Extracting a Proof of Coherence for Monoidal Categories from a Proof of Normalization for Monoids

Ilya Beylin and Peter Dybjer

Department of Computing Science
Chalmers University of Technology
Göteborg, Sweden

Abstract. This paper studies the problem of coherence in category theory from a type-theoretic viewpoint. 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 this coherence proof in intensional intuitionistic type theory and show how it relies on explicit reasoning about proof objects for intensional equality. This formalization has been checked in the proof assistant ALF.

1 Introduction

Mac Lane [18, pp.161–165] proved a coherence theorem for monoidal categories. A basic ingredient in his proof is the normalization of object expressions. But it is only one ingredient and several others are needed too.

Here we show that almost a whole proof of this coherence theorem is hidden in a Curry-Howard interpretation of a proof of normalization for monoids.

The second point of the paper is to contribute to the development of constructive category theory in the sense of Huet and Saibi [16], who implemented part of elementary category theory in the proof assistant Coq. Here we extend the scope of constructive category theory to the area of coherence theorems (cf. also [9]). We have formalized our proof in Martin-Löf type theory and implemented it in the proof assistant ALF. An interesting aspect of this formalization is that the problem of reasoning about explicit proofs of equality in the object language (arrows in a free monoidal category) reduces to reasoning about explicit proofs of equality in the metalanguage (proof objects for intensional equality I).

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. In Section 4 we show how the proof can be formalized in intuitionistic type theory. Section 5 contains a few remarks about the implementation in ALF. Section 6 is about related work.

The ALF-implementation can be found on the web [11]. More discussion and a comparison with an implementation of the same proof in HOL can be found in Agerholm, Beylin, and Dybjer [2].

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

$$\begin{aligned} e &\in M \\ x &\in M \quad \text{for any } x \in X \\ a \otimes b &\in M \quad \text{for any } a, b \in M \end{aligned}$$

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 generated by X.

Moreover, 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 shall analyze the proof of the following "obvious" normalization theorem (see Hedberg [13]):

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

A simple way to construct such a function is by using that N^N together with function composition and the identity function forms a monoid. So let $Nf(a) = [\![a]\!](e)$, where $[\![\ ]\!] : M \to N^N$ is defined by

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

The theorem now follows from the following two lemmas:

Lemma 2. *If $a \sim b$ then $[\![a]\!] = [\![b]\!]$ and $Nf(a) = Nf(b)$.*

Proof. By induction on the proof of $a \sim b$.

Lemma 3. *$a \sim Nf(a)$.*

Proof. $a \sim e \otimes a \sim [\![a]\!](e) = Nf(a)$ using the following auxiliary lemma:

Lemma 4. *$n \otimes a \sim [\![a]\!](n)$.*

Proof. By induction on a.

In the next section we shall see how a kind of Curry-Howard interpretation of this proof yields a proof of coherence for monoidal categories.

3 Coherence for Monoidal Categories

3.1 Monoidal Categories

Definition 5. A *monoidal category* consists of a category $\mathcal{C}$; a bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$; an object e; and three natural isomorphisms

$$\begin{aligned} \alpha_{a,b,c} &: a\otimes(b\otimes c) \longrightarrow (a\otimes b)\otimes c \\ \lambda_a &: e\otimes a \longrightarrow a \\ \rho_a &: a\otimes e \longrightarrow a \end{aligned}$$

such that the following diagrams (called the coherence conditions) commute:

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

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

Mac Lane's [18, pp 158-159] definition contained a third coherence condition $\lambda_e = \rho_e$ which was later proved to be redundant by Kelly. Our proof of coherence does not use this condition.

Definition 6. The category $\mathcal{M}$ has elements (binary words) of M as objects. Its Hom-sets are equivalence classes of arrow expressions. The arrow expressions are inductively generated as follows:

$g \circ f : a \longrightarrow c$	if $g : b \longrightarrow c$ and $f : a \longrightarrow b$
$id : a \longrightarrow a$	if a is an object
$f\otimes g : a\otimes b \longrightarrow a'\otimes b'$	if $f : a \longrightarrow a'$ and $g : b \longrightarrow b'$
$\alpha : a\otimes(b\otimes c) \longrightarrow (a\otimes b)\otimes c$	if a, b, c are objects
$\alpha^{-1} : (a\otimes b)\otimes c \longrightarrow a\otimes(b\otimes c)$	if a, b, c are objects
$\lambda : e\otimes a \longrightarrow a$	if a is an object
$\lambda^{-1} : a \longrightarrow e\otimes a$	if a is an object
$\rho : a\otimes e \longrightarrow a$	if a is an object
$\rho^{-1} : a \longrightarrow a\otimes e$	if a is an object

Equivalence of arrow expression is the congruence relation inductively generated by associativity and identity laws (making $\mathcal{M}$ a category with composition $\circ$ and identity id); the interchange laws (making $\otimes$ a bifunctor); laws making α, λ and ρ natural isomorphisms with inverses α^{-1}, λ^{-1}, and ρ^{-1} respectively; and finally the two coherence conditions. (See also the ALF implementation of $\mathcal{M}$ in section 4.)

We will use arrow expressions to denote arrows in $\mathcal{M}$.

Proposition 7. *(i) $\mathcal{M}$ is a free monoidal category generated by X. (ii) $\mathcal{M}$ is a groupoid, that is, all its arrows are isomorphisms. (iii) There is an arrow $f : a \longrightarrow b$ in $\mathcal{M}$ iff $a \sim b$ in M.*

Proof. (i) $\mathcal{M}$ is a free monoidal category by construction.

(ii) follows directly by induction on arrows of $\mathcal{M}$.

(iii) By induction on f and, in the other direction, on the proof that $a \sim b$.

3.2 The Coherence Theorem

Theorem 8. *If $f, f' : a \longrightarrow b$ are arrows in $\mathcal{M}$ then $f = f'$.*

Before we prove the theorem we note that it implies the coherence theorem as formulated by Mac Lane. He defines a category $\mathcal{W}$ which has the same objects as $\mathcal{M}$ (when $X = \{\square\}$). Moreover, there is a unique arrow between two binary words in $\mathcal{W}$ iff the two words have the same length. So $\mathcal{W}$ is a preorder with every arrow invertible. Mac Lane's coherence theorem ([18, p 162, theorem 1]) states that $\mathcal{W}$ is a free monoidal category. But our theorem entails that $\mathcal{M}$ is isomorphic to $\mathcal{W}$ so Mac Lane's theorem is a corollary.

Proof of Theorem 8. Let $\mathcal{N}$ be the set N considered as a discrete category. The coherence theorem follows immediately from the categorical counterparts of Lemmas 2 and 3: that Nf can be extended to a functor $Nf : \mathcal{M} \to \mathcal{N}$ (Lemma 9) and that there is a natural isomorphism $\nu_a : a \cong Nf(a)$ (Lemma 10). Indeed, if $f, f' : a \longrightarrow b$, then $f = \nu_b^{-1} \circ id \circ \nu_a = f'$ because of naturality of ν:

$$\begin{tikzcd}
a \arrow[r, shift left, "f"] \arrow[r, shift right, "f'"'] \arrow[d, "\nu_a"'] \arrow[dr, phantom, "\text{(nat)}"] & b \arrow[d, "\nu_b"] \\
Nf(a) \arrow[r, "id"'] & Nf(b)
\end{tikzcd}$$

Lemma 9. *The functions $Nf : M \to N$ and $[\![\ ]\!] : M \to N^N$ can be extended to functors $Nf : \mathcal{M} \to \mathcal{N}$ and $[\![\ ]\!] : \mathcal{M} \to \mathcal{N}^{\mathcal{N}}$.*

Proof. Follows immediately from Lemma 2.

Lemma 10. *There is a natural isomorphism $\nu_a : a \cong Nf(a)$.*

Proof. Let

$$\nu_a : a \xrightarrow{\lambda^{-1}} e \otimes a \xrightarrow{\xi_a^e} [\![a]\!] e$$

where ξ is an auxiliary natural isomorphism constructed in Lemma 11.(For typographical reasons we write its components as ξ_a^n rather than $\xi_{n,a}$.) Then naturality of ν follows from the naturality of ξ:

$$\begin{tikzcd}
a \arrow[rrr, "f"] \arrow[dddd, "\nu_a"'] \arrow[ddr, "\lambda^{-1}"] & & & b \arrow[dddd, "\nu_b"] \arrow[ddl, "\lambda^{-1}"'] \\
& \arrow[r, phantom, "((\mathrm{nat})_\lambda)"] & & \\
& e\otimes a \arrow[r, "id\otimes f"] \arrow[ddl, "\xi^e_a"'] & e\otimes b \arrow[ddr, "\xi^e_b"] & \\
& \arrow[r, phantom, "((\mathrm{nat})_\xi)"] & & \\
Nf(a) \arrow[rrr, "id"'] & & & Nf(b)
\end{tikzcd}$$

Lemma 11. *There is a natural isomorphism* $\xi^n_a : n\otimes a \cong \llbracket a\rrbracket(n)$.

Proof. ξ^n_a is defined by recursion on a:

$$\begin{aligned}
\xi^n_{a\otimes b} &: n\otimes(a\otimes b) \xrightarrow{\alpha} (n\otimes a)\otimes b \xrightarrow{\xi^n_a\otimes id} \llbracket a\rrbracket(n)\otimes b \xrightarrow{\xi^{\llbracket a\rrbracket n}_b} \llbracket b\rrbracket(\llbracket a\rrbracket(n)) = \llbracket a\otimes b\rrbracket n \\
\xi^n_e &: n\otimes e \xrightarrow{\rho} e \\
\xi^n_x &: n\otimes x \xrightarrow{id} n\otimes x
\end{aligned}$$

There is a dual definition of $(\xi^n_a)^{-1}$.

We prove that ξ^n_a is natural in a by induction on arrow expressions in $\mathcal{M}$. (Naturality in n is trivial, since $\mathcal{N}$ is a discrete category.) We justify each case of the induction by 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 $g \circ f$.

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

Case id .

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

Case $f\otimes g$.

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

Case α .

$$\begin{tikzcd}
n\otimes(a\otimes(b\otimes c)) \arrow[rrrr, "id\otimes\alpha"] \arrow[dr, "\alpha"] \arrow[ddd, "\xi^n_{a\otimes(b\otimes c)}"'] \arrow[from=1-1, to=2-5, phantom, "\text{(coh)}"] & & & & n\otimes((a\otimes b)\otimes c) \arrow[dl, "\alpha"'] \arrow[ddd, "\xi^n_{(a\otimes b)\otimes c}"] \\
& (n\otimes a)\otimes(b\otimes c) \arrow[r, "\alpha"] \arrow[d, "\xi^n_a\otimes id"'] \arrow[dr, phantom, "\text{(nat)}"] & ((n\otimes a)\otimes b)\otimes c \arrow[d, "(\xi^n_a\otimes id)\otimes id"] & (n\otimes(a\otimes b))\otimes c \arrow[l, "\alpha\otimes id"'] \arrow[d, "\xi^n_{a\otimes b}\otimes id"] & \\
& \llbracket a\rrbracket(n)\otimes(b\otimes c) \arrow[r, "\alpha"'] \arrow[dl, "\xi^{\llbracket a\rrbracket(n)}_{b\otimes c}"] & (\llbracket a\rrbracket(n)\otimes b)\otimes c \arrow[r, "\xi_b^{\llbracket a\rrbracket(n)}\otimes id"'] & \llbracket b\rrbracket(\llbracket a\rrbracket(n))\otimes c \arrow[dr, "\xi_c^{\llbracket b\rrbracket(\llbracket a\rrbracket(n))}"'] & \\
\llbracket c\rrbracket(\llbracket b\rrbracket(\llbracket a\rrbracket(n))) \arrow[rrrr, "id"'] & & & & \llbracket c\rrbracket(\llbracket b\rrbracket(\llbracket a\rrbracket(n)))
\end{tikzcd}$$

Cases λ *and* ρ .

$$\begin{tikzcd}
n\otimes(e\otimes a) \arrow[rr, "id\otimes\lambda"] \arrow[dr, "\alpha"'] \arrow[dd, "\xi^n_{e\otimes a}"'] \arrow[rr, phantom, shift right=4ex, "(coh)"] & & n\otimes a \arrow[dd, "\xi^n_a"] \\
& (n\otimes e)\otimes a \arrow[ur, "\rho\oplus id"'] & \\
\llbracket a\rrbracket(n) \arrow[rr, "id"'] & & \llbracket a\rrbracket(n)
\end{tikzcd}
\qquad
\begin{tikzcd}
n\otimes(a\otimes e) \arrow[rr, "id\otimes\rho"] \arrow[dr, "\alpha"'] \arrow[ddd, "\xi^n_{a\otimes e}"'] \arrow[rr, phantom, shift right=4ex, "(coh)"] & & n\otimes a \arrow[ddd, "\xi^n_a"] \\
& (n\otimes a)\otimes e \arrow[ur, "\rho"'] \arrow[d, "\xi^n_a\otimes id"'] \arrow[d, phantom, shift left=8ex, "(nat)"] & \\
& \llbracket a\rrbracket(n)\otimes e \arrow[dl, "\xi_e^{\llbracket a\rrbracket(n)}"'] \arrow[dr, "\rho"] & \\
\llbracket a\rrbracket(n) \arrow[rr, "id"'] & & \llbracket a\rrbracket(n)
\end{tikzcd}$$

In the last diagram the top triangle is the derived coherence equation (9) in Mac Lane [18, p 159].

4 Formalizing the Proof in Intuitionistic Type Theory

We have formalized the coherence proof in Martin-Löf intuitionistic type theory using the proof assistant ALF developed in Göteborg by Coquand, Magnusson, Nordlander, and Nordström [3].

When we formalize the free monoid in type theory we introduce explicit proofs that two elements a and b of a monoid are equal. These proofs correspond to arrow expressions in the free monoidal category $\mathcal{M}$. To get the full definition of $\mathcal{M}$ from M we just need to add the definition of equivalence of arrow expressions. Moreover, ν_a and ξ^n_a will appear as proof objects of $a \sim Nf(a)$ and $n\otimes a \sim \llbracket a\rrbracket(n)$. To show coherence it essentially only remains to show naturality.

Below we explain the essential features of the formalization, 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 [11].

We begin by reviewing the treatment of equality in intensional type theory. Then we show how to formalize monoids and monoidal categories. Finally, we show how to refine the informal proof into a formal proof in type theory. *Remark.* Throughout this section standard mathematical terms, such as category, functor, etc., refer to their formalization in type theory unless stated otherwise.

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. 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:

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

(We often simplify notation by omitting some arguments. For example, the proper form of r is $r(A, a)$.) We have the following rule of substitutivity of intensional equality:

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

Note that the conclusion depends on the *proof* c of equality. There is a definitional equation for *I-elim*:

$$\textit{I-elim}(r, d) = d$$

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

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

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 $A \rightarrow B$ is one example. We shall follow Hofmann [14] 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 much the same way as a "datatype" in a functional programming language.

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 $\langle M, \sim_M \rangle$ with an element e and a binary map $\circ$, such that the laws expressing that e is a unit and $\circ$ is associative are satisfied up to $\sim_M$.

We call a monoid *strict* if $\sim_M$ is intensional equality I. The terminology is intended to suggest that the distinction between strict and non-strict monoids in type theory is reminiscent of the distinction between strict and non-strict categorical notions, such as strict and relaxed monoidal categories in category theory, where a strict monoidal category is one where the monoidal laws hold up to equality and not only up to isomorphism [18].

4.3 The Normalization Proof for Monoids

There is no difficulty in principle in formalizing the normalization proof for monoids. See also Hedberg [13]. A minor point is that we define N as the set of lists with elements in X and introduce an explicit injection $J : N \to M$. (There are no "true" subsets in type theory.) Hence the normalization function is defined as follows:

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

4.4 Monoidal Categories

We follow Aczel [1], Huet and Saibi [16], and Dybjer and Gaspes [10] 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.

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 *monoidal category* refers to. Hence it is also clear how to define this notion inside type theory. In Figure 1 we show the implementation of $\mathcal{M}$ and of the coherence proposition in ALF.

One can show that our formalization of monoidal categories is adequate with respect to the standard set-theoretic definition in the following sense. The essential idea is to use the naive interpretation in classical set theory of Martin-Löf type theory [8]. But we need to interpret the Hom-setoids as Hom-sets of equivalence classes rather than as set-theoretic setoids, and similarly for the interpretation of maps.

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 elements of N and the arrows are proofs of intensional equality:

$$\mathcal{N}(m, n) = I(N, m, n) \qquad [m, n : N]$$

We define J on arrows by equality elimination:

$$J(h) = \textit{I-elim}(h, id_{Jm}) : \mathcal{M}(Jm, Jn) \qquad [m, n : N;\ h : \mathcal{N}(m, n)],$$

Definition of $\mathcal{M}$

the object part

the arrow part

equivalence relation on the arrows

$\circ$-*laws*

$\otimes$-*laws*

naturality conditions

isomorphism

coherence conditions

Statement of Coherence theorem

Fig. 1. Excerpts of the ALF code

which implies that $J(h)$ must be an identity arrow. However, the equality $J(h) = id_{Jn}$ is ill-typed unless h has already type $\mathcal{N}(n,n)$.

Moreover, when we define the arrow part of the functor $\llbracket\ \rrbracket : \mathcal{M} \to \mathcal{N}^{\mathcal{N}}$, we cannot simply put

$$\llbracket f \rrbracket_n = id_{\llbracket a \rrbracket(n)} = id_{\llbracket b \rrbracket(n)} \qquad [a, b : M;\ f : \mathcal{M}(a,b)]$$

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

Instead we use proof objects which witness the *propositional equality* in N:

$$\llbracket f \rrbracket_n : \mathcal{N}(\llbracket a \rrbracket(n), \llbracket b \rrbracket(n)) \qquad [a, b : M;\ n : N;\ f : \mathcal{M}(a,b)]$$

We also have to verify that this family of arrows is natural in n.

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

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

$$\begin{tikzcd}
n \otimes a \arrow[rrrr, "id \otimes (g \circ f)"] \arrow[rrd, "id \otimes f"] \arrow[dddd, "\xi_a^n"'] & & & & n \otimes c \arrow[dddd, "\xi_c^n"] \\
& & n \otimes b \arrow[rru, "id \otimes g"'] \arrow[dd, "\xi_b^n"] & & \\
& \text{(ind)} & & \text{(ind)} & \\
& & \llbracket b \rrbracket(n) \arrow[rrd, "id"] & & \\
\llbracket a \rrbracket(n) \arrow[rru, "id"] \arrow[rrrr, "id"'] & & & & \llbracket c \rrbracket(n)
\end{tikzcd}$$

informal version

$$\begin{tikzcd}
Jn \otimes a \arrow[rrrr, "id \otimes (g \circ f)"] \arrow[rrd, "id_{Jn} \otimes f"] \arrow[ddddd, "\xi_a^n"'] & & & & Jn \otimes c \arrow[ddddd, "\xi_c^n"] \\
& & Jn \otimes b \arrow[rru, "id_{Jn} \otimes g"'] \arrow[dd, "\xi_b^n"] & & \\
& \text{(ind)} & & \text{(ind)} & \\
& & J(\llbracket b \rrbracket(n)) \arrow[rrdd, "J(\llbracket g \rrbracket_n)"] & & \\
& & (\circ) & & \\
J(\llbracket a \rrbracket(n)) \arrow[rruu, "J(\llbracket f \rrbracket_n)"] \arrow[rrrr, "J(\llbracket g \circ f \rrbracket_n)"'] & & & & J(\llbracket c \rrbracket(n))
\end{tikzcd}$$

type-theoretic version

Note the difference between the two lower triangles. Previously commutativity followed directly from the identity law. But in the type-theoretic formalization we have to appeal to the functoriality of J and $\llbracket\ \rrbracket$.

We next consider the case for multiplication:

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

informal version

$$\begin{tikzcd}
Jn\otimes(a\otimes b) \arrow[rrrr, "id\otimes(f\otimes g)"] \arrow[dr, "\alpha"] \arrow[dddddd, "\xi^n_{a\otimes b}"'] & & & & Jn\otimes(a'\otimes b') \arrow[dl, "\alpha"'] \arrow[dddddd, "\xi^n_{a'\otimes b'}"] \\
 & & \text{(nat)} & & \\
 & (Jn\otimes a)\otimes b \arrow[rr, "(id\otimes f)\otimes g"] \arrow[dd, "\xi^n_a\otimes id"'] & & (Jn\otimes a')\otimes b' \arrow[dd, "\xi^n_{a'}\otimes id"] & \\
 & & \text{(ind)} & & \\
 & J(\llbracket a\rrbracket(n))\otimes b \arrow[rr, "J(\llbracket f\rrbracket_n)\otimes g"'] \arrow[ddl, "\xi_b^{\llbracket a\rrbracket(n)}"] & & J(\llbracket a'\rrbracket(n))\otimes b' \arrow[ddr, "\xi_{b'}^{\llbracket a'\rrbracket(n)}"'] & \\
 & & (*) & & \\
J(\llbracket b\rrbracket(\llbracket a\rrbracket(n))) \arrow[rrrr, "J(\llbracket f\otimes g\rrbracket_n)"'] & & & & J(\llbracket b'\rrbracket(\llbracket a'\rrbracket(n)))
\end{tikzcd}$$

type-theoretic version

The difference is in the lower trapezium (*). To prove that it commutes we analyze the type-theoretic version into

$$\begin{tikzcd}
J(\llbracket a\rrbracket(n))\otimes b \arrow[rr, "J(\llbracket f\rrbracket_n)\otimes g"] \arrow[dr, "id\otimes g"'] \arrow[dddd, "\xi_b^{\llbracket a\rrbracket(n)}"'] & \text{(fun)} & J(\llbracket a'\rrbracket(n))\otimes b' \arrow[dddd, "\xi_{b'}^{\llbracket a'\rrbracket(n)}"] \\
& J(\llbracket a\rrbracket(n))\otimes b' \arrow[ur, "J(\llbracket f\rrbracket_n)\otimes id"'] \arrow[d, "\xi_{b'}^{\llbracket a\rrbracket(n)}"] & \\
\text{(ind)} & J(\llbracket b'\rrbracket(\llbracket a\rrbracket(n))) \arrow[ddr, "J(\llbracket b'\rrbracket\llbracket f\rrbracket_n)"] & \text{(nat)} \\
& (\otimes) & \\
J(\llbracket b\rrbracket(\llbracket a\rrbracket(n))) \arrow[uur, "J(\llbracket g\rrbracket_{\llbracket a\rrbracket(n)})"] \arrow[rr, "J(\llbracket f\otimes g\rrbracket_n)"'] & & J(\llbracket b'\rrbracket(\llbracket a'\rrbracket(n)))
\end{tikzcd}$$

The right trapezium (nat) commutes because of naturality of ξ_b^n in n.

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 about substitutivity of equality, which is not explicitly represented in the diagram from the informal proof.

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 misled by the size of the diagrams here, they analyze each case in great detail.

What did we learn from mechanizing the proof in ALF? Firstly, we had to understand how to reason with explicit proofs of I-equality during diagram chasing, we so as to speak had to think of questions of "meta-coherence".

There were also several practical problems, which made the task of mechanizing the proof quite tedious. 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 full form of the `oE`-rule (congruence of `==` with respect to composition) is

$$\frac{\begin{array}{c} A,B,C\ :\ Obj \qquad g,g'\ :\ Hom(B,C) \qquad f,f'\ :\ Hom(A,B) \\ p\ :\ \ ==(B,C,g,g') \qquad q\ :\ \ ==(A,B,f,f') \end{array}}{==(A,C,o(A,B,C,g,f),o(A,B,C,g',f'))}$$

Only the last two parameters (p and q) matter here, the others are not controlled by the user and usually hidden (cf. the compact definition on Figure 1). In our application the superfluous parts of terms tended to dominate (the internal representation of a term was sometimes 20 times bigger than the "polymorphic" term displayed on the screen).

6 Related Work

Discussions with Martin Hyland and John Power revealed that the extracted proof is essentially a logical version of the proof of coherence for bicategories (in the special case of monoidal categories) given in the recent paper by Gordon, Power, and Street [12]. Their proof relies on Street's bicategorical Yoneda lemma. In our case a proof with similar structure was instead discovered by using the Curry-Howard interpretation which makes explicit the connection between the *formal proof* of normalization and the proof of coherence.

The present work can be seen as an application of a certain approach to normalization in logical calculi: so-called "reduction-free" normalization [5, 7, 6, 4]. 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 inversion is application to unit. Another proof of coherence in this style is Lafont's for *cccs* [17].

We would also like to mention the proof of coherence for monoidal categories by Huet [15]. In contrast to ours his approach is reduction-based and uses the method of Knuth-Bendix completion from rewriting theory.

Acknowledgement. The first author was supported by a grant from the Swedish Institute. The second author was supported in part by the ESPRIT projects TYPES and CLICS-II, by TFR (the Swedish Technical Research Council), and by the Isaac Newton Institute for Mathematical Sciences.

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. S. Agerholm, I. Beylin, and P. Dybjer. A comparison of HOL and ALF formalizations of a categorical coherence theorema. In *Theorem Proving in Higher Order Logic (HOL'96)*. Springer LNCS, 1996. To appear.
   Available on `http://www.cs.chalmers.se/~ilya/FMC/hol_alf.ps.gz`.
3. T. Altenkirch, V. Gaspes, B. Nordström, and B. von Sydow. A user's guide to ALF. Draft, January 1994.
4. T. Altenkirch, M. Hofmann, and T. Streicher. Categorical reconstruction of a reduction free normalization proof. In D. Pitt, D. E. Rydeheard, and P. Johnstone, editors, *Springer LNCS 953, Category Theory and Computer Science, 6th International Conference, CTCS '95, Cambridge, UK*, August 1995.

5. 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.
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. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In *Logical Frameworks*, pages 280–306. Cambridge University Press, 1991.
9. P. Dybjer. Internal type theory. In *TYPES '95, Types for Proofs and Programs*, Lecture Notes in Computer Science. Springer, 1996.
10. P. Dybjer and V. Gaspes. Implementing a category of sets in ALF. Manuscript, 1993.
11. Formal proof of coherence theorem. Home page. `http://www.cs.chalmers.se/~ilya/FMC/`.
12. R. Gordon, A. J. Power, and R. Street. Coherence for tricategories. In *Memoirs of the American Mathematical Society*. To appear.
13. M. Hedberg. Normalizing the associative law: an experiment with Martin-Löf's type theory. *Formal Aspects of Computing*, pages 218–252, 1991.
14. 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.
15. 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.
16. G. Huet and A. Saibi. Constructive category theory. In *Proceedings of the Joint CLICS-TYPES Workshop on Categories and Type Theory, Göteborg*, January 1995.
17. Y. Lafont. *Logique, Categories & Machines. Implantation de Langages de Programmation guidée par la Logique Catégorique*. PhD thesis, l'Universite Paris VII, January 1988.
18. S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

This article was processed using the LaTeX macro package with LLNCS style