Models and Termination of Proof Reduction in the $\lambda\Pi$-Calculus Modulo Theory

Gilles Dowek

Inria and École normale supérieure de Paris-Saclay, Cachan Cedex, France
gilles.dowek@ens-paris-saclay.fr

Abstract

We define a notion of model for the $\lambda\Pi$-calculus modulo theory and prove a soundness theorem. We then define a notion of super-consistency and prove that proof reduction terminates in the $\lambda\Pi$-calculus modulo any super-consistent theory. We prove this way the termination of proof reduction in several theories including Simple type theory and the Calculus of constructions.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases model, proof reduction, Simple type theory, Calculus of constructions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.109

1 Introduction

1.1 Models and termination

In Predicate logic, a model is defined by a domain $\mathcal{M}$, a set $\mathcal{B}$ of truth values, and an interpretation function, parametrized by a valuation ϕ, mapping each term t to an element $[\![t]\!]_\phi$ of $\mathcal{M}$, and each proposition A to an element $[\![A]\!]_\phi$ of $\mathcal{B}$.

Predicate logic can be extended to Deduction modulo theory [11, 12], where a congruence on propositions defining a computational equality, also known as definitional equality in Constructive type theory [17], is added. Proofs of a proposition A are then considered to also be proofs of any proposition congruent to A. In Deduction modulo theory, like in Predicate logic, a model is defined by a domain $\mathcal{M}$, a set $\mathcal{B}$ of truth values, and an interpretation function.

Usually, the set $\mathcal{B}$ is the two-element set $\{0, 1\}$, but the notion of model can be extended to a notion of many-valued model, where $\mathcal{B}$ is an arbitrary Boolean algebra, a Heyting algebra, a pre-Boolean algebra [5], or a pre-Heyting algebra [9]. Boolean algebras permit to introduce intermediate truth values for propositions that are neither provable nor disprovable, Heyting algebras to construct models of constructive logic, and pre-Boolean and pre-Heyting algebras, where the order relation $\leq$ is replaced by a pre-order relation, to distinguish a notion of weak equivalence: $[\![A]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![B]\!]_\phi \leq [\![A]\!]_\phi$, for all ϕ, from a notion of strong equivalence: $[\![A]\!]_\phi = [\![B]\!]_\phi$, for all ϕ. In Deduction modulo theory, the first corresponds to the provability of $A \Leftrightarrow B$ and the second to the congruence.

In a model valued in a Boolean algebra, a Heyting algebra, a pre-Boolean algebra, or a pre-Heyting algebra, a proposition A is said to be valid when it is weakly equivalent to the proposition $\top$, that is when, for all ϕ, $[\![A]\!]_\phi \geq \tilde{\top}$, and this condition can be rephrased as $[\![A]\!]_\phi = \tilde{\top}$ in Boolean and Heyting algebras. A congruence $\equiv$ defined on propositions is said to be valid when, for all A and B such that $A \equiv B$, A and B are strongly equivalent, that is, for all ϕ, $[\![A]\!]_\phi = [\![B]\!]_\phi$. Note that the relation $\leq$ is used in the definition of the validity of a proposition, but not in the definition of the validity of a congruence.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 109; pp. 109:1–109:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Proof reduction terminates in Deduction modulo a theory defined by a set of axioms $\mathcal{T}$ and a congruence $\equiv$, when this theory has a model valued in the pre-Heyting algebra of reducibility candidates [12]. As a consequence, proof reduction terminates if the theory is super-consistent, that is if, for all pre-Heyting algebras $\mathcal{B}$, it has a model valued in $\mathcal{B}$ [9]. This theorem permits to completely separate the semantic and the syntactic aspects that are often mixed in the usual proofs of termination of proof reduction. The semantic aspect is in the proof of super-consistency of the considered theory and the syntactic in the universal proof that super-consistency implies termination of proof reduction.

For the termination of proof reduction, the congruence matters, but the axioms do not. Thus, the pre-order relation $\leq$ does not matter in the algebra of reducibility candidates and it is possible to define it as the trivial pre-order relation such that $C \leq C'$, for all C and C'. Such a pre-Heyting algebra is said to be trivial. As the pre-order is trivial, all the conditions defining pre-Heyting algebras, such as $a \tilde{\wedge} b \leq a$, $a \tilde{\wedge} b \leq b$... are always satisfied in a trivial pre-Heyting algebra, and a trivial pre-Heyting algebra is just a set equipped with arbitrary operations $\tilde{\wedge}$, $\tilde{\Rightarrow}$... Thus, in order to prove that proof reduction terminates in Deduction modulo a theory defined by a set of axioms $\mathcal{T}$ and a congruence $\equiv$, it is sufficient to prove that for all trivial pre-Heyting algebras $\mathcal{B}$, the theory has a model valued in $\mathcal{B}$.

1.2 The λΠ-calculus modulo theory

In Predicate logic and in Deduction modulo theory, terms, propositions, and proofs belong to three distinct languages. But, it is more thrifty to consider a single language, such as the λΠ-calculus modulo theory [8], which is implemented in the Dedukti system [1], or Martin-Löf's Logical Framework [21], and express terms, propositions, and proofs, in this language. For instance, in Predicate logic, 0 is a term, $P(0) \Rightarrow P(0)$ is a proposition and $\lambda\alpha : P(0)\ \alpha$ is a proof of this proposition. In the λΠ-calculus modulo theory, all these expressions are terms of the calculus. Only their types differ: 0 has type nat, $P(0) \Rightarrow P(0)$ has type $Type$ and $\lambda\alpha : P(0)\ \alpha$ has type $P(0) \Rightarrow P(0)$.

Like the λΠ-calculus, the λΠ-calculus modulo theory is a λ-calculus with dependent types, but, like in Deduction modulo theory, its conversion rule is extended to an arbitrary congruence, typically defined with a confluent and terminating rewrite system. This idea of extending the conversion rule beyond β-reduction is already present in Martin-Löf type theory. It is used, in various ways, in different systems [20, 6, 13, 3].

1.3 From pre-Heyting algebras to Π-algebras

The first goal of this paper is to extend the notion of pre-Heyting algebra to a notion of Π-algebra, adapted to the λΠ-calculus modulo theory.

In Predicate logic and in Deduction modulo theory, the propositions are built from atomic propositions with the connectors and quantifiers $\top$, $\bot$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, and $\exists$. Accordingly, the operations of a pre-Heyting algebra are $\tilde{\top}$, $\tilde{\bot}$, $\tilde{\wedge}$, $\tilde{\vee}$, $\tilde{\Rightarrow}$, $\tilde{\forall}$, and $\tilde{\exists}$. In the λΠ-calculus and in the λΠ-calculus modulo theory, the only connector is Π. Thus, a Π-algebra mainly has an operation $\tilde{\Pi}$. As expected, its properties are a mixture of the properties of the implication and of the universal quantifier of the pre-Heyting algebras.

1.4 Layered models

The second goal of this paper is to extend the usual notion of model to the λΠ-calculus modulo theory.

Extending the notion of model to many-sorted predicate logic requires to consider not just one domain $\mathcal{M}$, but a family of domains $\mathcal{M}_s$ indexed by the sorts. For instance, in a model of Simple type theory, the family of domains is indexed by simple types. In the $\lambda\Pi$-calculus modulo theory, the sorts also are just terms of the calculus. Thus, we shall define a model of the $\lambda\Pi$-calculus modulo theory by a family of domains $(\mathcal{M}_t)_t$ indexed by the terms of the calculus and a function $[\![.]\!]$ mapping each term t of type A and valuation ϕ to an element $[\![t]\!]_\phi$ of $\mathcal{M}_A$.

The functions $\mathcal{M}$ and $[\![.]\!]$ are similar, in the sense that both their domains is the set of terms of the calculus. The goal of the model construction is to define the function $[\![.]\!]$ and the function $\mathcal{M}$ can be seen as a tool helping to define this function. For instance, if f is a constant of type $A \to A$, where A is a term of type $Type$, and we have the rule $f(x) \longrightarrow x$, we want to define the interpretation $[\![f]\!]$ as the identity function over some set, but to state which, we must first define the function $\mathcal{M}$ that maps the term A to a set $\mathcal{M}_A$, and then define $[\![f]\!]$ as the identity function over the set $\mathcal{M}_A$.

In Predicate logic and in Deduction modulo theory, terms may be typed with sorts, but the sorts themselves have no type. In the $\lambda\Pi$-calculus modulo theory, in contrast, terms have types that have types... This explains that, in some cases, constructing the function $\mathcal{M}$ itself requires to define first another function $\mathcal{N}$, that is used as a tool helping to define this function. This can be iterated to a several layer model, where the function $[\![.]\!]$ is defined with the help of a function $\mathcal{M}$, that is defined with the help of a function $\mathcal{N}$, that is defined with the help... The number of layers depends on the model. Such layered constructions are common in proofs of termination of proof reduction [14, 18, 4], for instance for Pure Type Systems where sorts are stacked: $Type_0 : Type_1 : Type_2 : Type_3$.

Note that, in this definition of the notion of model, when a term t has type A, we do not require $[\![t]\!]_\phi$ to be an element of $[\![A]\!]_\phi$, but of $\mathcal{M}_A$. This is consistent with the notion of model of many-sorted predicate logic, where we require $[\![t]\!]_\phi$ to be an element of $\mathcal{M}_s$ and where $[\![s]\!]_\phi$ is often not even defined.

Valuations must be handled with care in such layered models. In a three layer model, for instance, the definition of $\mathcal{N}_t$ is absolute, the definition of $\mathcal{M}_t$ is relative to a valuation ψ, mapping each variable of type A to an element of $\mathcal{N}_A$, and the definition of $[\![t]\!]$ is relative to a valuation ψ and to a valuation ϕ mapping each variable of type A to an element of $\mathcal{M}_{A,\psi}$.

1.5 Super-consistency and proof reduction

The third goal of this paper is to use this notion of Π-algebra to define a notion of super-consistency and to prove that proof reduction, that is β-reduction, terminates in the $\lambda\Pi$-calculus modulo any super-consistent theory.

We prove this way the termination of proof reduction in several theories expressed in the $\lambda\Pi$-calculus modulo theory, including Simple type theory [11] and the Calculus of constructions [8]. Together with confluence, this termination of proof reduction is a property required to define these theories in the system Dedukti [1].

In Section 2, we recall the definition of the $\lambda\Pi$-calculus modulo theory and give three examples of theories expressed in this framework. In Section 3, we introduce the notion of Π-algebra and that of model for the $\lambda\Pi$-calculus modulo theory and we prove a soundness theorem. In Section 4, we define the notion of super-consistency and prove that the three theories introduced in Section 2 are super-consistent. In Section 5, we prove that proof reduction terminates in the $\lambda\Pi$-calculus modulo any super-consistent theory.

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

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A\ \text{well-formed}}\ x \notin \Gamma\ \textbf{Declaration (for object variables)}$$

$$\frac{\Gamma \vdash A : Kind}{\Gamma, x : A\ \text{well-formed}}\ x \notin \Gamma\ \textbf{Declaration (for type family variables)}$$

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

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

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}\ \textbf{Product (for types)}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}\ \textbf{Product (for kinds)}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction (for objects)}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction (for type families)}$$

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

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

$$\frac{\Gamma \vdash A : Kind \quad \Gamma \vdash B : Kind \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}\ A \equiv_\beta B\ \textbf{Conversion (for kinds)}$$

Figure 1 The λΠ-calculus.

2 The λΠ-calculus modulo theory

2.1 The λΠ-calculus

The syntax of the λΠ-calculus is

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

and the typing rules are given in Figure 1.

As usual, we write $A \to B$ for $\Pi x : A\ B$ when x does not occur in B. The α-equivalence relation is defined as usual and terms are identified modulo α-equivalence. The relation β – one step β-reduction at the root – is defined as usual. If r is a relation on terms, we write $\longrightarrow_r^1$ for the congruence closure of r, $\longrightarrow_r^+$ for the transitive closure of $\longrightarrow_r^1$, $\longrightarrow_r^*$ for its reflexive-transitive closure, and $\equiv_r$ for its reflexive-symmetric-transitive closure.

If Σ, Γ, and Δ are contexts, a substitution θ, binding the variables of Γ, is said to *have type* $\Gamma \rightsquigarrow \Delta$ *in* Σ if for all $x : A$ in Γ, we have $\Sigma, \Delta \vdash \theta x : \theta A$. In this case, if $\Sigma, \Gamma \vdash t : B$, then $\Sigma, \Delta \vdash \theta t : \theta B$.

Types are preserved by β-reduction. The β-reduction relation is confluent and strongly terminating. And each term has a unique type modulo β-equivalence [16].

A term t, well-typed in some context Γ, is a *kind* if its type in this context is $Kind$. For instance, $Type$ and $nat \to Type$ are kinds. It is a *type family* if its type is a kind. In particular, it is a *type* if its type is $Type$. For instance, nat, $array$, and $(array\ 0)$ are type families, among which nat and $(array\ 0)$ are types. It is an *object* if its type is a type. For instance, 0 and [0] are objects.

$$
\begin{array}{rcl}
\iota & : & Type \\
o & : & Type \\
\Rightarrow & : & o \to o \to o \\
\forall_A & : & (A \to o) \to o \\
\varepsilon & : & o \to Type \\
\\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall_A\ x)) & \longrightarrow & \Pi z : A\ (\varepsilon\ (x\ z))
\end{array}
$$

with a finite number of quantifiers $\forall_A$

Figure 2 Simple type theory.

2.2 The λΠ-calculus modulo theory

▶ **Definition 1** (Rewrite rule). A *rewrite rule* is a triple $l \longrightarrow^{\Gamma} r$ where Γ is a context and l and r are β-normal terms. Such a rule is *well-typed* in the context Σ if, in the $\lambda\Pi$-calculus, the context Σ, Γ is well-formed and there exists a term A such that the terms l and r both have type A in this context.

If Σ is a context, $l \longrightarrow^{\Gamma} r$ is a rewrite rule well-typed in Σ and θ is a substitution of type $\Gamma \rightsquigarrow \Delta$ in Σ, then the terms θl and θr both have type θA in the context Σ, Δ. The relation $\mathcal{R}$ – one step $\mathcal{R}$-reduction at the root – is defined by: $t\ \mathcal{R}\ u$ is there exists a rewrite rule $l \longrightarrow^{\Gamma} r$ and a substitution θ such that $t = \theta l$ and $u = \theta r$. The relation $\beta\mathcal{R}$ – one step $\beta\mathcal{R}$-reduction at the root – is the union of β and $\mathcal{R}$.

▶ **Definition 2** (Theory). A *theory* is a pair formed with a context Σ, well-formed in the $\lambda\Pi$-calculus, and a set of rewrite rules $\mathcal{R}$, well-typed in Σ in the $\lambda\Pi$-calculus.

The variables declared in Σ are called *constants*. They replace the sorts, the function symbols, the predicate symbols, and the axioms of a theory in Predicate logic.

▶ **Definition 3** (The λΠ-calculus modulo theory). The $\lambda\Pi$-*calculus modulo* $\Sigma, \mathcal{R}$ is the extension of the $\lambda\Pi$-calculus obtained modifying the **Declaration** rules to replace the condition $x \notin \Gamma$ with $x \notin \Sigma, \Gamma$, the **Variable** rules to replace the condition $x : A \in \Gamma$ by $x : A \in \Sigma, \Gamma$, and the **Conversion** rules to replace the condition $A \equiv_{\beta} B$ with $A \equiv_{\beta\mathcal{R}} B$.

In this paper, we assume that the relation $\longrightarrow^{1}_{\beta\mathcal{R}}$ is confluent and has the subject reduction property. Confluence and subject reduction are indeed needed to build models and prove termination of proof reduction. This is consistent with the methodology proposed in [2]: first prove confluence and subject reduction, then termination.

2.3 Examples of theories

Simple type theory can be expressed in Deduction modulo theory [10]. The main idea in this presentation is to distinguish terms of type o from propositions. If t is a term of type o, the corresponding proposition is written $\varepsilon(t)$. The term t is a *propositional content* or a *code* of the proposition $\varepsilon(t)$. This way, it is not possible to quantify over propositions, but it is possible to quantify over codes of propositions: there is no proposition $\forall X\ (X \Rightarrow X)$, but there is a proposition $\forall x\ (\varepsilon(x) \Rightarrow \varepsilon(x))$, respecting the syntax of Predicate logic, where the predicate symbol ε is applied to the variable x to form a proposition. In this presentation,

$$
\begin{array}{rcl}
type & : & Type \\
\iota & : & type \\
o & : & type \\
arrow & : & type \to type \to type \\
\eta & : & type \to Type \\
\Rightarrow & : & (\eta\ o) \to (\eta\ o) \to (\eta\ o) \\
\forall & : & \Pi a : type\ (((\eta\ a) \to (\eta\ o)) \to (\eta\ o)) \\
\varepsilon & : & (\eta\ o) \to Type \\
 & & \\
(\eta\ (arrow\ x\ y)) & \longrightarrow & (\eta\ x) \to (\eta\ y) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z))
\end{array}
$$

Figure 3 Simple type theory with a parametric quantifier.

each simple type is a sort and, for each simple type A, there is a quantifier $\forall_A$. Thus, the language contains an infinite number of sorts and an infinite number of constants.

This presentation can be adapted to the λΠ-calculus modulo theory. To avoid declaring an infinite number of constants for simple types, we can just declare two constants ι and o of type $Type$ and use the product of the λΠ-calculus modulo theory to represent the simple types $\iota \to \iota$, $\iota \to \iota \to \iota$, $\iota \to o$... We should declare an infinite number of quantifiers $\forall_A$, indexed by simple types, but this can be avoided as, in each specific proof, only a finite number of such quantifiers occur. This leads to the theory presented in Figure 2.

Another possibility is to add the type A as an extra argument of the quantifier $\forall$. To do so, we need to introduce a type $type$ for codes of simple types, two constants ι and o, of type $type$, and not $Type$, a constant $arrow$ of type $type \to type \to type$, and a decoding function η of type $type \to Type$. This way, the quantifier $\forall$ can be given the type $\Pi a : type\ (((\eta\ a) \to (\eta\ o)) \to (\eta\ o))$. This leads to the theory presented in Figure 3.

The Calculus of constructions [7] can also be expressed in the λΠ-calculus modulo theory [8] as the theory presented in Figure 4.

3 Algebras and Models

3.1 Π-algebras

The notion of Π-algebra is an adaptation of that of pre-Heyting algebra to the λΠ-calculus.

▶ **Definition 4** (Π-algebra)**.** A Π-algebra is formed with

- a set $\mathcal{B}$,
- a pre-order relation $\leq$ on $\mathcal{B}$,
- an element $\tilde{\top}$ of $\mathcal{B}$,
- a function $\tilde{\wedge}$ from $\mathcal{B} \times \mathcal{B}$ to $\mathcal{B}$,
- a subset $\mathcal{A}$ of $\mathcal{P}^+(\mathcal{B})$, the set of non-empty subsets of $\mathcal{B}$,
- a function $\tilde{\Pi}$ from $\mathcal{B} \times \mathcal{A}$ to $\mathcal{B}$,

such that

- $\tilde{\top}$ is a maximal element for $\leq$, that is for all a in $\mathcal{B}$, $a \leq \tilde{\top}$,
- $a\ \tilde{\wedge}\ b$ is a greatest lower bound of $\{a, b\}$ for $\leq$, that is $a\ \tilde{\wedge}\ b \leq a$, $a\ \tilde{\wedge}\ b \leq b$, and for all c, if $c \leq a$ and $c \leq b$, then $c \leq a\ \tilde{\wedge}\ b$,
- $a \leq \tilde{\Pi}(b, S)$ if and only if for all c in S, $a\ \tilde{\wedge}\ b \leq c$.

A Π-algebra is *full* if $\mathcal{A} = \mathcal{P}^+(\mathcal{B})$, that is if $\tilde{\Pi}$ is total on $\mathcal{B} \times \mathcal{P}^+(\mathcal{B})$.

$$
\begin{array}{rcl}
type & : & Type \\
o & : & type \\
\eta & : & type \to Type \\
\varepsilon & : & (\eta\ o) \to Type \\
\dot{\Pi}_{KK} & : & \Pi x : type\ (((\eta\ x) \to type) \to type) \\
\dot{\Pi}_{TT} & : & \Pi x : (\eta\ o)\ (((\varepsilon\ x) \to (\eta\ o)) \to (\eta\ o)) \\
\dot{\Pi}_{KT} & : & \Pi x : type\ (((\eta\ x) \to (\eta\ o)) \to (\eta\ o)) \\
\dot{\Pi}_{TK} & : & \Pi x : (\eta\ o)\ (((\varepsilon\ x) \to type) \to type)
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (\dot{\Pi}_{KK}\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\eta\ (y\ z)) \\
(\varepsilon\ (\dot{\Pi}_{TT}\ x\ y)) & \longrightarrow & \Pi z : (\varepsilon\ x)\ (\varepsilon\ (y\ z)) \\
(\varepsilon\ (\dot{\Pi}_{KT}\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z)) \\
(\eta\ (\dot{\Pi}_{TK}\ x\ y)) & \longrightarrow & \Pi z : (\varepsilon\ x)\ (\eta\ (y\ z))
\end{array}
$$

Figure 4 The Calculus of constructions.

Note that is the relation $\leq$ is a pre-order, and not necessarily an order, greatest lower bounds are not necessarily unique, when they exist.

Note also that, from the operation $\tilde{\Pi}$, we can define an exponentiation operation $b \tilde{\to} c = \tilde{\Pi}(b, \{c\})$ that verifies the usual properties of exponentiation: $a \leq b \tilde{\to} c$ if and only if $a\ \tilde{\wedge}\ b \leq c$. When the set S has a greatest lower bound $\tilde{\bigwedge} S$, the operation mapping b and S to $b \tilde{\to} \tilde{\bigwedge} S$ verifies the same properties as $\tilde{\Pi}$: $a \leq b \tilde{\to} \tilde{\bigwedge} S$ if and only if $a\ \tilde{\wedge}\ b \leq \tilde{\bigwedge} S$ if and only if for all c in S, $a\ \tilde{\wedge}\ b \leq c$. But this decomposition is possible only when all sets of $\mathcal{A}$ have greatest lower bounds.

▶ **Example 5.** The algebra $\langle \{0,1\}, 1, \tilde{\wedge}, \mathcal{P}^+(\{0,1\}), \tilde{\Pi} \rangle$, where $\tilde{\wedge}$ and $\tilde{\Pi}$ are defined by the tables below, is a Π-algebra. Note that, dropping the middle column of the table of $\tilde{\Pi}$, we get the table of implication and, dropping the first line, that of the universal quantifier.

$\tilde{\wedge}$	0	1
0	0	0
1	0	1

$\tilde{\Pi}$	$\{0\}$	$\{0,1\}$	$\{1\}$
0	1	1	1
1	0	0	1

3.2 Models valued in a Π-algebra $\mathcal{B}$

▶ **Definition 6** (Model). A *model* is a family of interpretation functions $\mathcal{D}^1, ..., \mathcal{D}^n$ such that for all i, $\mathcal{D}^i$ is a function mapping each term t of type B in some context Γ, function ϕ_1 mapping each variable $x : A$ of Γ to an element of $\mathcal{D}^1_A$, ..., and function ϕ_{i-1} mapping each variable $x : A$ of Γ to an element of $\mathcal{D}^{i-1}_{A,\phi_1,...,\phi_{n-2}}$, to some $\mathcal{D}^i_{t,\phi_1,...,\phi_{i-1}}$ in $\mathcal{D}^{i-1}_{B,\phi_1,...,\phi_{i-2}}$, and for all t, u, ϕ_1, ..., ϕ_{n-1}

$$\mathcal{D}^n_{(u/x)t,\phi_1,...,\phi_{n-1}} = \mathcal{D}^n_{t,(\phi_1,x=\mathcal{D}^1_u),...,(\phi_{n-1},x=\mathcal{D}^{n-1}_{u,\phi_1,...,\phi_{n-2}})}$$

For the last function $\mathcal{D}^n$, we write $[\![t]\!]_{\phi_1,...,\phi_{n-1}}$ instead of $\mathcal{D}^n_{t,\phi_1,...,\phi_{n-1}}$.

In the examples presented in this paper, we use the cases $n = 2$ and $n = 3$ only. The general definition then specializes as follows.

▶ **Example 7.** When $n = 2$, a model is given by two functions $\mathcal{M}$ and $[\![.]\!]$ such that
- $\mathcal{M}$ is a function mapping each term t of type B in Γ to some $\mathcal{M}_t$,
- $[\![.]\!]$ is a function mapping each term t of type B in Γ and function ϕ mapping each variable $x : A$ of Γ to an element of $\mathcal{M}_A$, to some $[\![t]\!]_\phi$ in $\mathcal{M}_B$, such that for all t, u and ϕ

$$[\![(u/x)t]\!]_\phi = [\![t]\!]_{\phi, x=[\![u]\!]_\phi}.$$

This generalizes of the usual definition of *model* for many-sorted predicate logic.

Note that if f is a constant of type $A \to A \to A$, we can define the function $\hat{f}$ mapping a and b in $\mathcal{M}_A$ to $[\![(f\ x\ y)]\!]_{x=a,y=b}$. Using the property $[\![(u/x)t]\!]_\phi = [\![t]\!]_{\phi,x=[\![u]\!]_\phi}$, we then get $[\![(f\ t\ u)]\!]_\phi = \hat{f}([\![t]\!]_\phi, [\![u]\!]_\phi)$, which is the usual definition of an interpretation.

Note also that the first interpretation function $\mathcal{M}$ does not depend on any valuation, so it must be very rudimentary. For instance in Definition 15 below, for all objects and most types, we have $\mathcal{M}_t = \{e\}$. Only the types o, $o \to o$... are interpreted in a non trivial way. Nevertheless, it is sufficient to support the definition of the function $[\![.]\!]$.

▶ **Example 8.** When $n = 3$, a model is given by three functions $\mathcal{N}$, $\mathcal{M}$, and $[\![.]\!]$ such that
- $\mathcal{N}$ is a function mapping each term t of type B in Γ to some $\mathcal{N}_t$,
- $\mathcal{M}$ is a function mapping each term t of type B in Γ and function ψ mapping each variable $x : A$ of Γ to an element of $\mathcal{N}_A$, to some $\mathcal{M}_{t,\psi}$ in $\mathcal{N}_B$,
- $[\![.]\!]$ is a function mapping each term t of type B in Γ, function ψ mapping each variable $x : A$ of Γ to an element of $\mathcal{N}_A$, and function ϕ mapping each variable $x : A$ of Γ to an element of $\mathcal{M}_{A,\psi}$, to some $[\![t]\!]_{\psi,\phi}$ in $\mathcal{M}_{B,\psi}$, such that for all t, u, ψ, and ϕ

$$[\![(u/x)t]\!]_{\psi,\phi} = [\![t]\!]_{(\psi, x=\mathcal{M}_{u,\psi}),(\phi, x=[\![u]\!]_{\psi,\phi})}.$$

▶ **Definition 9** (Model valued in a Π-algebra $\mathcal{B}$)**.** Let $\mathcal{B} = \langle \mathcal{B}, \tilde{\top}, \tilde{\wedge}, \mathcal{A}, \tilde{\Pi} \rangle$ be a Π-algebra. A model is *valued in* $\mathcal{B}$ if
- $\mathcal{D}^{n-1}_{Kind,\phi_1,...,\phi_{n-2}} = \mathcal{D}^{n-1}_{Type,\phi_1,...,\phi_{n-2}} = \mathcal{B}$,
- $[\![Kind]\!]_{\phi_1,...,\phi_{n-1}} = [\![Type]\!]_{\phi_1,...,\phi_{n-1}} = \tilde{\top}$,
- $[\![\Pi x : C\ D]\!]_{\phi_1,...,\phi_{n-1}} = \tilde{\Pi}([\![C]\!]_{\phi_1,...,\phi_{n-1}}, \{[\![D]\!]_{(\phi_1,x=c_1),...,(\phi_{n-1},x=c_{n-1})} \mid c_1 \in \mathcal{D}^1_C, ..., c_{n-1} \in \mathcal{D}^{n-1}_{C,\phi_1,...,\phi_{n-2}}\})$.

We often write $\overline{\phi}$ for a sequence $\phi_1, ..., \phi_n$ and, if $\overline{c} = c_1, ..., c_n$, we write $\overline{\phi}, x = \overline{c}$ for the sequence $(\phi_1, x = c_1), ..., (\phi_n, x = c_n)$.

▶ **Definition 10** (Validity)**.** A model $\mathcal{M}$ valued in some Π-algebra $\mathcal{B}$ is *model* of a theory $\Sigma, \mathcal{R}$, or the theory is *valid* in the model, if
- for all constants $c : A$ in Σ, we have $[\![A]\!] \geq \tilde{\top}$,
- and for all A and B well-typed in a context Γ, such that $A \equiv_{\beta\mathcal{R}} B$, we have for all i, for all $\overline{\phi}$, $\mathcal{D}^i_{A,\overline{\phi}} = \mathcal{D}^i_{B,\overline{\phi}}$.

▶ **Theorem 11** (Soundness)**.** *Let $\mathcal{M}$ be a model, valued in some Π-algebra $\mathcal{B}$, of a theory $\Sigma, \mathcal{R}$. Then, for all judgments $x_1 : A_1, ..., x_p : A_p \vdash t : B$ derivable in $\Sigma, \mathcal{R}$, and for all $\overline{\phi}$, we have*

$$[\![A_1]\!]_{\overline{\phi}} \tilde{\wedge} ... \tilde{\wedge} [\![A_p]\!]_{\overline{\phi}} \leq [\![B]\!]_{\overline{\phi}}.$$

▶ **Corollary 12** (Consistency)**.**
- *Let $\mathcal{M}$ be a model, valued in some Π-algebra $\mathcal{B}$, of a theory $\Sigma, \mathcal{R}$. Then, for all judgments $\vdash t : B$ derivable in $\Sigma, \mathcal{R}$, we have $[\![B]\!]_{\overline{\phi}} \geq \tilde{\top}$.*

- *Let $\mathcal{M}$ be a model, valued in the two-element* Π*-algebra of Example 5, of a theory* $\Sigma, \mathcal{R}$*. Then, for all judgments* $\vdash t : B$*, derivable in this theory, we have* $[\![B]\!]_{\overline{\phi}} = 1$*.*
- *Let* $\Sigma, \mathcal{R}$ *be a theory that has a model, valued in the two-element* Π*-algebra of Example 5. Then, there is no term* t *such that the judgment* $P : Type \vdash t : P$ *is derivable in* Σ, Γ*.*

4 Super-consistency

4.1 Super-consistency

We now want to define a notion of notion of *super-consistency*: a theory is super-consistent if for every Π-algebra, there exists a model of this theory valued in this algebra.

Unfortunately, this constraint is sometimes too strong, as it does not allow to define interpretations as fixed points, for instance if we have a rule $P \longrightarrow ((P \Rightarrow Q) \Rightarrow Q)$, we want to define the interpretation of P as the fixed point of the function mapping b to $(b \mathrel{\tilde{\Rightarrow}} a) \mathrel{\tilde{\Rightarrow}} a$, where a is the interpretation of Q, but this function does not have a fixed point in all Π-algebras. Thus, we weaken this constraint, requiring the existence of model for *complete* Π-algebras only. Defining this notion of completeness requires to introduce an order relation $\sqsubseteq$, that need not be related to the pre-order $\leq$.

▶ **Definition 13** (Ordered, complete Π-algebra)**.** A Π-algebra is *ordered* if it is equipped with an order relation $\sqsubseteq$ such that the operation $\tilde{\Pi}$ is left anti-monotonic and right monotonic with respect to $\sqsubseteq$, that is

- if $a \sqsubseteq b$, then for all S, $\tilde{\Pi}(b, S) \sqsubseteq \tilde{\Pi}(a, S)$,
- if $S \sqsubseteq T$, then for all a, $\tilde{\Pi}(a, S) \sqsubseteq \tilde{\Pi}(a, T)$,

where the relation $\sqsubseteq$ is extended to sets of elements of $\mathcal{B}$ in a trivial way: $S \sqsubseteq T$ if for all a in S, there exists a b in T such that $a \sqsubseteq b$.

It is *complete* if every subset of $\mathcal{B}$ has a least upper bound for the relation $\sqsubseteq$.

▶ **Definition 14** (Super-consistency)**.** A theory $\Sigma, \mathcal{R}$, is *super-consistent* if, for every full, ordered and complete Π-algebra $\mathcal{B}$, there exists a model $\mathcal{M}$, valued in $\mathcal{B}$, of $\Sigma, \mathcal{R}$.

We now prove that the three theories presented in Section 2.3 are super-consistent.

4.2 Simple type theory

Let $\mathcal{B} = \langle \mathcal{B}, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{B}), \tilde{\Pi} \rangle$ be a full Π-algebra. We construct a model of Simple type theory, valued in $\mathcal{B}$, in two steps. The first is the construction of the interpretation function $\mathcal{M}$ and the second the construction of the interpretation function $[\![.]\!]$. The key idea in this construction is to take $\mathcal{M}_o = \mathcal{B}$, to interpret ε as the identity over $\mathcal{B}$, and $\Rightarrow$ like $\rightarrow$ in order to validate the rewrite rule

$$\varepsilon\ (\Rightarrow\ \ x\ y) \longrightarrow (\varepsilon\ x) \rightarrow (\varepsilon\ y)\,.$$

Let S and T be two sets, we write $\mathcal{F}(S, T)$ for the set of functions from S to T. Let $\{e\}$ be an arbitrary one-element set such that e is not in $\mathcal{B}$.

▶ **Definition 15** (A model of Simple type theory)**.**

- $\mathcal{M}_{Kind} = \mathcal{M}_{Type} = \mathcal{B}$,
- $\mathcal{M}_{\Pi x:C\ D} = \mathcal{F}(\mathcal{M}_C, \mathcal{M}_D)$, except if $\mathcal{M}_D = \{e\}$, in which case $\mathcal{M}_{\Pi x:C\ D} = \{e\}$,
- $\mathcal{M}_\iota = \mathcal{M}_\Rightarrow = \mathcal{M}_{\forall_A} = \mathcal{M}_\varepsilon = \{e\}$,
- $\mathcal{M}_o = \mathcal{B}$,

- $\mathcal{M}_x = \{e\}$,
- $\mathcal{M}_{\lambda x:C\ t} = \mathcal{M}_t$,
- $\mathcal{M}_{(t\ u)} = \mathcal{M}_t$.

- $[\![Kind]\!]_\phi = [\![Type]\!]_\phi = \tilde{\top}$,
- $[\![\Pi x : C\ D]\!]_\phi = \tilde{\Pi}([\![C]\!]_\phi, \{[\![D]\!]_{\phi,x=c} \mid c \in \mathcal{M}_C\})$,
- $[\![\iota]\!]_\phi = \tilde{\top}$,
- $[\![o]\!]_\phi = \tilde{\top}$,
- $[\![\Rightarrow]\!]_\phi$ is the function mapping a and b in $\mathcal{B}$ to $\tilde{\Pi}(a, \{b\})$,
- $[\![\forall_C]\!]_\phi$ is the function mapping f in $\mathcal{F}(\mathcal{M}_C, \mathcal{B})$ to $\tilde{\Pi}([\![C]\!]_\phi, \{(f\ c) \mid c \in \mathcal{M}_C\})$,
- $[\![\varepsilon]\!]_\phi$ is the identity on $\mathcal{B}$,
- $[\![x]\!]_\phi = \phi x$,
- $[\![\lambda x : C\ t]\!]_\phi$ is the function mapping c in $\mathcal{M}_C$ to $[\![t]\!]_{\phi,x=c}$, except if for all c in $\mathcal{M}_C$, $[\![t]\!]_{\phi,x=c} = e$ in which case $[\![\lambda x : C\ t]\!]_\phi = e$,
- $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi\ [\![u]\!]_\phi$, except if $[\![t]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = e$.

We prove that if $\Gamma \vdash t : B$, and ϕ is a function mapping variables $x : A$ of Γ to elements of $\mathcal{M}_A$, then $[\![t]\!]_\phi \in \mathcal{M}_B$. That for all t, u and ϕ, $[\![(u/x)t]\!]_\phi = [\![t]\!]_{\phi,x=[\![u]\!]_\phi}$. And that if $t \equiv_{\beta\mathcal{R}} u$ then $\mathcal{M}_t = \mathcal{M}_u$ and $[\![t]\!]_\phi = [\![u]\!]_\phi$. We thus get the following theorem.

▶ **Theorem 16.** *Simple type theory is super-consistent.*

4.3 Simple type theory with a parametric quantifier

In a model of Simple type theory with a parametric quantifier, like in the previous section, we want to take $\mathcal{M}_o = \mathcal{B}$. But, unlike in the previous section, we do not have $o : Type$, but $o : type : Type$. So o is now an object.

In the previous section, we took $\mathcal{M}_t = \{e\}$ for all objects. This permitted to define $\mathcal{M}_{(t\ u)}$ and $\mathcal{M}_{\lambda x:C\ t}$ as $\mathcal{M}_t$ and validate β-reduction trivially. But this is not possible anymore in Simple type theory with a parametric quantifier, where $\mathcal{M}_o$ is $\mathcal{B}$ and $\mathcal{M}_{arrow(o,o)}$ is $\mathcal{F}(\mathcal{B}, \mathcal{B})$. So, we cannot define $\mathcal{M}_{\lambda x:type\ x}$ to be $\mathcal{M}_x$, but we need to define it as a function. To help to construct this function, we need to construct first another interpretation function $(\mathcal{N}_t)_t$ and parametrize the definition of $\mathcal{M}_t$ itself by a function ψ mapping variables of type A to elements of $\mathcal{N}_A$. Thus the model is a three layer model.

Like in the previous section, we want to define $\mathcal{M}_{\Pi x:C\ D,\psi}$, as the set of functions from $\mathcal{M}_{C,\psi}$ to $\mathcal{M}_{D,\psi'}$. But to define this set $\mathcal{M}_{D,\psi'}$, we need to extend the function ψ, mapping x to an element of $\mathcal{N}_C$. To have such an element of $\mathcal{N}_C$, we need to define $\mathcal{M}_{\Pi x:C\ D,\psi}$ as the set of functions mapping $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$ to an element of $\mathcal{M}_{D,(\psi,x=c')}$. As a consequence, if ϕ is a function mapping x of type A to some element of $\mathcal{M}_A$, we need to define $[\![(t\ u)]\!]_\phi$ not as $[\![t]\!]_\phi\ [\![u]\!]_\phi$ but as $[\![t]\!]_\phi\ \langle \mathcal{M}_{u,\psi}, [\![u]\!]_\phi\rangle$. As a consequence $[\![.]\!]$ must be parametrized by both ψ and ϕ.

Let $\mathcal{B} = \langle \mathcal{B}, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{B}), \tilde{\Pi}\rangle$ be a full Π-algebra.

Let $\{e\}$ be an arbitrary one-element set. Let $\mathcal{U}$ be a set containing $\mathcal{B}$ and $\{e\}$, and closed by function space and Cartesian product, that is such that if S and T are in $\mathcal{U}$ then so are $S \times T$ and $\mathcal{F}(S, T)$. Such a set can be constructed, with the replacement scheme, as follows

$$\begin{aligned} \mathcal{U}_0 &= \{\mathcal{B}, \{e\}\}, \\ \mathcal{U}_{n+1} &= \mathcal{U}_n \cup \{S \times T \mid S, T \in \mathcal{U}_n\} \cup \{\mathcal{F}(S, T) \mid S, T \in \mathcal{U}_n\}, \\ \mathcal{U} &= \bigcup_n \mathcal{U}_n. \end{aligned}$$

Then, let $\mathcal{V}$ be the smallest set containing $\{e\}$, $\mathcal{B}$, and $\mathcal{U}$, and closed by Cartesian product and dependent function space, that is, if S is in $\mathcal{V}$ and T is a family of elements of $\mathcal{V}$ indexed by S, then the set of functions mapping an element s of S to an element of T_s is an element of $\mathcal{V}$. As noted in [19], the construction of the set $\mathcal{V}$, unlike that of $\mathcal{U}$, requires an inaccessible cardinal. Note that $\mathcal{U}$ is both an element and a subset of $\mathcal{V}$.

▶ **Definition 17** (A model of Simple type theory with a parametric quantifier)**.**

- $\mathcal{N}_{Type} = \mathcal{N}_{Kind} = \mathcal{V}$,
- $\mathcal{N}_{\Pi x:C\ D}$ is the set $\mathcal{F}(\mathcal{N}_C, \mathcal{N}_D)$, except if $\mathcal{N}_D = \{e\}$, in which case $\mathcal{N}_{\Pi x:C\ D} = \{e\}$,
- $\mathcal{N}_{type} = \mathcal{U}$,
- $\mathcal{N}_\iota = \mathcal{N}_o = \mathcal{N}_{arrow} = \mathcal{N}_\Rightarrow = \mathcal{N}_\forall = \mathcal{N}_\eta = \mathcal{N}_\varepsilon = \{e\}$,
- $\mathcal{N}_x = \{e\}$,
- $\mathcal{N}_{\lambda x:C\ t} = \mathcal{N}_t$,
- $\mathcal{N}_{(t\ u)} = \mathcal{N}_t$.

- $\mathcal{M}_{Kind,\psi} = \mathcal{M}_{Type,\psi} = \mathcal{B}$,
- $\mathcal{M}_{\Pi x:C\ D,\psi,\phi}$ is the set of functions f mapping $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$ to an element of $\mathcal{M}_{D,(\psi,x=c')}$, except if for all c' in $\mathcal{N}_C$, $\mathcal{M}_{D,(\psi,x=c')} = \{e\}$, in which case $\mathcal{M}_{\Pi x:C\ D,\psi} = \{e\}$,
- $\mathcal{M}_{type,\psi} = \mathcal{B}$,
- $\mathcal{M}_{\eta,\psi}$ is the function of $\mathcal{F}(\mathcal{U}, \mathcal{V})$ mapping S to S,
- $\mathcal{M}_{\varepsilon,\psi}$ is the function of $\mathcal{F}(\{e\}, \mathcal{V})$, mapping e to $\{e\}$,
- $\mathcal{M}_{\iota,\psi} = \{e\}$,
- $\mathcal{M}_{o,\psi} = \mathcal{B}$,
- $\mathcal{M}_{arrow,\psi}$ is the function mapping S and T in $\mathcal{U}$ to the set $\mathcal{F}(\{e\} \times S, T)$, except if $T = \{e\}$ in which case it maps S and T to $\{e\}$,
- $\mathcal{M}_{\Rightarrow,\psi} = \mathcal{M}_{\forall,\psi} = e$,
- $\mathcal{M}_{x,\psi} = \psi x$,
- $\mathcal{M}_{\lambda x:C\ t,\psi}$ is the function mapping c in $\mathcal{N}_C$ to $\mathcal{M}_{t,(\psi,x=c)}$, except if for all c in $\mathcal{N}_C$, $\mathcal{M}_{t,(\psi,x=c)} = e$, in which case $\mathcal{M}_{\lambda x:C\ t,\psi} = e$,
- $\mathcal{M}_{(t\ u),\psi} = \mathcal{M}_{t,\psi}\ \mathcal{M}_{u,\psi}$, except if $\mathcal{M}_{t,\psi} = e$ in which case $\mathcal{M}_{(t\ u),\psi} = e$.

- $[\![Kind]\!]_{\psi,\phi} = [\![Type]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![\Pi x : C\ D]\!]_{\psi,\phi} = \tilde{\Pi}([\![C]\!]_{\psi,\phi}, \{[\![D]\!]_{(\psi,x=c'),(\phi,x=c)} \mid c' \in \mathcal{N}_C, c \in \mathcal{M}_{C,\psi}\})$,
- $[\![type]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![\iota]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![o]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![arrow]\!]_{\psi,\phi}$ is the function from $\mathcal{U} \times \mathcal{B}$ and $\mathcal{U} \times \mathcal{B}$ to $\mathcal{B}$ mapping $\langle S, a\rangle$ and $\langle T, b\rangle$ to $\tilde{\Pi}(a, \{b\})$,
- $[\![\Rightarrow]\!]_{\psi,\phi}$ is the function $\{e\} \times \mathcal{B}$ and $\{e\} \times \mathcal{B}$ to $\mathcal{B}$ mapping $\langle e, a\rangle$ and $\langle e, b\rangle$ to $\tilde{\Pi}(a, \{b\})$,
- $[\![\forall]\!]_{\psi,\phi}$ is the function mapping $\langle S, a\rangle$ in $\mathcal{U} \times \mathcal{B}$, and $\langle e, g\rangle$ in $\{e\} \times \mathcal{F}(\{e\} \times S, \mathcal{B})$ to $\tilde{\Pi}(a, \{(g\ \langle e, s\rangle) \mid s \in S\})$,
- $[\![\eta]\!]_{\psi,\phi}$ is the function from $\mathcal{U} \times \mathcal{B}$ to $\mathcal{B}$, mapping $\langle S, a\rangle$ to a,
- $[\![\varepsilon]\!]_{\psi,\phi}$ is the function from $\{e\} \times \mathcal{B}$ to $\mathcal{B}$, mapping $\langle e, a\rangle$ to a,
- $[\![x]\!]_{\psi,\phi} = \phi x$,
- $[\![\lambda x : C\ t]\!]_{\psi,\phi}$ is the function mapping $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$ to $[\![t]\!]_{(\psi,x=c'),(\phi,x=c)}$, except if for all $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$, $[\![t]\!]_{(\psi,x=c'),(\phi,x=c)} = e$, in which case $[\![\lambda x : C\ t]\!]_{\psi,\phi} = e$,
- $[\![(t\ u)]\!]_{\psi,\phi} = [\![t]\!]_{\psi,\phi}\ \langle \mathcal{M}_{u,\psi}, [\![u]\!]_{\psi,\phi}\rangle$, except if $[\![t]\!]_{\psi,\phi} = e$, in which case $[\![(t\ u)]\!]_{\psi,\phi} = e$.

We prove that if $\Gamma \vdash t : B$ and ψ is a function mapping the variables $x : A$ of Γ to elements of $\mathcal{N}_A$, then $\mathcal{M}_{t,\psi} \in \mathcal{N}_B$. That if $\Gamma \vdash t : B$, ψ is a function mapping variables $x : A$ of Γ to elements of $\mathcal{N}_A$, and ϕ is a function mapping variables $x : A$ of Γ to elements of $\mathcal{M}_{A,\psi}$, then $[\![t]\!]_{\psi,\phi} \in \mathcal{M}_{B,\psi}$ and $[\![(u/x)t]\!]_{\psi,\phi} = [\![t]\!]_{\psi,x=\mathcal{M}_{u,\psi}\phi,x=[\![u]\!]_{\psi,\phi}}$. And that if $t \equiv_{\beta\mathcal{R}} u$ then $\mathcal{N}_t = \mathcal{N}_u$, $\mathcal{M}_{t,\psi} = \mathcal{M}_{u,\psi}$, and $[\![t]\!]_{\psi,\phi} = [\![u]\!]_{\psi,\phi}$. We thus get the following theorem.

▶ **Theorem 18.** *Simple type theory with a parametric quantifier is super-consistent.*

Note that the set $\mathcal{V}$, thus an inaccessible cardinal, are not really needed to prove the super-consistency of Simple type theory with a parametric quantifier if we can adapt the notion of model in such a way that the family $\mathcal{N}$ is defined for type families only. The systematic development of this notion of partial interpretation is left for future work.

A similar proof for the Calculus of constructions is given in the long version of the paper.

5 Termination of proof reduction

We finally prove that proof reduction terminates in the λΠ-calculus modulo any super-consistent theory.

In Deduction modulo theory, we can define a congruence with non terminating rewrite rules, without affecting the termination of proof reduction. For instance, the rewrite rule $c \longrightarrow c$ does not terminate, but the congruence it defines is the identity and proofs modulo this congruence are just proofs in pure Predicate logic. Thus, proof reduction in Deduction modulo this congruence terminates. So, in the λΠ-calculus modulo this congruence, the β-reduction terminates, but the $\beta\mathcal{R}$-reduction does not, as the $\mathcal{R}$-reduction alone does not terminate. Here, we restrict to prove the termination of β-reduction, not $\beta\mathcal{R}$-reduction. In some cases, like for the three theories presented above, the termination of the $\beta\mathcal{R}$-reduction is a simple corollary of the termination of the β-reduction. In some others, it is not.

The main notion used in this proof is that of *reducibility candidate* introduced by Girard [15]. Our inductive definition, however, follows that of Parigot [22].

▶ **Definition 19** (Candidates). The set $\tilde{\top}$ is defined as the set of strongly terminating terms.

Let C be a set of terms and S be a set of sets of terms. The set $\tilde{\Pi}(C, S)$ is defined as the set of strongly terminating terms t such that if $t \longrightarrow^*_\beta \lambda x : A\ t'$ then for all t'' in C, and for all D in S, $(t''/x)t' \in D$.

Candidates are inductively defined by the three rules

- the set $\tilde{\top}$ of all strongly terminating terms is a candidate,
- if C is a candidate and S is a set of candidates, then $\tilde{\Pi}(C, S)$ is a candidate,
- if S is a non empty set of candidates, then $\bigcap S$ is a candidate.

We write $\mathcal{C}$ for the set of candidates. The algebra $\langle \mathcal{C}, \leq, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{C}), \tilde{\Pi}\rangle$, where $\leq$ is the trivial relation such that $C \leq C'$ always, and $\tilde{\wedge}$ is any function from $\mathcal{C} \times \mathcal{C}$ to $\mathcal{C}$, for instance the constant function equal to $\tilde{\top}$, is a full Π-algebra. It is ordered by the subset relation and complete for this order. If C is a candidate, then all the elements of C strongly terminate.

Consider a super-consistent theory $\Sigma, \mathcal{R}$. We want to prove that β-reduction terminates in the λΠ-calculus modulo this theory.

As usual, we want to associate a candidate $[\![A]\!]$ to each term A in such a way that if t is a term of type A, then $t \in [\![A]\!]$. In the λΠ-calculus modulo theory, the main difficulty is to assign a candidates to terms in such a way that if $A \equiv B$ then $[\![A]\!] = [\![B]\!]$. For instance, if we have the rule $P \longrightarrow P \Rightarrow P$ that permits to type all lambda-terms, including non terminating ones, we should associate, to the term P, a candidate C such that $C = C \tilde{\Rightarrow} C$,

but there is no such candidate. For super-consistent theories, in contrast, such an assignment exists, as the theory has a model $\mathcal{M}$ valued in the Π-algebra $\langle \mathcal{C}, \leq, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{C}), \tilde{\Pi} \rangle$. In this model, if t is a term of type B in some context Γ, then $[\![B]\!]_{\overline{\phi}}$ is a candidate.

We then prove that if Γ is a context, $\overline{\phi} = \phi_1, ..., \phi_n$ is be a sequence of valuations, σ is a substitution mapping every $x : A$ of Γ to an element of $[\![A]\!]_{\overline{\phi}}$ and t is a term of type B in Γ, then $\sigma t \in [\![B]\!]_{\overline{\phi}}$. We thus get the following theorem.

▶ **Theorem 20.** *In a super-consistent theory $\Sigma, \equiv$, all well-typed terms strongly terminate.*

Acknowledgements. The author wants to thank Frédéric Blanqui for very helpful remarks on a previous version of this paper.

References

1. A. Assaf, G. Burel, R. Cauderlier, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard. Dedukti: a logical framework based on the lambda-Pi-calculus modulo theory. `http://www.lsv.ens-cachan.fr/~dowek/Publi/expressing.pdf`, 2016.
2. A. Assaf, G. Dowek, J.-P. Jouannaud, and J. Liu. Untyped confluence in dependent type theories. Submitted to publication, 2017.
3. A. Bauer, G. Gilbert, P. Haselwarter, M. Pretnar, and Ch. A. Stone. Design and implementation of the andromeda proof assistant. Types, 2016.
4. F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.
5. A. Brunel, O. Hermant, and C. Houtmann. Orthogonality and boolean algebras for deduction modulo. In L. Ong, editor, *Typed Lambda Calculus and Applications*, volume 6690 of *Lecture Notes in Computer Science*, pages 76–90. Springer-Verlag, 2011.
6. H. Cirstea, L. Liquori, and B. Wack. Rewriting calculus with fixpoints: Untyped and first-order systems. In *Types*, volume 3085 of *Lectures Notes in Computer Science*. Springer-Verlag, 2003.
7. T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, pages 95–120, 1988.
8. D. Cousineau and G. Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. In S. Ronchi Della Rocca, editor, *Typed lambda calculi and applications*, volume 4583 of *Lecture Notes in Computer Science*, pages 102–117. Springer-Verlag, 2007.
9. G. Dowek. Truth values algebras and proof normalization. In Th. Altenkirch and C. McBride, editors, *Types for proofs and programs*, volume 4502 of *Lecture Notes in Computer Science*, pages 110–124. Springer-Verlag, 2007.
10. G. Dowek, Th. Hardin, and C. Kirchner. Hol-lambda-sigma: an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11:1–25, 2001.
11. G. Dowek, Th. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31:33–72, 2003.
12. G. Dowek and B. Werner. Proof normalization modulo. The Journal of Symbolic Logic, 68(4):1289–1316, 2003.
13. S. Foster and G. Struth. Integrating an automated theorem prover into agda. In M. Bobaru, K. Havelund, G.J. Holzmann, and R. Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*. Springer-Verlag, 2011.
14. H. Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In P. Dybjer, , B. Nordström, and J. Smith, editors, *Types for Proofs and Programs*, volume 996 of *Lecture Notes in Computer Science*, pages 14–38. Springer-Verlag, 1995.

15 J. Y. Girard. *Interprétation Fonctionnelle et Élimination des Coupures dans l'Arithmétique d'Ordre Supérieur*. PhD thesis, Université de Paris VII, 1972.

16 R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

17 P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

18 P.-A. Melliès and B. Werner. A generic normalisation proof for pure type systems. In E. Giménez and Ch. Paulin-Mohring, editors, *Types for Proofs and Programs*, volume 1512 of *Lecture Notes in Computer Science*, pages 254–276. Springer-Verlag, 1998.

19 A. Miquel and B. Werner. The not so simple proof-irrelevant model of CC. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs*, pages 240–258. Springer-Verlag, 2003.

20 Q. H. Nguyen, C. Kirchner, and H. Kirchner. External rewriting for skeptical proof assistants. *Journal of Automated Reasoning*, 29(309), 2002.

21 B. Nordström, K. Petersson, and J. M. Smith. Martin-löf's type theory. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 1–37. Clarendon Press, 2000.

22 M. Parigot. Proofs of strong normalization for second order classical natural deduction. In *Logic in Computer Science*, pages 39–46, 1993.