Denotational semantics for stabiliser quantum programs

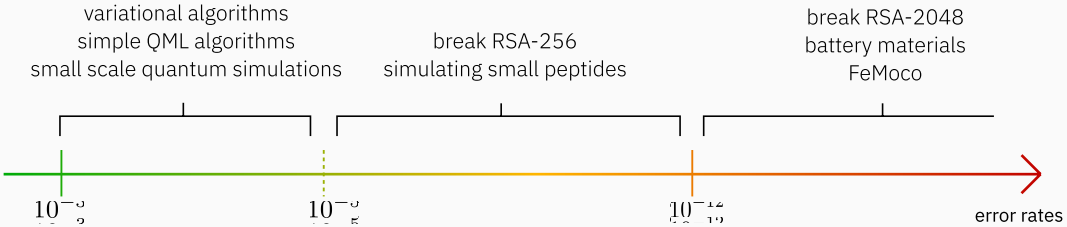
Robert I. Booth

Joint with: Cole Comfort

FSCD, 23 July 2026

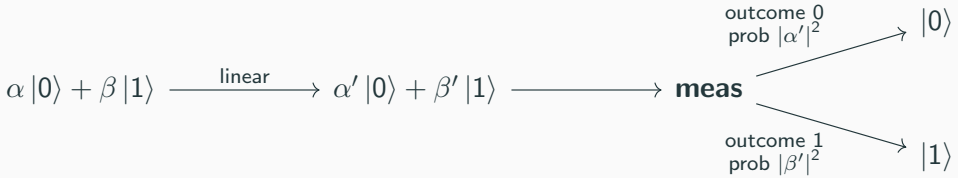
Noise in quantum hardware

error probability:	
transistor gates	qubits
0.000000000000001	0.001

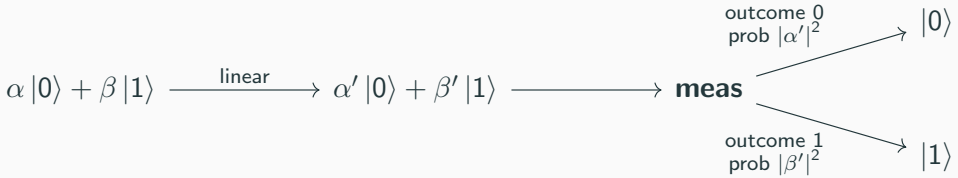


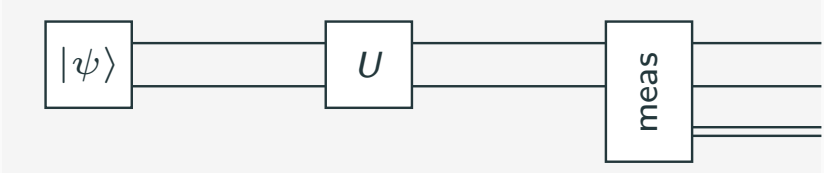
Credit: Tamas Noszko, Adithya Sireesh

One-slide quantum computing



One-slide quantum computing





Repetition code

codewords: $0 \mapsto 0000$ $1 \mapsto 1111$

Quantum error-correcting codes

Repetition code = invariance under permutations

codewords: $0 \mapsto 0000$ $1 \mapsto 1111$

Quantum repetition code

codewords: $|0\rangle \mapsto |0000\rangle$ $|1\rangle \mapsto |1111\rangle$

Quantum repetition code

codewords: $|0\rangle \mapsto |0000\rangle \qquad |1\rangle \mapsto |1111\rangle$

codevector: $\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0000\rangle + \beta|1111\rangle$

Quantum error-correcting codes

Quantum repetition code $=$ vectors invariant under a group representation

codewords: $|0\rangle \mapsto |0000\rangle \qquad |1\rangle \mapsto |1111\rangle$

codevector: $\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0000\rangle + \beta|1111\rangle$

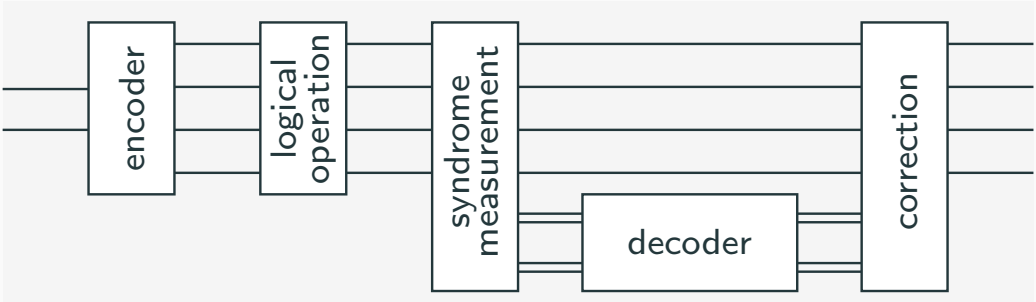
Quantum error-correcting codes

Quantum repetition code $=$ vectors invariant under a group representation

$$\begin{aligned} \text{codewords:} \quad & |0\rangle \mapsto |0000\rangle \qquad |1\rangle \mapsto |1111\rangle \\ \text{codevector:} \quad & \alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0000\rangle + \beta|1111\rangle \end{aligned}$$

For QEC, more complicated error types $\implies$ more intricate codes $\implies$ larger symmetry groups

Fault-tolerant quantum computing



The fault-tolerant compilation problem

There really two new correctness criteria here:

The fault-tolerant compilation problem

There really two new correctness criteria here:

1. logical correctness, i.e. the compiled program actually manipulates the logical data in the way we expect;

The fault-tolerant compilation problem

There really two new correctness criteria here:

1. logical correctness, i.e. the compiled program actually manipulates the logical data in the way we expect;
2. fault robustness, i.e. the compiled program actually deals with noise correctly.

Stabiliser Programming Language

More pragmatically, we want to be able to reason about a simple programming language for stabiliser quantum mechanics:

$$c, d \ ::= c \fatsemi d \mid \overbrace{\mathbf{init}\ \underline{x} \mid \underline{\mathbf{y}} = A * \underline{\mathbf{x}} \mid \mathbf{disc}\ \underline{x}}^{\textit{classical}} \mid \overbrace{\mathbf{qinit}\ \underline{x} \mid \underline{\mathbf{x}} *= U}^{\textit{quantum}}$$
$$\mid \underbrace{\mathbf{meas}\ \underline{x}}_{\textit{measurement}} \mid \underbrace{\mathbf{ctrl}_P\ \underline{x}\ \underline{y}}_{\textit{classical control}}$$

Stabiliser Programming Language

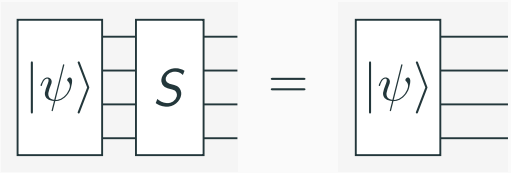
More pragmatically, we want to be able to reason about a simple programming language for stabiliser quantum mechanics:

$$c, d \ ::= \ c \fatsemi d \mid \overbrace{\mathbf{init}\ \underline{x} \mid \underline{\mathbf{y}} = A * \underline{\mathbf{x}} \mid \mathbf{disc}\ \underline{x}}^{\textit{classical}} \mid \overbrace{\mathbf{qinit}\ \underline{x} \mid \underline{\mathbf{x}} *= U}^{\textit{quantum}}$$
$$\mid \underbrace{\mathbf{meas}\ \underline{x}}_{\textit{measurement}} \mid \underbrace{\mathbf{ctrl}_P\ \underline{x}\ \underline{y}}_{\textit{classical control}}$$

We give denotational semantics tailored to fault-tolerant quantum computing, to support the development of practical tools for verifying and guaranteeing these correctness criteria.

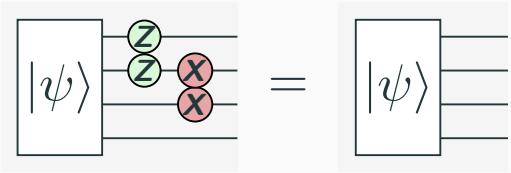
Stabiliser codes

Stabiliser codes are special QEC codes whose symmetries are *locally generated*:



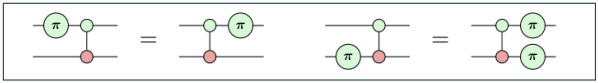
Stabiliser codes

Stabiliser codes are special QEC codes whose symmetries are *locally generated*:

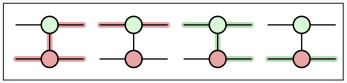


Pauli pushing in all its forms

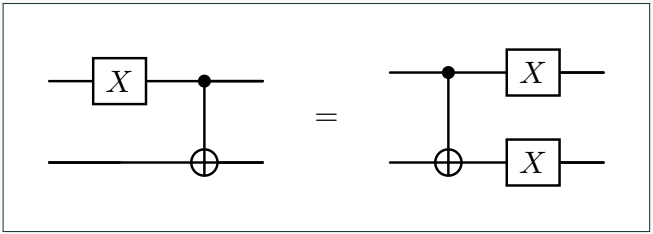
Pauli pushing



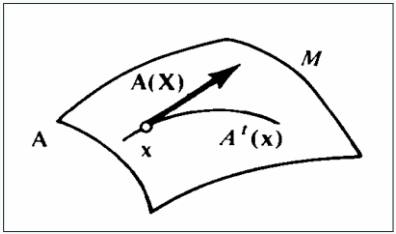
Pauli webs = stabiliser flows



Pauli propagation

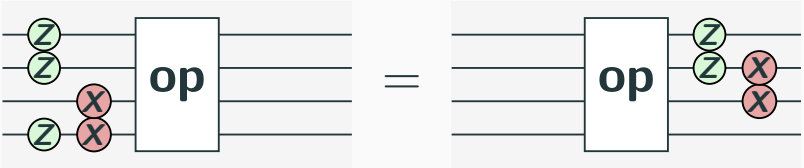


Lagrangian relations



Stabiliser maps

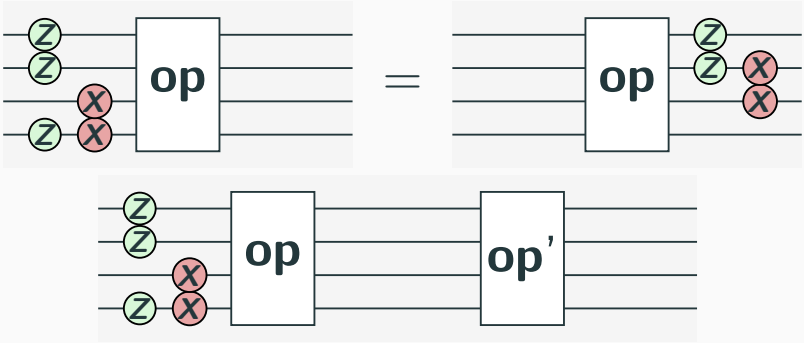
We can define a subcategory of FdHilb of linear maps that push/propagate tokens:



Cole Comfort and Aleks Kissinger. **"A Graphical Calculus for Lagrangian Relations".** In: (visited on 06/19/2023).

Stabiliser maps

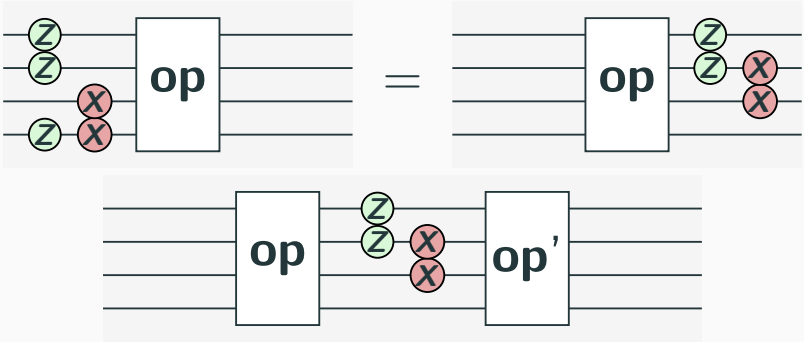
We can define a subcategory of FdHilb of linear maps that push/propagate tokens:



Cole Comfort and Aleks Kissinger. **"A Graphical Calculus for Lagrangian Relations".** In: (visited on 06/19/2023).

Stabiliser maps

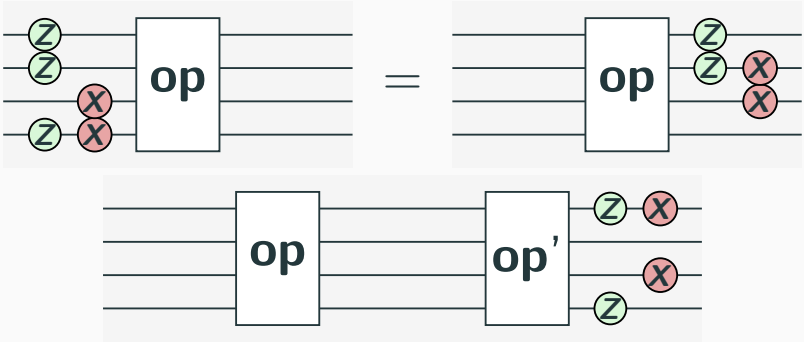
We can define a subcategory of FdHilb of linear maps that push/propagate tokens:



Cole Comfort and Aleks Kissinger. **"A Graphical Calculus for Lagrangian Relations".** In: (visited on 06/19/2023).

Stabiliser maps

We can define a subcategory of FdHilb of linear maps that push/propagate tokens:



Cole Comfort and Aleks Kissinger. **"A Graphical Calculus for Lagrangian Relations".** In: (visited on 06/19/2023).

Semantics overview

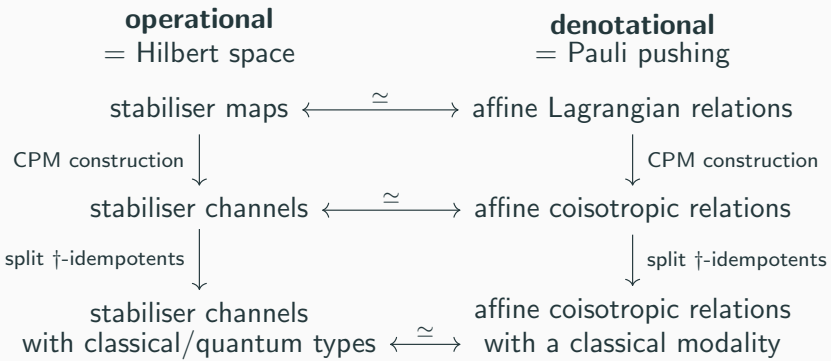
linear maps $\xrightarrow{\text{CPM construction}}$ quantum channels $\xrightarrow{\text{split †-idempotents}}$ QM with *classical types*

$$\text{FdHilb} \longrightarrow \text{MatAlg}_{CP} \longrightarrow \text{FdC}^*\text{-Alg}_{CP}$$

Peter Selinger. **"Dagger Compact Closed Categories and Completely Positive Maps: (Extended Abstract)".** In: ().

Peter Selinger. **"Idempotents in Dagger Categories".** In: ().

Semantics for measurement and classical control



CPM: Semantics of trace

CPM: Semantics of trace

$\rightsquigarrow$ the trace is just a discard!

Splitting: Semantics of measurements

Splitting: Semantics of measurements

$\rightsquigarrow$ relates the 2-dimensional space of red/green tokens to the 1-dimensional space of bit-flips

Summary of results

Full definability

For all typed environments Γ and Δ and morphisms $f : [\![\Gamma]\!] \to [\![\Delta]\!]$ in $\mathsf{AR}_{\mathsf{qc}}$, there exists a well-formed judgement $\Gamma \vdash c \triangleright \Delta$ such that

Full abstraction

Well-formed judgements c and d are observationally equivalent if and only if their denotations are equal $[\![c]\!] = [\![d]\!]$.

Ongoing and future work

- How do we recover this semantics for the qubit case?

Ongoing and future work

- How do we recover this semantics for the qubit case?
- How do we go beyond affine classical control?

Ongoing and future work

- How do we recover this semantics for the qubit case?
- How do we go beyond affine classical control?
- How can we use these semantics for fault-tolerant compilation?

Ongoing and future work

- How do we recover this semantics for the qubit case?
- How do we go beyond affine classical control?
- How can we use these semantics for fault-tolerant compilation? What is the compositional / algebraic / combinatorial structure of fault-tolerance?

The end

The symplectic rosetta stone

$$\gamma : \mathbb{F}_2^n \oplus \mathbb{F}_2^n \longrightarrow \mathcal{P}_n$$

$$(x|z) \longmapsto \prod_{j=1}^{n} X_j^{x_j} Z_j^{z_j} \qquad \text{so that} \qquad \gamma(s|t)\gamma(x|z) = e^{i\pi s^T z}\gamma(s+x|t+z)$$

Stabiliser	**Symplectic**
Product: $\gamma(s\|t)\,\gamma(x\|z)$	Sum: $(s+x \mid t+z)$
Commuting Paulis	$s^T z + t^T x = 0$
Subgroup of $\mathcal{P}_n$	Subspace of $\mathbb{F}_2^n \oplus \mathbb{F}_2^n$
Abelian subgroup	Isotropic subspace
Maximal abelian subgroup	Lagrangian subspace
Signs	Subspace character: $\langle v, \cdot \rangle$ (affine part)