$\mathcal{P}\mathcal{P}$ is not a monad?!

(where $\mathcal{P}$ is covariant)

Joshua Moerman

June 2017

Since the dawn of time, people have tried to use the double power-set functor to model alternating automata. Unfortunately, as Bartek Klin's notes show, many mistakes have been made in such attempts. In this note, I will give a concrete counterexample showing that a naive choice for η and μ do not define a monad $(\mathcal{P}\mathcal{P}, \eta, \mu)$. Additionally, it is shown abstractly how one of the monad laws fail. This may give insight as how to tackle the problem.

Let $\mathcal{P}$ be the *covariant*, finitary power-set functor from **Set** to **Set**. Typically, it is used to define non-deterministic automata (NFAs) coalgebraically: $X \to 2 \times \mathcal{P}(X)^A$. Successors are models as sets of states. Similarly, we may want to define alternating automata as coalgebras of the type $X \to 2 \times \mathcal{P}(\mathcal{P}(X))^A$. Here successors are modelled by a set of forks, which in turn are modelled as sets of states. A fork is accepting if all its branches are accepting (i.e. the inner power-set is considered conjunctively) and a set of forks accepts if one of the forks accepts (i.e. the outer power-set is considered disjunctively). Jurriaan Rot and Bartek Klin give trace semantics for this type of systems. Crucially to their approach is the following natural transformation (which is called the cross-cut operator on Wikipedia):

$$\begin{aligned}
&\chi : \mathcal{P}\mathcal{P} \Rightarrow \mathcal{P}\mathcal{P} \\
&\chi : \mathcal{S} \mapsto \{V \subseteq \bigcup \mathcal{S} \mid U \cap V \neq \emptyset \text{ for all } U \in \mathcal{S}\}.
\end{aligned}$$

Intuitively, this converts a CNF formula to a DNF formula. The authors warn that this map is not a distributive law (over the monad $\mathcal{P}$). Since mathematics is about wishful thinking, we can still try to define a monad structure with this map.

1 Attempt to make $\mathcal{P}\mathcal{P}$ into a monad

Let $(\mathcal{P}, \eta^1, \mu^1)$ be the covariant finitary power-set monad given on objects by $\mathcal{P}(X) = \{Y \subseteq X \text{ finite}\}$, on maps by $\mathcal{P}(f)(U) = \{f(x) \mid x \in U\}$, with the unit $\eta^1(x) = \{x\}$ and multiplication $\mu^1(\mathcal{U}) = \bigcup \mathcal{U}$.

We define a potential unit and multiplication map for $\mathcal{P}\mathcal{P}$:

$$\begin{aligned}
\eta^2 &: 1 \overset{\eta^1}{\Rightarrow} \mathcal{P} \overset{\eta^1}{\Rightarrow} \mathcal{P}^2 \\
\mu^2 &: \mathcal{P}^4 \overset{\mathcal{P}\chi}{\Longrightarrow} \mathcal{P}^4 \overset{\mu^1}{\Rightarrow} \mathcal{P}^3 \overset{\mathcal{P}\mu^1}{\Longrightarrow} \mathcal{P}^2.
\end{aligned}$$

(We note that $\eta^2 = P\eta^1 \circ \eta^1$ and $\mu^2 = \mu^1 \circ \mathcal{P}^2\mu^1 \circ \mathcal{P}\chi$ by naturality.) The multiplication is a natural thing to do: we start with a formula of the form $\bigvee \bigwedge \bigvee \bigwedge$, which is converted to $\bigvee \bigvee \bigwedge \bigwedge$ by $\mathcal{P}\chi$ and then we flatten twice to get $\bigvee \bigwedge$ as required.

This is the standard construction for a monad structure if χ is a distributive law. Now, χ is *not* a distributive law, but the construction might still give a monad (the requirement of being a distributive law is sufficient but not necessary). We will show that this is not the case.

1.1 The non-commuting triangle

In order for the standard proof to work, we need the following two triangles (and another square, not shown) to commute:

$$\begin{tikzcd}
\mathcal{P} \arrow[r, "\eta^1"] \arrow[dr, "\mathcal{P}\eta^1"] & \mathcal{P}\mathcal{P} \arrow[d, "\chi"] \\
& \mathcal{P}\mathcal{P}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{P} \arrow[r, "\mathcal{P}\eta^1"] \arrow[dr, "\eta^1"] & \mathcal{P}\mathcal{P} \arrow[d, "\chi"] \\
& \mathcal{P}\mathcal{P}
\end{tikzcd}$$

The first triangle commutes (note that $V \cap \{u\}$ for all u implies $U \subseteq V$):

$$\begin{aligned}
\chi(\mathcal{P}\eta^1(U)) &= \chi\{\{u\} \mid u \in U\} \\
&= \{V \subseteq U \mid V \cap \{u\} \neq \emptyset \text{ for all } u \in U\} \\
&= \{U\} \\
&= \eta^1(U).
\end{aligned}$$

The second does not commute:

$$\begin{aligned}
\chi(\eta^1(U)) &= \chi\{U\} \\
&= \{V \subseteq U \mid V \neq \emptyset\} \\
&\supsetneq \{\{u\} \mid u \in U\} \\
&= \mathcal{P}\eta^1(U).
\end{aligned}$$

Interestingly, $\chi(\eta^1(U))$ is the upward closure of $\mathcal{P}\eta^1(U)$. It does not commute, but it seems pretty harmless. Indeed, this was fine for the trace semantics in the paper by Jurriaan Rot and Bartek Klin. Let's see where the monad laws fail, now that we know that this particular triangle does not commute. The triangle is used to prove one of the unit laws for the monad. Here we spell out the standard proof using that triangle. We do so for general monads T and K (in order not to confuse the two $\mathcal{P}$s).

$$\begin{tikzcd}
TK \arrow[r, "\eta^T"] \arrow[dddrr, "\mathbf{id}"'] & TTK \arrow[r, "T\eta^K"] \arrow[dr, "TT\eta^K"] & TKTK \arrow[d, "T\chi"] \\
 & & TTKK \arrow[d, "TT\mu^K"] \\
 & & TTK \arrow[d, "\mu^T"] \\
 & & TK
\end{tikzcd}$$

proves

$$\begin{tikzcd}
TK \arrow[r, "\eta"] \arrow[dr, "\mathbf{id}"'] & TKTK \arrow[d, "\mu"] \\
 & TK
\end{tikzcd}$$

We see that the second triangle is used in the upper right corner (wrapped inside the T functor). We cannot yet conclude that the monad law does not hold. So far, we have only seen that the upper right corner of the big diagram does not commute. It may happen that the multiplication for the power-set monad does not care about upward-closures of sets, and that the big diagram still commutes. With a concrete calculation we will see that it does not commute.

Take $X = \{\clubsuit, \spadesuit\}$, and $\mathcal{S} = \{\{\clubsuit\}, \{\spadesuit\}\} \in \mathcal{P}\mathcal{P}X$. Then $\eta^2\mathcal{S} = \{\{\{\{\clubsuit\}, \{\spadesuit\}\}\}\}$. Applying χ on the inner set gives: $\chi\{\{\{\clubsuit\}, \{\spadesuit\}\}\} = \{\{\{\clubsuit\}\}, \{\{\spadesuit\}\}, \{\{\{\clubsuit\}, \{\spadesuit\}\}\}\}$. Then with both multiplications we get: $\mu^2\eta^2\mathcal{S} = \{\{\clubsuit\}, \{\spadesuit\}, \{\clubsuit, \spadesuit\}\} \neq \mathcal{S}$.

This concludes that $(\mathcal{P}\mathcal{P}, \eta^2, \mu^2)$ is not a monad. Note that this does not mean that there exists no monad structure on $\mathcal{P}\mathcal{P}$, it only means that this natural candidate does not work out.

2 Potential fixes

2.1 Lists

I have seen Filippo Bonchi and Fabio Zanasi use lists of lists instead of sets of sets to model alternating automata. Personally, I find that very unsatisfactory, as one then throws away associativity, commutativity and even idempotency. So I think it has little to do with non-determinism or alternating automata.

2.2 Distributive lattices

If we insist that the construction is made from the usual power-set, then I think we have to consider that monad as an adjunction:

$$\mathcal{P} : \mathbf{Set} \rightleftarrows \mathbf{JSL} : U,$$

where U is the forgetful functor and **JSL** is the category of join-semilattices (and monotone maps). Then we can ask ourselves: what is the appropriate power-set construction on **JSL**? We can define $\mathcal{P}'(X) = \{f : X \to \{\bot, \top\} \mid f \text{ monotone}\}$. This happens to be exactly the up-sets (or down-set, depending on perspective). And so these are upward closed (this resonates with the failure above). However, it is not immediately clear that there is a proper covariant functor with this definition.

I computed $\mathcal{P}'\mathcal{P}$ on small finite sets and got sets with the same sizes as free distributive lattices. This makes sense, because we are adding operations in two steps:

$$\mathbf{Set} \rightleftarrows \mathbf{JSL} \rightleftarrows \mathbf{DL},$$

where $\mathbf{DL}$ is for distributive lattices. Of course, one still needs to check that $\mathcal{P}'$ is indeed a functor (and, indeed, left adjoint to the forgetful functor). If this works out, it is particularly pretty, because it relates the definition of alternating automata using boolean formulas (i.e. the free distributive lattice) with a two-step power-set construction. It also means that one can determinise an alternating automata in either $\mathbf{JSL}$ (as non-deterministic automaton) or in $\mathbf{Set}$ (as deterministic automaton).

It may be necessary to consider $\mathcal{P}$ as functor $\mathbf{Set} \to \mathbf{MSL}$, using meet-semilattices instead. The category of meet-semilattices is equivalent to that of join-semilattices, the only difference is how we think of it. Only the details of $\mathcal{P}'$ will tell what the right interpretation is.

2.3 Still try $\mathcal{P}\mathcal{P}$

It is still possible, however, that some monad structure exists on $\mathcal{P}\mathcal{P}$. But it's not very easy to settle this issue. One approach might be by enumerating all natural transformations $\lambda : \mathcal{P}\mathcal{P} \Rightarrow \mathcal{P}\mathcal{P}$ and check whether it is a distributive law. But maybe a monad structure arises in a different way (not via the standard distributive law construction). Who knows?

To be continued...