How do we make a simple programming language ~~for~~ QC?

* Simple Types does not quite work

$\Delta = \lambda x.(x, x) \ : \ \text{Qubit} \to \text{Qubit} \times \text{Qubit}$

Δ violate no-cloning.

* Solution: Linear Types!

$A ::= \text{Bool} \mid \text{Qubit} \mid A \multimap B \mid A \otimes B \mid !A \mid \text{Unit}$

① informally $A \otimes B$ corresponds to the tensor product of A and B.

② $A \multimap B$ corresponds to "linear functions"

③ $!A$ indicates reusability, as we don't want everything to be linear.

We ~~write~~ write Φ to be "parameter context".

i.e. $\Phi = [x_1 : P_1, \cdots x_n : P_n]$ i.e. all the variable in Φ are reusable.

$P ::= \text{Bool} \mid !A \mid \text{Unit} \mid P \otimes P'$

Terms: $M ::= x \mid l_i \mid \lambda x.M \mid MN \mid \text{lift } M \mid$

$\text{force } M \mid (M,N) \mid$

$\text{let } (x,y) = M \text{ in } N \mid$

$\text{if } M \text{ then } N_1 \text{ else } N_2$.

Values

$V ::= \cdots \mid \text{lift} M \mid l_i$

$l \in \mathcal{L}$ countable inf. set.

~~U for gates~~

(We write Γ for any context,) ~~it can be a parameter context~~)

Typing Rules.

$$\boxed{\frac{\Phi}{\Phi, l : \text{Qubit} \vdash l : \text{Qubit}}}$$

$$\frac{}{\Phi \vdash () : \text{Unit}} \qquad \frac{}{\Phi, x:A \vdash x:A} \qquad \frac{\Gamma, x:A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}$$

$$\text{app}\ \frac{\Phi, \Gamma_1 \vdash M : A \multimap B \qquad \Phi, \Gamma_2 \vdash N : A}{\Phi, \Gamma_1, \Gamma_2 \vdash MN : B} \qquad \frac{\Phi, \Gamma_1 \vdash M : A \qquad \Phi, \Gamma_2 \vdash N : B}{\Phi, \Gamma_1, \Gamma_2 \vdash (M,N) : A \otimes B}$$

$$\frac{\Phi, \Gamma_1 \vdash M : A \otimes B \qquad \Phi, \Gamma_2, x:A, y:B \vdash N : C}{\Phi, \Gamma_1, \Gamma_2 \vdash \text{let } (x,y) = M \text{ in } N : C}$$

$$\text{lift}\ \frac{\Phi \vdash M : A}{\Phi \vdash \text{lift} M : !A} \qquad \frac{\Gamma \vdash M : !A}{\Gamma \vdash \text{force} M : A}$$

$$\frac{\Phi, \Gamma_1 \vdash P : Bool \qquad \begin{array}{c}\Phi, \Gamma_2 \vdash M : C \\ \Phi, \Gamma_2 \vdash N : C\end{array}}{\Phi, \Gamma_1, \Gamma_2 \vdash \text{if } P \text{ then } M \text{ else } N : C}$$

Note: ① Context ~~cate~~ play a more prominent role in linear type system, the ~~some~~ "lift" rule even requires context to be parameter context.

Also Note the variable rule is very different from STLC.

② App, pair rules split contexts.

③ can we type the following?

$\vdash \lambda x.(x, x) : Qubit \multimap Qubit \otimes Qubit.$

$$\frac{\dfrac{x : Qubit \vdash x : Qubit \qquad ? \vdash x : Qubit}{x : Qubit \vdash (x, x) : Qubit \otimes Qubit}}{\vdash \lambda x.(x, x) : Qubit \multimap Qubit \otimes Qubit}$$

Note: ① $\vdash \lambda x.(x, x) : !A \multimap !A \otimes !A$.

This is because $!A$ is a parameter type.

② $\cancel{\vdash} \lambda x.() : Qubit \multimap Unit$.

We can not 'forget' the existence of a qubit.

③ ~~$y : Qubit \vdash \lambda x$~~ The variable 'y' is used once, even though it occurs twice.

$z : Bool, y : Qubit \vdash \text{if } z \text{ then } \underline{y} \text{ else } H \underline{y}$.

* Incorporate quantum operations.

$$\frac{}{\vdash H : !(Qubit \multimap Qubit)} \qquad \frac{}{\vdash Init0 : !(Unit \multimap Qubit)}$$

X, Y, Z, T,

$$\frac{}{\vdash Meas : !(Qubit \multimap Bool)} \qquad \frac{}{\vdash CX : !(Qubit \otimes Qubit \multimap Qubit \otimes Qubit)}$$

We do have: $\vdash \lambda x.(\lambda y.())(\underline{\cancel{Meas}(\text{force } Meas)\, x}) : Qubit \multimap Unit$.