Fundamenta Informaticae 19 (1993) 51–85
IOS Press

SUBSTITUTION UP TO ISOMORPHISM

P.-L. CURIEN
LIENS (CNRS – Ecole Normale Supérieure), LIENS, 45 rue d'Ulm, 75230 Paris Cedex 05, e-mail curien@dmi.ens.fr

1 Introduction

The aim of this article is two fold. First we offer a pictorial description of the categorical semantics of dependent types, the essence of which was known since the mid seventies, after the works of Cartmell [Cart] along a computer science line, and of Seely [Se] along a categorical logic line. These early ideas have been revisited by several authors, including Ehrhard [Ehr], Jacobs [Jac], Lamarche [Lam], Obtulowicz [Ob], Pavlovic [Pav], Streicher [Stre], and Taylor-Hyland-Pitts [HyPi,Tay]. The goals were to accommodate these semantics with domain theory on one hand, and Grothendieck fibrations on the other hand, and to lift them to semantics of the calculus of constructions of Coquand-Huet. We follow here Seely's semantics in locally cartesian closed categories.

The exposition is biased towards a second aim, which is to solve a difficulty arising from a mismatch between syntax and semantics: in locally cartesian closed categories, substitution in types is modelled by pullbacks (more generally pseudo-functors), that is, only up to isomorphism, unless split fibrational hypotheses are imposed. Many semantics, like those based on families of sets, or of domains, as described by Dybjer [Dyb], and Palmgren and Stoltenberg-Hansen [PalmStol], do satisfy such hypotheses, but not all semantics do satisfy them, and in particular not the general description of the interpretation in an arbitrary locally cartesian closed category. In the general case, we have to show that the isomorphisms between types arising from substitution are *coherent* in a sense familiar to category theorists. Due to this coherence problem at the level of types, we are lead to:

- switch to a syntax where substitutions are explicitly present (in traditional syntaxes substitution is a meta-operation, defined by induction);

- include type equality judgements in this modified syntax: we consider here only equalities describing the stepwise performance of substitution.

These changes introduce a new "flaw". In the first-order λ-calculus, typing proofs are unique. This is not true anymore when type equality judgements are added: there are now different proofs that a given term has a given type. This happens because there is a rule which says that if a term M has type σ, and if σ is equal to σ', then M has also type σ', so that at any stage in a proof of well-typing of a term M, one may intersperce type equality judgements. Thus coherence arises not only at the level of types, but also at the level of terms.

We already investigated coherence problems in a different setting (a system with polymorphism and type inclusion, joint work with G. Ghelli) [CuGhe]. As in [CuGhe] we attack this problem with tools of rewriting theory. We exhibit equivalences between typing proofs which are valid in any locally cartesian closed category (and, we believe, in more general models like relatively locally cartesian closed categories [HyPi], or D-categories [Ehr]), and are complete in the sense that the following, informally stated coherence properties hold:

- any two proofs of the same type equality are provably equal,
- any two proofs establishing that a given term is well-typed derive provably equal types for this term,
- any two proofs deriving the same type for the same term are provably equal.

To our knowledge, the work presented here is the first solution to this problem, which, until very recently, had not even been clearly identified, mainly due to emphasis on interesting mathematical models rather than on syntactic issues.

Prerequisites are the notion of locally cartesian closed category, whose definition is recalled in Section 2, and of a calculus of dependent types (a first source and agreeable reference is [Mar]). Syntax is given when needed in the text. We assume some experience of categorical logic (including the quantifiers-as-adjoints paradigm: the source reference [Law] is a nice reading). We also refer to the survey paper [Cur2], where an effort is made to suggest the categorical structures from suitable presentations of syntax. In particular the reader may find there (but also in [Cur1], and in the source paper [Brui]) an account of De Bruijn's nameless notation, which is also adopted here, and which we now briefly recall.

De Bruijn's notational convention consists in replacing variable occurrences by a natural number recording their place in the environment, added to their binding depth, as illustrated by the following example: in the environment $\{t=\ldots, z=\ldots\}$, the terms t and $\lambda x.(\lambda y.y)z$ become 1 and $\lambda.(\lambda.1)3$, respectively. In order to find z's binder one has to "pass" over λx and the top t of the environment, viewed as a stack. This operational flavor has been exploited in the Categorical Abstract Machine [CouCurMau] (see also [ACCL] for more recent work on machine-oriented syntax).

In Section 2, we give a pictorial account of the interpretation of the syntax of the calculus of first-order dependent types in locally cartesian closed categories. In Section 3, we raise the mismatch problem quoted in the abstract, and prove our coherence results. We conclude in Section 4.

2 Interpreting dependent types in locally cartesian closed categories:

We recall successively the notion of locally cartesian closed category (Section 2.1), and the syntax of first-order λ-calculus with dependent types (that is, the kernel of Martin-Löf type theory, without identity types and without universes) (Section 2.2).We then turn to a name-free syntax (Section 2.3), which is more appropriate to describe the interpretation. Finally, we give a pictorial account of the semantics of this calculus in locally cartesian closed categories (Section 2.4).

2.1 Locally cartesian closed categories

Let $\mathbb{C}$ be a category, and A be an object of $\mathbb{C}$. The slice $\mathbb{C}/A$ is the category whose objects are the arrows $f{:}B\to A$ with codomain A, and whose homsets $(\mathbb{C}/A)(f,g)$ consist of the arrows h such that $g\circ h = f$.

Definition 1: A *locally cartesian closed category* (LCCC for short) is a category $\mathbb{C}$ which has a terminal object, and is such that all slices $\mathbb{C}/A$ are cartesian closed.

In particular a LCCC is cartesian closed, noticing that $\mathbb{C}$ and $\mathbb{C}/1$ are equivalent categories. The LCCC's admit a characterization by adjunctions, extending the characterization of categories with pullbacks.

Proposition 2: A category $\mathbb{C}$ is locally cartesian closed iff it has a terminal object, and for any $k{:}\ A\to B$ the functor $\sum k{:}\ \mathbb{C}/A \to \mathbb{C}/B$ defined by $(\sum k)(f) = k\circ f$ admits two successive right adjoints, written k^*, $\prod k$, that is:

- $\sum k \dashv k^* \dashv \prod k$.

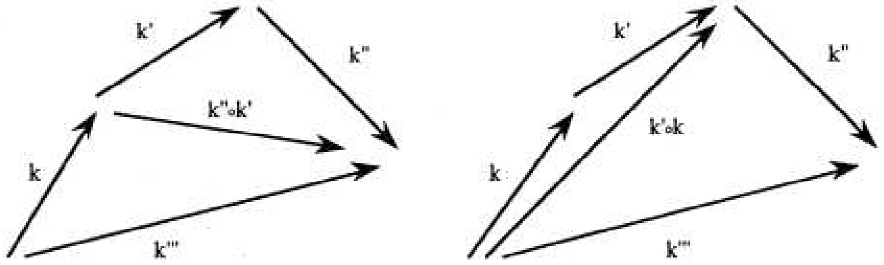
Proof: See [Fre].

We actually make use of this second definition.

Notation: We often write $\sigma[k]$ for $k^*\sigma$, and $\sum k.f$ for $(\sum k)(f)$ (and similarly for $\prod$).

A feature which is crucially used in the sequel is the *pseudo-functorial* character of the pullback. This can be explained as follows. For each pair of composable arrows t and s, there exists a natural iso $\psi_{s,t}{:}\ (s\circ t)^* \to t^*\circ s^*$ (ι for short), and those isos are coherent in the sense that the transformation between two paths connecting the same points in any commuting diagram, obtained by pasting those elementary isos, is independent of the decomposition of the pasting. This is made clear by the following example picture, which contains the essence of this coherence property. The picture shows the two ways to fill the space between the path k k' k" and the path $k'''\equiv(k''\circ k')\circ k$..

Figure 1



The two decompositions induce an equality of natural transformations: in the following equation, we use $\bullet$ to denote the vertical composition of natural transformations:

$-\ \psi_{k,k''\circ k'} \bullet \psi_{k',k''}k = k''\psi_{k,k'} \bullet \psi_{k'\circ k,k''}$.

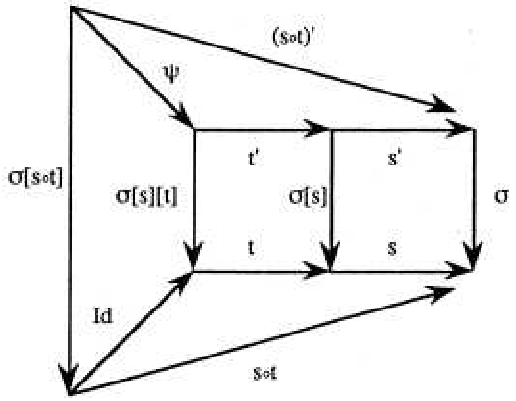
The two equated transformations are from $k''^{*}\circ k'^{*}\circ k^{*}$ to k'''^{*}.

It is well known that the satisfaction of this equation is sufficent to ensure that *all* pastings are coherent [2].

Why are pullbacks only pseudo-functorial? The point is that pullback diagrams compose, but *chosen* pullbacks do not in general. The isomorphism ψ can be constructed in two ways:

1) By a direct, "pointwise" argument, as illustrated on Figure 2, where the two inner squares and the outer rhombus are the chosen pullback diagrams of s and σ, t and $\sigma[s]$, and $s\circ t$ and σ, respectively. One first constructs the mediating arrow ψ' of $(s\circ t)'$ and $t\circ\sigma[s\circ t]$ (not shown on the picture), and then the mediating arrow ψ_σ (ψ for short) of ψ' and $\sigma[s\circ t]$.

Figure 2



2) By making use of the adjunctions $\sum k \dashv k^{*}$ and of the obvious (but special) property that $\Sigma(s\circ t) = \Sigma s \circ \Sigma t$ .The two inverse natural transformations obtained by this more abstract argument are given on Figures 3 and 4:

[2]The coherence of pseudo-functors can be reduced to the coherence in bicategories (which is the same as coherence of monoidal categories) [MacLaPar].

Figure 3

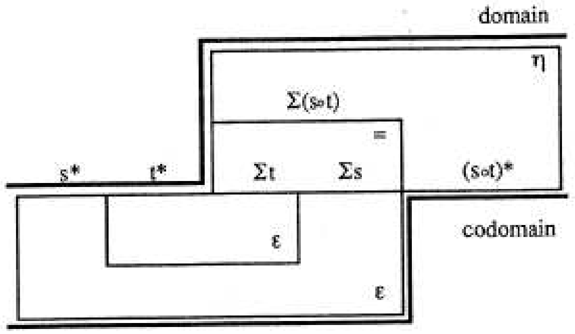
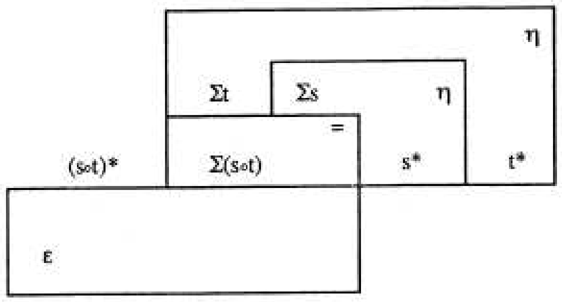


Figure 4



This way of representing natural transformations (more generally 2-cells) is known as pasting. It has been proved to be mathematically well defined only recently [Pow]. A pasting is a labelled planar graph (with additional properties, see [Pow]): points are categories (0-cells), arcs are functors (one-cells) and (bounded) regions are natural transformations (2-cells). In the pictorial representation of pastings adopted in Figures 3 and 4, the vertical lines are irrelevant, and the unlabelled horizontal lines are identity functors. The domain and the codomain of a natural transformation are retrieved by reading its upper and lower boundary. The limit between the upper and lower boundary is given by the leftmost and rightmost vertical lines of the boundary, respectively. We have represented the upper and lower boundaries of ψ^{-1} explicitly in Figure 3.

In Annex A.1 we give a pictorial evidence that these transformations are inverse. In Annex A.2 we prove the coherence condition displayed on Figure 1. The technique used is pasting rewriting: it has been conceived, and used in mathematical practice for at least twenty years, by the algebraists of the University of Santiago de Compostela. The rewritten pastings are called "Rodeja carpets" after the name of their initiator. Independently, another, dual notation is also since long in use among theoretical physicists working intensively with tensorial calculus. These dual pastings are called "Penrose diagrams", after their initiator. Lafont suggested that Penrose diagrams are more practical than Rodeja carpets (see Annex A.2). The theoretical study of the geometry of Penrose diagrams has been the subject of recent interest ([Bur, Laf, JoyStre1, JoyStre2]).

Remark: Pseudo-functoriality arises also for the identities: in the category Set of sets and functions, with the usual way of choosing pullbacks (take $\{(x,y) \mid s(x)=\sigma(y)\}$, we get that $\sigma[id]$ is not id. So one should also consider canonical isos between $\sigma[id]$ and id, and assume as a second coherence condition that the isos $k^* \circ id^* \leftrightarrow (id \circ k)^*$ ($= k^*$) and $k^* \circ id^* \leftrightarrow id \circ k^*$ ($= k^*$) coincide. In the present paper we assume though, for simplicity, that id^* is id, which can be done by choice. This choice is safe, since it is then clear that the iso $id^* \circ k^* \leftrightarrow (id \circ k)^*$ is the identity, by the uniqueness of mediating arrows. Thus, when choosing id^* as id, we may freely forget about the second coherence condition. Actually we assume more widely, still by choice, that ι^* is $\Sigma(\iota^{-1})$ for any iso ι (this is used in the proof of theorem 8, see Annex B.4).

Remark: When restricting attention to special classes of pullbacks, functoriality of pullbacks can be obtained. This happens in some models of dependent types, based on the idea of families of sets. We refer to [Dyb, PalmStol], and also to [Cur2] for details, and content ourselves here with the following hint. We restrict, in Set, the functors k^*: Set/B $\rightarrow$ Set/A to arrows σ which are first projection functions, that is, the domain of σ is a subset of a cartesian product $B \times C$, and σ is the (restriction of) the first projection function. The pullback of k and σ can then be chosen, not as

$\{(x,(y,z)) \mid k(x)=y$ and (y,z) is in the domain of $\sigma\}$,

but more simply as

$\{(x,z) \mid (k(x),z)$ is in the domain of $\sigma\}$.

For this choice, $\sigma[s \circ t]$ and $\sigma[s][t]$ coincide.

The adjunctions $\sum k \dashv k^*$, together with the pseudo-functoriality of $*$, determine for each commuting square a natural transformation $(\Sigma\sigma[s]) \circ s'^* \rightarrow s^* \circ (\Sigma\sigma)$, as shown on Figure 5:

Figure 5

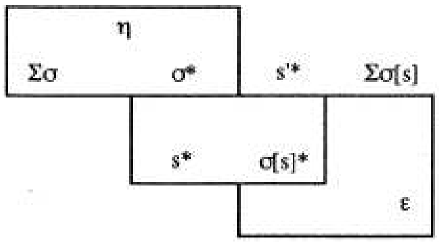
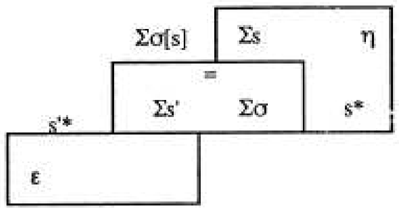


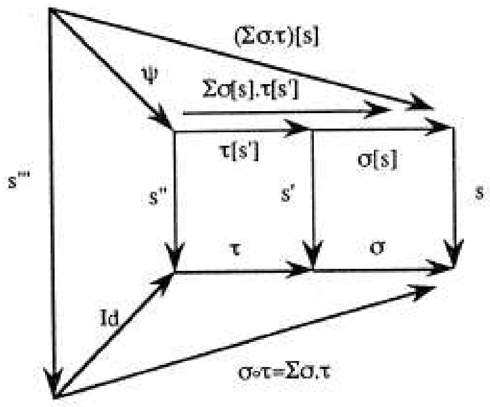
Figure 5 is based on the right inner pullback square of Figure 2. (At this stage we only use that it is a commuting diagram.) A more traditional way of describing this natural transformation is

$$((\Sigma\sigma[s] \circ s'^*)\, \eta) \bullet ((\Sigma\sigma[s])\, \iota\, (\Sigma\sigma)) \bullet (\varepsilon\, (s^* \circ \Sigma\sigma)).$$

This construction works in more general indexed categories than the indexed category of slices of a category with pullbacks. In the particular case where $*$ is pullback, there is a more direct way of describing this natural transformation, and moreover it is iso. One may start from the general form above, and open the middle "black box" rectangle, which consists of the composition of a ψ^{-1} with codomain $(\sigma \circ s')^*$, and of a ψ with domain $(s \circ \sigma[s])^*$. After two $\eta\varepsilon$-cancellations of the kind described in Annex A.1, we arrive at



To show that this transformation is iso, we go down one level of abstraction. We picture its pointwise inverse below. The picture is just an α-conversion of Figure 2. The component at τ of the natural transformation $s^* \circ (\Sigma\sigma) \to (\Sigma\sigma[s]) \circ s'^*$ is exactly the canonical ψ: $(\sigma \circ \tau)^* \to \tau^* \circ \sigma^*$ (but now it is in the slice over the domain of s, instead of being over the domain of τ). The right inner square is a pullback by assumption. The left inner square and the outer rhombus are pullbacks by construction.



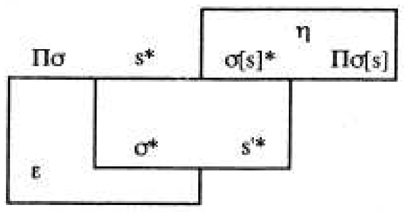
The property that the transformation $(\Sigma\sigma[s]) \circ s'^* \to s^* \circ (\Sigma\sigma)$ is iso is known as *Beck-Chevalley condition*. The notion is important for categorical logic, since it allows to interpret substitution across quantifiers. But it also appears in other geometric or topological applications of category theory. There exist more abstract versions of this condition [Gui]. The Beck-Chevalley condition can also be expressed in a synthetic way, using the notion of fibered adjunction [Jac].

Symmetrically, the right adjoints to pullbacks yield canonical transformations $s^* \circ (\Pi\sigma) \to (\Pi\sigma[s]) \circ s'^*$.

Lemma 3: The canonical transformations $(\Sigma\sigma[s]) \circ s'^* \to s^* \circ (\Sigma\sigma)$ are iso iff the canonical transformations $s^* \circ (\Pi\sigma) \to (\Pi\sigma[s]) \circ s'^*$ are iso.
Proof: See Annex A.3.

2.2 Syntax of first order λ-calculus with dependent types. We present a pure first-

Figure 6



order λ-calculus with dependent types. That may seem a rather frustrating calculus, because without some dependent constants, no true dependency arises (as was formally shown in [MeyRein]). But the syntax is prepared to accept such constants. The typical example from computer science is list(n), the type of lists of length at most n, a type depending on the type nat of natural numbers. The main conceptual step is independent of the specific choice of those constants. Dependent types, unlike simple types, have to be proved well-formed. One first defines a syntax of raw (or pre-well-formed) types and terms, given by

$$\sigma ::= \kappa \mid \prod x{:}\sigma.\sigma \mid \sum x{:}\sigma.\sigma \quad (\kappa \text{ base type})$$
$$M ::= x \mid \lambda x{:}\sigma.M \mid MM \mid (M,M) \mid \mathrm{fst}(M) \mid \mathrm{snd}(M)\,.$$

Dependency arises when a dependent constant κ can be formed from terms M (cf. list(n) above, or the equality type I(M,N) of Martin-Löf).

The typing rules are as follows. A context C is a sequence of assertions of the form x:σ. We say that x is defined in C if x:σ occurs in C, for some σ We denote by C(x) the first such σ, starting from the right. There are three kinds of judgements:

C context , $C \vdash \sigma$ type , and $C \vdash M{:}\sigma$.

Context formation rules

$\varnothing$ context

$$\frac{C \vdash \sigma \text{ type} \quad (x \text{ not defined in } C)}{C, x{:}\sigma \text{ context}}$$

Type formation rules

[Const]

κ type

(for true dependency one would have an inference: $C \vdash M_1{:}\sigma_1, \ldots, C \vdash M_n{:}\sigma_n$ entail $C \vdash \kappa(M_1,\ldots,M_n)$ type)

[∏] (and symmetrically [∑])

$$\frac{C, x{:}\sigma \vdash \tau \text{ type}}{C \vdash \prod x{:}\sigma.\tau \text{ type}}$$

Term formation rules

[Var]

$$\frac{C \text{ context} \quad (C(x) \text{ defined})}{C \vdash x{:}C(x)}$$

[Abs]

$$\frac{C, x{:}\sigma \vdash M{:}\tau}{C \vdash \lambda x{:}\sigma.M{:}\ \prod x{:}\sigma.\tau}$$

[App]

$$\frac{C \vdash M{:}\prod x{:}\sigma.\tau \quad C \vdash N{:}\sigma}{C \vdash MN{:}\ \tau[N/x]}$$

[Pair]

$$\frac{C \vdash M{:}\sigma \quad C \vdash N{:}\tau[M/x]}{C \vdash (M,N){:}\ \sum x{:}\sigma.\tau}$$

[fst]

$$\frac{C \vdash M{:}\sum x{:}\sigma.\tau}{C \vdash \mathrm{fst}(M){:}\ \sigma}$$

[snd]

$$\frac{C \vdash M{:}\sum x{:}\sigma.\tau}{C \vdash \mathrm{snd}(M){:}\ \tau[M/x]}$$

2.3 Name-free syntax Next we turn to a name-free syntax, which is well-suited to the description of the interpretation. Also, as quoted in the introduction, we pay particular attention to substitution, which is modelled in general only up to isomorphism. So we include an explicit syntax of substitutions, as already undertaken in [ACCL]. We refer to [ACCL] for an operational explanation of the notation and of the operations. But the reader can get a "graphical" insight from the pictures which follow.

Types:	$\sigma ::= \kappa \mid \prod\sigma.\sigma \mid \sum\sigma.\sigma \mid \sigma[s]$
Terms:	$M ::= 1 \mid \lambda\sigma.M \mid MM \mid (M,M) \mid \mathrm{fst}(M) \mid \mathrm{snd}(M) \mid M[s]$
Substitutions:	$s ::= id \mid \uparrow \mid M.s \mid s \circ s$

The contexts are now just sequences of types. There are four kinds of judgements: C context, $C \vdash \sigma$ type, $C \vdash M{:}\sigma$, and $C \vdash s{:}C'$.

Context formation rules

$$\varnothing \text{ context}$$

$$\frac{C \vdash \sigma \text{ type}}{C,\sigma \text{ context}}$$

Type formation rules

[Const]

$$\kappa \text{ type}$$

[$\prod$] (and symmetrically [$\sum$])

$$\frac{C, \sigma \vdash \tau \text{ type}}{C \vdash \prod\sigma.\tau \text{ type}}$$

[σClos]

$$\frac{C \vdash s{:}C' \quad C' \vdash \sigma \text{ type}}{C \vdash \sigma[s] \text{ type}}$$

Term formation rules

[Var]

$$\frac{C \vdash \sigma \text{ type}}{C,\sigma \vdash 1{:}\sigma[\uparrow]}$$

[Abs]

$$\frac{C,\sigma \vdash M{:}\tau}{C \vdash \lambda\sigma.M{:}\ \prod\sigma.\tau}$$

[App]

$$\frac{C \vdash M{:}\prod\sigma.\tau \quad C \vdash N{:}\sigma}{C \vdash MN{:}\ \tau[N.id]}$$

[Pair]

$$\frac{C \vdash M{:}\sigma \quad C \vdash N{:}\tau[M.id]}{C \vdash (M,N){:}\ \sum\sigma.\tau}$$

[fst]

$$\frac{C \vdash M{:}\sum\sigma.\tau}{C \vdash \text{fst}(M){:}\ \sigma}$$

[snd]

$$\frac{C \vdash M{:}\sum\sigma.\tau}{C \vdash \text{snd}(M){:}\ \tau[\text{fst}(M).id]}$$

[MClos]

$$\frac{C \vdash s{:}C' \quad C' \vdash M{:}\sigma}{C \vdash M[s]{:}\ \sigma[s]}$$

(As a hint for the "mutation" of σ in Var, notice that in the hypothesis σ has its De Bruijn indices refering to the sequence C, whereas $\sigma[\uparrow]$ in the conclusion refers to the larger context C,σ; see the rules MClos and $\uparrow$ below.)

Substitution formation rules

[id]

$$\frac{C \text{ context}}{C \vdash \mathrm{id}: C}$$

[↑]

$$\frac{C \vdash \sigma \text{ type}}{C,\sigma \vdash \uparrow: C}$$

[Cons]

$$\frac{C \vdash s: C' \quad C' \vdash \sigma \text{ type} \quad C \vdash M: \sigma[s]}{C \vdash M.s: C',\sigma}$$

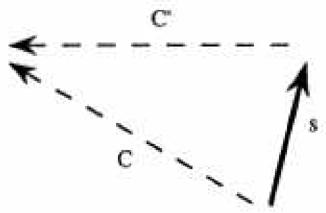
[Comp]

$$\frac{C \vdash s': C' \quad C' \vdash s: C''}{C \vdash s \circ s': C''}$$

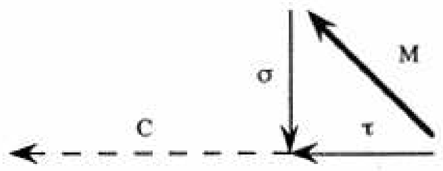
2.4 Semantics in LCCC's We turn to a "pictorial" interpretation of the calculus just defined in a locally cartesian closed category. A context is mapped to a sequence of consecutive arrows, the last one going into the (chosen) terminal object 1. A type is interpreted likewise, but setting a "marker" just after the first arrow of this sequence, the rest of the sequence being the meaning of the context with respect to which the type is well-formed.

The basic semantic ingredient here is "type-as-(projection) arrow": think of list(n) as represented by the first projection $(n,l) \rightarrowtail n$ on the infinite sum $\{(n,l) \mid l \in \mathrm{list}(n)\}$. Alternatively one may think of the interpretation of $C \vdash \sigma$ type as a meta (or global) sum "$\Sigma C.\sigma$" (to be contrasted to the "local" sums, say $\Sigma\sigma.\tau$ with $C,\sigma \vdash \tau$ type).

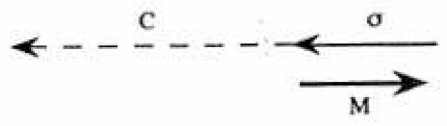
Judgements $C \vdash s: C'$ are mapped to (commuting) triangles:



Finally judgements $C \vdash M{:}\sigma$ are mapped to *kites*, which we define to be figures like the one below (where τ is the identity):



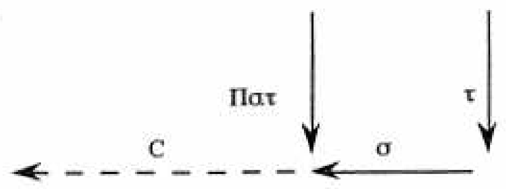
We often represent this picture in the following degenerate way:



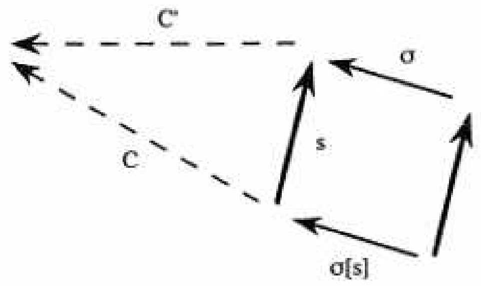
We stress that we do not give meanings to judgements, but to proofs of judgements. This does not matter at present, since there is only one way to prove a judgement, but will matter when we come to the discussion of equations. We present the definition of the interpretation rather informally, and make a notational confusion between syntax and meaning.

[Const] A basic type is mapped to an arrow into the terminal object.

[∏] (and symmetrically [∑]) The meaning of the ∏ and ∑ constructors is given by the adjoints ∑k and ∏k (thus we freely confuse, say ∏σ.τ and (∏σ)(τ)).



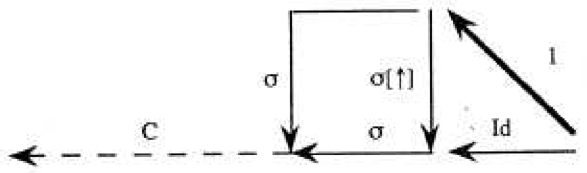
[σClos] σ[s] is interpreted as the pullback of σ along s.



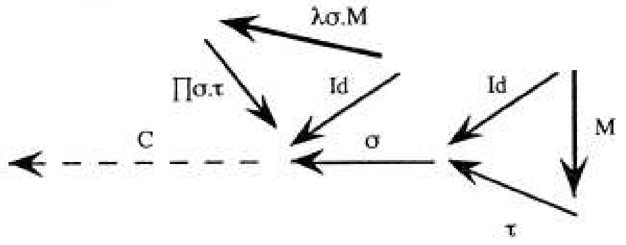
To illustrate the pullback as substitution idea, let us take σ=list(n), and s=succ. The obvious interpretation for list(n+1) is the first projection on {(m,l)| l∈list(m+1)}; the latter set is in one-to-one correspondence with {(m,(n,l)| (l∈list(n) and) n=m+1}, which is the pullback of succ and (n,l)↦n.

We turn to the interpretation of terms.

[Var] The meaning of 1 is the mediating arrow of id and id relative to the pullback of σ and σ. See the case ↑ for a justification of the arrow named σ[↑] on the picture.



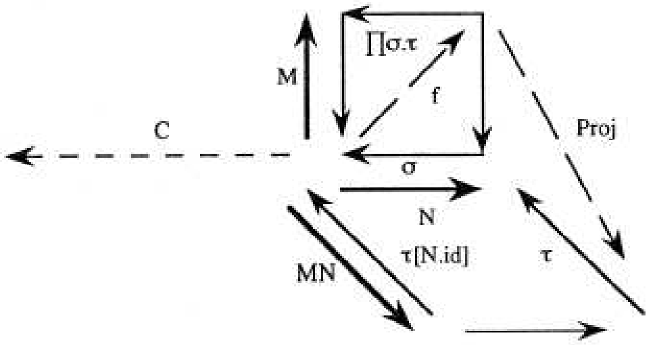
[Abs] We write λ for the natural bijection associated with the adjunction k* ⊣ ∏k, and we use that σ*(Id)=Id.



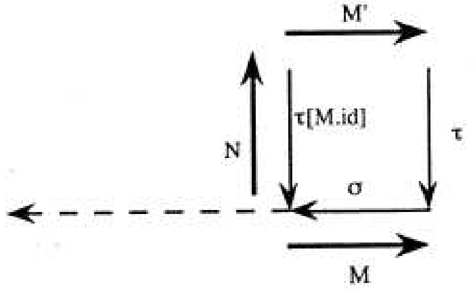
[App] This is the most complex picture in the translation:

- the intermediate arrow f is the mediating arrow for M and N relative to the pullback square 1;
- Proj is the counity of the adjunction k* ⊣ ∏k;
- MN is obtained as the mediating arrow of Proj ∘ f and id relative to the pullback square 2.

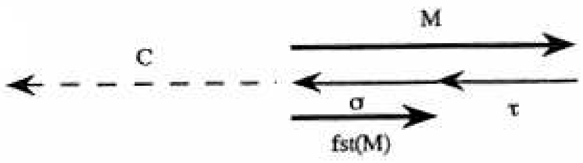
We anticipate the description of the meaning of [Cons] which justifies the identification made between N.id and N, viewed as a global triangle C → C,σ.



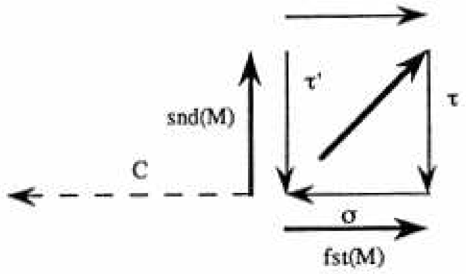
[Pair] Again M.id is identified with M as a global triangle C→C,σ. (M,N) is the composed arrow M' ∘ N, where M' is the last side of the pullback square of M and τ.



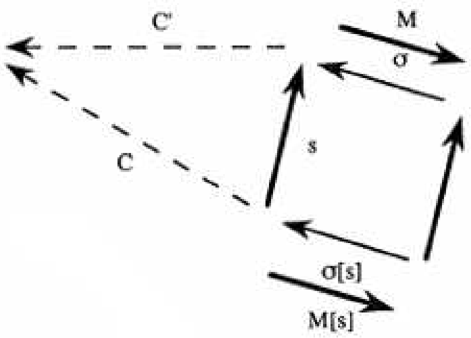
[fst] fst(M) is obtained as $\tau \circ M$.



[snd] snd(M) is obtained as the mediating arrow of M and id along the pullback square of τ and fst(M). On the figure, τ' is τ[fst(M).id].

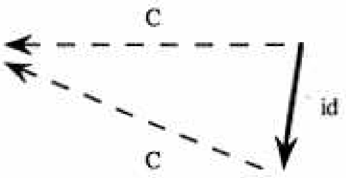


[MClos] The picture for M[s] is obtained by further decorating the picture for σ[s]. Specifically, M[s] is obtained as the mediating arrow of id and $M \circ s$ with respect to the pullback square of σ and s.

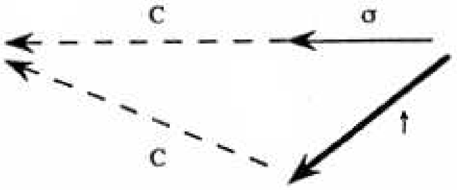


Finally we interpret substitutions.

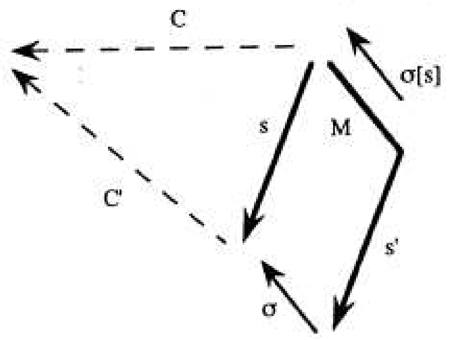
[id]



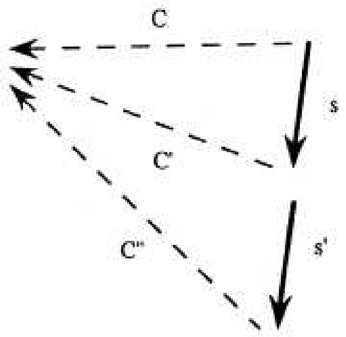
[↑] The interpretation of $C,\sigma \vdash \uparrow : C$ is σ, viewed as a global triangle $C,\sigma \rightarrow C$:



[Cons] The meaning of M.s is s'∘ M in the following picture:



[Comp]



3 The coherence problem

We first describe the equations of the namefree syntax (Section 3.1). We show the validity of one of the simplest equations (Section 3.2), and arrive on the way to the coherence problem. We are lead to add more typing rules (Section 3.3), and to design an intermediate syntax where the proofs of type equalities are recorded (Section 3.4). We state the coherence results (Section 3.5) and defer the detailed, pictorial treatment of some typical cases to the Annexes B.1-4.

Warning: For the sake of simplicity, we forget Σ-types in this section. They do not introduce any additional problem.

3.1 The equational theory In the usual syntax with variable names, the rules are β and η. But in the namefree syntax (cf. [Cur1,CouCurMau,ACCL]) β is decomposed in

Beta[3] $(\lambda M)N \rightarrow M[N.id]$

[3]Strictly speaking, the right hand side is not well-typed. The reader may check that we need to replace it with M[N[id].id]. However, we already assumed that Id* is the identity.

followed by the application of a number of rules to actually perform the substitutions Here is the list of rules considered in [ACCL]:

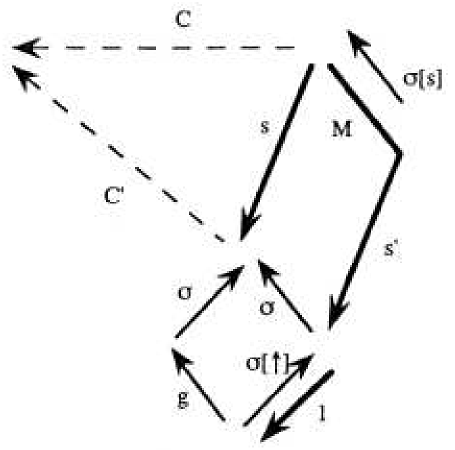
VarId	$1[id] \rightarrow 1$	IdL	$id \circ s \rightarrow s$
VarCons	$1[M.s] \rightarrow M$	ShiftId	$\uparrow \circ id \rightarrow \uparrow$
App	$(MN)[s] \rightarrow M[s])(N[s])$	ShiftCons	$\uparrow \circ [M.s] \rightarrow s$
Abs	$(\lambda\sigma.M)[s] \rightarrow \lambda\sigma.(M[1.s\circ\uparrow])$	Map	$(M.s) \circ t \rightarrow M[t] .(s \circ t)$
Clos	$M[s][t] \rightarrow M[s\circ t]$	Ass	$(s_1 \circ s_2) \circ s_3 \rightarrow s_1 \circ (s_2 \circ s_3)$

In the same way, substitution is distributed in types, giving rise to the following rules:

ΠAbs $(\Pi\sigma.\tau)[s] \rightarrow \Pi\sigma[s].\tau[1.s\circ\uparrow]$ (and similarly ΣAbs)
Pseudo $\sigma[s][t] \rightarrow \sigma[s\circ t]$

(When a dependent constant is eventually reached, the rule $(\kappa(M,N)[s] \rightarrow \kappa(M[s],N[s])$ should be applied.)

<u>3.2 Validity of VarCons</u> (The details of this subsection may be skipped at first reading.) We prove the validity of VarCons, whose intent should be clear: the value of variable 1 is to be found as the top M of the environment M.s. All the actors are drawn on the picture below:



We know that 1[M.s] is defined as a mediating arrow into the pullback square of $\sigma[\uparrow]$ and $s' \circ M$. Standard "categorical handwaving" goes like this: $\sigma[\uparrow]$ is already a pullback of σ along σ. Thus the pullback of $\sigma[\uparrow]$ along $s' \circ M$ is the pullback of σ along

$$\sigma \circ s' \circ M = s \circ \sigma[s] \circ M = s,$$

that is, σ[s], which is expected since the right hand side to be matched is M, which has type σ[s]. Formally, the fourth side of this square is the arrow f into the pullback of σ and σ which mediates $s' \circ M \circ \sigma[s]$ and s', as shown in the next picture:

$$\begin{tikzcd}
 & {} & & & {} \arrow[lll, "f"'] \arrow[dllll, dashed, "s'"'] \arrow[ddl, "\sigma[\uparrow][s'\circ M]"'] \arrow[dd, "\sigma[s]"] \\
{} \arrow[dr, "\sigma"'] & & {} \arrow[dl, "\sigma"] & & \\
 & {} & & {} \arrow[ul, "s'\circ M"'] & {} \arrow[lll, "s"]
\arrow[from=1-2, to=2-1, "g"']
\arrow[from=1-2, to=2-3, "\sigma[\uparrow]"]
\end{tikzcd}$$

All we have to do is to check that M is mediating $1 \circ s' \circ M$ and id relatively to the pullback of $\sigma[\uparrow]$ and $s' \circ M$. This amounts to checking:

- $\sigma[s] \circ M = id$ (by induction, because M has type $\sigma[s]$),
- $f \circ M = 1 \circ s' \circ M$, which is easily checked by composing with g and $\sigma[\uparrow]$ respectively.

3.3 Introducing type equality judgements The proof in Section 3.2 implicitly used the "equation" $\sigma[\uparrow][M.s]$ "=" $\sigma[s]$, where the crucial step is $\sigma[\uparrow][M.s]$ "=" $\sigma[\uparrow \circ (M.s)]$, which is only an isomorphism by the pseudo-functoriality of the pullback.

This point can be seen directly from the syntax, as it stands so far: $1[M.s]$ and M have respective types $\sigma[\uparrow][M.s]$ and $\sigma[s]$. If we want to include the rule formally in the syntactic theory, we need to introduce type equality judgements $C \vdash \sigma=\tau$ type, together with the reflexivity, symmetry and transitivity (or *cut*) rule (whose obvious description we omit), and to add the rule:

[EqType]

$$\frac{C \vdash M{:}\sigma \qquad C \vdash \sigma=\tau \text{ type}}{C \vdash M{:}\tau}$$

With the help of this new rule, we can derive the type $\sigma[s]$ for both $1[M.s]$ and M. The addition of the seemingly innocent rule EqType raises a coherence problem: now the same judgement can be proved well-formed in different ways. For example when typing $\lambda\sigma.M$, we may first show that M has type τ, then that the same M has type τ_1 for some τ_1 by EqType. The typing would then proceed with Abs, yielding type $\Pi\sigma.\tau_1$ for $\lambda\sigma.M$. But we could also apply Abs right after the derivation of type τ for M, getting $\lambda\sigma.M{:}\ \Pi\sigma.\tau$, and then apply EqType to obtain $\lambda\sigma.M{:}\ \Pi\sigma.\tau_1$. On the way we have used a congruence rule:

[ΠCong]

$$\frac{C \vdash \sigma=\sigma' \text{ type} \qquad C,\sigma \vdash \tau=\tau' \text{ type}}{C \vdash \Pi\sigma.\tau = \Pi\sigma'.\tau' \text{ type}}$$

This rule presents a mismatch. One expects (as a metatheorem on syntax) that if $C \vdash \sigma_1=\sigma_2$ type is derivable, then also $C \vdash \sigma_1$ type and $C \vdash \sigma_2$ type are derivable. But from the hypotheses of ΠCong we get $C,\sigma \vdash \tau'$ type, and not $C,\sigma' \vdash \tau'$ type, which we need in order to show $C \vdash \Pi\sigma'.\tau'$ type. This suggests to complete our extension of syntax by yet another kind of judgement: $\vdash C=C'$ context, and to add a "contravariant" counterpart of EqType (we state one for type judgements, but there should be one for each kind of judgement):

[CongCont]

$$\frac{C \vdash \sigma=\sigma' \text{ type} \qquad \vdash C=C' \text{ context}}{\vdash C,\sigma = C',\sigma' \text{ context}}$$

[EqCont]

$$\frac{C \vdash \sigma \text{ type} \qquad \vdash C=C' \text{ context}}{C' \vdash \sigma \text{ type}}$$

Finally, there is another congruence rule, which creates type equalities from substitution equalities:

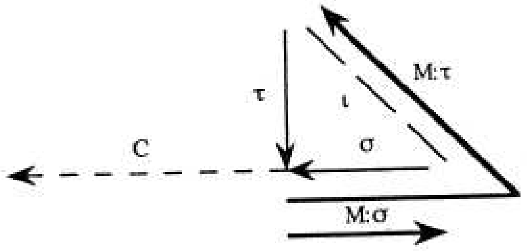
[ClosCong]

$$\frac{C \vdash s=t:C' \qquad C' \vdash \sigma \text{ type}}{C \vdash \sigma[s]=\sigma[t] \text{ type}}$$

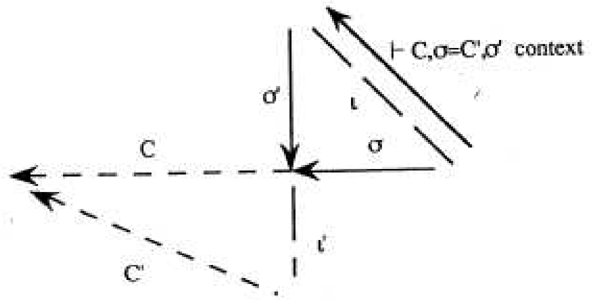
we

We extend our pictorial semantics in the following way. We interpret judgements $C \vdash \sigma=\tau$ type by an isomorphism ι between (the domains of) σ and τ, and similarly we interpret judgements $\vdash C=C'$ context by an isomorphism ι between (the domains of) C and C'.

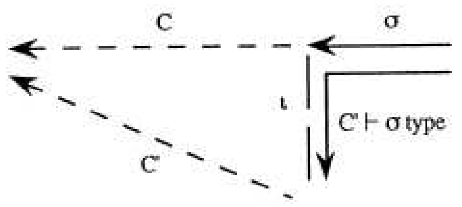
[EqType] If M abbreviates the meaning of (the proof of) $C \vdash M:\sigma$, then the meaning of $C \vdash M:\tau$ is M composed with ι.



[CongCont] The iso $C,\sigma \rightarrow C',\sigma'$ is exactly the iso $\sigma\rightarrow\sigma'$.



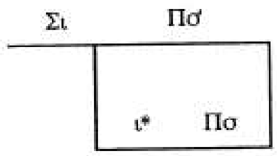
[EqCont] This is dual to EqType.



We decompose the congruence in two cases.

[ΠCong,σ fixed] The meaning of $C \vdash \Pi\sigma.\tau = \Pi\sigma.\tau'$ type is $\Pi_\sigma(\iota)$ if ι interprets $\tau=\tau'$.

[ΠCong,τ fixed] The meaning of $C \vdash \Pi\sigma.\tau = \Pi\sigma'.\tau$ type is the inverse (notice the *contravariance)* of the instance at τ of the following natural transformation, where the 2-cell is the canonical Π transformation associated with the pullback square $\sigma' \circ \iota = \mathrm{Id} \circ \sigma$ (ι is as in the figure for CongCont). Notice that since we have chosen ι^{-1*} as $\Sigma\iota$, the picture describes a transformation $\Pi\sigma' \circ \Sigma\iota \rightarrow \Pi\sigma$.



[ClosCong] This rule does not create isomorphic types, but equal ones: equal substitutions s and t (and terms) of the same type are interpreted by equal arrows (see Section 3.4), thus $\sigma[s]$ and $\sigma[t]$ coincide, being both the chosen pullback of the same pair of arrows.

3.4 Explicit syntax The point of coherence is that we have to show that the meaning of a term or of a type equality does not depend on its proofs of well typing. To establish this result, we introduce an intermediate language where we keep an explicit track of the whole proofs of well-typings. The following case is added to the syntax of terms:

$$M ::= c_{\sigma,\tau}\langle M\rangle,$$

and EqType is replaced in this explicit language by

[CoerceType] $$\frac{C \vdash M:\sigma \qquad C \vdash \sigma=\tau \text{ type}}{C \vdash c_{\sigma,\tau}\langle M\rangle:\tau}$$

The equational theory of Section 3.1 has to be revised, in order to be able to map both sides of each equality to strictly equal meanings. For example:

VarCons $c_{\sigma[\uparrow][M.s],\sigma[s]}<1[M.s]> \rightarrow M$

App $c_1<(MN)[s]> \rightarrow (c_2<M[s]>)(N[s)$ where

$c_1 = c_{\tau[N.id][s],\tau[1.s\circ\uparrow][N[s].id]}$ and $c_2 = c_{(\prod\sigma.\tau)[s],\prod\sigma[s].\tau[1.\sigma\circ\uparrow]}$.

We call these new equations the *explicit versions* of the rules of Section 3.1. Strictly speaking, we should be even more explicit, and develop a language for describing *coercions*, that is, the various proofs of, say $\sigma=\tau$. Such a completely explicit syntax, where each syntactic construct has at most one proof of well-typing, may be found, for a different calculus, in [CuGhe]. We should also have included in the syntax traces of the uses of EqCont ("$c_{C,C'}<\sigma>$"). We refrain here to do so, to avoid a heavy notational apparatus, but some of the discussion below assumes that we work in such a completely explicit syntax.

3.5 Coherence The coherence results can be now stated and proved. Some lemmas are needed on the way.

Lemma 4: The explicit version of the equations Beta, VarId, VarCons, App, Abs, Clos, and Ass are valid (up to equality) in all LCCC's.
Proof: We proved this for VarCons using standard categorical reasoning. We detail another case (App) in Annex B.1, with a graphical proof technique.

Remark: We should stress here the power of the Beck-Chevalley condition. When dependent types are interpreted in split fibrations (that is when the substitution functor is truly functorial), one has to require *both* that ΠAbs is interpreted up to equality, and that the counities of the $^*\Pi$-adjunctions are preserved on the nose by substitution (cf. [CoqEhr]), whereas this second condition comes for free in the general non split setting. The point is that in split fibrations the power of the formulation "the canonical transformation ... is iso" is lost.

Lemma 5: The rewriting system defined on closed types by ΠAbs+ΣAbs+Pseudo is confluent and strongly terminating.
Proof: We refer to Annex B.3 for the only critical pair (there is also one for Σ). Termination is proved by a straightforward recursive path ordering, or a polynomial argument.

Theorem 6: For any C, σ, and τ, any two proofs of $C \vdash \sigma=\tau$ type receive equal interpretations in any LCCC.
Proof: The proof follows essentially the same line as the proof of coherence of, say

monoidal categories. We first transform any proof into a proof where cuts occur only at the end (notice that this is converse to usual proof-theoretic cut elimination). The crucial part of the justification that these transformations leave the meaning unchanged is described in Annex B.2, where the two proofs

$$\dfrac{\dfrac{C \vdash \sigma=\sigma' \text{ type} \quad C \vdash \sigma'=\sigma'' \text{ type}}{C \vdash \sigma=\sigma'' \text{ type}} \qquad C,\sigma \vdash \tau \text{ type}}{C \vdash \Pi\sigma.\tau=\Pi\sigma''.\tau \text{ type}}$$

and

$$\dfrac{\dfrac{C \vdash \sigma=\sigma' \text{ type}}{C \vdash \Pi\sigma.\tau=\Pi\sigma'.\tau \text{ type}} \qquad \dfrac{C \vdash \sigma'=\sigma'' \text{ type}}{C \vdash \Pi\sigma'.\tau=\Pi\sigma''.\tau \text{ type}}}{C \vdash \Pi\sigma.\tau=\Pi\sigma''.\tau \text{ type}}$$

are shown equal in all interpretations[4].

We further restrict our attention to the proofs where not only the cuts occur at the end, but also the types connected by the sequence of cuts form a rewriting sequence (according to the rules ΠAbs,Pseudo and all the rules on terms and substitutions, accessed through ClosCong). These proofs are in one-to-one correspondence with rewriting sequences of types. The coherence proof then follows exactly the Knuth-Bendix completion procedure [Hue]: one needs to consider the critical pairs, to complete them, and to check that the proofs corresponding to the two paths of the local confluence diagram receive the same meaning. Since the rule ClosCong is interpreted up to equality, we don't have to care for the critical pairs arising from rules on terms and substitutions[5]. We are left with just one critical pair, between ΠAbs and Pseudo. The analysis of this critical pair is presented in Annex B.3.

Finally we use Lemma 5 to extend the coherence result to all proofs where the cuts occur at the end, and corresponding to zigzags between the connected types. One shows by induction on the length of the zigzag that the iso pasted along the zigzag, composed with the iso pasted along any path from the end point of the zigzag to its normal form, is equal to the iso pasted along any path from the start point of the zigzag to its normal form.

[4]We already noticed in 3.3 that, by contravariance of Π in its first argument, an iso $\sigma\rightarrow\sigma'$ determines an iso $\Pi\sigma'.\tau\rightarrow\Pi\sigma.\tau$. Thus to describe the natural transformations "witnessing" rewritings of arbitrary sub types (and subterms), we may need to reverse the directions of arrows: in particular Beck-Chevalley condition (which states that a canonical arrow is iso) is essential.

[5]To make this argument completely precise, we should state and prove a postponment lemma, saying that the rewriting of terms and substitutions appearing as subterms of types can all be done at the end.

Before stating coherence at the level of terms, we need one more lemma.

Lemma 7 If $\Pi\sigma.\tau$ and $\Pi\sigma'.\tau'$ are provably equal, then so are σ and σ', τ and τ', respectively.
Proof: Observe that when rewriting both $\Pi\sigma.\tau$ and $\Pi\sigma'.\tau'$ to their common normal form $\Pi\sigma''.\tau''$, the head form $\Pi\_.\_$ is untouched. It follows easily that $\sigma\rightarrow^*\sigma''$ and $\sigma'\rightarrow^*\sigma''$, and similarly for τ, τ', τ''.

Theorem 8: For any C and M, if $C\vdash M{:}\sigma$ and $C\vdash M{:}\sigma'$ are provable, then $\sigma=\sigma'$ is provable. Moreover, for any C, M, and σ, any two proofs of $C\vdash M{:}\sigma$ receive equal interpretations in any LCCC (and similarly for C, s, C').
Proof: By induction on the sum of the lengths of these proofs, and by cases on the shape of M. We have to generalize the statement to take care of the possible uses of EqType and EqCont. What we prove is actually:

- For any C, M, and C', if $C\vdash M{:}\sigma$, $C'\vdash M{:}\sigma'$ and $\vdash C=C'$ are provable, then $\sigma=\sigma'$ is provable.

- For any C, M, σ, C', and σ' such that $\vdash C,\sigma = C',\sigma'$ is provable, any two proofs of $C\vdash M{:}\sigma$ and $C'\vdash M{:}\sigma'$, respectively, receive interpretations which become equal when composed with the isos $C\leftrightarrow C'$ and $\sigma\leftrightarrow\sigma'$, appropriately oriented (and similarly for C, s, C').

The case for application essentially amounts to check that the two proofs (cf. the rule App' of [CuGhe]) (we keep τ fixed, and omit contexts, for simplicity)

$$\dfrac{\dfrac{M{:}\ \Pi\sigma.\tau \qquad \dfrac{\sigma=\sigma'}{\Pi\sigma.\tau=\Pi\sigma'.\tau}}{M{:}\ \Pi\sigma'.\tau} \qquad N{:}\sigma'}{MN{:}\tau[N.id]}$$

and

$$\dfrac{M{:}\ \Pi\sigma.\tau \qquad \dfrac{N{:}\sigma' \qquad \dfrac{\sigma=\sigma'}{\sigma'=\sigma}}{N{:}\sigma}}{MN{:}\tau[N.id]}$$

are equal in all interpretations. This is shown in Annex B.4 . Notice that, hidden behind this equality of proofs, is the following other equality of proofs:

$$\tau[c_{\sigma',\sigma}\langle N\rangle.id] = c_{(C,\sigma),(C,\sigma')}\langle\tau\rangle[N.id]\ .$$

Similarly, for the case of abstraction one has to check the equality of the proofs given at the beginning of Section 3.4 to illustrate coherence of proofs (this corresponds to the rule λ of [CuGhe]).

4. Conclusions and future work

Beyond the technical coherence result, we believe that the graphical representations of proofs whith which we learned to play while writing the paper provide an interesting geometric view of syntax. It urges the need for software tools to automate the construction and manipulation of such drawings.

We would like to extend the results to more complicated calculi, like the calculus of constructions. Without going so far, the treatment of constants should be made more precise. The resistance of the coherence results to the introduction of equality types of various flavours should be investigated too.

Acknowledgements Gérard Huet first pointed to the author the connection between coherence and Knuth-Bendix. I am grateful to the Departamento de Algebra of the university of Santiago de Compostela for introducing Rodeja carpets to me. I wish also to thank my coauthors of [ACCL] and [CuGhe], on which the present paper is built up. I benefited from interesting discussions with T. Coquand, Y. Lafont and A. Scedrov.

References:

[ACCL] M. Abadi, L. Cardelli, P.-L. Curien, J.-J. Lévy, Explicit Substitutions, in Proc.ACM Principles of Programming Languages, San Francisco (1990), long version in Journal of Functional Programing 3 (1992).

[Brui] N. De Bruijn, Lambda-calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, Indag Math. 34 (1972).

[Bur] A. Burroni, Higher Dimensional Word Problem, in Proc. Category Theory and Computer Science 91, Paris, Lecture Notes in Computer Science 530, 94-105 (1991), long version to appear in Theoretical Computer Science.

[Cart] J. Cartmell, Generalized Algebraic Theories and Contextual Categories, Thesis, Oxford (1978).

[CoqEhr] T. Coquand, T. Ehrhard, An Equational Presentation of Higher-Order Logic, Proc. 2nd Conf. on Category Theory and Computer Science, Lecture Notes in Comput. Sci. 283 (1987).

[CouCurMau] G. Cousineau, P.-L. Curien, M. Mauny, The Categorical Abstract Machine, Science of Computer Programming 8, pp. 173-202 (1987).

[CuGhe] P.-L. Curien, G. Ghelli, Coherence of Subsumption, Mathematical Structures in Computer Science, 2(1), 1992, pp.55-91.

[Cur1] P.-L. Curien, Categorical Combinators, Sequential Algorithms and Functional Programming, Pitman (1986). Second, revised edition, Birkhaüser, to appear (1993).

[Cur2] P.-L. Curien, α-conversion, Conditions on Variables and Categorical Logic, Studia Logica XLVIII (3), 319-360 (1990).

[CurHarRio] P.-L. Curien, T. Hardin, A. Rios, Normalisation of substitutions, Proc. Mathematical Foundations of Computer Science, Prague, Lecture Notes in Computer Science (1992).

[Dyb] P. Dybjer, Inductive Sets and Families in Martin-Löf Type Theory and their Set-theoretic Semantics, in Logical Framework, G. Plotkin and G. Huet eds, Cambridge University Press (1992).

Science 88, Edinburgh (1988).

[Fre] P. Freyd, Aspects of Topoi, Bull. Austral. Math. Soc. 7 (1972).

[Guit] R. Guitart, Relations et Carrés Exacts, Annales Sci. Math. du Québec, Vol. IV , No2, pp. 103-125 (1980).

[HarLa] T. Hardin, A. Laville, Proof of Termination of the Rewriting System Subst on CCL, Theoret. Comput. Sci. 46, pp. 305-312 (1986).

[HueOp] G. Huet, D. Oppen, Equations and Rewrite Rules: a Survey, in Formal Languages, Perspectives and Open Problems, R. Book ed., Academic Press (1980).

[HyPit] M. Hyland, A. Pitts, The Theory of Constructions: Categorical Semantics and Topos-theoretic Models, in J. Gray and A. Scedrov (eds), Proc. Categories in Computer Science and Logic, Boulder 1987, Contemporary Mathematics 92 (1989).

[Jac] B. Jacobs, Categorical Type Theory, PhD Thesis, University of Nijmegen, (1991).

[JoyStre1] A. Joyal, R. Street, Braided tensor categories, draft.

[JoyStre2] A. Joyal, R. Street, the Geometry of Tensor Calculus I, draft.

[Laf] Y. Lafont, Penrose diagrams and 2-dimensional rewriting, in Applications of Categories in Computer Science, Cambridge University Press, 191-202 (1992).

[Lam] F. Lamarche, A Simple Model of the Theory of Constructions, cf. [HyPit].

[Law] W. Lawvere, Adjointness in Foundations, Dialectica 23(3/4) (1969).

[MacLaPar] S. Mac Lane, R. Paré, Coherence for Bicategories and Indexed Categories, Journal of Pure and Applied Logic 37, 59-80 (1985).

[Mar] P. Martin-Löf, Intuitionistic Type Theory, Bibliopolis (1984).

[MeyRein] A. Meyer, M. Reinhold, 'Type' is not a Type, Proc. Logic in Computer Science 86.

[Ob] A. Obtulowicz, Categorical and Algebraic Aspects of Martin-Löf Type Theory, Studia Logica XLVIII (3), 319-360 (1990).

[PalmStol] E. Palmgren, V. Stoltenberg-Hansen, Domain Interpretations of Martin-Löf's Type Theory, Annals of Pure and Applied Logic 48, 135-196 (1990).

[Pav] D. Pavlovic, Predicates and Fibrations, Phd Thesis, Utrecht University (1990).

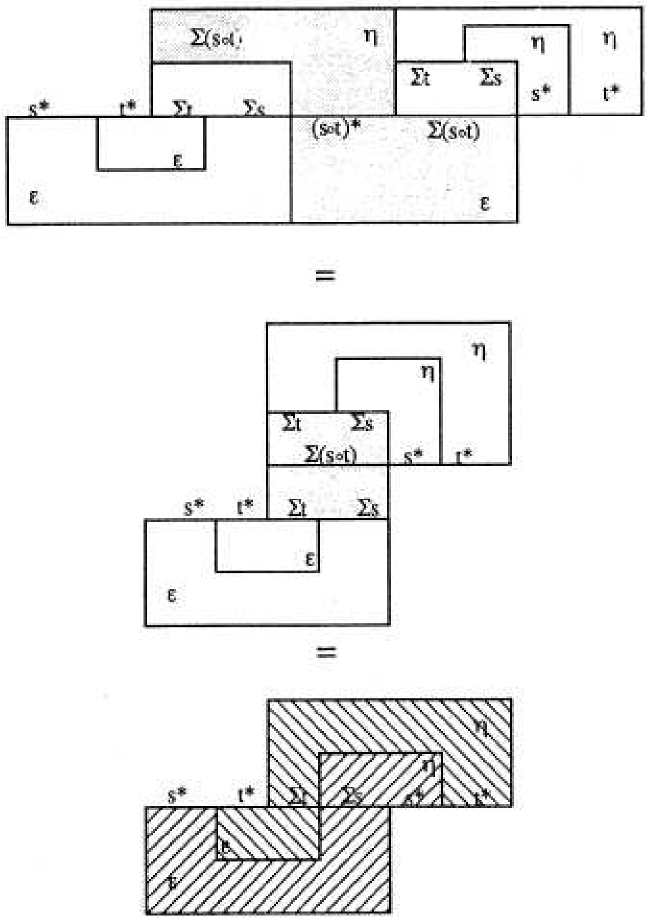
[Pow] A. Power, A 2-categorical Pasting Theorem, Journal of Algebra, to appear.

[Se] R. Seely, Locally cartesian closed categories and type theory, Math. Proc. Camb. Phil. Soc. 95 (1984).

[Stre] T. Streicher, Correctness and Completeness of a Semantics of the Calculus of Constructions with Respect to Interpretation in Doctrines of Constructions, Thesis, Universität Passau (1988), Birkhaüser (1992).

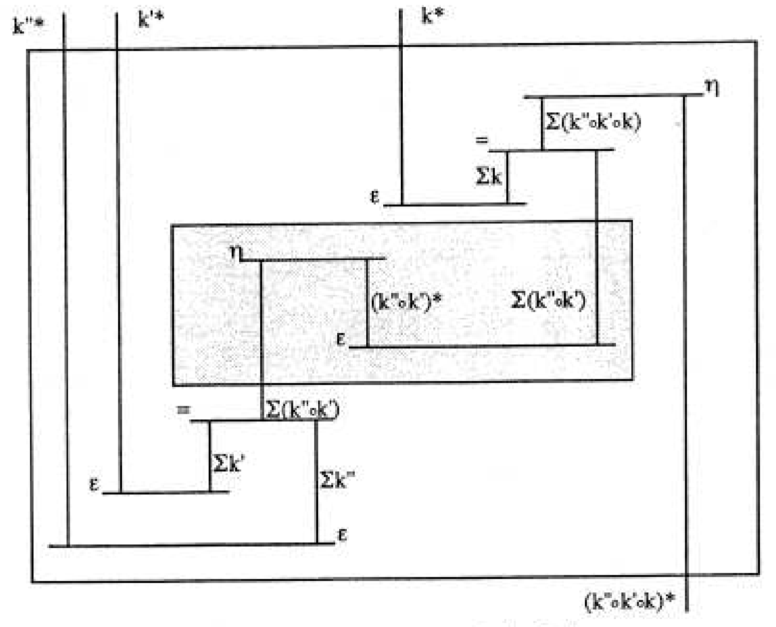
[Tay] P. Taylor, Recursive Domains, Indexed Category Theory and Polymorphism, PhD Thesis, University of Cambridge (1986).

Annex A.1 *(Pseudo-functoriality of pullback).* Here is a hopefully self-explanatory rewriting sequence of pastings establishing one direction of the proof that the transformations $t^* \circ s^* \leftrightarrow (s \circ t)^*$ exhibited in Figures 3 and 4 are inverse. Redexes are highlighted (the graphical definition of adjunction is by $\eta\varepsilon$-cancellation). At the end, after the two last cancellations (which can be performed in parallel), we arrive at the identity $s^* \circ t^* \rightarrow s^* \circ t^*$.

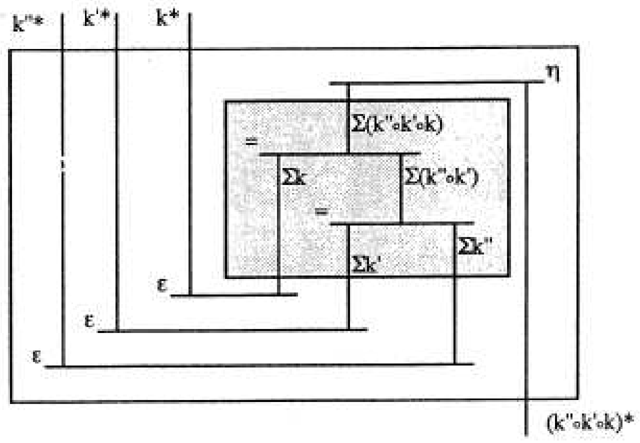


Annex A.2 *(Coherence of pullbacks).* The two pastings $k^* \circ k'^* \circ k''^* \to (k'' \circ k' \circ k)^*$ are equal. Following a suggestion of Lafont, we use the "dual" notation of Penrose diagrams, which is more convenient to handle. Natural transformations are now points (actually horizontal lines). The arcs are still functors, but now they connect the natural transformations previously pasted on both sides, and the regions (not named in the drawings below) are now the previous points, that is, categories. Redexes are isolated by appropriately stretching a "dual pasting diagram", both horizontally and vertically. The redisplay after reduction is straightforward with this representation. In the presentation by pastings, after an $\eta\varepsilon$-cancellation has occurred, it is not always immediate to see how to redisplay the neigbouring arcs and regions after the reduction.

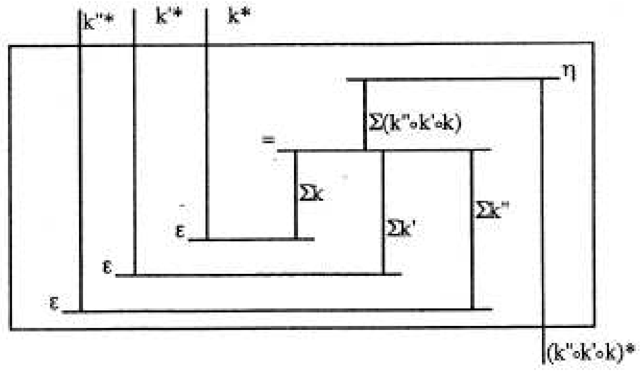
The Penrose diagram corresponding to the left-hand side $\psi_{k,k''\circ k'} \cdot \psi_{k',k''}k$ of the coherence equation is:



By performing the highlighted reductions, we successively obtain:



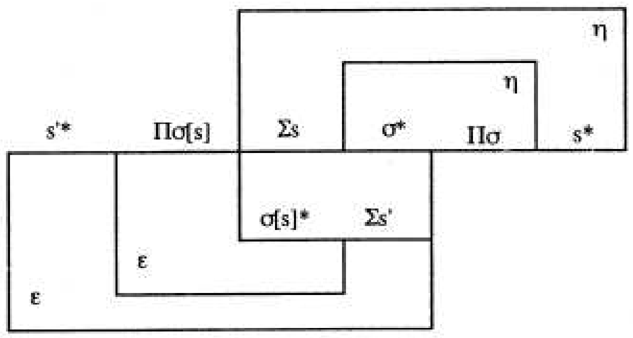
and



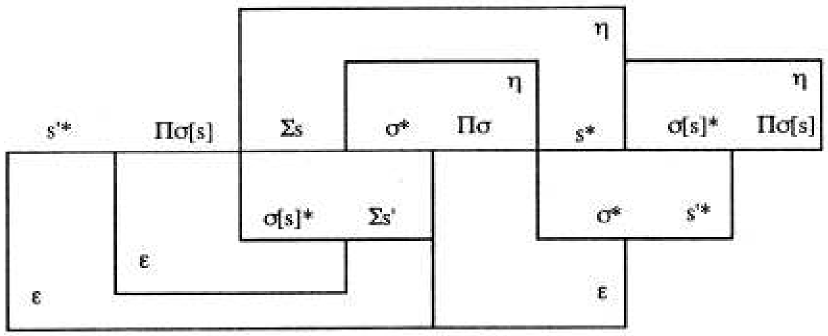
And similarly for the right hand side $k''\psi_{k,k'} \cdot \psi_{k'\circ k,k''}$.

We shall use Penrose diagrams in Annexes B.1, B.2, and B.4.

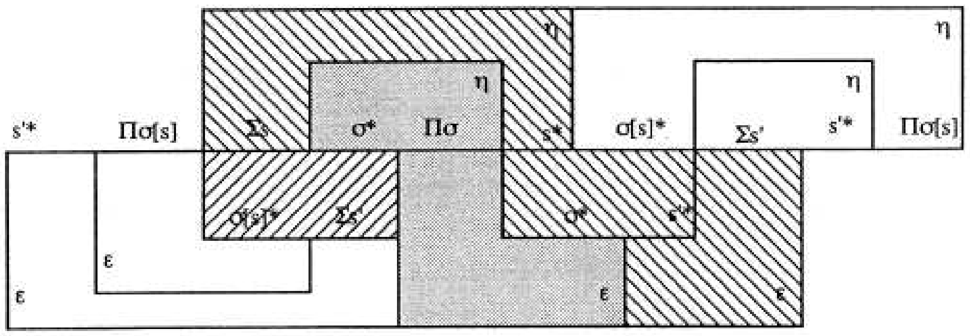
Annex A.3 *(Pictorial proof of Lemma 3).* First we define as candidate for the inverse of the canonical Π transformation ϕ (Figure 6) the following transformation χ, built with the help of the inverse of the canonical Σ transformation.



As a first step towards showing that $\phi \circ \chi = \mathrm{Id}$, we paste the drawings together.

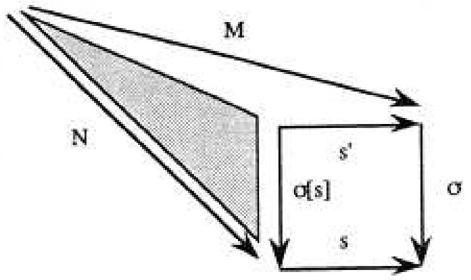


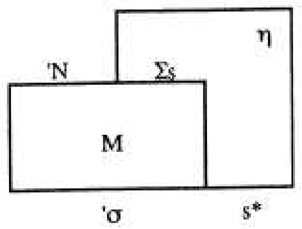
There is an encouraging central $\eta\varepsilon$-cancellation. But we also have to make the canonical transformation $\Sigma s' \circ \sigma[s]^* \rightarrow \sigma^* \circ \Sigma s$ explicit. For this purpose we $\eta\varepsilon$-expand the down right s'*:



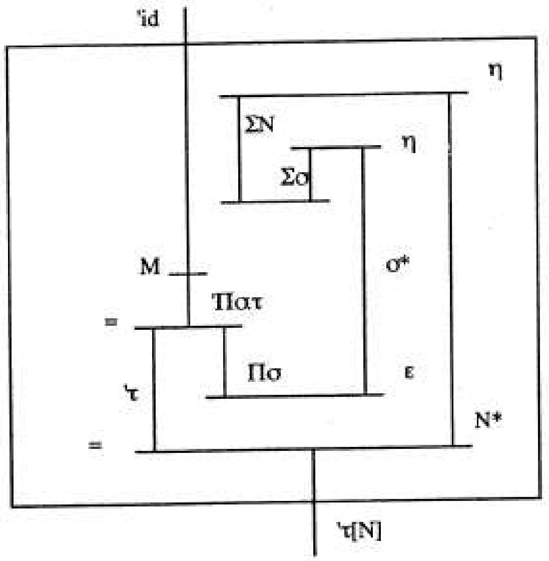
After removing the two areas which are highlighted, the inner (outer) η and ε come in contact, so that all the figure gets squeezed to the identity.

Annex B.1 *(Validation of App)* (Lemma 4). In order to validate the rules of the explicit calculus, we have to reexpress the semantic interpretation in 2-categorical terms. If k is an arrow into A, we denote by 'k the constant functor with value k from the terminal category **1** to the slice **C**/A. We have then $(\Sigma l) \circ ('k) = '(l \circ k)$ and $g^* \circ ('k) = 'k[g]$, for l and g of appropriate types. We represent an arrow f: k→l of **C**/A as a natural transformation from 'k to 'l. The main tool used in Section 2.4 for the description of the interpretation is the notion of mediating arrow. Here is a pictorial representation of the mediating arrow of M and N with respect to the pullback diagram of s and σ, considered as an arrow N→σ[s]:

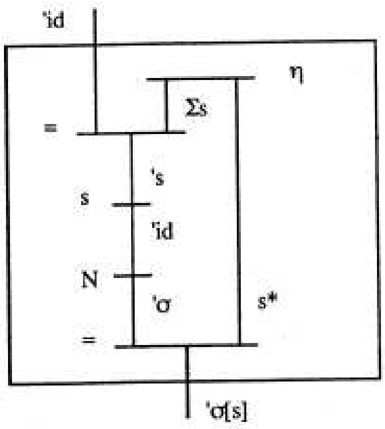




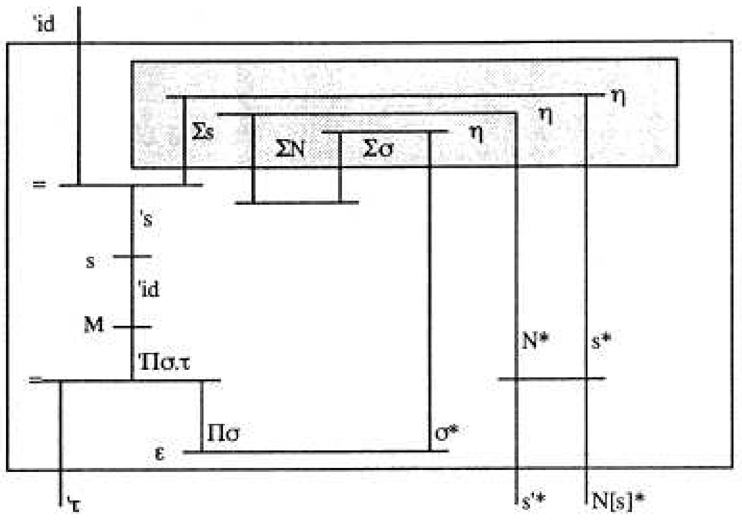
Now we can express MN as a natural transformation. In the following Penrose diagram, the upper and lower η correspond to the two successive mediating arrows used in the construction of the interpretation of MN in Section 2.4.:



and similarly for the meaning of N[s]:

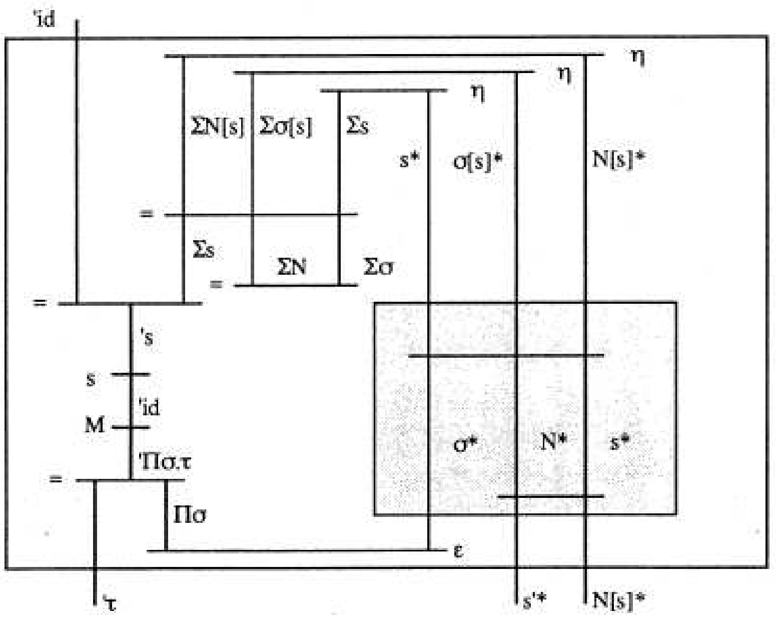


Now we are able to draw c_1<(MN)[s]>, and to rewrite it.

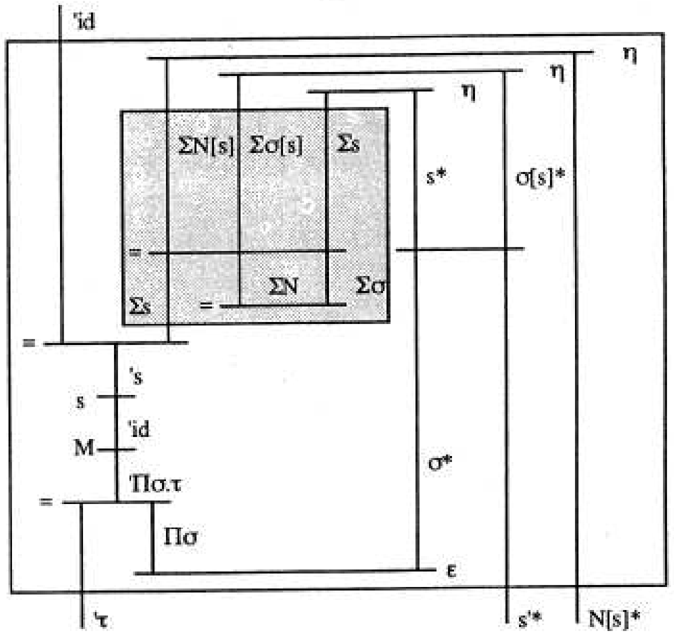


(The redex is a variant of Figure 7, see Annex B.2.)

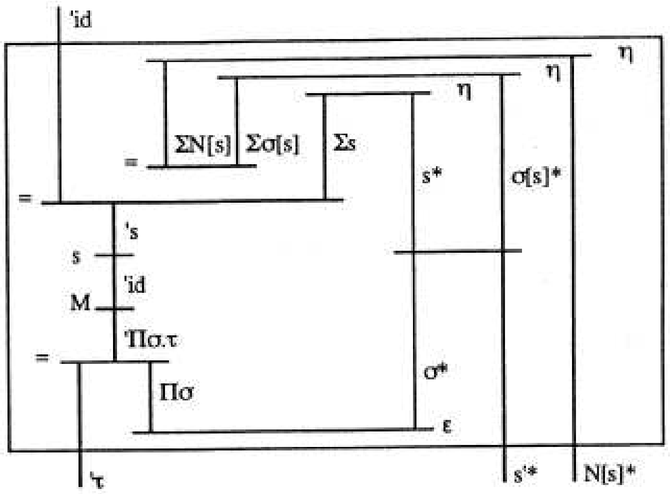
=



=

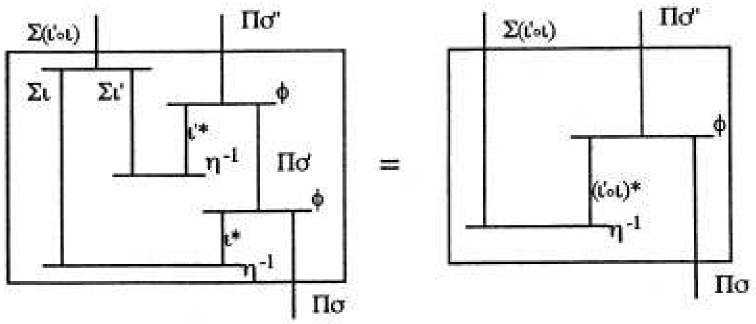


=



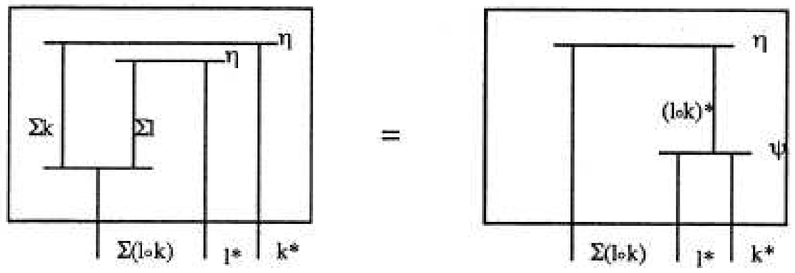
We leave the reader check that the drawing obtained by ηε-expanding the vertical σ[s]* line is (c_2<M[s]>)(N[s]) .

Annex B.2 *(Contravariant congruence of type equality)* (Theorem 6). Denoting by ϕ the canonical Π transformations of Figure 6, as in Annex A.3, we want to show

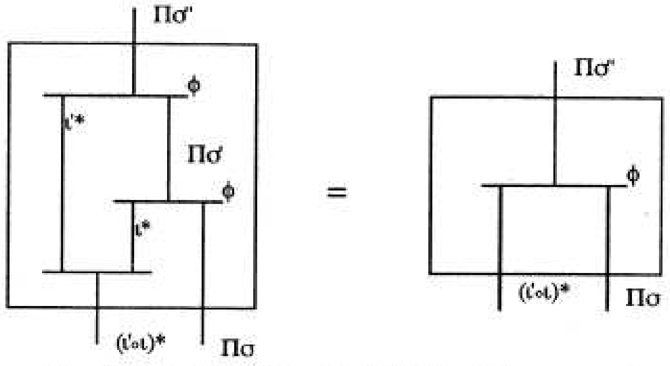


We use the following equation, which is easily derivable from the definition of ψ (Figure 4).

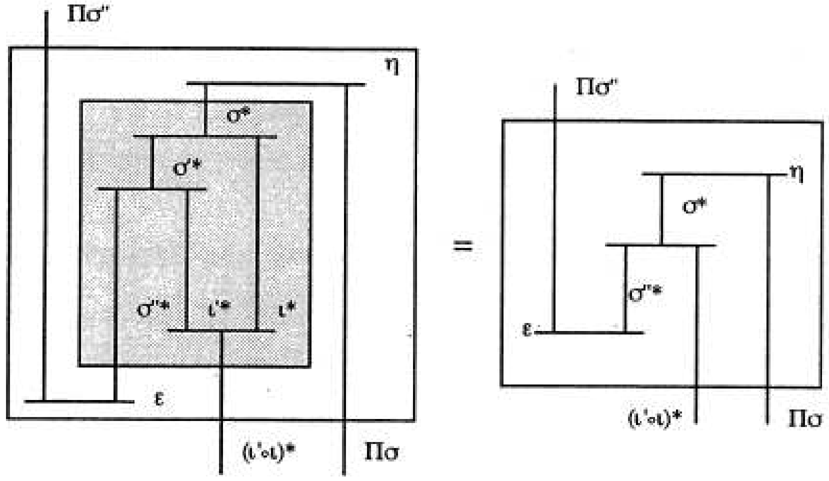
Figure 7



We are reduced to show:

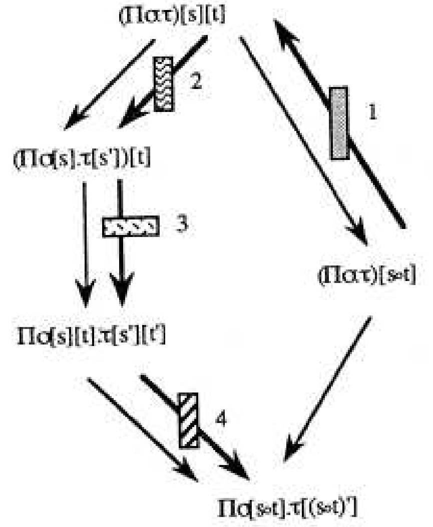


Now, expanding in the left hand side the definition of ϕ, we get, after an $\eta\varepsilon$-cancellation:

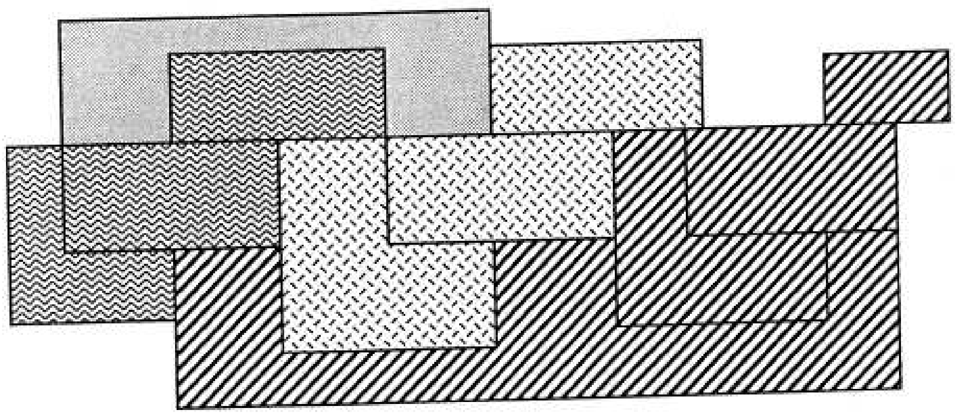


which is the expansion of the right hand side.

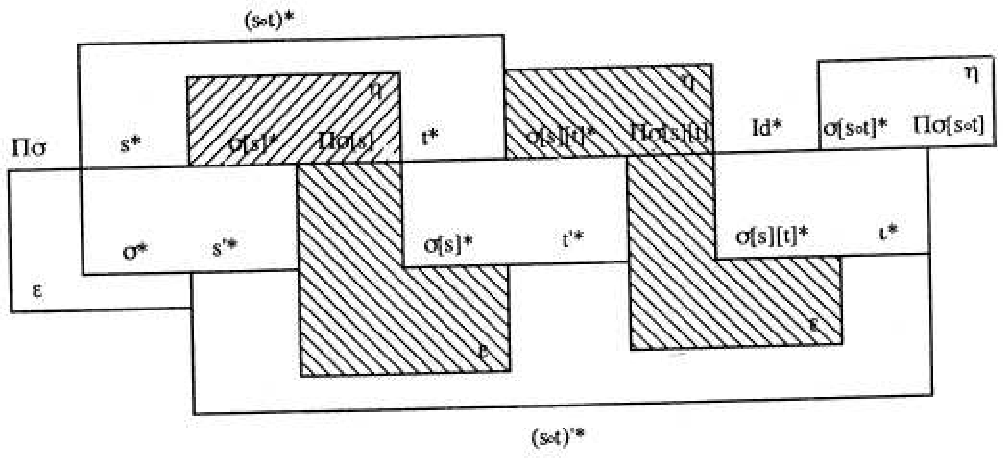
Annex B.3 *(The critical pair ΠAbs-Pseudo)* (Theorem 6). The notation is taken from Figure 2. We first describe the completion of the critical pair, then we build and paste the isomorphisms along one path (we have reversed one direction for convenience). Finally we perform pasting rewritingss until we reach the iso corresponding to the other path $(\Pi\sigma.\tau)[s\circ t] \rightarrow \Pi\sigma[s\circ t].\tau[(s\circ t)']$.



(Πστ)[s][t]
(Πσ[s].τ[s'])[t]
Πσ[s][t].τ[s'][t']
(Πστ)[s∘t]
Πσ[s∘t].τ[(s∘t)']
1
2
3
4

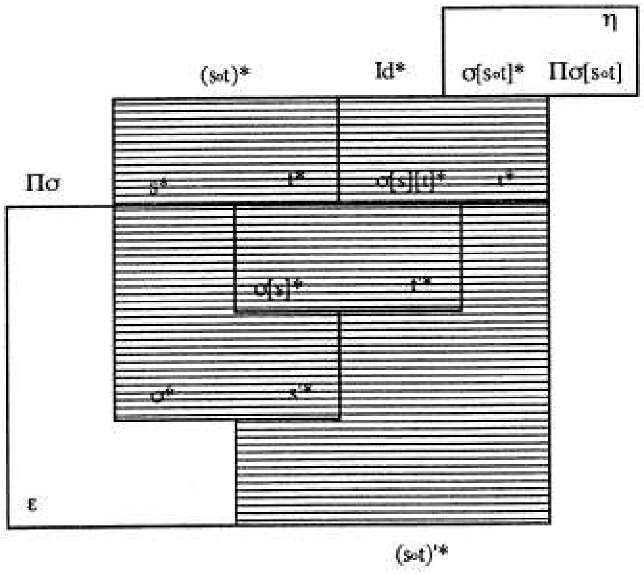


=



(s∘t)*
η
η
η
Πσ
s*
σ[s]*
Πσ[s]
t*
σ[s][t]*
Πσ[s][t]
Id*
σ[s∘t]*
Πσ[s∘t]
σ*
s'*
σ[s]*
t'*
σ[s][t]*
t*
ε
ε
ε
(s∘t)'*

=



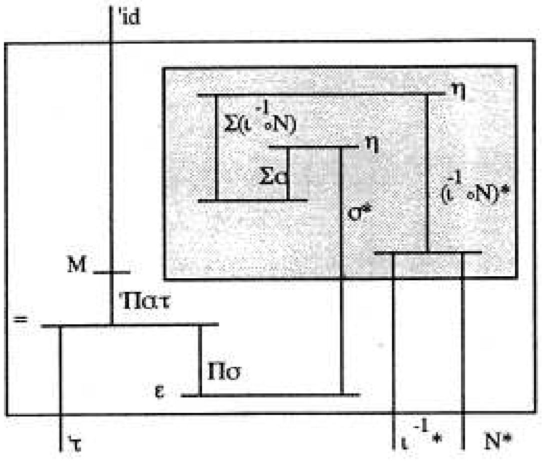
Annex B.4 *(Permutation of application and type equality)* We check the equality of the interpretations of $(c_1{<}M{>})N$ and of $M(c_2{<}N{>})$, where

- c_3 proves $\sigma=\sigma'$,
- c_2 is c_3 followed by symmetry, and
- c_1 is c_3 followed by ΠCong (τ fixed).

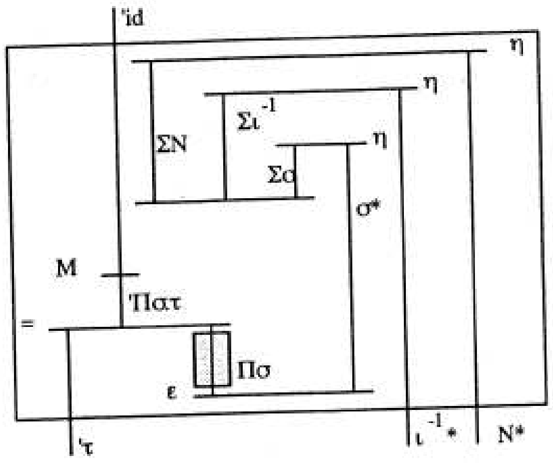
The natural iso $(\iota^{-1}\circ N)^* \leftrightarrow N^* \circ \iota^{-1*}$, which we include in the Penrose diagram representing $M(c_2{<}N{>})$, accounts for the equality

$$\tau[c_{\sigma',\sigma}{<}N{>}.\mathrm{id}] = c_{(C,\sigma),(C,\sigma')}{<}\tau{>}[N.\mathrm{id}]$$

("EqType, then Cons, then σClos = EqCont and Cons, then σClos"). We rewrite this drawing until we get the representation of $(c_1{<}M{>})N$:

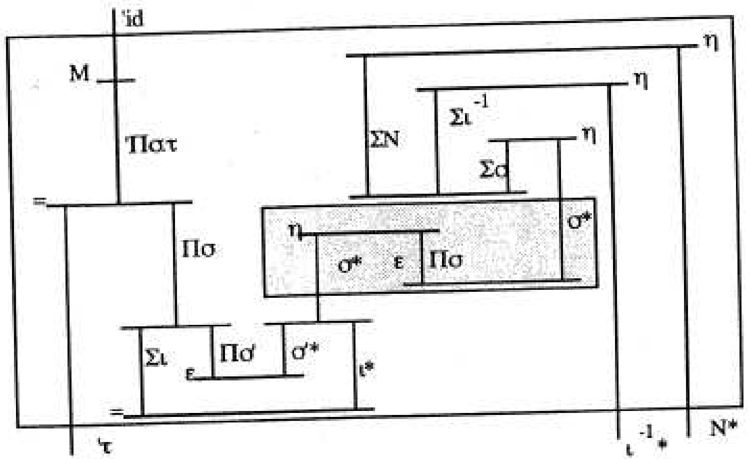


=

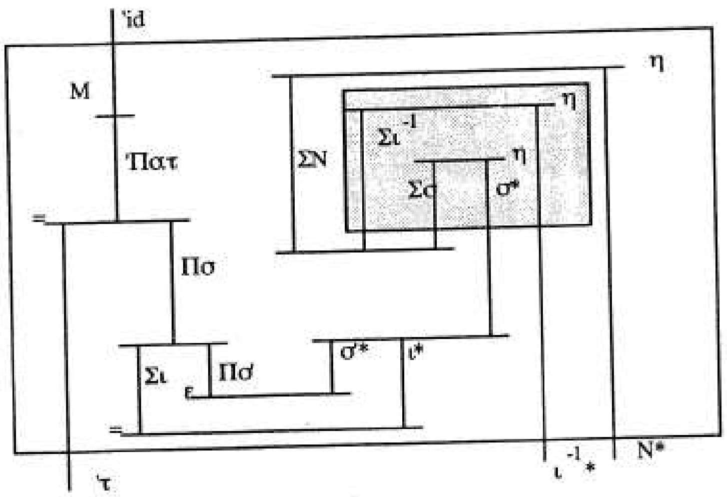


(We expand the inverse iso $\Pi\sigma \leftrightarrow \Pi\sigma' \circ \Sigma\iota$.)

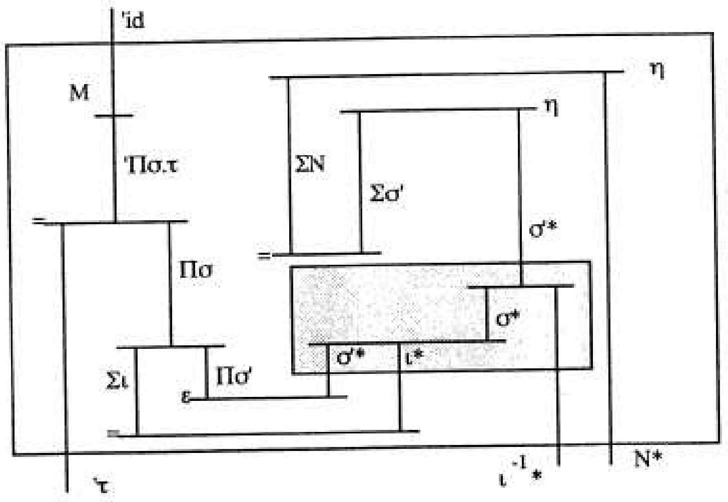
=



=

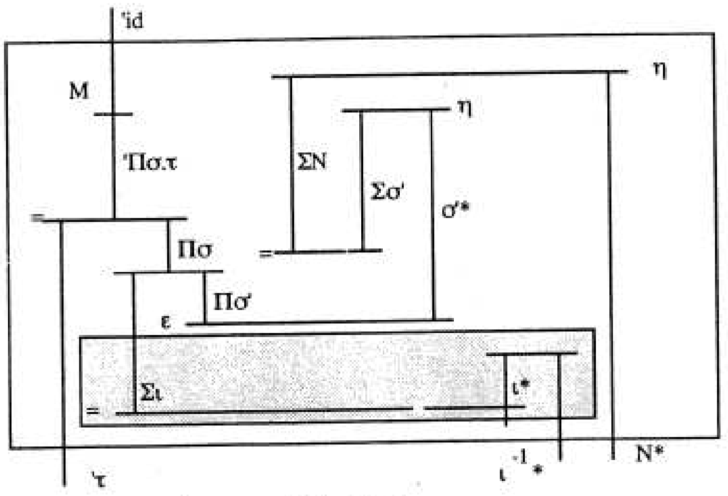


=



(We recall that $\Sigma\iota = \iota^{-1*}$.)

=



After the obvious reduction, we get $M(c_2\langle N\rangle)$.