DIAGRAMMES

P.-L. CURIEN

Substitution up to isomorphism

Diagrammes, tome 23 (1990), p. 43-66

<http://www.numdam.org/item?id=DIA_1990__23__43_0>

© Université Paris 7, UER math., 1990, tous droits réservés.

L'accès aux archives de la revue « Diagrammes » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/legal.php). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

DIAGRAMMES VOLUME 23 , 1990

ACTES DES JOURNEES E.L.I.T.

(UNIV. PARIS 7, 27 JUIN-2 JUILLET 1988)

SUBSTITUTION UP TO ISOMORPHISM

Extended Abstract

P.-L. Curien, LIENS (CNRS URA 1327)[1]

1 Introduction

The essence of the categorical semantics of dependent types is known since the mid seventies (J. Cartmell [Cart] along a computer science line, R. Seely [Se] along a categorical logic line). These early ideas have been revisited by a number of authors [HyPi,Tay,Ehr,Jac,Lam,Ob,Stre], with the aim

- of accommodating them with domain theory on one hand, Grothendieck fibrations on the other hand,
- of lifting them to a categorical semantics of the calculus of constructions of Coquand-Huet.

The aim of this article is to solve a difficulty arising from a serious mismatch between syntax and semantics: substitution in types is modelled by pullbacks (more generally pseudo-functors), that is only up to isomorphism, unless split fibrational hypotheses are imposed, which seems a bit unnatural in category theory, at least viewed from usual standards of mathematics. We have to show that these isomorphisms are *coherent* in a sense familiar to category theorists. Due to this coherence problem at the level of types, we have to

- switch to a syntax where substitutions are explicitly present (whereas in traditional syntaxes substitution is meta, defined by induction),
- include type equality judgements in this modified syntax (we consider here only equalities describing the stepwise performance of substitution).

This introduces a new "flaw": there are now different proofs that a given term has a given type, since at any stage in a proof of well-typing 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 and inclusion; joint work with G. Ghelli) [GheCu]. As in [CuGhe] we attack this problem with tools of rewriting theory (and coherence in category theory).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

[1]Ecole Normale Supérieure, 45 rue d'Ulm 75230 Paris Cedex 05, e-mail curien@dmi.ens.fr

reçu le 22/02/90

syntactic issues.

Prerequisites are the notion of locally cartesian closed category, of Beck-Chevalley condition (see [Se], there is a quick summary in 2.1), and of calculi of dependent types (a first source and agreeable reference is [Mar]). Syntax will be given when needed in the text. We assume also some experience of categorical logic (the quantifiers-as-adjoints paradigm mainly: again the source reference [Law] is a nice reading). The reader may find it helpful to 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 will find there (but also in [Cur1] and in the source paper [Brui]) an account of De Bruijn's nameless notation, which will be also adopted here, and which we briefly recall now.

Roughly, De Bruijn's notational convention consists in replacing variable names by a natural number recording its place in the environment (where the values of free variables are recorded), added to its binding depth, as illustrated by the following example: in the environment cons((t=...$_1$),cons((z=...$_2$),nil)) (written in a Lisp like syntax), t and $\lambda x.(\lambda y.y)z$ become respectively 1 and $\lambda.(\lambda.1)3$. Here the number 3 decomposes as 1+1+1: in order to find z's binder one has to "pass" over λx and the top t=6 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).

2 Interpreting dependent types in locally cartesian closed categories:

2.1 Some categorical preliminaries Let us first fix some categorical notation. We shall use the following definition of locally cartesian closed categories (LCCC): a category $\mathbb{C}$ is locally cartesian closed iff it has a terminal object, and for any k: a$\rightarrow$b the functor Σ_k: $\mathbb{C}/a \rightarrow \mathbb{C}/a$ defined by $\Sigma_k(f) = k \circ f$ admits two successive right adjoints, written k^*, Π_k, i.e. $\Sigma_k \dashv k^* \dashv \Pi_k$.

A feature which is crucially used in the sequel is the *pseudo-functorial* character of the pullback (p.b. for short): for each pair of composable arrows t,s, there exists a natural iso $t^* \circ s^* \leftrightarrow (s \circ t)^*$, 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[2].

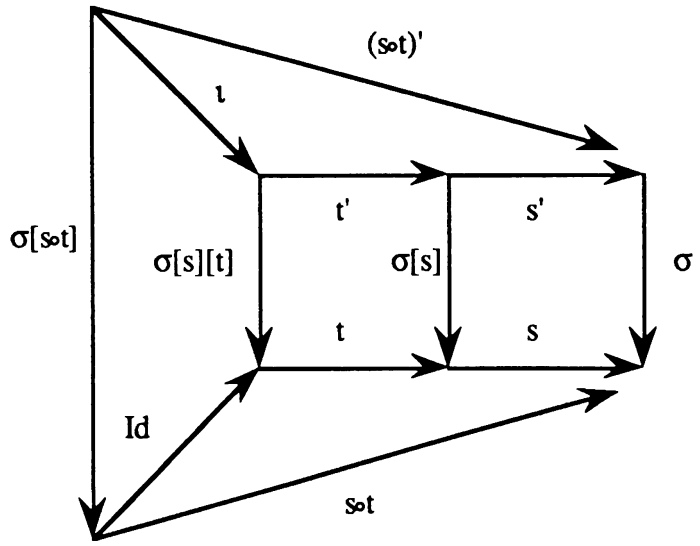
Why is the pullback only pseudo-functorial? The point is that p.b. diagrams compose, but *chosen* pullbacks do not in general. This can be seen in two ways:

1) By a direct argument, as illustrated on Figure 1, where the two inner squares and

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

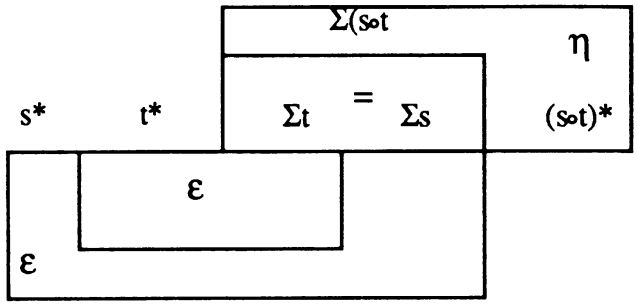
the outer trapezium are the chosen p.b. diagrams of s and σ, t and σ[s], s∘t and σ respectively (s*σ is renamed σ[s]).

Figure 1



2) By making use of $\Sigma_k \dashv k^*$ and $\Sigma_{s\circ t} = \Sigma_s \circ \Sigma_t$. This is more abstract than 1), which was defined "pointwise".The natural transformation $t^* \circ s^* \rightarrow (s\circ t)^*$ is shown on figure 2[3] .

Figure 2



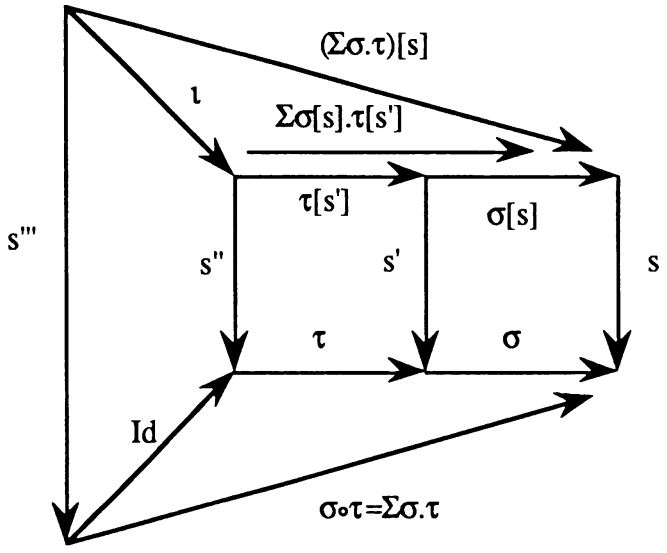
[3]This way of representing natural transformations (more generally 2-cells) is known as pasting. It has been proved to be mathematically well defined only recently [Pow]. A pasting is a labelled planar graph (with additional properties, see [Pow]): points are categories (0-cells), arrows are functors (one-cells) and (bounded) regions are natural transformations (2-cells).

Remark: Pseudo-functoriality arises also for the identities. So one should also consider canonical isos between $\sigma[id]$ and id, and assume as a second coherence condition that the isos $k^*\circ id^* \leftrightarrow (id\circ k)^*$ ($\equiv k^*$) and $k^*\circ id^* \leftrightarrow id\circ k^*$ ($\equiv k^*$) coincide. In the present paper we shall 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 shall 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).

The adjunctions $\Sigma_k \dashv k^*$, together with the pseudo-functoriality of *, determine for each commuting square a canonical natural transformation $\Sigma\sigma[s]\circ s'^* \rightarrow s^*\circ\Sigma\sigma$:

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

This is a quite general construction, which applies in more general indexed categories than the indexed category of slices of a category with pullbacks. In the particular case where * is pullback, this natural transformation, is iso. To show this, we need to go down one level of abstraction, and to have a a more direct, "pointwise" description. We picture its (pointwise) inverse below. The picture is just an "α-conversion" of Figure 1. The component at τ of the natural transformation $s^*\circ\Sigma\sigma \rightarrow \Sigma\sigma[s]\circ s'^*$ is exactly the canonical ι: $(\sigma\circ\tau)^* \rightarrow \tau^*\circ\sigma^*$ (but now it is in the slice over the domain of s, instead of being over the domain of τ).



The property that the transformation $\Sigma\sigma[s]\circ s'^* \rightarrow s^*\circ\Sigma\sigma$ is iso is the *Beck-Chevalley condition*[4]. Symmetrically, the right adjoints to pullbacks yield canonical

[4]Beck-Chevalley condition is important in categorical logic, since it allows to interpret substitution across quantifiers. But it also appears (as well as the notion of exact square [Gui], a suitable abstraction

transformations $s^* \circ \Pi\sigma \to \Pi\sigma[s] \circ s'^*$, and 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.

2.2 Syntax of first-order λ-calculus with dependent types We present a pure first-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), and the main conceptual step is independent of the specific choice of those constants. Dependent types, unlike simple Curry types, have to be proved well-formed. One first defines a syntax of raw (or pre-well-formed) types and terms, given by

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

Dependency will arise 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 have the following structure. A context C is a sequence of assertions of the form $x{:}\sigma$. C(x) is the σ of the first item $x{:}\sigma$ in the list, starting from the right. There are three judgements: C context , $C \vdash \sigma$ type , $C \vdash M{:}\sigma$.

2.3 Name-free syntax Next we turn to a name-free syntax, which is well-suited to the description of meaning. Also, as quoted in the introduction, we pay particular attention to substitution, which will be 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:	$M ::= 1 \mid \lambda\sigma.M \mid MM \mid (M,M) \mid fst(M) \mid snd(M) \mid M[s]$
Substitutions:	$s ::= id \mid \uparrow \mid M.s \mid s \circ s$

The typing rules are as follows (the contexts are now just sequences of types)

---

of it) in other geometric or topological applications of category theory. The Beck Chevalley condition can also be expressed in a synthetic way, using the notion of fibered adjunction [Jac].

Context formation rules

$$\frac{}{\varnothing \text{ context}}$$

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

Type formation rules

[Const]

$$\frac{}{\kappa \text{ type}}$$

[Π] (and symmetrically [Σ])

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

[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]

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

[App]

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

[Pair]

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

[fst]

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

[snd]

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

[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, think that σ in the hypothesis has its De Bruijn indices refering to the sequence C, whereas the second occurrence $\sigma[\uparrow]$ in the conclusion has to refer to the larger context C,σ. Now we refer the reader to the rules MClos and $\uparrow$ below.)

Substitution formation rules

[Id]

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

[$\uparrow$]

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

[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 w.r.t. 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 $(n,l) \mapsto n$ 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 $s: C \to C'$. Finally judgements $C \vdash M:\sigma$ are mapped to triangles $M:\mathrm{Id} \to \sigma$. It is time to 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 this will not be true anymore 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 (the standard way

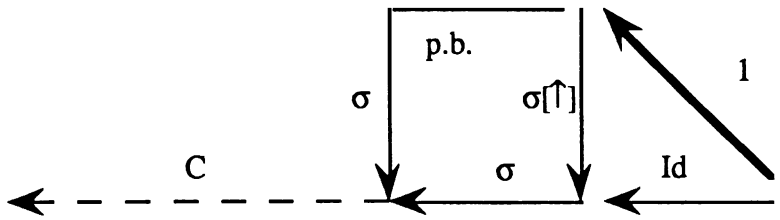
of turning objects into arrows).

[Π] (and symmetrically [Σ]) The meaning of the Π and Σ constructors is given by the adjoints Σ_k and Π_k (thus we shall freely confuse, say $\Pi\sigma.\tau$ and $\Pi_\sigma(\tau)$).

[σClos] $\sigma[s]$ is interpreted as the p.b. of σ along s.

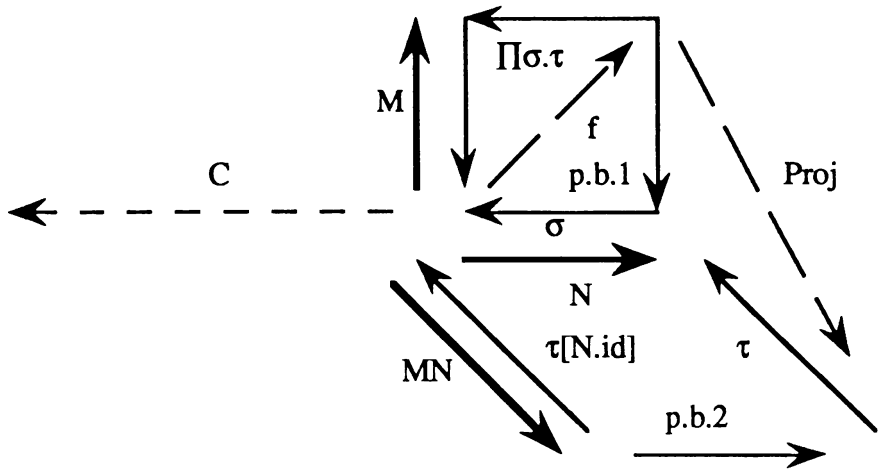
(The reader unfamiliar with the pullback as substitution idea may want to convince himself by taking, say σ=list(n), and s=succ. The obvious interpretation for list(n+1) is the first projection on $\{(m,l) \mid l \in list(m+1)\}$; now the latter set is in one-to-one correspondence with $\{(m,(n,l) \mid (l \in list(n)$ and) $n=m+1\}$, which is the pullback of succ and $(n,l) \mapsto n$.)

[Var] The meaning of 1 is the mediating arrow of id and id relative to the p.b. square of σ and σ. See the case $\uparrow$ for a justification of the arrow named $\sigma[\uparrow]$ on the picture.



[Abs] We write λ for the natural bijection associated with the adjunction $k^* \dashv \Pi_k$, and we use that $\sigma^*(Id)=Id$. Then the interpretation of $\lambda\sigma.M$ is just $\lambda(M)$.

[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 p.b. square 1. Proj is the counity of the adjunction $k^* \dashv \Pi_k$. MN is obtained as the mediating arrow of Proj $\circ$ f and id relative to the p.b. square 2. We anticipate the description of the meaning of [Cons] which will justify the identification made between N.id and N, viewed as a global triangle $C \to C,\sigma$.



[Pair] Again M.id is identified with M as a global triangle $C \to C,\sigma$. (M,N) is the composed arrow $M' \circ N$, where M' is the last side of the p.b. 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 p.b. square of τ and fst(M).

[MClos] M[s] is obtained as the mediating arrow of id and $M \circ s$ w.r.t. the p.b. square of σ and s.

[Id,Comp] Identity, composition!

[$\uparrow$] The interpretation of $C,\sigma \vdash \uparrow: C$ is $\sigma: C \circ \sigma \to C$:

[Cons] The meaning of M.s is $s' \circ M$ in the following picture where s' is the last side of the p.b. of σ and s.

3 The coherence problem[5]

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[6] $(\lambda M)N \to M[N.id]$

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

[6]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	$1[id] \to 1$	IdL	$id \circ s \to s$
VarCons	$1[M.s] \to M$	ShiftId	$\uparrow \circ id \to \uparrow$
App	$(MN)[s] \to M[s](N[s])$	ShiftCons	$\uparrow \circ [M.s] \to s$
Abs	$(\lambda\sigma.M)[s] \to \lambda\sigma.(M[1.s \circ \uparrow])$	Map	$(M.s) \circ t \to M[t].(s \circ t)$
Clos	$M[s][t] \to M[s \circ t]$	Ass	$(s_1 \circ s_2) \circ s_3 \to s_1 \circ (s_2 \circ s_3)$

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

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

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

3.2 Introducing type equality judgements When typing both sides of VarCons, we obtain different types: $\sigma[\uparrow][M.s]$ and $\sigma[s]$. The crucial step in showing that these types are equal is $\sigma[\uparrow][M.s]$ "=" $\sigma[\uparrow \circ (M.s)]$, which is only an isomorphism by the pseudo-functoriality of the pullback. We 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: indeed 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]

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

This rule presents a mismatch. One expects (as a metatheorem on syntax) that if $C \vdash \sigma=\tau$ type is derivable, then also $C \vdash \sigma$ type and $C \vdash \tau$ type are derivable. Now from the hypotheses, and the conclusion, respectively we get $C,\sigma \vdash \tau'$ type and $C,\sigma' \vdash \tau'$ type. This suggests to complete our extension of syntax by yet another kind of judgement: $\vdash C=C'$ context, and adding a "contravariant" counterpart of EqType (we state one for type judgements, but there should be one for each kind of judgement):

[CongCont]

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

[EqCont]

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

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

[ClosCong]

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

We extend the 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'.

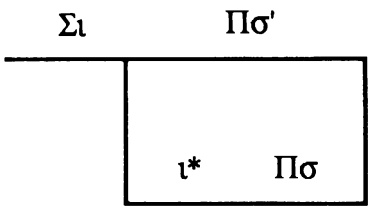
[EqType] If M abbreviates the meaning of (the proof of) $C \vdash M:\sigma$, then the meaning of $C \vdash M:\tau$ is M composed with ι.

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

[EqCont] This is dual to EqType.

[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 p.b. square $\sigma' \circ \iota = Id \circ \sigma$ (ι is as in the figure for CongCont). Notice that since we have chosen $\iota^{-1}*$ as $\Sigma\iota$, the picture describes indeed a transformation $\Pi\sigma' \circ \Sigma\iota \to \Pi\sigma$.



[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 3.3), thus σ[s] and σ[t] coincide, being both the chosen pullback of the same pair of arrows.

3.3 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 3.1 has to be revised, in order to be able to map both hand sides of each equality to strictly equal meanings. For example:

VarCons $c_{\sigma[\uparrow][M.s],\sigma[s]}\langle 1[M.s]\rangle \to M$

App $c_1\langle (MN)[s]\rangle \to (c_2\langle M[s]\rangle)(N[s])$ where

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

We shall call these new equations the *explicit versions* of the rules of 3.1. Strictly speaking, we should be even more explicit, and develop a language for describing

coercions, i.e. 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.

3.4 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 1, with a graphical proof technique.

Remark: We should stress here the power of the Beck 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 too, 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 terms) by the (explicit version of)

Beta +VarId+VarCons+App+Abs+Clos+Ass+ΠAbs+ΣAbs+Pseudo

is confluent and strongly terminating.
(The proof adapts quite easily from similar results in [ACCL,HarLa].)

Theorem 6: For any C,σ,τ, any two proofs of $C \vdash \sigma=\tau$ type receive equal interpretations in any LCCC.
Proof sketch: 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). Typically we have to check that the two proofs

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

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 equal in all interpretations[7].

We further restrict our attention to the proofs where not only the cuts occur at the end, but where 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 [HueOp]: 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. There is actually just one critical pair (if one forgets the critical pairs arising from rules on terms and substitutions, for which the verification is trivial, since ClosCong is interpreted up to equality), between ΠAbs and Pseudo. The analysis of this critical pair is presented in Annex 2.

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.

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 sketch: 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 $\sigma,\sigma' \rightarrow^* \sigma''$, and similarly for τ, τ', τ''.

Theorem 8: For any C,M, if $C \vdash M:\sigma$ and $C \vdash M:\sigma'$ are provable, then $\sigma=\sigma'$ is provable. Moreover, for any C, M, σ, any two proofs of $C \vdash M:\sigma$ receive equal

[7]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 condition (which states that a canonical arrow is iso) is essential.

interpretations in any LCCC (and similarly for C,s,C').

Proof sketch: By induction on the sum of the lengths of these proofs, and by cases on the shape of M.One has to generalize the statement to take care of possible uses of EqType and EqCont. What we prove is actually:

- For any C, M, 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' s.t. $\vdash C,\sigma=C',\sigma'$ are provable, any two proofs of $C \vdash M:\sigma$, $C' \vdash M:\sigma'$ respectively receive interpretations in any LCCC (and similarly for C,s,C') which become equal when composed with the isos $C \leftrightarrow C'$ and $s \leftrightarrow s'$, properly oriented.

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)

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

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. 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 3.3 to illustrate coherence of proofs (we omit the easy checking; this corresponds to the rule λ of [CuGhe]).

4. Conclusions and future work

Beyond the technical coherence result, we believe that the graphical representation 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 such drawing construction and manipulation.

We believe that the coherence results shown here also hold in more general models

(relatively locally cartesian closed categories [HyPi], D-categories [Ehr]). We would like to extend them to models of the calculus of constructions of T. Coquand anf G. Huet [Coq].

Acknowledgements The author wishes to acknowledge G. Huet (who first pointed out to him the connection between coherence and Knuth-Bendix), and the teaching staff of the Departamento de Algebra of the university of Santiago de Compostela (who introduced them to their graphical proof methods). I wish also to thank my coauthors of [ACCL] and [CuGhe], on which the present paper is built up. Finally the work 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).

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

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

[Coq] T. Coquand, Metamathematical Investigations of a Calculus of Constructions, Technical Report, Cambridge University (1987).

[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, accepted at CAAP 90, Copenhaguen.

[Cur1] P.-L. Curien, Categorical Combinators, Sequential Algorithms and Functional Programming, Pitman (1986).

[Cur2] P.-L. Curien, α-conversion, Conditions on Variables and Categorical Logic, Studia Logica, to appear.

[Ehr] T. Ehrhard, A Categorical Semantics of Constructions, LICS 88, Edinburgh (1988).

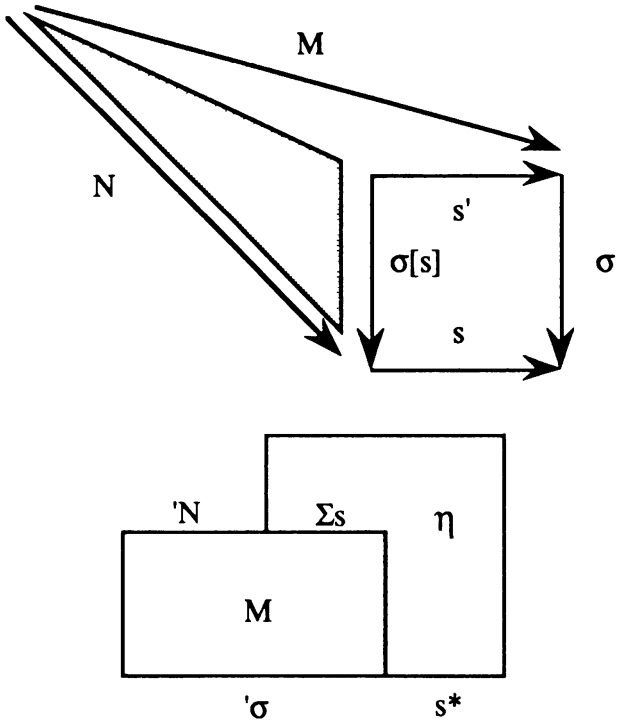
[Guit] R. Guitart, Relations et Carrés Exacts, Annales Sci. Math. du Québec, Vol. IV , No2, pp. 103-125 (1980).

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

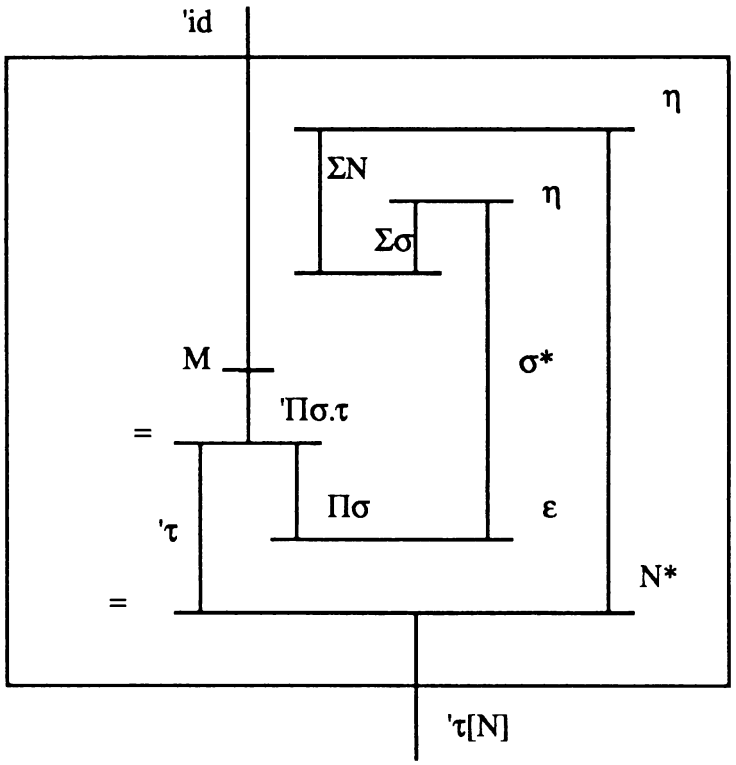
[HueOp] G. Huet, D. Oppen, Equations and Rewrite Rules: a Survey, in Formal

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, Proc. Conf. on Categories in Computer Science and Logic, to appear as Contemporary Mathematics volume, Boulder (1987).
[Jac] B. Jacobs, Some Notes on Fibred Categories and the Semantics of Dependent Types, Technical Report 22/89, Universita di Pisa, (1989).
[Laf] Y. Lafont, Personal communication.
[Lam] F. Lamarche, A Simple Model of the Theory of Constructions, in J. Gray and A. Scedrov (eds), Categories in Computer Science and Logic, Contemporary Mathematics 92, 1989.
[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. LICS 86.
[Ob] A. Obtulowicz, Categorical and Algebraic Aspects of Martin-Löf Type Theory, Studia Logica, to appear.
[Pow] A. Power, A 2-categorical Pasting Theorem, Journal of Algebra, to appear.
[Rod] E.G. Rodeja F., editor of various technical reports of the Departamento of Algebra y Fundamentos del Universidad de Santiago de Compostela, starting with Teoria de Triples by J.R. Caruncho Castro.
[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).
[Tay] P. Taylor, Recursive Domains, Indexed Category Theory and Polymorphism, PhD Thesis, University of Cambridge (1986).

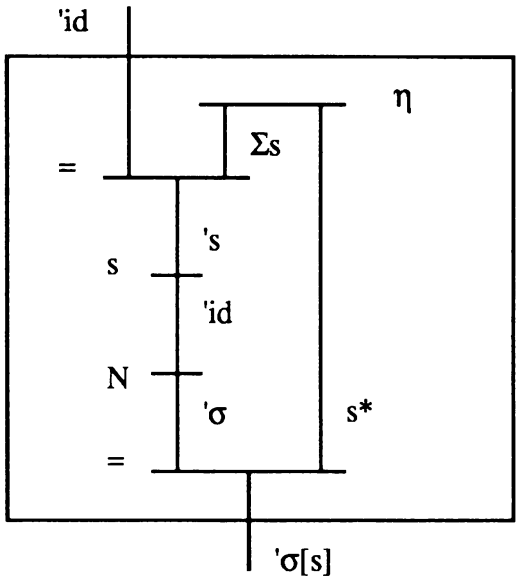
Annex 1 *(Validation of App)* In order to validate the rules of the explicit calculus, we have to reexpress the semantic interpretation in 2-categorical terms. If k is an arrow into a, we denote by 'k the constant functor with value k from the terminal category 1 to the slice $\mathbb{C}/a$. We have then $(\Sigma l)\circ('k)='(l\circ k)$, $g^*\circ('k)='k[g]$ for l,g of appropriate types. If f is an arrow $k\to l$ in $\mathbb{C}/a$, then we represent it as a natural transformation from 'k to 'l. The main tool used in 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,N w.r.t. the p.b. diagram of s and σ, considered as an arrow $N\to\sigma[s]$:



Now we can express the meaning MN as a natural transformation.We use a "dual" notation, suggested by Y. Lafont [Laf], 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, i.e. categories. Redexes are isolated by appropriately stretching a "dual pasting diagram", both horizontally and vertically. The redisplay after reduction is much easier with this representation. Indeed, in the presentation by pastings, after an $\eta\varepsilon$-cancellation has occurred, two arcs have to be merged (for example the two arcs $\Sigma(s \circ t)$ in the first picture of Annex A.1, or the two arcs $\Sigma(k'' \circ k')$ in the picture below), and it is not always immediate to see how to redisplay the neigbouring arcs and regions after merging (for example the counity ε at k in the dual of the picture below). In the dual representation, instead of having to merge arcs, we just need to tie them.

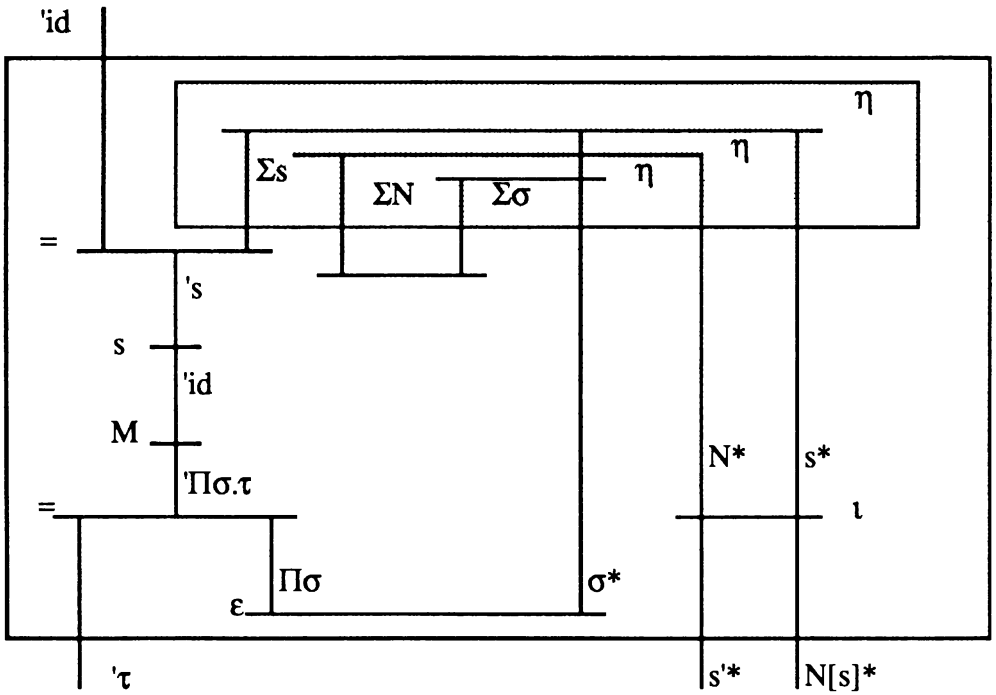


and similarly for the meaning of N[s]:

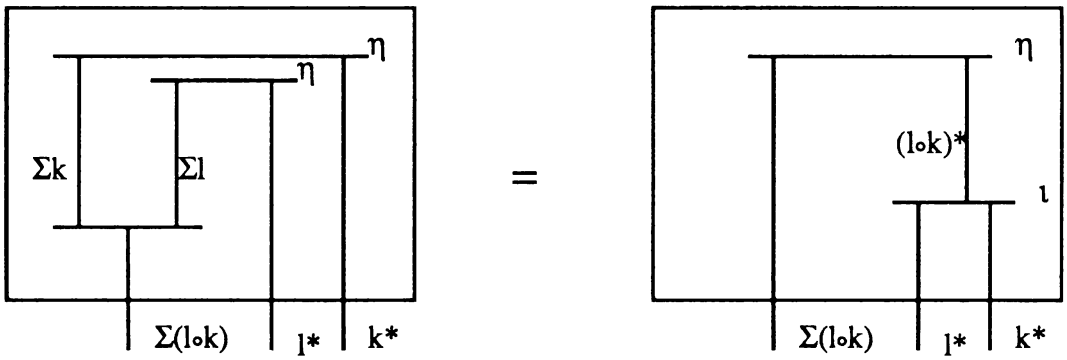


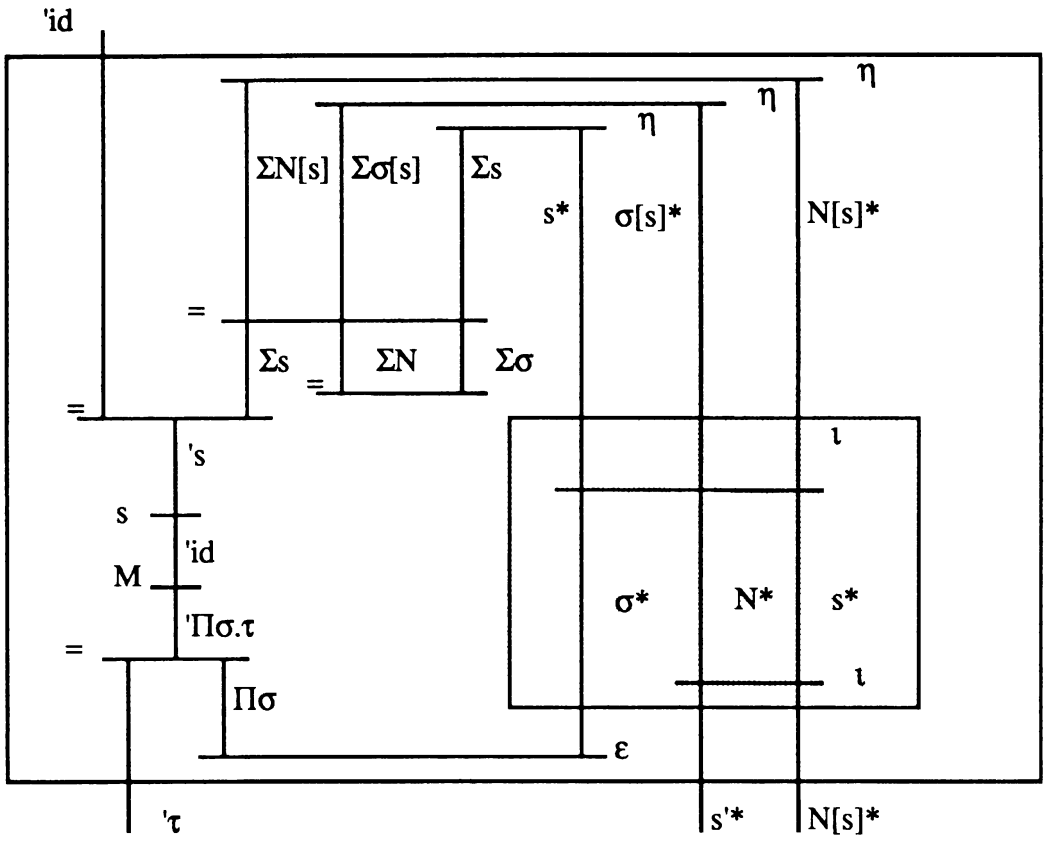
Now we are able to draw c_1<(MN)[s]>, and to rewrite it[8].

[8]The technique of pasting rewriting has been conceived by E. Rodeja, and used in mathematical practice for almost twenty years by the Santiago de Compostela algebra group [Rod].

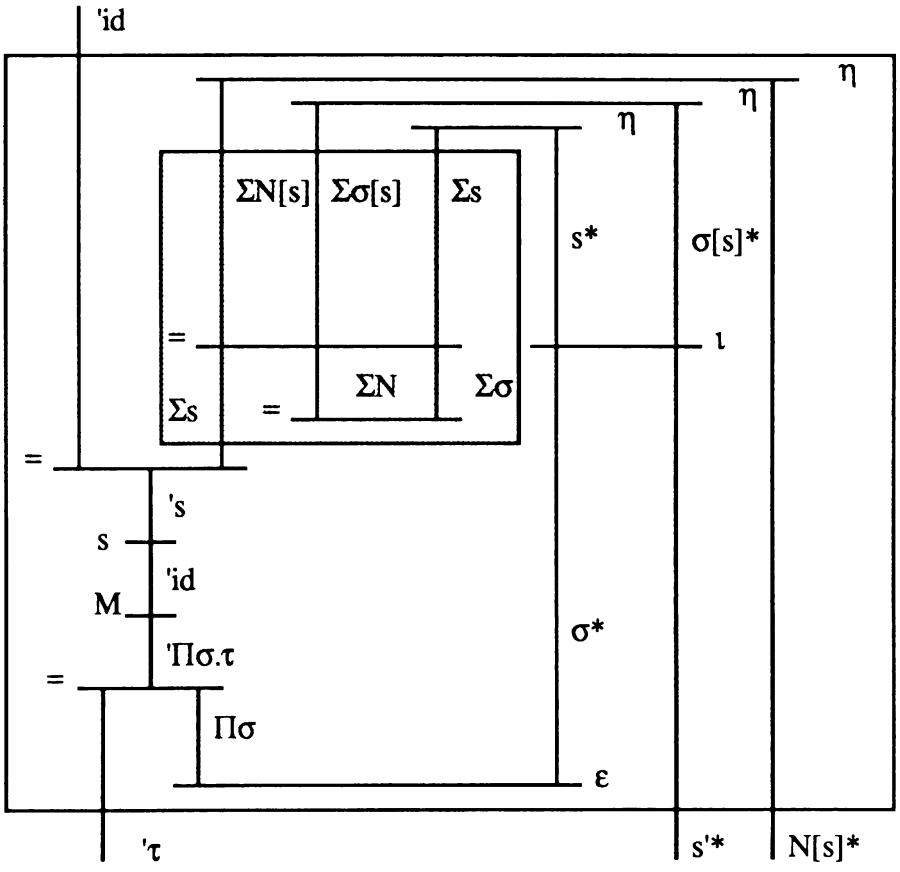


We use the following equation, easily derivable from the definition of ι.

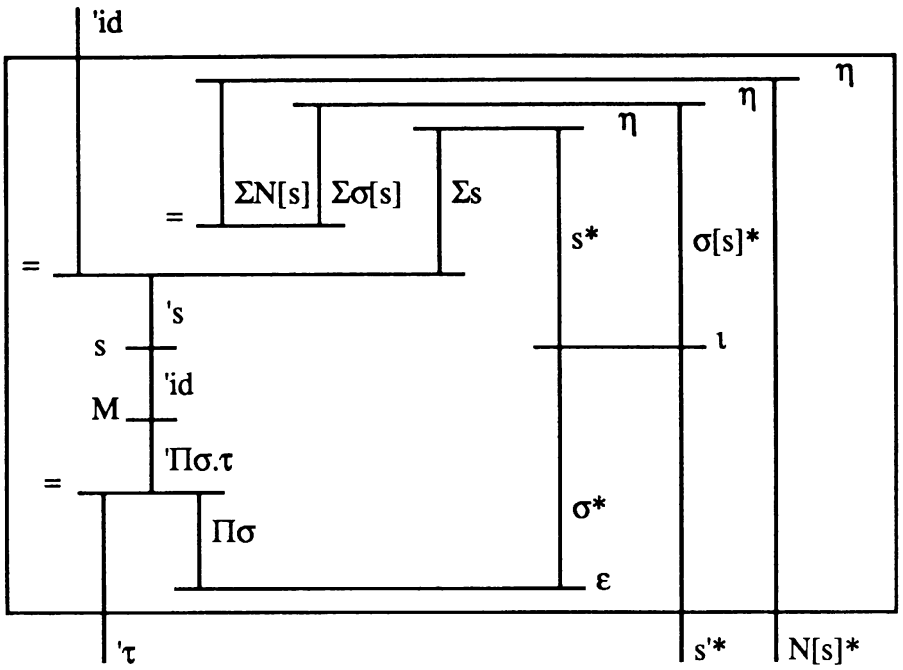



'id
η
η
η
ΣN[s]
Σσ[s]
Σs
s*
σ[s]*
N[s]*
=
Σs
=
ΣN
Σσ
=
ι
's
s
'id
M
'Πσ.τ
=
σ*
N*
s*
ι
Πσ
ε
't
s'*
N[s]*

=

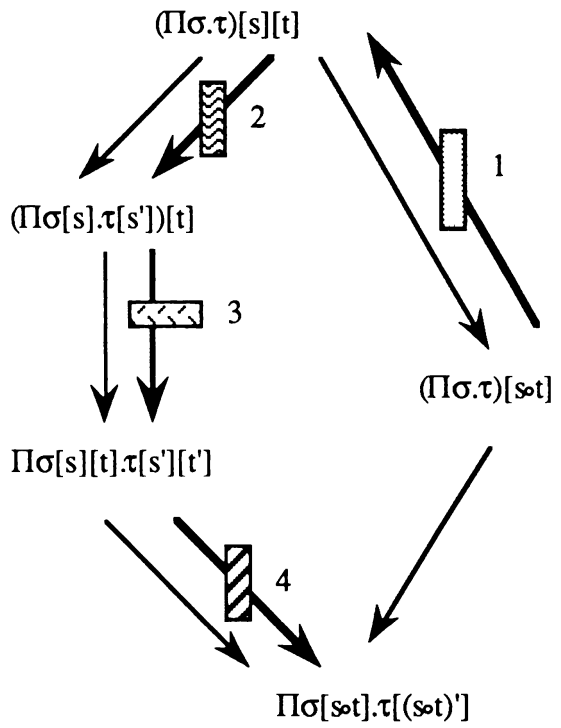

'id
η
η
η
ΣN[s]
Σσ[s]
Σs
s*
σ[s]*
=
ι
Σs
=
ΣN
Σσ
=
's
s
'id
M
'Πσ.τ
=
σ*
Πσ
ε
't
s'*
N[s]*

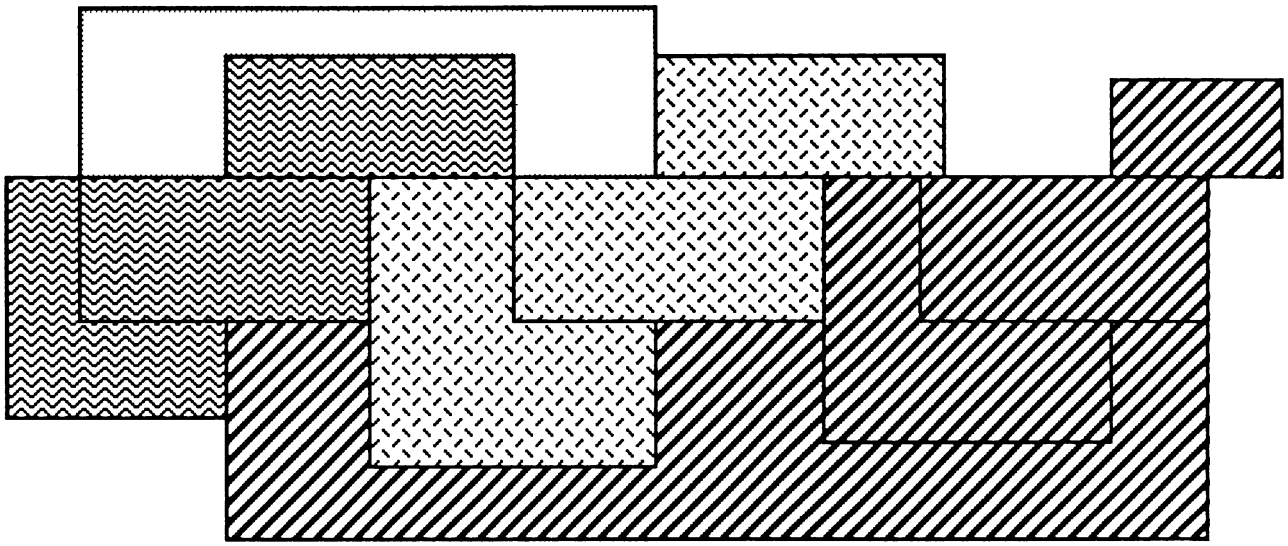
=



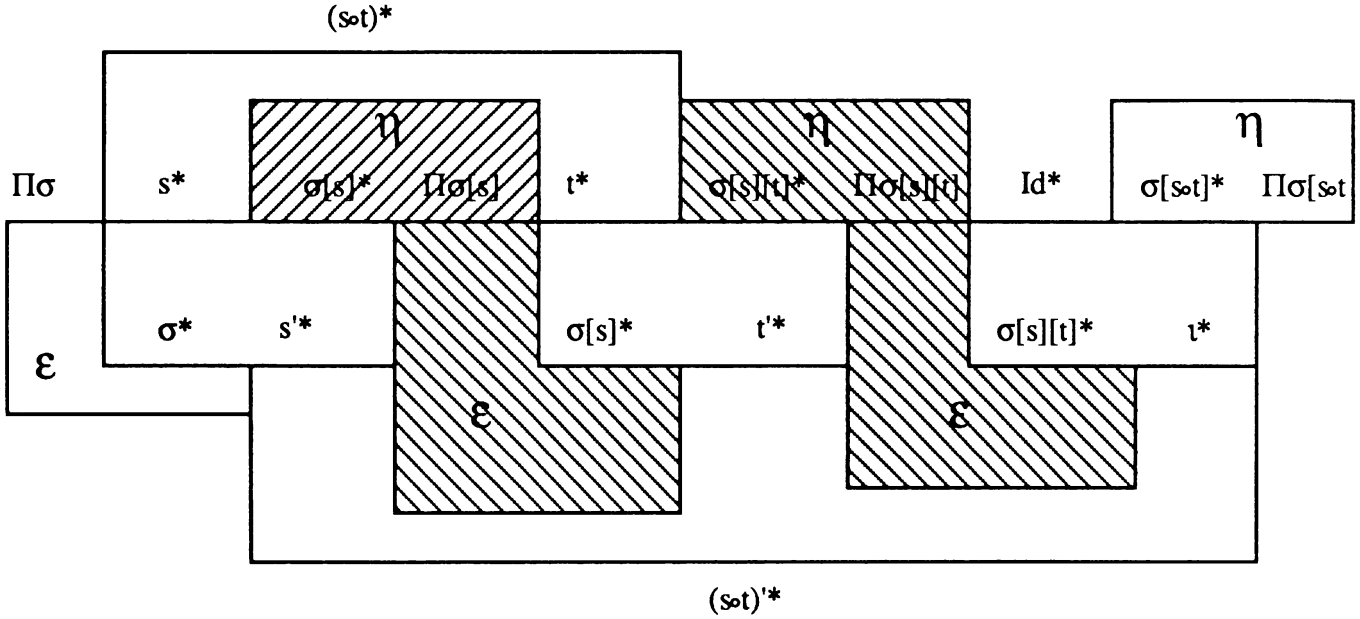
We leave the reader check that the drawing obtained by ηε-expanding the vertical σ[s]* line is $(c_2\langle M[s]\rangle)(N[s])$.

Annex 2 *(The critical pair ΠAbs-Pseudo)* (The notation is taken from Figure 1). We first describe the completion of the critical pair, then build and paste the isomorphisms along one path (we have reversed one direction for convenience); finally we rewrite those pastings until we obtain at the end the iso obtained by following the other path $(\Pi\sigma.\tau)[s\circ t]\rightarrow(\Pi\sigma[s\circ t].\tau[(s\circ t)']$.

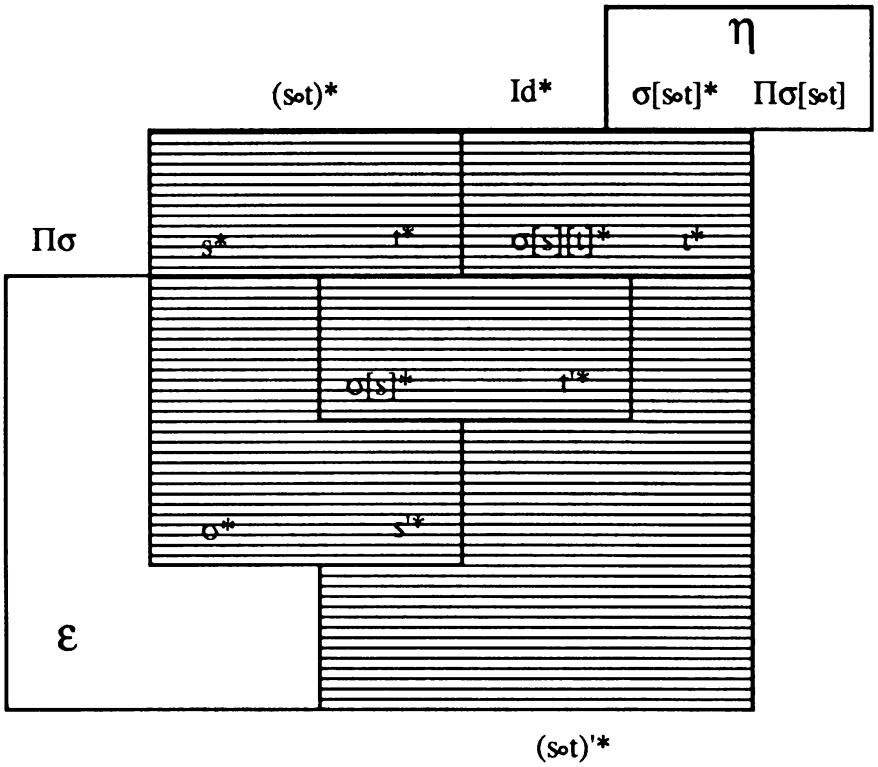




=


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

=


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