Concurrent monads, clones, pomsets and their relationship

SERGEY GONCHAROV and CRISTINA MATACHE, University of Birmingham, UK
HUGO PAQUET, Inria and École Normale Supérieure – PSL, France

We present work in progress about concurrent monads, an extension of monads designed to model parallel composition of programs. We take an algebraic perspective on concurrent monads via abstract clones, introducing a notion of concurrent clone and showing its correspondence to concurrent monads. We show preliminary results characterizing free concurrent clones using labelled partial orders (pomsets), and conclude with a discussion of promising research directions.

1 Background

Strong monads and algebraic effects [11, 15] have been successful both in semantics of effectful programs and for structured programming with effects, for example through Haskell's monads or via effect handlers [16]. Recently, there has been much interest in extending these models and techniques to concurrent computation: effect handlers underlie powerful concurrency libraries e.g. [14, 19], while algebraic theories have been proposed for shared-state concurrency [1, 5, 6] and process calculi [20, 21].

Another viewpoint, proposed by Rivas and Jaskelioff [17], is that of *concurrent monads*: a monad with extra structure $\| : TX \times TY \to T(X \times Y)$ for parallel composition, which performs two computations in parallel and collects their return values. Crucially, each TX carries a partial order and $\|$ must satisfy a version of the weak interchange law from concurrent Kleene algebra [8]: $(a\|b);(c\|d) \leq (a;c)\|(b;d)$. Concurrent monads have since been studied in ordered monoidal categories with applications to shared state [18], and in 2-dimensional categories [4, 13] with applications to concurrent game semantics and continuation semantics. In this work in progress, we study concurrent monads through the perspective of algebraic theories.

2 Concurrent monads and clones

We work with finitary monads on the category of sets, which have a well-known correspondence to algebraic theories [9, 10]. In practice, we think of such theories T as *abstract clones*, defined like Kleisli triples except TX appears only when X is a finite set $\{1, \ldots, n\}$. (We often denote the set $\{1, \ldots, n\}$ as n.) Then $T(n)$ is the set of equality classes of terms with n free variables, or equivalently the free model of the theory T on the set $\{x_1, \ldots, x_n\}$.

By analogy with clones, we propose a notion of *concurrent clone* that corresponds to the concurrent monads of Rivas and Jaskelioff [17] (whose definition we omit for brevity).

Definition 2.1. A *concurrent clone* is an ordinary clone $(T, \eta, \gg\!=)$ with a partial order $\leq_{Tn}$ on each set $T(n)$ and with a family of functions $*_{m,n}$, intuitively running computations in parallel:

$$*_{m,n} : T(m) \times T(n) \to T(m \times n)$$

We require $*$ to be associative, commutative, and have $\eta_1()$ as unit, and $*$ to be natural in both m and n with respect to set functions. Moreover, $\gg\!=$ and $*$ are required to be monotone with respect to the order on each $T(n)$ and to satisfy the weak interchange law:

$$(a * b) \gg\!= \lambda(i,j).f(i) * g(j) \;\leq_{T(m' \times n')}\; (a \gg\!= f) * (b \gg\!= g).$$

for $a \in T(m)$, $b \in T(n)$, $f : m \to T(m')$, $g : n \to T(n')$.

Example 2.2 (Corresponding to the concurrent monad of [17, Example 4.18]). Let Σ be an alphabet of symbols. Let $T_\Sigma(n)$ contain finite sets of traces consisting of elements of Σ and ending in a variable from n: $T_\Sigma(n) \coloneqq \mathcal{P}_f(\Sigma^* \times n)$. The order on $T_\Sigma(n)$ is subset inclusion and the parallel composition $*$ is given by interleaving of traces. This structure makes T_Σ into a concurrent clone.

Theorem 2.3. *To give a concurrent monad in the sense of Rivas and Jaskelioff [17, Definition 4.13] is equivalent to giving a concurrent clone in the sense of Definition 2.1.*

The proof relies on the existing correspondence between ordinary clones and finitary monads. A clone induces a monad via left-Kan extension along an inclusion functor, and this same construction can be extended to the concurrent structure.

3 Free concurrent clones

In the usual setting, free monads correspond to theories with no equations: the free monad at X is the set of trees with nodes labelled by operations and leaves by X. Manipulating these trees forms the basis of programming with algebraic effects [16]. We seek an analogous characterization of free concurrent clones, using series-parallel pomsets, as defined for example by Gischer [7].

Definition 3.1 ([7]). Fix an alphabet Σ. A pomset $(P, \leq, \ell)$ is a partially ordered set with a labelling function $\ell : P \to \Sigma$. We consider pomsets up to label-preserving order isomorphism. Parallel composition, $P_1 | P_2$, takes the disjoint union of the elements, orders and labellings; series composition, $P_1 \cdot P_2$, does the same but additionally places every element of P_1 below every element of P_2. A *series-parallel pomset* is either empty or a pomset obtained by series and parallel composition starting from singleton pomsets. The *subsumption order* $\preceq$ relates two pomsets P_1 and P_2 with the same elements and labelling as follows: $P_1 \preceq P_2$ iff the order of P_2 is contained in that of P_1.

Intuitively, a pomset is a program performing observable actions (its labels from Σ), with the order encoding their observable dependencies. In the pomset $\{\rho < \sigma_1, \rho < \sigma_2\}$, action ρ happens first, after which σ_1 and σ_2 may occur concurrently or in either order.

Example 3.2. Let $\mathbf{P}$ and $\mathbf{SP}$ be the sets of pomsets and of series-parallel pomsets respectively, for a fixed alphabet Σ. We can construct concurrent clones $T_{\mathbf{P}}$ and $T_{\mathbf{SP}}$ as $T_{\mathbf{P}}(n) := n \times \mathbf{P}$ and $T_{\mathbf{SP}}(n) := n \times \mathbf{SP}$. In both cases, the order is given by subsumption in the second argument: $(i, P_1) \preceq (i, P_2)$ iff $P_1 \preceq P_2$. The bind, $\gg\!=$, and parallel composition, $*_{m,n}$, extend the series and parallel composition of pomsets; the unit is the empty pomset $\emptyset$. The weak interchange law holds by definition of subsumption.

In fact, $(\mathbf{P}, \cdot, |, \emptyset)$ and $(\mathbf{SP}, \cdot, |, \emptyset)$ are concurrent monoids [8], also known as ordered bimonoids satisfying weak interchange [2]. The construction we used in Example 3.2 can be applied to any concurrent monoid to obtain a concurrent clone. The next theorem relies on $(\mathbf{SP}, \cdot, |, \emptyset)$ being the free concurrent monoid on the set of generators Σ [2, Appendix A1].

Theorem 3.3. *Consider the fixed alphabet Σ as a signature of unary operations (i.e. a functor from finite sets to sets). Then $T_{\mathbf{SP}}$ from Example 3.2 is the free concurrent clone on the signature Σ.*

4 Prospects

Our next step is to characterize free concurrent clones, as in Theorem 3.3, for signatures Σ with multi-ary operations—for instance an operation that reads a value stored in memory and branches depending on this value. Viewing the operations of Σ as effects, this is a step towards modelling concurrent programming with algebraic effects.

To achieve this characterization of free clones, we have two promising directions. One is to replace pomsets with event structures [12], using a conflict relation to model the distinct branches of a computation tree. We will need to define a subsumption order on event structures, possibly in terms of their maps [22]. Another direction is to add a non-deterministic choice operation to concurrent clones, with unit and suitable equations. We already characterize such free clones on multi-ary signatures as containing downward-closed sets of series-parallel pomsets, and expect to recover the free clone without non-determinism as a sub-clone, using techniques similar to [3].

References

[1] Martín Abadi and Gordon D. Plotkin. 2010. A Model of Cooperative Threads. *Log. Methods Comput. Sci.* 6, 4 (2010). doi:10.2168/LMCS-6(4:2)2010

[2] Stephen L. Bloom and Zoltán Ésik. 1996. Free Shuffle Algebras in Language Varieties. *Theor. Comput. Sci.* 163, 1&2 (1996), 55–98. doi:10.1016/0304-3975(95)00230-8

[3] Nathan J. Bowler, Paul Blain Levy, and Gordon D. Plotkin. 2018. Initial Algebras and Final Coalgebras Consisting of Nondeterministic Finite Trace Strategies. In *MFPS 2018*. doi:10.1016/J.ENTCS.2018.11.003

[4] Flavien Breuvart and Hugo Paquet. 2025. Categorical Continuation Semantics for Concurrency. In *FSCD 2025*. doi:10.4230/LIPICS.FSCD.2025.10

[5] Yotam Dvir, Ohad Kammar, and Ori Lahav. 2022. An Algebraic Theory for Shared-State Concurrency. In *APLAS 2022*. doi:10.1007/978-3-031-21037-2_1

[6] Yotam Dvir, Ohad Kammar, Ori Lahav, and Gordon D. Plotkin. 2025. Two-sorted algebraic decompositions of Brookes's shared-state denotational semantics. (2025). doi:10.1007/978-3-031-90897-2_18

[7] Jay L. Gischer. 1988. The equational theory of pomsets. *Theoretical Computer Science* 61, 2 (1988), 199–224. doi:10.1016/0304-3975(88)90124-7

[8] Tony Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. 2011. Concurrent Kleene Algebra and its Foundations. *J. Log. Algebraic Methods Program.* 80, 6 (2011), 266–296. doi:10.1016/J.JLAP.2011.04.005

[9] F. William Lawvere. 1963. Functorial Semantics of Algebraic Theories. *Proceedings of the National Academy of Sciences* 50, 5 (1963), 869–872.

[10] F. E. J. Linton. 1966. Some Aspects of Equational Categories. In *Proceedings of the Conference on Categorical Algebra*. 84–94.

[11] Eugenio Moggi. 1991. Notions of Computation and Monads. *Information and Computation* 93, 1 (1991), 55 – 92.

[12] Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. 1981. Petri Nets, Event Structures and domains, Part I. *Theoret. Comput. Sci.* 13 (1981). doi:10.1016/0304-3975(81)90112-2

[13] Hugo Paquet and Philip Saville. 2024. Effectful semantics in bicategories: strong, commutative, and concurrent pseudomonads. In *LICS 2024*. doi:10.1145/3661814.3662130

[14] Luna Phipps-Costin, Andreas Rossberg, Arjun Guha, Daan Leijen, Daniel Hillerström, K. C. Sivaramakrishnan, Matija Pretnar, and Sam Lindley. 2023. Continuing WebAssembly with Effect Handlers. *Proc. ACM Program. Lang.* 7, OOPSLA2 (2023), 460–485. doi:10.1145/3622814

[15] Gordon Plotkin and John Power. 2002. Notions of Computation Determine Monads. In *FOSSACS 2002*.

[16] Gordon D. Plotkin and Matija Pretnar. 2013. Handling Algebraic Effects. *Log. Methods Comput. Sci.* 9, 4 (2013). doi:10.2168/LMCS-9(4:23)2013

[17] Exequiel Rivas and Mauro Jaskelioff. 2019. Monads with merging. (June 2019). working paper or preprint. https://inria.hal.science/hal-02150199

[18] Exequiel Rivas and Tarmo Uustalu. 2024. Concurrent monads for shared state. In *PPDP 2024*. doi:10.1145/3678232.3678249

[19] K. C. Sivaramakrishnan, Stephen Dolan, Leo White, Tom Kelly, Sadiq Jaffer, and Anil Madhavapeddy. 2021. Retrofitting effect handlers onto OCaml. In *PLDI '21*. ACM, 206–221. doi:10.1145/3453483.3454039

[20] Ian Stark. 2008. Free-algebra models for the pi-calculus. *Theor. Comput. Sci.* 390, 2-3 (2008), 248–270. doi:10.1016/J.TCS.2007.09.024

[21] Rob J. van Glabbeek and Gordon D. Plotkin. 2010. On CSP and the Algebraic Theory of Effects. In *Reflections on the Work of C. A. R. Hoare*, A. W. Roscoe, Clifford B. Jones, and Kenneth R. Wood (Eds.). Springer, 333–369. doi:10.1007/978-1-84882-912-1_15

[22] Glynn Winskel. 1986. Event structures. In *Advanced course on Petri nets*. Springer, 325–392.