The field of the Quantum Programming Languages is growing fast since 1996. There are several contributions, especially on $\lambda$-Calculus and Functional Programming.

One of the most significant works on Quantum $\lambda$-Calculus was developed by André van Tonder\[4, 5\]. His Calculus uses the concept of the intensively studied Linear Logic. In fact, it was enough to use of the syntax introduced by Philip Wadler\[6\] for untyped Linear Logic.

van Tonder’s Calculus is not complete: Measurement is not in the Calculus. Peter Selinger argues\[2\] that it is not possible to add measurement to van Tonder’s Calculus without making it typed. Following his ideas, Selinger and Benoît Valiron developed a Quantum Lambda Calculus in 2006\[3\].

The purpose of our work is to add measurement to van Tonder’s Calculus, keeping it untyped.

To ensure reversibility, van Tonder added a history track that keeps the necessary information to be able to follow the reductions in backward and obtain the original state. To avoid superpositions between the history track and the computational state, he follow the ideas of the Linear Logic. However, Measurement was not included in the calculus by delaying it to the end of the algorithms. His calculus includes constants that denotes qubits and elementary gate operations on qubits.

The fist change that we needed to do, was to give syntax to the constants. Measurement cares about the shape of the qubit, so we had to made a syntax that ensures well-formed terms for qubits.

As Measurement is a probabilistic operation, we took some tools from the Probabilistic $\lambda$-Calculus defined by Di Pierro, Hanking and Wiklicky[1] which gave us a simple way to have probabilistic inference rules.

With that, it was possible to make an operational model that includes measurement.
References


