ECC, an Extended Calculus of Constructions

Zhaohui Luo*
Department of Computer Science
University of Edinburgh
King's Buildings
Edinburgh EH9 3JZ, U.K.

Abstract

We present a higher-order calculus **ECC** which can be seen as an extension of the calculus of constructions [CH88] by adding strong sum types and a fully cumulative type hierarchy. **ECC** turns out to be rather expressive so that mathematical theories can be abstractly described and abstract mathematics may be adequately formalized. It is shown that **ECC** is strongly normalizing and has other nice proof-theoretic properties. An ω–**Set** (realizability) model is described to show how the essential properties of the calculus can be captured set-theoretically.

1 Introduction

The calculus of constructions [CH88][Coq85] is a typed higher-order functional calculus which provides a nice formalism for constructive proofs in natural deduction style and can also be seen as a high-level functional programming language.

In this paper, we present an *Extended Calculus of Constructions*, **ECC**, which can be seen as an extension of the calculus of constructions with

- Σ*-types* (or, *strong sum types*), and
- *a fully cumulative type hierarchy*.

Σ-types in **ECC**, together with the type hierarchy, provides a powerful abstraction mechanism so that mathematical *theories* can be abstractly described and structured, leading to a comprehensive structuring of mathematical texts in interactive proof development and program specifications. The cumulative type hierarchy also increases the expressiveness in another aspect so that, for example, abstract mathematics (*e.g.*, abstract algebras, categories) may be *adequately* formalized. Furthermore, as the type hierarchy provides a rather strong and flexible form of polymorphism, **ECC** provides a potential higher-order module mechanism which supports *structure sharing by parameterization* in the style of programming language Pebble [BLam84][LB88][Bur84] where the type of all types exists and plays an important role.

The infinite type hierarchy in **ECC** is similar to that of Martin-Löf's type theory but is *fully cumulative* in the following sense. First, the lowest level $Prop$ is impredicative, and the propositions at this level are *lifted* as higher-level types. This lifting is essential for Σ-types in **ECC** to play their role as an abstraction mechanism, and it solves the technical difficulty that adding (type-indexed) strong sum to the impredicative proposition level of Constructions results in an inconsistent system in which Girard's paradox can be derived [Coq86a]. Secondly, type inclusions between the type universes are coherently expanded to the other types so that a strong form of type unicity is achieved; this yields a simple notion of *principal type* and a simple algorithm for type inference.

ECC has good proof-theoretic properties. Particularly, it is *strongly normalizing*, which shows the proof-theoretic consistency of **ECC** (and in general, Constructions with an infinite type hierarchy) and establishes the theoretical basis of an implementation (*e.g.*, decidability of convertibility and type checking).

We give an (intuitionistic) set-theoretic semantics of **ECC** in the framework of ω-sets [Mog85][LM88] [Hyl87], which captures the intuitive meanings of the constructs in the calculus and reflects its essential properties. In addition to its importance in getting better understanding of the calculus, such a model-theoretic semantics seems also useful when considering pragmatics of the calculus, *e.g.*, how to formalize mathematical problems *adequately* in Constructions.

We discuss how to structure mathematical texts

*Supported by a studentship of the University of Edinburgh and the ORS award.

and express abstract theories in **ECC**. Sharing by parametrization is explained by an example. Weak existential types are also discussed.

2 ECC

ECC consists of an underlying term calculus and a set of inference rules of judgements. The basic expressions of the term calculus, called *terms*, are inductively defined by the following clauses:

- The constants $Prop$ and $Type_i$ ($i \in \omega$), called *kinds*, are terms;
- Variables (x,y,...) are terms;
- If A, B, M and N are terms, so are $\Pi x{:}A.B$, $\lambda x{:}A.M$, MN, $\Sigma x{:}A.B$, $\mathbf{pair}_{\Sigma x:A.B}(M,N)$, $\pi_1(M)$ and $\pi_2(M)$.

Free and bound occurrences of variables and substitution $[N/x]M$ are defined as usual. Terms which are the same up to changes of bound variables are identified (we will use $\equiv$ for identity). Reduction ($\triangleright$) and conversion ($\simeq$) are defined as usual with respect to the following one-step $\beta\sigma$ contraction schemes:

$$(\lambda x{:}A.M)N \triangleright_1 [N/x]M$$

$$\pi_j(\mathbf{pair}_{\Sigma x:A.B}(M_1,M_2)) \triangleright_1 M_j \quad (j=1,2)$$

Remark Church-Rosser property holds for this term calculus, *i.e.*, $M_1 \simeq M_2 \;\Rightarrow\; \exists M.\; M_1 \triangleright M \wedge M_2 \triangleright M$. For the term calculus, the inclusion of either η-reduction or the rule for surjective pairing will make Church-Rosser fail [vD80][Klo80]. It is worth remarking that, with either of them, Church-Rosser for well-typed terms of **ECC** also fails because of the existence of type inclusions induced by universes. □

The kinds, also called *type universes*, and the type inclusions between them induce the type cumulativity that is syntactically characterized by the following partial order.

Definition 2.1 (type cumulativity) *Define $\preceq$ as the smallest partial order over terms w.r.t. conversion $\simeq$* [1] *such that*

1. $Prop \preceq Type_0 \preceq Type_1 \preceq \ldots$;
2. *if* $A \simeq A'$ *and* $B \preceq B'$, *then* $\Pi x{:}A.B \preceq \Pi x{:}A'.B'$;
3. *if* $A \preceq A'$ *and* $B \preceq B'$, *then* $\Sigma x{:}A.B \preceq \Sigma x{:}A'.B'$.

$A \prec B$ *if, and only if,* $A \preceq B$ *and* $A \not\simeq B$. □

Contexts are finite sequences of expressions of the form $x{:}M$, where x is a variable and M is a term. The empty context is denoted by $\langle\rangle$. *Judgements* are of the form Γ *is valid* or $\Gamma \vdash M{:}A$, where Γ is a context and M and A are terms. The sets of free variables in a context $x_1{:}A_1,...,x_n{:}A_n$ and a judgement $\Gamma \vdash M{:}A$ are defined as $\bigcup_{1\le i\le n}\{x_i\} \cup FV(A_i)$ and $FV(\Gamma)\cup FV(M)\cup FV(A)$, respectively.

The following are the *inference rules* of **ECC**, where $i \in \omega$:

$$(C1) \quad \frac{}{\langle\rangle \text{ is valid}}$$

$$(C2) \quad \frac{\Gamma \vdash A{:}Type_i}{\Gamma, x{:}A \text{ is valid}} \quad (x \notin FV(\Gamma))$$

$$(var) \quad \frac{\Gamma, x{:}A, \Gamma' \text{ is valid}}{\Gamma, x{:}A, \Gamma' \vdash x{:}A}$$

$$(K1) \quad \frac{\Gamma \text{ is valid}}{\Gamma \vdash Prop{:}Type_0}$$

$$(K2) \quad \frac{\Gamma \text{ is valid}}{\Gamma \vdash Type_i{:}Type_{i+1}}$$

$$(\Pi 1) \quad \frac{\Gamma, x{:}A \vdash P{:}Prop}{\Gamma \vdash \Pi x{:}A.P{:}Prop}$$

$$(\Pi 2) \quad \frac{\Gamma \vdash A{:}Type_i \quad \Gamma, x{:}A \vdash B{:}Type_i}{\Gamma \vdash \Pi x{:}A.B{:}Type_i}$$

$$(\lambda) \quad \frac{\Gamma, x{:}A \vdash M{:}B}{\Gamma \vdash \lambda x{:}A.M{:}\Pi x{:}A.B}$$

$$(app) \quad \frac{\Gamma \vdash M{:}\Pi x{:}A.B \quad \Gamma \vdash N{:}A}{\Gamma \vdash MN{:}[N/x]B}$$

$$(\Sigma) \quad \frac{\Gamma \vdash A{:}Type_i \quad \Gamma, x{:}A \vdash B{:}Type_i}{\Gamma \vdash \Sigma x{:}A.B{:}Type_i}$$

$$(pair) \quad \frac{\Gamma \vdash M{:}A \quad \Gamma \vdash N{:}[M/x]B \quad \Gamma, x{:}A \vdash B{:}Type_i}{\Gamma \vdash \mathbf{pair}_{\Sigma x:A.B}(M,N){:}\Sigma x{:}A.B}$$

$$(\pi 1) \quad \frac{\Gamma \vdash M{:}\Sigma x{:}A.B}{\Gamma \vdash \pi_1(M){:}A}$$

$$(\pi 2) \quad \frac{\Gamma \vdash M{:}\Sigma x{:}A.B}{\Gamma \vdash \pi_2(M){:}[\pi_1(M)/x]B}$$

$$(conv) \quad \frac{\Gamma \vdash M{:}A \quad \Gamma \vdash A'{:}Type_i}{\Gamma \vdash M{:}A'} \quad (A \simeq A')$$

$$(cum) \quad \frac{\Gamma \vdash M{:}A \quad \Gamma \vdash A'{:}Type_i}{\Gamma \vdash M{:}A'} \quad (A \prec A')$$

[1] That is, $\simeq$ is the identity referred to in reflexivity and anti-symmetry of $\preceq$.

A *derivation* of a judgement J is a finite sequence of judgements $J_1, ..., J_n$ with $J_n \equiv J$ such that, for all $1 \leq i \leq n$, J_i is the conclusion of some instance of an inference rule whose premises are in $\{ J_j \mid j < i \}$. A judgement J is *derivable* if there is a derivation of J. A term M is *well-typed (under* Γ*)* if $\Gamma \vdash M{:}A$ is derivable for some A. We shall write $\Gamma \vdash M{:}A$ for '$\Gamma \vdash M{:}A$ is derivable', and $\Gamma \vdash M \simeq N$ ($\Gamma \vdash M \preceq N$) for '$M$ and N are well-typed under Γ and $M \simeq N$ ($M \preceq N$)', respectively.

This completes our formal presentation of **ECC**. **ECC** extends the calculus of constructions [CH88] [Coq85] by adding Σ-types and a cumulative type hierarchy. It can also be seen as an extension of the core of Martin-Löf's type theory (with infinite type universes) [ML84] by adding a lowest *impredicative* level of propositions (the types of type $Prop$).

The propositions, which stand for the logical formulas by Curry-Howard correspondence, constitute the *impredicative* level $Prop$ of the type hierarchy. Viewing intuitively types as sets, we have

$$Prop \in Type_0 \in Type_1 \in ...$$

$$Prop \subseteq Type_0 \subseteq Type_1 \subseteq ...$$

Particularly, every proposition is *lifted* as a higher-level type. In appearance, it seems that this would propagate the impredicativity at the level of propositions to the higher levels. For instance, we can derive

$$\vdash \Pi x{:}Type_i \Pi B{:}Type_i \rightarrow Prop.Bx{:}Type_0$$

However, the type hierarchy except the lowest level $Prop$ is still stratified (predicative) in the sense that the types can be ranked in such a way that the existence of a *proper type* (a type that is not convertible to any proposition) is only dependent on those with lower ranks (see section 3). This stratification of type hierarchy is essential for the logical consistency of the calculus.

The idea of lifting propositions as types is essential for Σ-types in **ECC** to be useful as an abstraction tool to express abstract mathematical theories (section 5). The reason is that adding (type-indexed) Σ-types to the impredicative level of constructions would get an inconsistent system in which Girard's paradox can be derived [Coq86a]. Note that, in **ECC**, $\Sigma x{:}A.P$ is *not* a proposition even when P is. However, as propositions are lifted as types, we can derive

$$(*) \qquad \frac{\Gamma \vdash A{:}Type_i \quad \Gamma, x{:}A \vdash P{:}Prop}{\Gamma \vdash \Sigma x{:}A.P{:}Type_i}$$

This $\Sigma x{:}A.P$ intuitively represents the set of pairs of an element a of A and a proof of the proposition $P(a)$, *i.e.*, the intuitionistic subset type (*c.f.*[ML84]). It is this property that enables propositions to be used to express abstract axioms of a mathematical theory expressed as a Σ-type.[2]

The type hierarchy is *fully cumulative*. The inference rule (*cum*) is a design decision which achieves a strong form of *type unicity* so that there is a simple notion of *principal type* (theorem 3.2) and a very straightforward algorithm for type inference (theorem 3.4). The type hierarchy presented in [Coq86a] does not have this property; and therefore, although every well-typed term has a *minimum type*, it is *not* the most general one [Luo88b].

The pairs are heavily typed to avoid the undesirable type ambiguity which would make type inference and type-checking difficult (perhaps impossible) [Luo88a]. But note that, thanks to the full cumulativity of types, we still have as expected, say,

$$\vdash \mathbf{pair}_{Type_0 \times Type_0}(Prop, Prop){:}Type_3 \times Type_3$$

Finally, it seems necessary to remark that the partial order $\preceq$ defined in definition 2.1 is *not* completely contravariant: in the second clause of the definition, A is required to be *convertible* to A' instead of $A' \preceq A$. One might take the later decision and the proof-theoretic properties in the next section would still hold. Even the algorithm for type inference would remain the same. The only difference from the proof-theoretic point of view is that some terms get more types. For example, $\lambda x{:}Type_1.x$ will not only have types $Type_1 \rightarrow Type_j$, but have $Prop \rightarrow Type_j$ and $Type_0 \rightarrow Type_j$ ($j \geq 1$) as its types as well. However, semantically, the type inclusions thus defined would be reflected by coercions instead of by set inclusions as we explain in section 4.

3 Proof-theoretic Properties

In this section, we show that **ECC** has nice proof-theoretic properties. Particularly, we prove that **ECC** is strongly normalizing and that there is a straightforward algorithm which computes the principal type of a well-typed term. As a consequence, **ECC** is decidable.

First, some basic properties are stated as the following theorem.

Theorem 3.1 *In* **ECC**, *we have*

1. *Any derivation of* $\Gamma, x{:}A, \Gamma' \vdash M{:}B$ *has a sub-derivation of* $\Gamma \vdash A{:}K$ *for some kind* K.

[2] We remark that, rather than including propositions as types as we do in this paper, one may instead directly use rules like $(*)$ above to gain similar effects.

2. *Any derivation of $\Gamma, \Gamma' \vdash M{:}A$ has a sub-derivation of 'Γ is valid'.*

3. *If $\Gamma \vdash M{:}A$ and Γ' is a valid context which contains every component of Γ, then $\Gamma' \vdash M{:}A$.*

4. *If $\Gamma, x{:}A, \Gamma' \vdash N{:}B$ and $\Gamma \vdash M{:}A$, then $\Gamma, [M/x]\Gamma' \vdash [M/x]N{:}[M/x]B$.*

5. *If $\Gamma \vdash M{:}A$, then $\Gamma \vdash A{:}K$ for some kind K.*

6. *(subject reduction) If $\Gamma \vdash M{:}A$ and $M \rhd N$, then $\Gamma \vdash N{:}A$.*

7. *(strengthening) If $\Gamma, y{:}Y, \Gamma' \vdash M{:}A$ and $y \notin FV(M{:}A) \cup FV(\Gamma')$, then $\Gamma, \Gamma' \vdash M{:}A$;*

8. *(characterizing $\preceq$) If A and B are both well-typed under Γ, then, $A \preceq B$ if, and only if, $\Gamma, x{:}A \vdash x{:}B$, where $x \notin FV(\Gamma)$.* □

Because of the type inclusions induced by type universes, type uniqueness (upto conversion) fails. However, we have a simple characterization of the set of types of a well-typed term.

Definition 3.1 (principal type) *A is called a* principal type *of M (under Γ) if $\Gamma \vdash M{:}A$ and, for all A' such that $\Gamma \vdash M{:}A'$, $A \preceq A'$.* □

In other words, a principal type of a well-typed term is its minimum type with respect to the partial order $\preceq$; and, if exists, it is obviously unique (upto conversion). We have

Theorem 3.2 (existence of principal types) *Every well-typed term M (under a context Γ) has a principal type.*

Proof Sketch The theorem follows the following *diamond property* of $\preceq$:

$$\Gamma \vdash M{:}A, \Gamma \vdash M{:}B \;\Rightarrow\; \exists C \preceq A, B. \Gamma \vdash M{:}C$$

which is proved by induction on derivations. □

The existence of the principal type is not only a good proof-theoretic property but very important to implementation of an interactive proof development system. We denote as T_M the principal type of M (under Γ). T_M is indeed the *most general* type of M in the following sense.

Fact 3.1 *$\Gamma \vdash M{:}A$ if, and only if, $T_M \preceq A$ and $\Gamma \vdash A{:}K$ for some kind K.* □

Now, we come to the most important result in this section.

Theorem 3.3 (strong normalization) **ECC** *is strongly normalizing, i.e., if $\Gamma \vdash M{:}A$, then M is strongly normalizable.* □

The proof of this theorem is rather difficult. The key difficulty is that, unlike Constructions without type hierarchy, in **ECC** not only propositions but also proper types can be of the form MN or $\pi_j(M)$. This makes it very difficult to define a rank assignment for types like the complexity measure β in [Coq86b] which is essential for proving (strong) normalization theorems of constructions-like calculi according to the insight of Coquand.

To solve this problem, we first prove a *quasi-normalization* result which says that *every well-typed term can be reduced to a term which does not contain any σ-redex or β-redex R_1R_2 such that R_1 has a type which is a proper type.*[3] This further implies that every well-typed proper type can be reduced to one of the following (head normal) forms:

$$K, \pi_{i_1}...\pi_{i_j}(xA_1...A_m)B_1...B_{m'}, \Pi x{:}A.B, \Sigma x{:}A.B$$

where K is a kind, x is a variable, $j, m, m' \geq 0$ and $i_k \in \{1, 2\}$. With this result, we can then define a two-dimensional complexity measure of types which enables us to prove the above theorem following a similar pattern of the proof of SN theorem for constructions in [Coq86b].

Remark The above result shows the proof-theoretic consistency of Constructions with an infinite type hierarchy; it also applies to the Generalized Calculus of Constructions presented in [Coq86a] (See [Luo88b]). □

Corollary 3.1 (consistency) **ECC** *is logically consistent. Particularly, we have, for any term M, $\not\vdash M{:}\Pi x{:}Prop.x$.* □

Corollary 3.2 (decidability of convertibility) *It is decidable whether $M \simeq N$ for arbitrary well-typed terms M and N.* □

As convertibility for well-typed terms is decidable, so is $\preceq$. Hence, we have

Theorem 3.4 (type inference) *There is a simple algorithm $\mathcal{T}$ such that, when given a context Γ and*

[3] We succeed in proving this by extending the way of using a measure adopted by G.Pottinger and J.Seldin in their attempt to prove the SN theorem for the calculus of constructions [Pot87][PS86]; it is essentially in the same spirit as that used in [Pra65] for higher-order logic, but more complex.

a term M, $\mathcal{T}$ checks whether M is well-types under Γ, and if so, $\mathcal{T}(\Gamma; M) = T_M$, where T_M is the principal type of M under Γ.

Proof Sketch The algorithm is just a straightforward extension of that for the calculus of constructions described in [Coq86b] which follows [ML71]. We only consider several cases here.

- $M \equiv M_1 M_2$: check whether $\mathcal{T}(\Gamma; M_1) \rhd \Pi x{:}A.B$ and $\mathcal{T}(\Gamma; M_2) \preceq A$; and, if so, $\mathcal{T}(\Gamma; M_1 M_2) = [M_2/x]B$.
- $M \equiv \Sigma x{:}M_1.M_2$: check whether $\mathcal{T}(\Gamma; M_1) \rhd K \in \{Prop, Type_i\}$ and $\mathcal{T}(\Gamma, x{:}M_1; M_2) \rhd K' \in \{Prop, Type_i\}$; and, if so, $\mathcal{T}(\Gamma; \Sigma x{:}M_1.M_2) = max_{\preceq}\{K, K', Type_0\}$.
- $M \equiv \mathbf{pair}_{\Sigma x:A.B}(M_1, M_2)$: check whether $\mathcal{T}(\Gamma; \Sigma x{:}A.B) \rhd K \in \{Prop, Type_i\}$, $\mathcal{T}(\Gamma; M_1) \preceq A$ and $\mathcal{T}(\Gamma; M_2) \preceq [M_1/x]B$; and, if so, $\mathcal{T}(\Gamma; \mathbf{pair}_{\Sigma x:A.B}(M_1, M_2)) = \Sigma x{:}A.B$.

The soundness and completeness of the algorithm can be proved as usual. □

Remark Note that convertible terms may have un-convertible principal types. For example, $(\lambda x{:}Type_3.x)Prop$ and $Prop$ have $Type_3$ and $Prop$ as their principal types, respectively. □

By fact 3.1, the existence of the type inference algorithm implies the decidability of type-checking.

Corollary 3.3 (decidability of type-checking) **ECC** *is decidable, i.e., it is decidable whether $\Gamma \vdash M{:}A$ for arbitrary Γ, M and A.* □

4 An ω–Set Model of ECC

In this section, we sketch a realizability model of **ECC** which gives an (intuitionistic) set-theoretic semantics of the calculus.[4] Such a model captures the intuitive meanings of the constructs in the calculus and reflects its essential properties such as logical consistency and type cumulativity.

The main question in interpreting **ECC** is how to interpret the type universes and the type formation operators Π and Σ so that, intuitively, we have

1. $Prop \in Type_0 \in Type_1 \in \ldots$;
2. $Prop \subseteq Type_0 \subseteq Type_1 \subseteq \ldots$;
3. $Type_i$ is closed under Π and Σ;
4. $Prop$ is closed under Π.

These requirements prevent us from giving a naive non-trivial classical set-theoretic model of **ECC**. (See [Rey84][RP88][LM88][Pit87] for more discussions for the second-order λ-calculus [Gir86][Rey74].)

Fortunately, the idea of interpreting types as partial equivalence relations [Gir72][Tro73][Mog85] provides us a nice framework of ω-sets and modest sets [Mog85][LM88][Hyl87] in which there is an interpretation of **ECC** satisfying the above requirements.

Let ω–**Set** be the category of ω-sets and **M** the category of modest sets. (See [LM88] for these notions. We use $|A|$ and $\Vdash_A$ to denote the carrier set and the realizability relation of an ω-set A.) The *interpretation* of **ECC**, defined by induction on derivations of judgements, gives every *derivable judgement* a unique denotation such that

- If Γ is a valid context, then $[\![\Gamma]\!] \in \mathrm{Obj}(\omega\text{–}\mathbf{Set})$;
- If $\Gamma \vdash M{:}A$, then $[\![\Gamma \vdash M{:}A]\!] : [\![\Gamma]\!] \to_{FPP} [\![\Gamma, x{:}A]\!]$, that is, intuitively, M is interpreted as a Γ-indexed element of A; (see below)
- If $\Gamma \vdash A \preceq A'$, then there is an inclusion morphism $\mathbf{inc}_{A,A'}{:}\, [\![\Gamma, x{:}A]\!] \hookrightarrow [\![\Gamma, x{:}A']\!]$, and furthermore, if $\Gamma \vdash M{:}A$, then $[\![\Gamma \vdash M{:}A']\!] = \mathbf{inc}_{A,A'} \circ [\![\Gamma \vdash M{:}A]\!]$;
- If $\Gamma \vdash M{:}A$ and $\Gamma \vdash N{:}A'$, $M \simeq N$ and $A \simeq A'$, then $[\![\Gamma \vdash M{:}A]\!] = [\![\Gamma \vdash N{:}A']\!]$.

In fact, we only have to indicate how to interpret $\Gamma \vdash M{:}T_M$, where T_M is the principal type of M under Γ. Different from traditional simpler cases, types and objects in constructions-like calculi are mixed up. So, a type in fact has a 'double identity' in the model. In this paper, we do not give the details of the interpretation (see [Luo88a] for how details can be filled in), but only emphasize on how to interpret Π, Σ and the type universes.

Before explaining the model, we first introduce three ω-set constructors σ, σ_Γ and π_Γ.

Suppose $\Gamma \in \mathrm{Obj}(\omega\text{–}\mathbf{Set})$ and $A{:}|\Gamma| \to \omega\text{–}\mathbf{Set}$. Then, define $\sigma(\Gamma, A)$ to be the following ω-set:

$$|\sigma(\Gamma, A)| =_{\mathrm{df}} \{\, (\gamma, a) \mid \gamma \in |\Gamma|, a \in |A(\gamma)| \,\}$$

$$\langle m, n\rangle \Vdash_{\sigma(\Gamma,A)} (\gamma, a) \overset{\mathrm{df}}{\Leftrightarrow} m \Vdash_\Gamma \gamma \wedge n \Vdash_{A(\gamma)} a$$

[4] We do *not* mean that what we describe is *the* model of the calculus. There are other *reasonable* models. For example, we can give a truth-value model of **ECC** where propositions are interpreted as 0 or 1. However, it seems that some basic points can not be missed to interpret the constructs properly; we hope to make them explicit here.

σ is used to interpret valid contexts. The empty context is interpreted as the terminal object of ω–**Set** and $[\![\Gamma, x{:}A]\!] =_{\mathrm{df}} \sigma([\![\Gamma]\!], [\![\Gamma \vdash A{:}T_A]\!])$. The notation $f{:}[\![\Gamma]\!] \rightarrow_{FPP} [\![\Gamma, x{:}A]\!]$ used above means f satisfies the following *first projection property*:

$$\forall \gamma \in [\![\Gamma]\!] .\mathrm{first}(f(\gamma)) = \gamma$$

Now, suppose $B{:}|\sigma(\Gamma, A)| \rightarrow \omega\text{–}\mathbf{Set}$. Then, $\sigma_\Gamma(A, B)$ and $\pi_\Gamma(A, B)$ are functions from $|\Gamma|$ to ω–**Set** defined as, for all $\gamma \in |\Gamma|$,

$$|\sigma_\Gamma(A, B)(\gamma)| =_{\mathrm{df}} \{\, (a, b) \mid a \in |A(\gamma)|, b \in |B(\gamma, a)| \,\}$$

$$\langle m, n\rangle \Vdash_{\sigma_\Gamma(A,B)(\gamma)} (a, b) \overset{\mathrm{df}}{\Leftrightarrow} m \Vdash_{A(\gamma)} a \wedge n \Vdash_{B(\gamma,a)} b$$

and, $|\pi_\Gamma(A, B)(\gamma)|$ is defined as

$$\{\, f \in \Pi a \in |A(\gamma)|.|B(\gamma, a)| \mid \exists n \in \omega.n \Vdash_{\pi_\Gamma(A,B)(\gamma)} f \,\}$$

where Π denotes set product, and $n \Vdash_{\pi_\Gamma(A,B)(\gamma)} f$ if, and only if,

$$\forall a \in |A(\gamma)| \forall p \in \omega.(p \Vdash_{A(\gamma)} a \Rightarrow np \Vdash_{B(\gamma,a)} f(a))$$

1. *Interpretation of $Type_i$ and Π/Σ-types.* To interpret $Type_i$, we consider large set universes. A basic insight is that the notions of ω-sets and modest sets have nothing to do with *sizes* of the sets under consideration. Consider ZFC set theory with infinite *inaccessible cardinals*[5] ($\kappa_0 < \kappa_1 < \ldots$) and let V_α be the cumulative hierarchy of sets. We define ω–**Set**(i) to be the full subcategory of ω–**Set** whose objects are those ω-sets whose carriers are in V_{κ_i}. Then, roughly speaking, $Type_i$ is interpreted as ω–**Set**(i). There are two points here. First, as V_{κ_i} is a 'model' of ZFC, we have

Lemma 4.1 *Both σ_Γ and π_Γ are closed for ω–**Set**(i), i.e., if $A{:}|\Gamma| \rightarrow \omega$–**Set**$(i)$ and $B{:}\sigma(\Gamma, A) \rightarrow \omega$–**Set**$(i)$, then $\sigma_\Gamma(A, B), \pi_\Gamma(A, B){:}|\Gamma| \rightarrow \omega$–**Set**$(i)$.* □

The interpretations of a Σ-type and a Π-type whose principal type is $Type_i$, $[\![\Gamma \vdash \Sigma x{:}A.B{:}Type_i]\!]$ and $[\![\Gamma \vdash \Pi x{:}A.B{:}Type_i]\!]$, are defined as

$$\sigma_{[\![\Gamma]\!]}([\![\Gamma \vdash A{:}Type_i]\!], [\![\Gamma, x{:}A \vdash B{:}Type_i]\!])$$

$$\pi_{[\![\Gamma]\!]}([\![\Gamma \vdash A{:}Type_i]\!], [\![\Gamma, x{:}A \vdash B{:}Type_i]\!])$$

respectively. The closedness requirement 3 is satisfied by the above lemma.

Secondly, as $V_{\kappa_i} \subseteq V_{\kappa_{i+1}}$, ω–**Set**(i) is a full subcategory of ω–**Set**$(i + 1)$. This justifies the requirement 2 for $Type_i$. But, how about the requirement 1? Note that ω–**Set**(i)'s are all *small* categories. Therefore, they can be naturally made as ω-sets in the following special way:

$$\Delta_i =_{\mathrm{df}} (\mathrm{Obj}(\omega\text{–}\mathbf{Set}(i)), \omega \times \mathrm{Obj}(\omega\text{–}\mathbf{Set}(i)))$$

As $V_{\kappa_i} \in V_{\kappa_{i+1}}$, we have $\Delta_i \in \mathrm{Obj}(\omega\text{–}\mathbf{Set}(i + 1))$.[6]

2. *Interpretation of $Prop$ and propositions.* The category of ω-sets has a *small* full subcategory **PROP** whose objects are those of the form $(Q(R), \in)$, where $Q(R)$ is the quotient set of a partial equivalence relation R (over ω).[7] *Prop*, roughly speaking, is interpreted as **PROP**. Similarly, **PROP** is a full subcategory of ω–**Set**(0) and $(\mathrm{Obj}(\mathbf{PROP}), \omega \times \mathrm{Obj}(\mathbf{PROP})) \in \mathrm{Obj}(\omega\text{–}\mathbf{Set}(0))$.

We have the following lemmas.

Lemma 4.2 *([LM88]) π_Γ is closed for the modest sets, that is, for all $A{:}|\Gamma| \rightarrow \omega$–**Set**, $B{:}|\sigma(\Gamma, A)| \rightarrow \mathbf{M}$, we have $\pi_\Gamma(A, B){:}|\Gamma| \rightarrow \mathbf{M}$.* □

Lemma 4.3 *There is an equivalence of categories* **back**:$\mathbf{M} \rightarrow \mathbf{PROP}$ *such that, for $P \in \mathrm{Obj}(\mathbf{PROP})$,* **back**$(P) = P$. *(The inverse of* **back** *is the inclusion functor.)* □

$[\![\Gamma \vdash \Pi x{:}A.B{:}Prop]\!]$, the interpretation of a Π-proposition, is defined as

$$\mathbf{back} \circ \pi_{[\![\Gamma]\!]}([\![\Gamma \vdash A{:}T_A]\!], [\![\Gamma, x{:}A \vdash B{:}Prop]\!])$$

The closedness requirement 4 is satisfied by the above two lemmas.

Remark The existence of the category equivalence **back** is important to interpret *Prop* and propositions properly, as Π is closed for **M** but *not* for **PROP**. Note that it is not correct to interpret *Prop* in the constructions-like calculi as **M** (as in [Ehr88]), because **M** is *not* a small category. □

This completes our sketch of the realizability model of **ECC**. It is necessary to remark that the logical consistency stated in corollary 3.1 also follows the above model construction. In fact, $\Pi x{:}Prop.x$ is interpreted as the empty ω-set $(\emptyset, \emptyset)$.

[5] A cardinal κ is (strongly) inaccessible if it is uncountable and regular, and, for all $\lambda < \kappa$, $2^\lambda < \kappa$.

[6] This explains what we mean by 'double identity' before. Note that $\mathbf{Set}(|A|, |B|) = \omega\text{–}\mathbf{Set}(A, B)$ when B is of the form $(|B|, \omega \times |B|)$.

[7] **PROP** is isomorphic to the category of partial equivalence relations. It is easy to verify that **PROP** is also a full subcategory of **M**.

The above model gives more information than just the consistency. First, it captures the intuitive meanings of the constructs in the calculus. For example, $M{:}A$ means '$M \in A$' and the syntactic type inclusions ($A \preceq A'$) are reflected semantically by set inclusions ($\mathbf{inc}_{A,A'}$). As pointed out at the end of section 2, if the type inclusion were completely contravariant, it could then only be possibly reflected by a sort of coercion instead of the set inclusion.

Secondly, maybe more important, it seems that such a semantics also shows how one can *adequately* formalize mathematical problems. For example, it seems to be *not* adequate to formalize a *theory of groups* by assuming the carrier of a group as $X{:}Prop$ as we know that X, as a proposition, can not be viewed as an *arbitrary set*. But, it seems that assuming $X{:}Type_0$ is then more adequate as, in the model above, we can view $Type_0$ as containing *almost all* sets. More research is needed in this aspect.

5 Theory Abstraction in ECC

We briefly discuss in this section one of the pragmatic aspects of **ECC** — expressing and structuring mathematical theories.

5.1 Σ-types and theory abstraction

Σ-types in **ECC**, together with the type hierarchy, can be used to express abstract mathematical theories to gain a comprehensive structuring of mathematical texts. For example, instead of postulating a theory for rings as a context of the form

$$X{:}Type_0, +{:}X \to X \to X, 0{:}X, ..., ass{:}P_{ASS_+}, id{:}P_0, ...$$

where the P's are the propositions for ring axioms, we may express the abstract theory for rings as the following Σ-type:

$$Ring \equiv \Sigma s{:}Sig_{Ring}.Ax_{Ring}(s)$$

where

$$\begin{array}{rcl} Sig_{Ring} & \equiv & \Sigma X{:}Type_0.(X \to X \to X) \times X \times ... \\ Ax_{Ring} & \equiv & \lambda s{:}Sig_{Ring}.P_{ASS_+} \wedge P_0 \wedge ... \end{array}$$

In general, (an abstract presentation of) a mathematical theory T (say, *Ring*) consists of

- a *signature presentation* Sig_T, which is in general a Σ-type, and
- the *abstract axioms* over the signature, which can be expressed as a predicate function Ax_T of type $Sig_A \to Prop$.

Then a theory T is the following Σ-type:

$$T \equiv \Sigma s{:}Sig_T.Ax_T(s)$$

The proved (abstract) *theorems* of T can then be expressed as a function Thm_T of type $Sig_T \to Prop$; their proofs constitute a function Prf_T of type $\Pi t{:}T.Thm_T(\pi_1(t))$. These theorems and their proofs can be 'instantiated' as the corresponding theorems and proofs for particular algebraic structure by λ-application. For example, one may instantiate the abstract theorems and their proofs of the abstract theory for rings to those concrete ones for integers.

Functions between abstract theories can also be defined which may capture the idea of *lifting* proofs from an abstract theory to an extended theory [TL88]. The type hierarchy even allows the above approach of theory abstraction to be internally expressed (an idea due to Coquand and Pollack); for this the fourth level of the type hierarchy ($Type_2$) is used.

5.2 Sharing by parameterization

The type hierarchy of **ECC** provides a reasonably strong form of polymorphism and hence a potential facility of defining *higher-order modules*. With this, one can define functions between abstracted modules and express *sharing by parameterization* [Bur84][LB88]. We show this by a simple example.

Example We define a function $ringGen$ which results in a ring structure when given as arguments a monoid and an abelian group with the same carrier and a proof of the extra axiom for the distributed laws. Suppose the theories of monoids and abelian groups are defined as follows:

$$Mon \equiv \Sigma m{:}\Sigma X{:}Type_0.Mwrt(X).Ax_{Mon}(m)$$

$$AGrp \equiv \Sigma g{:}\Sigma X{:}Type_0.AGwrt(X).Ax_{AGrp}(g)$$

where $Mwrt$, $AGwrt : Type_0 \to Type_0$ and, when given $X{:}Type_0$ as carrier, they give as results the Σ-types for the operations for monoids and abelian groups with respect to X, respectively, and $Ax_{Mon}(m)$ and $Ax_{AGrp}(g)$ are the propositions expressing the axioms of theories for monoids and abelian groups.

$ringGen$ can then be defined as follows (we omit the associated typings for pairs for readability):

$$ringGen \equiv \lambda X{:}Type_0$$

$$
\begin{aligned}
&\lambda(*,1){:}Mwrt(X)\\
&\lambda p_M{:}Ax_{Mon}(X,*,1)\\
&\lambda(+,0,-1){:}AGwrt(X)\\
&\lambda p_{AG}{:}Ax_{AGrp}(X,+,0,-1)\\
&\lambda d{:}P_{DISTR}\\
&((X,+,0,-1,*,1), and(p_M,p_{AG},d))
\end{aligned}
$$

which is of type

$$
\begin{aligned}
&\Pi X{:}Type_0\\
&\Pi m{:}Mwrt(X)\Pi g{:}AGwrt(X)\Pi d{:}P_{DISTR}.\ Ring
\end{aligned}
$$

This example shows how the sharing style of Pebble described by Lampson and Burstall in [Bur84][LB88] is supported in **ECC** and used in particular to guarantee that the carriers of the two arguments are required to be the same. Note that $Mwrt$ and $AGwrt$ are a sort of 'parameterized modules'. This sort of facility of supporting higher-order modules is very useful. □

5.3 Existential types

Existential types, also called *weak sums*, which are used to describe abstract data types [MP85] can be defined in **ECC**. Besides the existential quantifier at the proposition level [MP85][Rey83][Pra65], we can also define existential types at the type levels. For example, we can define the ith level existential-type constructor as follows:

$$
\begin{aligned}
\exists^i \quad \equiv \quad & \lambda A{:}Type_i\lambda B{:}A\to Type_i.\\
& \Pi R{:}Type_i(\Pi x{:}A.(B(x)\to R))\to R
\end{aligned}
$$

which is of type $\Pi A{:}Type_i((A \to Type_i) \to Type_{i+1})$. Then, introduction and elimination operators $rep_{\exists^i x:A.B}$ and $abstype^i$ similar to those described in [MP85][Rey83] can be defined for each level which satisfy the desired properties such as

$$
\begin{aligned}
& abstype^i\ x\ with\ y{:}B\ is\ rep_{\exists^i x:A.B}(a,b)\ in\ N\\
\rhd_\beta\quad & [b/y][a/x]N
\end{aligned}
$$

Note that, different from the existential quantifier at the proposition level, these 'weak sums' are defined at the *predicative* levels. This seems to show that, for expressing abstract data types, the impredicativity is not important. Of course, we do not have these data types as values in the strong sense of [MP85]; *e.g.*, $\exists^0 x{:}A.B$ is of type $Type_1$ but *not* of type $Type_0$.

However, the weak sums are not satisfactory tools to express mathematical theories in proof development as its elimination operator is too weak. Particularly, there is no way to prove that the first component of a 'weak pair' of type $\exists^i x{:}A.B$ satisfies the property B. A comparison of strong and weak sums in the context of modular programming can be found in [Mac86].

6 Related work

The calculus of constructions (CC for short) was studied in [Coq85][CH88][CH85] etc., whose meta theory was studied in [Coq85][Coq86b] and [Pot87]. The idea of extending constructions by an infinite type hierarchy appeared in [Coq86a], where the Generalized Calculus of Constructions (GCC for short) is presented. The strong normalization result in this paper is the first attempt to prove SN of a system which extends CC by an infinite type hierarchy, which also applies to GCC [Luo88b].

The type-checking problem for GCC is considered in [HaP88]; because GCC does not have the property of type unicity, the resulted algorithm is rather complicated.

In [HyP87] a general approach to categorical semantics of constructions-like calculi is described, where an extension of constructions with Σ-types and unit type is presented with a motivation for discussing semantics. [Ehr88] also gives a rather general framework of categorical semantics for dependent types, in which a sketch of how to interpret the calculus of constructions in the ω–**Set** framework is given. A full description of an ω–**Set** model for constructions (with Σ-types) can be found in [Luo88a,c].

Σ-types are well-known in Martin-Löf's type theory [ML73,84]. A similar idea of using Σ-types to express modular structures occurs in researches of programming languages (*e.g.*, [BLam84] and [Mac86]). For programming language research, one does not need to consider logical consistency problem as we do.

7 Conclusion and Further Research

The Extended Calculus of Constructions **ECC** is presented and studied, which we believe is a very strong and promising calculus to formalize mathematical problems and to be a basis of structured proof development. We discuss briefly several related topics for further researches besides those already mentioned above.

By Curry-Howard principle of formulas-as-types, there is an embedded logic in **ECC**. We conjecture that this logic is a conservative extension of the intuitionistic higher-order logic HOL (*c.f.*, [Chu40][Tak75]) with respect to some reasonable interpretation. This is also relevant to the problem of adequate formalization of abstract mathematics discussed at the end of section 4. The connection is concerned with the following question: what is a proper way of interpreting the object set Obj in HOL? Our guess is that it should be a proper type instead of a proposition; otherwise, we conjecture, the interpretation would not be conservative (even unsound?) with the intuition that too much *computational* power is provided at the impredicative level.

The proof-theoretic power of the calculus is unknown. The model construction given in this paper uses large set universes to interpret the type hierarchy. But it seems that it may be possible to give a *small* model of **ECC**.

The approach to theory abstraction adopted in this paper (section 5) may be called '*theories as types*', particularly, as Σ-types. Another approach to theory structuring in proof development [SB83][BLuo88] borrows the idea from researches in algebraic specification languages like Clear [BG80]. This later approach may be called '*theories as values*', as there are *theory operations* to 'put theories together' in structured theory development. Although the type hierarchy in **ECC** enables us to view Σ-types (hence, theories) as a sort of values, it seems not flexible enough to have internal powerful theory operations to structure large theories from smaller ones. This seems to be a general unavoidable weakness of type systems which are necessary to be restrictive to be logically consistent. However, it might be interesting to combine the ideas of the above two approaches in such a way that the idea of 'theories as values' can be implemented at the meta level of a proof development system based on type theories.

R.Pollack in Edinburgh has developed an interactive proof development system LEGO for Constructions and extended it to incorporate Σ-types and type hierarchy. Further experience with the system should lead to a powerful proof development environment.

Acknowledgements I am grateful to S.Hayashi who kindly helped me check a draft of the SN proof and pointed out that the notion of inaccessible cardinal can be used to interpret type hierarchy in our joint effort to consider models of hierarchy. Thanks to E.Moggi and Th.Coquand for their insights and suggesting that I consider set universes to interpret the hierarchy. Thanks to R.Harper, G.Huet, R.Pollack and P.Taylor for many helpful discussions. Finally, special thanks to my supervisor R.Burstall, whose ideas on structured theories and sharing originate my motivations, for his very helpful and continuous guidance in the work.

References

[BG80] R.Burstall and J.Goguen, *'The Semantics of CLEAR, a Specification Language'*, LNCS 86.

[BLam84] R.Burstall and B.Lampson, *'Pebble, a Kernel Language for Modules and Abstract Data Types'*, LNCS 173.

[BLuo88] R.Burstall and Zhaohui Luo, *'A Set-theoretic Setting for Structuring Theories in Proof Development'*, Circulated notes. Apr. 1988.

[Bur84] R.Burstall, *'Programming with Modules as Typed Functional Programming'*, Proc. Inter. Conf. on Fifth Generation Computer Systems, Tokyo.

[CH85] Th.Coquand and G.Huet, *'Constructions:a Higher Order Proof System for Mechanizing Mathematics'*, EUROCAL 85.

[CH88] Th.Coquand and G.Huet, *'The Calculus of Constructions'*, Information and Computation, 2/3, vol. 76.

[Chu40] A. Church, *'A Formulation of the Simple Theory of Types'*, J. Symbolic Logic 5 (1).

[Coq85] Th.Coquand, *'Une Theorie des Constructions'*, PhD thesis, University of Paris VII.

[Coq86a] Th.Coquand, *'An Analysis of Girard's Paradox'*, LICS'86.

[Coq86b] Th.Coquand, *'A Calculus of Constructions'*. Nov. 1986.

[Ehr88] T. Ehrhard, *'A Categorical Semantics of Constructions'*, LICS'88.

[Gir72] J.-Y.Girard, *Interpretation fonctionelle et elimination des coupures de l'arithmetique d'ordre superieur*, These, Universite Paris VII.

[Gir86] J.-Y.Girard, *'The System F of Variable Types, Fifteen Years Later'*, Theoretical Computer Science 45.

[HaP88] R. Harper and R. Pollack, *'Type Checking, Universe Polymorphism, and Typical Ambiguity in the Calculus of Constructions'*, Draft. Sept. 1988.

[Hyl87] M.Hyland, *'A Small Complete Category',* Ann. Pure Appl. Logic. to appear.

[HyP87] M.Hyland and A.Pitts, *'The Theory of Constructions: Categorical Semantics and Topos-theoretic Models',* Categories in Computer Science and Logic, Boulder.

[Klo80] J.W.Klop, *Combinatory Reduction Systems,* Mathematical Center Tracts 127.

[LB88] B.Lampson and R.Burstall, *'Pebble, a Kernel Language for Modules and Abstract Data Types',* Information and Computation, 2/3, vol. 76.

[LM88] G.Longo and E.Moggi, *Constructive Natural Deduction and Its 'Modest' Interpretation,* Report CMU-CS-88-131, Computer Science Dept., Carnegie Mellon Univ.

[Luo88a] Zhaohui Luo, *A Higher-order Calculus and Theory Abstraction,* LFCS report ECS-LFCS-88-57, Dept. of Computer Science, Univ. of Edinburgh.

[Luo88b] Zhaohui Luo, $CC^{\infty}_{\subseteq}$ *and Its Meta Theory,* LFCS report ECS-LFCS-88-58, Dept. of Computer Science, Univ. of Edinburgh.

[Luo88c] Zhaohui Luo, *'A Higher-order Calculus and Its* ω–**Set** *Model',* circulated notes. Jan. 1988.

[Mac86] D.MacQueen, *'Using Dependent Types to Express Modular Structure',* 13th POPL.

[ML71] Per Martin-Löf, *A Theory of Types,* Report 71-3, Dept. of Math., Univ. of Stockholm.

[ML73] Per Martin-Löf, *'An Intuitionistic Theory of Types: Predicative Part',* in Logic Colloquium'73, (eds.) H.Rose and J.C.Shepherdson.

[ML84] Per Martin-Löf, *Intuitionistic Type Theory,* Bibliopolis.

[MP85] J.Mitchell and G.Plotkin, *'Abstract Types Have Existential Type',* 12th POPL.

[Mog85] E.Moggi, *'The PER-model as Internal Category with All Small Products',* manuscript.

[Pit87] A.Pitts, *'Polymorphism is Set Theoretic, Constructively',* Summer Conf. on Category Theory and Computer Science, Edinburgh.

[Pot87] G.Pottinger, *Strong Normalization for Terms of the Theory of Constructions,* TR 11-7, Odyssey Research Associates.

[Pra65] D. Prawitz, *Natural Deduction, a Proof-Theoretic Study,* Almqvist & Wiksell.

[PS86] G.Pottinger and J.Seldin, *'Normalization for the Theory of Constructions (Extended Abstract)',* Dec. 1986.

[Rey74] J.C.Reynolds, *'Towards a Theory of Type Structure',* LNCS 19.

[Rey83] J.C.Reynolds, *'Types, Abstraction and Parameter Polymorphism',* Information Processing 83.

[Rey84] J.C.Reynolds, *'Polymorphism is Not Set-theoretic',* LNCS 173.

[RP88] J.C.Reynolds and G.D.Plotkin, *On Functors Expressible in the Polymorphic Typed Lambda Calculus,* LFCS report, ECS-LFCS-88-53, Dept. of Computer Science, Univ. of Edinburgh.

[SB83] D.Sannella and R.Burstall, *'Structured Theories in LCF',* 8th Colloquium on Trees in Algebra and Programming.

[Tak75] G. Takeuti, *Proof Theory,* Stud. Logic 81.

[TL88] P.Taylor and Zhaohui Luo, *'Theories, Mathematical Structures and Strong Sums',* Preliminary notes. Dec. 1988.

[Tro73] A.S.Troelstra, *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis,* Lecture Notes in Mathematics, 344.

[vD80] D.T. van Daalen, *The Language Theory of Automath,* PhD Thesis. Technological Univ., Eindhoven.