PROGRAM LOGICS VIA DISTRIBUTIVE MONOIDAL CATEGORIES (EXTENDED ABSTRACT)

FILIPPO BONCHI, ELENA DI LAVORE, MARIO ROMÁN, AND SAM STATON

Abstract. We derive multiple program logics, including correctness, incorrectness, and relational Hoare logic, from the axioms of imperative categories: uniformly traced distributive copy-discard categories. We introduce an internal language for imperative multicategories, on top of which we derive combinators for an adaptation of Dijkstra's guarded command language. Rules of program logics are derived from this internal language. Extended version available at https://www.arxiv.org/abs/2507.18238.

Program logics. Program logics are sets of derivation rules used to reason about program behaviour under input and output conditions. Statements are written as triples $\{p\}\, c\, \{q\}$ of a command c, a precondition p and a postcondition q. The semantics of such a triple, though, depends on the behaviour one is interested in studying. For program correctness, intuitively, the triple is valid if, starting on input states that satisfy p, the output states of the program satisfy q. For example, the following rule of Hoare logic derives a correctness triple for a loop from the correctness triple of its body.

$$\frac{\{b \wedge p\}\, c\, \{p\}}{\{p\}\ \mathsf{while}\, b\, \mathsf{do}\, c\, \{(\neg b) \wedge p\}}$$

However, correctness is only one of the possible triple interpretations; intensive research has produced logics for a myriad of triple interpretations, and for multiple program semantics.

Program logics start by fixing a semantics for their commands, an interpretation for their triples, and derivation rules for its logic. Command semantics can be partial [Hoa69, Ben04], relational [Win93, O'H19] or stochastic [Kam18, BKOZB12, ZDS23]. Triples can capture program correctness [Hoa69], incorrectness [dVK11, O'H19] or quantitative aspects of execution [ZDS23, ABDG25]. After these two choices, the logic is completed with a set of derivation rules that capture the relevant behaviour and are sound for the intended semantics. While they appear to follow some general pattern, the rules of program logics are defined on a case-by-case basis.

Imperative categories. We propose the algebraic structure of *imperative categories*—a variant of Elgot distributive categories—as a foundation for program logics. An imperative category is a distributive copy-discard category that is enriched in posets and that has a posetally-uniform trace for the coproduct. From the axioms of imperative categories, we derive the usual rules of various program logics. The rules about program compositions derive from the categorical structure, those for if-else statements from coproducts and those for while loops from the uniform trace; the monoidal copy-discard structure gives the rules about variable assignment and sampling, and also allows us to capture relational program logics by considering monoidal products of programs. From the models of imperative categories, we expand the scope of these rules beyond a fixed semantics. The categorical structure is common to the usual relational, partial, and probabilistic semantics, while remaining more general. In particular, the rules that we derive are sound in the categories of sets and partial functions, sets and relations, sets and stochastic maps, standard Borel spaces and measurable stochastic kernels.

Imperative categories come with an internal language that we develop and employ through the paper: an internal language that mimics *unstructured programming*, with arbitrary jumps to labelled looping points (marked by "**loop**" followed by a label). Unstructured programming is needed for full expressivity, but certainly not always desirable [Dij68]; in fact, while unstructured and typed, the internal language is actually inspired by a structured and untyped one: the famous Dijkstra's *guarded command language* [Dij75]. Dijkstra's *command language* is recovered from the endomorphisms of imperative categories.

(Filippo Bonchi) Università di Pisa
(Elena Di Lavore, Mario Román, Sam Staton) University of Oxford

References

[ABDG25] Martin Avanzini, Gilles Barthe, Davide Davoli, and Benjamin Grégoire. A quantitative probabilistic relational Hoare logic. *Proceedings of the ACM on Programming Languages*, 9(POPL):1167–1195, 2025.

[Ben04] Nick Benton. Simple relational correctness proofs for static analyses and program transformations. *ACM SIGPLAN Notices*, 39(1):14–25, 2004.

[BKOZB12] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic relational reasoning for differential privacy. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 97–110, 2012.

[Dij68] Edsger W Dijkstra. Letters to the editor: go to statement considered harmful. *Communications of the ACM*, 11(3):147–148, 1968.

[Dij75] Edsger W Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

[dVK11] Edsko de Vries and Vasileios Koutavas. Reverse hoare logic. In Gilles Barthe, Alberto Pardo, and Gerardo Schneider, editors, *Software Engineering and Formal Methods*, pages 155–171. Springer Berlin Heidelberg, 2011.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[Kam18] Benjamin Lucien Kaminski. *Advanced weakest precondition calculi for probabilistic programs*. PhD thesis, RWTH Aachen University, 2018.

[O'H19] Peter W O'Hearn. Incorrectness logic. *Proceedings of the ACM on Programming Languages*, 4(POPL):1–32, 2019.

[Win93] Glynn Winskel. *The formal semantics of programming languages: an introduction*. MIT press, 1993.

[ZDS23] Noam Zilberstein, Derek Dreyer, and Alexandra Silva. Outcome logic: A unifying foundation for correctness and incorrectness reasoning. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1):522–550, 2023.