Applying the Formalization of Mathematics to Quantum Computation and Its Models

Takafumi Saikawa[1] Jacques Garrigue[1]
Shinya Katsumata[2]

[1]Graduate School of Mathematics, Nagoya University

[2]Kyoto Sangyo University

Outline

1. What is formalization?
2. Formalizing quantum circuits
3. Visualization

Formalization?

Terminology

Formalization

Writing a proof in full detail, down to the level of logic, and having it checked by an algorithm.

Proof assistant

Software that helps us write proofs and checks them.
Rocq, Isabelle/HOL, Agda, Lean, ...

Logical foundations

Reduce a mathematical theory to logical rules and axioms
$\Longrightarrow$ thus proofs can be checked by an algorithm.

How to reduce

- **Axiomatization**
  e.g. the field axioms
- **Concrete construction**
  e.g. building the reals $\mathbb{R}$

Reduced into

- **Logic**
  e.g. predicate logic
- **Theory for collections**
  set theory, *type theory*, ...

This work is based on **type theory**
(the approach of Rocq and Lean)

Mathematics in type theory

- A type is more or less a set
- Contains the λ-calculus: functions and computation are primitive notions
  - proofs like $1 + 1 = 2$ are completed instantly *by computation*

```
Lemma one_plus_one : 1 + 1 = 2.
Proof. reflexivity. Qed.
```

- Close to categorical logic and topos theory,
  but the notion of "*element of a type*" $(t : T)$ is primitive

Types vs. sets: subtle differences

1. How elements are built is fixed ⇒ induction for free

```
Inductive nat :=
| O : nat
| S : nat -> nat.
```

$$\texttt{nat_ind}: \varphi(0) \land (\forall n,\ \varphi(n) \to \varphi(S\,n)) \to \forall n,\ \varphi(n)$$

2. Elements of different types cannot be mixed directly

sets A, B

$A \cap B, \quad A \cup B$

always defined

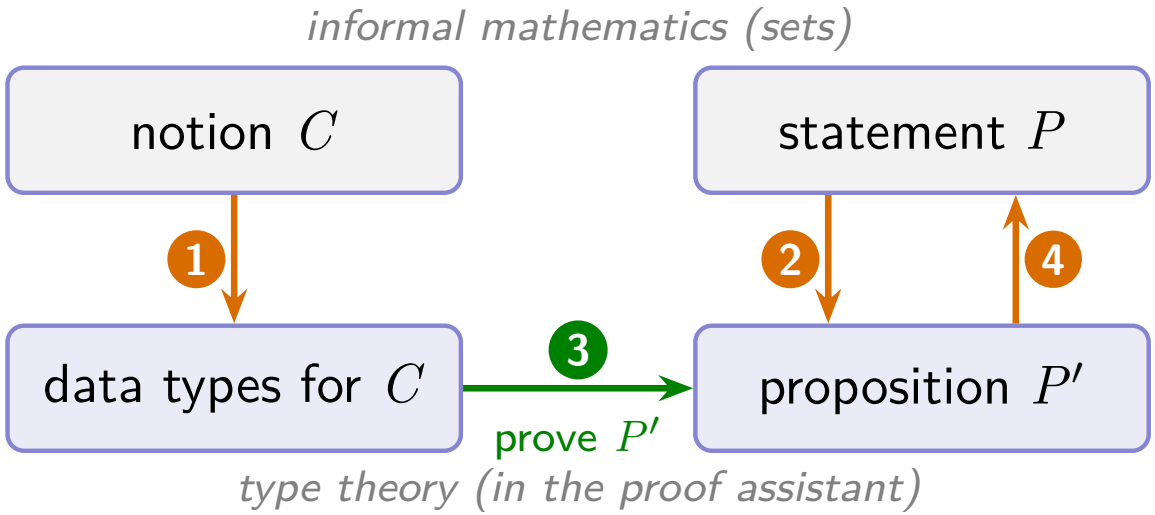
types T, U

$$\begin{array}{ccc} & & U \\ & & \downarrow \\ T & \longrightarrow & T \amalg U \end{array}$$

must be *constructed* from explicit maps

Workflow of a formalization

Formalizing a statement P
(written in ordinary, set-theoretic mathematics):



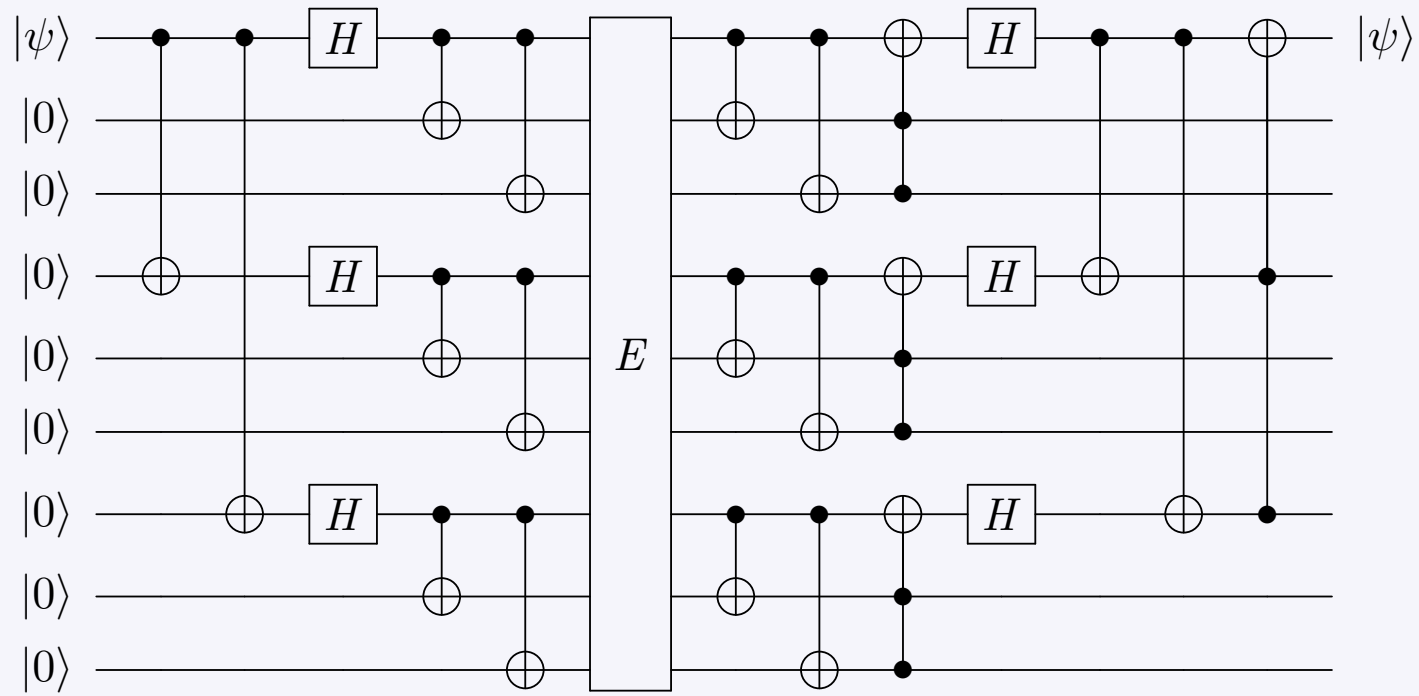
1. Define C and related structures as types, as faithfully as possible
2. Translate P into a type-theoretic proposition P'
3. Prove P' in type theory; the proof is machine-checked
4. Translate back to sets: regard the original P as proved

3: in the proof assistant $\Rightarrow$ highly reliable
1, 2, 4: manual translation / human inspection $\Rightarrow$ less reliable (!)

Quantum circuits

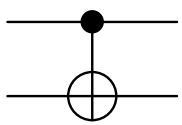
Example: Shor's 9-qubit code

Shor's code: corrects both bit flips and sign flips



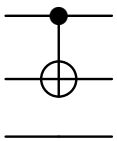
Outer layer: a *sign-flip* code on wires $0, 3, 6$ (bit-flip code conjugated by H); inner layer: a *bit-flip* code on each block $\{0, 1, 2\}$, $\{3, 4, 5\}$, $\{6, 7, 8\}$.

Matrix semantics, example: CNOT

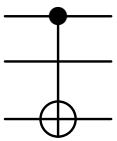


$$= \quad \texttt{CNOT} \quad = \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

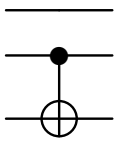
CNOT inside larger circuits



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



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



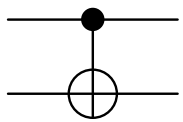
$$= I_2 \otimes \texttt{CNOT} = \begin{bmatrix} \texttt{CNOT} & 0 \\ 0 & \texttt{CNOT} \end{bmatrix}$$

Drawbacks of the matrix view

- non-adjacent wires need extra permutations (swaps)
- every gate becomes a $2^n \times 2^n$ matrix: exponential growth

Strategy

Define gates as **small matrices**, and compose them as **functions**



$$= \texttt{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \texttt{CNOT} : \mathbb{C}^4 \to \mathbb{C}^4$$

Question

How do we apply this function to a part of a general state $\mathbb{C}^{2^n} \cong (\mathbb{C}^2)^{\otimes n}$, $n \geq 2$,

Two tricks

1. Reshape the state (currying)

$$(\mathbb{C}^2)^{\otimes(n+m)} \cong \mathbb{C}^{2^{n+m}} \cong \left(\mathbb{C}^{2^n}\right)^{2^m}$$

a state on $n+m$ wires $=$ a vector of length 2^m whose entries are states on the other n wires

2. Make gates polymorphic

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

still defined when a, b, c, d are vectors

From a function $\mathbb{C}^4 \to \mathbb{C}^4$ to a family (polymorphic function) $\forall \mathbb{V},\ \mathbb{V}^4 \to \mathbb{V}^4$

Polymorphic functions

Functions that can act on many types (or sets).

Ad hoc

- Python's `print` (prints many kinds of input)

Behaves differently for each input type.

Parametric (uniform)

- identity, `swap`, `map`

One single definition acts on all types.

Quantum gates are defined from a *single matrix*, so they belong to the second kind.

Single matrix $\iff$ naturality

Gate $G : \forall \mathbb{V},\ \mathbb{V}^{2^m} \to \mathbb{V}^{2^m}$ is given by a single matrix:

$$\exists M : \mathcal{M}_{2^m}(\mathbb{C}),\ \forall \mathbb{V},\ \forall s : \mathbb{V}^{2^m},\quad G_{\mathbb{V}}(s) = Ms$$

This is equivalent to naturality: for every linear $\varphi : \mathbb{V} \to \mathbb{V}'$,

$$\begin{array}{ccc} \mathbb{V}^{2^m} & \xrightarrow{G_{\mathbb{V}}} & \mathbb{V}^{2^m} \\ {\scriptstyle \varphi^{2^m}}\downarrow & & \downarrow{\scriptstyle \varphi^{2^m}} \\ \mathbb{V}'^{2^m} & \xrightarrow[G_{\mathbb{V}'}]{} & \mathbb{V}'^{2^m} \end{array}$$

`naturalityP` (in Rocq)

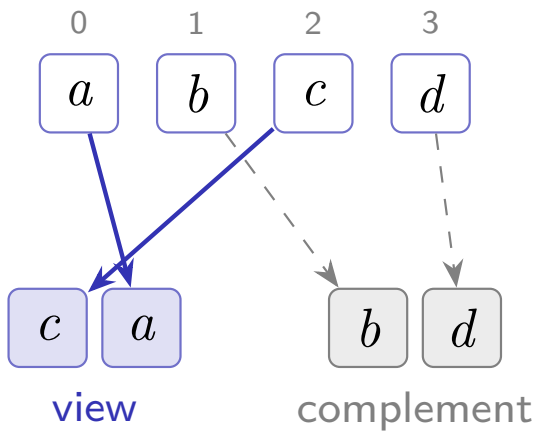
`naturality` $G \iff \exists M,\ \forall T\ s,\ G_T\ s = (\texttt{dpmor}\ M)_T\ s$

In proofs of circuit equivalences we mostly use the latter (naturality).

Lenses: wiring as injections

Definition

`lens n m` := injections $\{0, \ldots, m-1\} \rightarrowtail \{0, \ldots, n-1\}$
("m of the n wires, in order")



$\ell = \{0 \mapsto 2,\ 1 \mapsto 0\}$: `lens 4 2`

extract $\ell\ [a, b, c, d] = [c, a]$

$\ell^C = \{0 \mapsto 1,\ 1 \mapsto 3\}$

merge $\ell\ [a, b]\ [c, d] = [b, c, a, d]$

Defining circuit composition

- Reshape the state along a lens ℓ : `lens n m`:

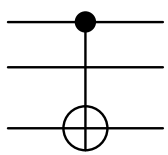
$$\texttt{curry}_\ell : \mathbb{V}^{2^n} \xrightarrow{\cong} \left(\mathbb{V}^{2^{n-m}}\right)^{2^m}, \qquad \texttt{uncurry}_\ell := \texttt{curry}_\ell^{-1}$$

  $\texttt{curry}_\ell\ \sigma\ v\ w := \sigma\,(\text{merge}\ \ell\ v\ w)$: index first by the wires in ℓ, then by the others

- Place a gate $G : \forall \mathbb{V},\ \mathbb{V}^{2^m} \to \mathbb{V}^{2^m}$ between the reshapings:

$$\texttt{focus}_\ell\ G := \texttt{uncurry}_\ell \circ G_{\mathbb{V}^{2^{n-m}}} \circ \texttt{curry}_\ell$$

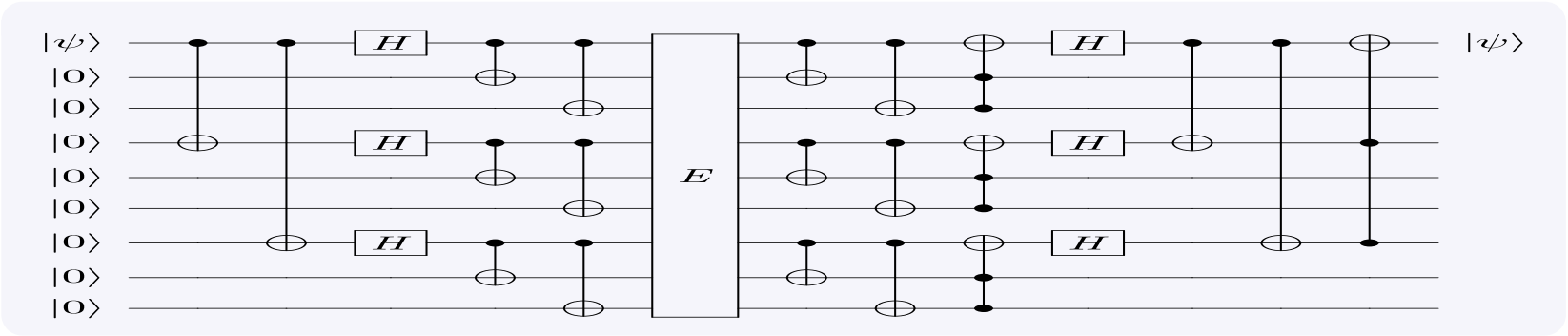
- Example:



$$= \texttt{focus}_{\{0\mapsto 0,\ 1\mapsto 2\}}\ \texttt{CNOT}$$

The gate never sees the other wires; no Kronecker products needed.

Circuits as code




```
Definition bit_flip_enc : endo3 :=
  focus [lens 0; 2] cnot • focus [lens 0; 1] cnot.
Definition bit_flip_dec : endo3 :=
  focus [lens 1; 2; 0] toffoli • bit_flip_enc.
Definition sign_flip_enc := hadamard3 • bit_flip_enc.
Definition sign_flip_dec := bit_flip_dec • hadamard3.

Definition shor_enc : endo9 :=
  focus [lens 0; 1; 2] bit_flip_enc •
  focus [lens 3; 4; 5] bit_flip_enc •
  focus [lens 6; 7; 8] bit_flip_enc •
  focus [lens 0; 3; 6] sign_flip_enc.
Definition shor_code (channel : endo9) :=
  shor_dec • channel • shor_enc.

Lemma shor_code_id i :
  shor_code id |i,0,0,0,0,0,0,0,0⟩ = |i,0,0,0,0,0,0,0,0⟩.
```

Sub-circuits are reused *as they are*, just focused on different wires.

Proof sketch for `shor_code_id`

Idea: pair each encoder with its decoder and turn the pair into a Toffoli gate, which is the identity when its controls are $|0\rangle$.

1. **Drop the channel**: it is the identity
2. **Bit-flip pairs $\to$ Toffoli**, on each block of three wires
   `bit_flip_toffoli :`
   bit_flip_dec $\bullet$ bit_flip_enc $=$ `focus`$_{[1;2;0]}$ toffoli
3. **These Toffolis vanish**: wires $0, 3, 6$ carry an unknown superposition, so expand it over the basis (`decompose_scaler`) and compute on basis vectors (`focus_dpbasis`, `dpmerge_dpbasis`)
4. **Sign-flip pair**: $H \bullet H = \mathrm{id}$, then the same argument on wires $0, 3, 6$

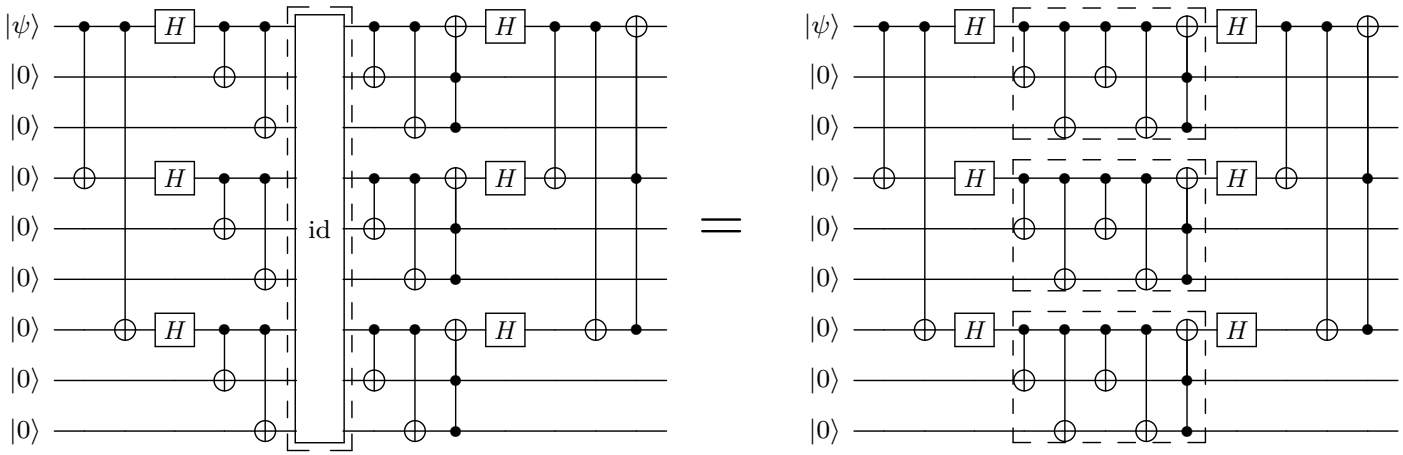
About 30 lines of proof, reusing $\sim$100 lines of lemmas on the bit-flip / sign-flip codes

The proof script (first part)

```
Lemma shor_code_id i :
  shor_code id |i,0,0,0,0,0,0,0,0⟩ = |i,0,0,0,0,0,0,0,0⟩.
Proof.
rewrite /=.
transitivity (focus [lens 0; 3; 6] (sign_flip_dec • sign_flip_enc)
                    |i,0,0,0,0,0,0,0,0⟩).
  rewrite focus_comp /= focus_dpbasis.
  set sfe := sign_flip_enc _ _.
  rewrite (decompose_scaler sfe) !linear_sum /=.
  apply: eq_bigr => t _.
  rewrite !linearZ_LR /= dpmerge_dpbasis.
  congr (_ *: focus _ sign_flip_dec _ _).
  case: t => -[|a [|b [|c []]]] Ht //=.
  by simpl_merge.
...
```

Steps 1–3: reduce the whole circuit to the sign-flip code on wires $0, 3, 6$ (the `transitivity` goal); the rest of the script does step 4.

The proof in pictures (1/4)

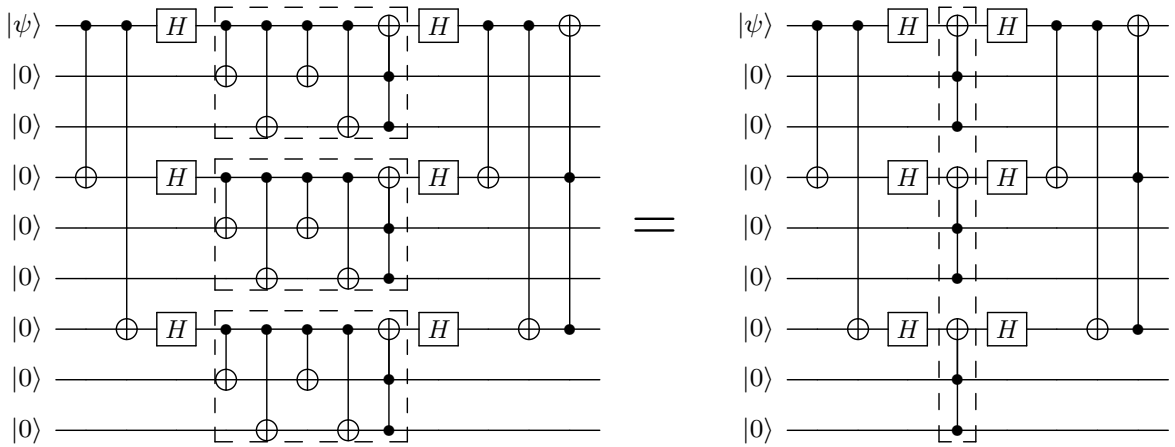


Step 1: drop the channel

The channel is the identity, so the encoder and decoder of each bit-flip code become adjacent.

Dashed boxes: the part rewritten in the next step.

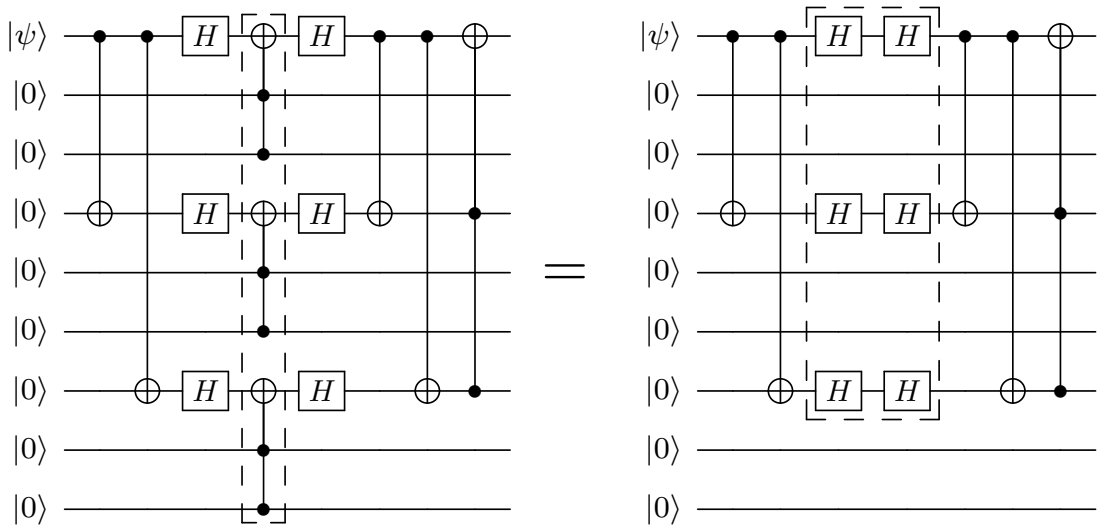
The proof in pictures (2/4)



Step 2: bit-flip pairs become Toffoli gates

On each block $\{0,1,2\}$, $\{3,4,5\}$, $\{6,7,8\}$:
bit_flip_dec • bit_flip_enc = $\texttt{focus}_{[1;2;0]}$ toffoli (`bit_flip_toffoli`, put in place with `focus_comp` and `focusM`).

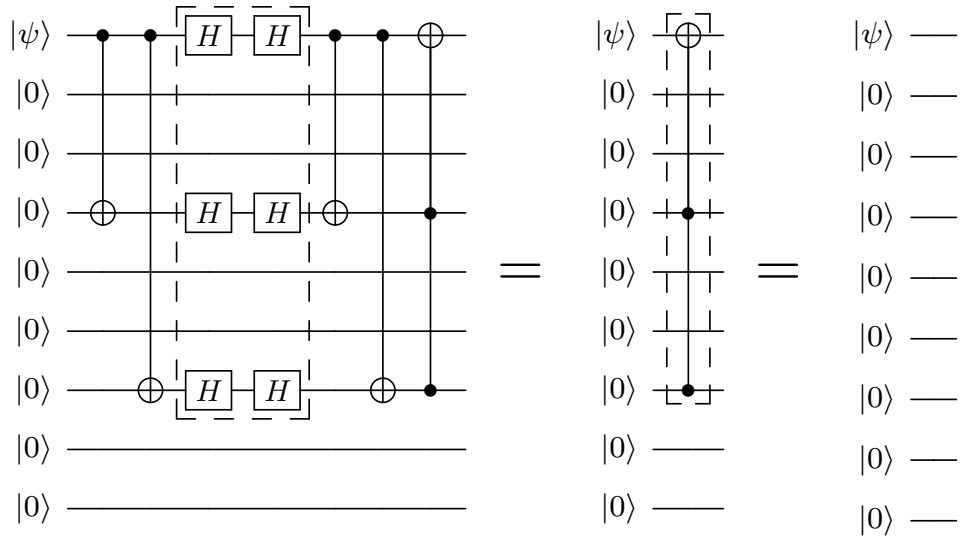
The proof in pictures (3/4)



Step 3: the Toffoli gates vanish

Their controls are $|0\rangle$, so each Toffoli is the identity. Its target (wire 0, 3 or 6) is in an unknown superposition: `decompose_scaler` expands it over the basis, `focus_dpbasis` / `dpmerge_dpbasis` compute on each basis vector, and linearity concludes.

The proof in pictures (4/4)



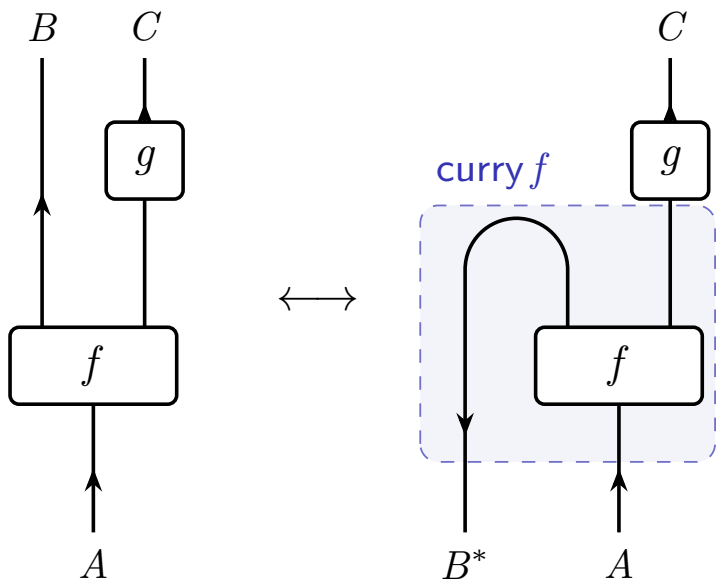
Step 4: the sign-flip pair

$H \bullet H = \mathrm{id}$, so on wires $0, 3, 6$ we again have bit_flip_dec $\bullet$ bit_flip_enc: a Toffoli gate with $|0\rangle$ controls, i.e. the identity. ■

Visualization

Outlook: monoidal currying

In a compact closed category (e.g. finite-dimensional Hilbert spaces) wires can be bent: $[A \to B \otimes C] \cong [B^* \otimes A \to C]$



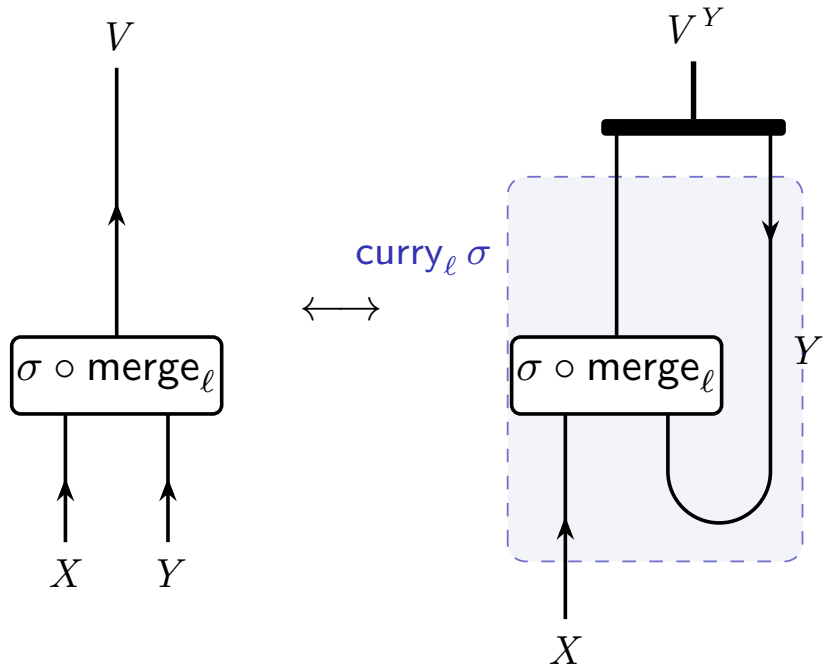
To apply g to *part* of the output of f: bend B away, post-compose, bend back,

$$(\mathrm{id}_B \otimes g) \circ f \;=\; \mathsf{uncurry}\,(g \circ \mathsf{curry}\, f),$$

the same shape as $\texttt{focus}_\ell\; G = \mathsf{uncurry}_\ell \circ G \circ \mathsf{curry}_\ell$.

Our `curry` is *cartesian* currying

A state is a function on basis labels, $\sigma : I^n \to V$, and $\mathsf{curry}_\ell\, \sigma\ v\ w := \sigma\,(\mathsf{merge}_\ell\ v\ w)$. With $X = I^m$ (wires in ℓ), $Y = I^{n-m}$ (the others): $\mathcal{S}\mathrm{et}(X \times Y,\ V) \cong \mathcal{S}\mathrm{et}(X,\ V^Y)$



$\mathcal{S}\mathrm{et}$ is cartesian closed but not compact closed: the bent Y wire is not a morphism by itself, so it is *clasped* to V; only the bundle V^Y is an object.

Cartesian vs. monoidal currying

The free vector space $K[X]$ links the two:

$$\mathcal{S}\mathrm{et}(X,\ V)\ \cong\ \mathrm{Vect}(K[X],\ V), \qquad K[X \times Y]\ \cong\ K[X] \otimes K[Y]$$

so cartesian currying of *labels* becomes monoidal currying of *state spaces*:

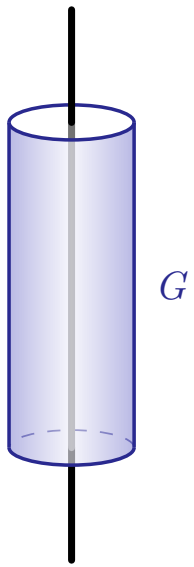
$$\mathrm{Vect}(K[X] \otimes K[Y],\ V)\ \cong\ \mathrm{Vect}(K[X],\ K[Y]^* \otimes V)$$

Dictionary

in the formalization	**in string diagrams**
$\mathrm{merge}_\ell : X \times Y \cong I^n$	wire permutation $\mathcal{H}^{\otimes n} \cong \mathcal{H}_\ell \otimes \mathcal{H}_{\bar{\ell}}$
curry_ℓ	bend the wires outside ℓ
G_W on $W^X,\ \ W = V^Y$	$M \otimes \mathrm{id}_W$
naturality of G	boxes on disjoint wires slide past each other
$\texttt{focus}_\ell\ G$	$\pi_\ell^{-1} \circ (M \otimes \mathrm{id}) \circ \pi_\ell$

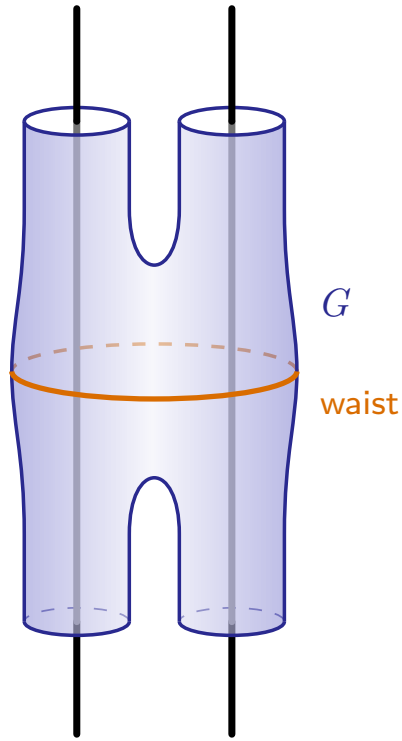
Polymorphism does the Kronecker padding for us: the untouched wires hide inside the coefficient type W.

Gates as surfaces (an idea)



unary gate: a hose

$(-)^2 \Rightarrow (-)^2$



binary gate: two pants glued at the waist

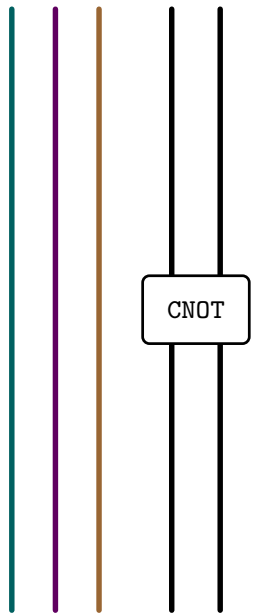
$(-)^4 \Rightarrow (-)^4$

A gate, as a natural transformation, is a thin-walled surface; a qubit is a wire threaded through it. Two qubits: two wires go in and come out.

Reading the picture

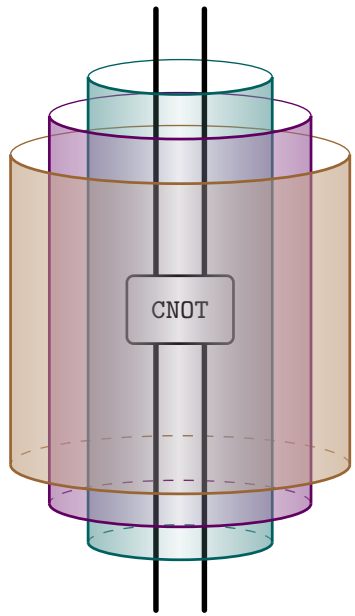
- composition $\bullet$: glue surfaces end to end
- $G_1 \otimes G_2$: two separate hoses; the shared waist is what lets the wires interact (e.g. `CNOT`)
- $\texttt{focus}_\ell$: thread only the wires in ℓ; the others pass outside
- thin walls: the surface holds nothing of its own, so G acts the same way for every $\mathbb{V}$ (naturality)

Idle wires as ambient cylinders



$\Longrightarrow$

each idle wire
becomes a cylinder

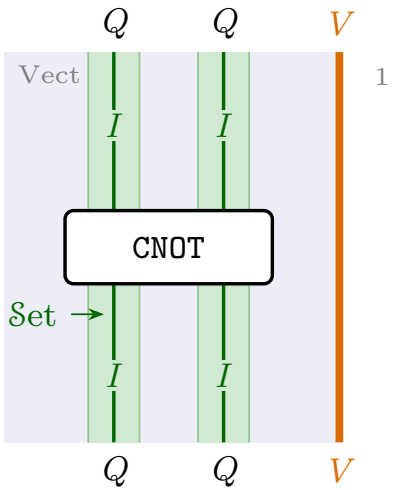


Leaving some room between the cylinders and the wires, the idle wires become ambient data around the gate: `CNOT` never touches them, so we can ignore them while handling the gate.

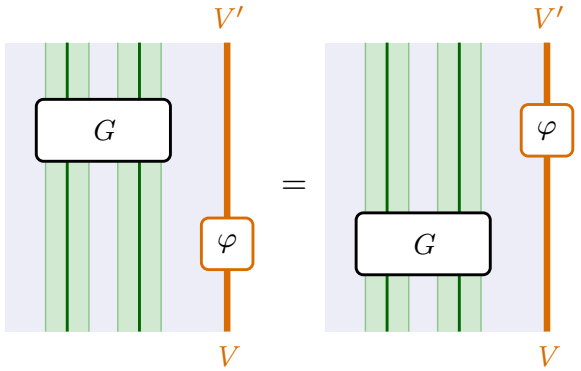
This is $\texttt{focus}_\ell\,\texttt{CNOT} = \text{uncurry}_\ell \circ \texttt{CNOT}_W \circ \text{curry}_\ell$ with $W = \mathbb{V}^{2^3}$: the cylinders are the coefficient type W, and naturality says $\texttt{CNOT}_W$ acts the same for every W.

Qubits as functors, gates as 2-cells

Regions = categories, wires = functors, nodes = natural transformations.



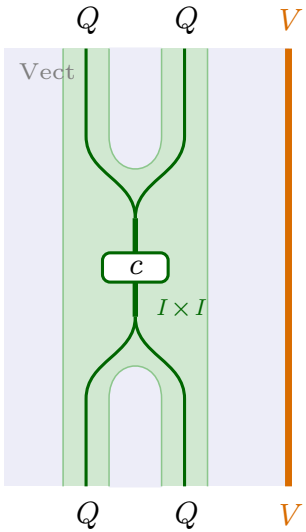
CNOT : $Q \circ Q \Rightarrow Q \circ Q$



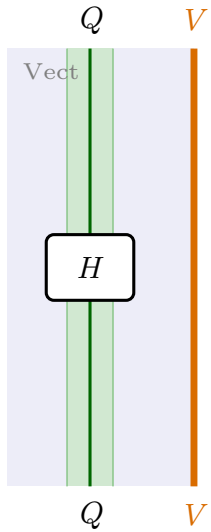
naturality: G and $\varphi : V \to V'$ slide past each other

- one qubit = a band: inside is the region $\mathcal{S}\mathrm{et}$ with the index set $I = \{0, 1\}$; the band is the functor $Q := (-)^I \cong K^2 \otimes -$ on Vect (a *functorial box* for $X \mapsto (-)^X$)
- n qubits $= Q \circ \cdots \circ Q = (-)^{I^n}$: $\times$ in $\mathcal{S}\mathrm{et}$ becomes bands side by side in Vect
- the coefficient V is a wire $1 \to \mathrm{Vect}$ on the right; a gate is a node on its own bands only (from now on a band is drawn as a plain wire Q)
- Yoneda (`naturalityP`): G is fixed by $G_K = M$; put $V := K$, erase the orange wire, and get the usual circuit

What the band borders mean



CNOT inside $\mathcal{S}\mathrm{et}$

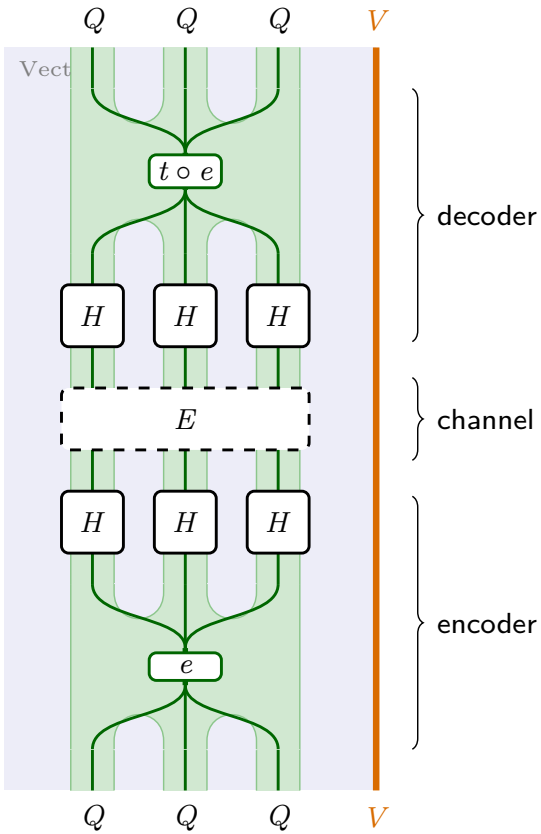


H in Vect

- **The border** applies the functor $X \mapsto (-)^X \cong K[X] \otimes -$ from $(\mathrm{FinSet}^{\mathrm{op}}, \times)$ to $(\mathrm{End}(\mathrm{Vect}), \circ)$ (a *functorial box*): crossing it turns an index set into a qubit wire.
- **Merging bands** (a pair of pants in the border): the functor is strong monoidal, $(-)^X \circ (-)^Y \cong (-)^{X \times Y}$, which is exactly cartesian currying.
- **Nodes inside** are maps of labels $f : X \to Y$, acting as $f^* : (-)^Y \Rightarrow (-)^X$: merge_ℓ, crossings, and classical gates ($\mathtt{CNOT} = c^*$, $c(i,j) = (i,\ i \oplus j)$; also X, SWAP, Toffoli).
- **Nodes outside** are genuinely linear: no map of labels induces H.

The border separates the combinatorics of wiring ($\mathcal{S}\mathrm{et}$) from the computational content of gates (Vect), which is what lenses are designed to do.

Example: the sign-flip code



- The classical parts become single $\mathcal{S}\mathrm{et}$ nodes, bijections of I^3 (shown by their action on basis labels):

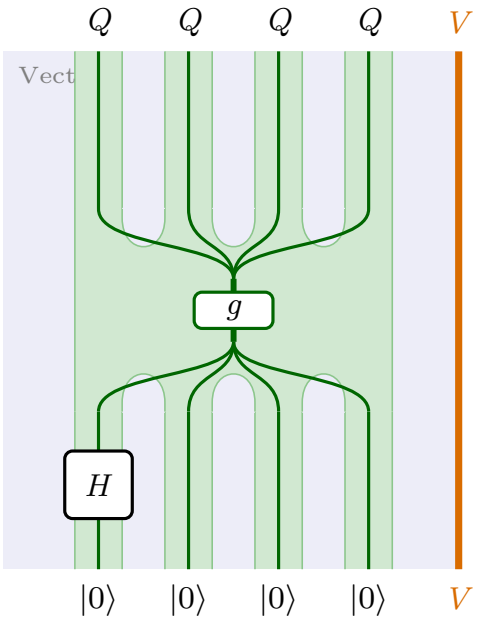
  $e(i,j,k) = (i,\ i \oplus j,\ i \oplus k)$
  $t(i,j,k) = (i \oplus jk,\ j,\ k)$
  (e: the two `CNOT`s, t: the Toffoli gate)

- Only H and the channel E are Vect nodes.

- With $E = \mathrm{id}$: $H \bullet H = \mathrm{id}$, the two $\mathcal{S}\mathrm{et}$ nodes meet in one band, and $t \circ e \circ e = t$, $t(i,0,0) = (i,0,0)$.

`bit_flip_toffoli` becomes a computation on 8 labels, inside $\mathcal{S}\mathrm{et}$.

Example: GHZ states for every n



$n = 4$

The usual circuit: H on wire 0, then a cascade of $n-1$ `CNOT`s. In bands, the whole cascade is one $\mathcal{S}\mathrm{et}$ node:

$$g(i, j_1, j_2, \dots) = (i,\ i \oplus j_1,\ i \oplus j_1 \oplus j_2,\ \dots)$$

The proof reads off the picture:

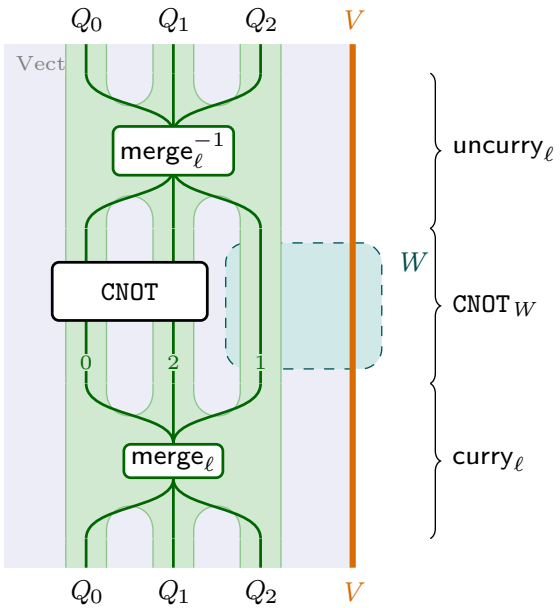
1. one Vect step:
   $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
2. one $\mathcal{S}\mathrm{et}$ step:
   $g(i, 0, \dots, 0) = (i, i, \dots, i)$

so the output is $\frac{1}{\sqrt{2}}(|0\cdots0\rangle + |1\cdots1\rangle)$, uniformly in n.

The Rocq proof `ghz_ok` is about 50 lines, roughly half the size of the corresponding SQIR proof.

Currying is regrouping

$$\texttt{focus}_\ell\,\texttt{CNOT} = \mathsf{uncurry}_\ell \circ \texttt{CNOT}_W \circ \mathsf{curry}_\ell, \qquad \ell = \{0 \mapsto 0,\ 1 \mapsto 2\}$$



- curry_ℓ happens entirely in $\mathcal{S}\mathrm{et}$: merge the bands, i.e. $(-)^I \circ (-)^I \circ (-)^I \cong (-)^{I^3}$ (cartesian currying); relabel by the bijection $\mathrm{merge}_\ell : I^2 \times I \cong I^3$; split into the bands of ℓ (wires $0, 2$) and the idle band (wire 1)
- nothing is bent: only merging, relabelling, splitting
- $\texttt{CNOT}_W$ is the only Vect node; $W = Q(V)$: the idle band joins the coefficient (the ambient cylinder of p. 31)
- polymorphism: a node ignores everything to its right

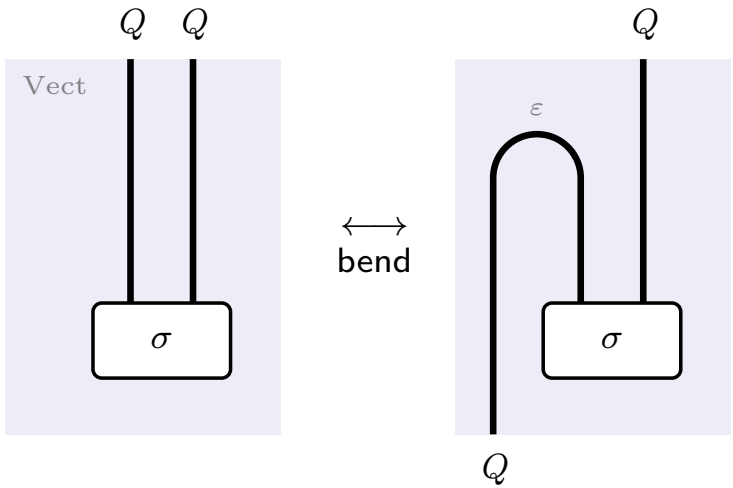
The lens is a $\mathcal{S}\mathrm{et}$ node, the gate a Vect node; `focus` composes them.

Three kinds of currying, revisited

	isomorphism	**effect on a state σ**
monoidal compact closed, p. 27	$\mathrm{Vect}(A,\, B \otimes C)$ $\cong \mathrm{Vect}(B^* \otimes A,\, C)$	becomes a map: *bend* a wire
cartesian, external p. 28	$\mathcal{S}\mathrm{et}(X \times Y,\, V)$ $\cong \mathcal{S}\mathrm{et}(X,\, V^Y)$	read as a function, V fixed: *bend* and clasp
cartesian, internal `curry`, p. 36	$(-)^{X \times Y} \cong (-)^X \circ (-)^Y$ natural in V	stays a state, V varies: *regroup* wires

The last two rows are the same exponential law $V^{X \times Y} \cong (V^Y)^X$, read with V fixed or with V varying. `curry` uses the second reading: it is natural in V, like the gates, so gates plug into curried states directly. Bending only appears when a state is turned into a map.

Bending, revisited



$\sigma : \mathrm{Id} \Rightarrow Q \circ Q$ (a two-qubit state) $\qquad$ $\hat{\sigma} : Q \Rightarrow Q$ (a one-qubit map)

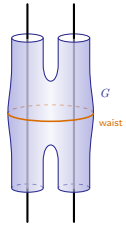
- I is finite, so $Q = (-)^I \cong \bigoplus_I (-)$ is its own adjoint: $Q \dashv Q$ gives caps ($\varepsilon : Q \circ Q \Rightarrow \mathrm{Id}$, a trace) and cups ($\eta : \mathrm{Id} \Rightarrow Q \circ Q$, $\sum_i |ii\rangle$)
- so the 2-categorical picture also has the wire bending of p. 27: it is map–state duality on qubit wires
- the clasp picture of p. 28 is the same bend with V fixed and σ read as a function on labels
- `focus` never needs it: a gate acts on a state by crossing and regrouping wires only (p. 36)

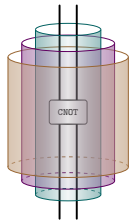
Dictionary, revisited

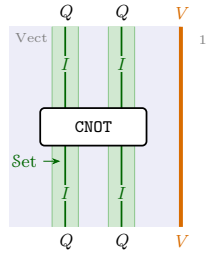
in the formalization	**in the 2-categorical picture**
index set I (in $\mathcal{S}\mathrm{et}$)	inside a band
V (any K-module)	coefficient wire on the right
$(-)^{I^n}$	n wires side by side
merge_ℓ	crossing of wires (a natural iso)
classical gates (e.g. `CNOT`)	nodes inside a band
curry_ℓ, $\mathrm{uncurry}_\ell$	crossing + regrouping, no bending
gate $G : \forall V,\ V^{I^m} \to V^{I^m}$	node on its m wires
naturality of G	interchange: G, φ slide past each other
`naturalityP` $(G \leftrightarrow M)$	Yoneda: put $V := K$
$G_W,\ \ W = V^{I^{n-m}}$	whiskering: idle wires join V
$\mathtt{focus}_\ell\, G$	cross, apply the node, uncross
states as maps	bending with $Q \dashv Q$

Polymorphism does the Kronecker padding for us: G_W is G whiskered by the idle wires.

Three pictures compared







	surfaces (p. 30)	cylinders (p. 31)	2-categorical (p. 32)
qubit	wire in a hose	wire	band $Q = (-)^I$ around I
gate	surface	box	node: in a band if classical ($\mathcal{S}\mathrm{et}$), outside if linear (Vect)
idle wires	not drawn	cylinders	wires on the right, joined with V into W
coefficient	implicit	implicit	the orange wire V
currying	—	wire $\mapsto$ cylinder	merging / splitting bands
status	intuition	intuition	equations (interchange, Yoneda)

The 2-categorical picture keeps both intuitions: the *waist* of the pants is a merged band $I \times I$, where an entangling classical gate such as `CNOT` lives; the *cylinders* are the region W to the right of the gate, though in $Q^m(W)$ the idle wires sit *inside*, next to the coefficients, rather than around the gate.

Future work

- Shor's code over *erroneous* channels
- An algebraic theory of lenses
- Relating `focus` to the categorical theory of *optics*
- String-diagrammatic reasoning on top of the lens-based semantics