A Model for Impredicative Type Systems, Universes, Intersection Types and Subtyping

Alexandre Miquel
Projet Coq,
INRIA Rocquencourt BP 105
78 153 Le Chesnay cedex, France
Alexandre.Miquel@inria.fr

Abstract

We introduce a new model based on coherence spaces for interpreting large impredicative type systems such as the Extended Calculus of Constructions (ECC). Moreover, we show that this model is well-suited for interpreting intersection types and subtyping too, and we illustrate this by interpreting a variant of ECC with an additional intersection type binder. Furthermore, we propose a general method for interpreting the impredicative level in a non-syntactical way, by allowing the model to be parametrized by an arbitrarily large coherence space in order to interpret inhabitants of impredicative types. As an application, we show that uncountable types such as the type of real numbers or Zermelo-Fränkel sets can safely be axiomatized on the impredicative level of, say, ECC, without harm for consistency.

1 Introduction

In this paper, we set out a new model based on coherence spaces for interpreting and proving the consistency of large impredicative type theories. Originally, this model was designed for proving the consistency of the Implicit Calculus of Constructions [10], a Curry-style variant of ECC with an additional intersection type binder. Nevertheless, it is powerful enough for proving the consistency of a large class of impredicative type systems, since it makes it possible to interpret universes, intersection types and subtyping in the same framework.

But the most interesting feature of the model is that the inhabitants of impredicative types are not interpreted in a syntactical way, but by the points of an arbitrary coherence space, provided it is a λ-model. Since the cardinality of such a λ-model may be arbitrarily large, we can show that in type theories like ECC, uncountable types can be axiomatized in a consistent way on the impredicative level.

Outline

In section 2, we start by introducing the Implicit Calculus of Constructions, since its interpretation will illustrate most of the capacities of the model.

Section 3 is devoted to the interpretation of types in coherence spaces. We also show how dependent products, intersection types, subtyping and universes can be interpreted within that framework.

After explaining a general method for interpreting impredicativity in coherence spaces (section 4), we finally build the model in section 5, and we use it for proving the consistency of a restriction of the Implicit Calculus of Constructions, which contains ECC.

2 The implicit calculus

2.1 Presentation

Basically, the Implicit Calculus of Constructions (ICC) or, shortly, the implicit calculus, is a variant of ECC [9] in which we distinguish two kinds of products: the *explicit* product and the *implicit* product, denoted by $\Pi x:T.U$ and $\forall x:T.U$ respectively.

The explicit product is the usual dependent product of Pure Type Systems (PTS), whereas the implicit product is much more an intersection type binder, like the impredicative product of the Curry-style system F and its extensions, such as the Type Assignment Systems [3].

But in contrast to these calculi, the implicit product can be used at any level—which may be impredicative or predicative. Since the two products are distinguished by the syntax and not only by the typing rules, they can freely be mixed in writing terms.

We also use a Curry-style λ-abstraction, since the PTS-style λ-abstraction *à la* Church would prevent us from writing implicitly polymorphic terms such as the identity $\lambda x\,.\,x$.

Initially, the Implicit Calculus of Constructions was introduced to give an answer to the problem of 'implicit arguments'[1] in the framework of proof assistants based on variants of the Calculus of Constructions such as the Coq system [2]. However, we strongly conjecture that the typing relation of ICC (which contains the Curry-style system F) is undecidable, so this calculus seems to be a bad candidate for solving our initial problem, at least from a practical point of view.

Nevertheless, a study of the syntactical and semantical properties of the implicit calculus is still interesting, since this calculus enjoys a rich subtyping relation which might be useful to understand subtyping in a framework that extends the scope of usual functional programming languages such as ML.

In the following, we will only recall some basic definitions and results whose proofs are detailed in [10].

2.2 Basic notations

In this section, we shall assume that the reader is familiar with PTS [5]. The set of sorts used in the implicit calculus is defined by

$$\mathcal{S} = \{\mathsf{Prop};\ \mathsf{Set}\} \cup \{\mathsf{Type}_i;\ i > 0\},$$

where Prop and Set denote the impredicative sorts, and $(\mathsf{Type}_i)_{i>0}$ the usual universe hierarchy of ECC.

Notice that here, we have two impredicative sorts, since it is convenient to distinguish a sort for propositional types (Prop) from a sort for impredicative data types (Set). Although they are completely isomorphic for the typing rules, it will be interesting to interpret them differently when building the corresponding model.

The terms of the implicit calculus are given by the following syntax:

$$\begin{array}{rcl} M,N,T,U & ::= & x \\ & \mid & \mathsf{Prop} \mid \mathsf{Set} \mid \mathsf{Type}_i \\ & \mid & \Pi x{:}T\,.\,U \mid \forall x{:}T\,.\,U \\ & \mid & \lambda x\,.\,M \mid M\,N \end{array}$$

As usual, we will consider terms up to α-conversion. In the following, we will denote by $FV(M)$ the set of free variables of a term M, and by $M[x{:=}N]$ the term built by substituting the term N to each free occurrence of x in the term M.

[1] That is arguments of functions that can be automatically infered by the system, and that the user does not wish to write, especially when developing large proofs.

Reduction rules of the implicit calculus are the usual β and η-reduction rules of the λ-calculus. For each rule $R \in \{\beta;\ \eta;\ \beta\eta\}$, $\rightarrow_R$ denotes the one-step R-reduction relation, while $\twoheadrightarrow_R$ denotes the reflexive and transitive closure of $\rightarrow_R$ and $\cong_R$ denotes the R-convertibility equivalence relation.

Proposition 2.1 (Church-Rosser) — *The β, η and $\beta\eta$-reduction rules are Church-Rosser.*

Notice that the $\beta\eta$-reduction is confluent since we have no type annotation on the λ.[2]

2.3 Typing rules

Before introducing typing rules, we have to define two sets $\mathbf{Axiom} \subset \mathcal{S}^2$ and $\mathbf{Rule} \subset \mathcal{S}^3$. The set $\mathbf{Axiom}$, defined by

$$\begin{array}{rcl} \mathbf{Axiom} & = & \{(\mathsf{Prop}, \mathsf{Type}_1);\ (\mathsf{Set}, \mathsf{Type}_1)\} \quad \cup \\ & & \{(\mathsf{Type}_i, \mathsf{Type}_{i+1});\quad i > 0\}, \end{array}$$

is used for typing sorts, whereas the set $\mathbf{Rule}$, defined by

$$\begin{array}{rcl} \mathbf{Rule} & = & \{(s, \mathsf{Prop}, \mathsf{Prop});\quad s \in \mathcal{S}\} \quad \cup \\ & & \{(s, \mathsf{Set}, \mathsf{Set});\quad s \in \mathcal{S}\} \quad \cup \\ & & \{(\mathsf{Type}_i, \mathsf{Type}_i, \mathsf{Type}_i);\quad i > 0\}, \end{array}$$

is used for typing products. Note that the same set is used for typing both explicit products and implicit products.

We also need to define an ordering relation between sorts, which is called the *cumulative order*. This ordering relation, denoted by $s_1 \leqslant s_2$, is defined by

$$\begin{array}{rclcrcl} \mathsf{Prop} & \leqslant & \mathsf{Prop}, & & \mathsf{Set} & \leqslant & \mathsf{Set}, \\ \mathsf{Prop} & \leqslant & \mathsf{Type}_i, & & \mathsf{Set} & \leqslant & \mathsf{Type}_i, \\ \mathsf{Type}_i & \leqslant & \mathsf{Type}_j & \text{if } i \leqslant j & & & \end{array}$$

The typing rules of the implicit calculus involve two judgments:

- a judgment denoted by $\Gamma \vdash$, which says: "the context Γ is well-formed";
- a judgment denoted by $\Gamma \vdash M : T$, which says: "under the context Γ, the term M has type T".

Both judgments are defined by mutual induction with the rules written in figure 1. These rules include:

- Rules for well-formed contexts (WF-E) and (WF-S).
- Logical rules (VAR), (SORT), (EXPPROD), (IMPPROD), (LAM) and (APP).

[2] It is well-known[4] that the $\beta\eta$-reduction is not confluent on raw terms in a Church-style PTS.

Rules for well-formed contexts

$$\frac{}{\varnothing \vdash} \text{ (WF-E)} \qquad \frac{\Gamma \vdash T : s \quad x \notin \mathrm{Dom}(\Gamma)}{\Gamma; x : T \vdash} \text{ (WF-S)}$$

Rules for well-typed terms

$$\frac{\Gamma \vdash \quad (x : T) \in \Gamma}{\Gamma \vdash x : T} \text{ (VAR)} \qquad \frac{\Gamma \vdash \quad (s_1, s_1) \in \mathbf{Axiom}}{\Gamma \vdash s_1 : s_2} \text{ (SORT)}$$

$$\frac{\Gamma \vdash T : s_1 \quad \Gamma; x : T \vdash U : s_2 \quad (s_1, s_2, s_3) \in \mathbf{Rule}}{\Gamma \vdash \Pi x : T . U : s_3} \text{ (EXPPROD)}$$

$$\frac{\Gamma \vdash T : s_1 \quad \Gamma; x : T \vdash U : s_2 \quad (s_1, s_2, s_3) \in \mathbf{Rule}}{\Gamma \vdash \forall x : T . U : s_3} \text{ (IMPPROD)}$$

$$\frac{\Gamma; x : T \vdash M : U \quad \Gamma \vdash \Pi x : T . U : s}{\Gamma \vdash \lambda x . M : \Pi x : T . U} \text{ (LAM)} \qquad \frac{\Gamma \vdash M : \Pi x : T . U \quad \Gamma \vdash N : T}{\Gamma \vdash M\ N : U[x := N]} \text{ (APP)}$$

$$\frac{\Gamma; x : T \vdash M : U \quad \Gamma \vdash \forall x : T . U : s \quad x \notin FV(M)}{\Gamma \vdash M : \forall x : T . U} \text{ (GEN)}$$

$$\frac{\Gamma \vdash M : \forall x : T . U \quad \Gamma \vdash N : T}{\Gamma \vdash M : U[x := N]} \text{ (INST)}$$

$$\frac{\Gamma \vdash M : T \quad \Gamma \vdash T' : s \quad T \cong_{\beta\eta} T'}{\Gamma \vdash M : T'} \text{ (CONV)} \qquad \frac{\Gamma \vdash T : s_1 \quad s_1 \leqslant s_2}{\Gamma \vdash T : s_2} \text{ (CUM)}$$

$$\frac{\Gamma \vdash \lambda x . (M\ x) : T \quad x \notin FV(M)}{\Gamma \vdash M : T} \text{ (EXT)} \qquad \frac{\Gamma \vdash M : \forall x : T . U \quad x \notin FV(U)}{\Gamma \vdash M : U} \text{ (STR)}$$

Figure 1. Typing rules of the implicit calculus

- A cumulativity rule (CUM) and a convertibility rule (CONV) for $\beta\eta$.
- Four specific rules (GEN), (INST), (EXT) and (STR) which are explained with more details below.

The logical rules are the usual logical rules of ECC, except that we have an extra rule for the implicit product—which shares the same premises and side-condition as the rule for the explicit product—and that the rule (LAM) now involves a Curry-style λ-abstraction, instead of the Church-style λ-abstraction of ECC. Another difference between ECC and ICC is that in the latter, the convertibility rule (CONV) identifies types up to $\beta\eta$-convertibility.

The rules (GEN) and (INST) are respectively the introduction and elimination rules for implicit product types. In contrast to the rules (LAM) and (APP), the rules (GEN) and (INST) have no associated constructors. Remark that the rule (GEN) involves a side-condition ensuring that the variable x whose type has to be generalized does not appear free in the term M.

The purpose of the next rule, called (EXT) for 'extensionality', is to enforce the η-subject reduction property in the implicit calculus.[3] This rule is desirable here, since it gives smoother properties for the subtyping relation, such as the contravariant/covariant subtyping rules in products.

The last rule, called (STR) for 'strengthening', gives the semantic of the non-dependent implicit product, by saying that a term of type $\forall x{:}T\,.\,U$ also has type U when x does not occur free in U. The main reason for adding this rule is that it is necessary for deriving the usual strengthening rule (hence its name), which would not hold without it.

2.4 Typing properties

The implicit calculus enjoys the $\beta\eta$-subject reduction property:

Proposition 2.2 (Subject reduction) — *If* $M \twoheadrightarrow_{\beta\eta} M'$, *then*

$$\Gamma \vdash M : T \quad \Rightarrow \quad \Gamma \vdash M' : T.$$

Such a result is not so easy to prove [10], because of all the rules which are not syntax-directed in the implicit calculus.

Besides the subject reduction property, one of the main properties of the implicit calculus is that typing rules induce a rich subtyping relation. This subtyping relation, which is denoted by $\Gamma \vdash T \leqslant T'$, can be defined directly from the typing judgment as follows:

$$\Gamma \vdash T \leqslant T' \quad \equiv \quad \Gamma; x : T \vdash x : T',$$

(where x is a fresh variable).

For each context Γ, the relation $\Gamma \vdash T \leqslant T'$ is a preordering over well-formed types. Its main properties, presented here as admissible rules, are the following:

- The rule of subtyping:

$$\frac{\Gamma \vdash M : T \quad \Gamma \vdash T \leqslant T'}{\Gamma \vdash M : T'} \text{ (SUB)}$$

- Contravariance/covariance in products:

$$\frac{\Gamma \vdash T' \leqslant T \quad \Gamma; x : T' \vdash U \leqslant U'}{\Gamma \vdash \Pi x{:}T\,.\,U \leqslant \Pi x{:}T'\,.\,U'}$$

$$\frac{\Gamma \vdash T' \leqslant T \quad \Gamma; x : T' \vdash U \leqslant U'}{\Gamma \vdash \forall x{:}T\,.\,U \leqslant \forall x{:}T'\,.\,U'}$$

The first of these rules is an immediate consequence of the (EXT) rule, and could not be derived without it.[4] Other interesting admissible rules concerning subtyping are given in [10].

In the implicit calculus, the 'false' proposition can be represented either by $\Pi A{:}\mathsf{Prop}\,.\,A$ or by $\forall A{:}\mathsf{Prop}\,.\,A$. However, both propositions are provably equivalent in the calculus. We have the following result:

Proposition 2.3 — *If the implicit calculus is strongly normalizing, then it is consistent.*

The proof is done as usual by considering a proof of 'false' reduced in head normal form, by discriminating cases and reasoning using inversion lemmas.

2.5 Restricting the calculus

The main limitation of the model that we will build in section 5 is that it cannot interpret the (STR) rule. For this reason, we will only prove the consistency of the *restricted implicit calculus* (ICC^-), which is the implicit calculus without the (STR) rule.

Indeed, the reader may have noticed that the (STR) rule is really needed when T is empty, since it can be derived by using the (INST) rule when T is not empty. One may fear that such a rule could jeopardize the consistency of the whole theory. Although we do not have actually any proof of the consistency of the whole system, we have several reasons to hope that this rule does not endanger the consistency of ICC. The first reason is that the strong normalization of ICC—still conjectured—implies the consistency of the whole system. Another reason is that we can prove in a

[3] Such a rule cannot be derived from the other rules, for the same reasons that it cannot be derived in the Curry-style system F which is included in ICC.

[4] In fact, the rule for contravariance/covariance in explicit products is equivalent to (EXT).

pure syntactical way that ICC is strongly normalizing iff ICC^- is strongly normalizing—in other words, the strong normalization of ICC does not depend on the presence of the (STR) rule (see the discussion about strong normalization in section 6).

3 Types in coherence spaces

In this section, we shall assume that the reader is familiar with coherence spaces [6], and more generally with domain theory. So we will only recall some basic definitions and notations.

3.1 Coherence spaces

A coherence space is a set of sets $\mathcal{A}$ satisfying the following criterions:

1. if a is an element of $\mathcal{A}$ then every subset of a is also an element of $\mathcal{A}$;
2. if M is a subset of $\mathcal{A}$ such that $a_1 \cup a_2 \in \mathcal{A}$ for every $a_1, a_2 \in M$, then $\bigcup M \in \mathcal{A}$.

Elements of a coherence space $\mathcal{A}$ are called *points*. Points of $\mathcal{A}$ also are sets, whose elements are called *atoms*, or *tokens*. The set of atoms of $\mathcal{A}$ is called the *web* of $\mathcal{A}$ and denoted by $|\mathcal{A}|$.

In the following, points will be denoted by lowercase latin letters a, b, c, and atoms by lowercase greek letters α, β, γ.

The web of $\mathcal{A}$ has a graph structure given by the reflexive and symmetric relation denoted by $\alpha_1 \frown \alpha_2 \pmod{\mathcal{A}}$ and defined by

$$\alpha_1 \frown \alpha_2 \pmod{\mathcal{A}} \quad \Leftrightarrow \quad \{\alpha_1; \alpha_2\} \in \mathcal{A}.$$

This graph structure completely determines the coherence space $\mathcal{A}$, whose points are exactly the *cliques* of $\mathcal{A}$, that is full subgraphs of $|\mathcal{A}|$.

Coherence spaces can be considered as a special case of Scott domains whose elements are ordered by inclusion. In particular, finite elements (at the sense of domain theory) are precisely the finite points. But in contrast to Scott domains, coherence spaces are not equipped with continuous functions, but with *stable functions*.

Let $\mathcal{A}$ and $\mathcal{B}$ be coherence spaces. A function $F : \mathcal{A} \to \mathcal{B}$ is said to be *stable* if it is Scott-continuous, and if

$$a_1 \cup a_2 \in \mathcal{A} \quad \Rightarrow \quad F(a_1 \cap a_2) = F(a_1) \cap F(a_2)$$

for all $a_1, a_2 \in \mathcal{A}$. If $F : \mathcal{A} \to \mathcal{B}$ is a stable function, we denote by $\mathsf{Tr}(F)$ its *trace*, which is formed by the set of pairs $(a_0, \beta) \in \mathcal{A} \times |\mathcal{B}|$ such that

$$\beta \in F(a_0) \quad \wedge \quad \forall a_1 \subsetneq a_0 \quad \beta \notin F(a_1).$$

The trace of F uniquely determines F, so it is common to identify F with $\mathsf{Tr}(F)$. The set of stable functions $F : \mathcal{A} \to \mathcal{B}$ is also identified with the set of their traces, which is denoted by $\mathcal{A} \to \mathcal{B}$. With such a representation, the set of (traces of) stable functions form a coherence space, whose web is given by

$$|\mathcal{A} \to \mathcal{B}| = \mathcal{A}_{\mathbf{fin}} \times |\mathcal{B}|,$$

where $\mathcal{A}_{\mathbf{fin}}$ denotes the set of finite points of $\mathcal{A}$, and whose coherence relation is defined by

$$\begin{aligned} & (a_1, \beta_1) \frown (a_2, \beta_2) \pmod{\mathcal{A} \to \mathcal{B}} \\ \Leftrightarrow \quad & a_1 \cup a_2 \in \mathcal{A} \Rightarrow \beta_1 \frown \beta_2 \pmod{\mathcal{B}} \quad \wedge \\ & (a_1 \cup a_2 \in \mathcal{A} \wedge a_1 \neq a_2) \Rightarrow \beta_1 \neq \beta_2 \end{aligned}$$

Notice that inclusion ordering over traces does not correspond to the usual point-wise ordering over functions, but to Berry's order, which is stronger.

Let $\mathcal{A}$ and $\mathcal{A}'$ be coherence spaces. A stable function $F : \mathcal{A} \to \mathcal{A}'$ is called a *rigid embedding* if there is a graph embedding $i : |\mathcal{A}| \to |\mathcal{A}'|$ such that

$$F(a) = i(a) = \{i(\alpha);\ \alpha \in a\}$$

for all $a \in \mathcal{A}$. By identifying each atom $\alpha \in |\mathcal{A}'|$ to its image $i(\alpha) \in |\mathcal{A}'|$, we can see any rigid embedding as a rigid inclusion between coherence spaces, that is an inclusion $|\mathcal{A}| \subset |\mathcal{A}'|$ such that

$$\alpha_1 \frown \alpha_2 \pmod{\mathcal{A}} \quad \Leftrightarrow \quad \alpha_1 \frown \alpha_2 \pmod{\mathcal{A}'}$$

for all $\alpha_1, \alpha_2 \in |\mathcal{A}|$. In the following, we will denote by $\mathcal{A} \Subset \mathcal{A}'$ the fact that $\mathcal{A}$ is rigidly embedded into $\mathcal{A}'$, and we will say that $\mathcal{A}'$ is an *extension* of $\mathcal{A}$.

3.2 Stability revisited

Since the presence of a universe hierarchy in our calculus, the usual continuity requirement appears to be too strong, especially for interpreting the two products. So we need to weaken this condition the following way:

Let $\mathfrak{a}$ be an infinite cardinal. We denote by $\mathcal{A}_{\mathfrak{a}}$ the set of all points $\mathfrak{a} \in \mathcal{A}$ such that $\mathrm{Card}(a) < \mathfrak{a}$. A map $F : \mathcal{A} \to \mathcal{B}$ is said to be

- $\mathfrak{a}$-*continuous* if

$$\forall\, a \in \mathcal{A} \qquad F(a) = \bigcup_{\substack{a_0 \subset a \\ \mathrm{Card}(a_0) < \mathfrak{a}}} F(a_0);$$

- *quasi-stable* if

$$F\Big(\bigcap_{i \in I} a_i\Big) = \bigcap_{i \in I} F(a_i)$$

for every family $(a_i)_{i \in I}$ such that $\bigcup a_i \in \mathcal{A}$;

- $\mathfrak{a}$-*stable* if it is $\mathfrak{a}$-continuous and quasi-stable.

Notice that both conditions of $\mathfrak{a}$-continuity and quasi-stability separately imply monotonicity. It is also straightforward to check that the usual notions of continuity and stability are special cases of $\mathfrak{a}$-continuity and $\mathfrak{a}$-stability corresponding to the case where $\mathfrak{a} = \aleph_0$.

As for stable functions, quasi-stable and $\mathfrak{a}$-stable functions are determined by their traces, so we will identify such functions with their traces. In the following, we denote by $\mathcal{A} \xrightarrow{q} \mathcal{B}$ the set of (traces of) quasi-stable functions $F : \mathcal{A} \to \mathcal{B}$, and by $\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}$ the set of (traces of) $\mathfrak{a}$-stable functions $F : \mathcal{A} \to \mathcal{B}$. We have the following proposition:

Proposition 3.1 — *The sets $\mathcal{A} \xrightarrow{q} \mathcal{B}$ and $\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}$ are coherence spaces whose webs are given by*

$$|\mathcal{A} \xrightarrow{q} \mathcal{B}| = \mathcal{A} \times |\mathcal{B}| \quad \textit{and} \quad |\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}| = \mathcal{A}_{\mathfrak{a}} \times |\mathcal{B}|,$$

and whose coherence relation is given by the same formula used for stable functions (see 3.1).

In fact, we have the following infinite sequence of rigid embeddings:

$$\mathcal{A} \xrightarrow{\aleph_0} \mathcal{B} \Subset \mathcal{A} \xrightarrow{\aleph_1} \mathcal{B} \Subset \mathcal{A} \xrightarrow{\aleph_2} \mathcal{B} \Subset \ldots \Subset \mathcal{A} \xrightarrow{q} \mathcal{B}.$$

In other words, the greater the cardinal $\mathfrak{a}$, the weaker the corresponding notion of $\mathfrak{a}$-stability, the weakest case corresponding to the coherence space of quasi-stable functions, when all continuity requirement has disappeared. In the following, we will call the *category of $\mathfrak{a}$-coherence spaces* the category whose objects are coherence spaces and whose morphisms are $\mathfrak{a}$-stable functions.

Proposition 3.2 — *For a given infinite cardinal $\mathfrak{a}$, the category of $\mathfrak{a}$-coherence spaces is a Cartesian closed category.*

3.3 Types and type bases

Let $\mathcal{A}$ be a coherence space, and $X \subset \mathcal{A}$ a set of points of $\mathcal{A}$. We say that

- X is a *type basis of $\mathcal{A}$* if $a_1 \cup a_2 \in \mathcal{A}$ implies $a_1 = a_2$ for all $a_1, a_2 \in X$;
- X is a *semantical type of $\mathcal{A}$* if X is upwards closed in $\mathcal{A}$, and if for each subset $M \subset X$, $\bigcup M \in \mathcal{A}$ implies $\bigcap M \in X$.

If $X \subset \mathcal{A}$ is a type basis, then its upwards closure is a semantical type of $\mathcal{A}$, denoted by $\overline{X}$ and called the *type generated by X in $\mathcal{A}$*. Conversely, if X is a semantical type of $\mathcal{A}$, then the set of its minimal elements form a type basis of $\mathcal{A}$, denoted by $\underline{X}$ and called the *type basis of X*. Moreover, these two maps are reciprocal, and they induce a one-to-one correspondence between type bases and semantical types of the coherence space $\mathcal{A}$.

Intuitively, a type basis X_0 can be thought as a specification, whereas the semantical type $\overline{X}$ generated by X in $\mathcal{A}$ *extensionally* represents all the points of $\mathcal{A}$ that meet this specification.

In the following, we will represent types *intensionally* by using the notion of type basis. The main reason for this choice is that type bases are invariant by rigid embeddings, which is not the case for semantical types.

Indeed, consider two coherence spaces $\mathcal{A}$ and $\mathcal{A}'$ such that $\mathcal{A} \Subset \mathcal{A}'$. A semantical type $X \subset \mathcal{A}$ is not in general a semantical type of $\mathcal{A}'$, since X is not necessary upwards closed in $\mathcal{A}'$. But if we consider the type basis $X_0 = \underline{X}$, the set X_0 is also a type basis in $\mathcal{A}'$. Hence, X_0 generates a semantical type of $\mathcal{A}'$, say X', which is nothing else but the upwards closure of X in $\mathcal{A}'$. In such a situation, we will say that X' is the *natural extension of X in $\mathcal{A}'$*.

In other words, a type basis $X_0 \subset \mathcal{A}$ does not only represent a unique semantical type $X \subset \mathcal{A}$, but it also represents all its natural extensions in coherence spaces extending $\mathcal{A}$.

Finally, another link between a semantical type and its type basis is the existence of a *coercion* from the former to the latter. More formally, consider a type basis $X_0 \subset \mathcal{A}$ and an element a of the type generated by X_0 into $\mathcal{A}$. Since elements of X_0 are mutually incompatible, there exists exactly one element $a_0 \in X_0$ such that $a_0 \subset a$. Such an element, which is denoted by $(a \downarrow X_0)$, is called the *coercion of a into X_0*.

3.4 Types as values

In the implicit calculus, types do not only appear at the right side of judgments, but may also appear in well-typed terms. Hence, we also need a method for encoding a type basis into a single point of a coherence space (*i.e.* a *type as a value*). As we will see later, continuity reasons make that such an encoding has no sense for all type bases, but only for *small* ones.

Let $\mathfrak{a}$ be an infinite cardinal, and $\mathcal{A}$ a $\mathfrak{a}$-coherence space. We say that a type basis $X \subset \mathcal{A}$ is *small* (with respect to $\mathfrak{a}$) if we have

$$\mathrm{Card}(X) < \mathfrak{a} \quad \text{and} \quad \mathrm{Card}(|X|) < \mathfrak{a},$$

where $|X|$ denotes the set of all atoms which appear in points belonging to X. In the same way, a semantical type is said to be small if its type basis is small. Notice that the cardinal of a small type may be arbitrarily big.

For each small type basis $X \subset \mathcal{A}$, we introduce a new atom, denoted by $\tau(X)$, for representing the semantical types that are generated by X in coherence spaces extending $\mathcal{A}$.

By definition, the *space of small types of* $\mathcal{A}$ is the coherence space denoted by $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})$, whose web is the set of atoms $\tau(X)$ where X ranges over the set of small type bases of $\mathcal{A}$, and whose coherence relation is the equality. The coherence space $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})$ is a flat coherence space, whose points are either the empty set $\varnothing$ (the undefined type) or singletons in the form $\{\tau(X)\}$ where X is a small type basis of $\mathcal{A}$. In the following, we will denote $\langle X\rangle = \{\tau(X)\}$ the value associated to the small type basis $X \subset \mathcal{A}$.

Since type bases are invariant by rigid embeddings, $\mathcal{A} \Subset \mathcal{A}'$ implies $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}) \Subset \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}')$. In other words, we have:

Proposition 3.3 — *The correspondence* $\mathcal{A} \mapsto \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})$ *is a covariant endofunctor in the category of coherence spaces equipped with rigid embeddings.*

In the following, we will denote $\mathsf{Ty}^*_{\mathfrak{a}}(\mathcal{A})$ the set defined by

$$\mathsf{Ty}^*_{\mathfrak{a}}(\mathcal{A}) \;=\; \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}) \setminus \{\varnothing\}.$$

Notice that $\mathsf{Ty}^*_{\mathfrak{a}}(\mathcal{A})$ is a semantical type of $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})$, which is equal to its type basis.

3.5 Product and intersection

Let $\mathcal{A}$ and $\mathcal{B}$ be coherence spaces, X a type basis of $\mathcal{A}$, and $(Y_a)_{a\in X}$ a family of type bases of $\mathcal{B}$ indexed over X. The set of all $\mathfrak{a}$-stable functions $F \in \mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}$ such that

$$\forall\, a \in \mathcal{A} \quad a \in X \;\Rightarrow\; F(a) \in \overline{Y_a}$$

is a semantical type of $\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}$, whose type basis is denoted by $\Pi^{\mathcal{A}\xrightarrow{\mathfrak{a}}\mathcal{B}}(a\in X; Y_a)$. In the same way, the intersection of the semantical types $\overline{Y_a}$ for $a \in X$ is a semantical type of $\mathcal{B}$, whose type basis is denoted by $\forall^{\mathcal{B}}(a\in X; Y_a)$.

Notice that the notation $\forall^{\mathcal{B}}(a \in X; Y_a)$ makes sense even if $X = \varnothing$, by assuming that the intersection of an empty family is equal to $\mathcal{B}$.[5] In that particular case, we have $\forall^{\mathcal{B}}(a\in X; Y_a) = \{\varnothing\}$, which is the type basis representing the whole space $\mathcal{B}$.

We have the following independence result:

Lemma 3.4 — *Let* $\mathcal{A}$, $\mathcal{A}'$, $\mathcal{B}$ *and* $\mathcal{B}'$ *be coherence spaces such that* $\mathcal{A} \Subset \mathcal{A}'$ *and* $\mathcal{B} \Subset \mathcal{B}'$. *If* X *is a type basis of* $\mathcal{A}$ *and* $(Y_a)_{a\in X}$ *a family of type bases of* $\mathcal{B}$ *indexed over* X, *then we have*

$$\begin{aligned}\Pi^{\mathcal{A}'\xrightarrow{\mathfrak{a}}\mathcal{B}'}(a\in X; Y_a) &= \Pi^{\mathcal{A}\xrightarrow{\mathfrak{a}}\mathcal{B}}(a\in X; Y_a)\\ \forall^{\mathcal{B}'}(a\in X; Y_a) &= \forall^{\mathcal{B}}(a\in X; Y_a)\end{aligned}$$

This allows us to drop the superscripts in the notations above, by writing shortly $\Pi(a\in X; Y_a)$ and $\forall(a\in X; Y_a)$.

In general, type smallness is not preserved by dependent product and intersection, except if the cardinal $\mathfrak{a}$ is either countable or inaccessible.

The cardinal $\mathfrak{a}$ is said to be *inaccessible* if it satisfies:

1. $\mathfrak{a} > \aleph_0$;
2. for all cardinal $\mathfrak{b}$, $\mathfrak{b} < \mathfrak{a}$ implies $2^{\mathfrak{b}} < \mathfrak{a}$;
3. for each family $(\mathfrak{b}_i)_{i\in\mathfrak{c}}$ indexed over a cardinal $\mathfrak{c} < \mathfrak{a}$ such that $\mathfrak{b}_i < \mathfrak{a}$ for all $i \in \mathfrak{c}$, we have $\sup_{i\in\mathfrak{c}} \mathfrak{b}_i < \mathfrak{a}$.

Although the existence of inaccessible cardinals cannot be proven in the ZF-set theory [8][6], such 'big' cardinals have been used for a while, especially for building set-theoretical models of type theories [9].

Lemma 3.5 — *Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be two coherence spaces equipped with* $\mathfrak{a}$*-stable functions, where* $\mathfrak{a}$ *is either countable or inaccessible. If* X *is a small type basis of* $\mathcal{A}$ *and if* $(Y_a)_{a\in X}$ *is a family of small type bases of* $\mathcal{B}$ *indexed over* X, *then the sets* $\Pi(a\in X; Y_a)$ *and* $\forall(a\in X; Y_a)$ *are small type bases of* $\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}$ *and* $\mathcal{B}$ *respectively.*

In the following, we will assume that $\mathfrak{a}$ is either countable or inaccessible.

Now, we can transpose the dependent product and type intersection operators at the level of types-as-values by introducing the functions

$$\begin{aligned}\pi^0 &: \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}) \times [\mathcal{A} \xrightarrow{\mathfrak{a}} \mathsf{Ty}_{\mathfrak{a}}(\mathcal{B})] \to \mathsf{Ty}_{\mathfrak{a}}(\mathcal{B})\\ \pi^1 &: \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}) \times [\mathcal{A} \xrightarrow{\mathfrak{a}} \mathsf{Ty}_{\mathfrak{a}}(\mathcal{B})] \to \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B})\end{aligned}$$

defined by:

1. if $t = \langle X\rangle$ and if $f(a) = \langle Y_a\rangle$ for all $a \in X$, then

$$\begin{aligned}\pi^0(t,f) &= \langle \forall(a\in X; Y_a)\rangle\\ \pi^1(t,f) &= \langle \Pi(a\in X; Y_a)\rangle\end{aligned}$$

2. in all other cases,

$$\pi^0(t,f) \;=\; \pi^1(t,f) \;=\; \varnothing.$$

Lemma 3.6 — *The functions* π^0 *and* π^1 *are* $\mathfrak{a}$*-stable functions from* $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}) \times [\mathcal{A} \xrightarrow{\mathfrak{a}} \mathsf{Ty}_{\mathfrak{a}}(\mathcal{B})]$ *to* $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{B})$ *and* $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B})$ *respectively.*

Quasi-stability of π^0 and π^1 is straightforward, and the $\mathfrak{a}$-continuity of these functions comes from the fact that X and Y_a are small types for all $a \in X$.

[5] Notice that this case can be inconsistent with the meaning of the (STR) rule, hence the reason for not interpreting it.

[6] Unless it turns out inconsistent.

3.6 Subtyping

Let $\mathcal{A}$ be a coherence space. It is natural to order the set of all type bases of $\mathcal{A}$ by the relation $X \leqslant Y$ defined by:

$$\begin{array}{rcl} X \leqslant Y & \Leftrightarrow & \overline{X} \subset \overline{Y} \\ & \Leftrightarrow & \forall a \in X \quad \exists a' \in Y \quad a' \subset a \end{array}$$

Notice that because of the second equivalence, this ordering does not depend on the coherence space $\mathcal{A}$ in which the semantical types generated by X and Y are considered. Moreover, dependent product and intersection satisfy the expected property with respect to this ordering:

Lemma 3.7 *Let $\mathcal{A}$ and $\mathcal{B}$ be two coherence spaces, X, X' two type bases of $\mathcal{A}$ and $(Y_a)_{a\in X}$, $(Y'_a)_{a\in X'}$ two families of type bases of $\mathcal{B}$ indexed over X and X' respectively. If $X' \leqslant X$ and $Y_a \leqslant Y'_{(a\downarrow X')}$ for all $a \in X$, then*

$$\begin{array}{rcl} \forall(a{\in}X;Y_a) & \leqslant & \forall(a{\in}X';Y'_a) \\ \textit{and} \quad \Pi(a{\in}X;Y_a) & \leqslant & \Pi(a{\in}X';Y'_a). \end{array}$$

4 Impredicativity

4.1 A trivial encoding

In every coherence space $\mathcal{A}$, there exists at least two different types which are:

- The empty type $\varnothing$, whose type basis is equal to $\varnothing$ and denoted by $\mathbf{0}$.
- The type $\mathcal{A}$ of all objects, whose type basis is equal to $\{\varnothing\}$ and denoted by $\mathbf{1}$.

Notice that the denotations $\mathbf{0} = \varnothing$ and $\mathbf{1} = \{\varnothing\}$ do not depend on the coherence space $\mathcal{A}$. Moreover, $\mathbf{0}$ and $\mathbf{1}$ are respectively the smallest and the largest type bases of $\mathcal{A}$ for the subtyping ordering. These types are good candidates for interpreting in a proof-irrelevant way the propositional types:

Lemma 4.1 — *Let $\mathcal{A}$ and $\mathcal{B}$ be two coherence spaces equipped with $\mathfrak{a}$-stable functions, X a type basis of $\mathcal{A}$ and $(Y_a)_{a\in X}$ a family of type bases of $\mathcal{B}$ indexed over X. If $Y_a \in \{\mathbf{0};\mathbf{1}\}$ for all $a \in X$, then*

- $\forall(a{\in}X;Y_a) = \Pi(a{\in}X;Y_a) = \mathbf{1}$ *if $Y_a = \mathbf{1}$ for all $a \in X$;*
- $\forall(a{\in}X;Y_a) = \Pi(a{\in}X;Y_a) = \mathbf{0}$ *if $Y_a = \mathbf{0}$ for some $a \in X$.*

Remark that $\mathbf{0}$ and $\mathbf{1}$ are small type basis for all (infinite) cardinal $\mathfrak{a}$, even if the type basis $\mathbf{1}$ represents all the points in all the coherence spaces.

4.2 The space of computational values

Although the trivial interpretation of impredicativity is sufficient for interpreting ICC^- (and proving its consistency), we need to refine the ideas expressed above if we want to interpret impredicativity in a non trivial way. Remark that such a requirement is necessary if we want to reuse our model for type systems with strong elimination, such as the Calculus of Inductive Constructions [11, 12].

Usually, impredicative types are interpreted by PER's or saturated sets, and proof terms are interpreted by syntactic constructs such as recursive functions or λ-terms. Although such an interpretation keeps quite close to the 'proofs-as-programs' paradigm, it is hard to use it—for not saying impossible—to interpret uncountable types at the impredicative level.

For that, a simple solution is to interpret the programs at the impredicative level not by syntax, but by the points of a λ-model in the category of coherence spaces that will play the same rôle.

Now, we will assume that $\mathfrak{k}$ is an infinite cardinal, and that $\mathcal{K}$ is a coherence space such that $[\mathcal{K} \xrightarrow{\mathfrak{k}} \mathcal{K}] \Subset \mathcal{K}$. Such a λ-model (for the β-reduction only) can be easily built by solving for example the equation $\mathcal{K} = \mathcal{B} \oplus [\mathcal{K} \xrightarrow{\mathfrak{k}} \mathcal{K}]$, where $\mathcal{B}$ denotes a coherence space used for representing ground values.

In the following, the coherence space $\mathcal{K}$ will be refered to as the space of *computational values*, and its points and atoms will be called the *computational points* and the *computational atoms* of $\mathcal{K}$ respectively.

4.3 $\mathcal{K}$-coherence spaces

A $\mathcal{K}$-coherence space is a coherence space $\mathcal{A}$ equipped with a coherence space $||\mathcal{A}||$, called its *computational part*, and two rigid embeddings

$$\mathcal{A} \overset{i_{\mathcal{A}}}{\hookleftarrow} ||\mathcal{A}|| \overset{i_{\mathcal{K}}}{\rightarrowtail} \mathcal{K}$$

Atoms and points of $||\mathcal{A}||$ are respectively called the *computational atoms* and the *computational points* of $\mathcal{A}$. Because of the two rigid embeddings, any atom (or point) of $||\mathcal{A}||$ can be seen both as an atom (or point) of $\mathcal{A}$ and as an atom (or point) of $\mathcal{K}$. In the following, we will often identify each atom α of $||\mathcal{A}||$ to its images $i_{\mathcal{A}}(\alpha)$ and $i_{\mathcal{K}}(\alpha)$ in $|\mathcal{A}|$ and $|\mathcal{K}|$ respectively.

For all $a \in \mathcal{A}$, we denote by $||a||$ the *computational contents* of a, which is defined by

$$||a|| \;=\; i^-_{\mathcal{A}}(a)$$

where $i^-_{\mathcal{A}} : \mathcal{A} \twoheadrightarrow ||\mathcal{A}||$ denotes the retraction associated to the rigid embedding $i_{\mathcal{A}} : ||\mathcal{A}|| \rightarrowtail \mathcal{A}$.

In the following, $\mathcal{K}$ will be considered as a $\mathcal{K}$-coherence space itself, simply by setting $||\mathcal{K}|| = \mathcal{K}$, both associated rigid embeddings being defined as the identity function.

4.4 $\mathcal{K}$-morphisms

Let $\mathfrak{a}$ be an infinite cardinal greater or equal to $\mathfrak{k}$, and $\mathcal{A}$, $\mathcal{B}$ two $\mathcal{K}$-coherence spaces. A $(\mathfrak{a},\mathcal{K})$-morphism—or, shortly, a $\mathcal{K}$-morphism—is a $\mathfrak{a}$-stable function $F : \mathcal{A} \to \mathcal{B}$ such that there exists a $\mathfrak{k}$-stable map $||F|| : ||\mathcal{A}|| \to ||\mathcal{B}||$ such that the following diagram commutes

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow[\mathfrak{a}]{F} & \mathcal{B} \\ {\scriptstyle ||.||}\downarrow & & \downarrow{\scriptstyle ||.||} \\ ||\mathcal{A}|| & \xrightarrow[\mathfrak{k}]{||F||} & ||\mathcal{B}|| \end{array}$$

that is:

$$\forall a \in \mathcal{A} \quad ||F(a)|| \;=\; ||F||\big(||a||\big).$$

Intuitively, the existence of $||F||$ means that

1. for all $a \in \mathcal{A}$, the computational contents of $F(a)$ only depends on the computational contents of a, and
2. the function $||F||$ which expresses that dependency is a $\mathfrak{k}$-stable function

Also notice that $||F||$, if it exists, is unique.

Fact 4.1 — *A function $F \in \mathcal{A} \xrightarrow{\mathfrak{a}} \mathcal{B}$ is a $\mathcal{K}$-morphism iff all the atoms $(a_0,\beta) \in \mathsf{Tr}(F)$ satisfy the condition*

$$\beta \in \big|||\mathcal{B}||\big| \quad \Rightarrow \quad a_0 \in ||\mathcal{A}|| \;\wedge\; \mathrm{Card}(a_0) < \mathfrak{k}.$$

Informally, a $\mathcal{K}$-morphism is an $\mathfrak{a}$-stable function such that every computational atom in its range is produced only by computational atoms in its domain, in a $\mathfrak{k}$-continuous way. As a consequence, the set of (traces of) $\mathcal{K}$-morphisms forms a coherence space, which is denoted by $\mathcal{A} \xrightarrow[\mathcal{K}]{\mathfrak{a}} \mathcal{B}$. Moreover, this coherence space is also a $\mathcal{K}$-coherence space, whose computational part is given by

$$||\mathcal{A} \xrightarrow[\mathcal{K}]{\mathfrak{a}} \mathcal{B}|| \;=\; ||\mathcal{A}|| \xrightarrow{\mathfrak{k}} ||\mathcal{B}|| \quad \Subset \quad \mathcal{K} \xrightarrow{\mathfrak{k}} \mathcal{K} \quad \Subset \quad \mathcal{K}.$$

By definition, the category of $(\mathfrak{a},\mathcal{K})$-*coherence spaces* is the category whose objects are $\mathcal{K}$-coherence spaces and whose morphisms are $(\mathfrak{a},\mathcal{K})$-morphisms.

Proposition 4.2 — *For all infinite cardinal $\mathfrak{a} \geqslant \mathfrak{k}$, the category of $(\mathfrak{a},\mathcal{K})$-coherence spaces is a Cartesian closed category.*

In the following, we will work in the category of $(\mathfrak{a},\mathcal{K})$-coherence spaces. Consequently, we need to do some minor changes in the definitions and results of section 3, by replacing '$\mathfrak{a}$-stable function' by '$(\mathfrak{a},\mathcal{K})$-morphism'—this essentially concerns the definition of the dependent product $\Pi(a\in X; Y_a)$, which now contains only $(\mathfrak{a},\mathcal{K})$-morphisms. We also need to give the structure of a $\mathcal{K}$-coherence space to $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})$, by simply setting $||\,\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})|| = 0$, which means that the computational contents of types as values is empty.

All the results stated in that section remain true in the category of $(\mathfrak{a},\mathcal{K})$-coherence spaces. In particular, lemma 3.6 remains true in the new category: the functions π^0 and π^1 are $\mathcal{K}$-morphisms, since they have no computational contents (*i.e.* $||\pi^0|| = ||\pi^1|| = \varnothing$).

Proposition 4.3 — *Let $\mathcal{A}$ and $\mathcal{B}$ be two coherence spaces equipped with $(\mathfrak{a},\mathcal{K})$-morphisms, X a type basis of $\mathcal{A}$ and $(Y_a)_{a\in X}$ a family of type bases of $\mathcal{B}$ indexed over X. If $Y_a \subset \mathcal{K}$ for all $a \in X$, then*

$$\forall (a\in X; Y_a) \subset \mathcal{K} \quad \textit{and} \quad \Pi(a\in X; Y_a) \in \mathcal{K}.$$

In other words, type bases of $\mathcal{K}$ are the good candidates for interpreting impredicativity.

5 A model for ICC^-

5.1 Building the model

Using the ideas expressed above, it is now quite simple to build a model of the restricted implicit calculus in which Set will be interpreted in a non-trivial way, by looking for solutions of the equation

$$\mathcal{A} \;=\; \mathcal{K} \underset{\mathcal{K}\xrightarrow{\mathfrak{k}}\mathcal{K}}{\oplus} [A \xrightarrow[\mathcal{K}]{\mathfrak{a}} \mathcal{A}] \oplus \mathsf{Ty}_{\mathfrak{a}}(\mathcal{A}) \qquad (1)$$

Intuitively, this equation says that a point of $\mathcal{A}$ is either a computational value, a function, or a small type of $\mathcal{A}$. Notice that in the first coalesced sum, we need to identify the copies of points of $\mathcal{K} \xrightarrow{\mathfrak{k}} \mathcal{K}$ which are both in $\mathcal{K}$ and in $\mathcal{A} \xrightarrow[\mathcal{K}]{\mathfrak{a}} \mathcal{A}$.

Proposition 5.1 — *For each infinite cardinal $\mathfrak{a}$, the equation (1) has solutions.*

We now need to give some lower bound on the cardinal $\mathfrak{a}$. In the following, we will assume that there is an ω-sequence $(\mathfrak{a}_i)_{i>0}$ of inaccessible cardinals such that:

1. $\mathfrak{a}_1 > \mathrm{Card}(|\mathcal{K}|)$;
2. $\mathfrak{a}_{i+1} > \mathfrak{a}_i$ for all $i > 0$.

Theorem 5.2 — *If $\mathfrak{a}$ is a cardinal such that $\mathfrak{a} > \mathfrak{a}_i$ for each $i > 0$, then each solution of (1) is a model for* ICC^-.

For proving that result, we essentially have to interpret the universe hierarchy and both products. For that, we need to introduce for each $i > 0$ the coherence space $\mathcal{A}_i$ defined as the smallest solution of the equation

$$\mathcal{A}_i \;=\; \mathcal{K} \;\underset{\mathcal{K}\xrightarrow{\mathrm{f}}\mathcal{K}}{\oplus}\; [A \xrightarrow[\mathcal{K}]{\mathfrak{a}_i} \mathcal{A}] \oplus \mathsf{Ty}_{\mathfrak{a}_i}(\mathcal{A}).$$

Intuitively, each $\mathcal{A}_i$ is a partial model corresponding to the first i-levels of the universe hierarchy. In particular, we have

$$\mathcal{A}_1 \Subset \mathcal{A}_2 \Subset \cdots \Subset \mathcal{A}_i \Subset \mathcal{A}_{i+1} \Subset \cdots \Subset \mathcal{A}.$$

Moreover, it is easy to check that for each level $i > 0$ we have $\mathrm{Card}(|\mathcal{A}_i|) = \mathfrak{a}_i$, hence

$$\langle \mathsf{Ty}^*_{\mathfrak{a}_i}(\mathcal{A}_i)\rangle \;\in\; \mathsf{Ty}^*_{\mathfrak{a}_{i+1}}(\mathcal{A}_{i+1}) \;\subset\; \mathcal{A}_{i+1}.$$

Now, we can interpret universes as follows:

- $[\![\mathsf{Prop}]\!] = \langle\{\langle \mathbf{0}\rangle;\langle \mathbf{1}\rangle\}\rangle$;
- $[\![\mathsf{Set}]\!] = \langle \mathsf{Ty}^*_{\mathfrak{a}_1}(\mathcal{K})\rangle$;
- $[\![\mathsf{Type}_i]\!] = \langle \mathsf{Ty}^*_{\mathfrak{a}_i}(\mathcal{A}_i)\rangle$.

Notice that because $\mathrm{Card}(|\mathcal{K}|) \leqslant \mathfrak{a}_1$, each type basis of $\mathcal{K}$ is a small type w.r.t $\mathfrak{a}_1$. Hence $\mathsf{Ty}^*_{\mathfrak{a}_1}(\mathcal{K})$ contains the value associated to any type basis of $\mathcal{K}$, without regarding its cardinal. This point is important, since we can not predict in advance which types of $\mathcal{K}$ will be useful in the construction, due to the impredicative nature of $\mathcal{K}$ in the model—remember that type bases of $\mathcal{K}$ can be built by explicit/implicit quantifications over types living in higher levels in the hierarchy. Also notice that the condition $\mathrm{Card}(|\mathcal{K}|) < \mathfrak{a}_1$ implies that

$$\langle \mathsf{Ty}^*(\mathcal{K})\rangle \;\in\; \mathsf{Ty}^*_{\mathfrak{a}_1}(\mathcal{A}_1) \;\subset\; \mathcal{A}_1.$$

Products are interpreted at each level $\mathcal{A}_i$ by constants $\pi^0_i, \pi^1_i \in \mathcal{A}_i$ defined the same way as in paragraph 3.5. Then we have $\pi^0_i \leqslant_{\mathbf{B}} \pi^0_{i+1}$ and $\pi^1_i \leqslant_{\mathbf{B}} \pi^1_{i+1}$ for all $i > 0$, so we can set

$$\pi^0 \;=\; \bigcup_{i>0} \pi^0_i \quad \text{and} \quad \pi^1 \;=\; \bigcup_{i>0} \pi^1_i.$$

(Note that we can not define π^0 and π^1 once for all for the whole model, since $\mathsf{Ty}_{\mathfrak{a}}(\mathcal{A})$ is not necessary closed by dependent product and intersection—remember that $\mathfrak{a}$ is not necessary an inaccessible cardinal, contrary to the intermediate cardinals $\mathfrak{a}_i$.)

5.2 Interpreting terms

For defining the interpretation of terms, it is convenient to consider the denotations $\Pi x\!:\!T\,.\,U$ and $\forall x\!:\!T\,.\,U$ as shorthands for

$$\begin{array}{rcl} \Pi x\!:\!T\,.\,U & \equiv & \Pi\, T\, \lambda x\,.\,U \\ \forall x\!:\!T\,.\,U & \equiv & \forall\, T\, \lambda x\,.\,U \end{array}$$

by viewing ICC as a Curry-style λ-calculus with constants Prop, Set, Type_i, Π and $\forall$.

As usual, the interpretation of a term is parametrized by a *valuation*, which is a map ρ from the set of variables to the model $\mathcal{A}$. If ρ is a valuation, x a variable and a a point of $\mathcal{A}$, then $(\rho; x \leftarrow a)$ denotes the valuation obtained by remapping x to a in ρ.

The interpretation $(M, \rho) \mapsto [\![M]\!]_\rho$ is defined inductively on M as follows :

$$\begin{array}{rcl} [\![x]\!]_\rho & = & \rho(x) \\ [\![\lambda x\,.\,M]\!]_\rho & = & a_{\in\mathcal{A}} \mapsto [\![M]\!]_{(\rho;x\leftarrow a)} \\ [\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) \\ [\![\Pi]\!]_\rho = \pi^1 \quad [\![\forall]\!]_\rho = \pi^0 \quad [\![s]\!]_\rho = [\![s]\!] & & (s \in \mathcal{S}). \end{array}$$

Notice that here, we interpret terms—even ill-typed ones—instead of interpreting judgments or derivations

Lemma 5.3 ($\beta\eta$-reduction) — *Let M and M' be terms, and ρ a valuation. Then*

- *$M \to_\beta M'$ implies $[\![M]\!]_\rho = [\![M']\!]_\rho$;*
- *$M \to_\eta M'$ implies $[\![M]\!]_\rho \subset [\![M']\!]_\rho$;*

It is important to note here that η-convertible terms have not necessary the same denotation in the model, even if they are well-typed (consider $\lambda x\,.\,x$ and $\lambda f x\,.\,f\ x$). Nevertheless, the situation becomes simpler if $M \twoheadrightarrow_{\beta\eta} M'$ when both denotations $[\![M]\!]_\rho$ and $[\![M']\!]_\rho$ are types-as-values in the model: in that case, we have the equality $[\![M]\!]_\rho = [\![M']\!]_\rho$ due to the trivial ordering between types-as-values. This argument will allow us to interpret the convertibility rule of ICC in the model, which asserts the extensional equivalence of $\beta\eta$-convertible types.

Let $\Gamma = [x_1 : T_1; \ldots; x_n : T_n]$ be a context. A valuation ρ is said to be *adapted* to Γ if for all $i \in [1..n]$ there exists a type basis $X_i \subset \mathcal{A}$ such that $[\![T_i]\!]_\rho = \langle X_i\rangle$ and $\rho(x) \in \overline{X_i}$.

Proposition 5.4 (Soundness) — *In* ICC^-, *$\Gamma \vdash M : T$ implies that for each valuation ρ adapted to Γ there exists a type basis $X \subset \mathcal{A}$ such that*

$$[\![T]\!]_\rho \;=\; \langle X\rangle \quad \textit{and} \quad [\![M]\!]_\rho \in \overline{X}.$$

Corollary 5.5 — ICC^- *is consistent.*

Proof. Check that $[\![\Pi A : \mathsf{Prop} \,.\, A]\!] = \langle \mathbf{0} \rangle$.

Since this model is clearly a model of ECC too, the proof outlined above is also another proof of the consistency of ECC.

5.3 Choosing the space $\mathcal{K}$

Interpreting uncountable impredicative types As noticed in paragraph 4.2, a simple way for building the coherence space $\mathcal{K}$ is to solve the equation

$$\mathcal{K} = \mathcal{B} \oplus [\mathcal{K} \xrightarrow{\mathfrak{k}} \mathcal{K}]$$

parametrized by a coherence space $\mathcal{B}$. A convenient choice for $\mathcal{B}$ is to take $\mathcal{B} = X_\perp$, where $X_\perp$ denotes the flat coherence space[7] built by *lifting* an arbitrary set X. The main advantage of this construction is that any function $f : X \to \mathcal{A}$ can be lifted to an $(\mathfrak{a}, \mathcal{K})$-morphism $f_\perp : \mathcal{K} \xrightarrow[\mathcal{K}]{\mathfrak{a}} \mathcal{A}$, without assuming any requirement on f.

By this way, it is quite simple to prove that we can safely axiomatize, say, the theory of real numbers at the impredicative level[8] without breaking the consistency of the underlying theory (here, ECC or ICC^-), by taking $X = \mathbb{R}$ and by checking carefully that all the primitive predicates and axioms of real number theory are realized by some points in the model—which is here the case.

ZF-set theory at the impredicative level A more ambitious example is to consider a set X satisfying the following criterions:

1. if $y \in x$ and $x \in X$ then $y \in X$;
2. $\mathbb{N} \in X$
3. if $x \in X$ then $\mathfrak{P}(x) \in X$;
4. if $(x_i)_{i \in y}$ is a family of elements of X indexed over an element $y \in X$, then $\bigcup_{i \in y} x_i \in X$.

In set theory [8], we can prove that such a set exists provided we assume the existence of an inaccessible cardinal. In fact, such a set X is a model of ZF-set theory, since it is stable by 'all the operations' that we can define in the usual set theory (*i.e.* without any inaccessible cardinal). Consequently, if we use X for building $\mathcal{K}$, and $\mathcal{K}$ for building $\mathcal{A}$, then we can show that $\mathcal{A}$ is a model of an axiomatization of ZF-set theory on the impredicative level (within ECC or ICC^-).

[7] $X_\perp$ is defined as the coherence space whose web is equal to X, and whose coherence relation is the identity. Points of $X_\perp$ are the singletons and the empty set.

[8] By this, we mean that the type of real numbers is declared in Set, whereas the axioms of real numbers are declared in Prop

Nevertheless, this result is not so immediate, since this axiomatization of set theory in type theory leads to a higher-order set-theory, which is logically stronger than ZF-set theory. The key point is that we used here a set-theoretical model of ZF, which is known to be not only a model of the usual first-order ZF-set theory, but also a model for the higher-order set theory.

This result would not hold if we had replaced X by a countable model of ZF-set theory given by the Lövenheim-Skolem theorem for example. Such an attempt would fail immediately for the interpretation of the comprehension scheme. Indeed, the interpretation of the comprehension scheme requires that for all $x \in X$ and for all predicate $p \in [\![\mathrm{ZF} \to \mathsf{Prop}]\!]$, there exists a denotation $y \in X$ which represents the set $\{z \in x;\ p(z)\}$. But in the countable model of ZF given by the Lövenheim-Skolem theorem, such a denotation exists if and only if p is (the denotation of) a predicate of first-order set-theory, which is far from being the case for all p belonging to $[\![\mathrm{ZF} \to \mathsf{Prop}]\!]$, which also contains all the (denotations of) higher-order predicates built by using quantifications on higher-order types in the universe hierarchy.

Beware of dangerous skolemizations When axiomatizing ZF-set theory on the impredicative level of ECC (or ICC^-), one must introduce skolemization symbols with the utmost care. Although usual skolem symbols for denoting the powerset or the union of (the elements of) a given set can be easily interpreted in our model, it is not possible to interpret in our framework a symbol

$$c : \mathrm{ZF} \to (\mathrm{ZF} \to \mathsf{Prop}) \to \mathrm{ZF}$$

such that $(c\ x\ p)$ represents the set $\{z \in x;\ p(z)\}$. Indeed, our interpretation of impredicativity is based on the fact that the computational contents of $f(x)$ only depends on the computational contents of x. For that reason, the denotation of such a symbol c must be constant w.r.t its second argument—which has no computational contents—which is clearly impossible if c is intended to interpret a skolem symbol for the comprehension scheme.[9] However, the problem disappears by writing the axioms of ZF in the style of first-order theories.

6 Future work

Connections with Ludics At the time we discovered this model, we didn't know anything about Ludics [7]. Nevertheless, it turns out that many notions defined in section 3

[9] The same problem arises if we want to introduce a symbol expressing the l.u.b of a bounded subset of $\mathbb{R}$ given by a predicate over the real numbers, in order to express the completeness axiom.

are very close to ideas developed by Girard in that theory.[10] For this reason, there is no doubt that a better understanding of the relationship between the present work and Ludics could help us to improve the model.

Strong normalization It is reasonable to think that the ideas introduced in this paper could be used also for building strong normalization models. Technically, we can add normalization information in the atoms $\tau(X)$ representing types as values, by giving to them a Λ-set structure [1]. But for achieving this goal, we first need to modify the model in such a way that all types becomes inhabited, since we want to interpret terms in all contexts. Actually, such a modification seems to be difficult, mainly in presence of intersection types. Finally, it is interesting to notice that within a strong normalization model of ICC^{-}, the (STR) rule becomes a special case of the (INST) rule—due to the fact that the interpretations of all types are inhabited—hence such a model is also a normalization model for the whole system ICC.

Interpreting inductive types A major improvement of our work would be the interpretation of a calculus with inductive types, such as the Calculus of Inductive Constructions (CIC) [11, 12]. Yet, we don't have any precise result concerning that point, but it seems that the model already contains all the material necessary for achieving such an interpretation.

7 Conclusion

In this paper, we have described a new model for interpreting impredicative type universe. We have shown that types, universes, dependent products, intersection types and subtyping can be easily interpreted in coherence spaces.

Our main result is that the impredicative level of large type theories like ECC can be interpreted in a parametrized way, by using an arbitrary λ-model in the category of coherence spaces for representing inhabitants of impredicative types.

On the other hand, the whole model is itself a model for the untyped λ-calculus (for the β-rule) if we consider types-as-values simply as ground values by forgetting the logical information that they carry.

Finally, the fact that this model allows us to prove the consistency of non-trivial axiomatizations on the impredicative level of type theories with universes illustrates the best its strength.

[10] In particular, the definition of semantical types is very close to the definition of behaviours in Ludics, and the relation between a semantical type and its type basis is exactly the same as the relation between a behaviour and its incarnation.

References

[1] T. Altenkirch. Constructions, Inductive types and Strong Normalization. Ph.D. Thesis, University of Edinburgh, 1993.

[2] B. Barras, S. Boutin, C. Cornes, J. Courant, J. Filliatre, E. Giménez, H. Herbelin, G. Huet, C. Muñoz, C. Murthy, C. Parent, C. Paulin, A. Saïbi, B. Werner. The Coq Proof Assistant Reference Manual - Version V6.1, research report No0203, INRIA, 1997.

[3] P. Giannini, F. Honsell, S. Ronchi. Type inference: some results, some problems. *Fundamenta Informaticae*, 19(1,2):87–126, 1993.

[4] J.H. Geuvers, The Church-Rosser property for $\beta\eta$-reduction in typed lambda calculi. In Proceedings of the seventh annual symposium on Logic in Computer Science, Santa Cruz, Cal., IEEE, pp 453–460.

[5] J. H. Geuvers, M. J. Nederhof. A modular proof of Strong Normalization for the Calculus of Constructions. Journal of Functional Programming 1,2(1991), 155-189.

[6] J.-Y. Girard, Y. Lafont, P. Taylor. Proofs and Types. Cambridge University Press, 1989.

[7] J.-Y. Girard. Locus Solus. 1999. Private communication.

[8] J.-L. Krivine. Théorie des ensembles. Cassini, 1998.

[9] Z. Luo. Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press, 1994.

[10] A. Miquel. Arguments implicites dans le Calcul des Constructions: étude d'un formalisme à la Curry. Mémoire de DEA. Université Paris VII, 1998. `http://pauillac.inria.fr/~miquel/`

[11] C. Paulin-Mohring. Définitions inductives en Théorie des Types d'Ordre Supérieur. Habilitation à diriger des recherches, Université Claude Bernard, Lyon I, 1996.

[12] B. Werner. Une théorie des Constructions Inductives. PhD Thesis, Université Paris VII, 1994.