The not so simple proof-irrelevant model of CC

Alexandre Miquel[1] and Benjamin Werner[2]

[1] Laboratoire de Recherche en Informatique, Université Paris-Sud, F - 91 405 ORSAY cedex, France
Alexandre.Miquel@lri.fr
[2] INRIA-Rocquencourt, BP 105, F - 78 153 LE CHESNAY cedex, France
Benjamin.Werner@inria.fr

Abstract. It is well-known that the Calculus of Constructions (CC) bears a simple set-theoretical model in which proof-terms are mapped onto a single object—a property which is known as *proof-irrelevance*. In this paper, we show that when going into the (generally omitted) technical details, this naive model raises several unexpected difficulties related to the interpretation of the impredicative level, especially for the soundness property which is surprisingly difficult to be given a *correct* proof in this simple framework. We propose a way to tackle these difficulties, thus giving a (more) detailed elementary consistency proof of CC without going back to a translation to F_ω. We also discuss some possible alternatives and possible extensions of our construction.

1 Introduction

Typed λ-calculi essentially describe the definition and the interaction of functions. When building a model for a type system, the denotation of $A \to B$ corresponds to a set of mappings from the denotations of A to the denotations of B. The idea of set-theoretical models is to simply choose the set of all such (set-theoretical) functions.

Such models are not very interesting from the proof-theoretical point of view but they have at least the virtue of simplicity. The validity of an axiom or an extension of the type system in this model is generally easy to justify. This point alone is important enough in modern complex type theories as implemented in proof-processing tools (Coq, Lego, PVS, NuPRL, Agda...), especially when dealing with formalizations of pieces of mathematics which can require some axiomatization.

Some of these type theories are *impredicative*, meaning a proposition may be formed by quantification over all propositions. Technically, this means they are extensions of Girard's system F [11]. As pointed out by Reynolds [17], the only way to give a set-theoretic account for system F (and thus its extensions) is to identify all elements of a given type in the denotation. Thus types are either interpreted by the empty set, or a singleton with a canonical element.

The Calculus of Constructions (CC) is one of the most well-known extensions of system F. In CC, a type is a term which is either of type $*$ or of type $\Box$. Types

of type $*$ are impredicative, that is $\forall x : * \,.\, x$ is also of type $*$. On the other hand, quantification over $\Box$ is prohibited for building objects of type $\Box$. One therefore talks of the *impredicative level* for objects of type $*$ and their elements, and of the *predicative level* for objects of type $\Box$ and their elements. This distinction of different levels is kept in extensions of CC with additional universes and inductive types.

There are two ways to look at the Calculus of Constructions and most of its extensions. In the first, the objects are defined on the impredicative level; this is typically the case when integers are defined impredicatively as Church numerals. In this vision, the calculus is essentially an impredicative variant of Martin-Löf's type theories. In the second, objects are axiomatized or defined on the predicative level; the impredicative level is therefore entirely dedicated to propositions and proofs. This amounts to view the calculus as an extension of Church's Higher-Order Logic (HOL) with proofs as objects.

Since any set-theoretical model will identify the Church numerals 0 and 1, it is clearly not interesting for the first vision of CC. It is however interesting for the second case; indeed, set-theoretical models are the easiest way to prove the coherence of CC, and many of its extensions provided these extensions take place on the predicative level and bear interpretations as sets. Such interpretations are called *proof-irrelevant*, since the content of the proof object is not relevant to the interpretation. We can also mention that proof-irrelevance can have some other positive side effects, we try to give a hint in to that respect in the conclusion.

Indeed, it has been repeatedly pointed out that the trivial boolean model of system F can be extended to the Calculus of Constructions, with a standard set-theoretical interpretation of the predicative level(s). However, maybe because it was not considered interesting, little attention has been paid to the details of its definition. In the next section, we try to explicitate some non trivial difficulties when doing so, mainly in the interaction of the predicative and impredicative levels.

In section three we give a slight reformulation of CC which allows us to tackle the soundness proof in section 4. We end up by discussing some alternative solutions and some consequences.

In what follows, we try to be as precise as possible, but for the usual choice between de Bruijn indices and named variables. Both choices require specific technical developments (see respectively [5] and [15]). We believe the present work can be fitted in both formalization styles.

2 A problematic proof-irrelevant model

2.1 The Calculus of Constructions

In the following, we shall assume that the calculus of constructions (CC) and more generally the theory of pure type systems (PTS) is familiar to the reader. We will denote by $*$ and $\Box$ the two sorts of CC, which are related by the axiom

$* : \Box$ and by the four rules

$(*, *)$	(propositions)
$(\Box, *)$	(polymorphism)
$(*, \Box)$	(dependent types)
$(\Box, \Box)$	(type constructors)

The syntax of raw-terms is given by

Sorts			s	$::=$	$*$	$\mid$	$\Box$			
Terms	t, u, T, U	$::=$	x	$\mid$	s	$\mid$	$\Pi x : T \,.\, U$	$\mid$	$\lambda x : T \,.\, t$	$\mid$	$tu.$

In the following, we shall denote by $t\{x := u\}$ the term obtained by substituting the term u to each free occurrence of the variable x in t. The *one step β-reduction* is defined as usual and will be denoted by $\rightarrow_\beta$. Its reflexive and transitive closure is denoted by $\twoheadrightarrow_\beta$, and the *β-conversion* equivalence by $=_\beta$. As in any other PTS [3], the β-reduction enjoys the Church-Rosser property :

If $t_1 =_\beta t_2$, then there exists t' such that $t_1 \twoheadrightarrow_\beta t'$ and $t_2 \twoheadrightarrow_\beta t'$.

Typing contexts are given by

Contexts	Γ	$::=$	$[]$	$\mid$	$\Gamma; [x : T]$	

We will not recall the typing rules of CC that can be found in [7, 9, 3], and we will just mention the subject-reduction property:

If $\Gamma \vdash t : T$ and $t \twoheadrightarrow_\beta t'$, then $\Gamma \vdash t' : T$.

2.2 Defining the model

The proof-irrelevant model of CC is based on the simple idea that each type-theoretical construct should be interpreted by its obvious set-theoretical equivalent. In this way, the dependent product $\Pi x : T \,.\, U_x$ (we add the subscript x in U_x to emphasize the dependency) is interpreted by the generalized cartesian product

$$\prod_{x \in T} U_x \quad = \quad \big\{ f \text{ function}; \quad \mathrm{Dom}(f) = T \quad \wedge \quad \forall x \in T \quad f(x) \in U_x \big\},$$

the λ-abstraction $\lambda x : T \,.\, t_x$ is interpreted by the set-theoretical abstraction that will be denoted by $(x_{\in T} \mapsto t_x)$ in the following, and the application tu is interpreted by the function application $t(u)$ of set-theory.

Interpreting proofs and propositions The delicate point here is of course the interpretation of proof-terms and propositions. In order to achieve proof-irrelevance, all proof-terms have to be interpreted by a single object $\bullet$, and consequently, propositions have to be interpreted either by the empty set $\varnothing$ ('falsity') or by the singleton $\{\bullet\}$ ('truth'), so that we naturally recover the classical interpretation of propositions as truth-values.

Let us recall that in CC, the sort $*$ is *impredicative*, since the dependent product $\Pi x:T\,.\,U_x$ of a family of propositions U_x indexed by some type T is still a proposition, even if T has type $\Box$. Regarding this point, it is important to notice that interpreting naively the dependent product as the generalized cartesian product of set theory is not completely sound. Indeed, if $(U_x)_{x\in T}$ is a family of sets indexed by some set T such that $U_x = \varnothing$ or $U_x = \{\bullet\}$ for all $x \in T$, then it is easy to check that

$$\prod_{x\in T} U_x \quad = \quad \begin{cases} \varnothing & \text{if } U_x = \varnothing \text{ for some } x \in T \\ \big\{(x_{\in T} \mapsto \bullet)\big\} & \text{if } U_x = \{\bullet\} \text{ for all } x \in T \end{cases}$$

so that in the second case, we do not obtain the expected singleton $\{\bullet\}$, but another singleton that contains the constant function mapping $\bullet$ to any $x \in T$.

Identifying constant proof-functions To keep a uniform interpretation of the dependent product (remember that the PTS-style syntax of CC makes no distinction between the different kinds of dependent product), the last remark shows that we have to identify all the constant functions of the form $(x_{\in T} \mapsto \bullet)$ with the object $\bullet$ itself. (Since the problem we want to point out comes from this identification, it is worth to say a few words about it.) Therefore, let us set

$$\mathsf{lam}(f) \quad = \quad \begin{cases} \bullet & \text{if } f(x) = \bullet \text{ for all } x \in \mathrm{Dom}(f) \\ f & \text{otherwise} \end{cases}$$

for any function f (the notation $\mathsf{lam}(f)$ being undefined otherwise). Intuitively, the operator lam acts as a very simple encoding mechanism that replaces any constant function of the form $(x_{\in T} \mapsto \bullet)$ by the proof-object $\bullet$ itself, and that leaves f unchanged otherwise. To prevent undesirable identifications, we have to assume that

Convention 1 — *The proof-object $\bullet$ is not a set-theoretical function*[3].

Symmetrically, the corresponding decoding operation is performed through the function application by setting

$$\mathsf{app}(u, x) \quad = \quad \begin{cases} \bullet & \text{if } u = \bullet \\ u(x) & \text{otherwise} \end{cases}$$

[3] Taking $\bullet \equiv \{\varnothing\}$ is a possible choice which fits all conventions throughout the article. Indeed, the empty set is not the set-theoretical code of pair (usually encoded as $\{\{a\};\{a;b\}\}$), hence the singleton $\{\varnothing\}$ is not the code of a set-theoretical function.

so that the proof-object $\bullet$ now behaves as a constant function mapping any object of the universe to $\bullet$ itself. Notice that the notation $\mathsf{app}(u,x)$ is defined when either $u = \bullet$ or u is a function and $x \in \mathrm{Dom}(u)$, and that both cases are disjoint due to our convention. It is then straightforward to check that

Fact 2 — *For any function f and for any object $x \in \mathrm{Dom}(f)$, the notation $\mathsf{app}(\mathsf{lam}(f),x)$ is defined and*

$$\mathsf{app}(\mathsf{lam}(f),x) = f(x)\,.$$

At this point, it is important to notice that the domain of f is completely lost via the encoding $f \mapsto \mathsf{lam}(f)$ when f is a constant function returning the proof-object. As a consequence, the notation $\mathsf{app}(\mathsf{lam}(f),x)$ may be defined whereas $f(x)$ is not, which is the case when f is a constant function returning the proof-object and when $x \notin \mathrm{Dom}(f)$. This remark will have a crucial importance when we will consider the problem of soundness of β-reduction in paragraph 2.4.

Finally, since we only want to work with encoded functions throughout the model construction, it is convenient to also define a modified cartesian product

$$\widehat{\prod_{x\in T}} U_x \quad = \quad \Big\{ \mathsf{lam}(f);\quad f \in \prod_{x\in T} U_x \Big\}$$

in which all the expected functions are replaced by their encoded form.

Interpreting the predicative sort To interpret the sort $\Box$, we simply have to consider a set $\mathcal{U}$ such that

1. $\varnothing \in \mathcal{U}$, $\{\bullet\} \in \mathcal{U}$ and $\big\{\varnothing;\{\bullet\}\big\} \in \mathcal{U}$;
2. $\mathcal{U}$ is closed under the (modified) generalized cartesian product, that is

$$T \in \mathcal{U} \quad \wedge \quad (\forall x \in T \;\; U_x \in \mathcal{U}) \quad \Rightarrow \quad \widehat{\prod_{x\in T}} U_x \;\in\; \mathcal{U}$$

for any family of sets $(U_x)_{x\in T}$.

A very simple candidate for $\mathcal{U}$ is the denumerable set V_ω of all *hereditarily finite sets*. (To fulfill the first condition, we have to assume that the proof-object $\bullet$ is hereditarily finite.) Notice that if we choose $\mathcal{U} = V_\omega$, we then obtain a model of CC in which the denotations of all types are finite (thus refuting the possibility of building any provably infinite data type in the formalism). On the other hand, it is also possible to choose $\mathcal{U}$ such that $\mathcal{U}$ contains at least an infinite set (for instance ω), but in this case, the set $\mathcal{U}$ can only be built under the assumption of the existence of an inaccessible cardinal—that cannot be derived in Zermelo-Fraenkel set theory [13].

2.3 Defining the interpretation function

The interpretation function is organized as follows:

- To each context Γ we associate a set $[\![\Gamma]\!]$ of Γ-*valuations*, that is lists of objects of the form $(\cdots((\mathsf{nil}, \alpha_1), \alpha_2)\cdots, \alpha_n)$ where n is the length of Γ. (Consing is performed rightwards as for contexts.)
- To each pair (Γ, t) formed by a context Γ and a term t, we associate a function $[\![\Gamma \vdash t]\!]$ which is *partially* defined on $[\![\Gamma]\!]$ (i.e. a function which is defined on a subset of $[\![\Gamma]\!]$).

Since the denotations $[\![\Gamma]\!]$ and $[\![\Gamma \vdash t]\!]$ will be defined by mutual induction on the size of their argument, we have to precise what we mean by the size of a context Γ and what we mean by the size of a pair $\Gamma \vdash t$:

- The size $|t|$ of a term t is (recursively) defined as the sum of the sizes of its immediate subterms plus 1. (In particular, $|t| \geq 1$ for any term t.)
- The size of a context $\Gamma = [x_1 : T_1; \ldots; x_n : T_n]$ is defined by

$$|\Gamma| = |T_1| + \cdots + |T_n| + 1/2\,.$$

- The size of a pair $\Gamma \vdash t$ (where $\Gamma = [x_1 : T_1; \ldots; x_n : T_n]$) is defined by

$$|\Gamma \vdash t| \;\; = \;\; |T_1| + \cdots + |T_n| + |t| \;\; = \;\; |\Gamma| + |t| - 1/2\,.$$

The '$\pm 1/2$' in the definitions of $|\Gamma|$ and $|\Gamma \vdash t|$ simply ensure that we have the strict inequalities $|\Gamma \vdash t| < |\Gamma; [x : t]|$ and $|\Gamma| < |\Gamma \vdash t|$ for all Γ and for all t. (These functions thus take their values in the set of positive semi-integers, which is still well-ordered.)

For each context Γ, the set $[\![\Gamma]\!]$ is defined by the following equations:

$$\begin{aligned}
[\![[]]\!] &= \{\mathsf{nil}\} \\
[\![\Gamma; [x : T]]\!] &= \big\{(\gamma, \alpha);\;\; \gamma \in [\![\Gamma]\!] \;\wedge\; [\![\Gamma \vdash T]\!]_\gamma \text{ defined } \wedge\; \alpha \in [\![\Gamma \vdash T]\!]_\gamma\big\}
\end{aligned}$$

Finally, the interpretation $[\![\Gamma \vdash t]\!]$ of a term t in a context Γ is defined for all $\gamma \in [\![\Gamma]\!]$ by:

$$\begin{aligned}
[\![\Gamma \vdash *]\!]_\gamma &= \{\varnothing;\; \{\bullet\}\} \\
[\![\Gamma \vdash \Box]\!]_\gamma &= \mathcal{U} \\
[\![\Gamma \vdash x]\!]_{(\cdots(\mathsf{nil}, \alpha_1)\cdots, \alpha_n)} &= \alpha_i \qquad \text{(if } x \text{ is the } i\text{-th declared variable in } \Gamma\text{)} \\
[\![\Gamma \vdash \Pi x : T\,.\,U]\!]_\gamma &= \widehat{\prod_{\alpha \in [\![\Gamma \vdash T]\!]_\gamma}} [\![\Gamma; [x : T] \vdash U]\!]_{(\gamma, \alpha)} \\
[\![\Gamma \vdash \lambda x : T\,.\,t]\!]_\gamma &= \mathsf{lam}\big(\alpha_{\in [\![\Gamma \vdash T]\!]_\gamma} \mapsto [\![\Gamma; [x : T] \vdash t]\!]_{(\gamma, \alpha)}\big) \\
[\![\Gamma \vdash tu]\!]_\gamma &= \mathsf{app}\big([\![\Gamma \vdash t]\!]_\gamma,\; [\![\Gamma \vdash u]\!]_\gamma\big)
\end{aligned}$$

(Notice that $[\![\Gamma]\!]$ is always defined before $[\![\Gamma \vdash t]\!]$.) It is important to make precise the cases where the denotation $[\![\Gamma \vdash t]\!]_\gamma$ is defined:

– $[\![\Gamma \vdash *]\!]_\gamma$ and $[\![\Gamma \vdash \Box]\!]_\gamma$ are always defined.
– $[\![\Gamma \vdash x]\!]_\gamma$ is defined if x is declared in Γ.
– $[\![\Gamma \vdash \Pi x : T \,.\, U]\!]_\gamma$ is defined if $[\![\Gamma \vdash T]\!]_\gamma$ is defined and $[\![\Gamma; [x : T] \vdash U]\!]_{(\gamma,\alpha)}$ is defined for all $\alpha \in [\![\Gamma \vdash T]\!]_\gamma$.
– $[\![\Gamma \vdash \lambda x : T \,.\, t]\!]_\gamma$ is defined if $[\![\Gamma \vdash T]\!]_\gamma$ is defined and $[\![\Gamma; [x : T] \vdash t]\!]_{(\gamma,\alpha)}$ is defined for all $\alpha \in [\![\Gamma \vdash T]\!]_\gamma$.
– $[\![\Gamma \vdash tu]\!]_\gamma$ is defined when $[\![\Gamma \vdash t]\!]_\gamma$ and $[\![\Gamma \vdash u]\!]_\gamma$ are defined, and when either $[\![\Gamma \vdash t]\!]_\gamma = \bullet$ or $[\![\Gamma \vdash t]\!]_\gamma$ is a function and $[\![\Gamma \vdash u]\!]_\gamma$ belongs to its domain.

Let us denote by $\Gamma; \Delta$ (resp. γ, δ) the concatenation of contexts (resp. valuations). We can now prove the following property:

Lemma 1 (Substitutivity). — *Let Γ be a context and let u and U be terms such that $[\![\Gamma \vdash u]\!]_\gamma \in [\![\Gamma \vdash U]\!]_\gamma$ for some $\gamma \in [\![\Gamma]\!]$ (by assuming that both members of the former equality are defined). Then for any term t such that $[\![\Gamma; [x : U] \vdash t]\!]_{(\gamma, [\![\Gamma \vdash u]\!]_\gamma)}$ is defined, the denotation $[\![\Gamma \vdash t\{x := u\}]\!]_\gamma$ is defined and*

$$[\![\Gamma \vdash t\{x := u\}]\!]_\gamma \;=\; [\![\Gamma; [x : U] \vdash t]\!]_{(\gamma, [\![\Gamma \vdash u]\!]_\gamma)} \,.$$

Proof. Let $\alpha = [\![\Gamma \vdash u]\!]_\gamma$ (Γ, u, U and γ being fixed for the rest of the proof). To prove the expected implication, one has to strengthen its statement as follows. Let $P(\Delta)$ be the predicate defined by

$$\begin{array}{ll} P(\Delta) & \equiv \\ \quad \forall \delta & (\gamma, \alpha), \delta \in [\![\Gamma; [x : T]; \Delta]\!] \quad \Rightarrow \quad \gamma, \delta \in [\![\Gamma; \Delta\{x := u\}]\!] \end{array}$$

for all context Δ, and $Q(\Delta \vdash t)$ the predicate defined by

$$\begin{array}{lll} Q(\Delta \vdash t) & \equiv & \\ \quad \forall \delta, t & (\gamma, \alpha), \delta \in [\![\Gamma; [x : U]; \Delta]\!] \;\wedge\; [\![\Gamma; [x : U]; \Delta \vdash t]\!]_{(\gamma,\alpha),\delta} \text{ defined} \\ & \Rightarrow \;\; [\![\Gamma; \Delta\{x := u\} \vdash t\{x := u\}]\!]_{\gamma,\delta} \text{ defined} \;\wedge & \\ & \quad\; [\![\Gamma; \Delta\{x := u\} \vdash t\{x := u\}]\!]_{\gamma,\delta} \;=\; [\![\Gamma; [x : U]; \Delta \vdash t]\!]_{(\gamma,\alpha),\delta} \end{array}$$

for all pair (Δ, t) formed by a context Δ and a term t. The assertions $P(\Delta)$ and $Q(\Delta \vdash t)$ are then proved for each Δ and t by a mutual induction on the size of their argument. (In particular, $P(\Delta)$ is proved before $Q(\Delta, t)$ for all t.) The expected result is then given by $Q([], t)$. □

2.4 The unattainable soundness

We now come to our main problem, which is to prove the soundness property that can be formulated as follows:

> *If $\Gamma \vdash t : T$, then for all $\gamma \in [\![\Gamma]\!]$ the denotations $[\![\Gamma \vdash t]\!]_\gamma$ and $[\![\Gamma \vdash T]\!]_\gamma$ are defined and $[\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash T]\!]_\gamma$.*

The natural way to prove this property is to proceed by induction over the derivation of the judgement $\Gamma \vdash t : T$. When writing down the proof completely, it appears that all the cases successfully pass the test, except for the conversion rule

$$\text{(Conv)} \qquad \frac{\Gamma \vdash t : T \qquad \Gamma \vdash T' : s}{\Gamma \vdash t : T'} \quad T' =_\beta T$$

In this case, we know (by the induction hypothesis) that for a given valuation $\gamma \in [\![\Gamma]\!]$, the denotations $[\![\Gamma \vdash t]\!]_\gamma$, $[\![\Gamma \vdash T]\!]_\gamma$ and $[\![\Gamma \vdash T']\!]s$ are defined, and that

$$[\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash T]\!]_\gamma \qquad \text{and} \qquad [\![\Gamma \vdash T']\!]_\gamma \in [\![\Gamma \vdash s]\!]_\gamma \,.$$

To be able to conclude, we first have to check that $[\![\Gamma \vdash T]\!]_\gamma = [\![\Gamma \vdash T']\!]_\gamma$— which seems quite plausible since the terms T and T' are convertible. However, we have to do it very carefully: since our interpretation function is partial, it could be possible that some of the intermediate conversion steps between T and T' bears no denotation (in the context Γ and for the valuation γ) although the denotations $[\![\Gamma \vdash T]\!]_\gamma$ and $[\![\Gamma \vdash T']\!]_\gamma$ are defined.

To escape this uncomfortable situation, it is natural to rely on the Church-Rosser property by considering the following plausible property:

Conjecture 1 (Soundness of β-reduction). — If the denotation $[\![\Gamma \vdash t]\!]_\gamma$ is defined for some $\gamma \in [\![\Gamma]\!]$, and if $t \to_\beta t'$, then $[\![\Gamma \vdash t']\!]_\gamma$ is defined and

$$[\![\Gamma \vdash t']\!]_\gamma = [\![\Gamma \vdash t]\!]_\gamma \,.$$

It is clear that the latter proposition entails the equality $[\![\Gamma \vdash T]\!]_\gamma = [\![\Gamma \vdash T']\!]_\gamma$ under the assumption $T =_\beta T'$ (this is obvious from the Church-Rosser property). Unfortunately, the conjecture 1 is wrong, as illustrated by the following counter-example:

A counter-example to the conjecture 1 Let us consider a closed term $T : *$ such that $[\![[] \vdash T]\!] = \{\bullet\}$. (We can take for instance $T = \Pi X : * \,.\, X \to X$.) Let us now consider the ill-typed term $t = (\lambda x : T \,.\, x)*$ (i.e. the identity function on T applied to the sort $*$). We have

$$[\![[] \vdash \lambda x : T \,.\, x]\!] \;=\; \mathsf{lam}\big(\alpha_{\in\{\bullet\}} \mapsto \alpha\big) \;=\; \bullet,$$

so that

$$[\![[] \vdash t]\!] \;=\; \mathsf{app}(\bullet, [\![[] \vdash *]\!]) \;=\; \bullet \,.$$

Of course, the term t reduces to the term $t' = *$, whose denotation is defined and different from $\bullet$. Intuitively, this counter-example can be understood as follows:

1. First, we have built a function f (i.e. the identity function) on a very simple domain (the singleton $\{\bullet\}$). Since this function is constant and returns the proof-object $\bullet$ *on its domain*, the encoding mechanism $f \mapsto \mathsf{lam}(f)$ replaced this function by the proof-object $\bullet$ itself, thus forgetting all information about the domain of f.

2. When we applied the object $\bullet = \mathsf{lam}(f)$ to an arbitrary object x (here the denotation of $*$), we obtained the proof-object as a result—since the proof-object behaves as a constant function *on the whole universe*. Of course, the object x had been chosen outside the domain of f, so that the result is meaningless. (And here different from the denotation of the β-reduced term.)

It is easy to build several variations on this counter-exemple, in order to get two terms t and t' such that $t \to_\beta t'$ and:

- $[\![\Gamma \vdash t]\!]_\gamma$ and $[\![\Gamma \vdash t']\!]_\gamma$ are defined and different; or
- $[\![\Gamma \vdash t]\!]_\gamma$ is defined and $[\![\Gamma \vdash t']\!]_\gamma$ is undefined; or
- $[\![\Gamma \vdash t]\!]_\gamma$ is undefined and $[\![\Gamma \vdash t']\!]_\gamma$ is defined.

However, our counter-example strongly relies on the fact that t is ill-typed, and it would be more desirable to replace the (wrong) conjecture 1 by :

Conjecture 2 (Soundness of β-reduction). — If the denotation $[\![\Gamma \vdash t]\!]_\gamma$ is defined for some $\gamma \in [\![\Gamma]\!]$, if $t \to_\beta t'$ and if t is well-typed in Γ, then $[\![\Gamma \vdash t']\!]_\gamma$ is defined and

$$[\![\Gamma \vdash t']\!]_\gamma = [\![\Gamma \vdash t]\!]_\gamma \,.$$

Of course, it is clear that in order to prove this conjecture, we need the soundness property... that needs this conjecture to be proved! From this, it appears that there is no obvious way to prove the soundness property for the 'simple' model we presented in this section.

Independence w.r.t. the encoding It is important to understand that the problem we presented is not due to the precise encoding/decoding mechanism we introduced in paragraph 2.2, but to the fact that the identification of all proof-terms requires to forget the domain of the corresponding functions. In particular, P. Aczel presents in [1] another mechanism (for the same purpose) which is based on the following definitions

$$\begin{aligned} \bullet \quad &= \varnothing \\ \mathsf{lam}(f) \quad &= \big\{(x,z); \quad x \in \mathrm{Dom}(f) \quad \wedge \quad z \in f(x)\big\} \\ \mathsf{app}(u,x) &= \big\{z; \quad (x,z) \in u\big\} \end{aligned}$$

and it is easy to check that our counter-example works exactly the same way with these new definitions.[4]

Another possibility is to introduce an undefined object so that the interpretation function becomes total. For that, we simply have to set

$$[\![\Gamma \vdash MN]\!]_\gamma \;=\; \mathsf{undefined}$$

when either

[4] It is interesting to notice that this alternative encoding of functions is very close to the encoding of Scott-continuous (or stable) functions by their traces in the theory of concrete Scott-domains (or in the theory of coherence spaces) [12].

– the denotation of M is neither a function nor the proof object, or
– the denotation of M is a set-theoretical function whose domain does not contain the denotation of N.

The other equations defining the interpretation function have then to be adapted in order to propagate the undefined object when one of the subterms is itself denoted by the undefined object (whose behaviour is the one of an exception).

However, considering a *partial* interpretation function (as we did above) or a *total* interpretation function with an undefined object is only a matter of presentation, and it is easy to check that the counter-example still holds in this alternative presentation.

3 The sorted type system

3.1 Definition

As we have seen, the problem lies not so much in the definition of the model, but in stating the subject-reduction property for the interpretation. This property is indeed true for well-typed terms but well-typedness is a too restrictive condition in order for the soundness proof to go through. In order to keep the simplicity of the construction, we propose a looser condition for subject reduction, building on the fact that in a functional PTS (like CC) every well-typed term has at most one sort.

The sort of any term is given unambiguously once the sorts of its free variables are known. We therefore build the model for a *sorted version* of the type system. This idea is due to Geuvers [10] (and is also used in [20] for more syntactical purposes).

Definition 1. *The definition of terms is unchanged, but we take a set of variables indexed by the sorts. In other words, we can describe the algebra of sorted terms by:*

$$s ::= * \mid \Box$$
$$t ::= x_s \mid tt \mid \lambda x_s : t.t \mid \Pi x_s : t.t \mid s$$

where x ranges over some usual set of variables.

An important point is that α-conversion does not allow renaming of sorts, that is $\lambda x_* : *.x_*$ and $\lambda x_\Box : *.x$ are not α-convertible[5]. Of course, the definitions of substitution, β-conversion and its properties remain unchanged.

A consequence of the Church-Rosser property is that:

Corollary 1 (Uniqueness of Type Formation).
If $\Pi x_s : A \,.\, B =_\beta \Pi x_{s'} : A' \,.\, B'$, then $s = s'$,$A =_\beta A'$ and $B =_\beta B'\{x'_s := x_s\}$.

[5] This is indeed more natural in a de Bruijn style formalization.

We write $\Gamma \vdash_{\mathsf{s}} t : T$ for the sorted typing judgements. The rules for $\vdash_{\mathsf{s}}$ are the sames as for $\vdash$ with exception of the rule Weak in which we require the sort of the variable to match the type of its type.

Definition 2 (The sorted typing rules).
The typing rules for the sorted calculus of constructions are:

$$[] \vdash_{\mathsf{s}} * : \Box \ \text{(Axiom)} \qquad \frac{\Gamma \vdash_{\mathsf{s}} A : B}{\Gamma \vdash_{\mathsf{s}} x_s : T}(x_s : T) \in \Gamma \ \text{(Var)}$$

$$\frac{\Gamma \vdash_{\mathsf{s}} T_1 : s_1 \qquad \Gamma;[x_s : T_1] \vdash_{\mathsf{s}} T_2 : s_2}{\Gamma \vdash_{\mathsf{s}} \Pi x_s : T_1.T_2 : s_2} \ \text{(Pi)}$$

$$\frac{\Gamma;[x_s : A] \vdash_{\mathsf{s}} t : B \qquad \Gamma;[x_s : A] \vdash_{\mathsf{s}} B : s'}{\Gamma \vdash_{\mathsf{s}} \lambda x_s : A.t : \Pi x_s : A.B} \ \text{(Lam)}$$

$$\frac{\Gamma \vdash_{\mathsf{s}} t : \Pi x_s : A.B \qquad \Gamma \vdash_{\mathsf{s}} u : A}{\Gamma \vdash_{\mathsf{s}} tu : B\{x_s := u\}} \ \text{(App)}$$

$$\frac{\Gamma \vdash_{\mathsf{s}} t : T \qquad \Gamma \vdash_{\mathsf{s}} A : s}{\Gamma;[x_s : A] \vdash_{\mathsf{s}} t : T} \ \text{(Weak)}$$

$$\frac{\Gamma \vdash_{\mathsf{s}} t : A \qquad \Gamma \vdash_{\mathsf{s}} B : s}{\Gamma \vdash_{\mathsf{s}} t : B} A =_\beta B \ \text{(Conv)}$$

3.2 Basic Metatheory

This presentation is essentially equivalent to the standard PTS one. The basic metatheory is similar to the usual one, but has to be done in the correct order.

Lemma 2. *If $\Gamma \vdash_{\mathsf{s}} t : T$ is derivable, then so is $\Gamma \vdash t : T$.*

Proof. This is a trivial induction over the structure of the derivation, since the rules for $\vdash_{\mathsf{s}}$ are more restrictive than the rules for $\vdash$.

The two following lemmas are proved exactly as their counterparts for regular PTSs, by induction over typing derivation.

Lemma 3 (Substitution). *If $\Gamma;[x_s : A]\Delta \vdash_{\mathsf{s}} t : T$ and $\Gamma \vdash_{\mathsf{s}} u : A$ are derivable, then so is $\Gamma\Delta\{x_s := u\} \vdash_{\mathsf{s}} t\{x_s := u\} : T\{x_s := u\}$.*

Lemma 4 (Stripping or **Inversion).**

If $\Gamma \vdash_{\mathsf{s}} x_s : T$	*then there exist* T' *and* s' *such that* $(x_s : T') \in \Gamma$, $T =_\beta T'$, $\Gamma \vdash_{\mathsf{s}} T : s'$ *and* $\Gamma \vdash_{\mathsf{s}} T' : s$.
If $\Gamma \vdash_{\mathsf{s}} \lambda x_s : A.t : T$	*then there exist* B, s' *and* s'' *such that:* $\Gamma \vdash_{\mathsf{s}} A : s$, $\Gamma; [x_s : A] \vdash_{\mathsf{s}} B : s''$, $\Gamma; [x_s : A] \vdash_{\mathsf{s}} t : B, \Gamma \vdash_{\mathsf{s}} T' : s'$ *and* $T =_\beta \Pi x_s : A.B$.
If $\Gamma \vdash_{\mathsf{s}} \Pi x_s : A.B : T$	*then there exists* s' *such that* $T =_\beta s'$ *and* $\Gamma; [x_s : A] \vdash_{\mathsf{s}} B : s'$. *Furthermore,* $T = s'$ *or* $\Gamma \vdash_{\mathsf{s}} T : s''$ *for some* s''.
If $\Gamma \vdash_{\mathsf{s}} tu : T$	*then there exist* s, A *and* B *such that* $\Gamma \vdash_{\mathsf{s}} t : \Pi x_s : A.B$, $\Gamma \vdash_{\mathsf{s}} u : A$, $T =_\beta B\{x_s := u\}$. *Furthermore* $T = B\{x_s := u\}$ *or* $\Gamma \vdash_{\mathsf{s}} T : s'$ *for some* s'.
$\Gamma \vdash_{\mathsf{s}} \Box : T$	*is not possible.*
If $\Gamma \vdash_{\mathsf{s}} * : T$	*then* $T = \Box$ *or* $T =_\beta \Box$ *and* $\Gamma \vdash_{\mathsf{s}} T : s$ *for some* s.

Using stripping, we can prove the two next results by induction over t (actually over the size of $|t| + |\Gamma|$). Again the proofs are quasi-identical to the ones for regular PTSs, in particular when proving subject reduction, one needs to consider also reductions inside contexts for induction loading.

Corollary 2 (Type uniqueness). *If* $\Gamma \vdash_{\mathsf{s}} t : T$ *and* $\Gamma \vdash_{\mathsf{s}} t : T'$, *then* $T =_\beta T'$.

Lemma 5 (Subject reduction). *If* $\Gamma \vdash_{\mathsf{s}} t : T$ *and* $t \to_\beta t'$, *then* $\Gamma \vdash_{\mathsf{s}} t' : T$.

3.3 Sorting terms and reductions

Using the marking of variables, we are able to define sorts even for non-typed terms. This is the key for a convenient restriction of β-reduction.

Definition 3. *The sort* $s(t)$ *of a term* t *is (partially) defined by the following equations :*

$$s(x_s) \equiv s \qquad s(\lambda x_{s'} : T.u) \equiv s(u)$$
$$s(tu) \equiv s(t) \qquad s(\Pi x_{s'} : T.u) \equiv s(u)$$

A term is called a proof-term *if its sort is* $*$, *it is called a* predicate *if its sort is* $\Box$.

Lemma 6. *Well-typed terms are well-sorted; more precisely: If* $\Gamma \vdash_{\mathsf{s}} t : T$, *then either* $T = \Box$ *or* $\Gamma \vdash_{\mathsf{s}} T : s(t)$.

Proof. By induction over the typing derivation. The cases of the rules Axiom, Var, Pi, Lam and Weak are trivial.

In the case of App the conclusion is $\Gamma \vdash_{\mathsf{s}} tu : B\{x_s := u\}$. We know that $\Gamma \vdash_{\mathsf{s}} \Pi x_s : A \,.\, B : s(t)$, which ensures that $\Gamma; [x_s : A] \vdash_{\mathsf{s}} B : s(t)$ by stripping. Since $\Gamma \vdash_{\mathsf{s}} u : A$, we have $\Gamma \vdash_{\mathsf{s}} B\{x_s := u\} : s(t)$ by substitution.

The case of Conv is more interesting. The conclusion is $\Gamma \vdash_{\mathsf{s}} t : B$; we know that B cannot be $\Box$, since it is well-typed. Therefore A is also different from $\Box$ and thus $\Gamma \vdash A : s(t)$. By Church-Rosser, A and B have a common reduct

C. By subject-reduction, we know that $\Gamma \vdash_{\mathsf{s}} C : s(t)$ (C is a reduct of A) and $\Gamma \vdash_{\mathsf{s}} C : s$ (C is a reduct of B). By type uniqueness $s =_\beta s(t)$ which implies $s = s(t)$ and so $\Gamma \vdash B : s(t)$.

Definition 4. *A β-reduction (resp. a sequence of β-reductions) is well-sorted iff the reduced redex(es) is (are) of the form $(\lambda x_s : A.t\ u)$ with $s(u) = s$.*

Lemma 7. *If t reduces to t' by a well-sorted reduction and $s(t)$ is defined, then so is $s(t')$ and $s(t) = s(t')$.*

Proof. By easy induction over the body of the reduced redex.

Lemma 8. *Any β-reduction of a well-typed term is well-sorted.*

Proof. Consider a redex $(\lambda x_s : A.t\ u)$ well-typed in some context Γ. Using stripping one easily checks that $\Gamma \vdash_{\mathsf{s}} A : s$ and $\Gamma \vdash_{\mathsf{s}} u : A$.

Lemma 9. *If there exists a derivation of $\Gamma \vdash t : T$, then there exists a mapping σ from the (sorted) variables to themselves, such that $\Gamma\{\sigma\} \vdash_{\mathsf{s}} t\{\sigma\} : T\{\sigma\}$ is derivable.*

Proof. When dealing with named variables, one first shows that derivable judgements are stable by α-conversion.

The lemma then is proved by induction over the typing derivation. In order for the induction to go through, one proves that the mapping is unique (modulo α-conversion). This is ensured by the type uniqueness property for regular PTSs, ensuring there is only one way to chose the sort of each variable.

4 Model Construction

We use the same set $\mathcal{U}$ as in section 2 and the interpretations $[\![\Gamma]\!]$ and $[\![\Gamma \vdash_{\mathsf{s}} t]\!]$ are defined using the same well-founded order as in 2.3. We however do not need the coding/decoding operators anymore. Instead, it is easy to proceed by case over the sort of the interpreted term.

The functions $[\![\Gamma]\!]$ and $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]$ are defined by:

$$\begin{aligned}
[\![[]]\!] &\equiv \{\mathsf{nil}\} \\
[\![\Gamma; [x:t]]\!] &\equiv \{(\gamma, \alpha);\ \gamma \in [\![\Gamma]\!] \wedge \alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma\} \\
&\text{if } p \text{ is a proof-term:} \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ p]\!]_\gamma &\equiv \bullet \\
&\text{in other cases:} \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ *]\!]_\gamma &\equiv \{\varnothing;\ \{\bullet\}\} \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ \Box]\!]_\gamma &\equiv \mathcal{U} \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ x_\Box]\!]_{(\ldots(\alpha_1,\alpha_2),\ldots\alpha_n)} &\equiv \alpha_i \qquad \text{if } x_\Box \text{ is the } i\text{th declared variable } \Gamma \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ tu]\!]_\gamma &\equiv [\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma([\![\Gamma \ \vdash_{\mathsf{s}} \ u]\!]_\gamma) \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ \lambda x_s : A.t]\!]_\gamma &\equiv \alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma \ \mapsto \ [\![\Gamma; [x_s : A] \ \vdash_{\mathsf{s}} \ t]\!]_{(\gamma,\alpha)} \\
&\text{if } B \text{ is a predicate:} \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ \Pi x_s : A.B]\!]_\gamma &\equiv \bigcap_{\alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma} [\![\Gamma; [x_s : A] \ \vdash_{\mathsf{s}} \ B]\!]_{(\gamma,\alpha)} \\
&\text{if } B \text{ is not a predicate:} \\
[\![\Gamma \ \vdash_{\mathsf{s}} \ \Pi x_s : A.B]\!]_\gamma &\equiv \Pi_{\alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma} [\![\Gamma \ \vdash_{\mathsf{s}} \ B]\!]_{(\alpha,\gamma)}
\end{aligned}$$

In the above, the interpretation of the predicate $\Pi x_s : A.B$ is not empty if $[\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma$ is. A more explicit formulation would be:

$$[\![\Gamma \ \vdash_{\mathsf{s}} \ \Pi x_s : A.B]\!]_\gamma \equiv \{f \in \{\bullet\} | \forall \alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma \,.\, f \in [\![\Gamma; [x_s : A] \ \vdash_{\mathsf{s}} \ B]\!]_{(\gamma,\alpha)}\}.$$

Like in 2, the use of set-theoretical function application for defining $[\![\Gamma \ \vdash_{\mathsf{s}} \ tu]\!]$ makes these definitions partial. The conditions for being defined are similar, but for the proof-terms.

Lemma 10 (substitutivity). *Let u be a term of sort s. Suppose $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma$ and write $\alpha \equiv [\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma$. Suppose $(\gamma, \alpha), \delta \in [\![\Gamma; [x_s : A]\Delta]\!]$. If $[\![\Gamma; [x_s : A]\Delta \ \vdash_{\mathsf{s}} \ u]\!]_{(\gamma,\alpha),\delta}$ is defined, then*

$$[\![\Gamma; [x_s : A]\Delta \ \vdash_{\mathsf{s}} \ u]\!]_{(\gamma,\alpha),\delta} = [\![\Gamma\Delta\{x_s := t\} \ \vdash_{\mathsf{s}} \ u\{x_s := t\}]\!]_{\gamma,\delta}.$$

The proof is similar to lemma 1.

Lemma 11 (soundness for reduction). *If $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma$ is defined, if $t \to_\beta t'$ by a well-sorted reduction, then $[\![\Gamma \ \vdash_{\mathsf{s}} \ t']\!]_\gamma$ is defined and*

$$[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma = [\![\Gamma \ \vdash_{\mathsf{s}} \ t']\!]_\gamma.$$

Proof. We proceed by structural induction over t. The only interesting cases are when t is reduced at the root.

- If t is a proof term, then so is t' (by lemma 7). Thus $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma = [\![\Gamma \ \vdash_{\mathsf{s}} \ t']\!]_\gamma = \bullet$.

– If $t = (\lambda x_* : A.u\ p)$ with p a proof-term, then $[\![\Gamma \ \vdash_{\mathsf{s}} \ p]\!]_\gamma = \bullet$ and the result is an easy instance of the substitutivity lemma.
– If $t = (\lambda x_\square : A.u\ v)$, and u and v are no proof-terms, then $[\![\Gamma \ \vdash_{\mathsf{s}} \ \lambda x_\square : A.u]\!]_\gamma$ is a function of domain $[\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma$. Since $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma$ is defined, we know that $[\![\Gamma \ \vdash_{\mathsf{s}} \ v]\!]_\gamma \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma$. The result follows easily by the substitutivity lemma.

Of course, as already pointed out, the previous lemma is *not true* for non-well-sorted reductions.

Theorem 3 (Soundness). *If $\Gamma \vdash_{\mathsf{s}} t : T$ is derivable, then:*

– *$[\![\Gamma]\!]$ is defined,*
– *for any $\gamma \in [\![\Gamma]\!]$, $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma$ and $[\![\Gamma \ \vdash_{\mathsf{s}} \ T]\!]_\gamma$ are defined and $[\![\Gamma \ \vdash_{\mathsf{s}} \ t]\!]_\gamma \in [\![\Gamma \ \vdash_{\mathsf{s}} \ T]\!]_\gamma$.*

Proof. We proceed by induction over the typing derivation. We detail some cases.

Axiom This is trivial since $\{\varnothing; \{\bullet\}\} \in \mathcal{U}$.
Var Trivial.
Pi Let us distinguish two cases: the first where T_2 is a predicate and the second where it is not. If T_2 is a predicate, then $s_2 = *$ and thus for all $\gamma \in [\![\Gamma]\!]$ and $\alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ T_1]\!]_\gamma$ we have $[\![\Gamma; [x : T_1] \ \vdash_{\mathsf{s}} \ t_2]\!]_(\gamma, \alpha)$ and thus $\bigcap_{\alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ T_1]\!]_\gamma} [\![\Gamma \ \vdash_{\mathsf{s}} \ T_2]\!]_{(\gamma,\alpha)}$ is either $\varnothing$ or $\{\bullet\}$.
If T_2 is not a predicate, then the result follows from the closure of $\mathcal{U}$ with respect to dependent product.
Lambda Again, the first case is when B is a predicate and thus $\lambda x_s : A.t$ is a proof. Then $[\![\Gamma \ \vdash_{\mathsf{s}} \ \lambda x_s : A.t]\!]_\gamma$ is defined and equal to $\bullet$. We are left with checking that $[\![\Gamma \ \vdash_{\mathsf{s}} \ \Pi x_s : A.B]\!]_\gamma$ is indeed not empty and equal to $\bullet$. This is the case since for any $\alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma$ the induction hypothesis ensures us that $\bullet = [\![\Gamma; [x_s : A] \ \vdash_{\mathsf{s}} \ t]\!]_{(\gamma,\alpha)} \in [\![\Gamma; [x : A] \ \vdash_{\mathsf{s}} \ B]\!]_{(\gamma,\alpha)}$.
If B is not a predicate, then t is not a proof-term and $[\![\Gamma \ \vdash_{\mathsf{s}} \ \lambda x_s : A.t]\!]_\gamma$ is interpreted by the set-theoretical function. Since $[\![\Gamma \ \vdash_{\mathsf{s}} \ \Pi x_s : A.B]\!]_\gamma$ is defined as the dependent product, the result follows from the induction hypothesis.
App If B is a predicate and t a proof-term, then

$$[\![\Gamma \ \vdash_{\mathsf{s}} \ \Pi x_s : A.B]\!]_\gamma = \bigcap_{\alpha \in [\![\Gamma \ \vdash_{\mathsf{s}} \ A]\!]_\gamma} [\![\Gamma; [x_s : A] \ \vdash_{\mathsf{s}} \ B]\!]_{(\gamma,\alpha)}$$

and it contains $\bullet$ by induction hypothesis. Thus

$$\bullet = [\![\Gamma \ \vdash_{\mathsf{s}} \ (t\ u)]\!]_\gamma \in [\![\Gamma; [x_s : A] \ \vdash_{\mathsf{s}} \ B]\!]_{([\![\Gamma \ \vdash_{\mathsf{s}} \ u]\!]_\gamma, \gamma)} = [\![\Gamma \ \vdash_{\mathsf{s}} \ B\{x_s := u\}]\!]_\gamma.$$

Conv This is a consequence of the previous lemma.

Corollary 3. *There is no term t such that $[] \vdash_{\mathsf{s}} t : \Pi x_\square : *x_\square$ or $[] \vdash t : \Pi x : *x$.*

5 Discussion

We have been able to build a proof-irrelevant model by making the stratification hidden in the usual PTS presentation explicit. We have tried to do this in an as light way as possible, avoiding a tedious stratification of the syntax where proofs, predicates and kinds are defined by mutual induction. Let us mention some applications and possible extensions to this work, as well as some alternative ways to proceed.

5.1 Extensions and Axioms

A particular useful application of set-theoretical models is the validation of axioms. In particular, Barbanera and Berardi's work [4] states that any model validating the excluded middle is necessarily proof-irrelevant. In addition to being proof-irrelevant, set-theoretical models are simple enough to easily validate a large class of axioms.

It therefore seems reasonable to have such a model at hand for theories actually used in proof-checkers. The recent work of Chicli, Pottier and Simpson [6] showed that unpleasant surprises are always possible when this is not the case. More generally, we believe that theories forbidding the excluded middle (i.e. which are not *sub-classical*) are indeed very delicate to handle.

Another advantage of interpretations like the one studied in this paper are that they can often deal with extensions carried out on the predicative level, or levels. Inductive types in $\Box$ can be interpreted as inductive sets, PTS style universes by large enough sets provided one assumes the existence of enough inaccessible cardinals. In that respect, this work can be seen as a cleaning-up of the encoding of Coq's type theory (without the impredicative sort `Set`) which is sketched in [19].

On the other hand, one should mention that adding the subsumption rule

$$\frac{\Gamma \vdash T : *}{\Gamma \vdash T : \Box}$$

to the system cannot be handled easily in our setting and could indeed become problematic when combined with other extensions.

5.2 Judgemental Equality

It is quite easy to see that, even if the presentations of section 2 and 4 differ, the defined denotations are actually equal for well-typed terms and types. That is, the problem presented in section 2 is mainly a circularity in the proof between the general soundness of the model and the soundness of β-reduction.

Another way to break this circularity is to consider judgemental equality, following Martin-Löf [14]. In this case, the β-conversion side condition in the Conv rule is replaced by an explicit judgement of the form $\Gamma \vdash A = B : s$. This corresponds to another style of presentation of type theories, where well-typedness is re-checked after every single step of β-reduction or expansion. When

modeling such theories, it is possible to prove soundness for the model *and* with respect to judgemental equality in a single mutual induction. This is done by Streicher [18] using a presentation involving a form of judgemental equality. As pointed out to us, one may notice that he also uses two different interpretations of the universal quantification depending of the predicative/impredicative level.

There is however a price to pay. Without going into details, it appears that the difficulty is then shifted to the (syntactic) subject-reduction property. This is not too surprising, considering that the derivations with judgemental equality can be almost arbitrarly larger than their counter-parts in the PTS style (this last point is of course crucial in implemented proof-checkers). Actually, establishing the equivalence of impredicative type theories with judgemental equality with their PTS style counter-parts seems related to the strong normalization property, which is notoriously more complex.

5.3 Typed reduction

This last remark is also illustrated by the interpretation carried out by Melliès and Werner in [16]. In this work, the semantic interpretation is carried out for a very restricted version of β-reduction, for which soundness is not problematic. The equivalence with conventional PTSs however is proved *after* strong normalization and heavily relies on the latter property.

5.4 Proof-Irrelevance *per se*

Let us finally mention another, non-axiomatic, extension of CC which is validated by the simple model. It is possible to relax the rule Conv by adding the following coercion : $t =_\beta u$ for any proof-terms t and u. That β-conversion is not to be checked inside proof-terms. The afferent type system is obviously validated by the constructed model. This relaxed conversion rule is particularly interesting when objects live in the predicative level. In [2], Altenkirch presents such a system together with a categorical model; he shows that proof-irrelevance combined with η-conversion in the conversion rule entails the extentionality principle for functions.

There seem to be several practical uses for such a feature. Suppose one defines some type $\mathsf{nat} : \Box$ describing natural numbers together with the usual $\leq$ relation, such that $i \leq j : *$ if $i, j : \mathsf{nat}$. An array of size $n + 1$ can be modelized as a function

$$\mathsf{ar} : \Pi i : \mathsf{nat}.0 \leq i \to i \leq n \to \mathsf{nat}.$$

Doing so in the usual setting, one has to add some conditions stating that $(\mathrm{ar}\ i\ p_1\ p_2)$ does not depend upon the form of p_1 and p_2. Furthermore, these conditions have to be invoqued every time the array is accessed, causing a dramatic increase of the proof-terms. This gets drasticly simplified in the proof-irrelevant version of the type theory. Notice that practical type-checking for such theories is indeed much simpler if proof-terms are tagged in the style of the sorted system.

We believe the study and use of such explicitly proof-irrelevant type theories can be a very fruitful topic.

Acknowledgement

The idea for the counter-example of soundness came up in a discussion of the first author with Thierry Coquand, who expressed doubts about the usual proof-irrelevant models. Anonymous referees expressed several useful remarks and comments.

References

1. P. Aczel. "On relating type theories and set theories", in *Types for Proofs and Programs*, edited by Altenkirch, Naraschewski and Reus, Proceedings of Types '98, LNCS 1657 (1999).
2. T. Altenkirch. "Extensional Equality in Intensional Type Theory". In *Proceedings of the fourteenth Annual IEEE Symposium on Logic in Computer Science*, IEEE, 1999.
3. H. Barendregt. Lambda Calculi with Types. Technical Report 91-19, Catholic University Nijmegen, 1991. In Handbook of Logic in Computer Science, Vol II, Elsevier, 1992.
4. F. Barbanera, S. Berardi. "Proof-irrelevance out of Excluded Middle and Choice in the Calculus of Constructions." *Journal of Functional Programming*, vol.6(3), 519-525, 1996.
5. B. Barras and B. Werner. Coq in Coq. Manuscript.
6. L. Chicli, L. Pottier and C. Simpson. "Quotient Types in Coq". Presentation at the TYPES'01 Workshop, Nijmegen, 2001.
7. Th. Coquand. *Une Théorie des Constructions*. Thèse de Doctorat, Université Paris 7, janvier 1985.
8. Th. Coquand. Metamathematical Investigations of a Calculus of Constructions. In P. Oddifredi (editor), *Logic and Computer Science*. Academic Press, 1990. Rapport de recherche INRIA 1088.
9. Th. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2/3), 1988.
10. H. Geuvers. *Logics and Type Systems*. PhD Thesis, Katholieke Universiteit Nijmegen, 1993.
11. J.-Y. Girard. Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, Thèse d'État, Université Paris 7, 1972.
12. J.-Y. Girard. Translation and appendices Y. Lafont and P. Taylor. *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1989.
13. J.-L. Krivine. *Théorie des ensembles*. Cassini, Paris, 1998.
14. P. Martin-Löf. *Intuitionistic Type Theory*. Studies in Proof Theory, Bibliopolis, 1984.
15. J. McKinna and R. Pollack. "Pure Type Systems formalized", in TLCA'93, M. Bezem and J. F. Groote Eds, LNCS 664, Springer-Verlag, Berlin, 1993.
16. P.-A. Melliès and B. Werner. "A Generic Normalization Proof for Pure Type System" in TYPES'96, E. Gimenez and C. Paulin-Mohring Eds, LNCS 1512, Springer-Verlag, Berlin, 1998.
17. J. Reynolds. "Polymorphism is not Set-Theoretic", *Semantics of Data Types*. G. Kahn, D. B. MacQueen and G. Plotkin Eds. LNCS 173, pp. 145-156, Springer-Verlag, Berlin, 1984.

18. T. Streicher. *Semantics of Type Theory*. Progress in Theoretical Computer Science. Birkhaeuser Verlag, Basel 1991.
19. B. Werner. "Sets in Types, Types in Sets". In, M. Abadi and T. Itoh (Eds), Theoretical Aspects of Computer Science, TACS'97, LNCS 1281, Springer-Verlag, 1997.
20. B. Werner. *Une Théorie des Constructions Inductives*. Thèse de Doctorat, Université Paris 7, 1994.