Constructions and Predicates

Duško Pavlović

Zevenwouden 223, Utrecht, The Netherlands

Abstract

In this paper, the *theory of constructions* is reinterpreted as a type theory of "sets" and "predicates". Following some set-theoretical intuitions, it is modified at two points: (1) a simple new operation is added – to represent a constructive version of the *comprehension principle*; (2) a restriction on contexts is imposed – "sets" must not depend on "proofs" of "predicates". The resulting theory is called *theory of predicates*. Sufficiently constructive arguments from naive set theory can be directly written down in it. On the other hand, modification (2) is relevant from a computational point of view, since it corresponds to a necessary condition of the modular approach to programming.

Our main result tells that, despite (2), the theory of predicates is as powerful as the theory of constructions: the constructions obstructed by (2) can be recovered in another form using (1). In fact, the theory of constructions is equivalent with a special case of the theory of predicates.

1. Introduction

The foundational role of type theory in computer science is comparable with the foundational role of set theory in mathematics. But the "set-theoretical" type theory of Russell and Church seems to have been less influential than the "logical" conception of *formulæ-as-types*, due to Curry and Howard (and traceable back to the Brouwer-Heyting-Kolmogorov interpretation of *proofs-as-constructions*). On the other hand, the experience of topos theory shows that the crucial set-theoretical notions can be given an elegant type-theoretical presentation (cf. Lambek-Scott 1986). So it seems worth-while to better explore the conceptual area in the intersection of type theory and set theory.

This paper reports on an effort to understand the *theory of constructions* (Coquand-Huet 1986, 1988, Hyland-Pitts 1989, Coquand 1990) as a strongly constructive theory of sets and propositions. With a similar idea, Ehrhard (1989) has argued that the categorical counterpart of the theory of constructions generalizes the notion of topos. Rather than semantically, we shall here approach the theory of constructions from another type theory, the *theory of predicates*.

Both these theories recognize two sorts of types, which can be understood as sets and propositions. So there are two *universes*. The universe of propositions is a type in the universe of sets; propositions appear as terms of this type. Terms in the universe of sets represent *elements*; terms in the universe of propositions are *proofs*. Viewed in this way, a family of propositions $\alpha(X)$ indexed by the elements of a set K is of course a *predicate* on K.

The theory of predicates starts from the idea that every predicate $\alpha(X)$ should be *comprehended* in the universe of sets by something like $\{X\in K/\ \alpha(X)\}$. An element of $\{X\in K/\ \alpha(X)\}$ would be a pair $\langle k,a\rangle$, where a is a proof of $\alpha(k)$. There may be many different constructive proofs of $\alpha(k)$ (i.e. many terms of this type) and the set $\{X\in K/\ \alpha(X)\}$, viewed constructively, may not be a subset of K.

Furthermore, indexing of a family of sets by proofs of a proposition will be forbidden in the theory of predicates. Philosophical justifications for this restriction (in the style: "all the elements must be created before proofs of propositions about them") become superfluous in the light of the main result of this paper, which tells that it really makes no difference – provided that predicates are comprehended among sets. We shall prove that the theory of predicates has slightly greater expressive power than the theory of constructions (although the latter theory imposes no special restrictions on indexing). In fact, the theory of constructions is equivalent (modulo a translation) to the *strict* theory of predicates, the one which satisfies a version of the ω-rule, well known from the untyped λ-calculus. Another characteristic of the strict theory of predicates is that every predicate $\alpha(X)$ in it can be recovered from (or even identified with) the set $\{X\in K/\ \alpha(X)\}$. In my thesis (1990) it was described how the theory of predicates corresponds to some small categories with small sums and products, while the theory of constructions and the strict theory of predicates correspond to those such categories which are (fully) generated by the terminal object.

And while conceptually nothing is lost by dumping the sets which depend on proofs, it seems that a lot can be gained. Some gains are technical: the imposed restriction reduces contexts to two layers (first sets, and then propositions), and many constructions – e.g. term models – become essentially simpler. But recent papers by Moggi (1990) and by Harper-Mitchell-Moggi

(1990) display this restriction as a *sine qua non* of the modular programming. Roughly speaking, Moggi understands as *programs* what I here call propositions, and my sets are for him *data types*. Clearly, a modular approach to programming can be effective only if the type-checking can be performed at compile time, before running actual programs. In other words, no type must depend on output of programs. This is called *phase distinction* between the compile-time and the run-time. Or between sets and propositions. It is amusing to think that this analogy of computational and foundational concepts is not accidental.[1]

2. Type theories

Keywords. We shall consider three kinds of *expressions*:

- *terms*, here denoted by metavariables p, q, r, s, t,
- *types*, denoted by P, Q, R, S , and
- *universes*, for which we use the letter $\mathcal{U}$.

The common name for terms and types is *constructions*; while *range* denotes a type or a universe. And now these expressions form two kinds of *judgements* (or *statements*):

- *equations*, or *conversion* judgements $T=T'$ between constructions T,T', and
- *formation* judgements $T{:}U$, meaning "the construction T has the range U".

The metavariable J will denote a judgement. The range of a construction can sometimes be indicated by a superscript: T^U.

The *variables* are special atomic terms. We use the letters X, Y, Z for them. If the terms are understood as programs, the variables are the input operations. Each term is represented by an expression $p(X_0,...X_n)$, in which the variables indicate the input gates. To supply input means to *substitute* a term $q(Y_0,...Y_m)$ for a variable X_i:

$$p(X_0,...,X_i,...,X_n)[q/X_i] := p(X_0,...,X_{i-1},q(Y_0,...Y_m),...,X_n).$$

Of course, q must have the same type as X_i. The type of a term-as-program is the type of its output data. Note that a data type may also vary, i.e. it may need some input before it is evaluated. A term must vary with its type. The universes will always remain constant – no variables can occur in them.

The actual objects of study in type theory are *sequents*

$$X_0{:}P_0,...,X_n{:}P_n \Rightarrow J \qquad (n \in \omega).$$

[1] Added in proof: Other such analogies can be found in Meseguer 1989.

An array $X_0{:}P_0,\ldots,X_n{:}P_n$ is called *context* and abbreviated by letters Γ or Δ. It can be understood as the list of declarations of the data used for constructions in J. All the variables occurring in these constructions must, of course, be declared. But a variable occurring only in the context of a construction, and not in the expression which actually names this construction, can not always be safely omitted. Intuitively, a program with some superfluous data among the declarations may change when this data is removed: if the superfluous data does not exist – if its type is empty –, a program containing it may never become executable.[2]

Sequents are derived using some *rules*, generally in the form

$$\frac{\Gamma, \Delta_0 \Rightarrow J_0 \qquad (\ldots) \qquad \Gamma, \Delta_n \Rightarrow J_n}{\Gamma \Rightarrow J}$$

The sequents above the line are *premises*, the one below is the *conclusion*. The variables from $\Delta_0,\ldots\Delta_n$ are said to be *bound* in the conclusion. Conventionally, we often omit the context Γ common to all the sequents in a rule. The rules by which the theories studied here are built up will be listed in Appendix I.

Derivations are trees built iteratively using the conclusions of some (instances of) rules as premises for other rules. This proces starts from *axioms*, which can be regarded as rules with empty set of premises.

A construction or a context is said to be *well-formed* (or *valid*, or *legal*) if it occurs in a derivable sequent. The name of a universe and the empty context are assumed to be well-formed.

A construction is *closed* when its context is not bigger than the context of its range. Thus, a closed type must have empty context (since the range of a type is a universe). A type is *inhabited* when it possesses a closed term.

If we allow not only the empty context, but also the "empty judgement", and assume the empty sequent $\Rightarrow$ (empty on both sides!) as an axiom, then we can show that a context Γ is well-formed iff the sequent $\Gamma\Rightarrow$ is derivable. (We can extend the notion of axiom to the rules with at most one premis, checking whether a context and a range are well-formed.)

In fact, the empty context and the empty judgement are a type – just as zero is a number. (This is essential for some proofs below.) In every universe $\mathcal{U}$ we shall assume a *unit* type $1{:}\,\mathcal{U}$,

[2] When is this the case and when not is a rather subtle matter: its categorical formulation leads into theory of *descent*. A forthcoming paper will explore this connection.

inhabited by a unique term $\emptyset{:}1$. The empty context and the empty judgement can now be written $\emptyset{:}1$ or $X{:}1$, which boils down to the same thing, since $X^1=\emptyset$.

Common to all type theories are also the *structural* rules, which govern manipulation with variables. The rules *Replacement* and *Typing* tell that equal constructions can replace each other: all the operations must preserve the convertibility relation (=). The rule *Assumption* tells that there is always a fresh variable of each well-formed type. Let me stress that this does *not* imply that each type must be inhabited (i.e. that data of each type must exist)!

To get an algebra from an algebraic theory, one can add some generators and equations (to the constants and equations included in the theory), and derive the well-formed expressions, which are then partitioned in the equivalence classes induced by equations. A type theory can similarly be *extended* by generators and additional equations. Generators must be given with well-formed contexts and ranges; equations may be imposed only on constructions with the same range and context. We call *system* the class of derivable formation sequents of an extended type theory; letters $\mathcal{M}$, $\mathcal{N}$ denote systems. (For convenience, we shall assume that a system also includes the names of universes.) Building a system is a dynamical process, since an atomic construction – a generator – can have a complex context and range, and can be thrown in only when they have been derived.

The convertibility relation (=) is extended from constructions to sequents in an obvous way – component-wise – modulo a renaming of variables (*α-rule*). Let us spell this out. By definition,

$$\left(X_0{:}P_0,\dots,X_m{:}P_m \Rightarrow T{:}U\right) = \left(X'_0{:}P'_0,\dots,X'_n{:}P'_n \Rightarrow T'{:}U'\right)$$

means that

- $m=n$, and
- the following sequents are derivable

$$Y_0{:}P''_0,\dots,Y_j{:}P''_j \Rightarrow P_{j+1}[\vec{Y}/\vec{X}]=P'_{j+1}[\vec{Y}/\vec{X}'],\ \text{for all } j<n;$$

$$Y_0{:}P''_0,\dots,Y_n{:}P''_n \Rightarrow U[\vec{Y}/\vec{X}]=U'[\vec{Y}/\vec{X}'];$$

$$Y_0{:}P''_0,\dots,Y_n{:}P''_n \Rightarrow T[\vec{Y}/\vec{X}]=T'[\vec{Y}/\vec{X}'];$$

where $P''_j{:}=P_j[\vec{Y}/\vec{X}]$, while $\vec{Y}:=(Y_0,\dots,Y_n)$ are fresh variables.

Partitioning a system of a type theory by the convertibility relation gives a *term model* for this theory. In the usual abuse of language, we often write T for whole sequent $\Gamma \Rightarrow T{:}U$, and even for its equivalence class; the context and range are meant to be kept implicite, and can be recovered by $CX(T)=\Gamma$ and $RG(T)=U$.[3]

The algebraic aspect of type theory is the study of the convertibility $(=) \subseteq \mathcal{M}\times\mathcal{M}$. Its proof-theoretical aspect concerns the relation of derivability $(\vdash) \subseteq \mathcal{M}^{*}\times\mathcal{M}$, transitive closure of all the instances of the given formation rules (together with the axioms and generators taken as rules), where $\mathcal{M}^{*} := \bigcup_{i\in\omega} \mathcal{M}^{i}$.

Theories of constructions and of predicates. The theory of constructions is a (Martin-Löf-style) type theory of sums and products – in two universes:

$\mathcal{S}$ – its types are called *sets*, its terms *elements* (or *functions*);

$\mathcal{P}$ – its types are *propositions*, terms are *proofs*.

Each of these universes is closed under all sums and products. Clearly, there are four possible kinds of indexing: $\mathcal{S}\Rightarrow\mathcal{S}$, $\mathcal{P}\Rightarrow\mathcal{P}$, $\mathcal{S}\Rightarrow\mathcal{P}$, $\mathcal{P}\Rightarrow\mathcal{S}$ – and four kinds of sums and products, two for each universe. The sums and products of propositions indexed over sets ($\mathcal{S}\Rightarrow\mathcal{P}$) are *quantifiers*. They will be written $\exists$ and $\forall$.

The axiom $\mathcal{P}:\mathcal{S}$ is assumed: "The universe of propositions is a set". It follows that every proposition is at the same time a type in $\mathcal{P}$ and a term in $\mathcal{S}$. So there are three levels of constructions:

$$\text{proofs } \begin{matrix} a,b,c \\ x,y,z \end{matrix} : \text{ propositions } \begin{matrix} \alpha,\beta,\gamma \\ \xi,\eta,\zeta \end{matrix} : \text{ sets } \mathcal{P}K := K\to\mathcal{P}.$$

Of course, sets which are not in the form $\mathcal{P}K$ may also be introduced. We denote by A,B,K sets in general, and their elements by f,g,k; the general element-variables remain X,Y,Z. We shall reserve $\phi:1$ for the *singleton*, unit of $\mathcal{S}$; the *truth*, unit of $\mathcal{P}$, will be denoted by $*:\mathcal{T}$.

The intended meaning of the operation of *extent* ι is to assign to each proposition the set of its proofs. A constructive version of the *comprehension principle* should be captured in this way. The *selection operator* ι, which Alonzo Church introduced in his *simple theory of types* (1940), is the classical ancestor of our ι – though based on a quite different idea. On the other hand, one version of the calculus of constructions (Coquand 1990) contained an operation T, which was meant to replace a proposition by the set of its proofs. But a proposition in the calculus (or theory) of constructions is, in a sense, nothing *but* the set of its proofs. Conceptually, the operation T does not do much; it is actually a syntactical device, introduced

[3] This is a notational convention. In general, a construction need not determine a unique context and range.

to secure the uniqueness of derivations. If all the ι-rules (T had only the introduction rule) would be added in the theory of constructions, the extent operation would just switch a proposition from universe to universe.

This operation is more interesting when combined with the *phase distinction*, the requirement that sets and elements never depend on proofs. (I.e., the indexing $\mathcal{P}\Rightarrow\mathcal{S}$ is forbidden.) The (implicite) context in all the extent rules – listed in Appendix I – must now consist of sets only: otherwise, a proposition contained in the context of a proposition α would be passed in the context of the set $\iota\alpha$. Therefore, only a *predicate* – a proposition indexed only by sets – can have an extent. The elements of the extent $\iota\alpha$ now correspond to the *logically closed proofs* of α, i.e. to those proofs which do not depend on other proofs (and have only some element-variables in their contexts). – This combination of the extent operation and the phase distinction characterizes the *theory of predicates.*

The fragments obtained by removing the Σ-operations from type theories will be called *calculi* here. We shall abbreviate by COC the calculus of constructions, and by COP the calculus of predicates. TOC and TOP will be the theory of constructions and the theory of predicates.

3. What can be expressed by predicates?

Now we shall list some facts which might offer an impression of the power of predicates, and of questions arising from them. The proofs are omitted; they are beyond the scope and the intention of this section. (Some of them can be found in my thesis.)

The notations are explained in Appendix I (or in section 2). "$\models\alpha$" means that "α is inhabited".

31. For every pair of functions $f,g:A\rightarrow B$, and elements $h,h':A$, all in the same context, the following statements are true:

$\models \forall X:A.fX\equiv gX$ $\quad\underline{\text{iff}}\quad$ $f=g$;

$\models \exists Z:\{X:A/\ fX\equiv gX\}.h\equiv\pi_0 Z$ $\quad\underline{\text{iff}}\quad$ $fh=gh$

$\models \forall XX':A.fX\equiv fX'\rightarrow X\equiv X'$ $\quad\underline{\text{iff}}\quad$ $fh=fh'$ $\underline{\text{implies}}$ $h=h'$

$\models \forall Y:B\exists X:A.gX\equiv Y$ $\quad\underline{\text{iff}}\quad$ g is a *quotient function*, i.e.

for every $k:A\rightarrow K$, such that $\models\forall XX':A.gX\equiv gX'\rightarrow kX\equiv kX'$ there is unique $\tilde{k}:B\rightarrow K$ such that $k=\tilde{k}\circ g$.

32. Writing $\wedge$ in place of $\times$, define

$$\exists!X:K.\gamma(X) := \exists X:K.\gamma(X)\wedge\forall XY:K.(\gamma(X)\wedge\gamma(Y))\rightarrow X\equiv Y.$$

Now consider the principle of *function coprehension*:

$$\models \forall X{:}A\,\exists! Y{:}B.\alpha(X,Y) \qquad \text{iff} \qquad \models \alpha(X,Y) \leftrightarrow fX \equiv Y \text{ for some } f.$$

In other words, the functions may be identified with the total and single-valued relations, as in set theory. The if-direction of the function comprehension is true in TOP: the graph $fX \equiv Y$ of a function f is provably total and single-valued. The then-direction, however, requires an operation ιX which would *extract singletons*, in the sense that

whenever $\models \exists! X{:}K.\gamma(X)$, then there is $\iota X.\gamma(X) : K$ with $\models \gamma\big(\iota X.\gamma(X)\big)$.

In Church's simple theory of types (1940), the operation ιX was derivable using the selector ι. (The logical systems of Frege, of Russell-Whitehead, of Hilbert-Bernays also contained operations like ιX.) Constructively, however, the function comprehension is independent from the set comprehension. It is not derivable in the theory of predicates[4], but it can be neatly introduced. For instance – by a slight intervention on the phase distinction:

$$\boxed{\textit{Predicate } \gamma \textit{ can occur in the context of a set only if } \models \gamma(X) \wedge \gamma(X') \to X \equiv X'}$$

Given $P{=}S{=}K$, $Q{=}\gamma$ and closed proofs $b{:}\exists X{:}K.\gamma$ and $c{:}\gamma(X) \wedge \gamma(X') \to X \equiv X$, the term

$$\iota X.\gamma(X) := \pi_0 b$$

can now be formed by ΣE and proved to be independent of b and c. (We assume that the condition $(S \leq Q)$ is omitted from ΣE in TOP. To introduce ιX in TOC, it is sufficient to strengthen ΣE by extending this condition to ($S \leq Q$ or $\models Q(X) \wedge Q(X') \to X \equiv X'$).)

33. Define

$$\upsilon_A \quad := \quad \lambda X^A Y^{\mathcal{P}A}.YX : A \to \mathcal{P}\mathcal{P}A, \text{ and}$$

$$\mathcal{P}f \quad := \quad \lambda Y^{\mathcal{P}B} X^A.\ Y(fX) : \mathcal{P}B \to \mathcal{P}A, \text{ for an arbitrary function } f{:}A \to B.$$

In ordinary set theory, for every set A there is a bijection

$$A \simeq \{X \in \mathcal{P}\mathcal{P}A /\ \upsilon_{\mathcal{P}\mathcal{P}A} X \equiv \mathcal{P}(\mathcal{P}\upsilon_A)X\}.$$

In TOP, we have a term u from left to right and – if the function comprehension is supported – a term n from right to left. They satisfy $n \circ u = id_A$, but not $u \circ n = id_{\{\ldots\}}$. An intuitive explanation can be that the set on the right side contains not just the *principal filters* on $\mathcal{P}A$, but also the proofs that they are principal filters, and there can be many of those for each of them.

Similar phenomena are met in encoding other set-theoretical constructions in TOP. E.g., the *disjoint union* can be defined by:

[4] To see this, consider a Heyting algebra $\mathcal{H}$ as a model for the theory of predicates. The sets are interpreted as the members of $\mathcal{H}$. For $a,b \in \mathcal{H}$, the relation $a \leq b$ represents a function from a to b. The type $\mathcal{P}$ of propositions will be the unit 1 of $\mathcal{H}$. (In terms of my thesis, we are looking at the category of predicates $id{:}\mathcal{H} \to \mathcal{H}$.) – The function comprehension fails in this model.

$$A_0+A_1 \ := \ \{X:\mathcal{P}(\mathcal{P}A_0\times\mathcal{P}A_1)/ \ \upsilon_{\mathcal{P}(\mathcal{P}A_0\times\mathcal{P}A_1)}X \equiv \mathcal{P}(\mathcal{P}\upsilon_{A_0}\times\mathcal{P}\upsilon_{A_1})X \ \}$$

Of course, there are inclusions $\kappa_i:A_i\rightarrow A_0+A_1$ ($i\in 2$) and the operation $[\_,\_]$, which assigns to each pair of terms $f_i:A_i\rightarrow B$ ($i\in 2$) a term $[f_0,f_1]:A_0+A_1\rightarrow B$, such that $[f_0,f_1]\circ\kappa_i=f_i$. However, $[\kappa_0,\kappa_1]=id$ need not be true.

Yet another example: If, except the powersets, no other products of sets were given in our theory, we could define them using the extents of some equations, just as above, adapting the constructions from topos theory. However, the λ-abstraction obtained in this way would not satisfy the Πη-rule.[5]

Morale: The constructions with constructive extents are not extensional, because these extents are blown up by some constructive proofs.

4. Comparing theories: the conceptual part

What are we going to do? The starting point of our reduction of TOC is a simple observation, formulated in lemmas 21, Appendix II:

- the universe of propositions is embedded in the universe of sets by the operation $\_\times 1:\mathcal{P}\longrightarrow\mathcal{S}$ (and $\_\times T:\mathcal{S}\longrightarrow\mathcal{P}$ is its reflection);
- this embedding preserves (up to isomorphism) all operations except the existential quantifier.

In particular, every sum or product over α is isomorphic with a sum resp. product over $\alpha\times 1$. This means that the theory of constructions is sufficiently redundant that propositions occurring in contexts can be replaced by sets. If we restrict TOC by allowing only sets to occur in the contexts – call such a theory TOC$_\mathcal{S}$ – and translate TOC-constructions into TOC$_\mathcal{S}$-constructions:

$$\big(...x:\alpha...\Rightarrow T(x)\big) \qquad \mapsto \qquad \big(...X:\alpha\times 1...\Rightarrow T(\pi_0 X)\big)$$

– nothing will be lost, in the sense that an isomorphic copy of each TOC-type will still be generated in TOC$_\mathcal{S}$.

[5] The exponent $A\rightarrow B$ could be obtained as a subset of $\mathcal{P}(A\times\mathcal{P}B)$. The sum $A+B$ is a subset of $\mathcal{P}(\mathcal{P}A\times\mathcal{P}B)$. Note the resemblance with classical logic, where $(A\rightarrow B)\leftrightarrow\neg(A\wedge\neg B)$ and $(A\vee B)\leftrightarrow\neg(\neg A\wedge\neg B)$.

But now, TOC$_{\mathcal{S}}$ respects the phase distinction, and can be translated in TOP. So TOC can be translated in TOP. On the other hand, TOP can surely be translated in TOC, since the extent operation is definable there:

$$\begin{aligned} \iota\alpha \quad &:= \quad \alpha\times 1 \\ \delta a \quad &:= \quad \langle a,\emptyset\rangle \\ \tau k \quad &:= \quad \pi_0 k. \end{aligned}$$

By this translation, however, many types which were not isomorphic in TOP become isomorphic in TOC; the former theory has "more" types. (Out of seven isomorphisms "through the border of the universes", which can be extracted from lemma 212 for TOC, only two exist in TOP: those from lemmas 222 and 223.) To relate the theories precisely, we added in TOP the terms $x{:}\alpha \Rightarrow \delta^* x{:}\ \iota\alpha\times\top$. They behave just like "$\langle\delta x,*\rangle$" would, if only δx could be formed. These terms force isomorphism of each predicate with (the reflection of) its extent (lemma 231). Consequently, the extent operation $\iota{:}\mathcal{P}\longrightarrow\mathcal{S}$ becomes an embedding, with the same preservation properties as $\_\times 1{:}\mathcal{P}\longrightarrow\mathcal{S}$ in TOC (lemma 232).

In the *strict theory of predicates* (STOP) – the one with $\delta^* x$ – the sums and products over propositions can be reduced to the sums and products over sets, just like in TOC. So we have a subtheory STOP$_{\mathcal{S}}\subseteq$STOP, just like TOC$_{\mathcal{S}}\subseteq$TOC. Moreover, STOP$_{\mathcal{S}}$ and TOC$_{\mathcal{S}}$ are isomorphic. The conclusion that STOP and TOC are equivalent can now be made following the topological idea that

> two spaces are homotopy equivalent iff they have isomorphic deformation retracts.

The next proposition shows the strict extents from another angle.

Proposition. (In STOP.) Let $T(x^{\alpha})$ and $T'(x^{\alpha})$ be arbitrary propositions, or proofs of the same proposition. The following rule is true:

ω if $T(a)=T'(a)$ for all logically closed proofs $a{:}\alpha$,
then $T(x)=T'(x)$.

In the presence of δ^* and $\delta^*\eta$, the ω-rule implies $\delta^*\beta$.

Proof. If $T(a)=T'(a)$ for all logically closed $a{:}\alpha$, then it holds for $\tau X{:}\alpha$, i.e.

$$X{:}\iota\alpha \Rightarrow T\big(\nu(\langle X,*\rangle,(X,*).\tau X)\big) = T(\tau X) = T'(\tau X) = T'\big(\nu(\langle X,*\rangle,(X,*).\tau X)\big).$$

According to lemma 14 (still Appendix II!), this implies

$$z{:}\ \iota\alpha\times\top\ \Rightarrow T\big(\nu(z,\ (X,*).\tau X)\big) = T'\big(\nu(z,\ (X,*).\tau X)\big).$$

Using $\delta^*\beta$, we get

$$x{:}\alpha \Rightarrow T(x) = T\big(\nu(\delta^* x,\ (X,*).\tau X)\big) = T'\big(\nu(\delta^* x,\ (X,*).\tau X)\big) = T'(x).$$

To derive $\delta^*\beta$ from ω, note that for

$$c\,(x) := v(\delta^* x,\ (X,*).b[\tau X/z]))$$

and for any logically closed $a{:}\alpha$ holds

$$c(a) = c(\tau(\delta a)) = v\big(\delta^* \tau(\delta a),\ (X,*).b[\tau X/z]\big) = v\big(\langle \delta a, * \rangle,\ (X,*).b[\tau X/z]\big) =$$
$$= b(\tau(\delta a)) = b(a).\bullet$$

Remarks. The last proposition is the type-theoretical version of the fact that the category $\mathcal{P}$ of propositions is generated by the terminal object in the models of TOC and STOP. This means that the operations

$$\begin{array}{lll} \beta(x) & \mapsto & \iota(\beta(\tau X)) \\ b(x) & \mapsto & \delta(b(\tau X)) \end{array}$$

are injective. In fact, a TOP-system supports the strict extents iff the second operation induces a bijection between the sets of closed terms of type $\alpha \rightarrow \alpha'$ and of $\iota\alpha \rightarrow \iota\alpha'$. (This can be deduced from III.4.3 and IV.2.2 in Pavlović 1990).

The ω-rule owes its name to the fact that it is an infinitary rule (with infinitely many premises). In our setting, however, it can be equivalently expressed with just one premis:

$$\omega \qquad \frac{X{:}\iota\alpha \Rightarrow T(\tau X) = T'(\tau X)}{x{:}\alpha \Rightarrow T(x) = T'(x)}$$

5. Comparing theories: the technical part

Instanciation. Consider a construction $T(X)$ and terms p and q which can be substituted for X. If for every judgement $J_T(X)$, involving $T(X)$ and possibly some more occurrences of X,

$J_T(p)$ implies $J_T(q)$,

then we say that $T(q)$ is an *instânce* of $T(p)$.

Usually, $T(p)$ is $T(X)$, and its instances are obtained by substitution. The example of the ω-rule shows, however, that this is not the only way to instanciate. (In the ω-rule, $T(q)$ is $T(x)$!) In the sequel, we shall actually use *instanciation* as the common name for the substitution and the ω-rule.

Equivalences. Let $\mathcal{M}$ and $\mathcal{N}$ be two systems. A *translation of systems* is a mapping $F{:}\mathcal{M} \longrightarrow \mathcal{N}$ which preserves the derivability ($\vdash$) and the convertibility ($=$). Moreover, it should be *coherent*, in the sense that

$$\left.\begin{array}{lll} F(\Gamma\Rightarrow T:U) & = & (\Gamma'\Rightarrow T':U') \\ F(\Gamma\Rightarrow U:V) & = & (\Gamma''\Rightarrow U'':V'') \end{array}\right\} \underline{\text{imply}}\ U' = U''.$$

Let M and N be two type theories. A *translation* $F:M \longrightarrow N$ assigns to every M-system $\mathcal{M}$ an N-system $F\mathcal{M}$ and a translation of systems $F_{\mathcal{M}}:\mathcal{M} \longrightarrow F\mathcal{M}$.

A subsystem $\mathcal{N} \subseteq \mathcal{M}$ is a *retract* of $\mathcal{M}$ if there is a translation $F:\mathcal{M} \longrightarrow \mathcal{N}$, which restricts to the identity on $\mathcal{N}$; moreover, every type Q from $\mathcal{M}$ must be isomorphic with an instance of $F(Q)$. More precisely, there is a chain of instanciations Ξ, which brings $F(Q)$ in the context of Q, and

$$F(Q)[\Xi] \simeq Q.$$

A subtheory $N \subseteq M$ is a *retract* of M if there is a translation $F:M \longrightarrow N$ such that every $F\mathcal{M}$ is a retract of $\mathcal{M}$ by $F_{\mathcal{M}}$.

Theories M and N are *equivalent* if there are translations $F:M \longrightarrow N$ *and* $G:N \longrightarrow M$, such that for every M-system $\mathcal{M}$ and N-system $\mathcal{N}$, $GF\mathcal{M}$ is a retract of $\mathcal{M}$ and $FG\mathcal{N}$ is a retract of $\mathcal{N}$.

Comments. Recall (from section 2) that a system is assumed to contain its universes, together with all "other" derivable formation sequents. The coherence requirement for translations applies therefore not only when U is a type, but also when it is a universe.

Usually, a subobject $I: \mathcal{N} \hookrightarrow \mathcal{M}$ is called retract of $\mathcal{M}$ when there is a map $F:\mathcal{M} \longrightarrow \mathcal{N}$ such that $F \circ I = id_{\mathcal{N}}$. The above definition requires $I \circ F \simeq id_{\mathcal{M}}$ too. Because of this, $\mathcal{N}$ can be understood as a *deformation* retract of $\mathcal{M}$; and our notion of equivalence can be understood as the *homotopy* equivalence. Note that each deformation retract of a system is equivalent to that system.

The idea is that theories should be equivalent if they have the same class of models.[6] For instance, the theory of Boolean algebras is equivalent with that of Boolean rings. The theory of Boolean algebras with the signature $\langle \vee, \rightarrow, 0\rangle$ is a retract of the one using $\langle \vee, \wedge, \rightarrow, \neg, 0, 1\rangle$. The cut elimination is a retraction of a sequent calculus.

Eliminating redundancies from a theory is like removing synonyms from a natural language. It becomes harder to speak, but easier to understand – closer to semantics. E.g., the cut-elimination yields unnatural proofs, but offers a crucial insight into what is provable.

[6] The morphisms which they induce on this class can be different.

As far as type theory is concerned, we want to consider as synonymous exactly those isomorphic types that would be identified semantically. (A complete semantics for the theory of predicates has been given in Pavlović 1990.)

Theorem. The theory of constructions (TOC) and the strict theory of predicates (STOP) are equivalent.

Proof. As explained in section 4, we shall define the following translations

$$\begin{tikzcd}
\mathrm{TOC} \arrow[d, shift left, "E"] & \mathrm{STOP} \arrow[d, shift right, "H"'] \\
\mathrm{TOC}_{\mathcal{S}} \arrow[u, hook, shift left, "\upharpoonright E"] \arrow[r, shift left, "F_{\mathcal{S}}"] & \mathrm{STOP}_{\mathcal{S}} \arrow[u, hook, shift right, "\upharpoonright H"'] \arrow[l, shift left, "G_{\mathcal{S}}"]
\end{tikzcd}$$

The subtheories which we consider are obtained from TOP resp. STOP by the restriction

$\mathrm{TOC}_{\mathcal{S}}$, $\mathrm{STOP}_{\mathcal{S}}$ $\boxed{\textit{Only sets may occur in contexts.}}$

In $\mathrm{TOC}_{\mathcal{S}}$, however, a provision must be made for the operation $\_x1{:}\mathcal{P} \longrightarrow \mathcal{S}$

$\mathrm{TOC}_{\mathcal{S}}$ $\boxed{\textit{The context of } 1{:}\mathcal{S} \textit{ may contain propositions.}}$

Translation *E*. For an arbitrary TOC-system $\mathcal{M}$, we simultaneously define two translations, D and $E{:}\mathcal{M} \longrightarrow \mathcal{M}$:

$$\begin{array}{lll}
D(\ldots X{:}Q\ldots \Rightarrow T{:}U) & := & (\ldots X{:}\lfloor Q \rfloor \ldots \Rightarrow \lfloor T \rfloor{:}\lfloor U \rfloor[d_Q X/X]), \\
E(\ldots X{:}Q\ldots \Rightarrow T{:}U) & := & (\ldots X{:}\lfloor Q \rfloor \ldots \Rightarrow \lceil T \rceil{:}\lceil U \rceil[d_Q X/X])^{7},
\end{array}$$

where $\lceil \_ \rceil$ and $\lfloor \_ \rfloor$ translate expressions as follows. ¢ denotes an atom, and $\Box$ stands for Σ or Π.

$$\begin{array}{lll}
\lceil ¢ \rceil & := & ¢ \\
\lceil \Box X{:}P.Q \rceil & := & \Box X{:}\, \lfloor P \rfloor . \lceil Q \rceil \\
\lceil \lambda X.q \rceil & := & \lambda X.\, \lceil q \rceil \\
\lceil pq \rceil & := & \lceil p \rceil \lfloor q \rfloor \\
\lceil \langle p,q \rangle \rceil & := & \langle \lfloor p \rfloor , \lceil q \rceil \rangle \\
\lceil \nu(r,\ (X,Y).s) \rceil & := & \nu(\lceil r \rceil,\ (X,Y).\lceil s \rceil)
\end{array}$$

[7] People who would prefer to change the name of a variable when translating it into a different type should assume a bookkeeping algorithm for variables here.

$$\begin{array}{llll}
\lfloor \mathcal{P} \rfloor & := \mathcal{S} & \lfloor \mathcal{S} \rfloor & := \mathcal{S} \\
\lfloor \alpha \rfloor & := \lceil \alpha \rceil \times 1 & \lfloor K \rfloor & := \lceil K \rceil \\
\lfloor a \rfloor & := \langle \lceil a \rceil, \phi \rangle & \lfloor k \rfloor & := \lceil k \rceil
\end{array}$$

Let us define the terms d_Q now. We want to substitute d_QX for $X{:}Q$ in order to replace $X{:}Q$ in a context by $X{:}D(Q)$. So we must have $d_Q{:}D(Q) \rightarrow Q[\Delta_Q]$, where Δ_Q is a sequence of substitutions of $d_PY{:}D(P)$ for each $Y{:}P$ in the context of Q. In other words, Δ_Q brings Q in the context of $D(Q)$ and $E(Q)$.

Note that $E(¢) = ¢[\Delta_¢]$, for a generator $\Gamma \Rightarrow ¢{:}U$.

$$\begin{array}{llll}
\multicolumn{4}{l}{d_Q : D(Q) \rightarrow Q[\Delta_Q]} \\
\quad d_\alpha & := e_\alpha \circ \pi_0 & \tilde{d}_\alpha & := \lambda x.\langle \tilde{e}_\alpha x, \phi \rangle \\
\quad d_K & := e_K & \tilde{d}_K & := \tilde{e}_K
\end{array}$$

$$\begin{array}{llll}
\multicolumn{4}{l}{e_Q : E(Q) \rightarrow Q[\Delta_Q]} \\
\quad e_¢ & := id_{¢[\Delta_¢]} & \tilde{e}_¢ & := id_{¢[\Delta_¢]} \\
\quad e_{\Box X:P.Q} & := v_\Box \circ w & \tilde{e}_{\Box X:P.Q} & := \tilde{w} \circ \tilde{v}_\Box
\end{array}$$

$$\begin{array}{ll}
\multicolumn{2}{l}{v_\Box : \big(\Box X{:}E(P).E(Q)\big) \rightarrow \big(\Box X{:}P.Q\big)} \\
\quad v_\Pi := \lambda Z.\, e_Q \circ Z \circ \tilde{e}_P & \tilde{v}_\Pi := \lambda Z.\, \tilde{e}_Q \circ Z \circ e_P \\
\quad v_\Sigma := v(Z, (X,Y).\langle e_PX, e_QY \rangle) & \tilde{v}_\Sigma := v(Z, (X,Y).\langle \tilde{e}_PX, \tilde{e}_QY \rangle)
\end{array}$$

$w : \big(\Box X{:}D(P).E(Q)\big) \rightarrow \big(\Box X{:}E(P).E(Q)\big)$ is the isomorphism from lemma 212; $\tilde{w}$ is its inverse.

This completes the definition of mappings D and E. Clearly, the substitution will be:

$$\begin{array}{lll}
D(T[p/X]) & := & D(T)[D(p)/X] \\
E(T[p/X]) & := & E(T)[D(p)/X].
\end{array}$$

A straightforward inductive argument shows that E and D are translations. The image of E is a TOC$_\mathcal{S}$-subsystem of $\mathcal{M}$. Call this subsystem $E\mathcal{M}$. Since all d_Q are isomorphisms, there are substitutions Ξ_Q which bring $D(Q)$ and $E(Q)$ back in the context of Q. (Ξ_Q puts $\tilde{d}_PY{:}P$ in place of $Y{:}D(P)$.) From the isomorphisms e_Q we get

$$e_Q[\Xi_Q] : E(Q)[\Xi_Q] \simeq Q$$

for every type Q from $\mathcal{M}$. Hence, $E\mathcal{M}$ is a retract of $\mathcal{M}$; TOC$_\mathcal{S}$ is a retract of TOC.

Translation *H*. The approach is completely the same: For an arbitrary STOP-system $\mathcal{N}$, we define two translations $I,H{:}\,\mathcal{N} \longrightarrow \mathcal{N}$, using $\lceil\_\rceil$ and $\lfloor\_\rfloor$ just as above: write H in place of E, I in place of D, and i_Q in place of d_Q.

The definition of $\lceil \_ \rceil = \lceil \_ \rceil_H$ is the same as that of $\lceil \_ \rceil_E$ above, plus:

$$\begin{array}{ll}
\lceil \iota\alpha \rceil & := \iota\lceil \alpha \rceil \\
\lceil \delta a \rceil & := \delta\lceil a \rceil \\
\lceil \tau k \rceil & := \tau\lceil k \rceil
\end{array}$$

$\lfloor \_ \rfloor_H$ deviates from $\lfloor \_ \rfloor_E$ a bit more:

$$\begin{array}{llcll}
\lfloor \mathcal{P} \rfloor & := \mathcal{S} & \qquad & \lfloor \mathcal{S} \rfloor & := \mathcal{S} \\
\lfloor \alpha \rfloor & := \iota\lceil \alpha \rceil & & \lfloor K \rfloor & := \lceil K \rceil \\
\lfloor a \rfloor & := \delta\lceil a \rceil & & \lfloor k \rfloor & := \lceil k \rceil
\end{array}$$

A real difference with respect to the situation in TOC is that there are no terms from propositions to sets in STOP – hence no isomorphisms between $I(\alpha)$ and α.

$i_Q : I(Q) \rightarrow Q[\Delta_Q]$

$$\begin{array}{ll}
i_\alpha & := h_\alpha \circ \tau \\
i_K & := h_K
\end{array}$$

$h_Q : H(Q) \rightarrow Q[\Delta_Q]$

$$\begin{array}{llcll}
h_¢ & := id_{¢[\Delta_¢]} & \qquad & \tilde{h}_¢ & := id_{¢[\Delta_¢]} \\
h_{\iota\alpha} & := \delta \circ h_\alpha \circ \tau & & \tilde{h}_{\iota\alpha} & := \delta \circ \tilde{h}_\alpha \circ \tau \\
h_{\Box X:P.Q} & := v_\Box \circ w & & \tilde{h}_{\Box X:P.Q} & := \tilde{w} \circ \tilde{v}_\Box
\end{array}$$

$v_\Box : \big(\Box X{:}H(P).H(Q)\big) \rightarrow \big(\Box X{:}P.Q\big)$ is defined exactly as in the E-part, but with h instead of e.

$w : \big(\Box X{:}I(P).H(Q)\big) \rightarrow \big(\Box X{:}H(P).H(Q)\big)$ is the isomorphism from lemma 232.

By a substitution Δ_Q along the terms i_P (for P from the context of Q), each type Q is brought in the context of $H(Q)$. The question is now how to get $H(Q)$ back in the context of Q without any inverses of i_P?

Note that the variables $X{:}I(\alpha)$ occurring in the context of $H(Q)$ are substituted in $\lceil Q \rceil$ by $[i_\alpha X/x]$. But $i_\alpha X = h_\alpha(\tau X)$. We can now instanciate by the ω-rule, and replace τX by x. So we put in the context of $H(Q)$ the variable $x{:}H(\alpha)$ in place of $X{:}I(\alpha)$ $(=\iota H(\alpha))$; and now we substitute: $\lceil Q \rceil[h_\alpha x/x]$.

If this is done for all propositions α occurring in the context of $H(Q)$, a chain of instanciations Θ_Q is obtained, which brings the term $h_Q{:}\, H(Q) \rightarrow Q[\Delta_Q]$ in a context "parallel" with that of Q. The only difference between the two contexts is that instead of $Y{:}P \in CX(Q)$, the context of $h_Q[\Theta_Q]$ contains $Y{:}H(P)$.

The terms h_Q and $\tilde{h}_Q$ remained, of course, inverse under the instanciation Θ_Q; hence $h_Q[\Theta_Q]:H(Q)[\Theta_Q] \simeq Q[\Delta_Q,\Theta_Q]$. To get these two terms back in the original context of Q, substitute now $\tilde{h}_P[\Theta_Q]Y$ for each $Y:H(P)$ in their contexts. Denote this sequence of substitutions by Ξ_Q.

It is not hard to see that $Q[\Delta_Q,\Theta_Q, \Xi_Q]=Q$. Namely, Δ_Q substituted i_PY for $Y:P$; Θ_Q replaced i_PY with h_PY; Ξ_Q put $\tilde{h}_PY$ in place of Y in h_PY; and $h_P(\tilde{h}_PY) = Y$. Hence

$$h_Q[\Theta_Q,\Xi_Q] : H(Q)[\Theta_Q,\Xi_Q] \simeq Q$$

for every type Q from $\mathcal{N}$. $H\mathcal{N}$ is a retract of $\mathcal{N}$; STOP$_\mathcal{S}$ is a retract of STOP.

Translations *F* and *G*. The maps $F_\mathcal{S}$: TOC$_\mathcal{S} \longrightarrow$ STOP$_\mathcal{S}$ and $G_\mathcal{S}$: STOP$_\mathcal{S} \longrightarrow$ TOC$_\mathcal{S}$ are easy to guess. The latter rewrites all the expressions from a STOP$_\mathcal{S}$-system $\mathcal{N}_\mathcal{S}$, replacing only:

$$\iota\alpha \quad \longmapsto \quad \alpha\times 1,$$
$$\delta a \quad \longmapsto \quad \langle a,\phi\rangle,$$
$$\tau k \quad \longmapsto \quad \pi_0 k;$$

the former goes the other way around. Note that the rules for ι and those for $\_\times 1$ are completely the same. So we have an isomorphism.

Given a TOC-system $\mathcal{M}$, define $F\mathcal{M}$ to be the smallest STOP-system containing the STOP$_\mathcal{S}$-system $F_\mathcal{S}\mathcal{M}_\mathcal{S}$. Given a STOP-system $\mathcal{N}$, let $G\mathcal{N}$ be the smallest TOC-system which contains $G_\mathcal{S}\mathcal{N}_\mathcal{S}$. Clearly, $GF\mathcal{M} \subseteq \mathcal{M}$ and $FG\mathcal{N} \subseteq \mathcal{N}$.

Further define for systems $\mathcal{M}$ and $\mathcal{N}$ the translations $F = F_\mathcal{M} : \mathcal{M} \longrightarrow F\mathcal{M}$ and $G = G_\mathcal{N} : \mathcal{N} \longrightarrow G\mathcal{N}$ as follows:

$$F := \upharpoonright_H \circ F_\mathcal{S} \circ E \text{ and}$$
$$G := \upharpoonright_E \circ G_\mathcal{S} \circ H.$$

Using $E\circ\upharpoonright_E=id$, $H\circ\upharpoonright_H=id$, $F_\mathcal{S}\circ G_\mathcal{S}=id$ and $G_\mathcal{S}\circ F_\mathcal{S}=id$, we get

$$G\circ F = \upharpoonright_E \circ E \text{ and}$$
$$F\circ G = \upharpoonright_H \circ H.$$

$F\circ G$ and $G\circ F$ are thus retractions, since E and H are.

Remark. The danger of working modulo isomorphisms is that whole groups (of automorphisms) can be swept away: reduced to an identity. This will not happen if ***unique canonical*** isomorphisms are used. The isomorphisms in the preceding theorem are clearly canonical, i.e. defined uniformly for all types. A curious reader will perhaps want to check that they are unique. (The assertions to be proved: For every canonical isomorphism $f_Q:E(Q)\rightarrow Q$, $D(f_Q)=id_{D(Q)}$ implies $f_Q=e_Q$; for every canonical $g_Q:H(Q)\rightarrow Q$, $I(g_Q)=id_{I(Q)}$ implies $g_Q=h_Q$.) –

For a full precision, the unicity requirement should be put in the definition of retracts. We refrained from this for the sake of simplicity.

6. How to compare calculi?

In the calculus of constructions, all operations can be reduced to those within the universe of sets: the exception from lemma 212 disappears. The restriction of the translation D on COC will therefore be a retraction. (Whole $D: \mathrm{TOC} \longrightarrow \mathrm{TOC}_S$ is not a retraction because of the mentioned exception: $D(\exists X{:}K.\beta) \not\simeq \exists X{:}K.\beta$.) So we can translate expression-wise here: a D-image of a sequent is obtained by simply applying $\lfloor\_\rfloor$ at each expression in it.

In the calculus of predicates, on the other hand, a new way of making extents strict must be invented, since the operation δ^* needs Σ. Two possibilities are suggested by proposition III.4.3 in my thesis. One is to force $\iota(\alpha \rightarrow \beta) \simeq \iota\alpha \rightarrow \iota\beta$ (by adding something like δ^*); otherwise force $\alpha \rightarrow \beta \simeq \forall X{:}\iota\alpha.\beta$. A proof of equivalence of the *strict calculus of predicates* – which contains these isomorphisms – and the calculus of constructions can be built along the same lines as the one presented above.

Appendix I

Rules

Structure (all type theories)

Assumption

$$\frac{\Gamma\Rightarrow P:\mathcal{U}}{\Gamma,X:P\Rightarrow X:P}\quad (X:P\notin\Gamma)$$

Weakening

$$\frac{\Gamma,\Delta\Rightarrow J\qquad\qquad \Gamma\Rightarrow P:\mathcal{U}}{\Gamma,X:P,\Delta\Rightarrow J}\quad (X:P\notin\Gamma,\Delta)$$

Substitution

$$\frac{\Gamma,X:P,\Delta\Rightarrow J\qquad\qquad \Gamma\Rightarrow p:P}{\Gamma,\Delta[p/X]\Rightarrow J[p/X]}$$

Replacement

$$\frac{\Gamma,X:P,\Delta\Rightarrow T:U\qquad\qquad \Gamma\Rightarrow p=q}{\Gamma,\Delta[p/X]\Rightarrow T[p/X]=T[q/X]}$$

Typing

$$\frac{\Gamma\Rightarrow p:P\qquad\qquad \Gamma\Rightarrow P=Q}{\Gamma\Rightarrow p:Q}$$

Equality (all)

$$\frac{}{p=p}\qquad\qquad \frac{p=q}{q=p}\qquad\qquad \frac{p=q\quad q=r}{p=r}$$

Unit (all)

$$\frac{}{1:\mathcal{U}}\qquad\qquad \frac{}{\emptyset:1}\qquad\qquad \frac{p:1}{p=\emptyset}$$

Universes (COC, COP, TOC, TOP, STOP)

$$\frac{}{\mathcal{P}:\mathcal{S}}$$

Products (COC, COP, TOC, TOP, STOP)

Π

$$\frac{X{:}P \Rightarrow Q : \mathcal{U}}{\Pi X{:}P.Q : \mathcal{U}}$$

ΠI

$$\frac{X{:}P \Rightarrow q{:}Q \qquad X{:}P \Rightarrow Q : \mathcal{U}}{\lambda X.q : \Pi X{:}P.Q}$$

ΠE

$$\frac{r : \Pi X{:}P.Q \qquad p{:}P}{rp : Q[p/X]}$$

$\Pi\beta$ $\quad (\lambda X.\, q)p = q[p/X]$

$\Pi\eta$ $\quad \lambda X.\,(tX) = t \qquad (X \notin CX(t))$

Sums (TOC, TOP, STOP)

Σ

$$\frac{X{:}P \Rightarrow Q : \mathcal{U}}{\Sigma X{:}P.Q : \mathcal{U}}$$

ΣI

$$\frac{p{:}P \qquad q{:}Q[p/X] \qquad X{:}P \Rightarrow Q : \mathcal{U}}{\langle p,q\rangle : \Sigma X{:}P.Q}$$

ΣE

$$\frac{r{:}\Sigma X{:}P.Q \quad X{:}P, Y{:}Q \Rightarrow s{:}S[\langle X,Y\rangle/Z] \quad Z{:}\Sigma X{:}P.Q \Rightarrow S : \mathcal{U}}{v(r,(X,Y).s) : S[r/Z]} \; (S \leq Q)$$

$\Sigma\beta$ $\quad v(\langle p,q\rangle, (X,Y).s) = s[p/X,\, q/Y]$

$\Sigma\eta$ $\quad v(r, (X,Y).t[\langle X,Y\rangle/Z]) = t[r/Z] \qquad (X,Y \notin CX(t))$

Comment. $S \leq Q$ means $RG(S){:}RG(Q)$ or $RG(S)=RG(Q)$. In other words, ΣE must not be applied when S is a set and Q a proposition. Due to the next rule, this cannot happen in (S)TOP at all; so that the condition can be omitted there.

Phase distinction (COP, TOP, STOP)

$$\boxed{\textit{The context of a set or an element must contain no propositions.}}$$

Extent (COP, TOP, STOP)

ι $$\frac{\alpha:\mathcal{P}}{\iota\alpha:\mathcal{S}}$$

ιI $$\frac{a:\alpha}{\delta a:\iota\alpha}$$ ιE $$\frac{k:\iota\alpha}{\tau k:\alpha}$$

ιβ $\tau(\delta a) = a$ ιη $\delta(\tau k) = k$

ιT $\iota T = 1$

Strict extent (STOP)

δ* $$\frac{a\ :\ \alpha}{\delta{*}a\ :\ \iota\alpha\times T}$$

δ*β $v(\delta{*}a, (X,*).b[\tau X/z]) = b[a/z]$

δ*η $\delta{*}(\tau k) = \langle k,*\rangle.$

Comment. Because of the phase distinction, the (implicite) context in all the extent rules may contain only sets; $\iota\alpha$ and δa can be formed only in such a context.

Notations

$P \rightarrow Q := \Pi X{:}P.Q \quad (X \notin CX(Q))$

$id_P := \lambda X^P.\, X^P$

$\pi_0 := \lambda Z.v(Z,(X,Y).X)$

$\mathcal{P}K := K \rightarrow \mathcal{P}$

$X \underset{K}{\equiv} Y := \forall \xi{:}\mathcal{P}K.\xi X \leftrightarrow \xi Y$

$P \times Q := \Sigma X{:}P.Q \quad (X \notin CX(Q))$

$p \circ q := \lambda X.p(qX)$

$\pi_1 := \lambda Z.v(Z,(X,Y).Y)$

$\{X{:}K/\ \alpha(X)\} := \Sigma X{:}K.\iota\alpha(X)$

Appendix II

Lemmas

1. About Σ. Due to the restriction on ΣE (in TOC, or to the phase distinction in TOP) the projection $\pi_0 : \exists X{:}P.Q \to P$ cannot be formed when P is a set and Q a proposition. The other three combinations of P and Q (set-set, proposition-set, proposition-proposition) allow both projections. In these situations, ΣE can be replaced by the projection rules, as in Hyland-Pitts 1989. (The equivalence of the two presentations follows from 11-13.)

11. $\pi_i \langle X_0, X_1 \rangle = X_i,\ i \in 2$ **12.** $\langle \pi_0 Z, \pi_1 Z \rangle = Z$

13. $s(\pi_0 Z, \pi_1 Z) = \nu(Z, (X_0, X_1).s)$ **14.** $\underline{\text{If}}\ s(\langle X,Y \rangle) = t(\langle X,Y \rangle)\ \underline{\text{then}}\ s = t.$

15. In the case when P is a set and Q a proposition, the rule ΣE can be modified (following the idea of $\exists$-elimination) by removing $Z{:}\Sigma X{:}P.Q$ from the context of S . In the theories considered here, the full ΣE-rule is still derivable from this modified instance. (Cf. Pavlović 1990, I.1.52.)

2. Isomorphisms are of course terms $\Gamma \Rightarrow p{:}P' \to P$ and $\Gamma \Rightarrow p'{:}P \to P'$, such that $p \circ p' = id$ $\underline{\text{and}}$ $p' \circ p = id$. We write $p{:}P' \simeq P$ to denote that p is an isomorphism, and $P' \simeq P$ to say that an isomorphism exists.

21. In TOC.

211. $\alpha \times 1 \simeq \alpha$

212. The statement:

$\underline{\text{if}}\ p{:}P' \simeq P\ \underline{\text{and}}\ Q(X^P) \simeq Q'(X^P)\ \underline{\text{then}}\ \Box X{:}P.\,Q(X) \simeq \Box X'{:}P'.\,Q'[pX'/X]$

holds for all types P, P', Q, Q' and for $\Box \in \{\Sigma, \Pi\}$, with one exception:

$A \simeq \alpha\ \underline{\text{does not imply}}\ \Sigma X{:}K.A \simeq \exists X{:}K.\alpha.$

22. In TOP.

221. $\iota(\iota\alpha \times T) \simeq \iota\alpha$

222. $\Sigma X{:}\iota\alpha.\iota\big(\beta(\tau X)\big) \simeq \iota(\Sigma x{:}\alpha.\beta)$

223. $\Pi X{:}A.\iota\beta \simeq \iota(\forall X{:}A.\beta)$

23. In STOP.

231. $\iota\alpha \times T \simeq \alpha$

232. For $\Box \in \{\Sigma, \Pi\}$ holds:

$$\begin{array}{ccccc} \iota(\Box x{:}\alpha.\beta) & \simeq & \iota(\Box X{:}\iota\alpha.\beta) & \simeq & \Box X{:}\iota\alpha.\iota\beta \\ \Box x{:}\alpha.\beta & \simeq & \Box X{:}\iota\alpha.\beta & \simeq & (\Box X{:}\iota\alpha.\iota\beta) \times T \end{array}$$

Some comments, some proofs.

212. The exception can perhaps be understood by looking at the set $A \simeq \alpha$ as the extent of α. The sum $\Sigma X{:}K.A$ is then the set $\{X{:}K/\ \alpha(X)\}$ of *all* the witnesses of $\alpha(X)$, while $\exists X{:}K.\alpha$ just says that there is *a* witness. – For a proof that these two types are not isomorphic one should consider a model (e.g. in Hyland-Pitts 1989).

221. The isomorphisms are:

$$X{:}\iota\alpha \Rightarrow \ \delta\langle X,*\rangle : \iota(\iota\alpha \times \top)$$
$$Y{:}\iota(\iota\alpha \times \top) \Rightarrow \ \delta\nu(\tau Y,\ (X,*).\tau X) : \iota\alpha.$$

We check one of two identities that must be proved:

$$\delta\Big\langle\delta\nu(\tau Y,(X,*).\tau X),\ *\Big\rangle \overset{\eta}{=} \delta\nu\Big(\tau Y,\ (X,*).\Big\langle\delta\nu(\langle X,*\rangle,(X,*).\tau X),\ *\Big\rangle\Big) \overset{\beta}{=}$$
$$\delta\nu\big(\tau Y,\ (X,*).\langle\delta\tau X,*\rangle\big) = \delta\tau Y = Y.$$

231. $x : \alpha \ \Rightarrow \ \delta^* x : \iota\alpha \times \top,$

$$z{:}\ \iota\alpha \times \top \ \Rightarrow \nu\big(z,\ (X,*).\tau Z\big){:}\ \alpha.$$

One identity:

$$\delta^*\nu\big(z,\ (X,*).\tau X\big) \overset{\eta}{=}$$
$$\nu\Big(z,\ (X,*).\delta^*\nu\big(\langle X,*\rangle,\ (X',*).\tau X'\big)\Big) \overset{\beta}{=}$$
$$\nu\big(z,\ (X,*).\delta^*(\tau X)\big) = \nu\big(z,\ (X,*).\langle X,*\rangle\big) = z$$

232. Everything follows from previous results, plus:

$$\Sigma x{:}\alpha.\beta \simeq \exists X{:}\iota\alpha.\beta(\tau X) \text{ and}$$
$$\iota(\Pi x{:}\alpha.\beta) \simeq \Pi X{:}\iota\alpha.\iota\big(\beta(\tau X)\big).$$

The second isomorphism is obtained using 231 and

$$\iota(\Pi x{:}(\iota\alpha\times\top).\gamma(x)) \simeq \ \Pi X{:}\iota\alpha.\iota\big(\gamma(\langle X,*\rangle)\big)$$

And this last iso is definable in the theory of predicates:

$$Z : \iota(\Pi x{:}(\iota\alpha\times\top).\gamma(x)) \Rightarrow \ \lambda X.\delta\big((\tau Z)\langle X,*\rangle\big) : \Pi X{:}\iota\alpha.\iota\big(\gamma(\langle X,*\rangle)\big)$$
$$Y{:}\Pi X{:}\iota\alpha.\iota\big(\gamma(\langle X,*\rangle)\big) \Rightarrow \ \delta\lambda x.\nu\big(x,\ (X,*).\tau(YX)\big){:}\ \iota(\Pi x{:}(\iota\alpha\times\top).\gamma(x))$$

As for the first of the above isomorphisms, we have

$$\Sigma x{:}\alpha.\beta \ \simeq \ \iota(\Sigma x{:}\alpha.\beta)\times\top \ \simeq \ (\Sigma X{:}\iota\alpha.\iota\beta)\times\top \ \overset{\#}{\simeq} \exists X{:}\iota\alpha.(\iota\beta\times\top) \ \simeq$$
$$\simeq \ \exists X{:}\iota\alpha.\beta.$$

The step (#) is a special case of $\exists Z{:}(\Sigma X{:}A.B).\gamma \simeq \exists X{:}A.\exists Y{:}B.\gamma$.

References

Cartmell, J.

(1986) Generalized algebraic theories and contextual categories, *Ann. Pure Appl. Logic* 32, 209-243

Church, A.

(1940) A Formulation of the Simple Theory of Types, *J. Symbolic Logic*, 5(1), pp. 56-68

Coquand, Th.

(1990) Metamathematical Investigations of a Calculus of Constructions, *Logic and Computer Science* (Academic Press)

Coquand, Th., Huet, G.

(1986) Constructions: A higher order proof system of mechanizing mathematics, *EUROCAL 85, Linz* , Lecture notes in Computer Science 203 (Springer, Berlin)

(1988) The Calculus of Constructions, *Information and Computation* 76, 95-120

Ehrhard, T.

(1989) Dictoses, *Category theory in computer science,* Lecture Notes in Computer Science 389 (Springer, Berlin), 213-223

Girard, J.-Y.

(1972) Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types, *Proceedings of the Second Scandinavian Logic Symposium* (North-Holland, Amsterdam) 63-92

Harper, R., Mitchell, J.C., Moggi, E.

(1990) Higher-Order Modules and the Phase Distinction, to appear in the *Proceedings of the 17th POPL ACM Conference*

Hyland, J.M.E., Pitts, A.M.

(1989) The theory of constructions: categorical semantics and topos-theoretic models, *Categories in Computer Science and Logic (Proc. Boulder 1987)*, Contemporary Math. (Amer. Math. Soc., Providence RI)

Lambek, J., Scott, P.J.

(1986) *Introduction to higher order categorical logic* , Cambridge studies in advanced mathematics 7 (Cambridge University Press, Cambridge)

Meseguer, J.

(1989) Relating Models of Polymorphism, *Conference Record of the XVI ACM POPL Symposium*, 228-241

Moggi, E.

(1990) A category-theoretic account of program modules, Manuscript

Pavlović, D.

(1990) *Predicates and Fibrations: From Type Theoretical to Category Theoretical Presentation of Constructive Logic*, Thesis (State University Utrecht)

Seely, R.A.G.

(1987) Categorical semantics for higher order polymorphic lambda calculus, *J. Symbolic Logic* 52(4), 969-989

Troelstra, A.S., Dalen, D. van

(1988) *Constructivism in Mathematics. An Introduction*, Studies in Logic and Foundations of Mathematics 121, 123 (North-Holland, Amsterdam)