Concurrent Monads for Shared State

Exequiel Rivas
Tallinn University of Technology, Estonia
exequiel.rivas@taltech.ee

Tarmo Uustalu
Reykjavik University, Iceland
Tallinn University of Technology, Estonia
tarmo@ru.is

In the context of programming with effects, sequential composition takes a special role as the primary control structure for combining computations. In this paper, we advocate the idea that parallel composition should also be treated as a control structure, on the same footing as sequential composition. We promote the concept of concurrent monad, which axiomatizes both sequential and parallel composition, and illustrate the approach by describing two concurrent monads for shared state: one of resumptions, the other of finite multisets of traces.

1 Introduction

This paper is about concurrency in effectful computation and programming.

We argue that parallel composition as a concurrency primitive is not an effect but a high-level control structure to be treated on the same footing as sequential composition. Monads axiomatize sequential composition. If we want to talk about not only sequential composition but also parallel composition, we need a different mathematical structure that axiomatizes both as well as, crucially, their interaction. Such an axiomatization is provided by a specialization of monads, called concurrent monads, first proposed and worked out by Rivas and Jaskelioff [4] within a certain approach to ordered structure for monads and elaborated by Paquet and Saville [3] in a bicategorical setting. Our aim in this paper is to promote concurrent monads and demonstrate their expressive power.

We develop the theory of concurrent monads in the setting of **Poset**-enrichment. We center our development around the fact that the Kleisli construction for a concurrent monad gives what we call a concurrent category. Concurrent categories can be seen as a typed generalization of concurrent monoids from concurrent Kleene algebra of Hoare et al. [2], going back to Gischer [1]. We show a number of examples, prominently among them two concurrent monads for (preemptive) interleaving shared state concurrency, one based on resumptions, the other on finite multisets of traces.

To appeal to programmers, we have implemented our development in Haskell. But the implementation is necessarily only an approximation of the theory since Haskell is more like **Set** than **Poset**.

The Haskell code, as well as the draft full paper on this material submitted to another conference, are available at `https://cs.ioc.ee/~tarmo/concurmonads/`.

2 A glimpse into details

We work in ordered (= **Poset**-enriched) category theory, i.e., with ordered categories, ordered functors and ordered natural transformations (the latter are the same as ordinary natural transformations). A category $\mathbb{C}$ being ordered means that each of its homsets is an ordered set and composition is monotone. A functor $F : \mathbb{C} \to \mathbb{C}$ is ordered if $f \leq g$ implies $Ff \leq Fg$.

Submitted to:
MSFP 2024

The archetypical ordered category is the category **Poset** of ordered sets and monotone functions, with the order on each homset of given pointwise. **Poset** is the base ordered category in all our examples.

A concurrent monad on an ordered monoidal category $(\mathbb{C}, \mathsf{I}, \otimes)$ is given by an ordered functor T that carries both a monad structure (η, μ) and a lax monoidal functor structure (ψ^0, ψ) satisfying four interchange inequations: $\eta_\mathsf{I} \leq \psi^0$, $\psi^0 \,; T\psi^0 \,; \mu_\mathsf{I} \leq \psi^0$, $\eta_{X\otimes Y} \leq \eta_X \otimes \eta_Y \,; \psi_{X,Y}$, $\psi_{TX,TY} \,; T\psi_{X,Y} \,; \mu_{X\otimes Y} \leq \mu_X \otimes \mu_Y \,; \psi_{X,Y}$. The 1st inequation is implied by the 4th. A concurrent monad is normal if the 1st inequation holds as an equality, i.e., $\eta_1 = \psi^0$.

If the interchange inequations hold as equalities, then a concurrent monad is just an ordered lax monoidal monad (commutative monad in Kock's terminology). If the order structure on $\mathbb{C}$ is trivial (all homsets are discretely ordered), then it is a (trivially ordered) commutative monad. We see that all expressive power of concurrent monads is thanks to the ordered setting, making it possible to stipulate interchange inequationally.

The Kleisli construction of a concurrent monad is what we call a concurrent category. It is an ordered monoidal-like category $(\mathbb{K}, \mathsf{I}^\mathsf{K}, \otimes^\mathsf{K})$, coming together with an identity-on-objects ordered monoidal-like functor from $(\mathbb{C}, \mathsf{I}, \otimes)$, its base. A key aspect of this monoidal-likeness is that I^K and $\otimes^\mathsf{K}$ are lax functors so that the interchange between $\mathsf{id}^\mathsf{K}_X \in \mathbb{C}(X, TX)$, $(;^\mathsf{K}) : \mathbb{C}(X, TY) \times \mathbb{C}(Y, TZ) \to \mathbb{C}(X, TZ)$ and $\mathsf{I}^\mathsf{K} \in \mathbb{C}(\mathsf{I}, T\mathsf{I})$, $\otimes^\mathsf{K} : \mathbb{C}(X, TY) \times \mathbb{C}(U, TV) \to \mathbb{C}(X \otimes U, Y \otimes V)$ (i.e., between sequential and parallel composition of effectful functions) is inequational, like in a concurrent monoid, but typed.

Similarly to how a monad on a category $\mathbb{C}$ is the same as a monoid wrt. the composition monoidal structure on the category $[\mathbb{C}, \mathbb{C}]$ of endofunctors, a concurrent monad on an ordered category $\mathbb{C}$ turns out to be a concurrent monoid wrt. the ordered composition and Day convolution duoidal structure on the ordered category $[\mathbb{C}, \mathbb{C}]$.

The familiar state monad, whose underlying functor is defined on objects by $TX = S \Rightarrow S \times X$, is ordered and extends to a non-normal concurrent monad if the given ordered set S of states is a lower semilattice. This concurrent monad implements a very simplistic type of shared state concurrency. If two effectful functions are composed in parallel, both are run from the given initial state independently and the final states are reconciled with the meet operation.

Concurrent monads are expressive enough to model customary (preemptive) interleaving shared state concurrency. We exhibit two concurrent monads that accomplish this. The first is based on resumptions and is very operational/intensional, the second on finite multisets (bags) of traces, much more denotational/extensional. Differently from the state concurrent monad, where computations consist of one big step, computations here are made of small steps, making interleaving possible.

The resumptions concurrent monad has the object mapping of the underlying functor defined by $TX = \mu Z. X + (S \Rightarrow \mathsf{List}(S \times Z))$ where the order on TX uses that $\mathsf{List}Y$ is ordered by list inclusion[1] combined with the order on Y. A resumption is either terminated or first grabs the state from the environment, then makes an internal choice nondeterministically and yields a state and a new resumption.

For the bags of traces concurrent monad, the object mapping of the underlying functor is $TX = \mathscr{M}_\mathrm{f}(\mu Z. X + S \times S \times Z)$. Here the order on TX uses that $\mathscr{M}_\mathrm{f} Y$ is ordered by multiset inclusion and the order on Y. Compared to resumptions, all nondeterminism (both from grabbing and own choice) is at the beginning of a computation, it is the choice of one trace from the collection.

These two concurrent monads are normal. There is a concurrent monad morphism between the resumptions and bags-of-traces concurrent monads.

[1] $xs \leq ys$ if xs is obtainable from ys by dropping elements at some positions.

Acknowledgements E.R. was supported by the Estonian Research Council grant no. PSG749. Both authors were supported by the Icelandic Research Fund grant no. 228684-052.

References

[1] Jay L. Gischer (1988): *The Equational Theory of Pomsets*. *Theor. Comput. Sci.* 61(2–3), pp. 199–224, doi:10.1016/0304-3975(88)90124-7.

[2] Tony Hoare, Bernhard Möller, Georg Struth & Ian Wehrman (2011): *Concurrent Kleene Algebra and Its Foundations*. *J. Log. Algebraic Program.* 80(6), doi:10.1016/j.jlap.2011.04.005.

[3] Hugo Paquet & Philip Saville (2024): *Effectful Semantics in Bicategories: Strong, Commutative, and Concurrent Pseudomonads*. In: *Proc. of 39th ACM/IEEE Symp. on Logic in Computer Science, LICS '24 (Tallinn, July 2024)*, ACM, pp. 61:1–61:15, doi:10.1145/3661814.3662130.

[4] Exequiel Rivas & Mauro Jaskelioff (2019): *Monads with Merging*. HAL eprint hal-02150199. Available at `https://inria.hal.science/hal-02150199`.