Extracting a proof of coherence for monoidal categories from a formal proof of normalization for monoids

Ilya Beylin and Peter Dybjer
Department of Computing Science
Chalmers University of Technology
Göteborg, Sweden

Abstract

This paper studies a connection between intuitionistic type theory and coherence problems in the sense of category theory. We first show how a Curry-Howard interpretation of a formal proof of normalization for monoids almost directly yields a coherence proof for monoidal categories. Then we formalize the coherence proof in type theory and show how this proof relies on "metacoherence", that is, the unicity of certain proofs of equality in intuitionistic type theory. This formalization has also been checked in the mechanical proof assistant ALF.

Let M be the set of binary words with variables in X, that is, the least set such that $e \in M$, if $a, b \in M$ then $a \otimes b \in M$, and if $x \in X$ then $x \in M$. Write $a \sim b$ if a and b are congruent with respect to associativity ($a \otimes (b \otimes c) \sim (a \otimes b) \otimes c$) and unit laws ($e \otimes a \sim a$ and $a \otimes e \sim a$). Hence $M/\sim$ is a free monoid.

Let N be the subset of normal binary words, that is, the least set such that $e \in N$ and if $n \in N$ and $x \in X$ then $n \otimes x \in N$. We can prove the following normalization theorem: there is a function $Nf : M \to N$, such that $a \sim b$ iff $Nf(a) = Nf(b)$.

To get a proof of coherence for monoidal categories we analyze the formal proof objects of this proof of normalization, that is, we use a 'Curry-Howard interpretation'. First construct a category $\mathcal{M}$ with objects binary words and arrows in $\mathcal{M}(a, b)$ formal proofs of $a \sim b$. With a suitable notion of equality of such proofs this category becomes a free monoidal category (generated by X). Coherence for monoidal categories states that any two formal proofs of $a \sim b$ are equal. Since $a \sim b$ implies $Nf(a) = Nf(b)$ we can extend Nf to a functor from $\mathcal{M}$ to $\mathcal{N}$ (the set N considered as a category). Moreover, we can check that the proof of $a \sim Nf(a)$ is a natural isomorphism ν_a. Hence coherence follows in the usual way: all proofs of $a \sim b$ are equal to $\nu_b^{-1} \circ \nu_a$.

Note how the proof of coherence for monoidal categories is more or less implicit in the normalization proof for monoids. Given the Curry-Howard interpretation of this proof the extra work is limited to a fairly mechanical check that ν is a natural transformation.

1 Introduction

MacLane [17, pp 161-165] proved a coherence theorem for monoidal categories. An important ingredient in his proof is the normalization of object expressions. But it is only one ingredient and there are several others needed to get the proof through.

Here we show that almost the whole proof of this coherence theorem is hidden in a proof of normalization for monoids, provided we consider proof objects for equality of monoidal expressions, and equalities of such proof objects for equality.

Discussions with John Power revealed that the extracted proof is essentially a logical version of the proof of coherence for monoidal categories obtained as a special case of the proof of coherence for bicategories given in the recent paper by Gordon, Power, and Street [9]. Their proof relies on Street's bicategorical Yoneda lemma. The interest of our proof is that it is a nice application of the Curry-Howard interpretation. It was discovered independently and uses only some elementary category theory and the elementary reasoning needed for a proof of normalization for monoids.

The paper is organized in the following way. In section 2 we prove a normalization theorem for monoids. In section 3 we introduce the notion of a monoidal category and prove coherence for it.

The this section a standard informal mathematical metalanguage is used. In section 4 we show how the proofs can be formalized in intuitionistic type theory. Section 5 contains a few remarks about our experience of implementing the proof in ALF. Section 6 refers to some related work.

2 Normalization for monoids

Let M be the set of binary words with variables in the set X, that is, the least set such that $e \in M$, if $a, b \in M$ then $a \otimes b \in M$, and if $x \in X$ then $x \in M$. Write $a \sim b$ if a and b are congruent with respect to associativity ($a \otimes (b \otimes c) \sim (a \otimes b) \otimes c$) and unit laws ($e \otimes a \sim a$ and $a \otimes e \sim a$). The subset N of normal binary words is the least set such that $e \in N$ and if $n \in N$ and $x \in X$ then $n \otimes x \in N$.

We have the following "obvious" normalization theorem (see Hedberg [10]):

Theorem 1 *There is a function $Nf : M \to N$, such that $a \sim b$ iff $Nf(a) = Nf(b)$.*

A simple way to construct such a function is by making use of the monoidal structure of N^N. So let $Nf(a) = [\![a]\!](e)$ where $[\![\ ]\!] : M \to N^N$ is defined by

$$\begin{aligned} [\![a \otimes b]\!](n) &= [\![b]\!]([\![a]\!](n)) \\ [\![e]\!](n) &= n \\ [\![x]\!](n) &= n \otimes x \end{aligned}$$

It follows (by induction on the proof that $a \sim b$) that if $a \sim b$ then $Nf(a) = Nf(b)$. Moreover, $n \otimes a \sim [\![a]\!](n)$ by induction on a. Hence $a \sim Nf(a)$. Hence $a \sim b$ iff $Nf(a) = Nf(b)$.

3 Coherence for monoidal categories

3.1 The generalized algebraic theory of monoidal categories

To fix notation we present the definition of a monoidal category as a generalized algebraic theory in the sense of Cartmell [5]:

There are two *sort symbols*:

$$\begin{array}{rl} Obj : sort & \\ a \longrightarrow b : sort & [a, b : Obj] \end{array}$$

the sort of objects and the sort of arrows indexed by two objects.

There are the following *operator symbols*:

$$\begin{array}{rl} e : Obj & \\ a \otimes b : Obj & [a, b : Obj] \\ g \circ f : a \longrightarrow c & [a, b, c : Obj; g : b \longrightarrow c; f : a \longrightarrow b] \\ id : a \longrightarrow a & [a : Obj] \\ f \otimes g : a \otimes b \longrightarrow a' \otimes b' & [a, a', b, b' : Obj; f : a \longrightarrow a'; g : b \longrightarrow b'] \\ \alpha : a \otimes (b \otimes c) \longrightarrow (a \otimes b) \otimes c & [a, b, c : Obj] \\ \alpha^{-1} : (a \otimes b) \otimes c \longrightarrow a \otimes (b \otimes c) & [a, b, c : Obj] \\ \lambda : e \otimes a \longrightarrow a & [a : Obj] \\ \lambda^{-1} : a \longrightarrow e \otimes a & [a : Obj] \\ \rho : a \otimes e \longrightarrow a & [a : Obj] \\ \rho^{-1} : a \longrightarrow a \otimes e & [a : Obj] \end{array}$$

We have used infix notation and omitted arguments. Without these conventions $g \circ f$ would be written $\circ(a, b, c, g, f)$ for example.

The *equational axioms* are those for a category, for bifunctoriality of $\otimes$, for expressing that α, λ, ρ are natural isomorphisms with inverses $\alpha^{-1}, \lambda^{-1}, \rho^{-1}$, and finally the following coherence equations:

$$\begin{tikzcd}
a \otimes (b \otimes (c \otimes d)) \arrow[r, "\alpha"] \arrow[d, "id \otimes \alpha"'] & (a \otimes b) \otimes (c \otimes d) \arrow[r, "\alpha"] & ((a \otimes b) \otimes c) \otimes d \\
a \otimes ((b \otimes c) \otimes d) \arrow[rr, "\alpha"'] & & (a \otimes (b \otimes c)) \otimes d \arrow[u, "\alpha \otimes id"']
\end{tikzcd}$$

$$\begin{tikzcd}
a \otimes (e \otimes b) \arrow[r, "\alpha"] \arrow[dr, "id \oplus \lambda"'] & (a \otimes e) \otimes b \arrow[d, "\rho \otimes id"] \\
& a \otimes b
\end{tikzcd}$$

In our proof we do not use the third coherence equation $\lambda_e = \rho_e$ given by MacLane [17, pp 158-159] which was later proved to be redundant by Kelly.

3.2 The coherence theorem

We first construct a free monoidal category $\mathcal{M}$ generated by the set X. Its objects are binary words with variables in X, that is, elements of M. Its hom-sets are equivalence classes of arrow expressions. We inductively generate the family of sets of arrow expressions (indexed by objects) by interpreting each operator symbol for arrows in the definition of a monoidal category as an inductive clause. Similarly, equivalence of arrow expressions is the least congruence relation satisfying all the equational axioms for a monoidal category. (We will use an arrow expressions f to stand for its equivalence class $[f]$. In intuitionistic type theory this is not just abuse of notation, since one does not form quotients by taking equivalence classes. Instead one uses pairs of sets together with equivalence relations. See the discussion on quotients and setoids in the type-theory formalization in section 4.)

It is easy to see that $\mathcal{M}$ is a groupoid and that $a \sim b$ in M iff $a \cong b$ in $\mathcal{M}$. Indeed, the operator symbols for arrows in the definition of a monoidal category correspond to proof objects for the axioms of equality in a monoid (except that there is no *general* 'inverse' operator corresponding to the symmetry law).

Moreover, let $\mathcal{N}$ be the set N (of normal words) considered as a category.

The coherence theorem refers to $\mathcal{M}$:

Theorem 2 *If $f, f' : a \longrightarrow b$, then $f = f'$.*

For the proof we extend the normalization function to a *functor*

$$Nf : \mathcal{M} \to \mathcal{N}$$

and construct a *natural isomorphism*

$$\nu_a : a \cong Nf(a)$$

witnessing the fact that a term is congruent to its normal form.

The theorem follows immediately: if $f, f' : a \longrightarrow b$, then $f = \nu_b^{-1} \circ \nu_a = f'$.

First, we extend the function $[\![\;]\!] : M \to N^N$ to a functor

$$[\![\;]\!] : \mathcal{M} \to \mathcal{N}^{\mathcal{N}},$$

since it follows by induction on f that if $f \in a \longrightarrow b$ then $[\![a]\!] = [\![b]\!]$. Hence

$$[\![f]\!] = id_{[\![a]\!]} = id_{[\![b]\!]}$$

is well-defined.

Then we let

$$Nf(f) = [\![f]\!]_e = id_{Nf(a)} = id_{Nf(b)}$$

and

$$\nu_a = \xi_{a,e} \circ \lambda^{-1}$$

where the natural isomorphism

$$\xi_{a,n} : n \otimes a \cong \llbracket a \rrbracket(n)$$

is defined by recursion on the form of objects of $\mathcal{M}$:

$$\xi_{a \otimes b,n} : n \otimes (a \otimes b) \xrightarrow{\alpha} (n \otimes a) \otimes b \xrightarrow{\xi_{a,n} \otimes id} \llbracket a \rrbracket(n) \otimes b \xrightarrow{\xi_{b,\llbracket a \rrbracket n}} \llbracket b \rrbracket(\llbracket a \rrbracket(n))$$

$$\xi_{e,n} : n \otimes e \xrightarrow{\rho} e$$

$$\xi_{x,n} : n \otimes x \xrightarrow{id} n \otimes x$$

(Note that there are dual definitions of the inverses ν_a^{-1} and $\xi_{a,n}^{-1}$.)

Naturality of ξ is proved by induction on the structure of an arrow expression of $\mathcal{M}$. For each case we draw a commuting diagram. In these we indicate explicitly when we have used an induction hypothesis (ind), a coherence equation (coh), functoriality (fun) of $\otimes$, or naturality (nat). When there is no explicit indication we have simply unfolded the definition of ξ or used a basic category law.

Case $g \circ f$.

$$\begin{tikzcd}
n \otimes a \arrow[rr, "id \otimes (g \circ f)"] \arrow[dr, "id \oplus f"] \arrow[ddd, "\xi_{a,n}"'] & \arrow[d, phantom, "(\text{fun})"] & n \otimes c \arrow[ddd, "\xi_{c,n}"] \\
& n \otimes b \arrow[ur, "id \oplus g"'] \arrow[d, "\xi_{b,n}"] & \\
(\text{ind}) & \llbracket b \rrbracket(n) \arrow[dr, "id"] & (\text{ind}) \\
\llbracket a \rrbracket(n) \arrow[ur, "id"] \arrow[rr, "id"'] & & \llbracket c \rrbracket(n)
\end{tikzcd}$$

Case id.

$$\begin{tikzcd}
n \otimes a \arrow[r, "id \otimes id"] \arrow[d, "\xi_{a,n}"'] \arrow[dr, phantom, "(\text{fun})"] & n \otimes a \arrow[d, "\xi_{a,n}"] \\
\llbracket a \rrbracket(n) \arrow[r, "id"'] & \llbracket a \rrbracket(n)
\end{tikzcd}$$

Case $f \otimes g$.

$$\begin{tikzcd}
n \otimes (a \otimes b) \arrow[rrrr, "id \otimes (f \otimes g)"] \arrow[ddr, "\alpha"] \arrow[dddddd, "\xi_{a \otimes b, n}"'] & & & & n \otimes (a' \otimes b') \arrow[ddl, "\alpha"'] \arrow[dddddd, "\xi_{a' \otimes b', n}"] \\
& & \text{(nat)} & & \\
& (n \otimes a) \otimes b \arrow[rr, "(id \otimes f) \otimes g"] \arrow[dd, "\xi_{a,n} \otimes id"'] & & (n \otimes a') \otimes b' \arrow[dd, "\xi_{a',n} \otimes id"] & \\
& & \text{(ind)} & & \\
& \llbracket a \rrbracket (n) \otimes b \arrow[rr, "id \otimes g"'] \arrow[ddl, "\xi_{b, \llbracket a \rrbracket (n)}"] & & \llbracket a' \rrbracket (n) \otimes b' \arrow[ddr, "\xi_{b', \llbracket a' \rrbracket (n)}"] & \\
& & \text{(ind)} & & \\
\llbracket b \rrbracket (\llbracket a \rrbracket (n)) \arrow[rrrr, "id"'] & & & & \llbracket b' \rrbracket (\llbracket a' \rrbracket (n))
\end{tikzcd}$$

Case α.

$$\begin{tikzcd}
n \otimes (a \otimes (b \otimes c)) \arrow[rrrr, "id \otimes \alpha"] \arrow[ddr, "\alpha"] \arrow[dddd, "\xi_{a \otimes (b \otimes c), n}"'] & & & & n \otimes ((a \otimes b) \otimes c) \arrow[ddl, "\alpha"'] \arrow[dddd, "\xi_{(a \otimes b) \otimes c, n}"] \\
& & \text{(coh)} & & \\
& (n \otimes a) \otimes (b \otimes c) \arrow[r, "\alpha"] \arrow[d, "\xi_{a,n} \otimes id"'] \arrow[dr, phantom, "\text{(nat)}"] & ((n \otimes a) \otimes b) \otimes c \arrow[d, "(\xi_{a,n} \otimes id) \otimes id"] & (n \otimes (a \otimes b)) \otimes c \arrow[l, "\alpha \otimes id"'] \arrow[d, "\xi_{a \otimes b, n} \otimes id"] & \\
& \llbracket a \rrbracket (n) \otimes (b \otimes c) \arrow[r, "\alpha"'] \arrow[dl, "\xi_{b \otimes c, \llbracket a \rrbracket (n)}"] & (\llbracket a \rrbracket (n) \otimes b) \otimes c \arrow[r, "\xi_{b, \llbracket a \rrbracket (n)} \otimes id"'] & \llbracket b \rrbracket (\llbracket a \rrbracket (n)) \otimes c \arrow[dr, "\xi_{c, \llbracket b \rrbracket (\llbracket a \rrbracket (n))}"] & \\
\llbracket c \rrbracket (\llbracket b \rrbracket (\llbracket a \rrbracket (n))) \arrow[rrrr, "id"'] & & & & \llbracket c \rrbracket (\llbracket b \rrbracket (\llbracket a \rrbracket (n)))
\end{tikzcd}$$

Case λ.

$$\begin{tikzcd}
n \otimes (e \otimes a) \arrow[rr, "id \otimes \lambda"] \arrow[ddr, "\alpha"'] \arrow[ddd, "\xi_{e \otimes a, n}"'] & & n \otimes a \arrow[ddd, "\xi_{a,n}"] \\
& \text{(coh)} & \\
& (n \otimes e) \otimes a \arrow[uur, "\rho \oplus id"'] & \\
\llbracket a \rrbracket (n) \arrow[rr, "id"'] & & \llbracket a \rrbracket (n)
\end{tikzcd}$$

Case ρ.

$$\begin{tikzcd}
n \otimes (a \otimes e) \arrow[rr, "id \otimes \rho"] \arrow[dr, "\alpha"'] \arrow[ddd, "\xi_{a \otimes e, n}"'] & & n \otimes a \arrow[ddd, "\xi_{a,n}"] \\
& (n \otimes a) \otimes e \arrow[ur, "\rho"'] \arrow[d, "\xi_{a,n} \otimes id"'] \arrow[from=1-1, to=1-3, phantom, shift right=8ex, "\text{(coh)}"] & \\
& \llbracket a \rrbracket(n) \otimes e \arrow[dl, "\xi_{e, \llbracket a \rrbracket(n)}"] \arrow[dr, "\rho"] \arrow[from=2-2, to=3-2, phantom, shift left=8ex, "\text{(nat)}"] & \\
\llbracket a \rrbracket(n) \arrow[rr, "id"'] & & \llbracket a \rrbracket(n)
\end{tikzcd}$$

In the last diagram the top triangle is the derived coherence equation (9) in MacLane [17, p 159].

4 Formalizing the proof in intuitionistic type theory

We have formalized the coherence proof in Martin-Löf's intuitionistic type theory using the proof assistant ALF developed in Göteborg by Coquand, Magnusson, Nordlander, and Nordström [2].

It is significant that ALF is an implementation of *intensional* intuitionistic type theory. The treatment of equality in this theory forces us to consider "metacoherence" problems: we shall need to use the fact that certain proofs of equality of objects are unique. As Hofmann and Streicher [13] has proved, it is not generally the case that there is a unique proof of intensional equality in intensional type theory. However, unicity holds for decidable intensional equality, see Hedberg [11].

The formalization will make the connection between normalization in monoids and coherence for monoidal categories precise. We will see formally how a monoid defined in type theory is almost like a monoidal category, since we introduce explicit proofs that two elements a and b of a monoid are equal and these explicit proofs correspond to arrows from a to b in a monoidal category. A "whole" monoidal category is then obtained by introducing an equivalence relation on such proofs.

We can also view the present formalization as part of the larger enterprise of developing a formal constructive category theory inside intuitionistic type theory, see for example Aczel [1], Huet and Saibi [15], and Dybjer and Gaspes [8]. We can compare this category theory to Rydeheard and Burstall's [18] Computational Category Theory. An essential difference is that the *programming language* of type theory, with dependent types and Curry-Howard interpretation of the logical constants, is much more expressive than a simply typed functional language like ML.

Below we explain the essential features of the formalization in type theory, but we do not recapitulate the whole proof in type-theoretic notation. In fact the description below is a rational reconstruction of the actual implementation. The interested reader can retrieve the implementation from `ftp.cs.chalmers.se` in the directory `~ilya/FMC`, see the `README`-file for details.

4.1 Equality in intensional type theory

We need to consider three kinds of equality in intensional type theory.

Firstly, we have *definitional equality* expressed by the *equality judgement* $a = b : A$. Two expressions are definitionally equal iff they have the same normal form. When normalizing an expression we use both β, η, and equations coming from recursive function definitions. Definitionally equal objects can be substituted for each other everywhere:

$$\frac{a = b : A \qquad \mathcal{J}[a]}{\mathcal{J}[b]},$$

where $\mathcal{J}[x]$ is an arbitrary judgement depending on x.

Secondly, we have basic *intensional equality* expressed by the *equality proposition* $I(A,a,b)$. This relation is inductively generated by the reflexivity axiom:

$$\frac{a : A}{r : I(A,a,a)}$$

(As for generalized algebraic theories we allow ourselves to omit arguments: proper notation for r is $r(A,a)$.)

I is an *intensional* equality, since we cannot prove

$$\forall x : A . I(B, f(x), g(x)) \to I(A \to B, f, g).$$

We have the following rule of substitutivity of equality for intensional equality:

$$\frac{c : I(A,a,b) \qquad d : P(a)}{I - elim(c,d) : P(b)}$$

Note that the conclusion depends on the *proof* c of equality. This is the source of the "metacoherence problem" we shall encounter later.

There is a definitional equation for $I - elim$:

$$I - elim(r,d) = d$$

Thirdly, it is often necessary to introduce a special equivalence relation which will play the role of equality on a certain set (a *book equality* in AUTOMATH terminology). Extensional equality of functions in a set $A \to B$ is one example. We shall follow Hofmann [12] and call such pairs of sets and equivalence relations *setoids*. It is necessary to work with setoids, since one cannot form a new set by taking the quotient of a set with respect to the equivalence relation. In intuitionistic type theory the term *set* is reserved for something which is inductively generated by "constructors" in essentially the same way as a "datatype" in a functional programming language. (We consider only "total", "well-founded" elements however.) (The situation is similar in Bishop's constructive mathematics [4], but he uses the terms *preset* and *set*, where we use *set* and *setoid* respectively.)

Furthermore, given two setoids $\langle A, \sim_A \rangle$ and $\langle B, \sim_B \rangle$, we will be interested in pairs of functions from A to B and proofs that the function preserves equivalence. We call such pairs *setoid-maps* (or just *maps*).

4.2 Monoids

A *monoid* in type theory can now be defined as a setoid $\mathcal{M} = \langle M, \sim_M \rangle$ with an element $e : M$ and a binary setoid-map $\circ$ on $\mathcal{M}$, such that the laws expressing that e is a unit and $\circ$ is associative are satisfied up to $\sim_M$.

Why could we not just have defined a monoid to be a *set* M together with an element $e : M$ and a binary function $\circ$, such that the laws expressing that e is a unit and $\circ$ is associative are satisfied up to *intensional equality*? The problem is that this would have ruled out certain constructions of monoids which are relevant to us. Both the free monoid M of binary trees under $\sim$ (as defined in section 2) and the monoid of endofunctions under extensional equality have setoids but not sets as carriers. There is in fact reason to believe that intensional equality between functions or between elements of transfinite sets, where branching is indexed by a function, is quite useless.

However, some monoids will be *strict* in the sense that the identity on the carrier setoid really is intensional equality. (This choice of terminology is intended to suggest that the distinction strict versus non-strict for algebraic notions in type theory is reminiscent of the distinction between strict and non-strict categorical notions, such as strict and relaxed monoidal categories.)

4.3 The normalization proof for monoids

The monoid N of binary words in normal form is an example of a strict monoid. However, there are no subsets in intensional type theory so we cannot define $N \subseteq M$. Instead we define N as the set of lists with elements in X and introduce an explicit injection $J : N \to M$. Hence the the normalization function is defined as follows:

$$Nf(a) = J([\![a]\!](e))$$

Apart from this there is no difficulty in principle in formalizing the normalization proof for monoids. See also Hedberg [10].

4.4 Monoidal categories

We follow Aczel, Huet and Saibi, and Dybjer and Gaspes and formalize a notion of category in intuitionistic type which does not have equality of objects as part of the structure. A *category* thus consists of a *set* of objects, but *setoids* of arrows indexed by pairs of objects. There is a family of identity arrows indexed by objects (that is, a *function* that to each object assigns a arrow), and a family of composition maps indexed by three objects.

Hence the object part of a *functor* is a *function* between object sets, whereas the arrow part is a *map* between hom-setoids in the appropriate way. Hence a functor can also be viewed as a triple: the object part, the arrow part and the part which witnesses the preservation of equality of arrows.

A *natural transformation* is defined as a family of arrows together with a proof of commutativity of the naturality diagram. Two natural transformations are *equal* iff their arrow components are extensionally equal. We can prove that functors and natural transformations under this equality form a category.

We have thus gone through all notions that the definition of a *relaxed monoidal category* refers to. Hence it is also clear how to define this notion inside type theory.

We end this subsection with the remark that a setoid in type theory is a notion which lies between the notions of set and category/groupoid. Similarly a monoid lies between the notions of strict monoid and monoidal category. This is the fundamental intuition which this paper is based on.

4.5 The coherence proof

Since there are no subsets in type theory, there are no subcategories either. So analogously to the monoid case we have to construct $\mathcal{M}$ and $\mathcal{N}$ independently and then define an injection functor $J : \mathcal{N} \to \mathcal{M}$. The objects of $\mathcal{N}$ are lists and the arrows are proofs of intensional equality:

$$\mathcal{N}(a, b) = I(N, a, b)$$

When we define J on morphisms, we do it by equality elimination, that is, by induction on the (unique) proof that $I(N, a, b)$:

$$J(r) = id_{J(a)}$$

for $r : I(N, a, a)$!

The proof that if $f : a \to b$ and $n : N$ then $I(N, [\![a]\!](n), [\![b]\!](n))$ is a natural transformation

$$[\![f]\!]_n : \mathcal{N}([\![a]\!](n), [\![b]\!](n))$$

that is, the arrow part of the functor $[\![\ ]\!]$. Observe that we cannot simply put

$$[\![f]\!]_n = id_{[\![a]\!](n)} = id_{[\![b]\!](n)}$$

as in the informal account, since we do not have the definitional equality $[\![a]\!](n) = [\![b]\!](n)$. The point is that even though $[\![f]\!]_n = id$ for any fixed f and n, we cannot write the general statement.

This forces a slight modification in the formalization of the proof of naturality of ξ. All base cases go through exactly like in the informal account, but the induction steps need to be modified as follows.

Consider first the diagram for the case of composition in the proof above:

$$\begin{tikzcd}
n \otimes a \arrow[rr, "id \otimes (g \circ f)"] \arrow[dr, "id \oplus f"] \arrow[dddd, "\xi_{a,n}"'] & & n \otimes c \arrow[dddd, "\xi_{c,n}"] \\
& n \otimes b \arrow[ur, "id \oplus g"] \arrow[dd, "\xi_{b,n}"] & \\
& & \\
& \llbracket b \rrbracket(n) \arrow[dr, "id"] & \\
\llbracket a \rrbracket(n) \arrow[ur, "id"] \arrow[rr, "id"'] & & \llbracket c \rrbracket(n)
\end{tikzcd}$$

The corresponding type-theoretic diagram is

$$\begin{tikzcd}
n \otimes a \arrow[rr, "id \otimes (g \circ f)"] \arrow[dr, "id \oplus f"] \arrow[dddd, "\xi_{a,n}"'] & & n \otimes c \arrow[dddd, "\xi_{c,n}"] \\
& n \otimes b \arrow[ur, "id \oplus g"] \arrow[dd, "\xi_{b,n}"] & \\
& & \\
& J(\llbracket b \rrbracket(n)) \arrow[dr, "J(\llbracket g \rrbracket_n)"] & \\
J(\llbracket a \rrbracket(n)) \arrow[ur, "J(\llbracket f \rrbracket_n)"] \arrow[rr, "J(\llbracket g \circ f \rrbracket_n)"'] & & J(\llbracket c \rrbracket(n))
\end{tikzcd}$$

Note the difference between the two lower triangles. Previously commutativity followed directly from the identity laws. But in the type-theoretic formalization it depends on the functoriality of J and $\llbracket\ \rrbracket$. Differently put, we can say that the lower triangle presents a metacoherence problem. It relies on the fact that there is a unique proof that two objects in $\mathcal{M}$ or in $\mathcal{N}$ are intensionally equal. Hence there is also a unique arrow between two intensionally equal objects in $\mathcal{N}$.

We next consider the case for multiplication:

$$\begin{tikzcd}
n \otimes (a \otimes b) \arrow[rrr, "id \otimes (f \otimes g)"] \arrow[dr, "\alpha"] \arrow[ddd, swap, "\xi_{a \otimes b, n}"] & & & n \otimes (a' \otimes b') \arrow[dl, swap, "\alpha"] \arrow[ddd, "\xi_{a' \otimes b', n}"] \\
& (n \otimes a) \otimes b \arrow[r, "(id \otimes f) \otimes g"] \arrow[d, swap, "\xi_{a,n} \otimes id"] & (n \otimes a') \otimes b' \arrow[d, "\xi_{a',n} \otimes id"] & \\
& \llbracket a \rrbracket(n) \otimes b \arrow[r, swap, "id \otimes g"] \arrow[dl, "\xi_{b, \llbracket a \rrbracket(n)}"] & \llbracket a' \rrbracket(n) \otimes b' \arrow[dr, "\xi_{b', \llbracket a' \rrbracket(n)}"] & \\
\llbracket b \rrbracket(\llbracket a \rrbracket(n)) \arrow[rrr, swap, "id"] & & & \llbracket b' \rrbracket(\llbracket a' \rrbracket(n))
\arrow[from=1-2, to=2-3, phantom, "\text{(nat)}"]
\arrow[from=2-2, to=3-3, phantom, "\text{(ind)}"]
\arrow[from=3-2, to=4-3, phantom, "\text{(ind)}"]
\end{tikzcd}$$

In the type-theoretic formalization this becomes

$$\begin{tikzcd}
n \otimes (a \otimes b) \arrow[rrr, "id \otimes (f \otimes g)"] \arrow[dr, "\alpha"] \arrow[ddd, swap, "\xi_{a \otimes b, n}"] & & & n \otimes (a' \otimes b') \arrow[dl, swap, "\alpha"] \arrow[ddd, "\xi_{a' \otimes b', n}"] \\
& (n \otimes a) \otimes b \arrow[r, "(id \otimes f) \otimes g"] \arrow[d, swap, "\xi_{a,n} \otimes id"] & (n \otimes a') \otimes b' \arrow[d, "\xi_{a',n} \otimes id"] & \\
& J(\llbracket a \rrbracket(n)) \otimes b \arrow[r, swap, "J(\llbracket f \rrbracket_n) \otimes g"] \arrow[dl, "\xi_{b, \llbracket a \rrbracket(n)}"] & J(\llbracket a' \rrbracket(n)) \otimes b' \arrow[dr, "\xi_{b', \llbracket a' \rrbracket(n)}"] & \\
J(\llbracket b \rrbracket(\llbracket a \rrbracket(n))) \arrow[rrr, swap, "J(\llbracket f \otimes g \rrbracket_n)"] & & & J(\llbracket b' \rrbracket(\llbracket a' \rrbracket(n)))
\arrow[from=1-2, to=2-3, phantom, "\text{(nat)}"]
\arrow[from=2-2, to=3-3, phantom, "\text{(ind)}"]
\arrow[from=3-2, to=4-3, phantom, "\text{(ind)}"]
\end{tikzcd}$$

The difference is in the lower square. To prove that it commutes we analyze the type-theoretic version into

$$\begin{tikzcd}
J(\llbracket a \rrbracket(n)) \otimes b \arrow[rr, "J(\llbracket f \rrbracket_n) \otimes g"] \arrow[dr, "id \oplus g"] \arrow[ddd, swap, "\xi_{b, \llbracket a \rrbracket(n)}"] & & J(\llbracket a' \rrbracket(n)) \otimes b' \arrow[ddd, "\xi_{b', \llbracket a' \rrbracket(n)}"] \\
& J(\llbracket a \rrbracket(n)) \otimes b' \arrow[ur, "J(\llbracket f \rrbracket_n) \oplus id"] \arrow[d, "\xi_{b', \llbracket a \rrbracket(n)}"] & \\
& J(\llbracket b' \rrbracket(\llbracket a \rrbracket(n))) \arrow[dr, "J(\llbracket b \rrbracket(\llbracket f \rrbracket_n))"] & \\
J(\llbracket b \rrbracket(\llbracket a \rrbracket(n))) \arrow[rr, swap, "J(\llbracket f \otimes g \rrbracket_n)"] \arrow[ur, "J(\llbracket g \rrbracket_{\llbracket a \rrbracket(n)})"] & & J(\llbracket b' \rrbracket(\llbracket a' \rrbracket(n)))
\end{tikzcd}$$

It may seem as though the type-theoretic proof is significantly more complex, but this is not quite true. Any formal proof would need to perform some reasoning, which is not explicitly represented in the original diagram, about substitutivity of equality.

5 Experience of the ALF-implementation

The basic message of this paper is that a proof of coherence for monoidal categories is implicit in a proof of normalization for monoids. The additional work just involves checking the naturality of ν. This work is fairly "mechanical" for a human – no ingenuity is required. The reader should not be mislead by the size of the diagrams here, they analyze each case in great detail.

What did we then learn from mechanizing the proof in ALF? Firstly, we had to understand the issue of metacoherence. This was the essential theoretical insight which was needed. But there were also several practical problems, which made the task of mechanizing the essentially "mechanical"(!) proof quite tedious. These problems arose in part because ALF is not tailored for equational reasoning and for handling the large terms which arise in a type-theoretic formalization of category theory. But there is reason to believe that there is scope for improvement here – these deficiencies should not be intrinsic problems of type theory.

For example, an essential part of our proof consists in chasing a few non-trivial diagrams. When doing this each equality inference had to be given explicitly to the machine including informally trivial steps using transitivity, congruence, associativity of composition, etc. which are hidden in the diagrammatic notation.

ALF stores proof terms in their full "monomorphic" form, even if some subexpressions can be deduced from others. For instance, the frequently used rule that expresses congruence with respect to composition is

$$\frac{A, B, C : Obj \quad f, f' : Hom(B, C) \quad g, g' : Hom(A, B) \quad p : Eq(B, C, f, f') \quad q : Eq(A, B, g, g')}{Eq(A, C, o(A, B, C, g, f), o(A, B, C, g', f'))}$$

in the full form maintained by the implementation. Only the last two parameters (p and q) matter here, the others are not controlled by the user and usually hidden. In the present application the hidden parts of terms tend to dominate (the internal representation of a term can be 20 times bigger than the "polymorphic" term displayed on the screen).

6 Related work

The present work can be seen as an instance of a certain approach to normalization in logical calculi: so-called "reduction-free" normalization [3, 7, 6]. The idea is to construct an appropriate model of the calculus and a function which inverts the interpretation function. Here the appropriate model is the category $\mathcal{N}^{\mathcal{N}}$ and the inversion functor is application to the unit. Another proof of coherence in this style is Lafont's for cccs [16].

We would also like to mention the proof of coherence for monoidal categories by Huet [14]. In contrast to ours his approach is reduction-based and uses the method of Knuth-Bendix completion from rewriting theory.

Acknowledgement. The authors wish to thank Michael Hedberg for his thorough analysis of normalization of the associative law which formed the basis for the present work.

References

[1] P. Aczel. Galois: a theory development project. A report on work in progress for the Turin meeting on the Representation of Logical Frameworks, 1993.

[2] T. Altenkirch, V. Gaspes, B. Nordström, and B. von Sydow. A user's guide to ALF. Draft, January 1994.

[3] U. Berger and H. Schwichtenberg. An inverse to the evaluation functional for typed λ-calculus. In *Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science, Amsterdam*, pages 203–211, July 1991.

[4] E. Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, 1967.

[5] J. Cartmell. Generalized algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[6] C. Coquand. From semantics to rules: a machine assisted analysis. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Proceedings of CSL '93, LNCS 832*, 1993.

[7] T. Coquand and P. Dybjer. Intuitionistic model constructions and normalization proofs. Preliminary Proceedings of the 1993 TYPES Workshop, Nijmegen, 1993.

[8] P. Dybjer and V. Gaspes. Implementing a category of sets in ALF. Manuscript, 1993.

[9] R. Gordon, A. J. Power, and R. Street. Coherence for tricategories. In *Memoirs of the American Mathematical Society.* To appear.

[10] M. Hedberg. Normalizing the associative law: an experiment with Martin-Löf's type theory. *Formal Aspects of Computing*, pages 218–252, 1991.

[11] M. Hedberg. Uniqueness and internal decidability in type theory. Draft paper, 1995.

[12] M. Hofmann. Elimination of extensionality and quotient types in Martin-Löf's type theory. In *Types for Proofs and Programs, International Workshop TYPES'93, LNCS 806*, 1994.

[13] M. Hofmann and T. Streicher. A groupoid model refutes uniqueness of identity proofs. In *LICS 1994*, pages 208–212. IEEE Press, 1994.

[14] G. Huet. Initiation à la Théorie des Catégories. Notes de cours du DEA Fonctionnalité, Structures de Calcul et Programmation donné à l'Université Paris VII en 1983-84 et 1984-85, 1987.

[15] G. Huet and A. Saibi. Constructive category theory. In *Proceedings of the Joint CLICS-TYPES Workshop on Categories and Type Theory, Göteborg*, January 1995.

[16] Y. Lafont. *Logique, Categories & Machines. Implantation de Langages de Programmation guidée par la Logique Catégorique.* PhD thesis, l'Universite Paris VII, January 1988.

[17] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag, 1971.

[18] D. E. Rydeheard and R. M. Burstall. *Computational Category Theory.* Prentice Hall, 1988.