

Monad Combinators, Non-Determinism and Probabilistic Choice

[Extended abstract]

Neil Ghani
Dept. of Math. and Comp. Sci.
University of Leicester
University Road
Leicester LE1 7RH, UK
ng13@mcs.le.ac.uk

Tarmo Uustalu
Inst. of Cybernetics
Tallinn University of Technology
Akadeemia tee 21
EE-12618 Tallinn, Estonia
tarmo@cs.ioc.ee

Abstract

We test Lüth and Ghani’s proposal [11, 12, 13] to use coproducts of monads as a basis for modularity by applying their ideas to Varacca’s work [20] on combining non-determinism and probabilistic choice. In particular, we discuss i) the coproduct of non-determinism and probabilistic choice; ii) how monad compositions based upon distributivity possess a universal property; and iii) Capretta’s [4] treatment of looping as an effect. We take advantage of the fact that all three effects are usefully modelled by ideal monads.

1 Introduction

It has long been a goal of research within the theoretical computer science community to provide a modular semantics for programming languages. One exciting possibility was Moggi’s proposal [16] to use monads to structure denotational semantics. Moggi pointed out that a number of computational features could be modelled by monads. Wadler [21] took Moggi’s ideas further by showing how one could actually program with monads. For example monads can be used to support imperative features within a purely functional language. Soon, however, it became clear that despite the undoubted value of monads from both the semantic and programming perspectives, composing monads would prove to be a significant challenge. Put briefly, monads just do not seem to compose in any general manner. Thus, a variety of different methods to combine monads were proposed, most notably in the use of *distributive laws* [3] and *monad transformers* [8]. While being useful in specific situations, these theories do not cover all situations and, furthermore, can sometimes seem rather ad-hoc.

Lüth originally proposed an alternative and considerably more systematic method of using the *coproduct* of monads as a monad combinator and applied this idea to term rewriting systems. This idea has subsequently been developed by Lüth, Ghani and Uustalu [11, 12, 13, 5] who gave simple constructions of the coproduct in important special cases as well as by Plotkin, Power and Hyland [6] who have suggested another unified account of modularity of effects proceeding from enriched Lawvere theories instead of monads and emphasizing the role of the operations inducing effects. In that account, the commutative combination and coproduct of Lawvere theories are the combination mechanisms.

Non-determinism and probabilistic choice are two commonplace effects that have turned out to

be notoriously difficult to combine and especially so in the presence of recursion. Indeed, the combination of probabilistic choice and recursion alone is problematic already. Probabilistic choice has been studied by Saheb-Djahromi [17], Jones and Plotkin [7], Jung and Tix [9, 18, 19] and others, and combination with non-determinism by Mislove et al. [14, 15] and Varacca [20]. This paper is inspired by Varacca’s work [20] where probabilistic choice is modelled by indexed valuations and combined with non-determinism using a distributive law, in sets and continuous domains. Our contribution is to

- Show that, in settings such as **Set**, non-determinism and probabilistic computation can be combined using coproducts of monads as opposed to Varacca’s solution of a distributive law. We discuss the concrete implications of this choice.
- Show that the possibility for looping can be added to the above combination without moving to a domains-based setting by following Capretta’s [4] idea of treating this as an effect.
- Show that, in general, all monad combinations based upon distributivity possess a universal property of being a coequalizer.

The remainder of this extended abstract is organized as follows. In Section 2, we review and compare three ways of combining monads—distributive laws, monad transformers and coproducts. In particular, we recall the concept of ideal monads and the construction of their coproduct and show that the monad induced by a distributive law is always a quotient of the coproduct. We also point out that all strong monads distribute over the left-action of a monoid. In Section 3, we show, how to capture non-determinism and probabilistic choice by ideal monads, find their coproduct and relate it to Varacca’s monad. In Section 4, we introduce the idea that looping is an effect. We introduce completely iterative monads and they can be used to model looping and to add looping to non-determinism and probabilistic choice. In Section 5, we list our conclusions and outline future work.

2 Coproducts of Monads

The best-known methods for combining monads are distributive laws and monad transformers. A *distributive law* [3] of a monad S over a monad R is a natural transformation $\lambda : SR \rightarrow RS$ behaving well wrt. the unit and multiplication of both S and R . That is, $\lambda \cdot \eta^S R = R\eta^S$, $\lambda \cdot S\eta^R = \eta^R S$, $\lambda \cdot S\mu^R = \mu^R S \cdot R\lambda \cdot \lambda R$ and $\lambda \cdot \mu^S R = R\mu^S \cdot \lambda S \cdot S\lambda$. The key result is that if there is a distributive law λ of a monad S over a monad R , then the functor composition RS is a monad. The unit is $\eta^{RS} = \eta^R \eta^S$ and the multiplication is $\mu^{RS} = \mu^R \mu^S \cdot R\lambda S$. In some situations, distributive laws are very helpful and exist canonically. e.g., if R is a strong monad and E is an internal monoid, then the strength is a distributive law of $\mathbf{Id} \times K_E$ over R , and hence $R(\mathbf{Id} \times K_E)$ is a monad. Also, for any monad R and object E , there is a canonical distributive law of $\mathbf{Id} + K_E$ over R , implying that $R(\mathbf{Id} + K_E)$ is a monad. In fact, $R(\mathbf{Id} + K_E)$ is also the coproduct of R and $\mathbf{Id} + K_E$.

A *monad transformer* is a pointed endofunctor on the category of monads on the base category. The interest in monad transformers originates from programming language semantics [8]. The idea is to capture an effect by a monad transformer for adding this particular effect uniformly to any monad. There are a number of them in existence but, as with distributive laws, they do not always exist, can seem ad-hoc in their definition, and do not (always) have associated universal properties to facilitate reasoning.

A considerably more widely applicable and systematic way of combining monads is the *coproduct* of two monads [11, 12, 13], by which we mean, of course, their coproduct in the category of monads. The coproduct of a monad with a fixed monad is a monad transformer. Therefore, coproducts can, if one wishes, be seen as a canonical way of building monad transformers.

Coproducts of monads exist under mild conditions [10] but cannot be computed pointwise. The general construction is rather involved [10], so a number of useful special cases have been considered. For example, the coproduct of two free monads F^* and G^* is $(F + G)^*$ and the coproduct of a free monad F^* and a monad S is $S(FS)^*$ [6]. Intuitively, the coproduct of monads R and S consists of all finite interleavings of R - and S -layers but potential problems arise because of layer collapse. Ideal monads, introduced by Adámek et al. [1], prevent this from occurring by separating trivial computations (values) from non-trivial computations and requiring that multiplication is non-trivializing. Ideal monads therefore provide a large class of monads for which the coproduct can be computed by simple fixed point formulae.

Definition 2.1 *An ideal monad is a monad (T, η, μ) together with a functor T_0 and natural transformations $\tau : T_0 \rightarrow T$, $\mu_0 : T_0T \rightarrow T$ such that T with η , τ is a coproduct of the functors Id and T_0 and $\tau \cdot \mu_0 = \mu \cdot \tau T$.*

In fact, an ideal monad is fully specified by its data (T_0, μ_0) . Given some (T_0, μ_0) , by defining $T =_{\text{df}} \text{Id} + T_0$, $\eta =_{\text{df}} \text{inl}_{\text{Id}, T_0}$, $\tau =_{\text{df}} \text{inr}_{\text{Id}, T_0}$ and $\mu =_{\text{df}} [T, \tau \cdot \mu_0]$, one obtains an ideal monad whenever $T_0 = \mu_0 \cdot T_0\eta$ and $\mu_0 \cdot \mu_0T = \mu_0 \cdot T_0\mu$. Examples of ideal monads include free monads, free completely iterative monads [1], non-empty lists and non-empty finite multisets. As shown in [5], the coproduct of two ideal monads $R \cong \text{Id} + R_0$ and $S \cong \text{Id} + S_0$ is given by finite alternations of R_0 and S_0 (i.e., non-trivial R - and S -layers) as follows:

Theorem 2.2 *Let $R \cong \text{Id} + R_0$ and $S \cong \text{Id} + S_0$ be two ideal monads. Then, provided that the initial algebras*

$$(T_1(X), T_2(X)) =_{\text{df}} \mu(Y_1, Y_2).(R_0(X + Y_2), S_0(X + Y_1))$$

exist, the coproduct of R and S exists, is ideal and its underlying functor is $\text{Id} + (T_1 + T_2)$.

Alternatively, avoiding initial algebras in the product category, one can state that $(R \oplus S)(X) = S(\mu Y.(X + R_0(X + S_0(Y))))$.

We finish this section by proving a nice result that relates distributivity to coproducts and shows that monads combined using distributivity have a universal property. Formally, the monad RS induced by a distributive law λ of S over R is a coequalizer of the coproduct $R \oplus S$. In the following, recall that the free monad F^* on a functor F satisfies the property that, for every monad T , there is a bijection between natural transformations $F \rightarrow T$ and monad morphisms $F^* \rightarrow T$.

Theorem 2.3 *Let R and S be monads such that their coproduct $R \oplus S$ and the free monad $(SR)^*$ exist and let λ be a distributive law of S over R . The monad RS together with the monad map $[R\eta^S, \eta^R S] : R \oplus S \rightarrow RS$ is a coequalizer in the category of monads of the monad maps $(SR)^* \rightarrow R \oplus S$ induced by the natural transformations $\mu^{R \oplus S} \cdot \text{inl}^{R, S} \cdot \text{inr}^{R, S} \cdot \lambda$ and $\mu^{R \oplus S} \cdot \text{inr}^{R, S} \cdot \text{inl}^{R, S} : SR \rightarrow R \oplus S$.*

Intuitively, computations in the monad RS are obtained from those in $R \oplus S$ by permuting the S -layers in them down past the R -layers.

These results, taken with those existing in the literature, mean that we are close to achieving a rich and expressive set of combinators for monads with universal properties to reason with. Their usefulness in practice requires their application in a number of case studies and now we turn to one such.

3 Combining Non-Determinism and Probabilistic Choice

In semantic universes like **Set** (where there are true coproducts, lists etc.), non-determinism and probabilistic choice are very easily modelled by monads. Non-determinism can be modelled by

finite sets, multisets or lists of values, depending on whether the multiplicity and order of the alternative outcomes from a computation are considered to be semantically significant or not. It can also be modelled by the corresponding forms of well-founded trees, if the individual choices made in a computation are relevant, but we will not discuss this option. Denote the sets of finite sets and multisets over a set X by $P(X)$, $M(X)$ respectively, with possible subscripts for cardinality restrictions. For probabilistic choice, one can use weighted versions of the above options, so denote by $WP(X)$ and $WM(X)$ the sets of weighted sets and finite weighted multisets defined by $WP(X) = P(X \times \mathbb{R}^+)$ and $WM(X) = M(X \times \mathbb{R}^+)$. That WP and WM are monads follows immediately from \mathbb{R}^+ being a monoid and the fact that, for any strong monad R , there is a distributive law of $\text{Id} \times K_{\mathbb{R}^+}$ over R .

While the monads P , M and $P_{\geq 1}$ are not ideal, the monad $M_{\geq 1} \cong \text{Id} + M_{\geq 2}$ of non-empty finite multisets is ideal. Similar results hold for $WP(X)$ and $WM(X)$. Thus we can apply Theorem 2.2 to the ideal monads $M_{\geq 1}$ and $WM_{\geq 1}$ and take their coproduct as a candidate for the combination of non-determinism and probabilistic choice. The theorem tells us that $(M_{\geq 1} \oplus WM_{\geq 1})(X)$ is the set of well-founded trees with leafs labelled by elements of X , branching nodes alternately of two types n and p , every branching node having at least two children, arcs from every n -node unlabelled, arcs from every p -node labelled with positive real weights.

Varacca [20] observed that, although there is no distributive law of WP over $P_{\geq 1}$ or of $P_{\geq 1}$ over WP , there exists a distributive law λ of WM over $P_{\geq 1}$. This is given by

$$\lambda(\langle (\alpha_1, p_1), \dots, (\alpha_n, p_n) \rangle) = \{ \langle (a_1, p_1), \dots, (a_n, p_n) \rangle \mid a_1 \in \alpha_1, \dots, a_n \in \alpha_n \}$$

Consequently, his combination of non-determinism and probabilistic choice for **Set** was the monad structure on the functor $P_{\geq 1} WM$. Comparing Varacca's solution of $P_{\geq 1} WM$ with ours of $M_{\geq 1} \oplus WM_{\geq 1}$ we see that there are some minor differences, e.g., ours uses multisets and the nonempty restriction for both monads. Despite this, a number of interesting observations can be made.

- To obtain a distributivity law Varacca replaced WP by its multiset version WM and P by the non-empty version $P_{\geq 1}$. In order to use our construction for the coproduct of ideal monads, we had to apply similar measures.
- Varacca's monad is essentially a quotient of ours obtained by replacing M by P and using the fact that distributivity is a quotient of the coproduct.
- In a programming language semantics example, Varacca employed an intermediate data structure which is almost the coproduct, viz. trees as in the coproduct but with branching nodes allowed to have one child (corresponding to trivial layers).
- The essential difference between the coproduct and Varacca's quotient seems to be the law

$$a \mid_p (b \mid c) = (a \mid_p b) \mid (a \mid_p c)$$

which corresponds to the rather strong requirement of scheduling non-deterministic choice deterministically.

In the full paper we shall be more precise on this comparison. However, it is clear already from the considerations here that the calculus of monadic combinators offers both new possible semantics, e.g., the coproduct, and sheds further light on those, such as Varacca's, already existing.

4 Completely Iterative Monads and Looping

Language features such as looping (iteration) are traditionally handled in categories such as **CPO** where every object comes with a fixed-point operation. But there is also different approach: we can start with a simpler category such as **Set** as the basic semantic universe, consider looping as

an effect, capture this effect with a monad and then use the Kleisli category of the monad as the improved semantic universe.

The appropriate monad for purely functional settings is the ideal monad with the underlying functor $It \cong \text{Id} + It$ given by $It(X) =_{\text{df}} \nu Y.X + Y$. This is, in general, different from $\text{Id} + K_1$ and $\text{Id} \times K_{\mathbb{N}} + K_1$, which correspond to decidable partiality and looping with decidable looping time, respectively. The elements of $It(X)$ are non-wellfounded terms of the signature with one unary operation. Intuitively, one should think of them as possibly infinite “delays” of elements of X . The unit of the monad is given by the zero delay and the multiplication by addition of delays. Adámek et al. [1] have characterized this monad as the free completely iterative monad on the identity functor Id .

Definition 4.1 *An ideal monad $T \cong \text{Id} + T_0$ is completely iterative, if, for every map $f : X \rightarrow Y + T_0(X + Y)$ (guarded equation), there exists a unique map $g : X \rightarrow T(Y)$ such that $g = [\eta_Y, \tau_Y \cdot \mu_{0Y} \cdot T_0([g, \eta_Y])] \cdot f$ (solution).*

Theorem 4.2 *Let F be an endofunctor. If the final coalgebras $F^\infty(X) =_{\text{df}} \nu Y.X + F(Y)$ exist, then $F^\infty \cong \text{Id} + FF^\infty$ is the underlying functor of the free completely iterative monad on F .*

From the definition and theorem, it is immediate that the monad It is very appropriate for modelling looping: for any map $f : X \rightarrow It(X + Y)$, there exists a unique map $g : X \rightarrow It(Y)$ such that $g = \tau_Y \cdot \mu_{0Y} \cdot It_0([g, \eta_Y]) \cdot f$.

Looping can also be added to other effects. This is again reasonably easy for an ideal monad. Given an ideal monad $R \cong \text{Id} + R_0$, the functor $R^\circ \cong \text{Id} + (R^\circ + R_0(\text{Id} + R^\circ))$ given by $R^\circ(X) =_{\text{df}} \nu Y.R(X + Y)$ carries a completely iterative monad structure which is clearly suitable for modelling looping as, for any map $f : X \rightarrow R^\circ(X + Y)$, there exists a unique map $g : X \rightarrow R^\circ(Y)$ such that $g = \tau_Y \cdot \mu_{0Y} \cdot R_0^\circ([g, \eta_Y]) \cdot \text{inl} \cdot f$. Moreover, there is an obvious ideal monad map $\iota : R \rightarrow R^\circ$. Hence, $(\cdot)^\circ$ is an ideal monad transformer.

The set $(M_{\geq 1} \oplus WM_{\geq 1})^\circ(X)$ consists of non-wellfounded leaf trees with leaves labelled by elements of X and branching nodes of three types n , p and d such that every n -node has at least two children with the arcs to them unlabelled, every p -node has at least two children with the arcs to them labelled with positive real weights, every d -node has exactly one child with the arc to it unlabelled and such that no path contains two consecutive n -nodes, two consecutive p -nodes or an infinite sequence of n - and p -nodes consecutively.

5 Conclusions and Future Work

We have shown that the coproduct of two ideal monads provides a meaningful combination of non-determinism and probabilistic choice which compares favorably with Varacca’s solution for **Set**. In particular, our solution does not choose a scheduler. We have also shown how completely iterative monads can be used to model looping both in a purely functional setting and in the presence of non-determinism and probabilistic choice. This modelling relies on the idea that looping is an effect. The relation to domain-theoretic solutions is a topic for future work.

Acknowledgements The authors were supported by a Royal Society ESEP joint research project. The second author was also supported by the Estonian Science Foundation under grant No. 5567.

References

- [1] P. Aczel, J. Adámek, S. Milius, and J. Velebil, Infinite trees and completely iterative theories: a coalgebraic view, *Theor. Comput. Sci.* **300**(1–3) (2003) 1–45.
- [2] M. Barr and C. Wells, *Toposes, Triples and Theories*, *Grundlehren der mathematischen Wissenschaften* **275**, Springer-Verlag, Berlin (1985).
- [3] J. Beck, Distributive laws, in *Seminar on Triples and Categorical Homology Theory (ETH, 1966/67)*, edited by B. Eckmann, *Lect. Notes in Math.* **80**, Springer-Verlag, Berlin (1969) 119–140.
- [4] V. Capretta, General recursion via coinductive types, draft manuscript (2004).
- [5] N. Ghani and T. Uustalu, Coproducts of ideal monads, *Theor. Inform. and Appl.* (to appear).
- [6] M. Hyland, G. Plotkin, and J. Power, Combining computational effects: commutativity and sum, in *Proc. of IFIP 17th World Computer Congress, TC1 Stream / 2nd IFIP Int. Conf. on Theoretical Computer Science, TCS 2002 (Montreal, Aug. 2002)*, edited by A. Baeza-Yates, U. Montanari, and N. Santoro, *IFIP Conf. Proc.* **223**, Kluwer Academic Publishers, Dordrecht (2002) 474–484.
- [7] C. Jones and G. D. Plotkin, A probabilistic powerdomain of evaluations, in *Proc. of 4th Ann. IEEE Symp. Logic in Computer Science, LICS'89 (Pacific Grove, CA, June 1989)*, IEEE CS Press, Washington, DC (1989) 186–195.
- [8] M. Jones and L. Duponcheel, Composing monads, Techn. report RR-1004, Dept. of Comput. Sci, Yale Univ. (1993).
- [9] A. Jung and R. Tix, The troublesome probabilistic powerdomain, in *Proc. of 3rd Wksh. on Computation and Approximation, Comprox III (Birmingham, Sept. 1997)*, edited by A. Edalat et al., *Electr. Notes in Theor. Comput. Sci.* **13**, Elsevier, Amsterdam (1998).
- [10] G. M. Kelly, A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves and so on, *Bull. of Australian Math. Soc.* **22** (1980) 1–83.
- [11] C. Lüth and N. Ghani, Monads and modular term rewriting, in *Proc. of 7th Int. Conf. on Category Theory in Computer Science, CTCS'97 (Santa Margherita Ligure, Sept. 1997)*, edited by E. Moggi and G. Rosolini, *Lect. Notes in Comput. Sci.* **1290**, Springer-Verlag, Berlin (1997) 69–86.
- [12] C. Lüth and N. Ghani, Monads and modularity, in *Proc. of 4th Int. Wksh. on Frontiers of Combining Systems, FroCoS 2002 (Santa Margherita Ligure, Apr. 2002)*, edited by A. Armando, *Lect. Notes in Artif. Intell.* **2309**, Springer-Verlag, Berlin (2002) 18–32.
- [13] C. Lüth and N. Ghani, Composing monads using coproducts, In *Proc. of 7th ACM SIGPLAN Int. Conf. on Functional Programming, ICFP'02 (Pittsburgh, PA, Oct. 2002)*, ACM Press, New York (2002) 133–144.
- [14] M. Mislove, Nondeterminism and probabilistic choice: obeying the laws, in *Proc. 11 Int. Conf. on Concurrency Theory, CONCUR 2000 (University Park, PA, Aug. 2000)*, edited by C. Palamidessi, *Lecture Notes in Comput. Sci.* **1877**, Springer-Verlag, Berlin (2000) 350–364.
- [15] M. Mislove, J. Ouaknine, J. Worrell, Axioms for probability and nondeterminism, *Electr. Notes in Theor. Comput. Sci.* **96** (2004) 7–28.
- [16] E. Moggi, Notions of computation and monads, *Inform. and Computat.* **93**(1) (1991) 55–92.
- [17] N. Saheb-Djahromi, CPO's of measures for nondeterminism, *Theor. Comput. Sci.* **12** (1980) 19–37.
- [18] R. Tix, Some results on Hahn-Banach-type theorems for continuous D -cones, *Theor. Comput. Sci.* **264**(2) (2001) 205–218.
- [19] R. Tix, Convex power constructions for continuous D -cones, in *Proc. of Wksh. on Domains IV (Rolandseck, Oct. 1998)*, edited by D. Spreen, *Electr. Notes in Theor. Comput. Sci.* **35**, Elsevier, Amsterdam (2000).
- [20] D. Varacca, The powerdomain of indexed valuations, in *Proc. of 17th Ann. IEEE Symp. on Logic in Computer Science, LICS'02 (Copenhagen, July 2002)*, IEEE CS Press, Los Alamitos, CA (2002) 299–308.
- [21] P. Wadler, The essence of functional programming, In *Conf. Record of 19th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'92 (Albuquerque, NM, Jan. 1992)*, ACM Press, New York (1992) 1–14.