

# Dyads, a generalisation of monads

Maarten Fokkinga

*University of Twente, Dept INF, PO Box 217, NL 7500 AE Enschede*

e-mail: fokkinga@cs.utwente.nl

Version of December 6, 1993

The concept of *dyad* is defined as the least common generalisation of *monads* and *co-monads*. So, taking some of the ingredients to be the identity, the concept specialises to the concept of monad, and taking other ingredients to be the identity it specialises to co-monads. Except for one axiom, all have a nice (“natural”) form.

## Introduction: monads

Let us first describe one way in which monads can be motivated. We shall motivate dyads by a similar argument later. (Other descriptions have been given by Barr and Wells [1] and Wadler [2].) We use a fairly standard notation:  $\mathcal{A}, \mathcal{B}, \dots$  denote categories,  $a, b, \dots$  objects,  $f, g, \dots$  arrows,  $F, G, \dots$  functors, greek letters denote natural transformations, and  $x, y, \dots$  various things. Composition is denoted in diagrammatic order:  $f ; g$  is normally written  $g \circ f$ .

**Example.** Let  $L$  be the list functor:  $La$  is the set of lists over  $a$ , and for  $f: a \rightarrow b$  we have  $Lf: La \rightarrow Lb$  as the well-known  $f$ -map. Given list producing functions  $f: a \rightarrow Lb$  and  $g: b \rightarrow Lc$ , we often see the list producing “composition”:

$$f ; Lg ; \# / \quad : \quad a \rightarrow Lc \quad .$$

Here  $\# /: LL \rightarrow L$  is the flattening, or concatenation, of lists of list into lists. Functions of type  $a \rightarrow Lb$ , for varying  $a, b$ , turn up frequently in actual programming. For example, unconditional list comprehensions can be described in that way:

$$x \mapsto [z \mid y \leftarrow fx; z \leftarrow gy] \quad = \quad f ; Lg ; \# / \quad .$$

Notice, moreover, the existence of a particularly nice list producing function of type  $a \rightarrow Lb$  with  $a = b$ , namely the singleton former  $\eta = \lambda x :: [x]$ . It has the property that  $f ; L\eta ; \# / = f = \eta ; Lf ; \# /$ .

**Generalisation.** The situation above can be described more elegantly in categorical terms as follows. Let  $F$  be an endofunctor. Under what conditions can we consider arrows of type  $f: a \rightarrow Fb$ , for varying  $a, b$ , to be arrows of type  $a \rightarrow b$  in another category? This question means, amongst others, that there must be a way to “compose” arrows  $f: a \rightarrow Fb$  and  $g: b \rightarrow Fc$  into an arrow of type  $a \rightarrow Fc$ , and that this composition is associative, and that there exists a function  $\eta_a: a \rightarrow Fa$  for each  $a$  that is the identity for the new “composition”.

The new category is known as the Kleisli category, and we shall indicate it by  $\mathcal{K}(F)$ , or simply  $\mathcal{K}$  if the functor  $F$  is understood. The construction of  $\mathcal{K}$  is straightforward. Define:

$$\begin{aligned}
a \text{ object in } \mathcal{K} &\equiv a \text{ object in the given category} \\
f \text{ arrow in } \mathcal{K} &\equiv f \text{ arrow in the given category} \\
&\quad \text{of type } a \rightarrow Fb \text{ for some } a, b \\
f: a \rightarrow_{\mathcal{K}} b &\equiv f: a \rightarrow Fa \\
f \ ;_{\mathcal{K}} g &= f \ ; Fg \ ; \mu \quad \text{whenever } f: a \rightarrow Fb, \quad g: b \rightarrow Fc \\
id_{\mathcal{K},a} &= \eta_a
\end{aligned}$$

where natural transformations  $\mu$  and  $\eta$  are assumed to exist:

$$\begin{aligned}
\mu &: FF \rightarrow F \\
\eta &: I \rightarrow F \quad .
\end{aligned}$$

The associativity of composition  $\ ;_{\mathcal{K}}$ , and the neutrality of  $id_{\mathcal{K}}$  for this composition, are equivalent to the following three properties of  $\mu$  and  $\eta$ :

$$\begin{aligned}
F\mu \ ; \mu &= \mu F \ ; \mu \\
F\eta \ ; \mu &= \eta F \ ; \mu = id F \quad .
\end{aligned}$$

The first two of these say that  $F\mu$  equals  $\mu F$ , and  $F\eta$  equals  $\eta F$ , when followed by  $\mu$ . Indeed, a proof of associativity reads:

$$\begin{aligned}
&(f \ ;_{\mathcal{K}} g) \ ;_{\mathcal{K}} h = f \ ;_{\mathcal{K}} (g \ ;_{\mathcal{K}} h) \\
\equiv &\quad \text{definition } \ ;_{\mathcal{K}} \\
&(f \ ; Fg \ ; \mu) \ ; Fh \ ; \mu = f \ ; F(g \ ; Fh \ ; \mu) \ ; \mu \\
\equiv &\quad \text{category, functor} \\
&f \ ; Fg \ ; \mu \ ; Fh \ ; \mu = f \ ; Fg \ ; FFh \ ; F\mu \ ; \mu \\
\equiv &\quad \text{in lhs: naturality } \mu \\
&f \ ; Fg \ ; FFh \ ; \mu F \ ; \mu = f \ ; Fg \ ; FFh \ ; F\mu \ ; \mu \\
\equiv &\quad \text{for } \Leftarrow: \text{Leibniz; for } \Rightarrow: \text{instantiate with } f, g, h \text{ all equal to } id \\
&\mu F \ ; \mu = F\mu \ ; \mu \quad .
\end{aligned}$$

A proof of neutrality of  $id_{\mathcal{K}}$  for  $;\kappa$  reads:

$$\begin{aligned}
& f ;_{\kappa} id_{\mathcal{K}} = f = id_{\mathcal{K}} ; f \\
\equiv & \quad \text{definition } ;_{\kappa} \text{ and } id_{\mathcal{K}} \\
& f ; F\eta ; \mu = f = \eta ; Ff ; \mu \\
\equiv & \quad \text{naturality } \eta \\
& f ; F\eta ; \mu = f = f ; \eta F ; \mu \\
\equiv & \quad \text{for } \Leftarrow : \text{Leibniz}; \text{ for } \Rightarrow : \text{instantiate with } f = id \\
& F\eta ; \mu = idF = \eta F ; \mu \quad .
\end{aligned}$$

The three remaining category axioms for  $\mathcal{K}$  read:

$$\begin{aligned}
& f: a \rightarrow_{\kappa} b, \quad g: b \rightarrow_{\kappa} c \quad \Rightarrow \quad f ;_{\kappa} g: a \rightarrow_{\kappa} c \\
& id_{\mathcal{K}}: a \rightarrow_{\kappa} a \\
& f: a \rightarrow_{\kappa} b, \quad f: a' \rightarrow_{\kappa} b' \quad \Rightarrow \quad a = a' \quad \wedge \quad b = b' \quad .
\end{aligned}$$

The first two of these are evidently true; the latter one isn't true in general. So the construction yields a pre-category, and by a standard construction one obtains a category, the Kleisli category  $\mathcal{K}$ .

In summary, the ingredients that enable the construction of a Kleisli category are:

$$\begin{aligned}
& (F, \quad \mu: FF \rightarrow F, \quad \eta: Id \rightarrow F) \\
& \text{satisfying} \\
& F\mu ; \mu = \mu F ; \mu \\
& F\eta ; \mu = \eta F ; \mu = idF \quad .
\end{aligned}$$

Such a triple is known as a **monad**.

## A generalisation wanted

Loosely formulated the above construction of the Kleisli category enables us to “compose” arrows of type  $a \rightarrow Fb$  (considering them to be of type  $a \rightarrow' b$ ) for fixed  $F$  and varying  $a, b$ . Doaitse Swierstra posed the problem of “composing” arrows of type  $a \times b \rightarrow Fc$ , for fixed  $F$  and varying  $a, b, c$ . The Kleisli construction doesn't help here.

Abstracting a little bit, the new problem is to construct a Kleisli-like category for arrows of type  $Fa \rightarrow Gb$  (considering them to be of type  $a \rightarrow' b$ ), for fixed  $F, G$  and varying  $a, b$ . Taking  $F = Id$  we get the original problem that is solved by Kleisli's construction and the existence of a monad for  $G$ . Taking  $G = Id$  we get the dual problem; solved by the dual of Kleisli's construction and the existence of a co-monad for  $F$ . Thus our current *problem* formulation is the least common generalisation of the “Kleisli problem” and its dual; and our *solution* will be the least common generalisation of the Kleisli construction and its dual. We will term the ingredients of the solution a **dyad**: the least common generalisation of a monad and a co-monad.

## Dyads

Let  $F, G$  be functors with a common source and a common target. We will construct a category  $\mathcal{D}(F, G)$ , or simply  $\mathcal{D}$ , whose arrows (of type  $a \rightarrow_{\mathcal{D}} b$ ) are the given arrows of type  $Fa \rightarrow Gb$ . To this end we define:

$$\begin{aligned}
 a \text{ object in } \mathcal{D} &\equiv a \text{ object in source category of } F, G \\
 f \text{ arrow in } \mathcal{D} &\equiv f \text{ arrow in target category of } F, G \\
 &\quad \text{of type } Fa \rightarrow Gb \text{ for some } a, b \\
 f: a \rightarrow_{\mathcal{D}} b &\equiv f: Fa \rightarrow Gb \\
 f \text{ ;}_{\mathcal{D}} g &= \mu; Ff; \gamma; Gg; \nu \quad (\text{explained below}) \\
 id_{\mathcal{D}} &= \eta,
 \end{aligned}$$

where natural transformations  $\gamma, \eta, \mu, \nu$  are assumed to exist:

$$\begin{aligned}
 \gamma &: FG \rightarrow GF && F, G\text{-commuting transformation} \\
 \eta &: F \rightarrow G && F\text{- to } G\text{-unit transformation} \\
 \mu &: F \rightarrow FF && F\text{-generating transformation} \\
 \nu &: GG \rightarrow G && G\text{-reducing transformation} .
 \end{aligned}$$

Notice that  $G$  and  $\nu$  play the role of  $F$  and  $\mu$  in the usual nomenclature for monads and the Kleisli construction. The above definition of  $f \text{ ;}_{\mathcal{D}} g$  is the simplest general way to combine  $f$  and  $g$  in that order into an arrow of type  $Fa \rightarrow Gc$ . Indeed, in order that a transformed  $f$  and transformed  $g$  both have the same ingredients in their target and source type, respectively,  $F$  has to be applied to  $f$  and  $G$  to  $g$ ; this gives  $Ff: FFa \rightarrow FGb$  and  $Gg: GFb \rightarrow GGc$ :

$$Fa \xrightarrow{?} FFa \xrightarrow{Ff} FGb \xrightarrow{?} GFb \xrightarrow{Gg} GGb \xrightarrow{?} Gb .$$

Now an arrow of type  $Fa \rightarrow Gc$  may be obtained by an  $F$ -generating transformation first, followed by  $Ff$  and  $Gg$  with an  $F, G$ -commuting transformation in between, and a  $G$ -reducing transformation at the end:

$$Fa \xrightarrow{\mu} FFa \xrightarrow{Ff} FGb \xrightarrow{\gamma} GFb \xrightarrow{Gg} GGb \xrightarrow{\nu} Gb .$$

Clearly, the typing axioms, except for uniqueness of typing, are satisfied:

$$\begin{aligned}
 f: a \rightarrow_{\mathcal{D}} b, \quad g: b \rightarrow_{\mathcal{D}} c &\Rightarrow f \text{ ;}_{\mathcal{D}} g: a \rightarrow_{\mathcal{D}} c \\
 id_{\mathcal{D}, a}: a \rightarrow_{\mathcal{D}} a &.
 \end{aligned}$$

So in order to prove that  $\mathcal{D}$  is a pre-category, it remains to show that  $\text{ ;}_{\mathcal{D}}$  is associative and  $id_{\mathcal{D}}$  is neutral for  $\text{ ;}_{\mathcal{D}}$ . We shall now give those proofs, assuming suitable properties on

$\gamma, \eta, \mu, \nu$  along the way. First, for associativity, let  $f: a \rightarrow_{\mathcal{D}} b$ ,  $g: b \rightarrow_{\mathcal{D}} c$ , and  $h: c \rightarrow_{\mathcal{D}} d$  be arbitrary; then:

$$\begin{aligned}
& f \circ_{\mathcal{D}} (g \circ_{\mathcal{D}} h) = (f \circ_{\mathcal{D}} g) \circ_{\mathcal{D}} h \\
\equiv & \text{definition } \mathcal{D} \\
& Fa \xrightarrow{\mu} FFa \xrightarrow{Ff} FGb \xrightarrow{\gamma} GFb \xrightarrow{G\mu} GFFb \xrightarrow{GFg} FGc \xrightarrow{G\gamma} GGFc \xrightarrow{GGh} GGGd \xrightarrow{G\nu} GGd \xrightarrow{\nu} Gd \\
& \parallel \hspace{15em} \parallel \text{ commutes} \\
& Fa \xrightarrow{\mu} FFa \xrightarrow{F\mu} FFFa \xrightarrow{FFf} FFGb \xrightarrow{F\gamma} FGFb \xrightarrow{FGg} FGGc \xrightarrow{F\nu} FGc \xrightarrow{\gamma} GFc \xrightarrow{Gh} GGd \xrightarrow{\nu} Gd \\
(*) \quad \Leftarrow & \quad \{ \text{in order to split the terms into several parts at isomorphic objects:} \} \\
& \quad \mathbf{assume } \gamma \text{ has inverse } \check{\gamma}, \text{ so that } \gamma F: FGF \rightarrow GFF \text{ has inverse } \check{\gamma}F \\
& Fa \xrightarrow{\mu} FFa \xrightarrow{Ff} FGb \xrightarrow{\gamma} GFb \xrightarrow{G\mu} GFFb \xrightarrow{GFg} FGc \xrightarrow{G\gamma} GGFc \xrightarrow{GGh} GGGd \xrightarrow{G\nu} GGd \xrightarrow{\nu} Gd \\
& \parallel \hspace{15em} \parallel \text{ commutes} \\
& \hspace{10em} \gamma F \parallel \check{\gamma} F \quad \check{\gamma} G \parallel \gamma G \\
& Fa \xrightarrow{\mu} FFa \xrightarrow{F\mu} FFFa \xrightarrow{FFf} FFGb \xrightarrow{F\gamma} FGFb \xrightarrow{FGg} FGGc \xrightarrow{F\nu} FGc \xrightarrow{\gamma} GFc \xrightarrow{Gh} GGd \xrightarrow{\nu} Gd \\
\equiv & \text{diagram notation} \\
& \mu ; Ff ; \gamma ; G\mu = \mu ; F\mu ; FFf ; F\gamma ; \gamma F \quad \wedge \\
& GFg ; \check{\gamma}G = \check{\gamma}F ; FGg \quad \wedge \\
& \gamma G ; G\gamma ; GGh ; G\nu ; \nu = F\nu ; \gamma ; Gh ; \nu \\
\equiv & \quad \{ \text{for readability:} \} \\
& \quad \mathbf{define } \gamma_{2,1} = F\gamma ; \gamma F: FFG \rightarrow GFF \text{ and} \\
& \quad \mathbf{define } \gamma_{1,2} = \gamma G ; G\gamma: FGG \rightarrow GGF \\
& \mu ; Ff ; \gamma ; G\mu = \mu ; F\mu ; FFf ; \gamma_{2,1} \quad \wedge \\
& GFg ; \check{\gamma}G = \check{\gamma}F ; FGg \quad \wedge \\
& \gamma_{1,2} ; GGh ; G\nu ; \nu = F\nu ; \gamma ; Gh ; \nu \\
\equiv & \quad 2\text{nd conjunct: naturality } \check{\gamma}: GF \rightarrow FG ; \\
& \quad 1\text{st, 3rd conjunct: } \{ \text{aiming at the next two steps,} \} \\
& \quad \mathbf{assume } \gamma ; G\mu = \mu G ; \gamma_{2,1} \text{ and } F\nu ; \gamma = \gamma_{1,2} ; \nu F \\
& \mu ; Ff ; \mu G ; \gamma_{2,1} = \mu ; F\mu ; FFf ; \gamma_{2,1} \quad \wedge \\
& \gamma_{1,2} ; GGh ; G\nu ; \nu = \gamma_{1,2} ; \nu F ; Gh ; \nu \\
\equiv & \quad \text{naturality } \mu: F \rightarrow FF \text{ and } \nu: GG \rightarrow G \\
& \mu ; \mu F ; FFf ; \gamma_{2,1} = \mu ; F\mu ; FFf ; \gamma_{2,1} \quad \wedge \\
& \gamma_{1,2} ; GGh ; G\nu ; \nu = \gamma_{1,2} ; GGh ; \nu G ; \nu \\
\Leftarrow & \quad \text{Leibniz} \\
& \mu ; \mu F = \mu ; F\mu \quad \wedge \\
& G\nu ; \nu = \nu G ; \nu \\
\equiv & \quad \mathbf{assume } \mu ; \mu F = \mu ; F\mu \text{ and } G\nu ; \nu = \nu G ; \nu \\
& \text{true} \quad .
\end{aligned}$$

The  $\gamma_{2,1}$  and  $\gamma_{1,2}$  defined above are instances of a more general  $\gamma_{m,n}: F^m G^n \rightarrow G^n F^m$  that one may define easily by induction on  $m$  and  $n$  in several distinct but semantically equal ways. In step (\*) there is no other nontrivial way to split the giant terms into several parts than the way indicated: in general two intermediate objects can be isomorphic only if their denotations contain the same ingredients. More precisely, in step (\*) we have assumed that  $FGFb$  and  $GFFb$  are isomorphic via  $\gamma F$ ; an alternative is to assume that  $FFGb$  and  $GFFb$  are isomorphic, via  $\gamma_{2,1}$ . But since  $\gamma_{2,1} = F\gamma; \gamma F$  and since the term  $F\gamma$  is present at just the right place, this alternative assumption is equivalent to ours.

Here are the five assumptions made along the way (the equalities are assumed, the typing is provable):

$$\begin{aligned} & \gamma \text{ has inverse } \check{\gamma} \\ & \gamma; G\mu = \mu G; \gamma_{2,1} \quad : \quad FG \rightarrow GFF \\ & F\nu; \gamma = \gamma_{1,2}; \nu F \quad : \quad FGG \rightarrow GF \end{aligned}$$

and

$$\begin{aligned} & \mu; \mu F = \mu; F\mu \quad : \quad F \rightarrow FF \\ & G\nu; \nu = \nu G; \nu \quad : \quad GG \rightarrow G \end{aligned}$$

where

$$\begin{aligned} \gamma_{2,1} &= F\gamma; \gamma F \quad : \quad FFG \rightarrow GFF \\ \gamma_{1,2} &= \gamma G; G\gamma \quad : \quad FGG \rightarrow GGF \quad . \end{aligned}$$

Taking  $F = Id$  and also  $\mu, \gamma = id, id$ , these assumptions specialise to those that make the Kleisli composition for  $G, \nu$  associative.

The derivation of *nice* assumptions on  $\eta$  in order that  $id_{\mathcal{D}}$  is neutral for  $_{\mathcal{D}}$  is problematic. Let us first consider only the first equality in “ $f;_{\mathcal{D}} id = id;_{\mathcal{D}} f = f$ ”. So, let  $f: a \rightarrow_{\mathcal{D}} b$  be arbitrary; then:

$$\begin{aligned} & f;_{\mathcal{D}} id_{\mathcal{D}} = id_{\mathcal{D}};_{\mathcal{D}} f \\ \equiv & \quad \text{definition } \mathcal{D} \\ & Fa \xrightarrow{\mu} FFa \xrightarrow{Ff} FGb \xrightarrow{\gamma} GFB \xrightarrow{G\eta} GGb \xrightarrow{\nu} Gb = \\ & Fa \xrightarrow{\mu} FFa \xrightarrow{F\eta} FGa \xrightarrow{\gamma} GFa \xrightarrow{Gf} GGb \xrightarrow{\nu} Gb \\ \Leftarrow & \quad \text{Leibniz } \{ \text{no nontrivial intermediate isomorphism seems plausible} \} \\ (*) & \quad Ff; \gamma; G\eta = F\eta; \gamma; Gf \\ \Leftarrow & \quad \text{naturality } \eta: F \rightarrow G, \text{ so } Ff; \eta G = \eta F; Gf \\ (*) & \quad \gamma; G\eta = \eta G \quad \wedge \quad F\eta; \gamma = \eta F \quad . \end{aligned}$$

Equation (\*), for all  $f$ , is acceptably nice: it asserts a sort of naturality  $F \rightarrow G$ . Since we have already assumed natural transformation  $\eta: F \rightarrow G$ , it seems reasonable to require that  $\gamma; G\eta$  is  $\eta G$ , as in line (\*). However, when instantiating with  $F, \gamma := Id, id$ , both line (\*) and line (\*) give requirements that are stronger than those for a monad: line (\*) becomes  $f; G\eta = \eta; Gf$  (which is *not* just naturality  $\eta: Id \rightarrow G$ ), and line (\*) becomes

$G\eta = \eta G$ . So, we look for another sufficient condition that implies  $f ; id_{\mathcal{D}} = id_{\mathcal{D}} ; f$  for all  $f: a \rightarrow_{\mathcal{D}} b$ . The following line of reasoning has been suggested by Lambert Meertens.

$$\begin{aligned}
& f ;_{\mathcal{D}} id_{\mathcal{D}} = id_{\mathcal{D}} ;_{\mathcal{D}} f \\
\equiv & \text{definition } \mathcal{D} \\
& Fa \xrightarrow{\mu} FFa \xrightarrow{Ff} FGb \xrightarrow{\gamma} GFb \xrightarrow{G\eta} GGb \xrightarrow{\nu} Gb = \\
& Fa \xrightarrow{\mu} FFa \xrightarrow{F\eta} FGa \xrightarrow{\gamma} GFa \xrightarrow{Gf} GGb \xrightarrow{\nu} Gb \\
\equiv & \{ \text{in order to shift } Ff \text{ and } Gf \text{ towards each other, } \} \\
& \text{assume } \gamma ; G\eta ; \nu = \eta G ; \nu \text{ and } \mu ; F\eta ; \gamma = \mu ; \eta F \\
& \mu ; Ff ; \eta G ; \nu = \mu ; \eta F ; Gf ; \nu \\
\Leftarrow & \text{Leibniz} \\
& Ff ; \eta G = \eta F ; Gf \\
\equiv & \text{naturality } \eta: F \rightarrow G \\
& \text{true} .
\end{aligned}$$

The two assumptions are acceptably nice.

As regards to the second equality in “ $f ;_{\mathcal{D}} id = id ;_{\mathcal{D}} f = f$ ” we argue as follows. Let  $f: a \rightarrow_{\mathcal{D}} b$  be arbitrary, then:

$$\begin{aligned}
& f ;_{\mathcal{D}} id_{\mathcal{D}} = id_{\mathcal{D}} \\
\equiv & \text{definition } \mathcal{D} \\
& \mu ; Ff ; \gamma ; G\eta ; \nu = f \\
\equiv & \text{assumption in previous calculation} \\
(*) & \mu ; Ff ; \eta G ; \nu = f \\
\equiv & \text{naturality } \eta: F \rightarrow G \\
(*) & \mu ; \eta F ; Gf ; \nu = f .
\end{aligned}$$

Thus we are led to assume the equation of line (\*) (for all  $f: a \rightarrow_{\mathcal{D}} b$ ), or equivalently, of line (\*). However, both equations are too complicated to be called nice; in particular, the “ $f$ ” occurs in the *middle* of the term and not at one end, as in naturality assertions. Fortunately, the instantiation with  $F, \gamma, \mu := Id, id, id$  does give a monad law:

$$\begin{aligned}
& \mu ; Ff ; \eta G ; \nu = f \quad \text{for all } f: a \rightarrow_{\mathcal{D}} b \\
\equiv & \text{substitution } F, \gamma, \mu := Id, id, id \\
& f ; \eta G ; \nu = f \quad \text{for all } f: a \rightarrow Gb \\
\equiv & \text{for } \Leftarrow: \text{Leibniz;} \\
& \text{for } \Rightarrow: \text{take } a, f := Gb, id_{Gb} \\
& \eta G ; \nu = id_G .
\end{aligned}$$

**Summary.** Let  $F, G$  be functors. Then we call  $(F, G, \gamma, \eta, \mu, \nu)$  a **dyad** if:

$$\begin{aligned} \gamma & : FG \rightarrow GF \\ \eta & : F \rightarrow G \\ \mu & : F \rightarrow FF \\ \nu & : GG \rightarrow G \quad , \end{aligned}$$

satisfy the following conditions:

$$\begin{aligned} \mu ; \mu F & = \mu ; F\mu & : F \rightarrow FF \\ G\nu ; \nu & = \nu G ; \nu & : GG \rightarrow G \\ \gamma ; G\eta ; \nu & = \eta G ; \nu \\ \mu ; F\eta ; \gamma & = \mu ; \eta F \\ \mu ; Ff ; \eta G ; \nu & = f & \text{ for all } f: a \rightarrow_{\mathcal{D}} b \\ \gamma & \text{ has inverse } \check{\gamma} \\ \gamma ; G\mu & = \mu G ; \gamma_{2,1} & : FG \rightarrow GFF \\ F\nu ; \gamma & = \gamma_{1,2} ; \nu F & : FGG \rightarrow GF \end{aligned}$$

where

$$\begin{aligned} \gamma_{2,1} & = F\gamma ; \gamma F & : FFG \rightarrow GFF \\ \gamma_{1,2} & = \gamma G ; G\gamma & : FGG \rightarrow GGF \quad . \end{aligned}$$

After substituting  $F, \gamma, \mu := Id, id, id$ , these requirements are equivalent to the statement that  $(G, \eta, \nu)$  is a monad.

## References

- [1] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.
- [2] P. Wadler. Comprehending monads. In *ACM Conference on Lisp and Functional Programming*, June 1990. To appear in *Mathematical Structures in Computer Science*.