**Introduction**

Quipper is a functional programming language for quantum computing. Proto-Quipper is a family of languages aiming to provide a formal foundation for Quipper. We added a notion of dependent types to Proto-Quipper in 2020, which enables Proto-Quipper to express family of quantum circuits indexed by parameters. Recently, we have extended Proto-Quipper with a construct called dynamic lifting. Dynamic lifting is an operation that enables a state, such as the result of a measurement, to be lifted to a parameter, where it can influence the generation of the next portion of the circuit. As a result, dynamic lifting enables Proto-Quipper programs to interleave classical and quantum computation.

**Objectives of Proto-Quipper**

- Provide formal foundations for high-level quantum computing languages.
- Identify high-level programming constructs for quantum circuits.
- Support programming a wide range of quantum algorithms.

**A Repeat-Until-Success Algorithm**

The repeat-until-success paradigm provides a technique to apply a unitary that cannot be implemented exactly, at the cost of potentially running the same circuit multiple times. In order to apply a non-Clifford+$T$ gate $N$ to a target qubit $|\phi\rangle$, one first initializes several ancillary qubits before applying a well-chosen Clifford+$T$ circuit $C$ to the target and the ancillas. If all of the measurement results are 0, the target qubit is guaranteed to be in the state $N|\phi\rangle$. Otherwise, a correction is applied to the target to return it to its initial state and the process is repeated.

The following circuit gives an illustration of how to implement the repeat-until-success algorithm and Quantum Fourier Transform. They showcase the use of dynamic lifting and dependent types.

**Quantum Fourier Transform**

The quantum Fourier transform is the map defined by

$$|a_1, \ldots, a_n\rangle \mapsto (|0\rangle + e^{2\pi i a_1} |1\rangle) \cdots (|0\rangle + e^{2\pi i a_n} |1\rangle)|0\rangle + e^{2\pi i a_1} |1\rangle)$$

Let us define the controlled rotation gates $R(k)$ by

$$R(k) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i k/2^j} \end{pmatrix}.$$  

Applying the Hadamard gate to the first qubit produces the following state

$$R_i(a_1, \ldots, a_n) = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i a_i} |1\rangle) \otimes |a_2, \ldots, a_n\rangle,$$

where the subscript on the gate indicates the qubit on which the gate acts. We then apply a sequence of controlled rotations using the the first qubit as the target. This yields

$$R_{ij}(a_i, a_j) = R_j(\pi/2^i) R_i(a_1, \ldots, a_j) = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i a_i a_j} |1\rangle) \otimes |a_2, \ldots, a_n\rangle,$$

where the subscripts $i$ and $j$ in $R(k)_{ij}$ indicate the target and control qubit, respectively. The following circuit corresponds to QFT on 5 qubits.

**Proto-Quipper Program for $V_5$**

The following is a Proto-Quipper program that implements the repeat-until-success algorithm for $V_5$ gate outlined above:

```proto-quipper
v3 : !(Qubit -> Qubit)
v3 q =
  let a1 = tgate_inv H (Init1())
  a2 = H (Init1())
  (a1, a2) = CNot a1 a2
  a1 = H (TGate a1)
  in if dynlift (Meas a1) then
    let _ = Discard (Meas a2) in v3 q
  else let q = ZGate (TGate q) in if dynlift (Meas q) then v3 (ZGate q) else q
```

**Summary and Future Work**

We show Proto-Quipper programs for a repeat-until-success algorithm and Quantum Fourier Transform. They showcase the use of dynamic lifting and dependent types. For future work, on the aspect of language design, we are investigating the support for controlling and reversing quantum circuits in Proto-Quipper. On the aspect of implementation and compilation, we plan to explore the possibility of compiling Proto-Quipper to lower level languages such as OpenQASM and QIR.

**References**


**Acknowledgements**

Joint work with Kohei Kishida, Neil J. Ross and Peter Selinger.

**Contact Information**

- Gitlab repository: [https://gitlab.com/frank-peng-fu/dpqq-tutorial](https://gitlab.com/frank-peng-fu/dpqq-tutorial)
- Web: [https://cse.sc.edu/~pfu](https://cse.sc.edu/~pfu)
- Email: pfu@cse.sc.edu