Refutation of monads

© Copyright 2020 by Colin James III   All rights reserved.

Abstract: We evaluate the definition of monads as not tautologous. This refutes the definition of monad. What follows is that applying distributive laws to monads is denied, as is applying no-go theorems to monads. These results form a non tautologous fragment of the universal logic VŁ4.

We assume the method and apparatus of Meth8/VŁ4 with Tautology as the designated proof value, F as contradiction, N as truthity (non-contingency), and C as falsity (contingency). The 16-valued truth table is row-major and horizontal, or repeating fragments of 128-tables, sometimes with table counts, for more variables. (See ersatz-systems.com.)

LET ~ Not, ¬ ; + Or, V, U, U ; - Not Or; & And, ∧, ∩, , , ⊓, ⊗ ; \ Not And;
> Imply, greater than, →, ⇒ , ⊃, ⊢ ; < Not Imply, less than, ∈, ⊂, ⊄, #, ⇔, ⊑ ;
= Equivalent, ≡, :=, ⇔, ≈, ≅ ; @ Not Equivalent, ≠, ⊤ ;
% possibility, for one or some, ∃, ∃! , ∅, M ; # necessity, for every or all, ∀, □, L;
(z=z) T as tautology, T, ordinal 3; (z@z) F as contradiction, Ø, Null, ⊥ , zero;
(%z=>z) N as non-contingency, Δ, ordinal 1; (%z<#z) C as contingency, ⊤, ordinal 2;
(~( y < x) ( x ≤ y), ( x ⊆ y), ( x ⊆ y); (A=B) (A~B).

Note for clarity, we usually distribute quantifiers onto each designated variable.


Abstract. Monads are commonplace in computer science, and can be composed using Beck’s distributive laws. Unfortunately, finding distributive laws can be extremely difficult and error-prone. The literature contains some general principles for constructing distributive laws. However, until now there have been no such techniques for establishing when no distributive law exists. We present three families of theorems for showing when there can be no distributive law between two monads. The first widely generalizes a counterexample attributed to Plotkin. It covers all the previous known no-go results for specific pairs of monads, and includes many new results. The second and third families are entirely novel, encompassing various new practical situations. For example, they negatively resolve the open question of whether the list monad distributes over itself, reveal a previously unobserved error in the literature, and confirm a conjecture made by Beck himself in his first paper on distributive laws.

1. Introduction Monads have become a key tool in computer science. They are, amongst other things, used to provide semantics for computational effects such as state, exceptions, and IO. They are also used to structure functional programs , and even appear explicitly in the standard library of the Haskell programming language . Monads are a categorical concept. A monad on a category C consists of an endofunctor T and two natural transformations 1⇒T and μ:T*T⇒T satisfying the axioms described in Definition 2.1.

2. Preliminaries 2.1. Monads and distributive laws. We introduce monads, distributive laws, and various examples that will recur in later sections, primarily to fix notation. The material is standard, and may be skipped by the expert reader.
Definition 2.1 (Monad). For any category $\mathcal{C}$, a **monad** $(T, \eta, \mu)$ on $\mathcal{C}$ consists of an endofunctor $T: \mathcal{C} \to \mathcal{C}$, and natural transformations $\eta: 1 \Rightarrow T$ and $\mu: T \circ T \Rightarrow T$ referred to as the **unit** and **multiplication**, satisfying the following axioms:

\[
\begin{align*}
\mu \cdot T\eta &= \text{id} \quad \text{(unit 1)} \\
\mu \cdot \eta T &= \text{id} \quad \text{(unit 2)} \\
\mu \cdot T\mu &= \mu \cdot \mu T \quad \text{(associativity)}
\end{align*}
\]

LET $\ p, q, r, s: \ \mu, \eta, T, \text{id}.$

\[
(((p \& r) \& q) = s) \& (p \& (p \& r)) > (((s > r) > r) > q) \& (((r > r) > r) > p)) ;
\]

| TTTT | TTTT | TTTT | TTTT |

**Remark 2.1.1.2:** Eq. 2.1.1.2 as rendered is **not** tautologous. This refutes the definition of monad. What follows is that applying distributive laws to monads is denied, as is applying no-go theorems to monads.