Technical Report

UCAM-CL-TR-208
ISSN 1476-2986

Number 208

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Categorical combinators for the calculus of constructions

Eike Ritter

October 1990

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Categorical Combinators for the Calculus of Constructions

Eike Ritter
University of Cambridge
Computer Laboratory
Corn Exchange Street
Cambridge CB2 3QG
United Kingdom

October 4, 1990

Abstract

This report describes the derivation of a small and intuitive set of categorical combinators for the Calculus of Constructions. The choice of an appropriate categorical semantics is the crucial step. A modification of Ehrhard's higher-order closed summable fibrations, yielding so-called CC-categories, turns out to be the appropriate underlying categorical structure. Standard techniques can then be used to derive the combinators. The combinators can be turned directly into the classifying category for the Calculus of Constructions. This establishes a precise connection between the calculus, the combinators and the CC-categories.

Contents

1 Introduction

Categorical combinators are an equational presentation of a categorical structure. They provide a variable-free presentation of type theories, therefore eliminating the need to take care of the names of bound variables and α-conversion. The general framework is set by Burroni [Bur81] who shows how adjunctions and pullbacks can be characterized by an equational theory. Curien [CCM87] applies this framework to obtain categorical combinators for the simply typed λ-calculus, using cartesian closed categories. The combinators are the basis for the construction of a categorical abstract machine that reduces them to their normal form. Coquand and Ehrhard [CE87] go one step further and derive categorical combinators for the polymorphic λ-calculus (PLC), using the PLC-categories [See87]. In my Diplomarbeit [Rit89] I construct a categorical abstract machine for the PLC using these categorical combinators. As a first step towards the development of a categorical abstract machine for the Calculus of Constructions (CC for short), this report describes categorical combinators for the CC.

The Calculus of Constructions, due to Coquand and Huet [CH88], adds dependent types to the polymorphic λ-calculus. This yields a powerful type theory with dependent products and an impredicative universal quantification over propositions. The main application of the CC in computer science is the construction of theorem provers that allow the formulation of properties of programs and a proof that a program actually satisfies these properties. Examples of such theorem provers are LEGO at Edinburgh [Tay89] and CAML at INRIA [INR89].

The most important step in the derivation of categorical combinators for the CC is the choice of a suitable categorical semantics. A modification of Ehrhard's higher-order closed summable fibrations [Ehr88a] is proposed in this report as the adequate categorical structure. The modification consists in replacing the fibrations by indexed categories and Grothendieck's construction and requiring the strictness of every categorical concept involved. A small and very intuitive set of combinators can then be directly deduced from the standard equational presentation of an indexed category and adjunctions. Other categorical semantics yield either more complex combinators because of the intricate representation of pullbacks or no combinators at all.

Section 2 presents the version of the CC used in this report and explains why the use of de Bruijn-variables is adequate for the categorical semantics. The category-theoretic structure, the so-called CC-category, is defined in section 3, together with a discussion of the criteria for choosing this categorical semantics. The categorical combinators are derived in section 4 and it is shown that they can be converted directly to the classifying category for the CC, although they are not strictly equivalent to the CC in the type-theoretic sense. The last section contains conclusions and ideas for further research.

2 The Calculus of Constructions

This section presents the version of the Calculus of Constructions used in this report and explains the objectives for choosing this version.

2.1 The syntax of the calculus

The version of the Calculus of Constructions used here consists of the common concepts presented in all versions [CH88] [HP89] [Str89] [Luo88] [Ehr88a]. The expressions of the calculus are divided into two levels, which are called terms and types. Following the terminology of [CH88], we have a special type Prop, which should be thought of as the type of all propositions in the higher order logic corresponding to the Calculus of Constructions according to the propositions-as-types schema. The principal components of our version are as follows:

- dependent types
- dependent products, which are predicative quantifications over arbitrary types
- allquantifications, which are impredicative quantifications over propositions
- an explicit conversion operation concerning the transformation of a proposition p into a type, which should be regarded intuitively as the set of proofs of p.

The definition of well-formedness requires additionally contexts, i.e. lists of typing statements for variables. We add also a unit type and contextmorphisms, which are lists of terms and characterize the transition from one context to another. Both constructions are included because they are part of the categorical structure used for the interpretation of the Calculus of Constructions. So there are four kinds of raw expressions, which are defined as follows:

Definition 1 (Raw expressions) *The set of raw types E, of raw terms t, of raw contexts Γ and of raw contextmorphisms f are defined by the following BNF-expressions, where x denotes an element of an infinite set of variables:*

$$\begin{array}{rcl} \Gamma & ::= & [] \mid (\Gamma, x:E) \\ f & ::= & \{\} \mid \{f,t\} \\ E & ::= & 1 \mid \Pi x:E.E \mid \mathsf{Prop} \mid \mathsf{Proof}(t) \mid \\ t & ::= & () \mid x \mid \lambda x:E.t \mid \mathsf{App}(E,E,t,t) \mid \forall x:E.t \end{array}$$

We identify terms which are equivalent under α-conversion (the binding operations being Π, λ and $\forall$. Moreover, it is assumed that the variable x in $(\Gamma, x:E)$ is distinct from all the variables occurring in Γ. The length $|\Gamma|$ of a context Γ is defined inductively as follows:

$$\begin{array}{rcl} |[]| & = & 0 \\ |(\Gamma, x:E)| & = & |\Gamma| + 1 \end{array}$$

The length of a contextmorphism is defined similarly:

$$\begin{aligned} |\{\}| &= 0 \\ |\{f,t\}| &= |f|+1 \end{aligned}$$

In the second step, the following kinds of judgments are used in the type theory:

$\vdash \Gamma$ ctxt	Γ is a valid context
$\Gamma = \Gamma'$	Γ and Γ' are equal contexts
$\Gamma \vdash f : \Delta$	f is a contextmorphism from Γ to Δ
$\Gamma \vdash f = g : \Delta$	f and g are equal contextmorphisms from Γ to Δ
$\Gamma \vdash A$ type	A is a well-formed type in context Γ
$\Gamma \vdash A = B$	A and B are equal types in context Γ
$\Gamma \vdash t : A$	t has type A in context Γ
$\Gamma \vdash t = s : A$	t and s are equal terms in context Γ

The rules for valid judgments must be defined in one huge inductive definition because the mutual dependencies between types, terms and context do not allow to define well-formed contexts, types and terms separately. However, one can split the definition into several parts, namely the parts dealing with contexts, general rules concerning judgments, rules concerning the dependent product and the rules for propositions.

The first part of the rules is concerned with the formation of contexts:

$$\text{Empty} \qquad \frac{}{\vdash [] \text{ ctxt}}$$

$$\text{Cont} - \text{Intro} \qquad \frac{\Gamma \vdash A \text{ type}}{\vdash (\Gamma, x : A) \text{ ctxt}} \quad (x \notin \mathsf{FV}(\Gamma))$$

Next, the rules for the unit type and for variables are given:

$$\text{Unit} - \text{type} \qquad \frac{}{\Gamma \vdash 1 \text{ type}}$$

$$\text{Unit} - \text{morph} \qquad \frac{}{\Gamma \vdash () : 1}$$

$$\text{Var} \qquad \frac{\vdash (\Gamma, x : A, \Gamma') \text{ ctxt}}{(\Gamma, x : A, \Gamma') \vdash x : A}$$

The third part lists the usual rules for equality:

$$\text{Cequ}\quad \frac{\vdash (\Gamma, x:A, \Gamma')\ \text{ctxt} \qquad \Gamma \vdash A = B}{(\Gamma, x:A, \Gamma') = (\Gamma, x:B, \Gamma')}$$

$$\text{Refl}\quad \frac{\vdash \Gamma\ \text{ctxt}}{\Gamma = \Gamma} \quad \frac{\Gamma \vdash f : \Delta}{\Gamma \vdash f = f : \Delta} \quad \frac{\Gamma \vdash t : A}{\Gamma \vdash t = t : A} \quad \frac{\Gamma \vdash A\ \text{type}}{\Gamma \vdash A = A}$$

$$\text{Symm}\quad \frac{\Gamma = \Gamma'}{\Gamma' = \Gamma} \quad \frac{\Gamma \vdash f = g : \Delta}{\Gamma \vdash g = f : \Delta} \quad \frac{\Gamma \vdash t = s : A}{\Gamma \vdash s = t : A} \quad \frac{\Gamma \vdash A = B}{\Gamma \vdash B = A}$$

$$\text{Trans}\quad \frac{\Gamma = \Gamma' \qquad \Gamma' = \Gamma''}{\Gamma = \Gamma''} \qquad \frac{\Gamma \vdash f = g : \Delta \qquad \Gamma \vdash g = h : \Delta}{\Gamma \vdash f = h : \Delta}$$

$$\frac{\Gamma \vdash t = t' : A \qquad \Gamma \vdash t' = t'' : A}{\Gamma \vdash t = t'' : A} \qquad \frac{\Gamma \vdash A = B \qquad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

$$\text{Conv1}\quad \frac{\Gamma \vdash f : \Delta \qquad \vdash \Delta = \Delta'}{\Gamma \vdash f : \Delta'} \qquad \frac{\Gamma \vdash A = B \qquad \Gamma \vdash t : A}{\Gamma \vdash t : B}$$

$$\text{Conv2}\quad \frac{\Gamma \vdash f = g : \Delta \qquad \Delta = \Delta'}{\Gamma \vdash f = g : \Delta'} \qquad \frac{\Gamma \vdash A = B \qquad \Gamma \vdash t = s : A}{\Gamma \vdash t = s : B}$$

The fourth part defines the rules for the dependent product:

$$\Pi - \text{form}\quad \frac{(\Gamma, x:A) \vdash B\ \text{type}}{\Gamma \vdash \Pi x : A.B\ \text{type}}$$

$$\Pi - \text{Equ}\quad \frac{\Gamma \vdash A = A' \qquad (\Gamma, x:A) \vdash B = B'}{\Gamma \vdash \Pi x : A.B = \Pi x : A'.B'}$$

$$\Pi - \text{Intro}\quad \frac{(\Gamma, x:A) \vdash t : B}{\Gamma \vdash (\lambda x : A.t) : \Pi x : A.B}$$

$$\xi - \text{rule}\quad \frac{(\Gamma, x:A) \vdash t = t' : B \qquad \Gamma \vdash A = A'}{\Gamma \vdash \lambda x : A.t = \lambda x : A'.t' : \Pi x : A.B}$$

$$\Pi - \text{elim}\quad \frac{\Gamma \vdash t : \Pi x : A.B \qquad \Gamma \vdash s : A \qquad (\Gamma, x:A) \vdash B\ \text{type}}{\Gamma \vdash \mathsf{App}(A, B, t, s) : B[x \leftarrow s]}$$

$$\Pi - \text{elimequ}\quad \frac{\Gamma \vdash A = A' \qquad \Gamma \vdash t = t' : \Pi x : A.B \qquad (\Gamma, x:A) \vdash B = B' \qquad \Gamma \vdash s = s' : A'}{\Gamma \vdash \mathsf{App}(A, B, t, s) = \mathsf{App}(A', B', t', s') : B'[x \leftarrow s]}$$

$$\beta - \text{rule}\quad \frac{(\Gamma, x:A) \vdash t : B \qquad \Gamma \vdash s : A \qquad (\Gamma, x:A) \vdash B\ \text{type}}{\Gamma \vdash \mathsf{App}(A, B, \lambda x : A.t, s) = t[x \leftarrow s] : B[x \leftarrow s]}$$

$$\eta - \text{rule}\quad \frac{\Gamma \vdash t : \Pi x : A.B \qquad (\Gamma, x:A) \vdash B\ \text{type}}{\Gamma \vdash \lambda x : A.\mathsf{App}(A, B, t, x) = t : \Pi x : A.B}\ (x \text{ not free in } t)$$

The last part gives the rules for propositions:

$$\mathsf{Prop} \qquad \frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash \mathsf{Prop} \text{ type}}$$

$$\mathsf{Proof} \qquad \frac{\Gamma \vdash p : \mathsf{Prop}}{\Gamma \vdash \mathsf{Proof}(p) \text{ type}}$$

$$\mathsf{Prop} - \mathrm{equ} \qquad \frac{\Gamma \vdash p = p' : \mathsf{Prop}}{\Gamma \vdash \mathsf{Proof}(p) = \mathsf{Proof}(p')}$$

$$\forall - \mathrm{Intro} \qquad \frac{(\Gamma, x : A) \vdash p : \mathsf{Prop}}{\Gamma \vdash \forall x : A.p : \mathsf{Prop}}$$

$$\forall - \mathrm{equ} \qquad \frac{(\Gamma, x : A) \vdash p = p' : \mathsf{Prop} \quad \Gamma \vdash A = A'}{\Gamma \vdash \forall x : A.p = \forall x : A'.p' : \mathsf{Prop}}$$

$$\forall - \mathrm{elim} \qquad \frac{(\Gamma, x : A) \vdash p : \mathsf{Prop}}{\Gamma \vdash \mathsf{Proof}(\forall x : A.p) = \Pi x : A.\mathsf{Proof}(p)}$$

This completes the definition of the Calculus of Constructions.

Remark The properties of weakening and substitution are given by the following judgments, in which $\Gamma \Rightarrow J$ is an abbreviation for either $\vdash \Gamma$ ctxt, $\Gamma = \Gamma'$, $\Gamma \vdash A$ type, $\Gamma \vdash A = B$, $\Gamma \vdash t : A$ or $\Gamma \vdash A = s$, $\Gamma \vdash f : \Delta$, $\Gamma \vdash f = g : \Delta$.

$$\mathrm{Thin} \qquad \frac{(\Gamma, \Gamma') \Rightarrow J \quad \Gamma \vdash A \text{ type}}{(\Gamma, x : A, \Gamma') \Rightarrow J} \qquad (x \notin \mathsf{FV}(\Gamma) \cup \mathsf{FV}(\Gamma'))$$

$$\mathrm{Sub} \qquad \frac{(\Gamma, x : A, \Gamma') \Rightarrow J \quad \Gamma \vdash t : A}{(\Gamma, \Gamma'[x \leftarrow t]) \vdash J[x \leftarrow t]} \qquad (x \text{ free for } t \text{ in } \Gamma')$$

$$\mathrm{Sub} - \mathrm{Equ} \qquad \frac{\Gamma \vdash t = t' : A \quad (\Gamma, x : A, \Gamma') \Rightarrow J}{(\Gamma, \Gamma'[x \leftarrow t]) \vdash J[x \leftarrow t] = J[x \leftarrow t']} \qquad (x \text{ free for } t \text{ in } \Gamma')$$

The judgments can be derived from the other ones by an induction over the derivations which is omitted here.

The objective of this version of the Calculus of Constructions is to allow a neat correspondence between the syntax and the categorical combinators. In section 4 it is shown that dependent products and allquantification are modelled quite differently in the categorical combinators, because the first concept is a predicate quantification, whereas the second is an impredicative one. In order to achieve this neat correspondence, a distinction between these two kinds of quantifications is also introduced on the syntactic level. This distinction is not only necessary on the level of judgments, but also on the level of raw terms, because the correspondence between the calculus and the categorical combinators is based on the translation of raw terms, as is demonstrated in section 4. Therefore, this version differs from the versions of [CH88] and [Luo88], which identify these two notions on raw terms and distinguish them only at the level of judgments. The same line of thought applies

also to the explicit conversion operation between propositions and the type of their proofs, which is not present in [CH88] and [HP89]. The translation of raw terms into categorical combinators requires such an operation. Furthermore, the calculus has not only the β-rule, but also the η-rule, because the equivalents of both rules in the categorical combinators are required for establishing the correspondence of the categorical combinators to the CC-category.

2.2 de Bruijn numbers

As in the case of the simply typed λ-calculus [Cur86] and in the second-order λ-calculus [CE87] the relation between the syntax and the categorical combinators is easier to establish if variables are replaced by de Bruijn numbers. This applies also to the Calculus of Constructions, as section 4 shows. At the beginning of this subsection we define the raw expressions of the Calculus of Constructions in de Bruijn form and give a translation of the raw expressions of the ordinary calculus into this one. Afterwards we give an adapted version of the rules concerning well-formedness.

The raw expressions are as follows:

Definition 2 (Raw types and terms, raw contexts) *The set of raw types E, the set of raw terms t and the set of raw contexts Γ of the Calculus of Constructions in de Bruijn form are defined by the following BNF-expressions:*

$$\begin{array}{rcl} \Gamma & ::= & [] \mid (\Gamma, E) \\ f & ::= & \{\} \mid \{f, t\} \\ E & ::= & 1 \mid \Pi E.E \mid \mathsf{Prop} \mid \mathsf{Proof}(t) \\ t & ::= & \langle\rangle \mid n \mid \lambda E.t \mid \mathsf{App}(E, E, t, t) \mid \forall E.t \end{array}$$

Furthermore, the length $|\Gamma|$ of a context Γ is defined inductively as follows:

$$\begin{array}{rcl} |[]| & = & 0 \\ |(\Gamma, E)| & = & |\Gamma| + 1 \end{array}$$

The length of a contextmorphism is defined similarly:

$$\begin{array}{rcl} |\{\}| & = & 0 \\ |\{f, t\}| & = & |f| + 1 \end{array}$$

The translation of raw expressions of the calculus with variables in the calculus with de Bruijn numbers is given next. It is only possible if raw expressions in raw contexts are considered, because a variable x is translated to the number k indicating the position of the variable in the context $\Gamma = (x_{n-1} : A_{n-1}, \ldots, x_0 : A_0)$. The definition is as follows:

Definition 3 *The* translation $\mathsf{db}(\Gamma, E)$ *of a raw expression E with variables to a raw expression in de Bruijn form with respect to a context $\Gamma = (x_{n-1} : A_{n-1}, \ldots, x_0 : A_0)$*

is defined by induction over the structure of the expression as follows:

$$\begin{array}{rcll}
\mathsf{db}([]) & = & [] & ([]) \\
\mathsf{db}((\Gamma, x:E)) & = & (\mathsf{db}(\Gamma), \mathsf{db}(\Gamma, x:E)) & () \\
\mathsf{db}(\Gamma, \{\}) & = & \{\} & (\{\}) \\
\mathsf{db}(\Gamma, \{f,t\}) & = & \{\mathsf{db}(\Gamma, f), \mathsf{db}(\Gamma, t)\} & (\{-,-\}) \\
\mathsf{db}(\Gamma, 1) & = & 1 & (1) \\
\mathsf{db}(\Gamma, \Pi x:A.B) & = & \Pi \mathsf{db}(\Gamma, A).\mathsf{db}((\Gamma, x:A), B)) & (\Pi) \\
\mathsf{db}(\Gamma, \mathsf{Prop}) & = & \mathsf{Prop} & (\mathsf{Prop}) \\
\mathsf{db}(\Gamma, \mathsf{Proof}(t)) & = & \mathsf{Proof}(\mathsf{db}(\Gamma, t)) & (\mathsf{Proof}) \\
\mathsf{db}(\langle\rangle) & = & \langle\rangle & (\langle\rangle) \\
\mathsf{db}(\Gamma, x_k) & = & k\ (0 \leq k \leq n-1) & (\mathrm{Var}) \\
\mathsf{db}(\Gamma, \lambda x:A.t) & = & \lambda \mathsf{db}(\Gamma, A).\mathsf{db}((\Gamma, x:A), t) & (\lambda) \\
\mathsf{db}(\Gamma, \mathsf{App}(A,B,s,t)) & = & \mathsf{App}(\mathsf{db}(\Gamma, A), \mathsf{db}((\Gamma, x:A), B), \mathsf{db}(\Gamma, s), \mathsf{db}(\Gamma, t)) & (\mathsf{App}) \\
\mathsf{db}(\Gamma, \forall x:A.t) & = & \forall \mathsf{db}((\Gamma, A).\mathsf{db}(\Gamma, x:A), t) & (\forall)
\end{array}$$

The operations of weakening and substitution in de Bruijn form, which are to be defined next, are more complex than those in the previous case and require extra definitions, which must be given before the well-formed expressions can be defined. The intended meaning of the weakening operation U_i^m is that if for types A and $A_0, \ldots, A_{m-1}$ and contexts Γ and Γ' with $|\Gamma'| = i$

$$\begin{array}{rcl}
(\Gamma, \Gamma') & \vdash & A \text{ type} \\
(\Gamma, A_{m-1}, \ldots, A_{j+1}) & \vdash & A_j \text{ type } (0 \leq j \leq m-1)
\end{array}$$

are valid judgments, then the judgment

$$(\Gamma, A_{m-1}, \ldots, A_0, \mathsf{U}_i^m(\Gamma')) \vdash \mathsf{U}_i^m(A)$$

is valid. A similar claim applies for a term t and a contextmorphism f instead of a type A.

Definition 4 (Weakening) *The weakening of pure types, terms and contexts is given by the operation U_i^m, which is defined as follows:*

(i) on contexts:

$$\begin{array}{rcll}
\mathsf{U}_i^m([]) & = & [] & \\
\mathsf{U}_i^m((\Gamma, E)) & = & (\mathsf{U}_{i-1}^m(\Gamma), \mathsf{U}_i^m(E)) & (i \geq 0) \\
\mathsf{U}_i^m(\Gamma) & = & \Gamma & (i < 0)
\end{array}$$

(ii) on contextmorphisms

$$\mathsf{U}_i^m(\{f_{k-1}, \ldots, f_0\}) = \{\mathsf{U}_i^m(f_{k-1}), \ldots, \mathsf{U}_i^m(f_0)\}$$

(iii) on types:

$$\begin{array}{rcl}
\mathsf{U}_i^m(1) & = & 1 \\
\mathsf{U}_i^m(\Pi A.B) & = & \Pi \mathsf{U}_i^m(A).\mathsf{U}_{i+1}^m(B) \\
\mathsf{U}_i^m(\mathsf{Prop}) & = & \mathsf{Prop} \\
\mathsf{U}_i^m(\mathsf{Proof}(t)) & = & \mathsf{Proof}(\mathsf{U}_i^m(t))
\end{array}$$

(iv) on terms:

$$\begin{aligned}
\mathsf{U}_i^m(\langle\rangle) &= \langle\rangle \\
\mathsf{U}_i^m(k) &= \begin{cases} k & k < i \\ k+m & k \geq i \end{cases} \\
\mathsf{U}_i^m(\lambda E.t) &= \lambda \mathsf{U}_i^m(E).\mathsf{U}_{i+1}^m(t) \\
\mathsf{U}_i^m(\mathsf{App}(A,B,t,s)) &= \mathsf{App}(\mathsf{U}_i^m(A), \mathsf{U}_{i+1}^m(B), \mathsf{U}_i^m(t), \mathsf{U}_i^m(s)) \\
\mathsf{U}_i^m(\forall A.t) &= \forall \mathsf{U}_i^m(E).\mathsf{U}_{i+1}^m(t)
\end{aligned}$$

In the sequel we will abbreviate $\mathsf{U}_0^1(e)$ *by* $(e)\uparrow$.

The second definition concerns the substitution of a term s for the nth variable in a term t, a type A or a context Γ,which is denoted by $t[n \leftarrow s]$, $A[n \leftarrow s]$ and $\Gamma[n \leftarrow s]$ respectively. The precise definition is as follows:

Definition 5 (Substitution)

(i) in contexts

$$\begin{aligned}
[][n \leftarrow s] &= [] \\
(\Gamma, E)[n \leftarrow s] &= (\Gamma[n-1 \leftarrow t], E[n \leftarrow s]) \quad (n \geq 0) \\
\Gamma[n \leftarrow s] &= \Gamma \quad (n < 0)
\end{aligned}$$

(ii) in contextmorphisms

$$\{f_{k-1}, \ldots, f_0\}[n \leftarrow s] = \{f_{k-1}[n \leftarrow s], \ldots, f_0[n \leftarrow s]\}$$

(iii) in types

$$\begin{aligned}
(1)[n \leftarrow s] &= 1 \\
(\Pi A.B)[n \leftarrow s] &= \Pi A[n \leftarrow s].B[n+1 \leftarrow s] \\
\mathsf{Prop}[n \leftarrow s] &= \mathsf{Prop} \\
\mathsf{Proof}(t)[n \leftarrow s] &= \mathsf{Proof}(t[n \leftarrow s])
\end{aligned}$$

(iv) in terms

$$\begin{aligned}
()[n \leftarrow s] &= () \\
k[n \leftarrow s] &= \begin{cases} k & k < n \\ \mathsf{U}_0^n(s) & k = n \\ k-1 & k > n \end{cases} \\
(\lambda A.t)[n \leftarrow s] &= \lambda A[n \leftarrow s].t[n+1 \leftarrow s] \\
\mathsf{App}(A,B,s_1,s_2)[n \leftarrow s] &= \mathsf{App}(A[n \leftarrow s], B[n+1 \leftarrow s], s_1[n \leftarrow s], s_2[n \leftarrow s]) \\
(\forall A.t)[n \leftarrow s] &= \forall A[n \leftarrow s].t[n+1 \leftarrow s]
\end{aligned}$$

The adaptation of the rules describing well-formedness is given here only for the rules involving weakening and substitution, because all other rules are merely rewritten. However, the complete set of rules can be found in the appendix. The adapted rules look as follows:

2. Variable rule

$$\text{Var}\quad \frac{\vdash (\Gamma, A, \Gamma') \text{ ctxt}}{(\Gamma, A, \Gamma') \vdash |\Gamma'| : \mathsf{U}_0^{|\Gamma'|+1}(A)}$$

4. Rules for dependent product

$$\beta - \text{rule}\quad \frac{(\Gamma, A) \vdash t : B \qquad \Gamma \vdash s : A \qquad (\Gamma, A) \vdash B \text{ type}}{\Gamma \vdash (\lambda A.t)s = t[0 \leftarrow s] : B[0 \leftarrow s]}$$

$$\eta - \text{rule}\quad \frac{\Gamma \vdash t : \Pi A.B) \qquad (\Gamma, A) \vdash B \text{ type}}{\Gamma \vdash \lambda A.(\mathsf{U}_0^1(t)0) = t : \Pi A.B}$$

The rules for weakening and substitution are as follows:

$$\text{Thin}\quad \frac{(\Gamma, \Gamma') \Rightarrow J \qquad \Gamma \vdash A \text{ type}}{(\Gamma, A, \mathsf{U}^1_{|\Gamma'|}(\Gamma')) \Rightarrow \mathsf{U}^1_{|\Gamma'|+1}(J)}$$

$$\text{Sub}\quad \frac{(\Gamma, A, \Gamma') \Rightarrow J \qquad \Gamma \vdash t : A}{(\Gamma, \Gamma'[|\Gamma'| \leftarrow t]) \vdash J[|\Gamma'| \leftarrow t]}$$

$$\text{Sub} - \text{Equ}\quad \frac{(\Gamma, A, \Gamma') \Rightarrow J \qquad \Gamma \vdash t = t' : A}{(\Gamma, \Gamma'[|\Gamma'| \leftarrow t]) \vdash J[|\Gamma'| \leftarrow t] = J[|\Gamma'| \leftarrow t']}$$

3 The CC-category

This section deals with the category-theoretic structures, the so-called CC-categories, that are the basis for the categorical combinators. The definition of a CC-category and a discussion of the connections to the syntax of the previous section are given in the first part. The section is continued with a discussion of the objectives of the definition of a CC-category. The last part contains the proof of the equivalence between Ehrhard's higher-order closed summable fibration and the CC-categories.

3.1 Definition and connection to the syntax

This subsection presents the category-theoretic structures that are used for the interpretation of the several syntactic structures of the Calculus of Constructions, namely dependent types, dependent products and propositions.

For the first syntactic concept, namely dependent types, we consider the following category-theoretic structure:

- A category $\mathcal{B}$ with a terminal object $[\,]$.
- An indexed category $E : \mathcal{B}^{op} \rightarrow \mathbf{Cat}$, where E is a functor (and not only a pseudofunctor) and each fibre $E(\Gamma)$ has a terminal object 1 that is preserved on the nose by every functor $f^* := E(f)$.

The connection to the syntax is as follows:

Category	Syntax
object Γ of the base category $\mathcal{B}$	Context Γ
terminal object $[\,]$ in the base	empty context
morphism in the base	contextmorphism
object A in the fibre over Γ	term of type A in context Γ
"pulling back" of a morphism μ in $E(\Gamma)$ along $f : \Gamma \to \Gamma'$, i.e. $f^*(\mu)$	substitution of f in μ (see below)

The formation of new contexts is captured by the following adjunction, where $Gr(E)$ denotes the Grothendieck's construction of E:

- The functor $I' : \mathcal{B} \to Gr(E)$, given by

$$\begin{aligned} I'(\Gamma) &= (\Gamma, 1) \\ I'(f) &= (f, \mathsf{Id}) \end{aligned}$$

has a right adjoint $G' : Gr(E) \to \mathcal{B}$. The object $G'(\Gamma, A)$ is abbreviated $\Gamma \times A$ in the sequel. Furthermore $(\mathsf{Fst}, \mathsf{Snd}) : (\Gamma \times A, 1) \to (\Gamma, A)$ denotes the counit of this adjunction. The natural isomorphism between $\mathrm{Hom}_{Gr(E)}((-,1) \to (-,A))$ and $\mathrm{Hom}_{\mathcal{B}}(-, - \times A)$ is denoted by $\langle -, - \rangle$.

The functor G' models the formation of new contexts from old ones, i.e. $\Gamma \times A$ corresponds to the context (Γ, A). This adjunction is also used for the modelling of substitution. If μ is a term of type A in context Γ and ρ is a term of type B in context (Γ, A), then the morphism $\langle \mathsf{Id}, \mu \rangle^* \rho : 1 \to \langle \mathsf{Id}, \mu \rangle^* B$ in $E(\Gamma)$ corresponds to the substitution of the variable with de Bruijn-number 0 in the term ρ by μ. The weakening operation, which is also necessary for the definition of substitution (cf. section 2), is also modelled by the adjunction $I' \vdash G'$. More precisely, it corresponds to the functor $\mathsf{Fst}^*_A : E(\Gamma) \to E(\Gamma \times A)$.

The dependent products are modelled by the right adjoint $\Pi_{A'}$ to the weakening functor $\mathsf{Fst}^*_{A'}$ plus Beck-Chevalley condition, i.e. the requirement that

$$\begin{aligned} f^*(\Pi_{A'}(B')) &= \Pi_{f^*(A')}((f \times \mathsf{Id})^*(B')) \\ f^*(\mathsf{Cur}_{A'}(\mu)) &= \mathsf{Cur}_{f^*(A')}((f \times \mathsf{Id})^*(\mu)) \end{aligned}$$

holds for every $f : \Gamma \to \Gamma'$, $A' \in E(\Gamma')$, $B' \in E(\Gamma' \times A')$. In these equations, Cur denotes the natural isomorphism between $\mathrm{Hom}_{E(\Gamma \times A)}(\mathsf{Fst}^*_A(B), C)$ and $\mathrm{Hom}_{E(\Gamma)}(B, \Pi_A(C))$. See next subsection for the reasons why the strict version of the Beck-Chevalley-condition is used. This modelling of dependent products is similar to that used for allquantification in the polymorphic λ-calculus and the λ-abstraction in the simply typed λ-calculus, which are modelled as right adjoints to the weakening functor in the PLC-category [See87] and cartesian closed categories [LS85] respectively.

Now we turn to propositions. There are three concepts to be considered: Firstly, the type Prop of all propositions, then the type of all proofs of a given proposition

and at last the allquantification. Propositions are modelled via a special object Ω in the CC-category, i.e. we require

- There exists an object Ω in $E([\,])$

Ω corresponds to the type of all propositions that are well-formed in the empty context. The type Prop in any context Γ is then represented by the object $\langle\rangle^*(\Omega)$, where $\langle\rangle$ denotes the unique morphism from Γ to the terminal object $[\,]$.

For the modelling of the second concept, i.e. the type of all proofs of a proposition, an extra operation converting a morphism of type Prop into an object in the appropriate fibre is necessary, because there is in general no morphism in a CC-category that is also an object in any fibre. We consider at first the generic case, i.e. the case of a single variable of type Prop. This case is captured by the following additional property of a CC-category:

- There exists an object T in $E([\,] \times \Omega)$

The type of all proofs of this proposition corresponds just to T. In general, the type of all proofs of a proposition μ corresponds to the object obtained by substituting μ in T, i.e. the object $\langle\langle\rangle, \mu\rangle^* T$.

The third concept, i.e. the impredicative allquantification, applies only to terms of type Prop and yields a new proposition. Therefore it is not a special case of a dependent product which is a predicative quantification. Hence we need an extra operation in the CC-category to model it, i.e. we require the following:

- For every morphism $\mu : 1 \to \langle\rangle^*(\Omega)$ in $E(\Gamma \times A)$ there exists a morphism $\forall\mu : 1 \to \langle\rangle^*(\Omega)$ in $E(\Gamma)$ and the naturality condition

$$\forall((h \times \mathsf{Id}) * \mu) = h * \forall(\mu)$$

  holds for every $h : \Gamma' \to \Gamma$.

At last we have to look at the proof of an allquantified proposition $\forall\mu$. The coherence condition in the Calculus of Constructions tells us that a proof of such a proposition is a function assigning to every term of type A a proof of μ. Hence this condition relates predicative and impredicative quantification. Its categorical version

$$\Pi(A, \langle\langle\rangle, \mu\rangle^*(T)) = \langle\langle\rangle, \forall(\mu)\rangle^*(T)$$

is the last requirement for a CC-category.

If we put all parts of the definition together, we obtain the following definition of a CC-category:

Definition 6 *Let $\mathcal{B}$ be a category with a terminal object $[\,]$ and $E : \mathcal{B}^{op} \to \mathbf{Cat}$ an indexed category over $\mathcal{B}$. E is a* CC-category *if it satisfies*

(i) For every object Γ of $\mathcal{B}$ there exists a terminal object 1 in the category $E(\Gamma)$, which is preserved on the nose by every functor $E(f)$. We write f^* for $E(f)$ in the sequel.

(ii) a right adjoint G' to the functor $I' : \mathcal{B} \to Gr(E)$, where I' is defined by

$$\begin{aligned} I'(\Gamma) &= (\Gamma, 1) \quad (\Gamma \in \mathrm{Obj}(\mathcal{B})) \\ I'(f) &= (f, \mathsf{Id}) \quad (f \in \mathrm{Hom}(\Gamma, \Gamma')), \end{aligned}$$

and $Gr(E)$ is the category obtained by applying the Grothendieck's construction to E. We abbreviate $G'(\Gamma, A)$ to $\Gamma \times A$ and $G'(f, \mu)$ to $f \times \mu$ later on. Furthermore $(\mathsf{Fst}, \mathsf{Snd}) : (\Gamma \times A, 1) \to (\Gamma, A)$ denotes the counit of this adjunction.

(iii) For every object Γ of $\mathcal{B}$ and A of $E(\Gamma)$, the functor

$$\mathsf{Fst}_A^* : E(\Gamma) \to E(\Gamma \times A)$$

has a right adjoint

$$\Pi_A : E(\Gamma \times A) \to E(\Gamma).$$

The natural isomorphism between $\mathrm{Hom}_{E(\Gamma \times A)}(\mathsf{Fst}_A^*(B), C)$ and $\mathrm{Hom}_{E(\Gamma)}(B, \Pi_A(C))$ is denoted by Cur.

(iv) The Beck-Chevalley-condition for the adjunctions $\mathsf{Fst}_A^* \vdash \Pi_A$ is satisfied in the strict sense, i.e. the equations

$$\begin{aligned} f^*(\Pi_{A'}(B')) &= \Pi_{f^*(A')}((f \times \mathsf{Id})^*(B')) \\ f^*(\mathsf{Cur}_{A'}(\mu)) &= \mathsf{Cur}_{f^*(A')}((f \times \mathsf{Id})^*(\mu)) \end{aligned}$$

hold for every $f : \Gamma \to \Gamma'$, $A' \in E(\Gamma')$, $B' \in E(\Gamma' \times A')$.

(v) There exists an object Ω in the category $E([\,])$ and an object T in the category $E([\,] \times \Omega)$.

(vi) For every object Γ in $\mathcal{B}$, every object A in $E(\Gamma)$ and every morphism $\mu : 1 \to \Omega$ in $E(\Gamma \times A)$ there exists a morphism $\forall(\mu) : 1 \to \Omega$ in $E(\Gamma)$ such that the equations

$$\forall((h \times \mathsf{Id})^*(\mu)) = h^*(\forall(\mu)) \tag{1}$$

$$\Pi_A(\langle\langle\rangle, \mu\rangle^*(T)) = \langle\langle\rangle, \forall(\mu)\rangle^*(T) \tag{2}$$

are satisfied.

Remark The notation for the adjunction $I' \vdash G'$ is for three reasons similar to the notation for products in cartesian closed categories, which correspond to a special case of dependent types, namely multi-sorted algebraic theories. Firstly, a context in the case of algebraic theories is simply a list of types $(A_1, \ldots, A_n)$ without any restriction on the types because there are no dependencies between types. This list is represented by the cartesian product in the cartesian category. Therefore, the product notation is chosen for the contexts in the Calculus of Constructions. Secondly, substitution of a term for a variable is modelled by composing with the morphism $\langle \mathsf{Id}, \mu \rangle$ in the cartesian closed category and by applying the functor $\langle \mathsf{Id}, \mu \rangle^*$ in the

CC-category. Thirdly, the counit of the adjunction $I' \vdash G'$, a morphism which is denoted by $(\mathsf{Fst}, \mathsf{Snd}) : (\Gamma \times A, 1) \to (\Gamma, A)$, plays a similar role as the projection (π_1, π_2), which is the counit of the adjunction defining cartesian products. Both first projections are used for modelling weakening, more precisely the functor corresponding to weakening is Fst^* in the case of the Calculus of construction and $\mathrm{Hom}(\pi_1, -)$ in the case of algebraic theories. Also, the de Bruijn-variable k is represented by $\mathsf{Fst}^k * \mathsf{Snd}$, where Fst^k is an abbreviation for $\underbrace{\mathsf{Fst} \circ \mathsf{Fst} \circ \cdots \circ \mathsf{Fst}}_{k-\text{times}}$, and by $\pi_2 \circ \pi_1^k$ respectively. However, the second projection shows the limits of this analogy. In the cartesian category, it is a morphism from $\Gamma \times A$, i.e. from the the extended context to the second component, whereas in the CC-category it is a morphism $\mathsf{Snd} : 1 \to A$ in the fibre $E(\Gamma \times A)$ from the terminal object to the second component in the fibre over the extended context. The latter implies that it is not enough to define the fibre over Γ just to be the base category itself if a cartesian category should be turned into a special case of a CC-category. The reason is that there is no right adjoint to the functor I' in this case.

3.2 Objectives of the definition of a CC-category

The CC-category is used in the next section for the derivation of categorical combinators. This yields two principles for the definition of a CC-category:

- Turn all canonical isomorphisms into identities whenever possible
- Use those constructions that allow the simplest relation to the syntax

The application of the first principle makes extra combinators and equations for the canonical isomorphisms superfluous, whereas the second one allows the relation between syntax, CC-category and categorical combinators to be as neat as possible.

The definition of a CC-category is a reformulation of Ehrhard's higher-order closed summable fibration [Ehr88a]. These are defined using fibrations, whereas the definition of a CC-category uses indexed categories and the Grothendieck's construction instead. This has several advantages:

- all canonical isomorphisms that are involved in the definition of fibrations become identities
- Grothendieck's construction has an intuitive meaning ((Γ, A) denotes the type A in context Γ), which is hidden in the fibration
- no pullbacks are necessary for the definition of a CC-category. They are implicitly part of the definition of a fibration (cf. [Ehr88a, Prop. 2]).

Also the strict version of the Beck-Chevalley-condition is adopted because this eliminates further canonical isomorphisms. All these advantages lead to a much simpler system of combinators.

Other approaches [HP89][Pit89] use so-called *display maps*. They consider a category $\mathcal{B}$ with finite products, where the fibre over an object Γ of $\mathcal{B}$ is given by

a collection of distinguished objects in $\mathcal{B}/\Gamma$, the display maps. For every object Γ in $\mathcal{B}$ the full subcategory $\mathcal{B}/\Gamma$ of display maps is denoted by $E(\Gamma)$. They are not used here for two reasons. Firstly, the pullbacks involved yield quite complicated combinators and secondly I do not know how to capture the distinct role of display maps as certain morphisms in the base category. Streicher's extension of contextual categories [Str89] imposes a tree structure on the objects to capture the notion of type-in-context. This leads to complex categorical combinators, whereas in the CC-category terms-in-contexts are modelled by an adjunction that allows the derivation of simple combinators in a standard way.

3.3 The equivalence of Ehrhard's definition

The definition of a higher-order summable fibration, which corresponds to a CC-category, consists according to [Ehr88a] of three steps, namely the definition of a D-category, a closed D-category and finally of a higher-order summable fibration. We give for each step first the definition, adapted to the case of a split fibration, and then the proof of the equivalence to the corresponding part to definition 6.

3.3.1 Split D-categories

Definition 7 (Split D-category) *A* D-category $(p : \mathcal{F} \to \mathcal{B}, I, G)$ *consists of*

- *two categories $\mathcal{F}$ and B and a fibration $p : \mathcal{F} \to \mathcal{B}$*
- *an adjunction $p \dashv I$, its counit being an isomorphism*
- *an adjunction $I \dashv G$*

A split D-category *is a D-category where the fibration p is split.*

The relation to definition 6 is as follows:

Lemma 1 *For a category $\mathcal{B}$ with a terminal object $[\,]$ the following are equivalent:*

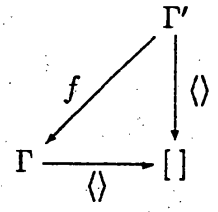
(i) $(p : \mathcal{F} \to \mathcal{B}, I, G)$ *is a split D-category*

(ii) the indexed category $E : \mathcal{B}^{op} \to$ Cat corresponding to the fibration p satisfies clauses (i) and (ii) of definition 6.

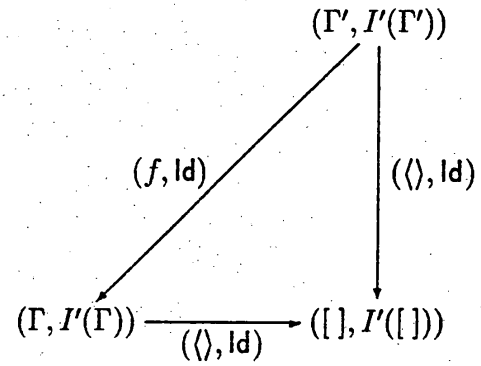
Proof We will only show that (i) implies the existence of a terminal object in each category $E(\Gamma)$ and that these terminal objects are preserved on the nose by every functor f^*.

Let $H : \mathcal{F} \to Gr(E)$ be the isomorphism of categories induced by the split fibration p. It is shown in [Ehr88a, Prop 7] that w.l.o.g. first $I(\Gamma)$ is terminal in $p^{-1}(\Gamma)$, second $pI = \mathsf{Id}$ and third the counit of the adjunction $p \dashv I$ is the identity. Therefore, we have $H(I(\Gamma)) = (\Gamma, I'(\Gamma))$, with $I'(\Gamma)$ being a terminal object in $E(\Gamma)$.

Now we show $f^*(I'(\Gamma)) = I'(\Gamma')$ for every $f : \Gamma' \to \Gamma$. The following diagram commutes



and therefore the diagram



commutes, because $(\langle\rangle, \mathsf{Id})$ is cartesian over $\langle\rangle$. Therefore $f^*(I'(\Gamma)) = I'(\Gamma')$, and we can define the object 1 in $E(\Gamma)$ to be $I'(\Gamma)$.

□

3.3.2 Closed D-categories

Before a closed D-category can be defined, some extra notation has to be introduced. The category of cartesian morphisms **Cart** is the subcategory of $Gr(E)$ consisting of all objects of $Gr(E)$ and morphisms of the form (f, Id). Let γ denote the injection functor $\mathbf{Cart} \to Gr(E)$. π_E denotes the split fibration $Gr(E) \to \mathcal{B}$. We need furthermore the categories $Gr(E \circ (\pi_E\gamma)^{op})$ and $Gr(E \circ (G\gamma)^{op})$. The objects of the first category are triples (Γ, A, B), where Γ is an object of $\mathcal{B}$ and A and B are objects of $E(\Gamma)$, and its morphisms are pairs $(f, \mu) : (\Gamma, f^*(A'), B) \to (\Gamma', A', B')$, where $(f, \mu) : (\Gamma, B) \to (\Gamma', B')$ is a morphism in $Gr(E)$. The second category is the subcategory of $Gr(E)$ of objects $(\Gamma \times A, B)$ and morphisms of the form $(f \times \mathsf{Id}, \mu) : (\Gamma \times f^*(A'), B) \to (\Gamma' \times A', B')$. The definition of a closed D-category is now as follows, using $Gr(E)$ instead of $\mathcal{F}$ for simplicity:

Definition 8 *A split D-category* $(p : \mathcal{F} \to \mathcal{B}, I, G)$ *is* closed *if the functor*

$$(\mathsf{Fst}\gamma)^* : Gr(E \circ (\pi_E\gamma)^{op}) \to Gr(E \circ (G\gamma)^{op})$$

defined by

$$(\mathsf{Fst}\gamma)^*(\Gamma, A, B) = (\Gamma \times A, \mathsf{Fst}^*(B)) \quad (3)$$

$$(\mathsf{Fst}\gamma)^*(f, \mu) = (f \times \mathsf{Id}, \mathsf{Fst}^*(\mu)) \quad (4)$$

has a fibred right adjoint.

The link to definition 6 is provided by

Lemma 2 *Let $\mathcal{B}$ a category with a terminal object $[\,]$ and let $D = (p : \mathcal{F} \to \mathcal{B}, I, G)$ be a split D-category. Then D is closed iff the conditions (iii) and (iv) of definition 6 hold.*

Proof Firstly, we spell out the condition required in the definition of the fibred adjunction $(\mathsf{Fst}\gamma)^* \dashv \Pi$. Afterwards we prove the equivalence between these requirements and the corresponding parts of definition 6.

It turns out that the definition of the fibred adjunction $(\mathsf{Fst}\gamma)^* \dashv \Pi$ amounts to requiring a functor $U : Gr(E \circ (G\gamma)^{op}) \to Gr(E \circ (\pi_E\gamma)^{op})$ satisfying the equations

$$\begin{aligned} U(\Gamma \times A, B) &= (\Gamma, A, \overline{U}(A, B)) && (5)\\ U((f \times id), \mu) &= (f, \overline{U}(\mu)) && (6)\\ U(f \times \mathsf{Id}, \mathsf{Id}) &= (f, \mathsf{Id}) && (7) \end{aligned}$$

and to having the following bijection between hom-sets, which is natural in both arguments:

$$\frac{(\Gamma \times f^*(A'), \mathsf{Fst}^*(B)) \xrightarrow{(f \times \mathsf{Id}, \mu)} (\Gamma' \times A', B')}{(\Gamma, f^*(A'), B) \xrightarrow{(f, \mathsf{Cur}(\mu))} (\Gamma', A', \overline{U}(A', B'))} \qquad (8)$$

Now we prove the equivalence of the existence of such an adjunction to clauses *(iii)* and *(iv)* of definition 6. Assume first, such an adjunction be given. Let Γ be an object of $\mathcal{B}$ and A, B be objects of $E(\Gamma)$. If we define

$$\begin{aligned} \Pi_A(B) &= \overline{U}(A, B)\\ \Pi_A(\mu) &= \overline{U}(\mu)\\ \mathsf{Cur}_A(\mu) &= \mathsf{Cur}(\mu) \end{aligned}$$

then Π_A is a functor because $\overline{U}(\mathsf{Id}) = \mathsf{Id}$ according to equation 7 and

$$(\mathsf{Id}, \Pi_A(\mu \circ \nu)) = U(\mathsf{Id}, \mu \circ \nu) = U(\mathsf{Id}, \mu) \circ U(\mathsf{Id}, \nu) = (\mathsf{Id}, \Pi_A(\mu) \circ \Pi_A(\nu))$$

The bijection 8 specializes to the following bijection

$$\frac{\mathsf{Fst}^*_A(B) \xrightarrow{\mu} B'}{B \xrightarrow{\mathsf{Cur}_A(\mu)} \Pi_A(B')}$$

which is natural in B and B'. Therefore

$$\mathsf{Fst}^*_A \dashv \Pi_A$$

Let $f : \Gamma \to \Gamma'$ be a morphism in $\mathcal{B}$ and A' be an object of $E(\Gamma')$. We now prove the Beck-Chevalley-condition. Equation 7 implies that Id is a morphism from $\overline{U}(f^*(A'), (f \times \mathsf{Id})^*(B'))$ to $f^*\overline{U}(A', B')$ in $E(\Gamma)$. Therefore

$$f^*(\Pi_{A'}(B')) = \Pi_{f^*(A')}((f \times \mathsf{Id})^*(B'))$$

The equation

$$f^*(\mathsf{Cur}_{A'}(\mu)) = \mathsf{Cur}_{f^*(A')}((f \times \mathsf{Id})^*(\mu))$$

follows from the naturality of the bijection 8.

Next we prove the other direction, i.e. that conditions (*iii*) and (*iv*) of definition 3 imply the above requirements for the fibred adjunction $(\mathsf{Fst}\gamma)^* \dashv \Pi$. Define

$$\begin{array}{rcl} U(\Gamma \times A, B) & := & (\Gamma, \Pi_A(B)) \\ U(f \times \mathsf{Id}, \mu) & := & (f, \Pi_{f^*(A')}(\mu)) \\ \mathsf{Cur}(\mu) & := & \mathsf{Cur}_A(\mu) \end{array}$$

with $(f, \mu) : (\Gamma \times f^*(A'), B) \to (\Gamma' \times A', B')$. It is easy to see that the equations 5 until 7 are satisfied. The adjunction $\mathsf{Fst}_A^* \dashv \Pi_A$ shows that 8 is a bijection. It follows furthermore that

$$(f, \mathsf{Cur}_A(\mu)) \circ (\mathsf{Id}, \nu) = (f, \mathsf{Cur}_A(\mu \circ \mathsf{Fst}^*(\nu))$$

$$(id, \Pi_A(\nu)) \circ (f, \mathsf{Cur}_A(\mu)) = (f, \mathsf{Cur}(\nu \circ \mu))$$

Using the Beck-Chevalley-condition, we get

$$(id, \mathsf{Cur}(\mu)) \circ (f, \mathsf{Id}) = (f, f^*(\mathsf{Cur}(\mu))) = (f, \mathsf{Cur}((f \times \mathsf{Id})^*(\mu)))$$

This yields the naturality of the bijection 8.

□

3.3.3 Higher-order summable fibrations

Also for the definition of a higher-order summable fibration we need some extra notation. Let Γ be any object of $\mathcal{B}$. d_Γ denotes the discrete indexed category which sends each object Γ' of $\mathcal{B}$ on $\mathrm{Hom}_{\mathcal{B}}(\Gamma', \Gamma)$. The category $Gr(d_\Gamma \circ (G\gamma)^{op})$ has then as objects pairs

$$(f : \Gamma' \times A' \to \Gamma, (\Gamma', A'))$$

where (Γ', A') is an object of $Gr(E)$ and f is a morphism in $\mathcal{B}$, and as morphisms pairs

$$(h \times \mathsf{Id}, (h, \mathsf{Id})) : (f, (\Gamma'', h^*(A'))) \to (g, (\Gamma', A'))$$

where $h \times \mathsf{Id} : \Gamma'' \times h^*(A') \to \Gamma' \times A'$ is a morphism in $\mathcal{B}$ satisfying $g \circ (h \times \mathsf{Id}) = f$. Similarly, the objects of $Gr(d_\Gamma \circ (\pi_E\gamma)^{op})$ are pairs

$$(f : \Gamma' \to \Gamma, (\Gamma', A'))$$

where (Γ', A') is an object of $Gr(E)$ and f is a morphism in $\mathcal{B}$, and its morphisms are pairs

$$(h, (h, \mathsf{Id})) : (f, (\Gamma'', h^*(A'))) \to (g, (\Gamma', A'))$$

where $h : \Gamma'' \to \Gamma'$ is a morphism in $\mathcal{B}$ satisfying $g \circ h = f$. We have then the following definition:

Definition 9 *A* higher-order summable fibration *is a split D-category* $(p : \mathcal{F} \to \mathcal{B}, I, G)$ *which is closed and satisfies*

(i) There exists an object Ω *of* $E([\,])$ *and an object* T *of* $E([\,] \times \Omega)$. *The fibration* $d_{[\,]\times\Omega}$ *is denoted* d *for short.*

(ii) There exists a functor

$$\forall : Gr(d \circ (G\gamma)^{op}) \to Gr(d \circ (\pi_E\gamma)^{op})$$

which is cartesian from the split fibration

$$\pi_{d,G} : Gr(d \circ (G\gamma)^{op}) \to \mathbf{Cart}$$

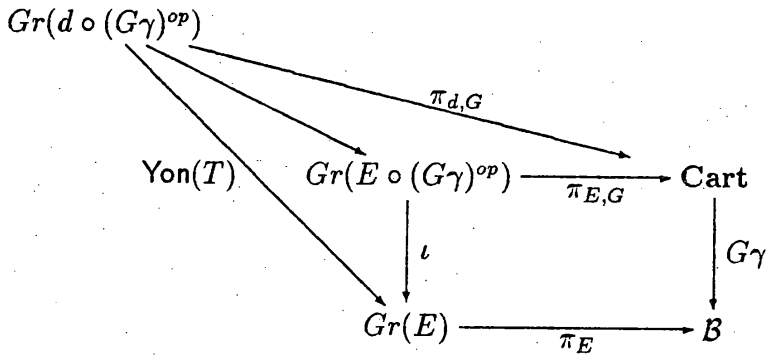
to the split fibration

$$\pi_{d,E} : Gr(d \circ (\pi_E\gamma)^{op}) \to \mathbf{Cart}$$

(iii) The following diagram of functors is commutative:

$$\begin{array}{ccc} Gr(d \circ (G\gamma)^{op}) & \xrightarrow{\mathsf{Yon}(T) \times \mathbf{Cart}} & Gr(E \circ (G\gamma)^{op}) \\ \downarrow \forall & & \downarrow \Pi \\ Gr(d \circ (\pi_E\gamma)^{op}) & \xrightarrow[\mathsf{Yon}(T) \times \mathbf{Cart}]{} & Gr(E \circ (\pi_E\gamma)^{op}) \end{array}$$

where the morphism $\mathsf{Yon}(T) \times \mathbf{Cart} : Gr(d \circ (G\gamma)^{op}) \to Gr(E \circ (G\gamma)^{op})$ *is the unique morphism making the following diagram, where* $\pi_{E,G}$ *is the split fibration* $Gr(E \circ (G\gamma)^{op}) \to \mathbf{Cart}$ *and* ι *the inclusion functor* $Gr(E \circ (G\gamma)^{op}) \to Gr(E)$,



commute, the square being a pullback and the functor $\mathsf{Yon}(T)$ *being defined by:*

$$\begin{array}{rcl} (f, (\Gamma, A)) & \overset{\mathsf{Yon}(T)}{\longmapsto} & (\Gamma \times A, f^*(T)) \\ (h, (h, \mathsf{Id})) & \overset{\mathsf{Yon}(T)}{\longmapsto} & (h \times \mathsf{Id}, \mathsf{Id}) \end{array}$$

The functor $\mathsf{Yon}(T) \times \mathbf{Cart} : Gr(d \circ (\pi_E\gamma)^{op}) \to Gr(E \circ (\pi_E\gamma)^{op})$ *is defined similarly.*

The relation to definition 6 is as follows:

Lemma 3 *Let $\mathcal{B}$ be a category with a terminal object $[\,]$ and let $D = (p : \mathcal{F} \to \mathcal{B}, I, G)$ be a split D-category which is closed. D is a CC-category iff clauses (v) and (vi) of definition 6 hold.*

Proof The equivalence between definition 9 (i) and clause (v) of definition 6 is obvious. The equivalence of definition $9(ii)$ and definition $6(vi)$ is shown as follows: The cartesianness of the functor $(\forall)$ is equivalent to

$$\forall(f,(\Gamma,A)) = (\forall'(f),(\Gamma,A)) \quad (9)$$
$$(\forall(h \times \mathsf{Id},(h,\mathsf{Id})) = (h,(h,\mathsf{Id})) \quad (10)$$

This shows that the functor $\forall$ can be defined using a function $\forall'$, which is required to satisfy

$$\forall'(f \circ (h \times \mathsf{Id})) = \forall'(f) \circ h$$

If we transpose f and $\forall'(f)$ over the adjunction $I' \dashv G'$, we get a function $\forall''$, assigning every morphism $\mu : 1 \to \Omega$ in $E(\Gamma \times A)$ a morphism $\forall''(\mu) : 1 \to \Omega$ in $E(\Gamma)$ and satisfying

$$\forall''((h \times \mathsf{Id})^*(\mu)) = h^*(\forall''(\mu))$$

A routine calculation, which is omitted here, shows that the clause (iii) of definition 9 is equivalent to

$$\Pi_A(f^*(T)) = \forall(f)^*(T).$$

Transposing this equation over the adjunction $I' \dashv G'$ yields

$$\Pi(A, \langle\langle\rangle, \mu\rangle^*(T)) = \langle\langle\rangle, \forall(\mu)\rangle^*(T)$$

This proves the remaining equivalence.

□

4 The combinators

We define in this section categorical combinators for the Calculus of Constructions and show how the calculus can be translated into the combinators. We also show the soundness of this translation and the equivalence of the CC and the equational theory of categorical combinators. In the last section it is proved that an initial CC-category can be defined using the combinators.

4.1 The equational presentation

In this subsection we define the equational theory of categorical combinators, which is a generalized algebraic theory in the sense of Cartmell [Car86]. The generalization concerns the sorts of the theory. Whereas in a normal multi-sorted algebraic theory the sorts are constants and may be interpreted as sets, they may in a generalized algebraic theory depend on a variable of a certain sort. Before giving the equational

theory of a CC-category in detail, we describe its structure. There are two kinds of statements in the theory. The first is

$$T \text{ type}$$

and indicates that T is a well-formed type. The second kind of statement

$$f \in T$$

where T is a well-formed type, indicates that f is a well-formed term. The theory contains also equations between well-formed terms of the same type. The type of the terms and the conditions under which it can be deduced are omitted if they can be derived from the terms.

We have the following rules concerning types:

$$(obj) \quad \frac{}{\text{Obj type}}$$

$$(Hom) \quad \frac{\Gamma, \Gamma' \in \text{Obj}}{\text{Hom}(\Gamma, \Gamma') \text{ type}}$$

$$(Fib) \quad \frac{\Gamma \in \text{Obj}}{\text{Fib}(\Gamma) \text{ type}}$$

$$(Fun) \quad \frac{\Gamma \in \text{Obj} \quad A, A' \in \text{Fib}(\Gamma)}{\text{Fun}_\Gamma(A, A') \text{ type}}$$

Elements $\Gamma, \Delta \cdots$ of type Obj and $f, g \cdots$ of type $\text{Hom}(\Gamma, \Gamma')$ correspond to objects and morphisms of the base category $\mathcal{B}$ respectively, whereas elements $A, B \cdots$ of type $\text{Fib}(\Gamma)$ and $\mu, \nu \cdots$ of type $\text{Fun}(\Gamma)$ denote objects and morphisms in the category $E(\Gamma)$ respectively. The following notations are used in the sequel:

$$\begin{aligned} f : \Gamma \to \Gamma' &\equiv f \in \text{Hom}(\Gamma, \Gamma') \\ \Gamma \rhd A &\equiv A \in \text{Fib}(\Gamma) \\ \Gamma \rhd \mu : A \to A' &\equiv \mu \in \text{Fun}_\Gamma(A, A') \end{aligned}$$

The signature of the equational theory is given by the following BNF-expressions:

$$\begin{aligned} \Gamma &::= [\,] \mid \Gamma \times A \\ f &::= \langle\rangle \mid \mathsf{Id} \mid f; f \mid \mathsf{Fst} \mid \langle f, \mu \rangle \\ A &::= 1 \mid f * A \mid \Pi(A, A) \mid \Omega \mid T \\ \mu &::= () \mid \mathsf{Id} \mid \mu; \mu \mid f * \mu \mid \mathsf{Snd} \mid \mathsf{Cur}(A, \mu) \mid \mathsf{App}(A, A) \mid \forall(A, \mu) \end{aligned}$$

We have the usual axioms for equality, saying that equality is an equivalence relation

and that one can substitute equals for equals in every expression. They are as follows:

$$\text{(Refl)} \qquad \Gamma = \Gamma \quad f = f \quad A = A \quad \mu = \mu$$

$$\text{(Symm)} \qquad \frac{\Gamma = \Gamma'}{\Gamma' = \Gamma} \quad \frac{f = f'}{f' = f} \quad \frac{A = A'}{A' = A} \quad \frac{\mu = \mu'}{\mu' = \mu}$$

$$\text{(Trans)} \qquad \frac{\Gamma = \Gamma' \quad \Gamma' = \Gamma''}{\Gamma = \Gamma''} \quad \frac{f = f' \quad f' = f''}{f = f''}$$

$$\frac{A = A' \quad A' = A''}{A = A''} \quad \frac{\mu = \mu' \quad \mu' = \mu''}{\mu = \mu''}$$

$$(\times) \qquad \frac{\Gamma = \Gamma' \quad A = A'}{\Gamma \times A = \Gamma' \times A'}$$

$$(*) \qquad \frac{f = f' \quad A = A'}{f * A = f * A'}$$

$$(\Pi) \qquad \frac{A = A' \quad B = B'}{\Pi(A, B) = \Pi(A', B')}$$

$$(;) \qquad \frac{f = f' \quad g = g'}{f; g = f'; g'}$$

$$(\langle -, - \rangle) \qquad \frac{f = f' \quad \mu = \mu'}{\langle f, \mu \rangle = \langle f', \mu' \rangle}$$

$$(;) \qquad \frac{\mu = \mu'; \quad \rho = \rho'}{\mu; \rho = \mu'; \rho'}$$

$$(*) \qquad \frac{f = f' \quad \mu = \mu'}{f * \mu = f' * \mu'}$$

$$\text{(Cur)} \qquad \frac{\mu = \mu' \quad A = A'}{\mathsf{Cur}(A, \mu) = \mathsf{Cur}(A', \mu')}$$

$$(\forall) \qquad \frac{A = A' \quad \mu = \mu'}{\forall(A, \mu) = \forall(A', \mu')}$$

$$(conv) \qquad \frac{f : \Gamma \to \Delta \quad A = A' \quad \Gamma = \Gamma'}{f : \Gamma' \to A}$$

$$\frac{\Gamma \rhd \mu : A \to B \quad \Gamma \rhd A = A' \quad \Gamma \rhd B = B'}{\Gamma \rhd \mu : A' \to B'}$$

Next we give the rules and equations for the indexed category $E : \mathcal{B}^{op} \to \mathbf{Cat}$.

Firstly, we say that $\mathcal{B}$ is a category with a terminal object:

$$([\,]) \qquad \frac{}{[\,] \in \mathsf{Obj}}$$

$$(\langle\rangle) \qquad \frac{\Gamma \in \mathsf{Obj}}{\langle\rangle : \Gamma \to [\,]}$$

$$(id - base) \qquad \frac{\Gamma \in \mathsf{Obj}}{\mathsf{Id} : \Gamma \to \Gamma}$$

$$(; - base) \qquad \frac{f : \Gamma \to \Gamma' \quad g : \Gamma' \to \Gamma''}{f; g : \Gamma \to \Gamma''}$$

$$\begin{array}{lrcl} (\langle\rangle) & \langle\rangle & = & \mathsf{Id} : [\,] \to [\,] \\ (\langle\rangle) & f; \langle\rangle & = & \langle\rangle \\ (idL) & f; \mathsf{Id} & = & f \\ (idR) & \mathsf{Id}; f & = & f \\ (; - assoc) & (f; g); h & = & f; (g; h) \end{array}$$

Secondly, we express that for every object Γ in the base category $\mathcal{B}$ there is a category $E(\Gamma)$:

$$(id - fib) \qquad \frac{\Gamma \rhd A}{\Gamma \rhd \mathsf{Id} : A \to A}$$

$$(; - fib) \qquad \frac{\Gamma \rhd \mu : A \to A' \quad \Gamma \rhd \nu : A' \to A''}{\Gamma \rhd \mu; \nu : A \to A''}$$

$$\begin{array}{lrcl} (idL) & \mu; \mathsf{Id} & = & \mu \\ (idR) & id; \mu & = & \mu \\ (; - assoc) & (\lambda; \mu); \nu & = & \lambda; (\mu; \nu) \end{array}$$

Thirdly, we have the following rules and equations concerning the functor $(\_)^*$, which is represented by the operator $*$ in the equational theory:

$$(* - obj) \qquad \frac{f : \Gamma \to \Gamma' \quad \Gamma' \rhd A'}{\Gamma \rhd f * A'}$$

$$(* - morph) \qquad \frac{f : \Gamma \to \Gamma' \quad \Gamma' \rhd \mu : A \to A'}{\Gamma \rhd f * \mu : f * A \to f * A'}$$

$$\begin{array}{rcl} \mathsf{Id} * A & = & A \\ \mathsf{Id} * \mu & = & \mu \\ f * \mathsf{Id} & = & \mathsf{Id} \\ f * (\mu; \nu) & = & (f * \mu); (f * \nu) \\ (f; g) * A & = & f * (g * A) \\ (f; g) * \mu & = & f * (g * \mu) \end{array}$$

After the equational theory of indexed categories we give the rules and equations for the terminal object 1 in each category $E(\Gamma)$, which is preserved on the nose by

every functor $E(f)$:

$$(1)\quad \frac{}{\Gamma \rhd 1}$$

$$()\quad \frac{\Gamma \rhd A}{\Gamma \rhd () : A \to 1}$$

$$\begin{aligned} () &= \mathsf{Id} : 1 \to 1 \\ \mu;() &= () \\ f * 1 &= 1 \end{aligned}$$

Remark The equation

$$f * () = ()$$

can be derived using the other rules and was therefore omitted above:

$$f * () = f * ();\mathsf{Id} = f * ();() = ()$$

The next part of the equational theory of CC-categories is concerned with the adjunction $I' \dashv G'$. The presentation is obtained using a general schema which is expressed in the following lemma:

Lemma 4 *Given the following data*

- *two small categories $\mathcal{C}$ and $\mathcal{D}$ and a functor $F : \mathcal{C} \to \mathcal{D}$*
- *a function G assigning to every object of $\mathcal{D}$ an object of $\mathcal{C}$*
- *a function Λ sending each morphism $f : FA \to U$ in $\mathcal{C}$ to a morphism $\Lambda(f) : A \to GU$ in $\mathcal{D}$.*
- *for every object U of $\mathcal{D}$ a morphism $\epsilon(U) : FGU \to U$*

If the equations

$$\begin{array}{lrcl} (\beta) & \epsilon(U) \circ F(\Lambda(f)) & = & f \\ (nat) & \Lambda(f) \circ g & = & \Lambda(f \circ F(g)) \\ (\eta) & \Lambda(\epsilon(U)) & = & \mathsf{Id}_{GU} \end{array}$$

hold for all objects U of $\mathcal{D}$ and all morphisms $f : FA \to U$, $g : B \to A$, then the function G can be extended to a functor $G : \mathcal{D} \to \mathcal{C}$, being a right adjoint to F. Furthermore, the bijection between the hom-sets of the adjunction is given by the function Λ, and ϵ is the counit of the adjunction.

Proof Define the effect of G on a morphism $g : U \to V$ by

$$G(g) := \Lambda(g \circ \epsilon(U))$$

and the inverse of Λ by

$$\Lambda^{-1}(h) := \epsilon(U) \circ F(h) \quad (h : A \to GU)$$

An easy calculation shows that G is a functor and that Λ and Λ^{-1} are inverse to each other and that the bijection between the hom-sets

$$\frac{f : FA \to U}{\Lambda(f) : A \to GU}$$

is natural in both A and U. Therefore $F \dashv G$, and ϵ is the counit of this adjunction.

□

This lemma is now applied to the adjunction $I' \dashv G'$ of definition 6 to get its equational presentation. We do not introduce a separate notation for the subcategory $Gr(E)$, but we use instead the corresponding expressions in terms of the indexed category E. This yields the following presentation:

$$(G' - obj) \quad \frac{\Gamma \rhd A}{\Gamma \times A \in \mathrm{Obj}}$$

$$(\langle\rangle) \quad \frac{f : \Gamma \to \Gamma' \quad \Gamma' \rhd A' \quad \Gamma \rhd \mu : 1 \to f * A'}{\langle f, \mu \rangle : \Gamma \to \Gamma' \times A'}$$

$$(\mathsf{Fst}) \quad \frac{\Gamma \rhd A}{\mathsf{Fst} : \Gamma \times A \to \Gamma}$$

$$(\mathsf{Snd}) \quad \frac{\Gamma \rhd A}{\Gamma \times A \rhd \mathsf{Snd} : 1 \to \mathsf{Fst} * A}$$

$$\begin{array}{lrcl} (\mathsf{Fst}) & \langle f, \mu \rangle ; \mathsf{Fst} & = & f \\ (\mathsf{Snd}) & \langle f, \mu \rangle * \mathsf{Snd} & = & \mu \\ (nat) & f ; \langle g, \mu \rangle & = & \langle f ; g, f * \mu \rangle \\ (\eta) & \langle \mathsf{Fst}, \mathsf{Snd} \rangle & = & \mathsf{Id} \end{array}$$

Remark The correspondence between the rules and equations given above and the general setting as described in Lemma 4 is as follows:

- $G'(\Gamma, A)$ corresponds to $\Gamma \times A$.
- The bijection Λ of the hom-sets corresponds to the operator $\langle\rangle$.
- The counit ϵ, which is a morphism $(\Gamma \times A, 1) \to (\Gamma, A)$, is represented by the morphisms Fst and Snd.
- The equation (β) corresponds to the equations Fst and Snd, because the equation (β) looks in this situation as follows:

$$(\mathsf{Fst}, \mathsf{Snd}) \circ (\langle f, \mu \rangle, \mathsf{Id}) = (f, \mathsf{Id})$$

which is equivalent to

$$(\mathsf{Fst} \circ \langle f, \mu \rangle, \langle f, \mu \rangle^*(\mathsf{Snd})) = \langle f, \mu \rangle$$

- The equations *nat* and (η) correspond directly to the same equations in lemma 4.

We use later on the abbreviation $f \times \mu$ also for the term $\langle \mathsf{Fst}; f, \mathsf{Snd}; \mathsf{Fst} * \mu \rangle$ of the equational theory.

The next part of the equational theory concerns the adjunctions $\mathsf{Fst}_A^* \dashv \Pi_A$ of definition 6. Lemma 4 yields the following equations and rules:

$$(\Pi) \quad \frac{\Gamma \rhd A \quad \Gamma \times A \rhd B}{\Gamma \rhd \Pi(A,B)}$$

$$(\mathsf{Cur}) \quad \frac{\Gamma \rhd A \quad \Gamma \times A \rhd \mu : \mathsf{Fst} * C \to B}{\Gamma \rhd \mathsf{Cur}(A,\mu) : C \to \Pi(A,B)}$$

$$(\mathsf{App}) \quad \frac{\Gamma \rhd A \quad \Gamma \times A \rhd B}{\Gamma \times A \rhd \mathsf{App}(A,B) : \mathsf{Fst} * \Pi(A,B) \to B}$$

$$\begin{array}{lrcl}
(\beta) & (\mathsf{Fst} * \mathsf{Cur}(A,\mu)); \mathsf{App}(A,B) & = & \mu \\
(nat) & \nu; \mathsf{Cur}(A,\mu) & = & \mathsf{Cur}(A, (\mathsf{Fst} * \nu); \mu) \\
(\eta) & \mathsf{Cur}(A, \mathsf{App}(A,B)) & = & \mathsf{Id}
\end{array}$$

The Beck-Chevalley condition yields directly the following equations:

$$\begin{array}{lrcl}
(\mathsf{Cur}) & f * \mathsf{Cur}(A,\mu) & = & \mathsf{Cur}(f * A, (f \times \mathsf{Id}) * \mu) \\
(\Pi) & f * \Pi(A,B) & = & \Pi(f * A, (f \times \mathsf{Id}) * B)
\end{array}$$

Finally, the clauses (v) and (vi) of definition 6 lead directly to the following additional rules and equations:

$$(\Omega) \quad \frac{}{[\,] \rhd \Omega}$$

$$(\forall) \quad \frac{\Gamma \times A \rhd \mu : 1 \to \langle\rangle * \Omega}{\Gamma \rhd \forall(A,\mu) : 1 \to \langle\rangle * \Omega}$$

$$(T) \quad \frac{}{[\,] \times \Omega \rhd T}$$

$$\begin{array}{lrcl}
(\forall - fun) & \forall(h * A, (h \times \mathsf{Id}) * \mu) & = & h * \forall(A,\mu) \\
(Coh) & \Pi(A, \langle\langle\rangle, \mu\rangle * T) & = & \langle\langle\rangle, \forall(A,\mu)\rangle * T
\end{array}$$

This completes the equational theory of a CC-category.

Remark Because the categorical combinators should be as close as possible to the CC-category, allquantification and dependent products are modelled by different combinators, as in the CC-category (cf. section 3). This distinction is therefore also introduced in the Calculus of Constructions (cf. section 2). For the same reason, an operation converting a proposition into a type is introduced in the theory of combinators. It corresponds to the substitution of the term μ in the object T in the CC-category.

4.2 The translation of the Calculus of Constructions

This section defines an interpretation of the type theory of section 2 in the equational presentation outlined in the previous section. This interpretation is given by a translation of raw types, terms and contexts of the type theory in the corresponding categorical combinators.

Definition 10 (Translation into categorical combinators) *The translation of raw types, terms and contexts into categorical combinators is given by the following function* $[\![\_]\!]$*, where* Fst^n *is an abbreviation for* $\underbrace{\mathsf{Fst};\mathsf{Fst};\cdots;\mathsf{Fst}}_{n-\text{times}}$*:*

(i) on contexts

$$\begin{aligned}[\![[]]\!] &= [] \\ [\![(\Gamma, A)]\!] &= [\![\Gamma]\!] \times [\![A]\!]\end{aligned}$$

(ii) on contextmorphisms

$$\begin{aligned}[\![\{\}]\!] &= \langle\rangle \\ [\![\{f,t\}]\!] &= \langle [\![f]\!], [\![t]\!] \rangle\end{aligned}$$

(iii) on types

$$\begin{aligned}[\![1]\!] &= 1 \\ [\![\Pi A.B]\!] &= \Pi([\![A]\!], [\![B]\!]) \\ [\![\mathsf{Prop}]\!] &= \langle\rangle * \Omega \\ [\![\mathsf{Proof}(t)]\!] &= \langle\langle\rangle, [\![t]\!]\rangle * T\end{aligned}$$

(iv) on terms

$$\begin{aligned}[\![()]\!] &= () \\ [\![n]\!] &= \mathsf{Fst}^n * \mathsf{Snd} \\ [\![\lambda A.t]\!] &= \mathsf{Cur}([\![t]\!]) \\ [\![\mathsf{App}(A,B,t,s)]\!] &= \langle \mathsf{Id}, [\![s]\!]\rangle * (\mathsf{Fst} * [\![t]\!]; \mathsf{App}([\![A]\!], [\![B]\!])) \\ [\![\forall A.t]\!] &= \forall([\![A]\!], [\![t]\!])\end{aligned}$$

The translation shows how neat the correspondence between the Calculus of Constructions and the categorical combinators is. The raw terms of the calculus can be translated directly into raw combinators. In order to establish the soundness of the translation it is therefore enough to show that every judgment in the calculus can be made also using the combinators obtained by translating the terms. If the categorical semantics of the CC is given directly by a translation of terms and types into morphisms and objects of a CC-category, the definition is more complex [Str89].

A partial translation has to be defined, and it has to be proved that this translation is defined on well-formed terms and types and respects the judgments.

The translation explains also why de Bruijn-variables are used in the definition of the Calculus of Constructions (cf. section 2). The reason is the clause concerning variables. The categorical combinator corresponding to a variable is the projection from the context, in which the variable is declared, to its type. This is exactly captured by the de Bruijn-number and therefore it is not necessary to introduce a translation of terms-in-contexts, as it would be required if normal variables were used.

4.3 Soundness

The soundness of the above translation can now be stated and proved in a fairly standard way.

Theorem 1 (Soundness) *For every judgment in the calculus there exists a corresponding judgment in the equational theory of combinators, more precisely:*

(Contexts)	$\Gamma \vdash$ *implies* $[\![\Gamma]\!] \in \mathsf{Obj}$
(Types)	$\Gamma \vdash A$ type *implies* $[\![\Gamma]\!] \triangleright [\![A]\!]$
(Terms)	$\Gamma \vdash t : A$ *implies* $[\![\Gamma]\!] \triangleright [\![t]\!] : 1 \to [\![A]\!]$
(Eq-Contexts)	$\Gamma = \Gamma'$ *implies* $[\![\Gamma]\!] = [\![\Gamma']\!] \in \mathsf{Obj}$
(Eq-Types)	$\Gamma \vdash A = B$ *implies* $[\![\Gamma]\!] \triangleright [\![A]\!] = [\![B]\!]$
(Eq-Terms)	$\Gamma \vdash t = s : A$ *implies* $[\![\Gamma]\!] \triangleright [\![t]\!] = [\![s]\!] : 1 \to [\![A]\!]$

Proof The proof of the theorem proceeds by induction over the derivation of the judgements in the type theory and depends crucially on the fact that weakening and substitution, which are meta-operations in the type theory, are translated into certain categorical combinators in the equational theory. This is expressed in the following two lemmata. Their proofs are routine inductions over the derivation of the judgments and are therefore omitted here.

Lemma 5 *Weakening corresponds to the application of the combinator*

$$\mathsf{S}_i^m := \mathsf{Fst}^m \underbrace{\times \mathsf{Id} \times \cdots \times \mathsf{Id}}_{i\text{-times}}.$$

More precisely, if we define $\mathsf{S}_i^m(\Gamma)$ *as an abbreviation for*

$$\begin{array}{rcll} \mathsf{S}_i^m([]) & = & [] & \\ \mathsf{S}_i^m(\Gamma \times A) & = & \mathsf{S}_{i-1}^m(\Gamma) \times \mathsf{S}_i^m(A) & (i \geq 0) \\ \mathsf{S}_i^m\Gamma & = & \Gamma & (i < 0), \end{array}$$

then for a given context $(\Gamma, \Gamma') \vdash$ ctxt *with* $|\Gamma'| = i$, $[\![(\Gamma, \Gamma')]\!] \in \mathsf{Obj}$ *and types* $A_{m-1}, \ldots, A_0$ *satisfying*

$$(\Gamma, A_{m-1}, \ldots, A_{j+1}) \vdash A_j \text{ type} \quad [\![(\Gamma, A_{m-1}, \ldots, A_{j+1})]\!] \triangleright [\![A_j]\!] \quad (m-1 \geq j \geq 0)$$

we have with $\Delta := (\Gamma, A_{m-1}, \ldots, A_0, \Gamma')$

(i) $[\![\Delta]\!] = [\![\Gamma]\!] \times [\![A_{m-1}]\!] \times \cdots \times [\![A_0]\!] \times \mathsf{S}_i^m([\![\Gamma']\!])$

(ii) $(\Gamma, \Gamma') \vdash A$ type and $[\![(\Gamma, \Gamma')]\!] \rhd [\![A]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{S}_i^m * [\![A]\!] = [\![\mathsf{U}_i^m(A)]\!]$

(iii) $(\Gamma, \Gamma') \vdash f : E$ and $[\![f]\!] : [\![(\Gamma, \Gamma')]\!] \to [\![E]\!]$ implies
$[\![\mathsf{U}_i^m(f)]\!] = \mathsf{S}_i^m; [\![f]\!] : [\![\Delta]\!] \to [\![E]\!]$

(iv) $(\Gamma, \Gamma') \vdash t : A$ and $[\![(\Gamma, \Gamma')]\!] \rhd [\![t]\!] : 1 \to [\![A]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{S}_i^m * [\![t]\!] = [\![\mathsf{U}_i^m(t)]\!] : 1 \to [\![\mathsf{U}_i^m(A)]\!]$

(v) $\vdash (\Gamma, \Gamma') = E$ and $[\![(\Gamma, \Gamma')]\!] = [\![E]\!]$ implies
$[\![\Delta]\!] = \mathsf{S}_i^m([\![E]\!])$

(vi) $(\Gamma, \Gamma') \vdash f = g : E$ and $[\![f]\!] = [\![g]\!] : [\![(\Gamma, \Gamma')]\!] \to [\![E]\!]$ implies
$\mathsf{S}_i^m; [\![f]\!] = \mathsf{S}_i^m; [\![g]\!] : [\![\Delta]\!] \to [\![E]\!]$

(vii) $(\Gamma, \Gamma') \vdash A = B$ and $[\![(\Gamma, \Gamma')]\!] \rhd [\![A]\!] = [\![B]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{S}_i^m * [\![A]\!] = \mathsf{S}_i^m * [\![B]\!]$

(viii) $(\Gamma, \Gamma') \vdash s = t : A$ and $[\![(\Gamma, \Gamma')]\!] \rhd [\![\mu]\!] = [\![\nu]\!] : 1 \to [\![A]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{S}_i^m * [\![s]\!] = \mathsf{S}_i^m * [\![t]\!]$

The next lemma is concerned with substitution, which is also modelled by an operator in the equational theory.

Lemma 6 *Substitution of a term s for the de Bruijn- variable n corresponds to the application of the combinator*

$$\mathsf{Su}_n([\![s]\!]) := \langle \mathsf{Id}, [\![s]\!] \rangle \underbrace{\times \mathsf{Id} \times \cdots \times \mathsf{Id}}_{n-\text{times}}$$

More precisely, if we define $\mathsf{Su}_n([\![s]\!])(\Gamma)$ *as an abbreviation for*

$$\begin{array}{rcll} \mathsf{Su}_n([\![s]\!])[] & = & [] & \\ \mathsf{Su}_n([\![s]\!])(\Gamma \times E) & = & \mathsf{Su}_{n-1}(\Gamma) \times \mathsf{Su}_n([\![s]\!]) * E & (n \geq 0) \\ \mathsf{Su}_n([\![s]\!])(\Gamma) & = & \Gamma & (n < 0) \end{array}$$

then for a given context $(\Gamma, A, \Gamma') \vdash$ ctxt *with* $|\Gamma'| = n$, $[\![(\Gamma, A, \Gamma')]\!] \in \mathrm{Obj}$ *and a term* s *satisfying* $\Gamma \vdash s : A$ *and* $[\![\Gamma]\!] \rhd [\![s]\!] : 1 \to [\![A]\!]$, *we have with* $\Delta := (\Gamma, \Gamma'[n \leftarrow s])$

(i) $[\![\Delta]\!] = [\![\Gamma]\!] \times \mathsf{Su}_n([\![s]\!])([\![\Gamma']\!])$

(ii) $(\Gamma, A, \Gamma') \vdash f : E$ and $[\![f]\!] : [\![(\Gamma, A, \Gamma')]\!] \to [\![E]\!]$ implies
$\mathsf{Su}_n([\![s]\!]); [\![f]\!] = [\![f[n \leftarrow s]]\!] : [\![\Delta]\!] \to [\![E]\!]$

(iii) $(\Gamma, A, \Gamma') \vdash B$ type and $[\![(\Gamma, A, \Gamma')]\!] \rhd [\![B]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{Su}_n([\![s]\!]) * [\![B]\!] = [\![B[n \leftarrow s]]\!]$

(iv) $(\Gamma, A, \Gamma') \vdash t : B$ and $[\![(\Gamma, A, \Gamma')]\!] \rhd [\![t]\!] : 1 \to [\![B]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{Su}_n([\![s]\!]) * [\![t]\!] = [\![t[n \leftarrow s]]\!] : 1 \to [\![B[n \leftarrow s]]\!]$

(v) $\vdash (\Gamma, A, \Gamma') = E$ and $[\![(\Gamma, \Gamma')]\!] = [\![E]\!]$ implies
$[\![\Delta]\!] = \mathsf{Su}_n([\![s]\!])([\![E]\!])$

(vi) $(\Gamma, A, \Gamma') \vdash f = g : E$ and $[\![f]\!] = [\![g]\!] : [\![(\Gamma, A, \Gamma')]\!] \to [\![E]\!]$ implies
$\mathsf{Su}_n([\![s]\!]); [\![f]\!] = \mathsf{Su}_n([\![s]\!]); [\![g]\!] : [\![\Delta]\!] \to [\![E]\!]$

(vii) $(\Gamma, A, \Gamma') \vdash B = C$ and $[\![(\Gamma, A, \Gamma')]\!] \rhd [\![B]\!] = [\![C]\!]$ implies
$[\![\Delta]\!] \vdash \mathsf{Su}_n([\![s]\!]) * [\![B]\!] = \mathsf{Su}_n([\![s]\!]) * [\![C]\!]$

(viii) $(\Gamma, A, \Gamma') \vdash t = u : B$ and $[\![(\Gamma, A, \Gamma')]\!] \rhd [\![t]\!] = [\![u]\!] : 1 \to [\![B]\!]$ implies
$[\![\Delta]\!] \rhd \mathsf{Su}_n([\![s]\!]) * [\![t]\!] = \mathsf{Su}_n([\![s]\!]) * [\![u]\!]$

The key parts of the proof of the theorem are the steps concerning β, η and the coherence rule. These parts are considered below, whereas all other parts are omitted here for brevity. The rules in braces like {} are rules of the equational theory, whereas rules of the type theory are denoted by braces like ().

(β – rule) The induction hypothesis yields:

$$\begin{aligned} [\![\Gamma]\!] \times [\![A]\!] &\rhd [\![t]\!] : 1 \to [\![B]\!] \\ [\![\Gamma]\!] &\rhd [\![A]\!] \\ [\![\Gamma]\!] \times [\![A]\!] &\rhd [\![B]\!] \\ [\![\Gamma]\!] &\rhd [\![s]\!] : 1 \to [\![A]\!] \end{aligned}$$

Rule {Cur} yields:

$$[\![\Gamma]\!] \rhd \mathsf{Cur}([\![A]\!], [\![t]\!]) : 1 \to \Pi([\![A]\!], [\![B]\!])$$

Using rules {App}, {⟨⟩}, one obtains

$$[\![\Gamma]\!] \rhd \langle \mathsf{Id}, [\![s]\!] \rangle * (\mathsf{Fst} * \mathsf{Cur}([\![A]\!], [\![t]\!]); \mathsf{App}([\![A]\!], [\![B]\!])) : 1 \to \langle \mathsf{Id}, [\![s]\!] \rangle * [\![B]\!]$$

Using rule {β} and letting

$$u := [\![\mathsf{App}(A, B, \lambda A.t, s)]\!] = \langle \mathsf{Id}, [\![s]\!] \rangle * (\mathsf{Fst} * \mathsf{Cur}([\![A]\!], [\![t]\!]); \mathsf{App}([\![A]\!], [\![B]\!]))$$

one gets

$$[\![\Gamma]\!] \rhd u = \langle \mathsf{Id}, [\![s]\!] \rangle * [\![t]\!] : 1 \to \langle \mathsf{Id}, [\![s]\!] \rangle * [\![B]\!]$$

Lemma 6 yields now the claim.

(η – rule) The induction hypothesis yields:

$$\begin{aligned} [\![\Gamma]\!] &\rhd [\![t]\!] : 1 \to \Pi([\![A]\!], [\![B]\!]) \\ [\![\Gamma]\!] \times [\![A]\!] &\rhd [\![B]\!] \end{aligned}$$

Lemma 5 and rule {App}, {Cur} and {Fst} imply then:

$$[\![\Gamma]\!] \rhd \langle \mathsf{Id}, \mathsf{Snd}\rangle * \mathsf{Cur}([\![A]\!], \mathsf{Fst} * \mathsf{Fst} * [\![t]\!]; \mathsf{App}([\![A]\!], [\![B]\!])) : 1 \to \Pi([\![A]\!], [\![B]\!])$$

Applying the rule {Cur} and letting

$$u = [\![\lambda A.\mathsf{App}(A, B, \mathsf{U}_0^1(t), 0)]\!] = \langle \mathsf{Id}, \mathsf{Snd}\rangle * \mathsf{Cur}([\![A]\!], \mathsf{Fst} * \mathsf{Fst} * [\![t]\!]; \mathsf{App}([\![A]\!], [\![B]\!])),$$

one obtains

$$[\![\Gamma]\!] \rhd u = \mathsf{Cur}([\![A]\!], (\langle \mathsf{Id}, \mathsf{Snd}\rangle \times \mathsf{Id}) * (\mathsf{Fst} * \mathsf{Fst} * [\![t]\!]; \mathsf{App}([\![A]\!], [\![B]\!]))) : \Pi([\![A]\!], [\![B]\!])$$

A routine calculation yields then

$$[\![\Gamma]\!] \rhd u = [\![t]\!]; \mathsf{Cur}([\![A]\!], \mathsf{App}([\![A]\!], [\![B]\!])) : \Pi([\![A]\!], [\![B]\!])$$

and after an application of the $\{\eta\}$-rule

$$[\![\Gamma]\!] \rhd u = [\![t]\!] : \Pi([\![A]\!], [\![B]\!])$$

$(\forall - elim)$ The induction hypothesis yields:

$$[\![\Gamma]\!] \times [\![A]\!] \rhd [\![p]\!] : 1 \to \langle\rangle * \Omega$$

Rule $\{\forall\}$ yields then

$$[\![\Gamma]\!] \rhd \forall [\![p]\!] : 1 \to \langle\rangle * \Omega$$

Rule $\{coh\}$ shows the claim.

□

4.4 Equivalence between CC and the combinators

The equivalence between categorical combinators and the CC is based on a translation from the combinators to expressions of the CC, which is defined first. We show afterwards that it respects the judgments and establish the equivalence.

However, the translation from combinators into expressions of the calculus is not as simple as the translation in the other direction. The reason is the following. Every combinator that is obtained as a translation of a well-formed term in the CC corresponds to a global section in a fibre, i.e. satisfies $\Gamma \rhd \nu : 1 \to A$ for some Γ and A. Therefore, a combinator μ satisfying $\Gamma \rhd \mu : A \to B$ is translated as if it were the combinator

$$\Gamma \rhd \mathsf{Cur}(A, \mathsf{Snd}; \mathsf{Fst} * \mu) : 1 \to \Pi(A, \mathsf{Fst} * B)$$

This has two consequences. First, $(-)^c$ is not the exact inverse of $[\![-]\!]$ on terms. However, $([\![t]\!])^c$ yields an equivalent term t' that satisfies $\mathsf{App}(1, (B)\uparrow, t', ()) = t$, and $[\![(\mu)^c]\!]$ is isomorphic to μ if $\Gamma \rhd \mu : 1 \to C$. Second, the translation $(-)^c$

cannot be defined as a total function on raw combinators because the type A in $(\mu)^c = \lambda A.t$ cannot be deduced from μ. Therefore we have to use a translation that depends also on A. The translation of composition in the fibre and of application makes eventually the use of Streicher's methods inevitable. The definition of the function $()^c$ is therefore given by inference rules, meaning that $(-)^c$ of the conclusion is defined with the given value iff the expressions in the premises are defined and have the required properties.

Definition 11 (Inverse translation) *The partial translation function $()^c$ is defined by induction over the structure of the raw combinator as follows, where $e[0 \leftrightarrow 1]$ is an abbreviation denoting the exchange of the variables 0 and 1 in e:*

1. on contexts

$$([\,]) \quad \frac{}{([\,])^c = [\,]}$$

$$(\times) \quad \frac{(\Gamma)^c \vdash (\Gamma, A)^c \text{ type}}{(\Gamma \times A)^c = ((\Gamma)^c, (\Gamma, A)^c))}$$

2. on contextmorphisms

$$(\langle\rangle) \quad \frac{\vdash (\Gamma)^c \text{ ctxt}}{(\Gamma, \langle\rangle)^c = \{\}}$$

$$(\mathsf{Id}) \quad \frac{\vdash (\Gamma)^c \text{ ctxt}}{(\Gamma, \mathsf{Id})^c = \{|\Gamma| - 1, \ldots, 0\}}$$

$$(;) \quad \frac{\begin{array}{c}(\Gamma)^c \vdash (\Gamma, f)^c : (\Delta)^c \qquad \vdash (\Gamma)^c \text{ ctxt} \\ (\Delta)^c \vdash (\Gamma, g)^c : (E)^c \qquad (\Gamma, f)^c = \{f_{k-1}, \ldots, f_0\}\end{array}}{(\Gamma, f; g)^c = (\Delta, g)^c[i \leftarrow f_i]}$$

$$(\mathsf{Fst}) \quad \frac{\vdash (\Gamma \times A)^c \text{ ctxt}}{(\Gamma \times A, \mathsf{Fst})^c = \{|\Gamma|, \ldots, 1\}}$$

$$(\langle -, - \rangle) \quad \frac{\begin{array}{c}(\Gamma)^c \vdash (\Gamma, f)^c : (\Delta)^c \qquad (\Gamma, f)^c = \{f_{k-1}, \ldots, f_0\} \\ (\Delta)^c \vdash (\Delta, A, \mu)^c : \Pi 1.((\Delta, A)^c[i \leftarrow f_i]) \uparrow\end{array}}{(\Gamma, \{f, \mu\})^c = \{(\Gamma, f)^c, \mathsf{App}(1, ((\Gamma, A)^c[i \leftarrow f_i]) \uparrow, (\Gamma, 1, \mu)^c, 1)\}}$$

3. on types

$$(1)\quad \frac{\vdash (\Gamma)^c \text{ ctxt}}{(\Gamma,1)^c = 1}$$

$$(*)\quad \frac{(\Gamma)^c \vdash (\Gamma,f)^c : (\Delta)^c \qquad (\Delta)^c \vdash (\Gamma,A)^c \text{ type} \qquad (\Gamma,f)^c = \{f_{k-1},\ldots,f_0\}}{(\Gamma, f*A)^c = (\Delta,A)^c[i \leftarrow f_i]}$$

$$(\Pi)\quad \frac{(\Gamma)^c \vdash (\Gamma,A)^c \text{ type} \qquad (\Gamma\times A)^c \vdash (\Gamma\times A, B)^c) \text{ type}}{(\Gamma,\Pi(A,B))^c = \Pi(\Gamma,A)^c.(\Gamma\times A,B)^c}$$

$$(\Omega)\quad \frac{}{([\,],\Omega)^c = \mathsf{Prop}}$$

$$(T)\quad \frac{}{([\,]\times\Omega,T)^c = \mathsf{Proof}(0)}$$

4. on terms

$$(())\quad \frac{(\Gamma)^c \vdash (\Gamma,A)^c \text{ type}}{(\Gamma,A,())^c = \lambda(\Gamma,A)^c.()}$$

$$(\mathsf{Id})\quad \frac{(\Gamma)^c \vdash (\Gamma,A)^c \text{ type}}{(\Gamma,A,\mathsf{Id})^c = \lambda(\Gamma,A)^c.0}$$

$$(;)\quad \frac{\begin{array}{c}(\Gamma)^c \vdash (\Gamma,A,\mu)^c : \Pi(\Gamma,A)^c.((\Gamma,B)^c)\uparrow \\ (\Gamma)^c \vdash (\Gamma,B,\nu)^c : \Pi(\Gamma,B)^c.((\Gamma,C)^c)\uparrow\end{array}}{\begin{array}{rl}(\Gamma,A,\mu;\nu)^c &= \lambda(\Gamma,A)^c.\mathsf{App}(((\Gamma,B)^c)\uparrow,((\Gamma,C)^c)\uparrow,((\Gamma,B,\nu)^c)\uparrow, \\ & \quad \mathsf{App}(((\Gamma,A)^c)\uparrow,((\Gamma,B)^c)\uparrow,((\Gamma,A,\mu)^c)\uparrow,0))\end{array}}$$

$$(*)\quad \frac{\begin{array}{c}(\Gamma)^c \vdash (\Gamma,f)^c : (\Delta)^c \qquad (\Gamma,f)^c = \{f_{k-1},\ldots,f_0\} \\ (\Delta)^c \vdash (\Delta,A,\mu)^c : \Pi(\Delta,A)^c.((\Delta,B)^c)\uparrow\end{array}}{(\Gamma, f*A, f*\mu)^c = (\Delta,A,\mu)^c[i \leftarrow f_i]}$$

$$(\mathsf{Snd})\quad \frac{(\Gamma)^c \vdash (\Gamma,A)^c \text{ type}}{(\Gamma\times A,1,\mathsf{Snd})^c = \lambda 1.1}$$

$$(\mathsf{Cur})\quad \frac{\begin{array}{c}(\Gamma)^c \vdash (\Gamma,A)^c \text{ type} \\ (\Gamma\times B)^c \vdash (\Gamma\times B, \mathsf{Fst}*A,\mu)^c : \Pi((\Gamma,A)^c)\uparrow, \mathsf{U}_0^2((\Gamma,C)^c)\end{array}}{\begin{array}{l}(\Gamma,A,\mathsf{Cur}(B,\mu))^c = \lambda(\Gamma,A)^c.\lambda((\Gamma,B)^c)\uparrow. \\ [\mathsf{App}(((\Gamma,A)^c)\uparrow,\mathsf{U}_0^2((\Gamma,C)^c),((\Gamma\times B,\mathsf{Fst}*A,\mu)^c)\uparrow,0)][0\leftrightarrow 1]\end{array}}$$

$$(\mathsf{App})\quad \frac{(\Gamma)^c \vdash (\Gamma,A)^c \text{ type} \qquad (\Gamma\times A)^c \vdash (\Gamma\times A,B)^c \text{ type}}{\begin{array}{l}(\Gamma\times A, \mathsf{Fst}*\Pi(A,B),\mathsf{App}(A,B))^c = \lambda((\Gamma,\Pi(A,B))^c)\uparrow. \\ \mathsf{App}(((\Gamma,\Pi(A,B))^c)\uparrow,((\Gamma,B)^c)\uparrow,0,1)\end{array}}$$

$$(\forall)\quad \frac{(\Gamma\times B)^c \vdash (\Gamma\times B,1,\mu)^c : \Pi 1.\mathsf{Prop}}{\begin{array}{l}(\Gamma,1,\forall(B,\mu))^c = \lambda 1.\forall((\Gamma,B)^c)\uparrow. \\ [\mathsf{App}(1,\mathsf{Prop},(\Gamma\times B,1,\mu)^c,0)][0\leftrightarrow 1]\end{array}}$$

Although the definition of the translation $()^c$ is complex, the usual technique of showing the preservation of judgments by $()^c$ applies here as well, namely induction over the derivation using appropriate lemmata for weakening and substitution.

Theorem 2 *The inverse translation respects the judgments, more precisely:*

(Contexts)	$\Gamma \in \mathrm{Obj}$ *implies* $\vdash (\Gamma)^c$ ctxt
(Contextmorphisms)	$f : \Gamma \to \Delta$ *implies* $(\Gamma)^c \vdash (\Gamma, f)^c : (\Delta)^c$
(Types)	$\Gamma \rhd A$ *implies* $(\Gamma)^c \vdash (\Gamma, A)^c$ type
(Terms)	$\Gamma \rhd \mu : A \to B$ *implies* $(\Gamma)^c \vdash (\Gamma, A, \mu)^c : \Pi(\Gamma, A)^c.((\Gamma, B)^c)\uparrow$
(Eq-Contexts)	$\Gamma = \Gamma'$ *implies* $(\Gamma)^c = (\Gamma')^c$
(Eq-Contextmorphisms)	$f = f' : \Gamma \to \Delta$ *implies* $(\Gamma)^c \vdash (\Gamma, f)^c = (\Gamma, f')^c$
(Eq-Types)	$\Gamma \rhd A = A'$ *implies* $(\Gamma)^c \vdash (\Gamma, A)^c = (\Gamma, A')^c$
(Eq-Terms)	$\Gamma \rhd \mu = \mu' : A \to B$ *implies* $(\Gamma)^c \vdash (\Gamma, A, \mu)^c = (\Gamma, A, \mu')^c : \Pi(\Gamma, A)^c.((\Gamma, B)^c)\uparrow$

Proof The theorem is shown by an induction over the derivation of judgments. As in the case of the translation $[\![-]\!]$, we need two lemmata concerning weakening and substitution. Their proofs are omitted because they are a routine induction over the structure of the derivation.

Lemma 7 (Weakening) *The translation* $()^c$ *respects weakening, i.e. for every object* $\Gamma \times \Gamma' \in \mathrm{Obj}$ *satisfying* $\vdash (\Gamma \times \Gamma')^c$ ctxt, $|\Gamma'| = i$ *and types* $A_{m-1}, \ldots, A_0$ *s.t.* $\Gamma \times A_{m-1} \times \cdots \times A_{j+1} \rhd A_j$ *and* $(\Gamma \times A_{m-1} \times \cdots \times A_{j+1})^c \vdash (\Gamma \times A_{m-1} \times \cdots \times A_{j+1}, A_j)^c$ type, *we have with* $\Delta := \Gamma \times A_{m-1} \times \cdots \times A_0 \times \Gamma'$ *and* $|\Gamma'| = i$, $|\Gamma \times \Gamma'| = k$*:*

1. *on contexts*
   $\vdash (\Delta)^c = \mathsf{U}_i^m((\Gamma \times \Gamma')^c)$
2. *on contextmorphisms*
   $(\Delta)^c \vdash \mathsf{U}_i^m((\Gamma \times \Gamma', f)^c) = (\Delta, \mathsf{S}_i^m * f)^c$
3. *on types*
   $(\Delta)^c \vdash \mathsf{U}_i^m((\Gamma \times \Gamma', A)^c) = (\Delta, \mathsf{S}_i^m * A)^c$
4. *on terms*
   $(\Delta)^c \vdash \mathsf{U}_i^m((\Gamma \times \Gamma', B, \mu)^c) = (\Delta, \mathsf{S}_i^m * B, \mathsf{S}_i^m * \mu)^c$

Lemma 8 (Substitution) *The translation* $(-)^c$ *respects substitution, i.e. given an object* $\Gamma \times \Gamma' \in \mathrm{Obj}$ *such that* $|\Gamma'| = n$, $\vdash (\Gamma \times \Gamma')^c$ ctxt *and a morphism* μ *satisfying*

$$(\Gamma)^c \vdash (\Gamma, 1, \mu)^c : \Pi 1.((\Gamma, A)^c)\uparrow$$

we have with $\Delta = \Gamma \times \mathsf{Su}_n(\mu)(\Gamma')$, $s = (\Gamma, 1, \mu)^c$, $k = |\Gamma \times \Gamma'|$

(i) $\vdash (\Delta)^c = ((\Gamma)^c, (\Gamma, A)^c, (\Gamma')^c)[n \leftarrow s]$

(ii) $(\Delta)^c \vdash (\Gamma \times A \times \Gamma', f)^c[n \leftarrow s] = (\Delta, \mathsf{Su}_n(\mu); f)^c$

(iii) $(\Delta)^c \vdash (\Gamma \times A \times \Gamma', B)^c[n \leftarrow s] = (\Delta, \mathsf{Su}_n(\mu) * B)^c$

(iv) $\Delta \vdash (\Gamma \times A \times \Gamma', B, \nu)^c[n \leftarrow s] = (\Delta, \mathsf{Su}_n(\mu) * B, \mathsf{Su}_n(\mu) * \nu)^c$

The key cases of the proof of the theorem are (β), (η) and (coh). They are verified here, whereas all other cases are omitted.

$(\beta - rule)$ The premise for this rule is

$$\Gamma \times A \rhd \mu : \mathsf{Fst} * C \to B$$

Therefore, the induction hypothesis and the weakening lemma 7 imply that

$$(\Gamma)^c \vdash (\Gamma, A)^c \text{ type}$$
$$(\Gamma \times A)^c \vdash (\Gamma \times A, \mathsf{Fst} * C, \mu)^c : \Pi((\Gamma, C)^c) \uparrow .((\Gamma, B)^c) \uparrow$$

The definition of $()^c$ implies now

$$(\Gamma \times A)^c \vdash (\Gamma \times A, \mathsf{Fst} * C, \mathsf{Fst} * \mathsf{Cur}(A, \mu); \mathsf{App}(A, B))^c : \Pi((\Gamma, C)^c) \uparrow .((\Gamma, B)^c) \uparrow$$

From now on we omit quite a few typing statements and write ts for $\mathsf{App}(A, B, t, s)$ because this makes the following calculations much easier to read. The definition of $()^c$ yields now

$$\begin{aligned}
(\Gamma \times A)^c \quad \vdash \quad & (\Gamma, \mathsf{Fst} * C, \mathsf{Fst} * \mathsf{Cur}(\mu); \mathsf{App})^c = \\
= \quad & \lambda((C)^c) \uparrow .((\mathsf{App})^c) \uparrow \cdot \{((\mathsf{Fst} * \mathsf{Cur}(\mu))^c) \uparrow \cdot 0\} \\
= \quad & \lambda((C)^c) \uparrow .\lambda \mathsf{U}_0^2(\Pi(A)^c.(B)^c).(0 \cdot 2) \cdot \\
& \left\{\lambda((C)^c) \uparrow .(\lambda \mathsf{U}_0^2((A)^c).[((\mu)^c) \uparrow \cdot 0][0 \leftrightarrow 1]) \uparrow \cdot 0\right\} \\
= \quad & \lambda((C)^c) \uparrow .\lambda \mathsf{U}_0^2(\Pi(A)^c.(B)^c).(0 \cdot 2) \cdot \left\{\lambda \mathsf{U}_0^2((A)^c).[\mathsf{U}_1^2((\mu)^c) \cdot 1]\right\} \\
= \quad & \lambda((C)^c) \uparrow .((\mu)^c) \uparrow \cdot 0 \\
\overset{(\eta)}{=} \quad & (\mu)^c
\end{aligned}$$

Note the use of the η- rule in the last step although only the β- rule has been verified.

$(\eta - rule)$ The premises of the η-rule are simply

$$\Gamma \rhd A \qquad \Gamma \times A \rhd B$$

The induction hypothesis implies therefore

$$(\Gamma)^c \vdash (A)^c \text{ type} \qquad (\Gamma \times A)^c \vdash (B)^c \text{ type}$$

(using the obvious abbreviations). Rule (App) and (Cur) of definition 11 show now that

$$(\Gamma)^c \vdash (\mathsf{Cur}(\mathsf{App}))^c : \Pi(\Pi A.B)^c.((\Pi A.B)^c) \uparrow$$

The definition of $()^c$ yields now

$$\begin{aligned}(\Gamma)^c \;\vdash\; & (\mathsf{Cur}(\mathsf{App}))^c = \\ = \;& \lambda(\Pi A.B)^c.\lambda((A)^c)\uparrow.[((\mathsf{App})^c)\uparrow\cdot 0][0 \leftrightarrow 1] \\ = \;& \lambda(\Pi A.B)^c.\lambda((A)^c)\uparrow.[\lambda((\Pi A.B)^c)\uparrow.(0\cdot 2)\cdot 0][0 \leftrightarrow 1] \\ = \;& \lambda(\Pi A.B)^c.\lambda(A)^c.[1\cdot 0] \\ = \;& \lambda(\Pi A.B)^c.0\end{aligned}$$

$(coh - rule)$ The induction hypothesis applied to the premises establishes:

$$(\Gamma \times A)^c \vdash (\mu)^c : \mathsf{Prop}$$

Therefore, $(\Gamma, \langle\langle\rangle, \forall(A,\mu)\rangle * T)^c$ is defined. This implies

$$\begin{aligned}(\Gamma)^c \;\vdash\; & (\Gamma, \langle\langle\rangle, \forall(A,\mu)\rangle * T)^c = \\ = \;& \mathsf{Proof}(0)[0 \leftarrow \forall(A)^c.(\mu)^c] \\ = \;& \mathsf{Proof}(\forall(A)^c, (\mu)^c) \\ = \;& \Pi(A)^c.\mathsf{Proof}((\mu)^c) \\ = \;& (\Pi(A, \langle\langle\rangle, \mu\rangle * T))^c\end{aligned}$$

□

Finally we can prove that the translations $()^c$ and $[\![]\!]$ are inverse to each other in the sense discussed above.

Theorem 3 *The translations* $()^c$ *and* $[\![]\!]$ *are related as follows:*

(i) $([\![-]\!])^c$ *is an equivalence on expressions.*

1. *on contexts:* $([\![\Gamma]\!])^c = \Gamma$ *if* $\vdash \Gamma$ ctxt
2. *on contextmorphisms:* $\Gamma \vdash ([\![f]\!])^c = f$ *if* $\Gamma \vdash f : \Delta$
3. *on types:* $\Gamma \vdash ([\![\Gamma]\!], [\![A]\!])^c = A$ *if* $\Gamma \vdash A$ type
4. *on terms:* $\Gamma \vdash \mathsf{App}(1,(A)\uparrow, ([\![\Gamma]\!], [\![A]\!], [\![t]\!])^c, ()) = t$ *if* $\Gamma \vdash t : A$

(ii) *The map* $[\![(-)^c]\!]$ *is an isomorphism for all combinators that are translations of raw expressions in the CC, i.e.*

1. *on contexts:* $[\![(\Gamma)^c]\!] = \Gamma$ *if* $\Gamma \in \mathrm{Obj}$
2. *on contextmorphisms:* $\Gamma \rhd [\![(\Gamma, f)^c]\!] = f$ *if* $\Gamma \vdash f : \Gamma \to \Delta$
3. *on types:* $\Gamma \rhd [\![(\Gamma, A)^c]\!] = A$ *if* $\Gamma \rhd A$
4. *on terms:* $\Gamma \rhd [\![(\Gamma, A, \mu)^c]\!] = \mathsf{Cur}(A, \mathsf{Snd}; \mathsf{Fst} * \mu) : \Pi(A, \mathsf{Fst} * B)$ *if* $\Gamma \rhd \mu : A \to B$ *(isomorphic to* μ *in case* $A = 1$*)*

Proof The proof proceeds by induction over the derivation of judgments.

(i) The verification of most cases involves only routine calculations and is therefore omitted. The more interesting cases are now verified:

(*Var*) The weakening lemma yields:

$$([\![(\Gamma, A, \Gamma')]\!], 1, [\![\Gamma']\!])^c = \mathsf{U}_0^{|\Gamma'|}(\lambda 1.1) = \lambda 1.(|\Gamma'| + 1)$$

An application of the (β)-rule shows now

$$\Gamma \vdash \mathsf{App}(1, \mathsf{U}_0^{|\Gamma'|+1}(A), t, ()) = |\Gamma'|$$

$(\Pi - \mathit{Intro})$ Let u be an abbreviation for

$$\mathsf{App}(1, (B)\uparrow, ([\![\Gamma, A]\!], 1, [\![t]\!])^c, ())$$

Because of the soundness of both translations we know that $([\![\Gamma,]\!], 1, [\![\lambda A.t]\!])^c$ is defined. The definition of the translations shows now

$$\begin{array}{lll} ([\![\Gamma]\!])^c & \vdash & \mathsf{App}(1, (\Pi A.B)\uparrow, ([\![\Gamma]\!], 1, [\![\lambda A.t]\!])^c, 1) \\ & = & \lambda([\![\Gamma]\!], [\![A]\!])^c.u \\ & \overset{I.H.}{=} & \lambda A.t \end{array}$$

$(\Pi - \mathit{Elim})$ Again, the soundness of the translation shows easily that

$$([\![\Gamma]\!], 1, [\![\mathsf{App}(A, B, t, s)]\!])^c$$

is defined. The typing information in the terms is omitted in the sequel because it makes the proof much harder to read and can be added easily. Furthermore we write ts instead of $\mathsf{App}(A, B, t, s)$. The definition of the translations yields now

$$\begin{array}{lll} ([\![\Gamma]\!])^c & \vdash & ([\![\Gamma]\!], 1, [\![ts]\!])^c \cdot () = \\ & = & ([\![\Gamma]\!], 1, \langle \mathsf{Id}, [\![s]\!] \rangle * (\mathsf{Fst} * [\![t]\!]; \mathsf{App}))^c \cdot () \\ & \overset{\mathit{Lemma\ 8}}{=} & ([\![\Gamma \times A]\!], 1, \mathsf{Fst} * [\![t]\!]; \mathsf{App})^c [0 \leftarrow ([\![\Gamma]\!], 1, [\![s]\!])^c \cdot ()] \cdot () \\ & \overset{I.H}{=} & \lambda [(\lambda(0 \cdot 1))\uparrow \cdot \{\mathsf{U}_0^2(([\![\Gamma]\!], 1, [\![t]\!])^c) \cdot 0\} [0 \leftarrow s]] \cdot () \\ & = & \lambda\lambda(0 \cdot \mathsf{U}_0^2(s)) \cdot \{(([\![\Gamma]\!], 1, [\![t]\!])^c)\uparrow \cdot 0\} \cdot () \\ & = & \lambda(0 \cdot (s)\uparrow) \cdot \{([\![\Gamma]\!], 1, [\![t]\!])^c \cdot ()\} \\ & \overset{I.H.}{=} & ts \end{array}$$

(ii) Similarly we verify only the cases (;), (Cur) and (App). In all these cases the soundness of the translations shows that $[\![(\Gamma, A, \mu)^c]\!]$ is defined. We simplify the notation in the interest of readability in the same way as above. Then we have

(;)

$$\begin{array}{lll}
\Gamma & \triangleright & [\![(\Gamma, A, \mu; \nu)^c]\!] = \\
& = & [\![\lambda((\Gamma, A, \nu)^c) \uparrow \cdot \{((\Gamma, A, \mu)^c) \uparrow \cdot 0\}]\!] \\
& \stackrel{I.H.}{=} & \mathsf{Cur}(\langle \mathsf{Id}, \langle \mathsf{Id}, \mathsf{Snd}\rangle * (\mathsf{Fst} * \mathsf{Fst} * \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mu); \mathsf{App})\rangle * \\
& & (\mathsf{Fst} * \mathsf{Fst} * \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \nu); \mathsf{App})) \\
& = & \mathsf{Cur}(\langle \mathsf{Id}, \langle \mathsf{Id}, \mathsf{Snd}\rangle * (\mathsf{Fst} * \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mathsf{Fst} * \mu); \mathsf{App})\rangle * \\
& & (\mathsf{Fst} * \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mathsf{Fst} * \nu); \mathsf{App})) \\
& \stackrel{(\beta)}{=} & \mathsf{Cur}(\langle \mathsf{Id}, \mathsf{Snd}; \mathsf{Fst} * \mu\rangle * (\mathsf{Snd}; \mathsf{Fst} * \mathsf{Fst} * \nu) \\
& = & \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mu; \mathsf{Fst} * \nu) \\
& = & \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * (\mu; \nu))
\end{array}$$

(Cur)

$$\begin{array}{lll}
\Gamma & \triangleright & [\![(\Gamma, A, \mathsf{Cur}(\mu))^c]\!] = \\
& = & [\![\lambda\lambda.[\![(\Gamma \times B, \mathsf{Fst} * A, \mu)^c]\!] \cdot 0]\!][0 \leftrightarrow 1] \\
& = & \mathsf{Cur}(\mathsf{Cur}((\mathsf{Fst} \times \mathsf{Id} \times \mathsf{Id}) * \langle \mathsf{Id}, \mathsf{Snd}\rangle * (\mathsf{Fst} * \mathsf{Fst} * \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mu); \mathsf{App}))) \\
& \stackrel{(\beta)}{=} & \mathsf{Cur}(\mathsf{Cur}(\langle\langle \mathsf{Fst}; \mathsf{Fst}, \mathsf{Snd}\rangle, \mathsf{Fst} * \mathsf{Snd}\rangle * \langle \mathsf{Id}, \mathsf{Snd}\rangle * (\mathsf{Fst} * \mathsf{Snd}; \mathsf{Fst} * \mathsf{Fst} * \mu))) \\
& = & \mathsf{Cur}(\mathsf{Cur}(\mathsf{Fst} * \mathsf{Snd}; (\mathsf{Fst} \times \mathsf{Id}) * \mu)) \\
& = & \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mathsf{Cur}(\mu))
\end{array}$$

(App) We need the naturality condition

$$(nat) \quad \mathsf{App}(\mathsf{Fst} * A, (\mathsf{Fst} \times \mathsf{Id}) * B) = (\mathsf{Fst} \times \mathsf{Id}) * \mathsf{App}(A, B)$$

for the combinator App in the calculation.

$$\begin{array}{lll}
\Gamma \times A & \triangleright & [\![(\Gamma \times A, \mathsf{Fst} * \Pi(A, B), \mathsf{App})^c]\!] = [\![\lambda(0 \cdot 1)]\!] = \\
& = & \mathsf{Cur}(\langle \mathsf{Id}, \mathsf{Fst} * \mathsf{Snd}\rangle * (\mathsf{Fst} * \mathsf{Snd}; \mathsf{App})) \\
& = & \mathsf{Cur}(\mathsf{Snd}; \langle \mathsf{Id}, \mathsf{Fst} * \mathsf{Snd}\rangle * \mathsf{App}) \\
& \stackrel{(nat)}{=} & \mathsf{Cur}(\mathsf{Snd}; \langle \mathsf{Id}, \mathsf{Fst} * \mathsf{Snd}\rangle * \langle \mathsf{Fst}; \mathsf{Fst}, \mathsf{Snd}\rangle * \mathsf{App}) \\
& = & \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \langle \mathsf{Id}, \mathsf{Snd}\rangle * \mathsf{App}) \\
& \stackrel{(nat)}{=} & \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \langle \mathsf{Id}, \mathsf{Snd}\rangle * \langle \mathsf{Fst}; \mathsf{Fst}, \mathsf{Snd}\rangle * \mathsf{App}) \\
& = & \mathsf{Cur}(\mathsf{Snd}; \mathsf{Fst} * \mathsf{App})
\end{array}$$

□

This equivalence is now reformulated in categorical terms via the notion of a classifying category [Pit88]. The technical problems caused by the additional morphisms in the fibre will then disappear. The general setting is as follows. We consider a type theory T, a category **TCat** of categories with such a structure

and structure-preserving morphisms that we can define the notion of a T-model in a category $\mathcal{D} \in \mathrm{Obj}(\mathbf{TCat})$ and the notion of a homomorphism between two models of T in $\mathcal{D}$. Given any other category $\mathcal{C} \in \mathrm{Obj}(\mathbf{TCat})$, then a functor $F \in \mathsf{Hom}_{\mathbf{TCat}}(\mathcal{C}, \mathcal{D})$ sends a T-model in $\mathcal{C}$ to T-models in $\mathcal{D}$, and similarly a natural transformation between two such functors induces a homomorphism between the models. In that way we can define for every model M of T in $\mathcal{C}$ an evaluation functor $ev_M : \mathsf{Hom}_{\mathbf{TCat}}(\mathcal{C}, \mathcal{D}) \to \mathsf{Mod}(T, \mathcal{D})$. A classifying category $\mathcal{C} \in \mathrm{Obj}(\mathbf{TCat})$ is a category with the property that there exists a T-model M in $\mathcal{C}$ (called the generic model) such that the functor ev_M is an equivalence of categories. As an example, the classifying category of the simply typed λ-calculus over a set I of ground types is the free cartesian closed category over I and the generic model is the standard interpretation of the λ-calculus in a cartesian closed category. In the case of the CC, the categorical structure is the category **CCcat** of all CC-categories and structure-preserving morphisms between them. The classifying category is given in the following theorem:

Theorem 4 *Let $\mathcal{C}$ be the indexed category $E : \mathcal{B}^{op} \to \mathbf{Cat}$ defined as follows:*

- *Objects of the base category $\mathcal{B}$ are equivalence classes of combinators Γ satisfying $\Gamma \in \mathrm{Obj}$ modulo the derivable equality.*
- *Morphisms from Γ to Δ in $\mathcal{B}$ are equivalence classes of combinators f s.t. $f : \Gamma \to \Delta$ is a valid judgment in the equational theory modulo derivable equality.*
- *The identity is the identity combinator, and composition in $\mathcal{B}$ is given by the composition operator on the combinators.*
- *Objects of the fibre $E(\Gamma)$ are equivalence classes of combinators satisfying $\Gamma \rhd A$ modulo derivable equality.*
- *Morphisms from A to B in $E(\Gamma)$ are equivalence classes of combinators satisfying $\Gamma \rhd \mu : A \to B$ modulo derivable equality .*
- *Identities and composition in the fibres are given by the corresponding combinators.*
- *The functor $E(f)$, with $f : \Gamma \to \Delta$ a morphism in $\mathcal{B}$ is given by the operation on the combinators, i.e.*

$$\begin{aligned} E(f)(A) &:= f * A \\ E(f)(\mu) &:= f * \mu \end{aligned}$$

Then $\mathcal{C}$ is the classifying category of the CC, and $[\![-]\!]$ is the generic model.

Proof The proof is a reformulation of previous results, namely of the soundness and equivalence theorems in this section. The soundness theorem 1 shows that $[\![-]\!]$ is a model, and lemma 4 is the key to prove that $\mathcal{C}$ is a CC-category. Given any

functor $F \in \mathsf{Hom}_{\mathbf{CCcat}}(\mathcal{C}, \mathcal{D})$, the corresponding CC-model in $\mathcal{D}$ is given by $F([\![-]\!])$, and vice versa, given any CC-model M' in $\mathcal{D}$, the corresponding functor is given by $M'(-)^c$. The soundness theorem 2 shows that $M'(-)^c$ is well-defined and the equivalence theorem 3 shows that this defines an equivalence of categories. The key point in the last statement is that combinators like App, Id and () in the fibre that distorted the equivalence of the type theories are morphisms that have to exist because of the structure of a CC-category. Therefore, once a functor is required to preserve the structure of a CC-category, theses morphisms have to be preserved as well.

□

5 Conclusions and further work

The derivation of the categorical combinators shows how important a careful choice of the categorical semantics is. The simplicity of the combinators depends on the categorical structure having as few natural isomorphisms as possible and on the use of Grothendieck's construction instead of fibrations. Because of the equivalence of the CC-category with higher-order closed summable fibrations (see section 3.3), the ω-set model of the latter category [Ehr88b] is also a model of the former. The categorical semantics has also influenced the version of the CC used here. The CC contains a unit type corresponding to the terminal object in the fibre. This type is required for the adjunction modelling context extension, i.e. the adjunction $I' \dashv G'$. However, this unit type does not change the power of the CC.

The combinators are simple and fit neatly within the usual connections between type theory and category theory. They can be turned easily into the classifying category of the CC, and the standard translation of the CC into the combinators is the generic model. This precise correspondence is not affected by the fact that the CC and the combinators are not strictly equivalent as type theories. The non-strictness is caused by extra combinators that do not correspond to expressions in the CC, and these are only necessary for the representation of categorical structures like identities and morphisms to the terminal object in the fibre.

The combinators were developed as the first stage in the construction of a categorical abstract machine for the CC. The next stage is the derivation of reduction rules for the combinators. Ordered categories seem to be the appropriate framework, where intuitively $a \geq b$ is valid for two morphisms a and b iff a can be reduced by the categorical equivalent of β-reduction to b. It would also be nice to prove strong normalization and confluence for the combinators by categorical method and not by a transfer of the corresponding results from the CC. These properties are important for the syntax as well as for the implementation, and the construction process of a categorical machine implies that they should be proved using the combinators and not the syntax. A categorical version of logical relations, referred to as the *glueing constructions* in [CP90], might be useful as an adequate categorical method.

Acknowledgements

I would like to thank Andy Pitts for many useful discussions and suggestions. I had also fruitful discussions with other members of the CLICS project in Cambridge. Andy Pitts and Roy Crole gave helpful comments on previous versions of this report.

A The Rules in de Bruijn form

This appendix contains the complete set of the rules for well-formed types, terms and contexts of the Calculus of Constructions in de Bruijn-form.

1. Formation of contexts:

$$\text{Empty} \quad \frac{}{\vdash [] \text{ ctxt}}$$

$$\text{Cont} - \text{Intro} \quad \frac{\Gamma \vdash A \text{ type}}{\vdash (\Gamma, A) \text{ ctxt}}$$

2. Unit type and variables

$$\text{Unit} - \text{type} \quad \frac{}{\Gamma \vdash 1 \text{ type}}$$

$$\text{Unit} - \text{morph} \quad \frac{}{\Gamma \vdash () : 1}$$

$$\text{Var} \quad \frac{\vdash (\Gamma, A, \Gamma') \text{ ctxt}}{(\Gamma, A, \Gamma') \vdash |\Gamma'| : \mathsf{U}_0^{|\Gamma'|+1}(A)}$$

3. Rules for Equality:

$$\text{Cequ} \quad \frac{\vdash (\Gamma, A, \Gamma') \text{ ctxt} \quad \Gamma \vdash A = B}{(\Gamma, A, \Gamma') = (\Gamma, B, \Gamma')}$$

$$\text{Refl} \quad \frac{\vdash \Gamma \text{ ctxt}}{\Gamma = \Gamma} \quad \frac{\Gamma \vdash f = f : \Delta}{\Gamma \vdash f : \Delta} \quad \frac{\Gamma \vdash t = t : A}{\Gamma \vdash t : A} \quad \frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A = A'}$$

$$\text{Symm} \quad \frac{\Gamma = \Gamma'}{\Gamma' = \Gamma} \quad \frac{\Gamma \vdash f = g : \Delta}{\Gamma \vdash g = f : \Delta} \quad \frac{\Gamma \vdash t = s : A}{\Gamma \vdash s = t : A} \quad \frac{\Gamma \vdash A = B}{\Gamma \vdash B = A}$$

$$\text{Trans} \quad \frac{\Gamma = \Gamma' \quad \Gamma' = \Gamma''}{\Gamma = \Gamma''} \quad \frac{\Gamma \vdash f = g : \Delta \quad \Gamma \vdash g = h : \Delta}{\Gamma \vdash f = h : \Delta}$$

$$\frac{\Gamma \vdash t = t' : A \quad \Gamma \vdash t' = t'' : A}{\Gamma \vdash t = t'' : A} \quad \frac{\Gamma \vdash A = B \quad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

$$\text{Conv1} \quad \frac{\Gamma \vdash f : \Delta \quad \vdash \Delta = \Delta'}{\Gamma \vdash f : \Delta'} \quad \frac{\Gamma \vdash A = B \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}$$

$$\text{Conv2} \quad \frac{\Gamma \vdash f = g : \Delta \quad \Delta = \Delta'}{\Gamma \vdash f = g : \Delta'} \quad \frac{\Gamma \vdash A = B \quad \Gamma \vdash t = s : A}{\Gamma \vdash t = s : B}$$

4. Rules for the dependent product:

$$\Pi - \text{form} \quad \frac{(\Gamma, A) \vdash B \text{ type}}{\Gamma \vdash \Pi A.B \text{ type}}$$

$$\Pi - \text{Equ} \quad \frac{\Gamma \vdash A = A' \quad (\Gamma, A) \vdash B = B'}{\Gamma \vdash \Pi A.B = \Pi A'.B'}$$

$$\Pi - \text{Intro} \quad \frac{(\Gamma, A) \vdash t : B}{\Gamma \vdash (\lambda A.t) : \Pi A.B}$$

$$\xi - \text{rule} \quad \frac{(\Gamma, A) \vdash t = t' : B \quad \Gamma \vdash A = A'}{\Gamma \vdash \lambda A.t = \lambda A'.t' : \Pi A.B}$$

$$\Pi - \text{elim} \quad \frac{\Gamma \vdash t : \Pi A.B \quad \Gamma \vdash s : A \quad (\Gamma, A) \vdash B}{\Gamma \vdash \mathsf{App}(A, B, t, s) : B[0 \leftarrow s]}$$

$$\Pi - \text{elimequ} \quad \frac{\Gamma \vdash A = A' \quad \Gamma \vdash t = t' : \Pi A.B \quad (\Gamma, A) \vdash B = B' \quad \Gamma \vdash s = s' : A'}{\Gamma \vdash \mathsf{App}(A, B, t, s) = \mathsf{App}(A', B', t', s') : B'[0 \leftarrow s]}$$

$$\beta - \text{rule} \quad \frac{(\Gamma, A) \vdash t : B \quad \Gamma \vdash s : A \quad (\Gamma, A) \vdash B \text{ type}}{\Gamma \vdash \mathsf{App}(A, B, \lambda A.t, s) = t[0 \leftarrow s] : B[0 \leftarrow s]}$$

$$\eta - \text{rule} \quad \frac{\Gamma \vdash t : \Pi A.B \quad (\Gamma, A) \vdash B \text{ type}}{\Gamma \vdash \lambda A.\mathsf{App}(A, B, (t)\uparrow, 0) = t : \Pi A.B}$$

5. Rules for propositions:

$$\mathsf{Prop1} \quad \frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash \mathsf{Prop} \text{ type}}$$

$$\mathsf{Proof} \quad \frac{\Gamma \vdash p : \mathsf{Prop}}{\Gamma \vdash \mathsf{Proof}(p) \text{ type}}$$

$$\mathsf{Prop} - \text{equ} \quad \frac{\Gamma \vdash p = p' : \mathsf{Prop}}{\Gamma \vdash \mathsf{Proof}(p) = \mathsf{Proof}(p')}$$

$$\forall - \text{Intro} \quad \frac{(\Gamma, A) \vdash p : \mathsf{Prop}}{\Gamma \vdash \forall A.p : \mathsf{Prop}}$$

$$\forall - \text{equ} \quad \frac{(\Gamma, A) \vdash p = p' : \mathsf{Prop} \quad \Gamma \vdash A = A'}{\Gamma \vdash \forall A.p = \forall A'.p' : \mathsf{Prop}}$$

$$\forall - \text{elim} \quad \frac{(\Gamma, A) \vdash p : \mathsf{Prop}}{\Gamma \vdash \mathsf{Proof}(\forall A.p) = \Pi A.\mathsf{Proof}(p)}$$

References

[Bur81] Albert Burroni. Algèbres graphiques. *Cahiérs de Topologie et Géometrie différentielle*, 22(3):249–265, 1981.

[Car86] John Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[CCM87] Guy Cousineau, Pierre-Louis Curien, and Michel Mauny. The categorical abstract machine. *Science of Computer Programming*, 8:173–202, 1987.

[CE87] Thierry Coquand and Thomas Ehrhard. An equational presentation of higher order logic. In D. Pitt, A. Poigné, and D. Rydeheard, editors, *Category Theory and Computer Science*, pages 40–56. LNCS 283, Springer, September 1987.

[CH88] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[CP90] Roy L. Crole and Andrew M. Pitts. New foundations for fixpoint computations. In *Fifth Annual Symposium on Logic in Computer Science*, pages 489–497. IEEE, 1990.

[Cur86] Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Pitman, 1986.

[Ehr88a] Thomas Ehrhard. A categorical semantics of constructions. In *Third Annual Symposium on Logic in Computer Science*, pages 264–273. IEEE, 1988.

[Ehr88b] Thomas Ehrhard. *Une sémantique catégorique des types dépendants: Application au Calcul des Constructions.* PhD thesis, Paris VII, Paris, 1988.

[HP89] J. Martin E. Hyland and Andrew M. Pitts. The theory of constructions: Categorical semantics and topos theoretic models. *Contemporary Mathematics*, 92:137–198, 1989.

[INR89] INRIA. The calculus of constructions: Documentation and user's guide, version 4.10. Technical Report, August 1989.

[LS85] Joachim Lambek and Philip J. Scott. *Introduction to Higher-Order Categorical Logic.* Cambridge University Press, 1985.

[Luo88] Zhaohui Luo. A higher-order calculus and theory abstraction. Technical Report ECS-LFCS-88-57, LFCS, Edinburgh, July 1988.

[Pit88] Andrew M. Pitts. Lectures on Categories and Types. Lectures delivered at the Summer School on Constructive Logics and Category Theory, Isle of Thorns, August 1988.

[Pit89] Andrew M. Pitts. Categorical semantics of dependent types. Talk given at SRI, June 1989.

[Rit89] Eike Ritter. *Eine kategorielle Maschine für den polymorphen λ-Kalkül.* Diplomarbeit, Universität Erlangen, 1989.

[See87] Robert A. G. Seely. Categorical semantics for higher-order polymorphic λ-calculus. *Journal of Symbolic Logic*, 52(4):969–989, December 1987.

[Str89] Thomas Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions.* PhD thesis, Universität Passau, Passau, West Germany, June 1989.

[Tay89] Paul Taylor. Playing with LEGO: Some examples of developing mathematics in the calculus of constructions. Technical Report ECS-LFCS-89-89, LFCS, Edinburgh, 1989.