M = N \twoheadrightarrow N' = (N'/x)M'$.
  - ⋆ if $M = y \neq x$, then, by the rule (α), $(N/x)M = y \twoheadrightarrow y = (N'/x)M'$ (and we conclude by the same way for $M = Type$ and $M = Kind$).
- if there exists terms A and B such that $M = \lambda y : A\ B$, then there exists terms A' and B' such that $M' = \lambda y : A'\ B'$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$ (because the only rules we can apply to an abstraction are (α) and (γ)). By induction hypothesis, we have $(N/x)A \twoheadrightarrow (N'/x)A'$ and $(N/x)B \twoheadrightarrow (N'/x)B'$. Therefore, using the rule (γ), we have $(N/x)M = \lambda y : ((N/x)A)\ (N/x)B \twoheadrightarrow \lambda y : ((N'/x)A')\ (N'/x)B' = (N'/x)M'$

- if there exists terms A and B such that $M = \Pi y : A\ B$, then there exists terms A' and B' such that $M' = \Pi y : A'\ B'$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$ (because the only rules we can apply to an abstraction are (α) and (δ)). By induction hypothesis, we have $(N/x)A \twoheadrightarrow (N'/x)A'$ and $(N/x)B \twoheadrightarrow (N'/x)B'$. Therefore, using the rule (δ), we have $(N/x)M = \Pi y : ((N/x)A)\ (N/x)B \twoheadrightarrow \Pi y : ((N'/x)A')\ (N'/x)B' = (N'/x)M'$
- if there exists terms P and Q such that $M = P\ Q$,
  if the last rule of the derivation of $M \twoheadrightarrow M'$ is:
  - (α) then $M' = M = P\ Q$ and we have $P \twoheadrightarrow P$ and $Q \twoheadrightarrow Q$, then, by induction hypothesis, $(N/x)P \twoheadrightarrow (N'/x)P$ and $(N/x)Q \twoheadrightarrow (N'/x)Q$. Therefore, using the rule (θ_1), we have $(N/x)M = (N/x)P\ (N/x)Q \twoheadrightarrow (N'/x)P\ (N'/x)Q = (N'/x)M'$.
  - (β) then there exists a rule $\langle s_1, s_2\rangle$ such that $M = \varepsilon_{s_2}\ \dot{s_1}$ and $M' = U_{s_1}$. Therefore, by (β), $(N/x)M = \varepsilon_{s_2}\ \dot{s_1} \twoheadrightarrow U_{s_1} = (N'/x)M'$
  - (θ_1) then there exists terms P' and Q' such that $M' = P'\ Q'$ with $P \twoheadrightarrow P'$ and $Q \twoheadrightarrow Q'$. By induction hypothesis, we have $(N/x)P \twoheadrightarrow (N'/x)P'$ and $(N/x)Q \twoheadrightarrow (N'/x)Q'$. Therefore, by (θ_1), $(N/x)M = (N/x)P\ (N/x)Q \twoheadrightarrow (N'/x)P'\ (N'/x)Q' = (N'/x)M'$
  - (θ_2) then there exists terms A, B, B', Q' such that $P = \lambda y : A\ B$ and $M' = (Q'/y)B'$ with $B \twoheadrightarrow B'$ and $Q \twoheadrightarrow Q'$. By induction hypothesis, we have $(N/x)B \twoheadrightarrow (N'/x)B'$ and $(N/x)Q \twoheadrightarrow (N'/x)Q'$. Therefore, by (θ_2),
    $(N/x)M = (\lambda y : (N/x)A\ (N/x)B)\ (N/x)Q \twoheadrightarrow ((N'/x)Q'/y)\ (N'/x)B'$
    $= (N'/x)((Q'/y)B') = (N'/x)M'$
  - (η_1) then there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, A', B, B' such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$ and $M' = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B')$, with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$. By induction hypothesis, we have $(N/x)A \twoheadrightarrow (N'/x)A'$ and $(N/x)B \twoheadrightarrow (N'/x)B'$. Therefore, by (η_1), $(N/x)M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ (N/x)A\ \ (N/x)B)$
    $\twoheadrightarrow \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ (N'/x)A'\ \ (N'/x)B') = (N'/x)M'$.
  - (η_2) then there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, A', B, B' such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$ and $M' = \Pi y : (\varepsilon_{s_1}\ A')\ (\varepsilon_{s_2}\ (B'\ y))$, with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$. By induction hypothesis, we have $(N/x)A \twoheadrightarrow (N'/x)A'$ and $(N/x)B \twoheadrightarrow (N'/x)B'$. Therefore, by (η_2), $(N/x)M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ (N/x)A\ \ (N/x)B)$
    $\twoheadrightarrow \Pi y : (\varepsilon_{s_1}\ (N'/x)A')\ (\varepsilon_{s_2}\ ((N'/x)B'\ y)) = (N'/x)M'$.

□

Then we associate, to each term t of $\lambda\Pi_P$, a term $t^\dagger$, obtained by reducing in parallel all its $\beta\mathcal{R}$-redices.

Definition 5.6. *Let t be a term of $\lambda\Pi_P$.*
We define, by induction on the structure of t, the term $t^\dagger$ as follows:

- $x^\dagger = x, \quad Type^\dagger = Type, \quad Kind^\dagger = Kind$
- $(\lambda x : A\ M)^\dagger = \lambda x : A^\dagger\ M^\dagger, \qquad (\Pi x : A\ B)^\dagger = \Pi x : A^\dagger\ B^\dagger$
- $((\lambda x : A\ M)\ N)^\dagger = (N^\dagger/x)M^\dagger$
- $(\varepsilon_{s_2}\ \dot{s_1})^\dagger = U_{s_1}$, *if* $\langle s_1, s_2\rangle \in \mathcal{A}$,
- $(\varepsilon_{s_3}(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle} A\ B))^\dagger = \Pi x : (\varepsilon_{s_1} A^\dagger)\ (\varepsilon_{s_2}(B^\dagger x))$, *if* $\langle s_1, s_2, s_3\rangle \in \mathcal{R}$,
- $(M\ N)^\dagger = M^\dagger N^\dagger$, *otherwise.*

And we verify that all terms t reduce to $t^\dagger$ in one step of parallel reduction.

Lemma 5.5. *If M is a term of $\lambda\Pi_P$, then $M \twoheadrightarrow M^\dagger$*

Proof. By induction on M.

- $x^\dagger = x$, $Type^\dagger = Type$, and $Kind^\dagger = Kind$, then by the rule (α), we have $x \twoheadrightarrow x^\dagger$, $Type \twoheadrightarrow Type^\dagger$ and $Kind \twoheadrightarrow Kind^\dagger$
- If we suppose, by induction hypothesis, $A \twoheadrightarrow A^\dagger$ and $N \twoheadrightarrow N^\dagger$, then, by the rule (γ), $\lambda x : A\ N \twoheadrightarrow \lambda x : A^\dagger\ N^\dagger = (\lambda x : A\ N)^\dagger$
- If $A \twoheadrightarrow A^\dagger$ and $B \twoheadrightarrow B^\dagger$, then by the rule (δ), $\Pi x : A\ B \twoheadrightarrow \Pi x : A^\dagger\ B^\dagger = (\Pi x : A\ B)^\dagger$
- If M is an application then we consider four cases.
  - ⋆ If there exists an axiom $\langle s_1, s_2\rangle$ such that $M = \varepsilon_{s_2}\ \dot{s_1}$, then, by the rule (β), we have $\varepsilon_{s_2}\ \dot{s_1} \twoheadrightarrow U_{s_1} = (\varepsilon_{s_2}\ \dot{s_1})^\dagger$.
  - ⋆ If there exists terms A, N and P such that $M = (\lambda x : A\ N)\ P$, and if we suppose, by induction hypothesis that $N \twoheadrightarrow N^\dagger$ and $P \twoheadrightarrow P^\dagger$, then using the rule (θ_2), we have $(\lambda x : A\ N)\ P \twoheadrightarrow (P^\dagger/x)N^\dagger = ((\lambda x : A\ N)\ P)^\dagger$.
  - ⋆ If there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A and B such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$, and if we suppose, by induction hypothesis that $A \twoheadrightarrow A^\dagger$ and $B \twoheadrightarrow B^\dagger$, then by the rule (η_2), we have $\varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B) \twoheadrightarrow \Pi x : (\varepsilon_{s_1}\ A^\dagger)\ (\varepsilon_{s_2}(B^\dagger\ x)) = (\varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B))^\dagger$.
  - ⋆ Otherwise, if there exists terms N and P such that $M = N\ P$, and if we suppose, by induction hypothesis, that $N \twoheadrightarrow N^\dagger$ and $P \twoheadrightarrow P^\dagger$, then by the rule (θ_1) we have $N\ P \twoheadrightarrow N^\dagger\ P^\dagger = (N\ P)^\dagger$.

□

Let us prove now that a one-step parallel reduct of a term M can be parallely reducted to $M^\dagger$ in one step of parallel reduction.

Lemma 5.6. *For all terms M and M' of $\lambda\Pi_P$,*
if $M \twoheadrightarrow M'$ then $M' \twoheadrightarrow M^\dagger$

Proof. By induction on the last rule of the derivation of $M \twoheadrightarrow M'$.
If the last rule is:

(α) then $M' = M$. By lemma 5.5 we have $M \twoheadrightarrow M^\dagger$

(β) then there exists a rule $\langle s_1, s_2\rangle$ such that $M = \varepsilon_{s_2}\ \dot{s_1}$ and $M' = U_{s_1}$. Therefore $M' \twoheadrightarrow U_{s_1} = M^\dagger$ by the rule (α).

(γ) then there exists terms A, A', P and P' such that $M = \lambda x : A\ P$ and $M' = \lambda x : A'\ P'$ with $A \twoheadrightarrow A'$ and $P \twoheadrightarrow P'$. By induction hypothesis, we have $A' \twoheadrightarrow A^\dagger$ and $P' \twoheadrightarrow P^\dagger$, then, by the rule ($\gamma$), $M' = \lambda x : A'\ P' \twoheadrightarrow \lambda x : A^\dagger\ P^\dagger = M^\dagger$

(δ) then there exists terms A, A', B and B' such that $M = \Pi x : A\ B$ and $M' = \Pi x : A'\ B'$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$. By induction hypothesis, we have $A' \twoheadrightarrow A^\dagger$ and $B' \twoheadrightarrow B^\dagger$, then, by the rule ($\delta$), $M' = \Pi x : A'\ B' \twoheadrightarrow \Pi x : A^\dagger\ B^\dagger = M^\dagger$

(θ_1) then there exists terms P, P', Q and Q' such that $M = P\ Q$ and $M' = P'\ Q'$ with $P \twoheadrightarrow P'$ and $Q \twoheadrightarrow Q'$.

- ⋆ If there exists terms A and B such that $P = \lambda x : A\ B$, then there exists terms A' and B' such that $P' = \lambda x : A'\ B'$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$ (because the only rules we can apply to an abstraction are (α) and (γ)). Therefore, by induction hypothesis, $B' \twoheadrightarrow B^\dagger$ and $Q' \twoheadrightarrow Q^\dagger$. And, using the rule ($\theta_2$), we have $M' = P'\ Q' = (\lambda x : A'\ B')\ Q' \twoheadrightarrow (Q^\dagger/x)B^\dagger = ((\lambda x : A\ B)\ Q)^\dagger = M^\dagger$
- ⋆ If there exists a axiom $\langle s_1, s_2\rangle$ such that $P = \varepsilon_{s_2}$ and $Q = \dot{s_1}$, then $P' = P$ and $Q' = Q$ (because the only rules we can apply to $(\varepsilon_{s_2}\ \dot{s_1})$ are (α), (β) and (θ_1 with (α) on both premises)).
  And, by (β), $M' = (\varepsilon_{s_2}\ \dot{s_1}) \twoheadrightarrow U_{s_1} = M^\dagger$
- ⋆ If there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, B, such that $P = \varepsilon_{s_3}$ and $Q = \dot{\Pi}_{\langle s_1,s_2,s_3\rangle} A\ B$, then $P' = P = \varepsilon_{s_3}$ (because the only rule we can apply is (α)), and there exists terms A' and B' such that $Q' = \dot{\Pi}_{\langle s_1,s_2,s_3\rangle} A'\ B'$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$ (because the only rule we can apply is (θ_1)). Therefore, by induction hypothesis, $A' \twoheadrightarrow A^\dagger$ and $B' \twoheadrightarrow B^\dagger$. And, by ($\eta_2$),
  $M' = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle} A'\ B') \twoheadrightarrow \Pi x : (\varepsilon_{s_1}\ A^\dagger)(\varepsilon_{s_2}\ (B^\dagger\ x) = M^\dagger$

⋆ Otherwise, $(P\ Q)^\dagger = P^\dagger\ Q^\dagger$. We have, by induction hypothesis, $P' \twoheadrightarrow P^\dagger$ and $Q' \twoheadrightarrow Q^\dagger$. Therefore, using the rule (θ_1), we have $M' = P'\ Q' \twoheadrightarrow P^\dagger\ Q^\dagger = M^\dagger$.

(θ_2) then there exists terms A, B, B', Q, Q' such that $M = (\lambda x : A\ B)\ Q$ and $M' = (Q'/x)B'$ with $B \twoheadrightarrow B'$ and $Q \twoheadrightarrow Q'$.
By induction hypothesis, we have $B' \twoheadrightarrow B^\dagger$ and $Q' \twoheadrightarrow Q^\dagger$.
Therefore, by lemma 5.4, $M' = (Q'/x)B' \twoheadrightarrow (Q^\dagger/x)B^\dagger = M^\dagger$

(η_1) then there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, B such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$ and $M' = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B')$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$.
By induction hypothesis, we have $A' \twoheadrightarrow A^\dagger$ and $B' \twoheadrightarrow B^\dagger$.
Therefore, by (η_2),
$M' = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B') \twoheadrightarrow \Pi x : (\varepsilon_{s_1} A^\dagger)\ (\varepsilon_{s_2}(B^\dagger x)) = M^\dagger$.

(η_2) then there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, B such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$ and $M' = \Pi x : (\varepsilon_{s_1} A')\ (\varepsilon_{s_2}(B' x))$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$.
By induction hypothesis, we have $A' \twoheadrightarrow A^\dagger$ and $B' \twoheadrightarrow B^\dagger$.
Therefore, by (α), (δ) and (η_1),
$M' = \Pi x : (\varepsilon_{s_1} A')\ (\varepsilon_{s_2}(B' x)) \twoheadrightarrow \Pi x : (\varepsilon_{s_1} A^\dagger)\ (\varepsilon_{s_2}(B^\dagger x)) = M^\dagger$.

□

As a corollary of the previous lemma we can prove now that the relation $\twoheadrightarrow$ is *strongly confluent* (i.e. one-step confluent).

Lemma 5.7. *The relation $\twoheadrightarrow$ is strongly confluent in $\lambda\Pi_P$,* i.e. *for all M, M' and M'', if $M \twoheadrightarrow M'$ and $M \twoheadrightarrow M''$ then there exists a term N such that $M' \twoheadrightarrow N$ and $M'' \twoheadrightarrow N$.*

Proof. By lemma 5.6, $M' \twoheadrightarrow M^\dagger$ and $M'' \twoheadrightarrow M^\dagger$. □

Finally, we focus on the relations between parallel reduction and usual reduction, to conclude that a term M can be reduced in an abitrary number of steps to a term M' if and only if it can be parallely reduced in an abitrary number of steps to M'.

Lemma 5.8. *For all terms M and M' of $\lambda\Pi_P$,*

1. *if $M \longrightarrow M'$ then $M \twoheadrightarrow M'$*
2. *if $M \twoheadrightarrow M'$ then $M \longrightarrow^* M'$*
3. *$M \twoheadrightarrow^* M'$ if and only if $M \longrightarrow^* M'$* (i.e. $\twoheadrightarrow^* = \longrightarrow^*$).

Proof.

1. If $M \longrightarrow M'$, then $M \longrightarrow_\beta M'$ or $M \longrightarrow_{\mathcal{R}} M'$
   - ⋆ If $M \longrightarrow_\beta M'$ then $M \twoheadrightarrow M'$, by (θ_2) and (α)
   - ⋆ If $M \longrightarrow_{\mathcal{R}} M'$ then $M \twoheadrightarrow M'$, by (β), or (η_2) and (α)

2. By induction on the last rule of the derivation of $M \twoheadrightarrow M'$.
   If the last rule is:

   (α) then $M' = M$, and we have $M \longrightarrow^* M$.

   (β) then there exists a rule $\langle s_1, s_2\rangle$ such that $M = \varepsilon_{s_2}\ \dot{s_1}$ and $M' = U_{s_1}$, and we have $\varepsilon_{s_2}\ \dot{s_1} \longrightarrow_{\mathcal{R}} U_{s_1}$, therefore $M \longrightarrow^* M'$.

   (γ) then there exists terms A, A', P and P' such that $M = \lambda x : A\ P$, $M' = \lambda x : A'\ P'$ with $A \twoheadrightarrow A'$ and $P \twoheadrightarrow P'$. By induction hypothesis, we have $A \longrightarrow^* A'$ and $P \longrightarrow^* P'$, therefore $M = \lambda x : A\ P \longrightarrow^* \lambda x : A'\ P' = M'$.

   (δ) then there exists terms A, A', B and B' such that $M = \Pi x : A\ B$, $M' = \Pi x : A'\ B'$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$. By induction hypothesis, we have $A \longrightarrow^* A'$ and $B \longrightarrow^* B'$, therefore $M = \Pi x : A\ B \longrightarrow^* \Pi x : A'\ B' = M'$.

   (θ_1) then there exists terms P, P', Q and Q' such that $M = P\ Q$ and $M' = P'\ Q'$ with $P \twoheadrightarrow P'$ and $Q \twoheadrightarrow Q'$. By induction hypothesis, we have $P \longrightarrow^* P'$ and $Q \longrightarrow^* Q'$, therefore $M = P\ Q \longrightarrow^* P'\ Q' = M'$

   (θ_2) then there exists terms A, B, B', Q, Q' such that $M = (\lambda x : A\ B)\ Q$ and $M' = (Q'/x)B'$ with $B \twoheadrightarrow B'$ and $Q \twoheadrightarrow Q'$. By induction
   hypothesis, we have $B \longrightarrow^* B'$ and $Q \longrightarrow^* Q'$, therefore $M = (\lambda x : A\ B)\ Q \longrightarrow^* (\lambda x : A\ B')\ Q' \longrightarrow_\beta (Q'/x)B' = M'$.

   (η_1) then there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, B such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$ and $M' = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B')$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$. By induction hypothesis, we have $A \longrightarrow^* A'$ and $B \longrightarrow^* B'$, and therefore $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B) \longrightarrow^* \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B') = M'$.

   (η_2) then there exists a rule $\langle s_1, s_2, s_3\rangle$ and terms A, B such that $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)$ and $M' = \Pi x : (\varepsilon_{s_1} A')\ (\varepsilon_{s_2}(B'x))$ with $A \twoheadrightarrow A'$ and $B \twoheadrightarrow B'$. By induction hypothesis, we have $A \longrightarrow^* A'$ and $B \longrightarrow^* B'$, therefore
   $M = \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B) \longrightarrow^* \varepsilon_{s_3}\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A'\ B')$
   $\longrightarrow_{\mathcal{R}} \Pi x : (\varepsilon_{s_1} A')\ (\varepsilon_{s_2}(B'x)) = M'$.

3. By induction on the number of reductions in $M \longrightarrow^* M'$ and the first point, for one way. And by induction on the length of the derivation of $M \twoheadrightarrow M'$ and the second point for the other way.

□

Finally, from strong confluence of $\twoheadrightarrow$, we can deduce confluence of $\longrightarrow$ in $\lambda\Pi_P$-calculus modulo.

Proposition 5.3.
The relation $\longrightarrow$ is confluent in $\lambda\Pi_P$, i.e. *for all M, M' and M'', if $M \longrightarrow^* M'$ and $M \longrightarrow^* M''$ then there exists a term N such that $M' \longrightarrow^* N$ and $M'' \longrightarrow^* N$.*

Proof. From lemma 5.7 the relation $\twoheadrightarrow$ is strongly confluent, hence it is confluent. Hence, by lemma 5.8 the relation $\longrightarrow$ is confluent. □

5.4.2 Which notion of conservativity?

Let P be a functional Pure Type System. Of course, since we can embed less powerful PTS (as the simply-typed λ-calculus), all terms of a $\lambda\Pi_P$-calculus modulo do not correspond to the translation of a term of P (think about the dependent product, namely). We could attempt to prove that if the type $\|A\|$ is inhabited in $\lambda\Pi_P$, then A is inhabited in P, and more precisely that if Γ is a context and A a term in P and t a term in $\lambda\Pi_P$, such that $\|\Gamma\| \vdash t : \|A\|$, then there exists a term u of P such that $|u| = t$ and $\Gamma \vdash u : A$. Unfortunately this property does not hold in general as shown by the following counterexamples.

Example 5.4. *If P is the simply-typed lambda-calculus, then the polymorphic identity is not well-typed in P, in particular:*

$$nat : Type \not\vdash ((\lambda X : Type\ \lambda x : X\ x)\ nat) : (nat \Rightarrow nat)$$

however, in $\lambda\Pi$, we have

$$nat : \|Type\| \vdash ((\lambda X : \|Type\|\ \lambda x : \|X\|\ x)\ |nat|) : \|nat \Rightarrow nat\|$$

Example 5.5. *If $\langle s_1, s_2, s_3\rangle$ is a rule of P, then we have $\Sigma_P \vdash \dot{\Pi}_{\langle s_1,s_2,s_3\rangle} : \|\Pi X : s_1\ ((X \Rightarrow s_2) \Rightarrow s_3)\|$ but the term $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ is not the translation of any term of P.*

Therefore, we shall prove a slightly weaker property: that if the type $\|A\|$ is inhabited *by a normal term* in $\lambda\Pi_P$, then A is inhabited in P. Notice that this restriction vanishes if $\lambda\Pi_P$ is terminating.

We shall prove, in a first step, that if $\|\Gamma\| \vdash t : \|A\|$, and t is a weak η-long normal term then there exists a term in u such that such that $|u| = t$ and $\Gamma \vdash u : A$. Then we shall get rid of this restriction on weak η-long forms.

Definition 5.7 (Weak η-long terms)**.**
*A term t of $\lambda\Pi_P$ is a weak η-long term if and only if each occurrence of $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ in t, is in a subterm of the form $(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ t_1\ t_2)$ (*i.e. *each occurrence of $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ is η-expanded).*

5.4.3 Back translation

Let us define now a back translation from terms of $\lambda\Pi_P$-calculus modulo to terms of the former PTS P. We suppose, fot that purpose, that P contains at least one sort, wich we note s_0 (PTS without sorts are not very expressive by the way).

Definition 5.8 (Back translation)**.**
We define a translation from $\lambda\Pi_P$ to P as follows:

- $x^* = x$,
- $s^* = s_0$
- $\dot{s}^* = s$,
- $U_s^* = s$,
- $(\Pi x : A.\ B)^* = \Pi x : A^*.\ B^*$,
- $(\lambda x : A.\ t)^* = \lambda x : A^*.\ t^*$,
- $(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)^* = \Pi x : A^*.\ (B^*\ x)$,
- $(\varepsilon_s\ u)^* = u^*$,
- $(t\ u)^* = t^*\ u^*$ *otherwise.*

Remark 5.1. *Notice that $\Pi x : A^*\ B^*$ is not necessarily well-formed in P even if $\Pi x : A\ B$ is well-formed in $\lambda\Pi_P$ (remind our example when P is the simply-typed λ-calculus which is not polymorphic but $\lambda\Pi_P$ is). That's why we shall not be able to prove that all back translations of a typing judgement in $\lambda\Pi_P$ are typing judgements in P. We shall only prove that if the context and the type of a typing jugement can be seen as translations as types of terms of P, then the back translation of this typing judgement is a typing judgement in P.*

Let us prove now that that this back translation is actually a right inverse of the translations as a term and as a type.

Lemma 5.9. *For all terms t of P,*
*if $|t|$ (*resp. *$\|t\|$) is well defined, then $|t|^* \longrightarrow_\beta t$ (*resp. *$\|t\|^* \longrightarrow_\beta t$).*

Proof. • Let t a term of P such that $|t|$ is defined. By induction on the structure of t.

– $|x|^* = x^* = x$

– $|s|^* = \dot{s}^* = s$

–

$$\begin{array}{rcl} |\Pi x : A\ B|^* & = & (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ |A|\ (\lambda x : (\varepsilon_{s_1}\ |A|)\ |B|))^* \\ & = & \Pi x : |A|^*.\ ((\lambda x : (\varepsilon_{s_1}\ |A|).\ |B|)^*\ x) \\ & = & \Pi x : |A|^*.\ ((\lambda x : (\varepsilon_{s_1}\ |A|)^*.\ |B|^*)\ x) \\ & = & \Pi x : |A|^*.\ ((\lambda x : |A|^*.\ |B|^*)\ x) \\ & \longrightarrow_\beta^* & \Pi x : A.\ ((\lambda x : A.\ B)\ x) \\ & & \quad \text{by induction hypothesis} \\ & \longrightarrow_\beta & \Pi x : A.\ B \end{array}$$

where $\langle s_1, s_2, s_3\rangle$ is the rule used to form $\Pi x : A.B$.

–

$$\begin{array}{rcl} |\lambda x : A.\ t|^* & = & (\lambda x : (\varepsilon_s\ |A|).\ |t|)^* \\ & = & (\lambda x : (\varepsilon_s\ |A|)^*.\ |t|^* \\ & = & \lambda x : |A|^*.\ |t|^* \\ & \longrightarrow_\beta^* & \lambda x : A.\ t \\ & & \quad \text{by induction hypothesis} \end{array}$$

– $|t\ u|^* = (|t|\ |u|)^* = |t|^*\ |u|^* \longrightarrow_\beta^* t\ u$ by induction hypothesis.

• Let t a term of P such that $\|t\|$ is defined. Then either t is a ill-typed sort s, in this case $\|s\|^* = U_s^* = s$. Or $\|t\| = \varepsilon_s|t|$ with s the type of t. In this case, $\|t\|^* = (\varepsilon_s|t|)^* = |t|^* \longrightarrow_\beta^* t$ by the first point.

□

Let us now prove some lemmas useful to prove our property of conservativity. The first one states that the back reduction of a substitution is equal to the substitution of the back translations of the considered terms.

Lemma 5.10.
For all terms t, u and variables x of $\lambda\Pi_P$, $((u/x)t)^* = (u^*/x)t^*$

Proof. By induction on the structure of t.

- If t is a variable y, then either $y = x$ and in this case, $((u/x)t)^* = u^* = (u^*/x)t^*$ since $t^* = x^* = x$. Or $y \neq x$, and in this case, $((u/x)t)^* = y^* = y = (u^*/x)y^*$. Notice that if t is an actor $\dot{s}$ or a scene U_s, then we can use the same reasoning as for the second point.

- If t is a sort s, then $(u/x)s = s$ therefore $((u/x)s)^* = s^* = s_0 = (u^*/x)s_0$ since $s_0 \neq x$.

- If t is a product $\Pi y : A.B$ then

$$\begin{aligned}((u/y)(\Pi x : A.B))^* &= (\Pi x : (u/y)A.(u/y)B)^* \\ &= \Pi x : ((u/y)A)^*.((u/y)B)^* \\ &= \Pi x : ((u^*/y)A^*).((u^*/y)B^*) \\ &\qquad \text{(by induction hypothesis)} \\ &= (u^*/y)(\Pi x : A^*.B^*) \\ &= (u^*/y)(\Pi x : A.B)^*\end{aligned}$$

- If t is an abstraction $\lambda y : A.v$ then

$$\begin{aligned}((u/y)(\lambda x : A.v))^* &= (\lambda x : (u/y)A.(u/y)B)^* \\ &= \lambda x : ((u/y)A)^*.((u/y)v)^* \\ &= \lambda x : ((u^*/y)A^*).((u^*/y)v^*) \\ &\qquad \text{(by induction hypothesis)} \\ &= (u^*/y)(\lambda x : A^*.v^*) \\ &= (u^*/y)(\lambda x : A.B)^*\end{aligned}$$

- If t is of the form $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B$,

$$\begin{aligned}((u/y)(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B))^* &= (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ (u/y)A\ (u/y)B)^* \\ &= \Pi x : ((u/y)A)^*.(((u/y)B)^*\ x) \\ &= \Pi x : ((u^*/y)A^*).(((u^*/y)B^*)\ x) \\ &\qquad \text{(by induction hypothesis)} \\ &= (u^*/y)(\Pi x : A^*.(B^*\ x)) \\ &= (u^*/y)(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ A\ B)^*\end{aligned}$$

- If t is of the form $\varepsilon_s\ v$, then $((u/y)(\varepsilon_s\ v)^* = (\varepsilon_s\ (u/y)v)^* = ((u/y)v)^* = (u^*/y)v^*$ by induction hypothesis and it is therefore equal to $(u^*/y)(\varepsilon_s\ v)^*$.

- If t is another application $t_1\ t_2$, then $((u/y)(t_1\ t_2))^* = ((u/y)t_1\ (u/y)t_2)^* = ((u/y)t_1\ (u/y)t_2)^* = ((u/y)t_1)^*\ ((u/y)t_2)^* = ((u^*/y)t_1^*)\ ((u^*/y)t_2^*)$ by induction hypothesis, and it is therefore equal to $(u^*/y)(t_1^*\ t_2^*) = (u^*/y)(t_1\ t_2)^*$.

□

Let us prove now that if a term is a β-reduct of another one, then the back translation of the former one is also a β-reduct of the latter one.

Lemma 5.11. *For all terms t, u of $\lambda\Pi_P$, if $t \longrightarrow u$ then $t^* \longrightarrow_\beta^* u^*$ in P.*

Proof.

- If $t \longrightarrow_\beta u$ then $t^* \longrightarrow_\beta u^*$. By induction on the position of the reduced β-redex in t. If $t = (\lambda x : A.t_1)t_2$, and $u = (t_2/x)t_1$, then $t^* = (\lambda x : A^*.t_1^*)t_2^* \longrightarrow_\beta (t_2^*/x)t_1^* = u^*$ by lemma 5.10. The other cases are proven by induction hypothesis.

- If $t \longrightarrow_\mathcal{R} u$, then $t^* = u^*$. By induction on the position of the reduced $\mathcal{R}$-redex in t. Notice that $(\varepsilon_{s_2}(\dot{s}_1))^* = (\dot{s}_1)^* = s_1 = (U_{s_1})^*$ and $(\varepsilon_{s_3}(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ X\ Y))^* = (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ X\ Y)^* = \Pi x : X^*.(Y^*x) = \Pi x : (\varepsilon_{s_1}\ X)^*\ (\varepsilon_{s_2}\ (Y\ x))^* = (\Pi x : (\varepsilon_{s_1}\ X)\ (\varepsilon_{s_2}\ (Y\ x)))^*$.

□

We give, in the following, a last lemma about translations and equivalence between terms.

Lemma 5.12. *For all terms A, B of P and C, D of $\lambda\Pi_P$ (such that $\|A\|$ and $\|B\|$ are well defined),*

1. *If $A \equiv_\beta B$, then $\|A\| \equiv \|B\|$.*
2. *If $C \equiv D$, then $C^* \equiv_\beta D^*$.*
3. *If $\|A\| \equiv \|B\|$, then $A \equiv_\beta B$.*
4. *If $C \equiv \|A\|$, then $C \equiv \|C^*\|$.*

Proof.

1. By induction on the length of the path of β-reductions and β-expansions between A and B, and by lemma 5.2.
2. By the same reasoning as for the first point, using lemma 5.11.
3. By the second point and lemma 5.9.
4. By the first and second points and lemma 5.9.

□

And we are finally able to prove the following property of conservativity for our translations as a term and as a type: if a term in weak η-long normal form is typed by the translation (as a type) of a type A of P in a translated context Γ of P, then it is equivalent to the translation (as a term) of a term which is of type A in Γ.

Proposition 5.4 (Conservativity)**.**
If there exists a context Γ, a term A of P, and a term t, in weak η-long normal form, of $\lambda\Pi_P$, such that $\|\Gamma\| \vdash t : \|A\|$, Then there exists a term u of P such that $|u| \equiv t$ and $\Gamma \vdash u : A$.

Proof. By induction on t.

- If t is a well-typed product or sort, then it cannot be typed by a translated type (by confluence of $\lambda\Pi_P$).
- If $t = \lambda x : B\ t'$. The term t is well typed, thus there exists a term C of $\lambda\Pi_P$, such that $\|\Gamma\| \vdash t : \Pi x : B\ C$. Therefore $\|A\| \equiv \Pi x : B\ C$ (α).
  And $\Pi x : B\ C \equiv \|(\Pi x : B\ C)^*\| = \|\Pi x : B^*\ C^*\| \equiv \Pi x : \|B^*\|\ \|C^*\|$.
  In particular (by confluence of $\lambda\Pi_P$),

  $$B \equiv \|B^*\|, \quad C \equiv \|C^*\| \quad \text{and} \quad \|\Gamma\| \vdash \lambda x : B\ t' \ :\ \Pi x : \|B^*\|\ \|C^*\|$$

  Therefore $\|\Gamma\|, x : \|B^*\| \vdash t' : \|C^*\|$. The term $\lambda x : B\ t'$ is in weak η-long normal form, thus t' is also in weak η-long normal form, and, by induction hypothesis, there exists a term u' of P, such that $|u'| \equiv t'$ and $\Gamma, x : B^* \vdash u' : C^*$. Therefore $\Gamma \vdash \lambda x : B^*\ u' \ :\ \Pi x : B^*\ C^*$ (β). Moreover, $A \equiv_\beta \Pi x : B^*\ C^*$ by (α) and lemma 5.12. Thus, by the conversion rule of P, we get $\Gamma \vdash \lambda x : B^*\ u' \ :\ A$.
  And $|\lambda x : B^*\ u'| = \lambda x : \|B^*\|\ |u'| \equiv \lambda x : B\ t' = t$.

- If t is an application or a variable, as it is normal, it has the form $x\ t_1\ ...\ t_n$ for some variable x and terms t_1, ..., t_n. We have $\|\Gamma\| \vdash x\ t_1\ ...\ t_n : \|A\|$ (α_0).

  - If x is a variable of the context Σ_P,
    - If $x = \dot{s_1}$ (where $\langle s_1, s_2\rangle$ is an axiom of P),
      then $n = 0$ (because t is well typed) and $\|A\| = U_{s_2}$.
      We have $\vdash s_1 : s_2$ in P, therefore $\Gamma \vdash s_1 : s_2$.
    - If $x = U_s$ (where s is a sort of P), then $n = 0$ and $\|A\| \equiv Type$. That's an absurdity by confluence of $\lambda\Pi_P$.
    - If $x = \varepsilon_s$ (where s is a sort of P), then, as t is well typed $n \leq 1$.

- ⋆ If $n = 1$, then $\|\Gamma\| \vdash t_1 : U_s$, and $\|A\| \equiv Type$ (absurdity).
- ⋆ If $n = 0$, then $\|A\| \equiv U_s \Rightarrow Type$, and therefore $U_s \Rightarrow Type \equiv \|(U_s \Rightarrow Type)^*\| = \|s \Rightarrow s_0\| \equiv \|s\| \Rightarrow \|s_0\|$, by lemmas 5.12 and 5.3. Therefore $Type \equiv \|s_0\|$ (absurdity).

∗ If $x = \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ (where $\langle s_1, s_2, s_3\rangle$ is a rule of P), then since t is well-typed and in weak η-long form, $n = 2$. We have $\|A\| \equiv U_{s_3}$ thus $A \equiv s_3$ by lemma 5.12.
And $\|\Gamma\| \vdash t_1 : U_{s_1}$ *i.e.* $\|\Gamma\| \vdash t_1 : \|s_1\|$.
And $\|\Gamma\|, t_1 : U_{s_1} \vdash t_2 : ((\varepsilon_{s_1} t_1) \Rightarrow U_{s_2})$ (α_1)
t_1 is also in weak η-long normal form, then, by induction hypothesis, there exists a term u_1 of P such that:

$$|u_1| \equiv t_1 \quad and \quad \Gamma \vdash u_1 : s_1 \quad (\beta_1)$$

Then, by (α_1), $\|\Gamma\|, t_1 : \|s_1\| \vdash t_2 : \|u_1 \Rightarrow s_2\|$.
In particular, $\|\Gamma\|, t_1 : \|s_1\| \vdash t_2 : \|u_1\| \Rightarrow \|s_2\|$.
However t_2 is also in weak η-long normal form, then there exists a term t_2' (in weak η-long normal form) of $\lambda\Pi_P$ such that

$$t_2 = \lambda x : \|u_1\|\ t_2' \quad and \quad \|\Gamma\|, x : \|u_1\| \vdash t_2' : \|s_2\|$$

By induction hypothesis, there exists a term u_2' of P, such that

$$|u_2'| \equiv t_2' \quad and \quad \Gamma, x : u_1 \vdash u_2' : s_2 \quad (\beta_2)$$

Then we choose $u = \Pi x : u_1\ u_2'$ that verifies $\Gamma \vdash u : s_3$, by (β_1), (β_2), and the fact that $\langle s_1, s_2, s_3\rangle$ is a rule of P. And, finally,
$|u| = \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}|u_1|\ (\lambda x : (\varepsilon_{s_1}|u_1|)\ |u_2'| \equiv \dot{\Pi}_{\langle s_1,s_2,s_3\rangle} t_1\ t_2 = t$

—∘ If x is a variable of the context Γ,
For $k \in \{0, .., n\}$, let (H_k) be the statement: *"The term $x\ t_1$ ... t_k is typable in $\|\Gamma\|$ and its type is in the image of $\|.\|$"*.
We first prove (H_0),...,(H_n) by induction.

- ⋆ $k = 0$: x is a variable of the context Γ, then there exists a well typed term or a sort T in P such that Γ contains $x : T$. Therefore $\|\Gamma\|$ contains $x : \|T\|$.
- ⋆ $0 \leq k \leq n - 1$: We suppose (H_k).
$x\ t_1$ ... t_{k+1} is well typed in Γ, then there exists terms D and E of $\lambda\Pi_P$ such that $\|\Gamma\| \vdash t_{k+1} : D$ (δ_1) ,
$\|\Gamma\| \vdash x\ t_1 \ldots t_k : \Pi y : D\ E$ (δ_2), and $\|\Gamma\| \vdash x\ t_1$ ... $t_{k+1} : E(\delta_3)$.
However, by (H_k), we can type $x\ t_1$ ... t_k by a translated type in $\|\Gamma\|$, then by (δ_2) and lemma 5.12, $\Pi y : D\ E \equiv \Pi y : \|D^*\|\ \|E^*\|$. In particular, $E \equiv \|E^*\|$ (η_1) ,
We conclude, by (δ_3), (η_1) and the conversion rule of $\lambda\Pi_P$.

Then, if $n = 0$, we take $u = x$ and Γ contains $x : T$ with $\|T\| \equiv \|A\|$. And, if $n > 0$, then, by (α_0), there exists terms B and C of $\lambda\Pi_P$ such that $\|\Gamma\| \vdash t_n : B \quad (\theta_1)$ and $\|\Gamma\| \vdash x\ t_1\ ...\ t_{n-1} : \Pi y : B\ C \quad (\theta_2)$ with $\|A\| \equiv (t_n/y)C \quad (\theta_3)$. Then, by (H_{n-1}), (θ_2), and lemma 5.12.4, we have $\Pi y : B\ C \equiv \Pi y : \|B^*\|\ \|C^*\|$, therefore $B \equiv \|B^*\|$ and $C \equiv \|C^*\|$.
Thus, $\|\Gamma\| \vdash t_n : \|B^*\|$ and $\|\Gamma\| \vdash x\ t_1\ ...\ t_{n-1} : \|\Pi y : B^*\ C^*\|$.
t_n and $x\ t_1\ ...\ t_{n-1}$ are both in weak η-long normal form, then, by induction hypothesis, there exists terms w_1 and w_2 of P such that:

$$|w_1| \equiv x\ t_1\ ...t_{n-1} \text{ and } \Gamma \vdash w_1 : \Pi y : B^*\ C^*$$
$$|w_2| \equiv t_n \text{ and } \Gamma \vdash w_2 : B^*$$

Let $u = w_1\ w_2$, we have:

$$|u| = |w_1|\ |w_2| \equiv x\ t_1\ ...\ t_{n-1}\ t_n \text{ and } \Gamma \vdash u : (w_2/y)C^*.$$

However, by (θ_3) and lemma 5.12, we have:
$A \equiv (t_n^*/y)C^* \equiv (w_2/y)C^*$, and, finally, $\Gamma \vdash u : A$.

□

5.4.4 Getting rid of weak η-long forms

Finally, we get rid of the weak η-long form restriction with the following lemmas. As we have seen, we have only proved, in proposition 5.4, that if t is a term in weak η-long normal form, and $\|\Gamma\| \vdash t : \|A\|$ is a typing judgement in $\lambda\Pi_P$, then t is equivalent in $\lambda\Pi_P$ to the translation (as a term) of a term in P, of type A in Γ. In order to enforce that conservativity property, by considering terms only in normal form, we prove, in the following, that the weak η-long form of a term has the same type as this term, and that it is equivalent to this term in $\lambda\Pi_P$.

Let us first define formally the weak η-long form of a term in $\lambda\Pi_P$. It is this term, in which all $\dot{\Pi}$s are η-expanded.

Definition 5.9 (Weak η-long form of a term)**.**
If t is a term of $\lambda\Pi_P$, its weak η-long form $t^\diamond$ is defined inductively as follows.

- $x^\diamond = x$,
- $s^\diamond = s$
- $\dot{s}^\diamond = \dot{s}$,

- $U_s^\diamond = U_s$,
- $(\Pi x : A.\ B)^\diamond = \Pi x : A^\diamond.\ B^\diamond$,
- $(\lambda x : A.\ t)^\diamond = \lambda x : A^\diamond.\ t^\diamond$,
- $(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle})^\diamond = \lambda x : U_{s_1}.\ \lambda y : ((\varepsilon_{s_1}\ x) \Rightarrow U_{s_2}).\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ x\ y)$,
  (we recall that $(\varepsilon_{s_1}\ x) \Rightarrow U_{s_2}$ *stands for* $\Pi y : (\varepsilon_{s_1}\ x).\ U_{s_2}$ *since* U_{s_2} *does not depend on* y*)*
- $(t\ u)^\diamond = t^\diamond\ u^\diamond$.

We prove, in the following lemma, some useful statements about reduction, weak η-long form and translation as a type. We prove that the weak η-long forms of two equivalent terms, are equivalent in $\lambda\Pi_P$. That a term in weak η-long form is β-equivalent to its $\diamond$-translation. That the translation as a type of a term of P is equivalent to its weak η-long form. And finally, the statement we need for our second conservativity property: if a term of $\lambda\Pi_P$ is equivalent to the translation as a type of a term of P, then it is equivalent to its $\diamond$-translation.

Lemma 5.13.
For all terms A, B *of* $\lambda\Pi_P$, *and for all well typed terms or sort* C *of* P,

1. *If* $A \longrightarrow B$ *then* $A^\diamond \longrightarrow^* B^\diamond$
2. *If* $A \equiv B$ *then* $A^\diamond \equiv B^\diamond$
3. *If* A *is in weak* η*-long form, then* $A^\diamond \longrightarrow_\beta^* A$, *in particular* $A^\diamond \equiv A$
4. $\|C\|^\diamond \equiv \|C\|$
5. *If* $A \equiv \|C\|$ *then* $A^\diamond \equiv A$

Proof. 1. If $A \longrightarrow_\beta B$, then $A^\diamond \longrightarrow_\beta B^\diamond$ (by induction on the position in A of the β-redex reduced).

If $A \longrightarrow_{\mathcal{R}} B$,

- for all axioms $\langle s_1, s_2\rangle$, $(\varepsilon_{s_2}\ (\dot{s_1}))^\diamond = \varepsilon_{s_2}\ (\dot{s_1}) \longrightarrow_{\mathcal{R}} U_{s_1} = (U_{s_1})^\diamond$.
- for all rules $\langle s_1, s_2, s_3\rangle$,

$$\begin{array}{rl} & (\varepsilon_{s_3}(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle} C\ D))^\diamond \\ = & \varepsilon_{s_3}((\lambda x : U_{s_1} \lambda y : ((\varepsilon_{s_1}\ x) \Rightarrow U_{s_2})\ (\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ x\ y))\ C^\diamond\ D^\diamond) \\ \longrightarrow_\beta^2 & \varepsilon_{s_3}(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle} C^\diamond\ D^\diamond) \\ \longrightarrow_{\mathcal{R}} & \Pi x : (\varepsilon_{s_1}\ C^\diamond)\ (\varepsilon_{s_2}\ (D^\diamond\ x)) \\ = & \Pi x : (\varepsilon_{s_1}\ C^\diamond)\ (\varepsilon_{s_2}\ (D\ x)^\diamond) \end{array}$$

2. By the first point and induction on the number of derivations and expansions from A to B.
3. By induction on A, remarking that $(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ t_1\ t_2)^\diamond \longrightarrow_\beta^2 \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ t_1^\diamond\ t_2^\diamond$.
4. By the third point and the fact that a translated term $\|C\|$ is in weak η-long form.
5. If $A \equiv \|C\|$ then $A^\diamond \equiv \|C\|^\diamond \equiv \|C\|$, by the the second and fourth points.

□

And we finally prove that the $\diamond$-translation of a term as the same type of this term, when this term is typed bt the translation (as a type) of a term of P in a translated (as a type) context.

Lemma 5.14. *Let t be a normal term of $\lambda\Pi_P$,*

$$\textit{if } \|\Gamma\| \vdash t : \|A\| \textit{ then } \|\Gamma\| \vdash t^\diamond : \|A\|$$

Proof. By induction on t.

- If t is a well-typed product or sort, then it cannot be typed by a translated type (by confluence of $\lambda\Pi_P$).
- If $t = \lambda x : B\ u$, then there exists a term C of $\lambda\Pi_P$, such that $\|A\| \equiv \Pi x : B\ C$ (α_1), with $\Gamma, x : B \vdash u : C$. By (α_1), we have $B \equiv \|B^*\|$ (α_2) and $C \equiv \|C^*\|$. Thus $\Gamma, x : \|B^*\| \vdash u : \|C^*\|$. Then, by induction hypothesis, we have $\Gamma, x : \|B^*\| \vdash u^\diamond : \|C^*\|$, therefore $\Gamma \vdash \lambda x : \|B^*\|\ u^\diamond : \Pi x : \|B^*\|\ \|C^*\| \equiv \|A\|$ thus $\Gamma \vdash \lambda x : B\ u^\diamond : \|A\|$, by (α_2). Finally, by (α_2) and lemma 5.13.5, $\lambda x : B\ u^\diamond \equiv \lambda x : B^\diamond\ u^\diamond$, therefore, by subject reduction, $\Gamma \vdash t^\diamond = \lambda x : B^\diamond\ u^\diamond : \|A\|$
- If t is an application or a variable, as it is normal, it has the form $x\ t_1\ ...\ t_n$ for some variable x and terms t_1, ..., t_n.
  We suppose $\|\Gamma\| \vdash x\ t_1\ ...\ t_n : \|A\|$ (α_0).
  - If x is a variable of the context Σ_P,
    - If $x = \dot{s_1}$ (where $\langle s_1, s_2\rangle$ is an axiom of P), then $n = 0$ (because t is well typed) and we have $(\dot{s_1})^\diamond = \dot{s_1}$.
    - If $x = U_s$ (where s is a sort of P), then $n = 0$ and $\|A\| \equiv Type$. That's an absurdity by confluence of $\lambda\Pi_P$.
    - If $x = \varepsilon_s$ (where s is a sort of P), then, as t is well typed $n \leq 1$.
      - If $n = 1$, then $\|\Gamma\| \vdash t_1 : U_s$, and $\|A\| \equiv Type$ (absurdity).
      - If $n = 0$, we have $(\varepsilon_s)^\diamond = \varepsilon_s$

- If $x = \dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ (where $\langle s_1, s_2, s_3\rangle$ is a rule of P), then since t is well-typed, $n \leq 2$. Moreover, $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$, $(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ t_1)$, and $(\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}\ t_1\ t_2)$ have the same types than their weak η-long forms.

—∘ If x is a variable of the context Γ,

- If $n = 0$, we have $x^\diamond = x$.
- If $n > 0$, then there exists terms B and C of $\lambda\Pi_P$ such that $\|\Gamma\| \vdash t_n : B \quad (\alpha_1)$ and $\|\Gamma\| \vdash x\ t_1\ ...\ t_{n-1} : \Pi y : B\ C \quad (\alpha_2)$ with $\|A\| \equiv (t_n/y)C \quad (\alpha_3)$. As in the proof of Proposition 5.4, we can type $x\ t_1\ ...\ t_{n-1}$ by a translated type, then $\Pi y : B\ C \equiv \Pi y : \|B^*\|\ \|C^*\|$. In particular, $B \equiv \|B^*\|$ and $C \equiv \|C^*\|$. Thus, $\|\Gamma\| \vdash t_n : \|B^*\|$ and we therefore also have $\|\Gamma\| \vdash x\ t_1\ ...\ t_{n-1} : \|\Pi y : B^*\ C^*\|$.
  By induction hypothesis, we have $\|\Gamma\| \vdash t_n^\diamond : \|B^*\|$ and $\|\Gamma\| \vdash x\ t_1^\diamond\ ...\ t_{n-1}^\diamond : \Pi y : \|B^*\|\ \|C^*\|$. Finally, by (α_3) and lemma 5.13.5, $\|\Gamma\| \vdash t^\diamond = x\ t_1^\diamond\ ...\ t_n^\diamond : (t_n^\diamond/y)C \equiv \|A\|$.

□

This leads to the following stronger second conservativity property: if $\lambda\Pi_P$ is strongly normalizing and a normal term of $\lambda\Pi_P$ is typed by the translation (as a type) of a type A of P in a translated context Γ of P, then it is equivalent to the translation (as a term) of a term which is of type A in Γ.

Proposition 5.5 (Conservativity)**.** *If $\lambda\Pi_P$ is stronly normalizing and there exists a context Γ, a term A of P, and a normal term t of $\lambda\Pi_P$, such that $\|\Gamma\| \vdash t : \|A\|$, Then there exists a term u of P such that $|u| \equiv t$ and $\Gamma \vdash u : A$.*

Proof. If $\|\Gamma\| \vdash t : \|A\|$, then $\|\Gamma\| \vdash t^\diamond : \|A\|$ by lemma 5.14 and so does the normal form t' of $t^\diamond$, by subject-reduction. Since t' is in weak η-long form, there exists u of P such that $|u| \equiv t'$ and $\Gamma \vdash u : A$, by proposition 5.4. Moreover, $|u| \equiv t' \equiv t^\diamond \equiv t$ by lemma 5.13.3.

□

And finally we get the following theorem: our embedding of Pure Type Systems in $\lambda\Pi$-calculus modulo is sound and conservative, in the sense that if $\lambda\Pi_P$ is strongly normalizing, then a type $\|A\|$ is inhabited by a closed term if and only if A is.

Theorem 5.1. *Let P be a functional Pure Type System, such that $\lambda\Pi_P$ is strongly normalizing. The type $\|A\|$ is inhabited by a closed term in $\lambda\Pi_P$ if and only if the type A is inhabited by a closed term in P.*

Proof. If A has a closed inhabitant in P, then by Proposition 5.1, $\|A\|$ has a closed inhabitant in $\lambda\Pi_P$. Conversely, $\|A\|$ has a closed inhabitant then the term u of proposition 5.5 is also closed.

□

Remark 5.2. *This conservativity property we have proved is similar to that of the Curry-de Bruijn-Howard correspondence. If the type A° is inhabited in $\lambda\Pi$-calculus, then the proposition A is provable in minimal natural deduction, but not all terms of type A° correspond to proofs of A. For instance, if A is the proposition $(\forall x\ P(x)) \Rightarrow P(c)$, then the normal term $\lambda\alpha : (\Pi x : \iota.\ (P\ x)).\ (\alpha\ c)$ corresponds to a proof of A but the term $\lambda\alpha : (\Pi x : \iota.\ (P\ x)).\ (\alpha\ ((\lambda y : \iota.\ y)\ c))$ does not.*

5.5 Implementation

This embedding of functional Pure Type Systems in $\lambda\Pi$-calculus modulo is the origin of an implementation of a translator from proofs of the well-known proof-assistant Coq, to a generic proof-checker based on the $\lambda\Pi$-calculus modulo, called Dedukti.

Coq is a proof-assistant which allows the interactive construction of formal proofs, and also the manipulation of functional programs consistently with their specifications. It is based on the logical framework called the calculus of inductive constructions (the calculus of constructions with inductive types).

More information is available at http://coq.inria.fr

Dedukti is a generic proof-checker based on the formalism of the $\lambda\Pi$-calculus modulo. The motivation is to provide an independent proof-checker, together with proof translators from several proof-assistants as Coq, PVS, HOL, PVS... to proofs of Dedukti, in order to provide a external check of proofs developed in those proof-assistants and increase the confidence we can have about these proofs. Dedukti has been developed by Mathieu Boespflug and Gilles Dowek. "Dedukti" means "to deduce" in Esperanto.

More information is available at http://www.lix.polytechnique.fr/dedukti

We have developed with Guillaume Burel a translator called CoqInE from Coq proofs to their equivalents expressed in $\lambda\Pi$-calculus modulo and therefore checkable by Dedukti. Since Coq-proofs are expressed in the calculus of *inductive* constructions, we have to extend the embedding presented in this chapter, in order to translate inductive types of Coq into the $\lambda\Pi$-calculus modulo. Some design choices of the theoritical aspect of this

embedding of inductive types are not completely made at the time of writing this thesis, so this embedding is not detailed in the present manuscript. But it will be soon presented in future work.

5.6 Conclusion

We have defined, in this chapter a sound and conservative embedding of functionnal Pure Type Systems into the logical framework of $\lambda\Pi$-calculus modulo, inspired both by the expression of simple type theory in Deduction modulo and by the mechanisms of universes *à la* Tarski [40] of Intuitionistic type theory. We have proposed, for each functionnal PTS P, a specific theory expressed in $\lambda\Pi$–calculus modulo, called $\lambda\Pi_P$ and given by a context (variables representing the different sorts and products of P) and rewrite rules modelizing the behaviour of those sorts and products. This embedding shows the strength of the logical framework since it subsumes all theories expressed in functionnal PTS. An interesting point is that it is, for example, possible to represent impredicative systems as System F or the Calculus of Constructions in $\lambda\Pi$-calculus modulo, whereas the $\lambda\Pi$-calculus is a predicative system.

The soundness of this embedding is comparatively easy to prove, by defining a translation from terms of P to terms of the associated $\lambda\Pi_P$ (more precisely by defining two translations, one *as a term* and one *as a type*). We were able to prove, this way, that if $\lambda\Pi_P$ is strongly normalizing then P is also strongly normalizing, since our translation maps β-redices of P to β-redices of $\lambda\Pi_P$. The conservativity is much more complicated to prove. Indeed, using the $\lambda\Pi$-calculus to simulate PTS is very convenient since we can use the dependent types of $\lambda\Pi$ and simulate therefore easily the behaviour of terms in PTS. The obverse of that convenience is that $\lambda\Pi_P$ can be more expressive than P, in the sense that there exists terms in $\lambda\Pi_P$ which do not correspond to translations of terms of P. The simpler example consists of the embedding of the simply-typed λ-calculus in the associated theory in $\lambda\Pi$-calculus modulo: dependent types are well-formed in the latter while they are not in the former. Moreover, we can build well-typed terms in $\lambda\Pi_P$ which are equivalent to translations of ill-typed terms of P. In the case of the embedding of the simply-typed λ-calculus, the translation of polymorphic typing is well-formed in the associated theory in $\lambda\Pi$-calculus modulo whereas polymorphic typing is not well formed in the simply-typed λ-calculus. We had therefore to be very precise concerning the form of statement of conservativity we wanted to prove. We proved that all terms in a specific form, typed in $\lambda\Pi_P$ by the translation of a type A, are equivalent to the translation of a term of type A in P. This particular form consists of normal terms (we had to prove an intermediate lemma concerning *weak*

η-long normal terms, but we finally got rid of this condition of weak η-long form). This conservativity property we have proved is similar to that of the Curry-de Bruijn-Howard correspondence, since this correspondence also consider only normal (proof-)terms. Finally, we were not able, with the back translation we proposed, to associate β-redices of P to redices of $\lambda\Pi_P$. Hence we could not prove that if P is strongly normalizing then so does $\lambda\Pi_P$. We shall discuss on this point in section 6.2.1.

Finally, let us make a remark on the computational behavior of our embedding. There are two ways to express proofs of simple type theory in the $\lambda\Pi$-calculus modulo. We can either use directly the fact that simple type theory can be expressed in Deduction modulo or first express the proofs of simple type theory in the Calculus of Constructions and then embed the Calculus of Constructions in the $\lambda\Pi$-calculus modulo. These two solutions have some similarities, in particular if we write o the symbol U_{Type}. But they have also some differences: the function $\lambda x\ x$ of simple type theory is translated as the symbol I — or as the term $\lambda 1$ — in the first case, using a symbol I — or the symbols λ and 1 — specially introduced in the context to express this particular theory, while it is expressed since $\lambda x\ x$ using the symbol λ of the $\lambda\Pi$-calculus modulo in the second. More generally in the second case, we exploit the similarities of the $\lambda\Pi$-calculus modulo and simple type theory — the fact that they both allow to express functions — to simplify the expression while the first method is completely generic and uses no particularity of simple type theory. This explains why this first expression requires only the $\lambda\Pi^{-}$-calculus modulo, while the second requires the conversion rule to contain β-conversion.

6

Conclusion and perspectives

6.1 Summary

In this work, we have investigated the notion of proof normalization from a semantical view. We have contributed to reveal the link between syntactic methods and semantics methods in proofs of cut elimination. One of the main contributions of this work is to refine Girard's apparently syntactic notion of reducibility candidates to obtain a sound (as before) and also complete semantics for strong normalization. This work can be seen as the following of Gödel's completeness theorem about $\{\top, \bot\}$-models and consistency in propositionnal logic, as we have seen that consistency is entailed by cut elimination and therefore by strong normalization.

We have defined a sound and complete semantics for strong normalization of theories expressed in minimal deduction modulo, and for theories expressed in $\lambda\Pi$-calculus modulo. For minimal deduction modulo, we have given a totally algebraic criterion, by defining a refinement of Dowek's Truth Values Algebras called Language-dependent Truth Values Algebras (as indicated by their names, those LDTVAs still depend on the set of closed terms of each sort we consider). We were able, with this definition, to exhibit one precise LDTVA $\mathcal{C}'$ (which is a refinement of the TVA $\mathcal{C}$ of usual reducibility candidates defined by Dowek) such that having a model valued in this algebra is equivalent for a theory to be strongly normalizing. The soundness part of this theorem follows the usual proof of Tait and Girard, while the completeness part uses an innovative method of building models using morphisms on LDTVAs. The idea is to build, when the theory is strongly normalizing, a first model valued in the LDTVA of well-typed reducibility candidates, which depends on the theory since typing does. And then map this model to a model valued in $\mathcal{C}'$ by using a morphism from each LDTVA of well-typed reducibility candidates to $\mathcal{C}'$. This new technique for building models may

be interesting for understanding the link between weak normalization and strong normalization in deduction modulo, λΠ-calculus modulo, and other logical frameworks. We shall discuss on that point in section 6.2.2.

We have defined for the moment the notion of algebra on which we should build models for λΠ-calculus modulo. However we have made a first step toward this goal by defining a notion of pre-model which provides a sound and complete semantics for strong normalization in λΠ-calculus modulo. We have followed the same way as Dowek and Werner had done for sound semantics for strong normalization in deduction modulo, by first providing a specific semantics, in order to understand on which algebra this sort of semantics can be defined thereafter. This notion of algebras may emerge from our definitions of LDTVAs and pre-models for λΠ-calculus modulo. This will be explored in future work.

Finally we proposed a sound and conservative embedding of functionnal Pure Type Systems in λΠ-calculus modulo, inspired both by the expression of simple type theory in deduction modulo and by the mechanisms of universes *à la* Tarski. We have defined, for each functionnal PTS P a theory expressed in λΠ-calculus modulo, we called $\lambda\Pi_P$, and provided a translation from terms of P to terms of $\lambda\Pi_P$. Although the soundness of this embedding was quite easy to prove, the conservativity was much more complicated and involved a lot of properties of $\lambda\Pi_P$ like its confluence for example. This embedding, combined with our notion of pre-model for λΠ-calculus modulo may be the origin of the definition of a sound and complete semantics for strong normalization in functionnal Pure Type Systems, as we shall see in section 6.2.1.

6.2 Future work

6.2.1 Toward a sound and complete semantics for strong normalization in Pure Type Systems

In order to to define a sound and complete semantics for strong normalization of Pure Type Systems, we could adapt the work we have followed for λΠ-calculus modulo, by defining a notion of pre-model for Pure Type Systems based on the work of [26] or [41], using our definition of the (CR_3') property and our technique for proving completeness of pre-models using (CR_3'). Another solution could be to use our embedding of functional Pure

Type Systems into $\lambda\Pi$-calculus modulo and our notion of sound and complete pre-models for $\lambda\Pi$-calculus modulo. We shall have a discussion on this second solution in the following.

In section 5.2.2, we have proved that for all functional PTS P, if $\lambda\Pi_P$ is strongly normalizing then P is also strongly normalizing. We therefore already have, this way, a sound semantics for strong normalization of functional PTS: if $\lambda\Pi_P$ has a pre-model, then it is strongly normalizing and so does P (although we have seen that building such a pre-model can be difficult). In order to prove that it also provides a complete semantics, we should prove that if a functional PTS P is strongly normalizing then so does $\lambda\Pi_P$ (and therefore $\lambda\Pi_P$ has a pre-model, as we have seen in section 4).

We can notice that in order to prove that strong normalization of $\lambda\Pi_P$ entails strong normalization of P, we have used the fact that well-typed terms of P were translated as well-typed terms of $\lambda\Pi_P$ and that a β-redex of P was translated as a β-redex in $\lambda\Pi_P$ (therefore a reductions sequence of a well-typed term in P is finite since its translation is finite, if $\lambda\Pi_P$ is strongly normalizing). In order to prove that if P is strongly normalizing then $\lambda\Pi_P$ is also strongly normalizing we would like to prove the two analog lemmas for the back translation: first, if a term is well-typed in $\lambda\Pi_P$ then so does its back translation in P, and secondly, a $\beta\mathcal{R}$-redex of $\lambda\Pi_P$ is back translated as a β-redex in P. Unfortunately, both are wrong, as explained in the following.

As we have seen in 5.4, the back translation of a product (defined as the product of the back translation of its arguments) may not be well-typed. This situation occurs when the embedded PTS is the simply-typed calculus for example. More generally, for all sorts s_1, s_2 of P the product $\Pi x : U_{s_1}.\ U_{s_2}$ is well-formed (and of type $Type$) in $\lambda\Pi_P$, since U_{s_1} and U_{s_2} are both of type $Type$ in $\lambda\Pi_P$, even if there is no sort s_3 such that $\langle s_1, s_2, s_3\rangle$ is a rule of P. Hence in all PTS P containing two sorts s_1, s_2 such that there exists no rule $\langle s_1, s_2, s_3\rangle$, there exists products in $\lambda\Pi_P$ for whom back translations do not correspond to well-formed products in P. We can still notice that the Calculus of Constructions is not one of those PTS. Nevertheless there exists also another problem with this back translation, even for the Calculus of Constructions: how to back translate $Type$ and $Kind$. In order to prove that the back translation of a term is typed by the back translation of its type, we have to suppose that the considered functional PTS contains at least two sorts s_1, s_2 and an axiom $\langle s_1, s_2\rangle$, and we also have to modify our back translation in order to map $Type$ to s_1 and $Kind$ to s_2. We would have also to suppose that the considered functional PTS also contains a rule $\langle s_1, s_2, s_2\rangle$, as the dependent product $\Pi x : A.B$ with A of type $Type$ and B of type $Kind$ is well formed and of type $Kind$ in $\lambda\Pi_P$. But we would still have a problem about typing of products: in $\lambda\Pi_P$, the product $\Pi x : U_{s_1}.U_{s_2}$ is of type $Type$ in $\lambda\Pi_P$ and not of type $Kind$ or U_{s_2} or any term for whom back

translation is β-equivalent to s_2. We would have therefore to modify again our back translation or obtain a weakened statement about typing of back translated terms (this second solution seems a better track to investigate).

If we manage to find a solution to this first problem, we shall still have another problem: the fact that a $\beta\mathcal{R}$ redex of $\lambda\Pi_P$ is not translated as a β-redex of P (remind that we would like to prove that a reductions sequence in $\lambda\Pi_P$ is translated as a longer reductions sequence in P: if the second one is finite then so is the first one). For example, if the considered PTS contains an axiom $\langle s_1, s_2\rangle$, then the $\mathcal{R}$-redex $(\varepsilon_{s_2}\ \dot{s_1})$ of $\lambda\Pi_P$ is back translated as the normal term of P: s_1. We think that this second problem can be easily circumvented: with the back translation we have defined in section 5.4.3, we can notice that β-redices are back translated as β-redices, and that two $\mathcal{R}$-equivalent terms have the same back translation. We can also remark that there cannot be an infinite $\mathcal{R}$-reductions sequence in $\lambda\Pi_P$, since the number of consecutive $\mathcal{R}$-reductions from a term t is lesser than or equal to the number of $\dot{s}$ (with s a sort of P) and $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ (with $\langle s_1, s_2, s_3\rangle$ a rule of P) appearing in t since if $t \longrightarrow_{\mathcal{R}} t'$ then the number of $\dot{s}$ and $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ appearing in t is stirctly greater than the one of t' (for each $\mathcal{R}$-reduction, one of those $\dot{s}$ or $\dot{\Pi}_{\langle s_1,s_2,s_3\rangle}$ disappear and none other is introduced).

Therefore the main problem for obtaining a complete semantics for functional PTS via our embedding in $\lambda\Pi$-calculus modulo, is the first one: the fact that the back translation of a well-typed term is not necessarly well-typed. A solution could be to prove that if a term is well-typed in $\lambda\Pi_P$ then its back translation is also well-typed not necessarily in P but in another PTS P' (containing more rules than P, typically all the rules $\langle s_1, s_2, s_2\rangle$ with s_1 and s_2 sorts of P such that there is no rule $\langle s_1, s_2, s_3\rangle$ in P), such that the strong normalization of P' is entailed by the one of P. For example, if P is the simply-typed λ-calculus, we should take the Calculus of Constructions for P'. We shall study this track in future work.

6.2.2 Weak and Strong normalization

The technique developped in chapter 3 could provide a new way to explore the link between weak normalization and strong normalization in deduction modulo, $\lambda\Pi$-calculus modulo and other logical frameworks like Pure Type Systems. In the case of minimal deduction modulo, the idea would be to define a notion of weakly normalizing well-typed reducibility candidates (i.e. a LDTVA), prove that we can build model valued in this LDTVA and then map this model to a model valued in $\mathcal{C}'$. Since having a $\mathcal{C}'$-valued model is a sound semantics for strong monrmalization, it would entail that weak normalization implies strong normalization in minimal deduction modulo. Adapting the notion of reducibility candidates to weak normalization (in a WN-complete way) is not very difficult, (CR_1) becomes "all proof-terms

of the considered set are weakly normalizing", (CR_2) becomes "all weakly normalizing β-reducts of an element of the considered set, are also in this set" and (CR_3) becomes "If a proof-term is a neutral and has a β-reduct in the considered set, then it is in this set". The main problem comes from the definition of the morphism, and how to map weakly normalizing terms to strongly normalizing terms, with respect to the properties of morphisms.

The naïve idea to filter the original sets of weakly normalizing terms, keeping only strongly normalizing terms, doesn't work. Indeed if proof-terms π, π' are respectively in the interpretations of propositions $A \Rightarrow B$ and A, and strongly normalizing, the proof-term $\pi\pi'$ is in the interpretation of B but may be non-strongly normalizing. A solution could be to consider for the interpretation of a proposition $A \Rightarrow B$ the set of proof-terms which map (only) proof-variables in the interpretation of A to proof-terms in the interpretation of B. Since for all proof-terms π and proof-variables α, π is strongly normalizing if and only if $\pi\alpha$ is strongly normalizing. Unfortunately we did not succeed in proving that this interpretation of $\Rightarrow$ was sufficient to prove soundness of models valued in $\mathcal{C}'$ equipped with this interpreation. If the conjecture of Jan Willem Klop that all not strongly normalizing proof-terms contain a subterm consisting of a proof-term applied itself, we should be able to prove the previous statement. Since as we have seen, in this case, we are able to type $(\lambda\alpha.\alpha\alpha)(\lambda\alpha.\alpha\alpha)$ which is not weakly normalizing.

Another idea would to be more precise on how to filter those sets of weakly normalizing proof-terms, and filter in regard to their associated types. We associate to each proposition A a subset of SN, SN_A as follows: $SN_P = SN$ for atomic propositions P and $SN_{A\Rightarrow B}$ is defined as the set of proof-terms which map elements of SN_A to elements of SN_B. For example, if P is an atomic proposition, $SN_{P\Rightarrow P}$ is the set of strongly normalizing proof-terms which are still strongly normalizing when applied to other strongly normalizing proof-terms (we can notice that we could also take a symmetric view by considering proof-terms π' such that $\pi\pi'$ is still strongly normalizing for all strongly normalizing proof-terms π). This solution raises difficulties when considering theories modulo, since we would like to filter a set A by the intersection of all $S_{A'}$, for A' a proposition equivalent to A, and we have not succeeded, for the moment, in resolving the problems it involves (notice that the symmetric solution seems to be less problematic). It will be investigated in future work, with the aim of giving a proof to the (functionnal form of the) Barendregt-Geuvers-Klop conjecture which states that all weakly normalizing (functionnal) Pure Type Systems are strongly normalizing. Solving problems detailed in this section and in the previous one would provide a solution.

6.3 Conclusion

This work provides new answers to the following two questions: what is the good notion of reducibility candidates. And what is the good notion of theory.

Girard's reducibility candidates provide a very powerful method for proving strong normalization. They have been used for proving strong normalization of a very large number of logical frameworks. However this method seems too powerful to capture exactly strongly normalizing logical frameworks. We have seen that the fact that a set of usual reducibility candidates can be built for a logical framework does not seem to be a complete criterion for strong normalization property. By providing a refinement of the (CR_3) property, we have weakened the notion of reducibility candidates just enough for this new notion to capture exactly strongly normalizing logical frameworks. Building those new reducibility candidates for theories expressed in deduction modulo, is not harder than building the original one but it is not the case for theories expressed in $\lambda\Pi$-calculus modulo when encoding polymorphic and types constructor products for example. Expressing this new notion of reducibility candidates from a algebraical point of view, as the LDTVA $\mathcal{C}'$, will certainly help us to understand even better the notion of reducibility candidates (by wondering whether this special LDTVA is an initial element of the category of LDTVAs for example).

Aristotle, Frege and Russell did not introduce the notion of theory since they thought that there was only one: a unique system in which we could express all mathematical reasonings. Since systems of Hilbert and Ackermann, logicians has thought that there exists not only one mathematical theory but many different ones. And that there exists, among them, reliable and not reliable theories. The notion of reliable theory changed over time. First, logicians considered as reliable consistent theories. However computers and in particular automated theorem provers changed the notion of reliable theory: a theory is reliable not only if it is consistent, but also if proofs of theorems that are provable can be computed by such a program. A theory is therefore considered as reliable if it has the cut elimination property, and moreover the strong normalization property. The fact that we have re-characterized this notion of reliable theory by another method which is model-theoritic seems to confirm that the strong normalization property provides a satisfying notion of reliable theory.

Bibliography

[1] P.B. Andrews. Resolution in type theory. *The Journal of Symbolic Logic*, 36(3):414-432, 1971.

[2] D. Baelde, On the interaction of minimal generic quantification and fixed points, *Logical Frameworks and Meta-languages: Theory and Practice*, 2008.

[3] S.C. Bailin, A normalization theorem for set theory, *The Journal of Symbolic Logic*, 53, pp. 673-695, 1988.

[4] H. Barendregt, Lambda calculi with types, *Handbook of Logic in Computer Science*, S. Abramsky, D. Gabbay, and T. Maibaum (eds.), Oxford University Press, pp. 117-309, 1992.

[5] S. Berardi, Towards a mathematical analysis of the Coquand-Huet Calculus of Constructions and the other systems in Barendregt's cube, *manuscript*, 1988.

[6] F. Blanqui, Definitions by rewriting in the Calculus of Constructions. *Mathematical Structures in Computer Science*, 15, 1, pp. 37-92, 2005.

[7] P. Brauner, C. Houtmann and C. Kirchner, Principles of Superdeduction, Proceedings of LICS 2007, pp. 41-50, 2007.

[8] G. Burel, Bonnes démonstrations en déduction modulo, PhD thesis, Université Henri Poincaré (Nancy 1), 2009.

[9] A. Church, A formulation of the simple theory of types, *The Journal of Symbolic Logic*, 5, pp. 56-68, 1940.

[10] M. Crabbé, Non-normalisation de ZF, manuscript, 1974.

[11] M. Crabbé, Stratification and cut-elimination, *The Journal of Symbolic Logic*, 56, pp. 213-226, 1991.

[12] T. Coquand and G. Huet, The Calculus of Constructions, *Information and Computation*, 76, pp. 95-120, 1988.

[13] D. Cousineau, Complete reducibility candidates, *Proof search in type theory*, pp 1-13, 2009.

[14] D. Cousineau and G. Dowek, Embedding Pure Types Systems in the lambda Pi-calculus modulo, *Typed Lambda calculi and Applications*. Lecture Notes in Computer Science 4583, Springer. pp. 102-117. 2007.

[15] M. De Marco and J. Lipton. Completeness and cut-elimination in the intuitionistic theory of types. *Journal of Logic and Computation*, 15:821-854, 2005.

[16] G.Dowek. Truth values algebras and proof normalization. *Types for proofs and programs*. Lecture Notes in Computer Science 4502, pp. 110-124, 2007.

[17] G. Dowek, Th. Hardin, and C. Kirchner, Theorem proving modulo, *Journal of Automated Reasoning*, 31, pp. 33-72, 2003.

[18] G. Dowek, Th. Hardin, and C. Kirchner, HOL-lambda-sigma: an intentional first-order expression of higher-order logic, *Mathematical Structures in Computer Science*, 11, pp. 1-25, 2001.

[19] G. Dowek and A. Miquel, Cut elimination for Zermelo set theory, *manuscript*, 2007.

[20] G. Dowek and B. Werner, Proof normalization modulo, *The Journal of Symbolic Logic*, 68, 4, pp. 1289-1316, 2003.

[21] G.Dowek and B.Werner. Arithmetic as a theory modulo. J. Giesel (Ed.), *Term rewriting and applications*, Lecture Notes in Computer Science 3467, Springer-Verlag, pp. 423-437, 2005.

[22] J. Ekman, Normal proofs in set theory, Doctoral thesis, Chalmers university of technology and University of Göteborg, 1994.

[23] M. Fiore, G. Plotkin and D. Turi. Abstract syntax and variable binding. *14th Annual Symposium on Logic in Computer Science*, pp. 193-202, 1999.

[24] G. Gentzen. Untersuchungen über das logische Schliessen. Mathematische Zeitschrift, 39 :176–210, 405–431, 1934.

[25] H. Geuvers and M.J. Nederhof, A modular proof of strong normalization of the calculus of constructions, *Journal of Functional Programming*, vol. 1 (2), pp. 155-189, 1991.

[26] H. Geuvers. A short and flexible proof of Strong Normalization for the Calculus of Constructions in *Types for Proofs and Programs*, Int. Workshop TYPES '94, Bastad, Sweden, Selected Papers, eds. P. Dybjer, B. Nordstr"om and J. Smith, LNCS 996, pp 14-38, Springer, 1995.

[27] 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, *2nd Scandinavian Logic Symposium*, pp. 63-92. North Holland, 1971.

[28] K.Gödel. Über die Vollständigkeit des Logikkalküls. *Doctoral dissertation*, University Of Vienna. 1929.

[29] L. Hallnäs, On normalization of proofs in set theory, Doctoral thesis, University of Stockholm, 1983.

[30] M. Hamana, Universal Algebra for Termination of Higher-Order Rewriting, *16th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 3467, Springer, pp. 135-149, 2005.

[31] R. Harper, F. Honsell, and G. Plotkin, A framework for defining logics, *Journal of the ACM*, 40, 1, pp. 143-184, 1993.

[32] L. Henkin, Completeness in the theory of types, *Journal of Symbolic Logic*, 15, pp. 81-91, 1950.

[33] R. Mc Dowell and D. Miller, Cut-Elimination for a Logic with Definitions and Induction, *Theoretical Computer Science* (232), pp. 91-119, 2000.

[34] O. Hermant. A model based cut elimination proof. In 2^{nd} *St-Petersbourg Days in Logic and Computability*, 2003.

[35] O. Hermant. *Méthodes sémantiques en déduction modulo*. Doctoral Thesis. Université de Paris 7, 2005.

[36] O. Hermant. Semantic cut elimination in the intuitionistic sequent calculus. In P. Urzyczyn, editor, *Typed Lambda Calculi and Applications*, number 3461 in Lectures Notes in Computer Science, pp. 221-233, 2005.

[37] S. Jaśkowski, S. 1934. On the Rules of Supposition in Formal Logic in *Studia Logica: Wydawnictwo Poświęcone Logice i jej Historii*, ed. by Jan Lukasiewicz,1, 1934.

[38] S. Kripke, A Completeness Theorem in Modal Logic, *Journal of Symbolic Logic*, 24(1), pp. 1-14, 1959.

[39] P. Martin-Löf, An intuitionistic type theory of types: predicative part, *Logic colloquium*, pp. 73-118, 1975

[40] P. Martin-Löf, Intuitionistic Type Theory, *Bibliopolis*, 1984.

[41] P.A.Melliès and B.Werner. A Generic Normalization Proof for Pure Type Systems, *Types for proofs and programs*, Lecture Notes in Computer Science 1512, 1996.

[42] B. Nordström, K. Petersson, and J.M. Smith, Martin-Löf's type theory. *Handbook of Logic in Computer Science*, S. Abramsky, D. Gabbay, and T. Maibaum (eds.), Clarendon Press, pp. 1-37, 2000.

[43] M. Okada. A uniform semantic proof for cut elimination and completeness of various first and higher order logics. *Theoretical Computer Science*, 281, pp. 471-498, 2002.

[44] E. Palmgren, On universes in type theory, *Twenty five years of constructive type theory*, Oxford Logic Guides, 36, Oxford University Press, 1998, pp. 191-204.

[45] D. Prawitz. Hauptsatz for higher order logic. *The Journal of Symbolic Logic*, pp. 452-457, 1968.

[46] C. Riba. On the Stability by Union of Reducibility Candidates. *10th International Conference on Foundations of Software Science and Computational Structures*, pp. 317-331, 2007.

[47] W. W. Tait. A non constructive proof of Gentzen's Hauptsatz for second order predicate logic. *Bulletin of the American Mathematical Society*, pp. 980-983, 1966.

[48] W.W. Tait. Intentional interpretations of functionals of finite type I. *The Journal of Symbolic Logic*, pp. 198-212, 1967.

[49] M. & O. Takahashi. A proof of cut-elimination theorem in simple type theory. *Journal of the Mathematical Society of Japan*, pp. 399-410, 1967.

[50] J. Terlouw, Een nadere bewijstheoretische analyse van GSTT's, *manuscript*, 1989.

[51] D. van Daalen, A description of AUTOMATH and some aspects of its language theory, *Synmposium on APL*, 1973.

[52] B. Wack, Typage et déduction dans le calcul de réécriture, PhD thesis, Université Henri Poincaré, Nancy 1, 2005.

Index of definitions

Résumé. La notion de théorie s'est séparée de la notion de logique à la fin des années 1920, lorsque Hilbert et Ackermann ont distingué les règles de déduction, indépendantes de l'objet du discours, des axiomes qui lui sont spécifiques. S'est alors posée la question de caractériser les théories, définies donc comme des ensembles d'axiomes, que l'on peut utiliser pour formaliser une partie du raisonnement mathématique. Un premier critère est la cohérence de cette théorie : le fait qu'on ne puisse pas démontrer toutes les propositions de cette théorie. Cependant il est progressivement apparu que la cohérence n'était pas une propriété suffisante. Le fait que les démonstrations constructives vérifient les propriétés de la dijonction ou du témoin, ou la complétude de certaines méthodes de démonstration automatique ne découlent pas de la seule cohérence d'une théorie. Mais toutes trois sont par contre conséquentes d'une même propriété : la normalisation des démonstrations.

En 1930, le théorème de complétude de Gödel montra que le critére de cohérence pouvait être vu sous différents angles. En plus de la définition précédente interne à la théorie de la démonstration, on peut également définir de manière algébrique la cohérence d'une théorie comme le fait qu'elle possède un modèle. L'équivalence entre ces deux définitions constitue un outil fondamental, qui a permis notamment la démonstration de la cohérence de nombreuses théories : la théorie des ensembles avec la négation de l'axiome du choix par Fraenkel et Mostovski, la théorie des ensembles avec l'axiome du choix et l'hypothèse du continue par Gödel, la théorie des ensembles avec la négation de l'hypothèse du continu par Cohen, ...

A l'inverse, la normalisation des démonstrations semblait ne pouvoir se définir que de manière interne à la théorie de la démonstration. Certains critères inspirés de la théorie des modèles étaient certes parfois utilisés pour démontrer la propriété de normalisation des démonstrations de certaines théories, mais la nécéssité de ces critéres n'avait pas été établie.

Nous proposons dans cette thèse un critère algébrique à la fois nécessaire et suffisant pour la normalisation des démonstrations. Nous montrons ainsi que la propriété de normalisation des démonstrations peut également se définir comme un critère algébrique, à l'instar de la propriété de cohérence. Nous avons pour cela défini une nouvelle notion d'algèbre de valeurs de vérités (TVA) appelée *algèbres de vérité dépendant du langage* (LDTVA). La notion de TVA permet d'exhiber l'algèbre de valeurs de vérité des candidats de réductibilité définis par Girard en 1970. L'existence d'un modèle à valeurs dans cette algèbre définit un critère algébrique suffisant pour la propriété de normalisation des démonstrations. Puis nous avons défini un raffinement de la notion de candidats de réductibilité comme une de ces LDTVAs et avons montré que l'existence d'un modèle à valeurs dans cette algèbre définit un critère algébrique toujours suffisant mais également nécessaire pour la propriété de normalisation des démonstrations.

Ce critère est défini pour les cadres logiques de la déduction minimale et du $\lambda\Pi$-calcul modulo. Et nous exhibons finalement la puissance du $\lambda\Pi$-calcul modulo en montrant que tous les systèmes de types purs fonctionnels peuvent être simulés dans ce cadre logique.

Abstract. The notion of theory split from the notion of logics in the late 1920's when Hilbert and Ackermann distinguished between deduction rules, which do not depend on the object of the speech, and axioms, which depend on it. Then arose the question of how to characterize theories - defined by a set of axioms - which can be used to formalize mathematical reasoning. A first criterion is consistency : the fact that one cannot prove all propositions of a theory. However, it gradually appeared that this criterion was not a sufficient property. Witness and disjunction properties for constructive proofs, and completeness of certain methods of automated deduction are not entailed by the consistency of a theory. But those properties are all consequences of a single property : proof normalization.

In 1930, Gödel's completeness theorem exhibited the fact that consistency can be defined by different means. It can be defined internally to the theory, as we have seen, but il can also be defined in an algebraic way : a theory is consistent if and only if it has a model. The equivalence between those two définitions is a fundamental tool that allowed to prove consistency of many theories : set theory with negation of the axiom of choice by Mostovski and Fraenkel, set theory with the axiom of choice and the continuum hypothesis by Gödel, set theory with negation of the continuum hypothesis by Cohen, ...

On the contrary, proof normalization property seemed to be only definable internally to the theory. Some model-theory inspired criteria have been used to prove proof normalization for certain theories but those criteria were not proved to be necessary.

We propose in this thesis a algebraic criterion both necessary and sufficient for proof normalization. We show this way that proof normalization property can also be defined as a algebraic criterion like consistency property. For that purpose, we define a new notion of truth values algebras (TVA) called language dependent truth values algebras (LDTVAS). The notion of TVA allows to exhibit the algebra of reducibility candidates (the notion defined by Girard in 1970). The existence of a model valued on this algebra defines a algebraic sufficient criterion for proof normalization. We then define a refinement of the notion of reducibility candidates as one of those LDTVAS and prove that the existence of a model valued on this algebra defines a algebraic still sufficient but also necessary criterion for proof normalization.

This criterion is defined for the logical frameworks of minimal deduction modulo and $\lambda\Pi$-calculus modulo. We finally exhibit the strength of $\lambda\Pi$-calculus modulo by showing that all functional Pure Type Systems can be embedded in it.