On Circuit Description Languages, Indexed Monads, and Resource Analysis

This work proposes a new semantic foundation for circuit description languages used in quantum programming, with particular focus on the Proto-Quipper family of calculi inspired by the Quipper language. The central goal is to provide a clean and compositional way to reason about both program behavior and the quantum circuits produced during execution, especially with respect to resource usage.

Existing denotational models for Proto-Quipper typically rely on presheaf-based constructions. While powerful, these models tend to blur the distinction between the value computed by a program and the circuit generated as a side effect. As a result, it becomes difficult to extract precise information about circuit structure, such as its size or interface, and to justify advanced type systems that aim to control circuit resources.

To overcome this limitation, the paper introduces a monadic interpretation of circuit construction based on indexed monads. In this approach, program evaluation produces a pair consisting of a value and a circuit, and these two components are kept separate throughout the semantics. Indexed monads are essential here because the type of the generated circuit depends on the input and output wire interfaces, which vary across computations.

Building on this idea, the authors define a refined calculus called Proto-Quipper-C. This language extends Proto-Quipper-M by enriching function types with explicit information about the circuit data captured by closures. Making this information part of the type system is crucial for determining the interface of circuits generated by higher-order functions and for enabling modular reasoning about circuit composition.

The denotational semantics of Proto-Quipper-C is formulated using a parameterized Freyd category induced by the circuit indexed monad. Programs are interpreted compositionally, with values and circuits handled in distinct semantic components. The model is shown to be sound and computationally adequate with respect to the operational semantics, meaning that it precisely characterizes program evaluation and circuit generation.

The framework is further extended to support effect typing, allowing quantitative properties of circuits—such as size or width—to be tracked using abstract circuit algebras. This extension naturally accommodates circuit optimizations while preserving soundness. Finally, the paper outlines how dependent types could be incorporated into the system using families constructions, opening the door to even more expressive reasoning about quantum programs and their resource requirements.

Overall, the proposed monadic semantics provides a flexible and conceptually clear foundation for circuit description languages, supporting precise resource analysis and extensible type systems for quantum programming.