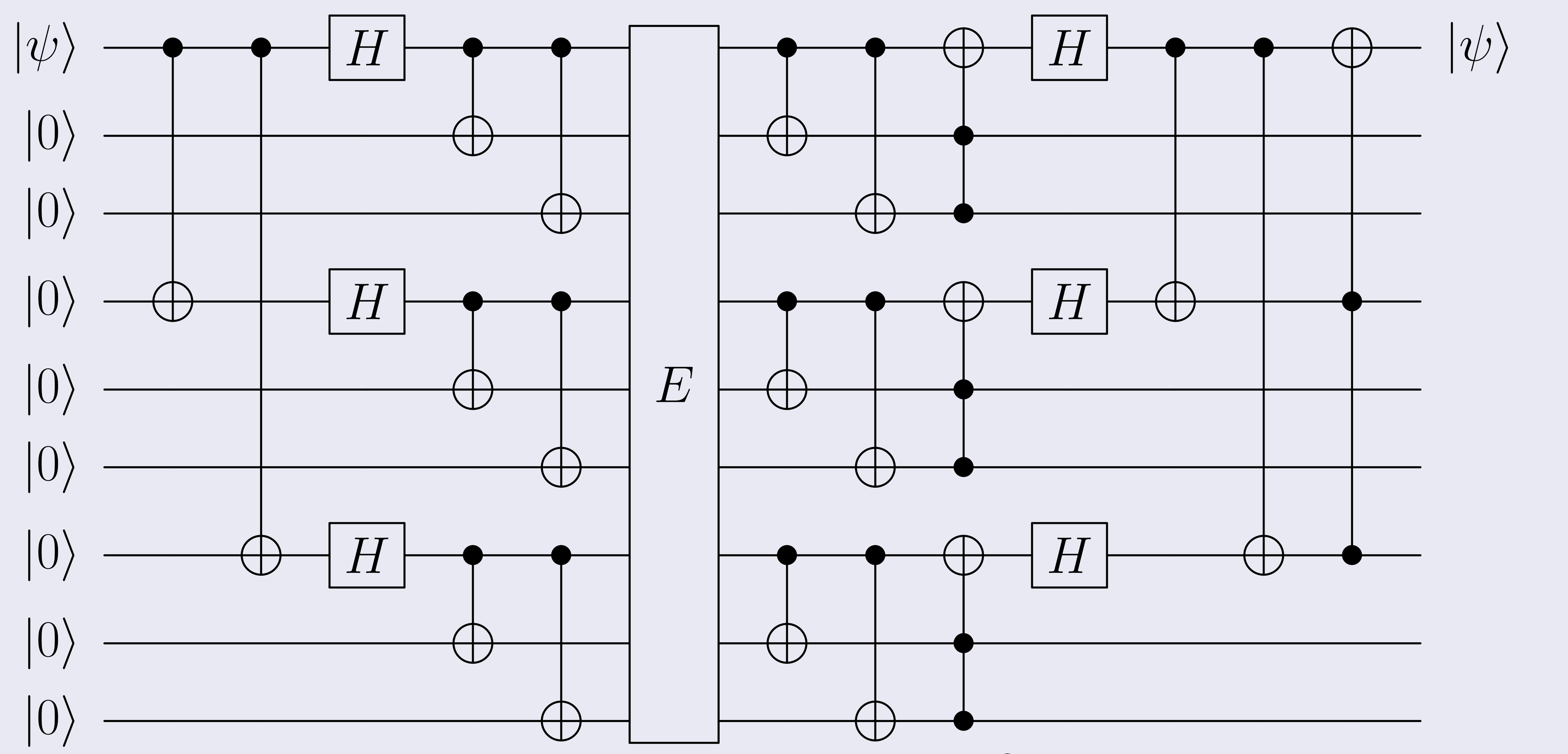
Formalizing quantum circuits with MathComp/Ssreflect

Takafumi Saikawa and Jacques Garrigue   Nagoya University

Motivation

We want to formalize quantum circuits

Shor's 9-qubit code (correcting both flips)

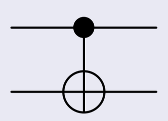


Basic differences: bits and qubits

Classical	Quantum
bit $\in \{0,1\}$	qubit $\in \mathbb{C}^2$
functions in **Set**	unitary transformations in **FdHilb**
direct product: $\mathbf{Set}(X \times Y, Z) \cong \mathbf{Set}(X, Z^Y)$	tensor product $\mathbf{FdHilb}(X \otimes Y, Z) \cong \mathbf{FdHilb}(X, Z^Y)$

Problem

- Each gate (= unitary transformation) is fairly simple:

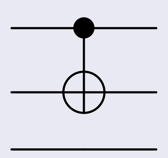


$$= \mathtt{CNOT} = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{bmatrix}$$

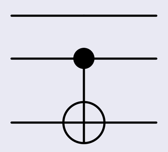
- but when put in a circuit, it becomes a monster:

$$\begin{bmatrix} x & y \\ z & w \end{bmatrix} \otimes \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} xa & xb & ya & yb \\ xc & xd & yc & yd \\ za & zb & wa & wb \\ zc & zd & wc & wd \end{bmatrix}$$

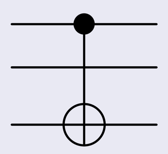
CNOT in 3-qubit circuits : 8×8 matrices



$$= \mathtt{CNOT} \otimes I_2 = \begin{bmatrix} I_2 & 0 & 0 & 0 \\ 0 & I_2 & 0 & 0 \\ 0 & 0 & 0 & I_2 \\ 0 & 0 & I_2 & 0 \end{bmatrix}$$



$$= I_2 \otimes \mathtt{CNOT} = \begin{bmatrix} \mathtt{CNOT} & 0 \\ 0 & \mathtt{CNOT} \end{bmatrix}$$



$$= \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

Our first example (Shor's code) becomes a 512×512 matrix!

The code is available at: `https://github.com/t6s/qecc/`

Lens

Tensor power

Tensor power

- Tensor power $V^{\otimes n}$
  = iterated tensor product $V \otimes \cdots \otimes V$
- If $V = K^m$, $V^{\otimes n} \cong \mathbf{Set}(m^n, K)$

Array of qubits

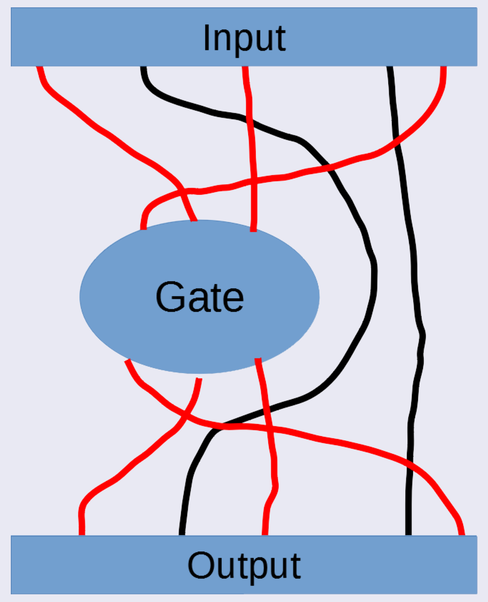
- Qubit $\in \mathbb{C}^2$
- Array of qubits $\in (\mathbb{C}^2)^{\otimes n}$

Operator

Operators on qubits must be

- linear: addition and scalar action must be preserved
- unitary: norm must be preserved

Lens, curry-uncurry, focus



- Lens = injection between finite ordinals, indicating the choice of wires (red wires in the picture)
- Curry / Uncurry = currying along a given lens which quotients away the unused (black) wires
- Focusing = composing curry, gate and uncurry to build the diagram

`lens`

$$\mathtt{lens\ n\ m} = (\{1,\ldots,m\} \rightarrowtail \{1,\ldots,n\})$$

`curry` and `uncurry`

For T a vector space and ℓ : `lens n m`,

$$(T^2)^{\bigotimes n} \cong T^{2^n} \xrightarrow{\mathtt{curry}_\ell} \left(T^{2^{n-m}}\right)^{2^m} \cong \left(\left(T^{2^{n-m}}\right)^2\right)^{\bigotimes m}$$

$$\left(T^{2^n} = \mathbf{Set}(2^n, T) \cong \mathbf{Set}(2^m, \mathbf{Set}(2^{n-m}, T))\right)$$

$$\mathtt{uncurry}_\ell = \mathtt{curry}_\ell^{-1}$$

`focus`

And for G unitary,

$$\mathtt{focus}_\ell\ G = \mathtt{uncurry}_\ell \circ G \circ \mathtt{curry}_\ell$$

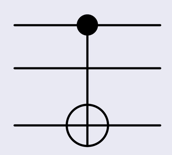
Polymorphic operator

For `focus` to typecheck, the unitary operator G must actually be polymorphic:

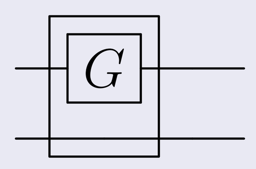
$$G : \forall T : \text{vector sp.}, (T^2)^{\bigotimes n} \xrightarrow{\text{unitary}} (T^2)^{\bigotimes n}$$

$$\mathtt{focus}_\ell\ G = \lambda T.(\mathtt{uncurry}_\ell \circ G_{T^{2^{n-m}}} \circ \mathtt{curry}_\ell)$$

Examples:



$$= \mathtt{focus}_{\{1\mapsto 1, 2\mapsto 3\}_3}\ \mathtt{CNOT}$$



$$= \mathtt{focus}_{\{1\mapsto 1, 2\mapsto 2\}_3}(\mathtt{focus}_{\{1\mapsto 1\}_2}\ G)$$

Parametricity and naturality

Polymorphism is not enough

$$\mathtt{focus}_\ell\ G = \mathtt{uncurry}_\ell \circ (G_{T^{2^{n-m}}}) \circ \mathtt{curry}_\ell$$

- We know from the type that G is polymorphically linear / unitary
- But they could be unitary / linear differently for each T
- I.e., the matrix representing the linearity might differ between different `T`s
- And `focus` does change `T`

Parametricity

We want G to be represented by a single matrix:

$$\exists M : \text{matrix}, \ \forall T : \text{vector sp.}, \ \forall v : (T^2)^{\bigotimes n}, \ G_T(v) = Mv.$$

Naturality

We can rephrase this parametricity without the existential reference to a matrix, i.e., naturality:

$$\begin{tikzcd} T \arrow[d, "\forall\varphi"] & T^{\otimes I^k} \arrow[r, "f_T"] \arrow[d, "\varphi^{\otimes I^k}"'] & T^{\otimes I^k} \arrow[d, "\varphi^{\otimes I^k}"] \\ T' & T'^{\otimes I^k} \arrow[r, "f_{T'}"'] & T'^{\otimes I^k} \end{tikzcd}$$

Applications

Shor's code

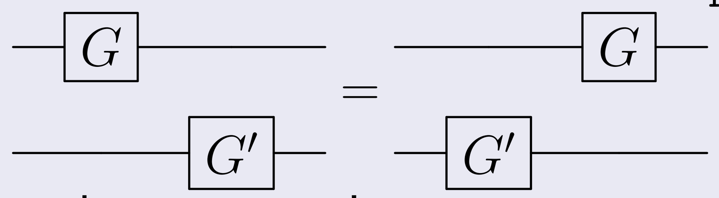
```
Definition bit_flip_enc : endo 3 :=
  tsapp [lens 0; 2] cnot \v  tsapp [lens 0; 1] cnot.
Definition bit_flip_dec : endo 3 :=
  tsapp [lens 1; 2; 0] toffoli \v bit_flip_enc.

Definition sign_flip_dec := bit_flip_dec \v hadamard3.
Definition sign_flip_enc := hadamard3 \v bit_flip_enc.

Definition shor_enc : endo 9 :=
  focus [lens 0; 1; 2] bit_flip_enc \v
  focus [lens 3; 4; 5] bit_flip_enc \v
  focus [lens 6; 7; 8] bit_flip_enc \v
  focus [lens 0; 3; 6] sign_flip_enc.
Definition shor_dec : endo 9 :=
  focus [lens 0; 3; 6] sign_flip_dec \v
  focus [lens 0; 1; 2] bit_flip_dec \v
  focus [lens 3; 4; 5] bit_flip_dec \v
  focus [lens 6; 7; 8] bit_flip_dec.

Definition shor_code (chan : endo 9) :=
  shor_dec \v chan \v shor_enc.
```

Proofs of properties

- If G is unitary, so is $\mathtt{focus}_\ell G$.
- For ℓ : `lens n m` and ℓ' : `lens m p`, $\mathtt{focus}_{\ell\circ\ell'} = \mathtt{focus}_\ell \circ \mathtt{focus}_{\ell'}$
- 
- and many more!