Proto-Quipper with dynamic lifting

Frank Fu

Joint work with K. Kishida, N.J. Ross and P. Selinger

NYUAD Quantum Colloquium

Background: Quipper

- Quantum circuit description language.
- Support high-level quantum circuit operations.
- Batch processing, two runtimes.
- Allows interleaving runtimes via *dynamic lifting*.

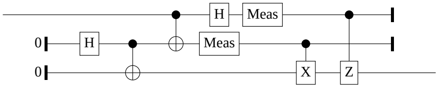
Background: Proto-Quipper

- Provide formal foundation for features of Quipper.
  - Formal type system.
  - Operational semantics.
  - Categorical semantics.
- Proto-Quipper-M (Rio and Selinger 2018).
- Proto-Quipper-D (Fu, Kishida and Selinger 2020).
- Proto-Quipper-Dyn (Fu, Kishida, Ross and Selinger 2022).

Programming quantum circuits in Proto-Quipper

```
tele : !(Qubit -> Qubit)
tele q =
  let (b, a) = bell00 ()
      (x, y) = alice a q
      z = bob b x y
  in z

boxTele : Circ(Qubit , Qubit)
boxTele = box Qubit tele
```

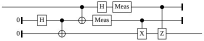


Two runtimes

```
alice2 : !(Qubit -> Qubit -> Bool * Bool)
alice2 a q =
  let (a, q) = CNot a q
      q = H q
  in (dynlift (Meas a), dynlift (Meas q))

bob2 : !(Qubit -> Bool -> Bool -> Qubit)
bob2 q x y =
  let q = if x then QNot q else q
      q = if y then ZGate q else q
  in q
```

Circuit generation time



Circuit execution time

Values in the two runtimes

- Values in circuit generation time.
  Parameters (e.g., **Nat**, **Bool**).
- Values in circuit execution time.
  States (e.g., **Qubit**, **Bit**).
  Measurement is a gate: **Qubit** $\rightarrow$ **Bit**.
- Dynamic lifting.
  A *language construct* that "lifts" a **Bit** to **Bool**.

Dynamic lifting

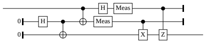
Code

```
alice2 : !(Qubit -> Qubit -> Bool * Bool)
alice2 a q =
  let (a, q) = CNot a q
      q = H q
  in (dynlift (Meas a), dynlift (Meas q))

bob2 : !(Qubit -> Bool -> Bool -> Qubit)
bob2 q x y =
  let q = if x then QNot q else q
      q = if y then ZGate q else q
  in q
```

Nat, Bool

Circuit



Qubit, Bit

Quantum computer

Why dynamic lifting?

- Interleaving the two runtimes.
  - Admit more quantum algorithms.
- Incorporation of circuit execution time.
  - Potential to incorporate multiple “backends”.
  - Test quantum circuits.

Modalities for dynamic lifting

- Mode $\alpha = 0 \mid 1$.
  - 1 indicates boxable circuits.
  - 0 indicates dynamic lifting.
- Types.

$$A, B ::= \mathbf{Qubit} \mid \mathbf{Bit} \mid A \otimes B \mid A \multimap_{\alpha} B \mid !_{\alpha} A \mid \mathbf{Circ}(S, U)$$

- Typing judgment.

$$\Gamma \vdash_{\alpha} M : A$$

Type system for dynamic lifting

- Dynamic lifting.

$$\frac{\Gamma \vdash_\alpha M : \mathbf{Bit}}{\Gamma \vdash_0 \mathsf{dynlift}\ M : \mathbf{Bool}}$$

- Modality indicates boxability.

$$\frac{\Gamma \vdash_\alpha M : !_1(S \multimap_1 U)}{\Gamma \vdash_\alpha \mathsf{box}\ S\ M : \mathbf{Circ}(S, U)}$$

- Type system tracks modalities.

$$\frac{\Gamma, x : A \vdash_\alpha M : B}{\Gamma \vdash_1 \lambda x.M : A \multimap_\alpha B} \qquad \frac{\Gamma_1 \vdash_{\alpha_1} M : A \multimap_{\alpha_2} B \quad \Gamma_2 \vdash_{\alpha_3} N : A}{\Gamma_1, \Gamma_2 \vdash_{\alpha_1 \& \alpha_2 \& \alpha_3} MN : B}$$

Operational Semantics

- Circuit generation time: $(C, M) \Downarrow (C', V)$
- Circuit execution time: $(Q, M) \Downarrow \sum_{i \in [n]} p_i (Q_i, V_i)$

$$\frac{(Q, M) \Downarrow (Q', \ell) \quad \text{read}(Q', \ell)}{(Q, \mathsf{dynlift}\, M) \Downarrow p_1 (Q_1, \text{True}) + p_2 (Q_2, \text{False})}$$

Symmetric monoidal categories for the two runtimes

- Category of quantum circuits $\mathbf{M}$.
  - Objects.
    $\mathbf{Bit}$, $\mathbf{Qubit}$ etc.
  - Morphisms.
    $\mathsf{Meas} : \mathbf{Qubit} \to \mathbf{Bit}$, $H : \mathbf{Qubit} \to \mathbf{Qubit}$.
- Category of quantum operations $\mathbf{Q}$.
  - $\mathbf{Bit} = I + I$.
  - $\mathbf{Q}$ is enriched with convex space.
- Identity-on-object symmetric monoidal functor $J : \mathbf{M} \to \mathbf{Q}$.
- We assume $\mathbf{M}, \mathbf{Q}, J$ are given.

Categorical semantics for dynamic lifting

What would a category $\mathbf{A}$ for dynamic lifting looks like?

- $\mathbf{A}$ has to admit a linear-non-linear adjunction.

$$p \dashv \flat : \mathbf{Set} \to \mathbf{A}$$

- Dynamic lifting.

$$\begin{array}{ccc} \mathbf{Bool} & \xrightarrow{\text{init}} & \mathbf{Bit} \\ & \searrow^{\eta} & \downarrow{\scriptstyle \text{dynlift}} \\ & & T\mathbf{Bool} \end{array}$$

- Modeling the two runtimes.

$$\begin{array}{ccc} \mathbf{M} & \hookrightarrow & \mathbf{A} \\ \downarrow{\scriptstyle J} & & \downarrow \\ \mathbf{Q} & \hookrightarrow & Kl_T(\mathbf{A}) \end{array}$$

Interpretation of the type system

- Modality interpreted by the monad T when $\alpha = 0$.
- Types: $!_\alpha A$ and $A \multimap_\alpha B$.
  - $[\![!_\alpha A]\!] = p\flat\alpha[\![A]\!]$
  - $[\![A \multimap_\alpha B]\!] = [\![A]\!] \multimap \alpha[\![B]\!]$.
- Typing judgments.
  - $\Gamma \vdash_0 M : A$.
    $[\![M]\!] : [\![\Gamma]\!] \to T[\![A]\!]$
  - $\Gamma \vdash_1 M : A$.
    $[\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$

How to construct a category for dynamic lifting?

Our answer: biset-enrichment

- The category of *bisets*, i.e., $\mathbf{Set}^{\mathbf{2}^{\mathrm{op}}}$.
- Objects: (X_0, X_1, f).

$$\begin{array}{c} X_1 \\ \downarrow{\scriptstyle f} \\ X_0 \end{array}$$

- Morphisms: (h_0, h_1).

$$\begin{array}{ccc} X_1 & \xrightarrow{h_1} & Y_1 \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle g} \\ X_0 & \xrightarrow{h_0} & Y_0 \end{array}$$

- Monad $T(X_0, X_1, f) = (X_0, X_0, \mathrm{Id})$.

$\mathcal{V}$-enriched categories

Let $\mathcal{V}$ be a monoidal category. A $\mathcal{V}$-enriched category $\mathbf{A}$ is given by the following:

- A class of objects, also denoted $\mathbf{A}$.
- For any $A, B \in \mathbf{A}$, an object $\mathbf{A}(A, B)$ in $\mathcal{V}$.
- For any $A \in \mathbf{A}$, a morphism $u_A : I \longrightarrow \mathbf{A}(A, A)$ in $\mathcal{V}$.
- For any $A, B, C \in \mathbf{A}$, a morphism $c_{A,B,C} : \mathbf{A}(A, B) \otimes \mathbf{A}(B, C) \longrightarrow \mathbf{A}(A, C)$ in $\mathcal{V}$.
- u and c must satisfy suitable diagrams in $\mathcal{V}$.

The biset-enriched category $\mathbf{C}$

- $\operatorname{obj}(\mathbf{C}) = \operatorname{obj}(\mathbf{Q}) = \operatorname{obj}(\mathbf{M})$.
- For any $A, B \in \mathbf{C}$, the hom-object $\mathbf{C}(A, B)$ is a biset:

$$\begin{array}{c} \mathbf{M}(A, B) \\ \big\downarrow {\scriptstyle J_{AB}} \\ \mathbf{Q}(A, B) \end{array}$$

The biset-enriched category $\mathbf{C}$

A global map $f : 1 \to \mathbf{C}(A, B)$ is the following.

$$\begin{array}{ccc} 1 & \xrightarrow{f_1} & \mathbf{M}(A,B) \\ & \searrow^{f_0} & \downarrow J_{AB} \\ & & \mathbf{Q}(A,B) \end{array}$$

The biset-enriched category $\overline{\mathbf{C}}$

Define $\overline{\mathbf{C}} := \mathcal{V}^{\mathbf{C}^{\mathrm{op}}}$, where $\mathcal{V} = \mathbf{Set}^{\mathbf{2}^{\mathrm{op}}}$.

- $\overline{\mathbf{C}}$ is monoidal closed.
- $\mathbf{Bit} = y\mathbf{Bit} = \mathbf{C}(-, \mathbf{Bit})$.
- $\mathbf{Bool} = yI + yI = \mathbf{C}(-, I) + \mathbf{C}(-, I)$.
- A commuative $\mathcal{V}$-monad $\overline{T}(F) = T \circ F$.

The biset-enriched category $\overline{\mathbf{C}}$

$\overline{\mathbf{C}} := \mathcal{V}^{\mathbf{C}^{\mathrm{op}}}$, where $\mathcal{V} = \mathbf{Set}^{\mathbf{2}^{\mathbf{op}}}$.

$$\begin{array}{ccc} \mathbf{Bool} & \xrightarrow{\ \mathsf{init}\ } & \mathbf{Bit} \\ & {\scriptstyle \eta}\searrow & \downarrow {\scriptstyle ?} \\ & & \overline{T}\mathbf{Bool} \end{array}$$

$$\begin{array}{ccc} \mathbf{Q}(-,I) + \mathbf{Q}(-,I) & \longrightarrow & \mathbf{Q}(-,I+I) \\ & {\scriptstyle \mathrm{Id}}\searrow & \downarrow {\scriptstyle ?} \\ & & \mathbf{Q}(-,I) + \mathbf{Q}(-,I) \end{array}$$

The biset-enriched category $\overline{\mathbf{C}}$

$\overline{\mathbf{C}} := \mathcal{V}^{\mathbf{C}^{\mathrm{op}}}$, where $\mathcal{V} = \mathbf{Set}^{\mathbf{2}^{\mathrm{op}}}$.

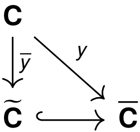
$$\begin{array}{ccc} \mathbf{Bool} & \xrightarrow{\mathsf{init}} & \mathbf{Bit} \\ & \underset{\eta}{\searrow} & \downarrow{?} \\ & & \overline{T}\mathbf{Bool} \end{array}$$

$$\begin{array}{ccc} \mathbf{Q}(-,I) + \mathbf{Q}(-,I) & \longrightarrow & \mathbf{Q}(-,I+I) \\ & \underset{\mathrm{Id}}{\searrow} & \downarrow{?} \\ & & \mathbf{Q}(-,I) + \mathbf{Q}(-,I) \end{array}$$

Yoneda embedding does not preserve coproducts!

The subcategory $\widetilde{\mathbf{C}}$

- For every $\mathcal{V}$-functor $F : \mathbf{C} \longrightarrow \mathcal{V} \in \widetilde{\mathbf{C}}$, there is an ordinary functor $F^0 : \mathbf{Q} \longrightarrow \mathbf{Set}$
- Define $\widetilde{\mathbf{C}}$ to be the full subcategory of $\overline{\mathbf{C}}$, where for every $F \in \widetilde{\mathbf{C}}$, the functor $F^0 : \mathbf{Q} \longrightarrow \mathbf{Set}$ is *product-preserving*.
- We call the embedding $\overline{y} : \mathbf{C} \longrightarrow \widetilde{\mathbf{C}}$ *Lambek embedding*.



- We take the underlying category of $\widetilde{\mathbf{C}}$ as $\mathbf{A}$.
  - $\mathrm{obj}(\mathbf{A}) = \mathrm{obj}(\widetilde{\mathbf{C}})$.
  - $\mathbf{A}(A, B) = \mathcal{V}(1, \widetilde{\mathbf{C}}(A, B))$.

Summary

- A type system for dynamic lifting.
- Categorical semantics for dynamic lifting.
- Construction of a categorical model for dynamic lifting.

Thank you!