Compound Monads and the Kleisli Category

Jeremy E. Dawson *

Logic and Computation Program, NICTA ** and
Computer Sciences Laboratory,
Australian National University, Canberra, ACT 0200, Australia
`Jeremy.Dawson@nicta.com.au` `http://csl.anu.edu.au/~jeremy/`

Abstract. We consider sets of monad rules derived by focussing on the Kleisli category of a monad, and from these we derive some constructions for compound monads. Under certain conditions these constructions correspond to a distributive law connecting the monads. We also show how these relate to some constructions for compound monads described previously.

Keywords: compound monad, Kleisli category

1 Introduction

Monads are convenient for structuring functional programs. For example, an algorithm requiring use of a mutable state may be coded in a purely functional style using the *state* monad, or an algorithm involving some non-deterministic steps may be conveniently coded using the *list* monad. Moggi [7] shows how to describe computational activity in terms of monads. Wadler [8] shows in particular how it is easy to change a program to introduce these features if it is written in a monadic style. However, to code (equally conveniently) a program involving both these aspects requires a monad which somehow combines both the state and the list monads.

A monad consists of a type constructor and several functions (of appropriate types), which must satisfy certain rules. There are several equivalent formulations of these rules. Once we have defined the type constructor for a compound monad, it is necessary to prove the rules for the compound monad, as they do not follow automatically from the rules for the individual monads. It is important to establish that the rules hold, since to use a type constructor wrongly thought to be a monad can lead to incorrect programs – two ways of coding a function, which are thought to be equivalent, may be different. Yet the constructions can be complex, leading to errors in proofs, for example see [3, §6.4.2].

* Supported by an Australian Research Council Large Grant

** National ICT Australia is funded by the Australian Government's Dept of Communications, Information Technology and the Arts and the Australian Research Council through *Backing Australia's Ability* and the ICT Research Centre of Excellence programs.

A monad is a category theoretic notion, and we focus on sets of rules relating to the Kleisli category of a monad, and show how these facilitate proofs of the monad rules for some compound monads.

In the rest of this section we introduce monads and the category theory that we use in the rest of the paper. In section 2 we give several sets of rules, and prove some useful results about monads. and in Appendix B we give examples of the easy use of some of these rules to certain compound monads. In section 3 we describe two constructions which are useful in defining a compound monad, and we set out the rules which must be satisfied in each case. We show how one of these constructions amounts to defining a monad in the Kliesli category of another monad. In Appendix A we show how the constructions of Jones & Duponcheel [3] are cases of ours. Section 4 concludes.

The results of this paper have been proved using the theorem prover Isabelle/HOL. The proofs are at `http://users.rsise.anu.edu.au/~jeremy/isabelle/monad/`. While most of the proofs are not difficult, much of the work involved showing the equivalence of different sets of axioms. In this regard the "book-keeping" aspect of using a theorem prover, to keep track of which results are being assumed, and which proved, at a particular point, was of great value.

1.1 Monads

A *monad* is a type constructor M (which we write postfix) together with functions of the types given below, which must satisfy certain rules. If we loosely say "M is a monad", this refers also to associated functions which will be either clear from the context, or have the names used below. Where several monads are involved, we may use subscripts to avoid ambiguity in the names of these functions. For the names of monad rules we write (for example) (E2N) to mean rule (E2) for the monad N, that is, $ext_N\ unit_N\ = id$.

$$\begin{aligned}
unit &: \alpha \to \alpha M \\
map &: (\alpha \to \beta) \to (\alpha M \to \beta M) \\
join &: \alpha M M \to \alpha M \\
ext &: (\alpha \to \beta M) \to (\alpha M \to \beta M) \\
bind &: \alpha M \to (\alpha \to \beta M) \to \beta M && \text{(infix)} \\
\odot &: (\beta \to \gamma M) \to (\alpha \to \beta M) \to (\alpha \to \gamma M) && \text{(infix)}
\end{aligned}$$

This set of functions is not minimal, they are related by the rules shown below. Note that id and $\circ$ denote the identity function and function composition respectively, and that $bind$ and $\odot$ are written in infix notation. (Infix operators have a lower precedence than function application).

$$\begin{aligned}
m\ bind\ f &= ext\ f\ m \\
ext\ f &= join \circ map\ f \\
join &= ext\ id \\
map\ f &= ext\ (unit \circ f) \\
g \odot f &= ext\ g \circ f \\
ext\ g &= g \odot id
\end{aligned}$$

For M and the functions to qualify as a monad, a set of rules must be satisfied. Several sets of rules have been given. Firstly, when the functions *unit*, *map* and *join* are given, rules (1) to (7) must be satisfied. This is the "monoid form" presentation of an algebraic theory of Manes [6, Chapter 1, Definition 3.17]. A *premonad* satisfies rules (1) to (3).

$$
\begin{aligned}
\mathit{map}\ \mathit{id} &= \mathit{id} && (1)\\
\mathit{map}\ f \circ \mathit{map}\ g &= \mathit{map}\ (f \circ g) && (2)\\
\mathit{unit} \circ f &= \mathit{map}\ f \circ \mathit{unit} && (3)\\
\mathit{join} \circ \mathit{map}\ (\mathit{map}\ f) &= \mathit{map}\ f \circ \mathit{join} && (4)\\
\mathit{join} \circ \mathit{unit} &= \mathit{id} && (5)\\
\mathit{join} \circ \mathit{map}\ \mathit{unit} &= \mathit{id} && (6)\\
\mathit{join} \circ \mathit{map}\ \mathit{join} &= \mathit{join} \circ \mathit{join} && (7)
\end{aligned}
$$

$$\mathit{ext}\ f = \mathit{join} \circ \mathit{map}\ f \tag{8}$$

$$\mathit{ext}\ (g \circ f) = \mathit{ext}\ g \circ \mathit{map}\ f \tag{9}$$

Then (8) defines *ext* in terms of *join* and *map*. At this point it is convenient to note rule (9), which clearly follows from (8) and (2).

Alternatively, a monad can be defined in terms of *unit* and *ext*. Wadler [8] shows the equivalence to analogues (in terms of *bind*) of rules (E1), (E2) and (E3′), which are given by Moggi [7]. This is also the "extension form" presentation of an algebraic theory of Manes [6, Chapter 1, §3, Exercise 12].

$$
\begin{aligned}
\mathit{ext}\ f \circ \mathit{unit} &= f && (\mathrm{E1})\\
\mathit{ext}\ \mathit{unit} &= \mathit{id} && (\mathrm{E2})\\
\mathit{ext}\ (\mathit{ext}\ g \circ f) &= \mathit{ext}\ g \circ \mathit{ext}\ f && (\mathrm{E3'})
\end{aligned}
$$

$$
\begin{aligned}
\mathit{join} &= \mathit{ext}\ \mathit{id} && (\mathrm{E4})\\
\mathit{map}\ f &= \mathit{ext}\ (\mathit{unit} \circ f) && (\mathrm{E5})
\end{aligned}
$$

If a monad is defined by giving *unit*, *map* and *join*, satisfying the rules (1) to (7), and if *ext* is defined by (8), then rules (E1), (E2), (E3′), (E4) and (E5) hold. Conversely, if a monad is defined by giving *unit* and *ext*, satisfying rules (E1), (E2) and (E3′), and if *join* and *map* are defined by (E4) and (E5), then the rules (1) to (8) hold. (Wadler states the analogous result for the formulation of rules (E1), (E2) and (E3′) in terms of *bind*.)

1.2 Some Category Theory

Monads have been known (sometimes under different names) as part of category theory before their use in describing computations, eg [5, 6]. A *category* consists of a collection of *objects* and a collection of *arrows*. Each arrow f has a *source* object $s(f)$ and a *target* object $t(f)$. Two arrows f and g, such that $t(f) = s(g)$, can be composed to give another arrow which we write $g \circ f$, with source $s(f)$ and target $t(g)$. Composition of arrows is associative. For each object α there is an *identity* arrow id_α, such that $id_\alpha \circ f = f = f \circ id_\alpha$. See [5] for more details.

Clearly then, if we let the objects be types (eg, α, β), and the arrows functions, such as $f : \alpha \to \beta$, we have a category $\mathcal{T}$, where $s(f) = \alpha$ and $t(f) = \beta$.

A *functor* from one category to another (or to the same category) is a mapping θ which takes objects to objects and arrows to arrows, and preserves

– the sources and targets of arrows, ie $s(\theta f) = \theta(s(f))$ and $t(\theta f) = \theta(t(f))$
– composition of arrows, ie $\theta(g \circ f) = \theta g \circ \theta f$
– identity arrows, ie $\theta(id_\alpha) = id_{\theta(\alpha)}$

If M and *map* satisfy the type information in §1.1 and the rules (1) and (2), then M and *map* give a functor θ from the category of types and functions to itself, where $\theta(\alpha) = \alpha M$ and $\theta(f) = map\ f$. (By an abuse of terminology we will often refer to the action of a function on arrows, such as *map*, or *ext* or *kl* below, as a "functor").

However there is another interesting category associated with a monad, namely the *Kleisli* category $\mathcal{K}_M$ (see [5, ChVI, §5]). We describe it in functional programming terms. Its objects are the types (eg, α, β), but an arrow with source α and target β is a function of type $\alpha \to \beta M$. The identity for object α is the function $unit : \alpha \to \alpha M$, and composition is the function $\odot$. There is a functor from the "ordinary" category $\mathcal{T}$ of types and functions to the Kleisli category $\mathcal{K}_M$. We call it *kl* ("Kleisli lift") and define its action (on arrows) by

$$\begin{aligned} kl &: (\alpha \to \beta) \to (\alpha \to \beta M) \\ kl\ f &= unit \circ f \end{aligned}$$

The function *ext* is a functor from $\mathcal{K}_M$ to $\mathcal{T}$ (on objects, the functor takes a type α to the type $\alpha\ M$), and we may note that the rule (E5) tells us that the composition of the functors *ext* and *kl* is the functor *map*. See [5, ChVI, §5, Theorem 1] for these results.

2 Axioms for a Single Monad

2.1 A Weaker Set of Monad Rules

First we give a set of monad rules (E1) to (E3) which is similar to those in §1.1, but replacing (E3′) by (E3). Note also that (E2) and (E3) express the fact that *ext* is a functor. Rule (E3) is apparently easier to satisfy than (E3′) since we may choose $\odot$ arbitrarily: if we add to (E3) the extra requirement of (E6) then we get (E3′). But in fact (E6) can be *proved* from rules (E1) and (E3).

$$ext\ f \circ unit = f \qquad \text{(E1)}$$
$$ext\ unit = id \qquad \text{(E2)}$$
$$ext\ (g \odot f) = ext\ g \circ ext\ f \qquad \text{(E3)}$$

$$g \odot f = ext\ g \circ f \qquad \text{(E6)}$$

In fact we can prove (E6), identity and associativity in the Kleisli category, and some other useful results, from rules (E1) to (E3).

Theorem 1. *Assuming rules (E1) to (E3)*

$$g \odot f = \mathit{ext}\ g \circ f \tag{E6}$$
$$(h \odot g) \circ f = h \odot (g \circ f) \tag{A6}$$
$$\mathit{ext}\ f = \mathit{ext}\ g \Rightarrow f = g \tag{EI}$$
$$f \odot \mathit{unit} = f \tag{A1}$$
$$\mathit{unit} \odot f = f \tag{A2}$$
$$h \odot (g \odot f) = (h \odot g) \odot f \tag{A3}$$

Proof. (E6): By (E3), $\mathit{ext}\ (g \odot f) \circ \mathit{unit} = \mathit{ext}\ g \circ \mathit{ext}\ f \circ \mathit{unit}$. Using (E1), this simplifies to $g \odot f = \mathit{ext}\ g \circ f$.

(EI): If $\mathit{ext}\ f = \mathit{ext}\ g$ then $\mathit{ext}\ f \circ \mathit{unit} = \mathit{ext}\ g \circ \mathit{unit}$, that is, by (E1), $f = g$.

(A6): Using (E6), both sides are equal to $\mathit{ext}\ h \circ g \circ f$.

(A1,A2,A3): One proof of these would proceed by rewriting these, using (E6) and (E3), to eliminate all uses of $\odot$. Then they follow from (E1), (E2) and the associativity of $\circ$ (since both sides of (A3) become $\mathit{ext}\ h \circ \mathit{ext}\ g \circ f$). But another proof which better reflects our focus on categories and functors is to observe that in each case, by (EI), it suffices to prove $\mathit{ext}(\mathit{lhs}) = \mathit{ext}(\mathit{rhs})$. In each case, this may be proved using the fact that *ext* is a functor, ie, using (E3) and (E2). That is, we use the identity and associativity rules for *id* and $\circ$ to prove the corresponding rules for *unit* and $\odot$.

for (A1): $\mathit{ext}\ (f \odot \mathit{unit}) = \mathit{ext}\ f \circ \mathit{ext}\ \mathit{unit} = \mathit{ext}\ f \circ \mathit{id} = \mathit{ext}\ f$

for (A2): $\mathit{ext}\ (\mathit{unit} \odot f) = \mathit{ext}\ \mathit{unit} \circ \mathit{ext}\ f = \mathit{id} \circ \mathit{ext}\ f = \mathit{ext}\ f$

for (A3): $\mathit{ext}\ (h \odot (g \odot f)) = \mathit{ext}\ h \circ (\mathit{ext}\ g \circ \mathit{ext}\ f) =$
$(\mathit{ext}\ h \circ \mathit{ext}\ g) \circ \mathit{ext}\ f = \mathit{ext}\ ((h \odot g) \odot f)$ □

The following theorem is useful later.

Theorem 2. *In a monad the following are equivalent*

(i) $\mathit{ext}\ g = g \circ \mathit{join}$
(ii) $g = \mathit{ext}\ (g \circ \mathit{unit})$
(iii) there exists f *such that* $g = \mathit{ext}\ f$
(iv) for all h, $\mathit{ext}\ (g \circ h) = g \circ \mathit{ext}\ h$

Proof. (i) ⇒ (ii): Assume (i). Then $\mathit{ext}\ g \circ \mathit{map}\ \mathit{unit} = g \circ \mathit{join} \circ \mathit{map}\ \mathit{unit}$ and so, by (9) and (6), $\mathit{ext}\ (g \circ \mathit{unit}) = g$.

(ii) ⇒ (iii): is obvious.

(iii) ⇒ (iv): When $g = \mathit{ext}\ f$, (iv) is just (E3′).

(iv) ⇒ (i): Set $h = \mathit{id}$ and use (E4).

(iv) ⇒ (ii): Set $h = \mathit{unit}$ and use (E2). □

Note also that in a monad, these properties of g are preserved under composition, that is, if they hold for g_1 and g_2 then they hold for $g_1 \circ g_2$. This can be seen for (iii) from (E3), or for (i) from (8) and (9).

Finally, we include a proof that *kl* is a functor [5, ChVI, §5].

Theorem 3. *kl is a functor.*

Proof. Clearly *kl* preserves identity: $kl\ id = unit \circ id = unit$. For composition:

$$\begin{aligned} kl\ g \odot kl\ f &= (unit \circ g) \odot (unit \circ f) \\ &= ((unit \circ g) \odot unit) \circ f && \text{(A6)} \\ &= (unit \circ g) \circ f = kl\ (g \circ f) && \text{(A1)}\ \square \end{aligned}$$

From rules (E1) to (E3), and so (E6), we have (E3′), and so, assuming (E4) and (E5) as definitions, we can therefore assume rules (1) to (8) (Wadler[8, §2.10]). However it is now easy to outline proofs of rules (1) to (8). Using (E4) and (E5), (8) is proved using (E3′) and (E1). Using (8), (6) is just (E2). Using (8), then (E5) and (E4) respectively, (4) and (7) are special cases of Theorem 2, (iii) ⇒ (i). Using (E5) and (E4) respectively, (3) and (5) are cases of (E1). Finally, as noted earlier, (E5) shows that $map = ext \circ kl$. As *ext* and *kl* are functors, so is *map*, which gives us (1) and (2).

2.2 Monad Rules Based on the Kleisli Category

We have proved the three rules, (A1) to (A3), which reflect that the Kleisli category is indeed a category. We may ask whether these are sufficient to establish a monad. If we could assume also (E6), then it would be easy to show rules (E1) to (E3). However, if we are (hoping to) define a monad in terms of these rules, and in terms of the functions *unit* and $\odot$, then we can take (A5) as a definition, but that does not give us (E6).

If we add a fourth rule, (A4) (see below), governing $\odot$, then that is sufficient. For, clearly, (A4) and (A5) imply (E6). Note that (A4) is the special case $g = id$ of (A6) (shown again below) and is therefore true in any monad, see Theorem 1. Therefore we have the following theorem. Note that, if we replaced (A4) by (A4′), we would get, in effect, the "clone form" presentation of an algebraic theory of Manes [6, Chapter 1, Definition 3.2]. In fact it is easy to show (A6) directly from either (A4) or (A4′) using the associativity of $\circ$ or $\odot$ respectively.

$$f \odot unit = f \tag{A1}$$
$$unit \odot f = f \tag{A2}$$
$$h \odot (g \odot f) = (h \odot g) \odot f \tag{A3}$$
$$(h \odot id) \circ f = h \odot f \tag{A4}$$

$$h \odot (unit \circ f) = h \circ f \tag{A4′}$$

$$ext\ g = g \odot id \tag{A5}$$

$$(h \odot g) \circ f = h \odot (g \circ f) \tag{A6}$$

Theorem 4. *Rules (A1) to (A4) and definitions (A5), (E4), (E5) give a monad.*

In Appendix B are some examples showing how easy it often is to prove the monad rules when using rules (A1) to (A4). These include two examples of compound monads, formed by starting with a monad M and developing a more complex monad from it.

2.3 A Free Theorem

In the examples in Appendix B it is trivial to prove rules (A1) to (A3); in these examples the rule (A4) is the only one of the four whose proof is not quite immediate (that is, although easy, it is tedious, involving several steps). We may therefore ask whether it is really necessary. It turns out that in typical cases it is a "free theorem" (Wadler [9]). We consider it in the "naive set-theoretic" fashion of [9, §2,§3]. The latter sections of that paper indicate the semantic complexities involved in applying such results in a setting such as a typical functional programming language, where the types cannot be regarded as sets. Therefore the application of these results to any particular functional programming language depends on the semantics of that language and its type system. First, we state a more general result. In category theoretic terms, this result it that θ is a natural transformation between the functors $(\_ \to \beta)$ and $(\_ \to \gamma)$.

Proposition 5. *If $\theta : \forall \alpha.\ (\alpha \to \beta) \to (\alpha \to \gamma)$ is a function which is polymorphic in α, but where β and γ are fixed types, then*

(i) there exists $h : \beta \to \gamma$ such that (for all $f : \alpha \to \beta$) $\theta f = h \circ f$
(ii) for all $f : \alpha \to \alpha'$ and $g : \alpha' \to \beta$, $\theta\ (g \circ f) = \theta\ g \circ f$

The result (i) may be explained intuitively as follows. Since θ is polymorphic in α, the coding of the function θ cannot use any information about the type α. That is, although an argument $a : \alpha$ to the function θf may be an integer, tree of strings, or whatever, θ (on its own) can only treat a as a "black box". So the only thing that θf can do with a (other than ignore it) is to apply f to it.

Wadler [9, §3] shows how to obtain results of this nature about particular polymorphic functions, and we follow his approach. In particular, we write the proof as though types were sets.

Proof. Clearly the two parts are equivalent (h being $\theta\ id$). We prove (ii). Wadler's general result is $\theta \sim \theta$, where $\sim$ denotes the relation appropriate to the type of the terms related, which, for θ, is $\forall \alpha.\ (\alpha \to \beta) \to (\alpha \to \gamma)$. This means that for all α, α' and relation $\sim \subseteq \alpha \times \alpha'$, for all $k : \alpha \to \beta$, $k' : \alpha' \to \beta$, $a : \alpha$ and $a' : \alpha'$, whenever $k \sim k'$ and $a \sim a'$, then $\theta\ k\ a \sim \theta\ k'\ a'$. Now $k \sim k'$ means whenever $x \sim x'$, for $x : \alpha$ and $x' : \alpha'$, then $k\ x \sim k'\ x'$.

We choose the relation $\sim \subseteq \alpha \times \alpha'$ to be $a \sim a' \iff a' = f\ a$; the relations on (fixed) types β and γ are the identity. So we can simplify the above. We get $k \sim k'$ iff, for all $x : \alpha$, $k\ x = k'\ (f\ x)$, that is, $k = k' \circ f$. Then, letting $k' = g$ and $k = g \circ f$, we have $k \sim k'$. So for all $a : \alpha$, we can let $a' = f\ a$, so $a \sim a'$ and $\theta\ k\ a \sim \theta\ k'\ a'$, that is, $\theta\ (g \circ f)\ a = \theta\ g\ (f\ a)$. Therefore, $\theta\ (g \circ f) = \theta\ g \circ f$. □

Proposition 6. *Assuming that $\odot : (\beta \to \gamma M) \to (\alpha \to \beta M) \to (\alpha \to \gamma M)$ is polymorphic in α, the rule (A6) holds.*

Proof. Given $h : \beta \to \gamma M$, define $\theta\ g \equiv h \odot g$. Then Theorem 5(ii) gives $h \odot (g \circ f) = (h \odot g) \circ f$, as required. □

3 Compound Monad Constructions

If type constructors M and N, with associated functions, are monads, then although it is not necessarily the case that we can combine these two monads to form a third, this is sometimes possible.

Compound monads can arise naturally and be practically useful ([8], [3]). In this section we consider two constructions for compound monads and discuss the rules that need to be satisfied in each case. We note that when the conditions are satisfied for both constructions to be applicable, then we have the distributive law for monads described by Manes [6] and Barr & Wells [1]: see §3.8. In this case the monads are "compatible" [1, §9.2], and our results show how some of the conditions given in [1] for this are redundant.

3.1 Compound Monads *via* Partial Extension

Let M be a monad, and consider compound monads where the compound monad type is $(\alpha N)M$, which we will just write as αNM. To define a compound monad NM, we will need a function ext_{NM}, so-called, presumably, because it "extends" a function f from a "smaller" domain, α, to a "larger" one, αNM. Consider, therefore, a "partial extension" function *pext* which does part of this job:

$$\begin{aligned} \mathit{ext}_{NM} &: (\alpha \to \beta NM) \to (\alpha NM \to \beta NM) \\ \mathit{pext} &: (\alpha \to \beta NM) \to (\alpha N \to \beta NM) \end{aligned}$$

Rules (E1K) to (E3K) are sufficient to define a compound monad using such a function *pext*. We assume nothing about the functions $\odot_{NM}$ or unit_{NM}, except that they have the appropriate types. We need not assume that N is a monad, although in many examples rules (E1K) to (E3K) are proved to hold for any monad M, when setting M to the identity monad gives a monad N.

$$\mathit{pext}\ f \odot_M \mathit{unit}_{NM} = f \tag{E1K}$$
$$\mathit{pext}\ \mathit{unit}_{NM} = \mathit{unit}_M \tag{E2K}$$
$$\mathit{pext}\ (g \odot_{NM} f) = \mathit{pext}\ g \odot_M \mathit{pext}\ f \tag{E3K}$$

$$\mathit{kjoin} = \mathit{pext}\ \mathit{unit}_M \tag{E4K}$$
$$\mathit{kmap}\ f = \mathit{pext}\ (\mathit{unit}_{NM} \odot_M f) \tag{E5K}$$

$$g \odot_{NM} f = \mathit{pext}\ g \odot_M f \tag{E6K}$$

By comparing (E1K) to (E3K) with (E1) to (E3) we see that we have the three rules needed for a monad N in $\mathcal{K}_M$, the Kleisli category of M. We will refer to this monad as $N_{\mathcal{K}M}$. Thus the treatment of a single monad described in §1, §2.1 and §2.2 applies to this monad. We define the counterparts of *map* and *join*, calling them *kmap* and *kjoin*: note how rules (E4K) and (E5K) correspond to (E4) and (E5). As in §2.1, we deduce (E6K), giving $\odot_{NM}$ in tems of *pext*. The various functions and theorems involved have counterparts of which examples are tabulated below, where the left-hand side shows the standard treatment (set out as for a monad N), and the right-hand side shows the monad $N_{\mathcal{K}M}$ in $\mathcal{K}_M$.

$$\begin{array}{rl@{\qquad}rl}
id : & \alpha \to \alpha & unit_M : & \alpha \to \alpha M \\
unit_N : & \alpha \to \alpha N & unit_{NM} : & \alpha \to \alpha NM \\
map_N : & (\alpha \to \beta) \to \alpha N \to \beta N & kmap : & (\alpha \to \beta M) \to \alpha N \to \beta NM \\
join_N : & \alpha NN \to \alpha N & kjoin : & \alpha NN \to \alpha NM \\
ext_N : & (\alpha \to \beta N) \to \alpha N \to \beta N & pext : & (\alpha \to \beta NM) \to \alpha N \to \beta NM \\
ext_N\ g = & g \odot_N\ id & pext\ g = & g \odot_{NM}\ unit_M \\
g \odot_N\ f = & ext_N\ g \circ f & g \odot_{NM}\ f = & pext\ g \odot_M\ f \\
join_N = & ext_N\ id & kjoin = & pext\ unit_M \\
map_N\ f = & ext_N\ (unit_N \circ f) & kmap\ f = & pext\ (unit_{NM} \odot_M\ f) \\
ext_N\ f = & join_N \circ map_N\ f & pext\ f = & kjoin \odot_M\ kmap\ f \\
h \odot_N\ f = & (h \odot_N\ id) \circ f & h \odot_{NM}\ f = & (h \odot_{NM}\ unit_M) \odot_M\ f
\end{array}$$

We will refer to the rules and results about this monad by putting “K” after the names used in §1 and §2. We also obtain rules (1K) to (8K) which are the counterparts of (1) to (8). Then, just as in §1, these seven rules, with *pext* and $\odot_{NM}$ defined from *kjoin* and *kmap* by (8K) and (E6K), are sufficient to show rules (E1K) to (E3K) and the converse definitions of *kjoin* and *kmap* in terms of *pext*, (E4K) and (E5K).

This monad $N_{\mathcal{K}M}$ in $\mathcal{K}_M$, the Kleisli category for M, gives rise to a further Kleisli category, which we may describe as the Kleisli category for $N_{\mathcal{K}M}$ in $\mathcal{K}_M$. Its identity is $unit_{NM}$ and its composition function is $\odot_{NM}$. Note also that the two rules (E2K) and (E3K) express the fact that *pext* is (the action on arrows of) a functor, from this compound Kleisli category to $\mathcal{K}_M$.

$$kmap\ unit_M = unit_M \tag{1K}$$
$$kmap\ f \odot_M\ kmap\ g = kmap\ (f \odot_M\ g) \tag{2K}$$
$$unit_{NM} \odot_M\ f = kmap\ f \odot_M\ unit_{NM} \tag{3K}$$
$$kjoin \odot_M\ kmap\ (kmap\ f) = kmap\ f \odot_M\ kjoin \tag{4K}$$
$$kjoin \odot_M\ unit_{NM} = unit_M \tag{5K}$$
$$kjoin \odot_M\ kmap\ unit_{NM} = unit_M \tag{6K}$$
$$kjoin \odot_M\ kmap\ kjoin = kjoin \odot_M\ kjoin \tag{7K}$$

$$pextf = kjoin \odot_M\ kmap\ f \tag{8K}$$

This monad can also be characterised by four rules analogous to (A1) to (A4): that is, the rules (A1K) to (A4K). We also show (A5K) and (A6K).

$$f \odot_{NM}\ unit_{NM} = f \tag{A1K}$$
$$unit_{NM} \odot_{NM}\ f = f \tag{A2K}$$
$$h \odot_{NM}\ (g \odot_{NM}\ f) = (h \odot_{NM}\ g) \odot_{NM}\ f \tag{A3K}$$
$$(h \odot_{NM}\ unit_M) \odot_M\ f = h \odot_{NM}\ f \tag{A4K}$$

$$pext\ g = g \odot_{NM}\ unit_M \tag{A5K}$$
$$(h \odot_{NM}\ g) \odot_M\ f = h \odot_{NM}\ (g \odot_M\ f) \tag{A6K}$$

Now, to show that NM is a compound monad, we also need to show four rules analogous to (A1) to (A4), namely (A1NM) to (A4NM). Of these, the first three, (A1NM) to (A3NM), are the same as (A1K) to (A3K); only (A4NM) is different. So to show that NM (defined by unit_{NM} and $\odot_{NM}$) is a monad, we need only show (A4NM), and to do this we have available both (A4K) and (A4M).

$$(h \odot_{NM} \mathit{id}) \circ f = h \odot_{NM} f \qquad \text{(A4NM)}$$
$$(h \odot_{M} \mathit{id}) \circ f = h \odot_{M} f \qquad \text{(A4M)}$$

Theorem 7. *Assume that M is a monad and that functions pext, $\odot_{NM}$ and unit_{NM} of the appropriate types are given, satisfying rules (E1K) to (E3K). Then $\odot_{NM}$ and unit_{NM} define a monad, and, using (A5NM) to define ext_{NM},*

$$\mathit{ext}_{NM}\ f = \mathit{ext}_M\ (\mathit{pext}\ f) \qquad \text{(EC)}$$
$$\mathit{pext}\ f = \mathit{ext}_{NM}\ f \circ \mathit{unit}_M \qquad \text{(PE)}$$
$$\mathit{ext}_M\ (\mathit{ext}_{NM}\ f) = \mathit{ext}_{NM}\ f \circ \mathit{join}_M \qquad \text{(J1S)}$$

Proof. Rules (E1K) to (E3K) establish that *pext*, $\odot_{NM}$ and unit_{NM} define a monad in $\mathcal{K}_M$, and so rules (A1K) to (A4K) are satisfied. Since M is a monad, (A4M) holds, and so we use (A4M) and (A4K) to show (A4NM) as follows:

$$\begin{aligned}(h \odot_{NM} \mathit{id}) \circ f &= ((h \odot_{NM} \mathit{unit}_M) \odot_M \mathit{id}) \circ f && \text{(A4K)}\\ &= (h \odot_{NM} \mathit{unit}_M) \odot_M f = h \odot_{NM} f && \text{(A4M, A4K)}\end{aligned}$$

(EC): By (A5M) and (A5NM) this is (E6K).
(PE): By (E6NM) this is (A5K).
(J1S): By Theorem 2 for M, (iii) $\Rightarrow$ (i), this follows from (EC). □

In Theorem 7 we proved rules (A1) to (A4) for NM, so proving that NM is a monad. However, we note that it is also trivial to prove rules (E1) to (E3) for NM. In fact, using (EC), each rule for NM follows from the same rule for M and the corresponding rule from among (E1K) to (E3K).

We may next ask which compound monads can be constructed from such a function *pext*, satisfying rules (E1K) to (E3K). Taking as read the requirement that the monadic type is $(\alpha N)M$, (which we will regard as implicit in the notation NM), the previous theorem provides a necessary condition, namely that (J1S) holds. In fact, this condition is also sufficient.

Theorem 8. *Let M and NM be monads, such that (J1S) (see Theorem 7) holds. Then $\odot_{NM}$ also defines a monad in the category $\mathcal{K}_M$, and, using (PE) to define pext, (EC) holds.*

Proof. We need to show that $\odot_{NM}$ satisfies the four rules (A1K) to (A4K). Now (A1K) to (A3K) are the same as (A1NM) to (A3NM), which hold as NM is a monad. So we need only (A4K), which we show follows from (J1S).

$$\begin{aligned}(h \odot_{NM} \mathit{unit}_M) \odot_M f &= \mathit{ext}_M\ (\mathit{ext}_{NM}\ h \circ \mathit{unit}_M) \circ f && \text{(E6NM, E6M)}\\ &= \mathit{ext}_M\ (\mathit{ext}_{NM}\ h) \circ \mathit{map}_M\ \mathit{unit}_M \circ f && \text{(9M)}\\ &= \mathit{ext}_{NM}\ h \circ \mathit{join}_M \circ \mathit{map}_M\ \mathit{unit}_M \circ f && \text{(J1S)}\\ &= \mathit{ext}_{NM}\ h \circ f = h \odot_{NM} f && \text{(6M, E6NM)}\end{aligned}$$

Then NM is a monad in which *pext* is defined by (A5K); but, by (E6NM), this is equivalent to (PE). Then (EC) follows from Theorem 7 (alternatively, from (J1S) by Theorem 2 for M, (i) $\Rightarrow$ (ii)). $\Box$

This shows that, given a monad M, that the compound monads NM obtainable using the construction via a monad N in $\mathcal{K}_M$ are precisely the monads NM such that (J1S) is satisfied.

3.2 Relation to the *prod* construction of Jones & Duponcheel

We have, in Theorems 7 and 8, shown that the compound monads which can be defined in terms of a function *pext* (equivalently, in terms of functions *kjoin* and *kmap*) are precisely those satisfying (J1S).

Jones & Duponcheel [3] consider only compound monads which satisfy (UC) and (MC). Note that, as shown in [3, §3], when M and N are premonads and (UC) and (MC) hold, then NM is a premonad. Assuming that M is a monad and that N is a premonad, they show how to define a compound monad given a function *prod*, satisfying four rules P(1) to P(4) (see Appendix A.1), and show that the compound monads NM which can be defined using *prod* are precisely those satisfying (J1) (which is the case $f = id$ of (J1S)). In fact (MC) and (J1) are sufficient to imply (J1S). We also give a condition (E1D), which is also implied by (MC) and its case $f = id$, and which, assuming (UC), is equivalent to (J1S). The proof of (J1S) from (E1D) is thanks to an observation by Michael Barr (see §3.8).

$$unit_{NM}\ f = unit_M\ (unit_N\ f) \tag{UC}$$
$$map_{NM}\ f = map_M\ (map_N\ f) \tag{MC}$$
$$ext_M\ join_{NM} = join_{NM} \circ join_M \tag{J1}$$
$$ext_{NM}\ f \circ map_M\ unit_N = ext_M\ f \tag{E1D}$$
$$join_{NM} \circ map_M\ unit_N = join_M \tag{E1DI}$$

Lemma 9. *If M and NM are monads, then*

(i) (MC) and (J1) imply (J1S)
(ii) (MC) and (E1DI) imply (E1D)
(iii) assuming (UC), (J1S) and (E1D) are equivalent.

Proof. (i) By Theorem 2 for M, (i) $\Rightarrow$ (iii), we can assume, from (J1), that $join_{NM} = ext_M\ g$. By Theorem 2, (iii) $\Rightarrow$ (i), for any given f, it is enough to find h such that $ext_{NM}\ f = ext_M\ h$.

$$\begin{aligned} ext_{NM}\ f &= join_{NM} \circ map_{NM}\ f && \text{(8NM)} \\ &= ext_M\ g \circ map_M\ (map_N\ f) && \text{(MC)} \\ &= ext_M\ (g \circ map_N\ f) && \text{(9M)} \end{aligned}$$

(ii) Assuming (E1DI),

$$\begin{aligned} \mathit{ext}_{NM}\ f \circ \mathit{map}_M\ \mathit{unit}_N &= \mathit{join}_{NM} \circ \mathit{map}_{NM}\ f \circ \mathit{map}_M\ \mathit{unit}_N && \text{(8NM)} \\ &= \mathit{join}_{NM} \circ \mathit{map}_M\ (\mathit{map}_N\ f \circ \mathit{unit}_N) && \text{(MC, 2M)} \\ &= \mathit{join}_{NM} \circ \mathit{map}_M\ \mathit{unit}_N \circ \mathit{map}_M\ f && \text{(3N, 2M)} \\ &= \mathit{join}_M \circ \mathit{map}_M\ f = \mathit{ext}_M\ f && \text{(E1DI, E4M)} \end{aligned}$$

(iii) Let (J1S) hold; we show (E1D). Using Theorem 2 for NM, (i) $\Rightarrow$ (ii),

$$\begin{aligned} \mathit{ext}_{NM}\ f \circ \mathit{map}_M\ \mathit{unit}_N &= \mathit{ext}_M\ (\mathit{ext}_{NM}\ f \circ \mathit{unit}_M) \circ \mathit{map}_M\ \mathit{unit}_N \\ &= \mathit{ext}_M\ (\mathit{ext}_{NM}\ f \circ \mathit{unit}_M \circ \mathit{unit}_N) && \text{(9M)} \\ &= \mathit{ext}_M\ (\mathit{ext}_{NM}\ f \circ \mathit{unit}_{NM}) = \mathit{ext}_M\ f && \text{(UC, E1NM)} \end{aligned}$$

Now, suppose that (E1D) holds; we show (J1S).

$$\begin{aligned} \mathit{ext}_M(\mathit{ext}_{NM}\ f) &= \mathit{ext}_{NM}\ (\mathit{ext}_{NM}\ f) \circ \mathit{map}_M\ \mathit{unit}_N && \text{(E1D)} \\ &= \mathit{ext}_{NM}\ f \circ \mathit{ext}_{NM}\ \mathit{id}\ \circ \mathit{map}_M\ \mathit{unit}_N && \text{(E3}'\text{NM)} \\ &= \mathit{ext}_{NM}\ f \circ \mathit{ext}_M\ \mathit{id}\ = \mathit{ext}_{NM}\ f \circ \mathit{join}_M && \text{(E1D, E4M)} \end{aligned}$$

Thus the *pext* construction is applicable whenever the *prod* construction is, and the converse is true if we can assume that (UC) and (MC) hold. We give another useful lemma.

Lemma 10. *Assume NM is constructed as in §3.1. If (MC) holds, then*

$$\mathit{pext}\ (g \circ f) = \mathit{pext}\ g \circ \mathit{map}_N\ f \qquad \text{(PO)}$$
$$\mathit{kmap}\ (g \circ f) = \mathit{kmap}\ g \circ \mathit{map}_N\ f \qquad \text{(KO)}$$

and if (UC) holds, then

$$\mathit{pext}\ f \circ \mathit{unit}_N = f \qquad \text{(E1K}'\text{)}$$
$$\mathit{ext}_{NM}\ f \circ \mathit{map}_M\ \mathit{unit}_N = \mathit{ext}_M\ f \qquad \text{(E1D)}$$

Proof. Using (PE), the following gives (PO).

$$\begin{aligned} \mathit{ext}_{NM}\ (g \circ f) \circ \mathit{unit}_M &= \mathit{ext}_{NM}\ g \circ \mathit{map}_{NM}\ f \circ \mathit{unit}_M && \text{(9NM)} \\ &= \mathit{ext}_{NM}\ g \circ \mathit{unit}_M \circ \mathit{map}_N\ f && \text{(MC,3M)} \end{aligned}$$

(KO):

$$\begin{aligned} \mathit{kmap}\ (g \circ f) &= \mathit{pext}\ (\mathit{unit}_{NM} \odot_M g \circ f) && \text{(E5K, A6M)} \\ &= \mathit{pext}\ (\mathit{unit}_{NM} \odot_M g) \circ \mathit{map}_N\ f && \text{(PO)} \\ &= \mathit{kmap}\ g \circ \mathit{map}_N\ f && \text{(E5K)} \end{aligned}$$

(E1K′):

$$\begin{aligned} \mathit{pext}\ f \circ \mathit{unit}_N &= \mathit{pext}\ f \odot_M (\mathit{unit}_M \circ \mathit{unit}_N) && \text{(A4}'\text{M)} \\ &= \mathit{pext}\ f \odot_M \mathit{unit}_{NM} = f && \text{(UC, E1K)} \end{aligned}$$

(E1D):

$$\begin{aligned} \mathit{ext}_{NM}\ f \circ \mathit{map}_M\ \mathit{unit}_N &= \mathit{ext}_M\ (\mathit{pext}\ f) \circ \mathit{map}_M\ \mathit{unit}_N && \text{(EC)} \\ &= \mathit{ext}_M\ (\mathit{pext}\ f \circ \mathit{unit}_N) = \mathit{ext}_M\ f && \text{(9M, E1K}'\text{)} \ \square \end{aligned}$$

Then the relationships between *prod* and *pext* are given by these equalities, which are equivalent when (PO) and (1N) hold. See Appendix §A.1 for details.

$$\mathit{prod} = \mathit{pext}\ \mathit{id} \qquad\qquad \mathit{pext}\ f = \mathit{prod} \circ \mathit{map}_N\ f$$

3.3 Lifting monad N to $\mathcal{K}_M$

By a *lifting* of one monad to another (maybe in different categories) we understand a functor F which is also a functor between their Kleisli categories. We consider whether the monad $N_{\mathcal{K}M}$ in the Kleisli category $\mathcal{K}_M$ is a lifting of the monad N. This would require

$$F\ id = unit_M \qquad \text{(LI)}$$
$$F\ (g \circ f) = F\ g \odot_M F\ f \qquad \text{(LO)}$$
$$F\ unit_N = unit_{NM} \qquad \text{(LU)}$$
$$F\ (g \odot_N f) = F\ g \odot_{NM} F\ f \qquad \text{(LA)}$$

Instead of (LA), we may use (LE), or (LM) and (LJ).

$$F\ (ext_N\ f) = pext\ f \qquad \text{(LE)}$$
$$F\ (map_N\ f) = kmap\ f \qquad \text{(LM)}$$
$$F\ join_N = kjoin \qquad \text{(LJ)}$$

Assuming (UC), $F = kl$, where $kl\ f = unit_M \circ f$ satisfies (LI), (LO) and (LU).

With $F = kl$, note that (LJ) is just (KJ) of Theorem 14, and, indeed, kl is a lifting if and only if (J2) (see §3.5) holds.

3.4 A generalisation of earlier axiom systems

We now present a generalisation of the system of axioms (1) to (8), and their equivalence to (E1) to (E5). This was motivated by the construction by Jones & Duponcheel [3, §3.3] using their function *dorp* (see §3.5).

We found that the rules (G1) to (G8), which are analogous to rules (1) to (8), are sufficient to establish that NM is a monad, without assuming that either N or M is even a premonad. In these rules, we make no assumptions about the functions we call ext_{NM}, $join_{NM}$, map_{NM}, $unit_{NM}$ or $unit_M$.

These rules also use three more functions of the following types:

$$\begin{aligned} dunit &: \alpha M \to \alpha NM \\ dmap &: (\alpha \to \beta M) \to (\alpha NM \to \beta NM) \\ djoin &: \alpha NNM \to \alpha NM \end{aligned}$$

$$dmap\ unit_M = id \qquad \text{(G1)}$$
$$dmap\ (f \circ h) = dmap\ f \circ map_{NM}\ h \qquad \text{(G2)}$$
$$dmap\ f \circ unit_{NM} = dunit \circ f \qquad \text{(G3)}$$
$$djoin \circ dmap\ (dmap\ f) = dmap\ f \circ join_{NM} \qquad \text{(G4)}$$
$$djoin \circ dunit = id \qquad \text{(G5)}$$
$$djoin \circ dmap\ unit_{NM} = id \qquad \text{(G6)}$$
$$djoin \circ dmap\ djoin = djoin \circ join_{NM} \qquad \text{(G7)}$$
$$ext_{NM}\ f = djoin \circ dmap\ f \qquad \text{(G8)}$$

Theorem 11. *Assume rules (G1) to (G8). Then ext_{NM}, $join_{NM}$, map_{NM} and $unit_{NM}$ give a monad NM, where also*

$$
\begin{aligned}
djoin &= ext_{NM}\ unit_M && \text{(G9)}\\
dmap\ f &= ext_{NM}\ (dunit \circ f) && \text{(G10)}\\
unit_{NM} &= dunit \circ unit_M && \text{(G11)}\\
map_{NM}\ f &= dmap\ (unit_M \circ f) && \text{(G12)}
\end{aligned}
$$

Proof. We prove rules (E1), (E2), (E3′), (E4) and (E5) for NM; this proof corresponds almost step-by-step to the proof of these rules from (1) to (8).

(E1NM): use (G8), (G3) and (G5).
(E2NM): use (G8) and (G6).
(E4NM): use (G8), (G4) and (G1).
(9NM): use (G8) and (G2).
(8NM): use (E4NM) and (9NM).
(E5NM): use (9NM) and (E2NM).
(G9): use (G8) and (G1).
(G11): use (G3) and (G1).
(G12): use (G2) and (G1).

$$
\begin{aligned}
\text{(G10): } ext_{NM}\ (dunit \circ f) &= ext_{NM}\ (dmap\ f \circ unit_{NM}) && \text{(G3)}\\
&= ext_{NM}\ (dmap\ f) \circ map_{NM}\ unit_{NM} && \text{(9NM)}\\
&= dmap\ f \circ join_{NM} \circ map_{NM}\ unit_{NM} && \text{(G8, G4)}\\
&= dmap\ f && \text{(8NM, E2NM)}
\end{aligned}
$$

$$
\begin{aligned}
\text{(E3′NM):}\quad & ext_{NM}\ (ext_{NM}\ g \circ f) \\
&= djoin \circ dmap\ (djoin \circ dmap\ g \circ f) && \text{(G8)}\\
&= djoin \circ join_{NM} \circ map_{NM}\ (dmap\ g \circ f) && \text{(G2, G7)}\\
&= djoin \circ djoin \circ dmap\ (dmap\ g \circ f) && \text{(8NM, G8)}\\
&= djoin \circ dmap\ g \circ join_{NM} \circ map_{NM}\ f && \text{(G2, G4)}\\
&= ext_{NM}\ g \circ ext_{NM}\ f && \text{(G8, 8NM)}\ \square
\end{aligned}
$$

The following converse result tells us when a monad NM can be defined in this way. Note that we are still not assuming that M or N is a monad.

Theorem 12. *Assume that NM is a monad. Also assume that rules (G5) and (G9) to (G11) hold. Then the remaining rules among (G1) to (G8) hold.*

Proof. (G1)**:** use (G10), (G11) and (E2NM).
(G2)**:** use (G10) and (9NM).
(G3)**:** use (G10) and (E1NM).
(G8)**:** use (G9), (G10), (E3′NM) and (G5).
(G6)**:** use (G8) and (E2NM).
(G4)/(G7)**:** use (G8), Theorem 2 for NM, (iii) $\Rightarrow$ (i), and (G10)/(G9). $\square$

Given a monad NM, we consider when it can be constructed using Theorem 11. First we note that if NM is constructed as in §3.1, and (UC) holds, then we can define functions *dunit*, *dmap* and *djoin* by (DU), (G10) and (G9). Then (G11) holds by (3M), and (G5) becomes (G5′) which holds by (E1D) (see Lemma 10) and (E2M). Thus Theorem 12 holds.

$$\begin{aligned} \mathit{dunit} &= \mathit{map}_M\ \mathit{unit}_N && \text{(DU)} \\ \mathit{ext}_{NM}\ \mathit{unit}_M \circ \mathit{map}_M\ \mathit{unit}_N &= \mathit{id} && \text{(G5}'\text{)} \end{aligned}$$

3.5 Relation to the *dorp* construction of Jones & Duponcheel

Jones & Duponcheel [3] also show how to define a compound monad using a function *dorp*, satisfying four rules D(1) to D(4) (see Appendix A.2), and, assuming (UC) and (MC) and that M is a premonad and N is a monad, show that the compound monads which can be defined using *dorp* are precisely those satisfying (J2). We show corresponding results about when a compound monad can be constructed using (G1) to (G8).

$$\begin{aligned} \mathit{join}_{NM} \circ \mathit{map}_{NM}\ (\mathit{map}_M\ \mathit{join}_N) &= \mathit{map}_M\ \mathit{join}_N \circ \mathit{join}_{NM} && \text{(J2)} \\ \mathit{ext}_{NM}\ (\mathit{map}_M\ \mathit{join}_N) &= \mathit{map}_M\ \mathit{join}_N \circ \mathit{join}_{NM} && \text{(J2}'\text{)} \end{aligned}$$

We first relate (J2) to the conditions of Theorem 12.

Lemma 13. *Assume that NM is a monad, that M is a premonad and N is a monad, and that (UC) holds. Then (J2) (equivalently, (J2′)) holds if and only if $\mathit{ext}_{NM}\ \mathit{unit}_M = \mathit{map}_M\ \mathit{join}_N$.*

Proof. $\Leftarrow$: by Theorem 2 for NM, (iii) $\Rightarrow$ (i).
$\Rightarrow$: by applying Theorem 2, (i) $\Rightarrow$ (ii), to (J2′), gives

$$\begin{aligned} \mathit{map}_M\ \mathit{join}_N &= \mathit{ext}_{NM}\ (\mathit{map}_M\ \mathit{join}_N \circ \mathit{unit}_{NM}) \\ &= \mathit{ext}_{NM}\ (\mathit{map}_M\ \mathit{join}_N \circ \mathit{unit}_M \circ \mathit{unit}_N) && \text{(UC)} \\ &= \mathit{ext}_{NM}\ (\mathit{unit}_M \circ \mathit{join}_N \circ \mathit{unit}_N) && \text{(3M)} \\ &= \mathit{ext}_{NM}\ \mathit{unit}_M && \text{(6N)}\ \square \end{aligned}$$

With the assumptions of Lemma 13, and assuming that (J2) holds, we can define functions *dunit*, *dmap* and *djoin* by (DU), (G10) and (DJ), then (G9), (G11) and (G5) hold, so Theorem 12 applies.

$$\begin{aligned} \mathit{djoin} &= \mathit{map}_M\ \mathit{join}_N && \text{(DJ)} \\ \mathit{dunit} &= \mathit{map}_M\ \mathit{unit}_N && \text{(DU)} \end{aligned}$$

Note that we have shown that *either* (J1) *or* (J2) can be used to show that Theorem 12 holds, and so NM can be constructed using Theorem 11. This is related to the interesting fact that the equality (G5′) can be proved using *either* (J1) *or* (J2), as it follows easily from (E1D) or from Lemma 13. The same holds of (DJK) (see §3.6).

Finally, if we can define a compound monad NM satisfying (G1) to (G8) by using (DJ) to define *djoin*, then (G9) holds by Theorem 11, and so (J2) holds.

We can define the function *dorp* of Jones & Duponcheel [3] in terms of *dmap* and vice versa, as follows, and get the results relating to D(1) to D(4). See Appendix §A.2 for details.

$$dorp = \ dmap\ id \qquad\qquad dmap\ f = \ dorp \circ map_{NM}\ f$$

3.6 When both constructions apply

We consider the situation when both the constructions of §3.1 and §3.4 apply, that is when both (J1) and (J2) hold. In this case, we have a distributive law for monads: see §3.8. We collect some results involving both constructions. Note particularly that not all the results require all the assumptions of the theorem. Indeed, it can be seen that if *djoin* is defined by (G9), then (DJK) has two proofs, as given, one relying on (J1) and one on (J2).

Theorem 14. *Assume that NM, M and N are monads, and that (UC), (MC), (J1) and (J2) hold. Define djoin by (G9) or (DJ), dmap by (G10), dunit by (DU), pext by (A5K) or (PE), kjoin by (E4K) and kmap by (E5K). Then*

$$\begin{aligned} dunit &= ext_M\ unit_{NM} && \text{(DUK)}\\ djoin &= ext_M\ kjoin && \text{(DJK)}\\ dmap\ f &= ext_M\ (kmap\ f) && \text{(DMK)}\\ kjoin &= unit_M \circ join_N && \text{(KJ)} \end{aligned}$$

Proof. Note that, by Lemma 13, any two of (J2), (G9) and (DJ) imply the third, and that, by (E6NM), (A5K) holds iff (PE) holds. By Lemma 9(i), (J1S) holds, and so Theorem 8 applies and (EC) holds.

(DUK): Use (DU), (UC) and (E5M).
(DJK): Use (G9), (EC) and (E4K).

(DMK):
$$\begin{aligned} ext_M\ (kmap\ f) &= ext_M\ (pext\ (unit_{NM} \odot_M\ f)) && \text{(E5K)}\\ &= ext_M\ (pext\ (dunit \circ f)) && \text{(E6M, DUK)}\\ &= ext_{NM}\ (dunit \circ f) = dmap\ f && \text{(EC, G10)} \end{aligned}$$

(KJ):
$$\begin{aligned} kjoin &= pext\ unit_M = ext_{NM}\ unit_M \circ unit_M && \text{(E4K, PE)}\\ &= map_M\ join_N \circ unit_M = unit_M \circ join_N && \text{(Lemma 13, 3M)} \end{aligned}$$

(DJK): Use (DJ), (E5M) and (KJ) □

3.7 Relation to the *swap* construction of Jones & Duponcheel

Jones & Duponcheel [3] also show how to define a compound monad using a function *swap*, satisfying four rules S(1) to S(4), (see Appendix A.3), and, assuming (UC) and (MC) and that M and N are monads, show that a compound monad can be defined using *swap* iff it satisfies (J1) and (J2).

Given a compound monad NM satisfying (J1) and (J2), Lemma 9(i) gives (J1S), and hence, by Theorem 8, NM can be defined using *kjoin* and *kmap*. Then, defining $\mathit{swap} = \mathit{kmap}\ \mathit{id}$ we can prove S(1) to S(4) of [3].

Conversely, given monads M and N and a function *swap*, we can define *kmap* in terms of *swap* as below, but to define the monad NM we also need to define *kjoin*, which we can do using (KJ).

The relationships between *swap* and *kmap* are given by these equalities, which are equivalent when (KO) and (1N) hold. See Appendix §A.3 for more details.

$$\mathit{swap} = \mathit{kmap}\ \mathit{id} \qquad\qquad \mathit{kmap}\ f = \mathit{swap} \circ \mathit{map}_N\ f$$

3.8 Relation to the distributive law for monads

Manes [6] describes a “distributive law” for monads, at Chapter 4, pages 311-2, Definition 3.6, and page 334, Exercise 6. This is also described by Barr & Wells in [1], §9.2. Under the conditions of Theorem 14, the compound monad is based on a distributive law. The distributive law, (ie, the natural transformation λ of [6] and [1]) is the polymorphic function *swap*. Further details are in Appendix §A.4.

When NM, M and N are monads, assuming that (MC) holds, Barr & Wells give conditions (C1) to (C5) for NM to be “compatible” with N and M. These are in effect the conditions of Theorem 14, though some of their conditions are redundant. Of their five conditions, condition (C1) is (in effect) (UC). Condition (C5) is (J2), and Lemma 13 shows that it is equivalent to (C2).

Condition (C3) is (E1DI), that is, the case $f = \mathit{id}$ of (E1D), and condition (C4) is (J1), that is, the case $f = \mathit{id}$ of (J1S). These also are equivalent: by Lemma 9, (ii) and (i) these are equivalent to (E1D) and (J1S) respectively, which are equivalent by Lemma 9(iii).

Barr & Wells give these conditions in category theory notation, which reveals a duality between (C2),(C5) on the one hand, and (C3),(C4) on the other.

As Michael Barr has observed, the same duality should hold between the proofs (C2) $\Leftrightarrow$ (C5) and (C3) $\Leftrightarrow$ (C4), when these are written out in the category theory notation (the proof (C3) $\Rightarrow$ (C4) is thanks to this observation). See Appendix §A.4 for these proofs.

4 Conclusion

By focussing on the Kleisli category of a monad and the functors to and from it we have provided simple proofs of some compound monad constructions. This provided an interesting application of Wadler's parametricity theorem. We have described a construction which applies to several compound monads which is a functor *pext* from the Kleisli category of the compound monad NM, into $\mathcal{K}_M$, the Kleisli category of M, and we have shown how this is simply a monad in $\mathcal{K}_M$. Under further conditions, this construction is is equivalent to the *prod* construction of Jones & Duponcheel [3]. We have shown how the “map” function of the monad in $\mathcal{K}_M$ is similarly related to the *swap* construction of [3]. We developed a set of axioms generalising those of a monad to describe a way of constructing a compound monad which is similarly related to the *dorp* construction of [3].

Acknowledgements I thank unnamed referees for drawing my attention to the work of Manes and of Barr & Wells on a distributive law for monads, and for alterting me to the semantic issues involved in Propositions 5 and 6.

References

1. Michael Barr and Charles Wells. Toposes, Triples and Theories. Springer-Verlag, 1983, or see `http://www.cwru.edu/artsci/math/wells/pub/ttt.html`
2. Jeremy Dawson. Isabelle proofs: `http://users.rsise.anu.edu.au/~jeremy/isabelle/monad/`.
3. Mark P. Jones and Luc Duponcheel, Composing Monads, Research Report YALEU/DCS/RR-1004, Yale University, December 1993
4. Sheng Liang, Paul Hudak, and Mark P Jones. Monad Transformers and Modular Interpreters. In Symposium on Principles of Programming Languages (POPL'95), 1995, 333–343.
5. Saunders MacLane. Categories for the Working Mathematician. Graduate Texts in Mathematics, Springer, 1971.
6. Ernest G Manes. Algebraic theories. Graduate Texts in Mathematics, Springer, 1976.
7. Eugenio Moggi. Computational lambda-calculus and monads. In Proceedings of Logic in Computer Science (LICS'89), 1989.
8. Philip Wadler. The Essence of Functional Programming. In Symposium on Principles of Programming Languages (POPL'92), 1992, 1–14.
9. Philip Wadler. Theorems for free! In 4'th International Conference on Functional Programming and Computer Architecture, ACM Press, 1989, 347–359.

A Relation between our constructions and others previously described

In this section we compare our constructions with those of Jones & Duponcheel [3], and also with the distributive law for monads of Manes [6]. and Barr & Wells [1, §9.2].

Throughout this section we assume at least that M and N are premonads, and that we have functions unit_{NM} and map_{NM} for which (UC) and (MC) hold. We note that when N and M are premonads, then (UC) and (MC) give that NM is a premonad [3, §3].

A.1 Relation between the *pext* construction and the *prod* construction of Jones & Duponcheel

In this subsection we further assume that M is a monad. Under these assumptions Jones & Duponcheel [3, §3.2,§4.1] show that the compound monads NM which can be defined using *prod* and the four rules P(1) to P(4) are precisely those satisfying (J1).

$$\begin{aligned}
\mathit{prod} \circ \mathit{map}_N\ (\mathit{map}_{NM}\ f) &= \mathit{map}_{NM}\ f \circ \mathit{prod} && \text{P(1)}\\
\mathit{prod} \circ \mathit{unit}_N\ &= \mathit{id} && \text{P(2)}\\
\mathit{prod} \circ \mathit{map}_N\ \mathit{unit}_{NM}\ &= \mathit{unit}_M && \text{P(3)}\\
\mathit{prod} \circ \mathit{map}_N\ \mathit{join}_{NM}\ &= \mathit{join}_{NM} \circ \mathit{prod} && \text{P(4)}
\end{aligned}$$

We relate *prod* to *pext*, and list other relevant equalities, with an easy lemma relating them.

$$\begin{aligned}
\mathit{prod} &= \mathit{pext}\ \mathit{id} && \text{(PP1)}\\
\mathit{pext}\ f &= \mathit{prod} \circ \mathit{map}_N\ f && \text{(PP2)}\\
\mathit{pext}\ (g \circ f) &= \mathit{pext}\ g \circ \mathit{map}_N\ f && \text{(PO)}\\
\mathit{pext}\ f \circ \mathit{unit}_N &= f && \text{(E1K}'\text{)}
\end{aligned}$$

Lemma 15. *(i) (PP2) holds iff (PP1) and (PO) hold.*
(ii) (E1K′) holds iff (E1K) holds.
(iii) assuming (PP2), P(2) holds iff (E1K′) holds.
(iv) assuming (PP2), P(3) is just (E2K).

Theorem 16. *Assume that NM is a compound monad, for which (J1) holds. Use (PE) to define pext, and (PP1) to define prod. Then (J1S) holds, and (PP2), (PO) and P(1) to P(4) hold.*

Proof. By Lemma 9(i), (J1S) holds. We now have that Theorems 8 and 7 hold, and Lemma 10 gives (PO), from which (PP2) follows. To prove P(1) and P(4), we use the following result.

$$\begin{aligned}
\mathit{pext}\ (\mathit{ext}_{NM}\ g) &= \mathit{pext}\ (g \odot_{NM}\ \mathit{id}) && \text{(A5NM)}\\
&= g \odot_{NM}\ \mathit{pext}\ \mathit{id} && \text{(E3K, E6K)}\\
&= \mathit{ext}_{NM}\ g \circ \mathit{prod} && \text{(E6NM, PP1)}
\end{aligned}$$

From this we use (PP2), and (E5NM) and (E4NM) respectively, to get P(1) and P(4). Finally, Lemma 15 gives P(2) and P(3). □

We note that our definition of *prod* gives $\mathit{prod} \ = \mathit{pext}\ \mathit{id} \ = \mathit{ext}_{NM}\ \mathit{id} \circ \mathit{unit}_M \ = \mathit{join}_{NM} \circ \mathit{unit}_M$ which is the definition used in [3, §4.1].

The following converse theorem is [3, §3.2].

Theorem 17. *Let function prod be given, and define* join_{NM} *by* $\mathit{join}_{NM} = \mathit{ext}_M\ \mathit{prod}$. *Let P(1) to P(4) be satisfied. Then* NM *is a monad, satisfying (J1).*

Proof. Define *pext* using (PP2). Then Lemma 15 gives (PO), (E1K) and (E2K).

To define ext_{NM}, we first show the equivalence of two likely definitions.

$$\begin{aligned}\mathit{join}_{NM} \circ \mathit{map}_{NM}\ f &= \mathit{ext}_M\ \mathit{prod} \circ \mathit{map}_M\ (\mathit{map}_N\ f) && \text{(MC)}\\ &= \mathit{ext}_M\ (\mathit{prod} \circ \mathit{map}_N\ f) = \mathit{ext}_M\ (\mathit{pext}\ f) && \text{(9M, PP2)}\end{aligned}$$

So we define ext_{NM} so that (8NM) and (EC) hold, and define $\odot_{NM}$ using (E6NM), from which (A5NM) follows.

$$\begin{aligned}\text{(E3K):}\quad \mathit{pext}\ (g \odot_{NM} f) &= \mathit{pext}\ (\mathit{ext}_{NM}\ g \circ f) && \text{(E6NM)}\\ &= \mathit{prod} \circ \mathit{map}_N\ (\mathit{join}_{NM} \circ \mathit{map}_{NM}\ g \circ f) && \text{(PP2, 8NM)}\\ &= \mathit{join}_{NM} \circ \mathit{prod} \circ \mathit{map}_N\ (\mathit{map}_{NM}\ g \circ f) && \text{(2N, P(4))}\\ &= \mathit{join}_{NM} \circ \mathit{map}_{NM}\ g \circ \mathit{prod} \circ \mathit{map}_N\ f && \text{(2N, P(1))}\\ &= \mathit{ext}_{NM}\ g \circ \mathit{pext}\ f && \text{(8NM, PP2)}\\ &= \mathit{ext}_M\ (\mathit{pext}\ g) \circ \mathit{pext}\ f = \mathit{pext}\ g \odot_M \mathit{pext}\ f && \text{(EC, E6M)}\end{aligned}$$

Thus Theorem 7 applies and so ext_{NM}, $\odot_{NM}$ and unit_{NM} are functions of a monad NM. To show that join_{NM} and map_{NM} are the *join* and *map* functions of this monad, we need (E4) and (E5) for NM.

Since (E4NM) is clear from (8NM), it remains to show (E5NM).

$$\begin{aligned}\mathit{ext}_{NM}\ (\mathit{unit}_{NM} \circ f) &= \mathit{ext}_M\ (\mathit{pext}\ \mathit{unit}_{NM} \circ \mathit{map}_N\ f) && \text{(EC, PO)}\\ &= \mathit{ext}_M\ (\mathit{unit}_M \circ \mathit{map}_N\ f) && \text{(E2K)}\\ &= \mathit{map}_M\ (\mathit{map}_N\ f) = \mathit{map}_{NM}\ f && \text{(E5M, MC)}\end{aligned}$$

Finally, (J1), and also (J1S), follow from Theorem 2 for M, (iii) ⇒ (i). □

A.2 Relation between the *dmap* construction and the *dorp* construction of Jones & Duponcheel

In this subsection, as well as assuming that M and N are premonads and that (UC) and (MC) hold, we further assume that M is a monad. Under these assumptions Jones & Duponcheel [3, §3.3,§4.2] show that the compound monads

NM which can be defined using *dorp* and the four rules D(1) to D(4) are precisely those satisfying (J2). We relate *dorp* to *dmap* as shown.

$$\begin{aligned}
dorp \circ map_{NM}\ (map_M\ f) &= map_{NM}\ f \circ dorp && \text{D(1)}\\
dorp \circ unit_{NM} &= map_M\ unit_N && \text{D(2)}\\
dorp \circ map_{NM}\ unit_M &= id && \text{D(3)}\\
dorp \circ join_{NM} &= join_{NM} \circ map_{NM}\ dorp && \text{D(4)}
\end{aligned}$$

$$\begin{aligned}
dorp &= dmap\ id && \text{(DD1)}\\
dmap\ f &= dorp \circ map_{NM}\ f && \text{(DD2)}
\end{aligned}$$

Note that, analogously to Lemma 15(i), (DD2) holds iff (DD1) and (G2) hold.

Theorem 18. *Assume that NM is a compound monad for which (J2′) holds. Define djoin by (G9) or (DJ), and dmap and dunit by (G10) and (DU). Then (DJ) and (G1) to (G12) hold. Further, if dorp is defined by (DD1), then (DD2) and D(1) to D(4) hold.*

Proof. (G11) holds, using (3M). The equivalence of (DJ), that is, $djoin = map_M\ join_N$, and (G9) follows from Lemma 13. Then $djoin \circ dunit = map_M\ join_N \circ map_M\ unit_N = id$, by (2M), (5N) and (1M), that is, (G5) holds. Thus the conditions of Theorem 12 are satisfied, so (G1) to (G8) also hold, as does, by Theorem 11, (G12).

Now, using (DD1), we get (DD2) from (G2). Then D(3) is just (G1), and D(4) follows from (G4), by (G8) and (8NM). Using (DU), (G3) gives D(2).

We prove D(1) by

$$\begin{aligned}
dorp \circ map_{NM}\ (map_M\ f) &= dmap\ (map_M\ f) && \text{(DD2)}\\
&= ext_{NM}\ (dunit \circ map_M\ f) && \text{(G10)}\\
&= ext_{NM}\ (map_{NM}\ f \circ dunit) && \text{(MC, 2M, 3N, DU)}\\
&= map_{NM}\ f \circ ext_{NM}\ dunit && \text{(E5NM, E3′NM)}\\
&= map_{NM}\ f \circ dorp && \text{(G10, DD1)} \ \square
\end{aligned}$$

Note that our definition of *dorp* gives $dorp = dmap\ id = ext_{NM}\ dunit$, which corresponds to the definition used in [3, §4.2].

Jones & Duponcheel [3, §3.3] also show how to define a compound monad from a function *dorp*, satifying four rules D(1) to D(4), assuming that M is a premonad and N is a monad. They use (UC) and (MC) as definitions, and defining $join_{NM}$ by

$$join_{NM} = map_M\ join_N \circ dorp \qquad \text{(JD)}$$

They show that in such a compound monad, (J2) holds. We can prove this theorem via rules (G1) to (G8) and Theorem 11. However in doing so, we have to prove (4NM) on the way to proving (G4) which is rather unsatisfying.

Theorem 19. *[3, §3.3] Let N be a monad and M a premonad. Define $unit_{NM}$ and map_{NM} by (UC) and (MC). Define djoin and dunit by (DJ) and (DU). Let the function dorp satisfy D(1) to D(4), and define dmap by (DD2). Define $join_{NM}$ by (JD). Then NM is a monad satisfying (G1) to (G12) and (J2).*

Proof. (sketched: also proved in Isabelle). Firstly, note that NM is a premonad. From that fact, using (DD2), we can get (DD1) from (1NM), (G2) from (2NM), (G3) from (3NM) and D(2), while (G1) is just D(3).

(G5): Use (DJ), (DU), (2M), (5N) and (1M).

(G6): Use (DJ), (UC), (G2), (G1), (MC), (2M), (6N) and (1M).

We use (G8) as a definition, noting also that (8NM) holds, as

$$\begin{aligned} \mathit{join}_{NM} \circ \mathit{map}_{NM}\ f &= \mathit{map}_M\ \mathit{join}_N \circ \mathit{dorp} \circ \mathit{map}_{NM}\ f && \text{(JD)} \\ &= \mathit{djoin} \circ \mathit{dmap}\ f && \text{(DJ, DD2)} \end{aligned}$$

From (G8) and (G1) we get (G9).

To prove (G4) and (G7) we derive initially

$$\begin{aligned} \mathit{djoin} \circ \mathit{dmap}\ \mathit{map}_M\ g &= \mathit{map}_M\ \mathit{join}_N \circ \mathit{map}_{NM}\ g \circ \mathit{dorp} && \text{(DJ, DD2, D(1))} \\ &= \mathit{map}_M\ (\mathit{join}_N \circ \mathit{map}_N\ g) \circ \mathit{dorp} && \text{(MC, 2M)} \\ &= \mathit{map}_M\ (\mathit{ext}_N\ g) \circ \mathit{dorp} && \text{(8N)} \end{aligned}$$

Then, to get (G7), we set $g = \mathit{join}_N$ in this to get, on the right-hand side,

$$\begin{aligned} \mathit{map}_M\ (\mathit{ext}_N\ \mathit{join}_N) \circ \mathit{dorp} &= \mathit{map}_M\ (\mathit{join}_N \circ \mathit{join}_N) \circ \mathit{dorp} && \text{(7N, 8N)} \\ = \mathit{djoin} \circ \mathit{map}_M\ \mathit{join}_N \circ \mathit{dorp} &= \mathit{djoin} \circ \mathit{join}_{NM} && \text{(2M, DJ, JD)} \end{aligned}$$

and we set $g = \mathit{map}_N\ f$ in it to get, by a very similar argument, using (4N) instead of (7N), $\mathit{djoin} \circ \mathit{dmap}\ \mathit{map}_{NM}\ f = \mathit{map}_{NM}\ f \circ \mathit{join}_{NM}$. Rewriting using (G8) and (8NM), we have (4NM). Thus the property of h that $\mathit{join}_{NM} \circ \mathit{map}_{NM}\ h = h \circ \mathit{join}_{NM}$ holds for $h = \mathit{dorp}$, by D(4), and for $h = \mathit{map}_{NM}\ f$, above. So it is easy to see that this property holds for their composition, $h = \mathit{dorp} \circ \mathit{map}_{NM}\ f$, which, again using (G8) and (8NM), gives us (G4).

Thus Theorem 11 applies. As we now have both (G9) and (DJ), Lemma 13 shows that (J2) holds. □

A.3 Relation between the *kmap* construction and the *swap* construction of Jones & Duponcheel

Throughout this section we make the assumptions of §A.1 and of §A.2. That is, we assume that M and N are monads, and that (UC) and (MC) hold. Under these assumptions Jones & Duponcheel [3, §3.4,§4.3] show that the compound monads NM which can be defined using *swap* and the four rules S(1) to S(4) (where *prod* and *dorp* are defined in terms of *swap* as shown) are precisely those satisfying (J1) and (J2).

$$\begin{aligned} \mathit{swap} \circ \mathit{map}_N\ (\mathit{map}_M\ f) &= \mathit{map}_{NM}\ f \circ \mathit{swap} && \text{S(1)} \\ \mathit{swap} \circ \mathit{unit}_N &= \mathit{map}_M\ \mathit{unit}_N && \text{S(2)} \\ \mathit{swap} \circ \mathit{map}_N\ \mathit{unit}_M &= \mathit{unit}_M && \text{S(3)} \\ \mathit{prod} \circ \mathit{map}_N\ \mathit{dorp} &= \mathit{dorp} \circ \mathit{prod} && \text{S(4)} \end{aligned}$$

$$\mathit{prod} = \mathit{map}_M\ \mathit{join}_N \circ \mathit{swap} \qquad\qquad \mathit{dorp} = \mathit{ext}_M\ \mathit{swap}$$

$$\begin{aligned} \mathit{swap} &= \mathit{kmap}\ \mathit{id} && \text{(KS1)} \\ \mathit{kmap}\ f &= \mathit{swap} \circ \mathit{map}_N\ f && \text{(KS2)} \\ \mathit{kmap}\ (g \circ f) &= \mathit{kmap}\ g \circ \mathit{map}_N\ f && \text{(KO)} \end{aligned}$$

Note that, analogously to Lemma 15(i), (KS2) holds iff (KS1) and (KO) hold.

Theorem 20. *With the assumptions and definitions of Theorem 14, further define swap from kmap by (KS1). Then S(1) to S(4) hold.*

Proof. By Lemma 9(i), Theorem 8 applies. So by Lemma 10, (KO) and so (KS2) hold. So S(3) is just (1K). Now by (E5K), (E6M) and (DUK), $\mathit{kmap}\ f = \mathit{pext}\ (\mathit{unit}_{NM} \odot_M f) = \mathit{pext}\ (\mathit{dunit} \circ f)$, so $\mathit{swap} = \mathit{pext}\ \mathit{dunit}$ and, by (E1K′), $\mathit{swap} \circ \mathit{unit}_N = \mathit{dunit}$ and S(2) follows.

We show S(1), after simplifying it using (KS2), as follows:

$$\begin{aligned} \mathit{kmap}\ (\mathit{map}_M\ f) &= \mathit{kmap}\ ((\mathit{unit}_M \circ f) \odot_M \mathit{id}) && \text{(E5M, A5M)} \\ &= \mathit{kmap}\ (\mathit{unit}_M \circ f) \odot_M \mathit{kmap}\ \mathit{id} && \text{(2K)} \\ &= (\mathit{kmap}\ \mathit{unit}_M \circ \mathit{map}_N\ f) \odot_M \mathit{swap} && \text{(KO, KS1)} \\ &= \mathit{ext}_M\ (\mathit{unit}_M \circ \mathit{map}_N\ f) \circ \mathit{swap} && \text{(1K, E6M)} \\ &= \mathit{map}_{NM}\ f \circ \mathit{swap} && \text{(E5M, MC)} \end{aligned}$$

Now, for S(4) we first use our definitions of *dorp* and *prod*. By (DD1) and (DMK), $\mathit{dorp} = \mathit{ext}_M\ (\mathit{kmap}\ \mathit{id})$, and by (E5K) let $\mathit{kmap}\ \mathit{id}$ be $\mathit{pext}\ g$, so we have

$$\begin{aligned} \mathit{prod} \circ \mathit{map}_M\ \mathit{dorp} &= \mathit{pext}\ (\mathit{ext}_M\ (\mathit{kmap}\ \mathit{id})) && \text{(PP2)} \\ &= \mathit{pext}\ (\mathit{pext}\ g \odot_M \mathit{id}) && \text{(A5M)} \\ &= \mathit{pext}\ g \odot_M \mathit{pext}\ \mathit{id} && \text{(E3′K)} \\ &= \mathit{ext}_M\ (\mathit{kmap}\ \mathit{id}) \circ \mathit{prod} && \text{(PP1, E6M)} \end{aligned}$$

Finally, to show S(4) precisely as it is stated, we need to check that the various definitions of *prod* and *dorp* are consistent. By our definitions,

$$\begin{aligned} \mathit{prod} &= \mathit{pext}\ \mathit{id} = \mathit{kjoin} \odot_M \mathit{kmap}\ \mathit{id} && \text{(PP1, 8K)} \\ &= \mathit{ext}_M\ \mathit{kjoin} \circ \mathit{swap} = \mathit{map}_M\ \mathit{join}_N \circ \mathit{swap} && \text{(E6M, KS1, DJK, DJ)} \end{aligned}$$

$$\mathit{dorp} = \mathit{dmap}\ \mathit{id} = \mathit{ext}_M\ (\mathit{kmap}\ \mathit{id}) = \mathit{ext}_M\ \mathit{swap} \qquad \text{(DD1, DMK, KS1)}$$

so our definitions are consistent with those used above in stating S(4). □

We showed in the proof above that $\mathit{swap} = \mathit{pext}\ \mathit{dunit}$ which is consistent with the definition of *swap* used in [3, §4.3].

Conversely, from S(1) to S(4), (KS2) and (KJ) it is possible to prove the rules of (1K) to (7K) directly. These proofs have been done in Isabelle, see [2], functor `fromSwap`. Alternatively, see Jones & Duponcheel [3, §3.4] for proofs from S(1) to S(4) that NM is a monad.

Another interesting result is that both sides of S(4) equal $\mathit{swap} \odot_{NM} \mathit{swap}$. The key step to prove it is to modify the proof of S(4) above, to give

$$\begin{aligned} \mathit{prod} \circ \mathit{map}_M\ \mathit{dorp} &= \mathit{pext}\ (\mathit{pext}\ g \odot_M \mathit{id}) && \text{(as above)} \\ &= \mathit{pext}\ (\mathit{pext}\ g) \odot_M \mathit{kmap}\ \mathit{id} && \text{(9K)} \\ &= \mathit{pext}\ \mathit{swap} \odot_M \mathit{swap} = \mathit{swap} \odot_{NM} \mathit{swap} && \text{(KS1, E6K)} \end{aligned}$$

A.4 A distributive law for monads

Manes [6] describes a "distributive law" for monads, at Chapter 4, pages 311-2, Definition 3.6, and page 334, Exercise 6. This is also described by Barr & Wells in [1], §9.2, where rules (D1) to (D4) are given. His natural transformation λ is also the polymorphic function *swap*. Translating these rules into our terminology, and calling (D1) to (D4) (BWD1) to (BWD4), we have

$$\begin{aligned}
swap \circ map_N\ unit_M &= unit_M && \text{(BWD1)}\\
swap \circ unit_N &= map_M\ unit_N && \text{(BWD2)}\\
swap \circ map_N\ join_M &= join_M \circ map_M\ swap \circ swap && \text{(BWD3)}\\
swap \circ join_N &= map_M\ join_N \circ swap \circ map_N\ swap && \text{(BWD4)}
\end{aligned}$$

Of these, (BWD2) and (BWD4) are the diagrams in [6, Definition 3.16], and (BWD1) and (BWD3) are the diagrams in Exercise 6 on page 334 of [6]. He additionally specifies that *swap* is a natural transformation.

Theorem 21. *With the assumptions and definitions of Theorem 20, (BWD1) to (BWD4) hold and swap is a natural transformation.*

Proof. From Theorem 20 we have S(1) to S(4). That *swap* is a natural transformation is just S(1), and (BWD2) and (BWD1) are just S(2) and S(3). We can translate (BWD3) to $kmap\ (id \odot_M id) = kmap\ id \odot_M kmap\ id$, an instance of (2K). To prove (BWD4), we have, by Theorem 2 for $\mathcal{K}_M$, (iii) ⇒ (i), and (E5K),

$$\begin{aligned}
pext\ (kmap\ id) &= kmap\ id \odot_M kjoin\\
&= kmap\ id \odot_M unit_M \circ join_N && \text{(KJ, A6M)}\\
&= swap \circ join_N && \text{(KS1, A1M)}
\end{aligned}$$

Now $pext\ (kmap\ id) = prod \circ map_N\ swap$ which is, by the expression for *prod* in §A.3 above, equal to the left-hand side of (BWD4). □

Conversely, from S(1) and (BWD1) to (BWD4), (KS2) and (KJ) it is possible to prove the rules of (1K) to (7K), and the resulting compound monad satisfies (J1) and (J2) . These proofs have been done in Isabelle, see [2], functors `fromSextDL` and `fromSextF`.

For monads N, M and NM, where (MC) is assumed, Barr & Wells [1, §9.2] give five conditions, (C1) to (C5), for the existence of a distributive law, but in fact (C2) ⇔ (C5) and (C3) ⇔ (C4), as noted in §3.8. These conditions are given below, as are the proofs of these equivalences, in the category theory notation.

Michael Barr has observed that there is a duality between (C2,C5) and (C3,C4), when these are written in category theory notation, and so the same duality should hold between the proofs of implications between these conditions. The proof (C3) ⇒ (C4) is due to this observation. Interestingly, while the proofs are dual in a notational sense, the reasons for each step of the proofs are not so.

We indicate the correspondence between the category theory notation and the notation of this paper.

μ	$join_{NM} : \alpha NMNM \rightarrow \alpha NM$	
μF	$join_{NM} : \alpha FNMNM \rightarrow \alpha FNM$	
μ, μ_1, μ_2	$join_{NM}, join_N, join_M$	
η, η_1, η_2	$unit_{NM}, unit_N, unit_M$	
T, T_1, T_2	map_{NM}, map_N, map_M	
$T = T_2 T_1$	$map_{NM} = map_M \circ map_N$	(MC)
$\mu \circ \eta T = id$	$join \circ unit = id$	(5)
$\mu \circ T\eta = id$	$join \circ map\ unit = id$	(6)
$\mu \circ T\mu = \mu \circ \mu T$	$join \circ map\ join = join \circ join$	(7)

We first list the conditions (C1) to (C5).

$$\eta = T_2\eta_1 \circ \eta_2 = \eta_2 T_1 \circ \eta_1 \tag{C1}$$

$$\mu \circ T\eta_2 T_1 = T_2\mu_1 \tag{C2}$$

$$\mu \circ T_2\eta_1 T = \mu_2 T_1 \tag{C3}$$

$$\mu_2 T_1 \circ T_2\mu = \mu \circ \mu_2 T_1 T \tag{C4}$$

$$T_2\mu_1 \circ \mu T_1 = \mu \circ T T_2 \mu_1 \tag{C5}$$

Finally we show the proofs of equivalence. Note the duality between the notations of the proofs. Steps with no reasons given rely on either T or T_i being functors ($T(\phi \circ \psi) = T\phi \circ T\psi$) or on the definition of composition of natural transformations ($(\phi \circ \psi)T = \phi T \circ \psi T$). Observe how the duality between the proofs also uses the duality between rules (5) and (6), and the self-duality of rule (7) (see the table above). But also note the duality between the specific uses made of the naturality of η_1 and of μ.

(C4) ⇒ (C3):

$$\begin{aligned}
\mu \circ T_2\eta_1 T &= \mu \circ (\mu_2 \circ T_2\eta_2)\, T_1 T \circ T_2\eta_1 T && \text{(6 for } T_2\text{)}\\
&= \mu \circ \mu_2 T_1 T \circ T_2\eta_2 T_1 T \circ T_2\eta_1 T\\
&= \mu \circ \mu_2 T_1 T \circ T_2\,(\eta_2 T_1 \circ \eta_1)\, T\\
&= \mu \circ \mu_2 T_1 T \circ T_2\eta T && \text{(C1)}\\
&= \mu_2 T_1 \circ T_2\mu \circ T_2\eta T && \text{(C4)}\\
&= \mu_2 T_1 \circ T_2\,(\mu \circ \eta T) = \mu_2 T_1 && \text{(5 for } T\text{)}
\end{aligned}$$

(C3) ⇒ (C4):

$$\begin{aligned}
\mu_2 T_1 \circ T_2\mu &= \mu \circ T_2\eta_1 T \circ T_2\mu && \text{(C3)}\\
&= \mu \circ T_2\,(\eta_1 T \circ \mu)\\
&= \mu \circ T_2\,(T_1\mu \circ \eta_1 T^2) && (\eta_1 \text{ natural})\\
&= \mu \circ T\mu \circ T_2\eta_1 T^2\\
&= \mu \circ \mu T \circ T_2\eta_1 T^2 && \text{(7 for } T\text{)}\\
&= \mu \circ \mu_2 T_1 T && \text{(C3)}
\end{aligned}$$

(C5) ⇒ (C2):
$$\begin{aligned}
\mu \circ T\eta_2 T_1 &= \mu \circ TT_2\,(\mu_1 \circ \eta_1 T_1) \circ T\eta_2 T_1 && (5 \text{ for } T_1)\\
&= \mu \circ TT_2\mu_1 \circ TT_2\eta_1 T_1 \circ T\eta_2 T_1 \\
&= \mu \circ TT_2\mu_1 \circ T\,(T_2\eta_1 \circ \eta_2)\,T_1 \\
&= \mu \circ TT_2\mu_1 \circ T\eta T_1 && (\text{C1})\\
&= T_2\mu_1 \circ \mu T_1 \circ T\eta T_1 && (\text{C5})\\
&= T_2\mu_1 \circ (\mu \circ T\eta)\,T_1 = T_2\mu_1 && (6 \text{ for } T)
\end{aligned}$$

(C2) ⇒ (C5):
$$\begin{aligned}
T_2\mu_1 \circ \mu T_1 &= \mu \circ T\eta_2 T_1 \circ \mu T_1 && (\text{C2})\\
&= \mu \circ (T\eta_2 \circ \mu)\,T_1 \\
&= \mu \circ (\mu T_2 \circ T^2\eta_2)\,T_1 && (\mu \text{ natural})\\
&= \mu \circ \mu T \circ T^2\eta_2 T_1 \\
&= \mu \circ T\mu \circ T^2\eta_2 T_1 && (7 \text{ for } T)\\
&= \mu \circ TT_2\mu_1 && (\text{C2})
\end{aligned}$$

B Some examples using rules (A1) to (A4)

In this section we give some examples of monads which are easily shown to be monads using rules (A1) to (A4).

The two examples in §B.2 and §B.3 are (in different senses) compound monads. In both cases M is an arbitrary monad; we don't explicitly use the monad N, but proceed directly to define the compound monad N_M. The simple monad N will be a special case of N_M, which could be obtained by setting M to be the identity monad (in which *unit*, *ext*, *map* and *join* are all the identity function).

B.1 Example: the Continuation Monad

The continuation monad (eg, [8, §3.1]) is given by $\alpha K = (\alpha \to \mathsf{Ans}) \to \mathsf{Ans}$, where Ans is a fixed type, so $\alpha \to \beta K = \alpha \to (\beta \to \mathsf{Ans}) \to \mathsf{Ans}$. We use the function C, of type $(\gamma \to \delta \to \mathsf{Ans}) \to (\delta \to \gamma \to \mathsf{Ans})$, defined by $C\,f\,x\,y = f\,y\,x$. Note that $C\ (C\ f) = f$, so C is 1-1, and $C\ x = C\ y \Rightarrow x = y$. Then we define $\odot_K$ by $C\ (g \odot_K f) = C\,f \circ C\ g$, and $unit_K$ by $C\ unit_K = id$.

The $\odot$ rules are then easily proved. For each rule $lhs = rhs$, as C is 1-1, it suffices to prove $C(lhs) = C(rhs)$.

for (A1): $C\ (f \odot_K unit_K) = C\ unit_K \circ C\,f = id \circ C\,f = C\,f$
for (A2): $C\ (unit_K \odot_K f) = C\,f \circ C\ unit_K = C\,f \circ id = C\,f$
for (A3): $C\ (h \odot_K (g \odot_K f)) = (C\,f \circ C\ g) \circ C\ h =$
$C\,f \circ (C\ g \circ C\ h) = C\ ((h \odot_K g) \odot_K f)$

It is simple but tedious to confirm (A4) directly. We now check that this definition corresponds to the usual ones. Using (A5), we get

$unit_K\ a\ c = C\ id\ a\ c = id\ c\ a = c\ a$, and
$ext_K\ g\ k\ c = (g \odot_K id)\ k\ c = C\ (C\ id \circ C\ g)\ k\ c =$
$(C\ id \circ C\ g)\ c\ k = C\ id\ (C\ g\ c)\ k = id\ k\ (C\ g\ c) = k\ (\lambda f.\ g\ f\ c)$

Both of these agree with the definitions in [8, §3.1].

B.2 Example: the Compound State Monad

Let M be a monad. Let State be a fixed type, representing, for example, a program state. Then the state monad is given by the type $\alpha S = \mathsf{State} \to \alpha{*}\mathsf{State}$ (where $\alpha * \beta$ denotes a pair type). This represents a computation which takes a program state and returns a result of interest and a new state. The "compound" state monad is given by the type $\alpha S_M = \mathsf{State} \to (\alpha * \mathsf{State})M$. (Note that because αS_M is neither αSM nor αMS, §3.1 and following are *not* relevant to this example, which perhaps should not be called a "compound monad"). If, for example, M is the list monad, S_M could be used to represent a non-deterministic program, which may behave in any of a number of possible ways, each way returning a result and new state.

To show that this is a monad, we use the functions *curry* and *unc*

$$\mathit{curry}\ g\ x\ y = g\ (x, y) \qquad\qquad \mathit{unc}\ f\ (x, y) = f\ x\ y$$

noting that these are mutually inverse (so, in particular, *unc* is 1-1). Consider a function $f : \alpha \to \beta S_M$. Then $\mathit{unc}\ f$ is of type $\alpha * \mathsf{State} \to (\beta * \mathsf{State})\ M$. Given that M is a monad, it is easy to see how to compose functions of such a type – we define $\odot_{SM}$ and unit_{SM} by

$$\begin{aligned} \mathit{unc}\ (g \odot_{SM} f) &= \mathit{unc}\ g \odot_M \mathit{unc}\ f \\ \mathit{unc}\ \mathit{unit}_{SM} &= \mathit{unit}_M \end{aligned}$$

The $\odot$ rules are then easily proved. For each rule $\mathit{lhs} = \mathit{rhs}$, as *unc* is 1-1, it suffices to prove $\mathit{unc}\ (\mathit{lhs}) = \mathit{unc}\ (\mathit{rhs})$. The proofs use the corresponding rules for the monad M.

for (A1) and (A2):

$$\begin{aligned} &\mathit{unc}\ (f \odot_{SM} \mathit{unit}_{SM}) = \mathit{unc}\ f \odot_M \mathit{unc}\ \mathit{unit}_{SM} = \mathit{unc}\ f \odot_M \mathit{unit}_M = \mathit{unc}\ f \\ &\mathit{unc}\ (\mathit{unit}_{SM} \odot_{SM} f) = \mathit{unc}\ \mathit{unit}_{SM} \odot_M \mathit{unc}\ f = \mathit{unit}_M \odot_M \mathit{unc}\ f = \mathit{unc}\ f \end{aligned}$$

for (A3): $\mathit{unc}\ (h \odot_{SM} (g \odot_{SM} f)) = \mathit{unc}\ h \odot_M (\mathit{unc}\ g \odot_M \mathit{unc}\ f) =$
$(\mathit{unc}\ h \odot_M \mathit{unc}\ g) \odot_M \mathit{unc}\ f = \mathit{unc}\ ((h \odot_{SM} g) \odot_{SM} f)$

It is simple but tedious to confirm (A4) directly. We need only check that this definition corresponds to the usual ones. Using (A5),

$\mathit{unit}_{SM}\ a\ s = \mathit{curry}\ \mathit{unit}_M\ a\ s = \mathit{unit}_M\ (a{,}s)$, and
$\mathit{ext}_{SM}\ k\ m\ s = (k \odot_{SM} \mathit{id})\ m\ s = \mathit{curry}\ (\mathit{unc}\ k \odot_M \mathit{unc}\ \mathit{id})\ m\ s =$
$(\mathit{ext}_M\ (\mathit{unc}\ k) \circ \mathit{unc}\ \mathit{id})\ (m{,}s) = \mathit{ext}_M\ (\lambda(a, t).\ k\ a\ t)\ (m\ s)$

Both of these agree with the definitions in [4, §7.1] (modified to switch the members of pairs).

B.3 Example: the Compound Reader Monad

Again, let M be a monad. Let Env be a fixed type. Then the environment, or reader, monad is given by the type $\mathsf{Env} \to \alpha$ (evaluating any monadic value also involves reading the environment). The reader monad is given by the type $\alpha R = \mathsf{Env} \to \alpha$, and the compound reader monad ([4, §7.2], [3, §6.3]) is given by $\alpha R_M = \mathsf{Env} \to \alpha M = (\alpha M)R$.

To show that this is a monad, we define a function $ap_e\ f\ a = f\ a\ e$ (for $e : \mathsf{Env}$), of type $(\gamma \to \mathsf{Env} \to \delta) \to (\gamma \to \delta)$. In similar proofs earlier, we had a function which was 1-1; here we need to use the property

$$(\forall e. ap_e\ f = ap_e\ g) \Rightarrow f = g$$

We define $\odot_{RM}$ and $unit_{RM}$ by

$$\begin{aligned} (g \odot_{RM} f)\ a\ e &= (ap_e\ g \odot_M\ ap_e\ f)\ a \\ unit_{RM}\ a\ e &= unit_M\ a \end{aligned}$$

that is,

$$\begin{aligned} ap_e\ (g \odot_{RM} f) &= ap_e\ g \odot_M\ ap_e\ f \\ ap_e\ unit_{RM} &= unit_M \end{aligned}$$

The $\odot$ rules are then easily proved. For each rule $lhs = rhs$, it suffices to let e be arbitrary, and then to prove $ap_e(lhs) = ap_e(rhs)$. The proofs use the corresponding rules for the monad M.

for (A1) and (A2):

$$ap_e\ (f \odot_{RM} unit_{RM}) = ap_e\ f \odot_M\ ap_e\ unit_{RM} = ap_e\ f \odot_M\ unit_M = ap_e\ f$$
$$ap_e\ (unit_{RM} \odot_{RM} f) = ap_e\ unit_{RM} \odot_M\ ap_e\ f = unit_M \odot_M\ ap_e\ f = ap_e\ f$$

for (A3): $ap_e\ (h \odot_{RM} (g \odot_{RM} f)) = ap_e\ h \odot_M\ (ap_e\ g \odot_M\ ap_e\ f) =$
$(ap_e\ h \odot_M\ ap_e\ g) \odot_M\ ap_e\ f = ap_e\ ((h \odot_{RM} g) \odot_{RM} f)$

It is straightforward to confirm (A4) directly. We need only check that this definition corresponds to the usual ones. Using (A5),

$unit_{RM}\ a\ e = unit_M\ a$, and
$ext_{RM}\ k\ m\ e = (k \odot_{RM} id)\ m\ e = (ap_e\ k \odot_M\ ap_e\ id)\ m =$
$(ext_M\ (ap_e\ k) \circ ap_e\ id)\ m = ext_M\ (\lambda a.\ k\ a\ e)\ (m\ e)$

Both of these agree with the definitions in [4, §7.2].

C Examples of the Compound Monad Constructions

C.1 Example: the Compound Exception Monad

The exception (or error) monad is given by

```
datatype α E = Ok of α | Err of string
```

with $unit_E\ a = Ok\ a$.

With M an arbitrary monad we define the compound monad αEM by

$$\begin{array}{rlr} \textit{pext } f \textit{ (Ok a)} &= f\ a & \textit{(pext-Ok)} \\ \textit{pext } f \textit{ (Err msg)} &= \textit{unit}_M \textit{ (Err msg)} & \textit{(pext-Err)} \end{array}$$

(Letting M be the identity monad gives us the definition of ext_E, for then *pext* is ext_E). We also define $unit_{EM}$ and $\odot_{EM}$ by (UC) and (E6K). Then we prove the *pext* axioms (E1K) to (E3K) as follows. (E1K) is just (*pext-Ok*). Using the definitions just mentioned, we get (E2K) from

$$\begin{array}{c} \textit{pext unit}_{EM} \textit{ (Ok a)} = \textit{unit}_{EM}\ a = \textit{unit}_M \textit{ (Ok a)} \\ \textit{pext unit}_{EM} \textit{ (Err msg)} = \textit{unit}_M \textit{ (Err msg)} \end{array}$$

and using these definitions and also (E6M) we get (E3K) from

$$\begin{array}{rlr} \textit{pext } (g \odot_{EM} f) \textit{ (Ok a)} &= (\textit{pext } g \odot_M f)\ a & \textit{(pext-Ok, E6K)} \\ &= \textit{ext}_M\ (\textit{pext } g)\ (f\ a) & \textit{(E6M)} \\ &= \textit{ext}_M\ (\textit{pext } g)\ (\textit{pext } f \textit{ (Ok a)}) & \textit{(pext-Ok)} \\ &= (\textit{pext } g \odot_M \textit{pext } f) \textit{ (Ok a)} & \textit{(E6M)} \end{array}$$

$$\begin{array}{rlr} \textit{pext } (g \odot_{EM} f) \textit{ (Err msg)} &= \textit{unit}_M \textit{ (Err msg)} & \textit{(pext-Err)} \\ &= \textit{pext } g \textit{ (Err msg)} & \textit{(pext-Err)} \\ &= \textit{ext}_M\ (\textit{pext } g)\ (\textit{unit}_M \textit{ (Err msg)}) & \textit{(E1)} \\ &= \textit{ext}_M\ (\textit{pext } g)\ (\textit{pext } f \textit{ (Err msg)}) & \textit{(pext-Err)} \\ &= (\textit{pext } g \odot_M \textit{pext } f) \textit{ (Err msg)} & \textit{(E6M)} \end{array}$$

C.2 Other Examples; Isabelle Proofs

Example: the Compound Writer Monad Let Monoid be a type representing a monoid, where we use 0 for the identity value and + for the associative binary operation. Then let $\alpha W = (\mathsf{Monoid} * \alpha)$ and $\alpha WM = (\alpha W)M$. A function $f : \alpha \to \beta W$ represents a function from α to β which also produces some output m of type Monoid; when $ext_W\ f$ acts on *(acc, a)*, where *acc* represents output already accumulated, the output m is added to that in *acc*. That is, if $f\ a = (m, b)$, then $ext_W\ f\ (acc,a) = (acc + m,\ b)$. Naturally, $unit_W\ a = (0, a)$.

The compound writer monad is given by

$$\textit{pext } f\ (acc,a) = \textit{map}_M\ (\lambda(m,b).\ (acc + m,\ b))\ (f\ a)$$

That is, the addition of the new output m to the accumulated output *acc* is done "under the hood" of the monad M. This makes the proof of the axioms for *pext* more difficult, and we omit them. (Jones & Duponcheel [3, §6.5.1] also omit proofs for this monad).

Example: the Compound Reader Monad This was discussed in §B.3. We write the monad type as αNR, where R takes the place of M in our general results. Note that, here, the monad R is fixed, while N is arbitrary. The *pext* construction applies to this compound monad, but to use it, we need to show

(i) that R is a monad

(ii) that N is a monad
(iii) that the axioms for *pext*, (E1K) to (E3K), hold

Our treatment of compound monads using the *pext* axioms requires, as such, only (i) and (iii), but we need (ii) to prove (iii). These proofs are, however, not difficult, and are omitted.

Isabelle Proofs The proofs in this paper and the proofs for the compound reader and writer monads were performed using the theorem prover Isabelle, and are available at [2], or from the author's home page.

Jones & Duponcheel, [3, §6.4], discuss the compound list monad – α *list* M is a monad provided that M is a *commutative* monad, and they give proofs of this. We also have Isabelle proofs of the *pext* axioms for the compound list monad.