Control, complete, compute:
Rig categories in quantum computing

Chris Heunen  University of Edinburgh

Louis Lemonnier  University of Edinburgh

Jacques Carette  McMaster University

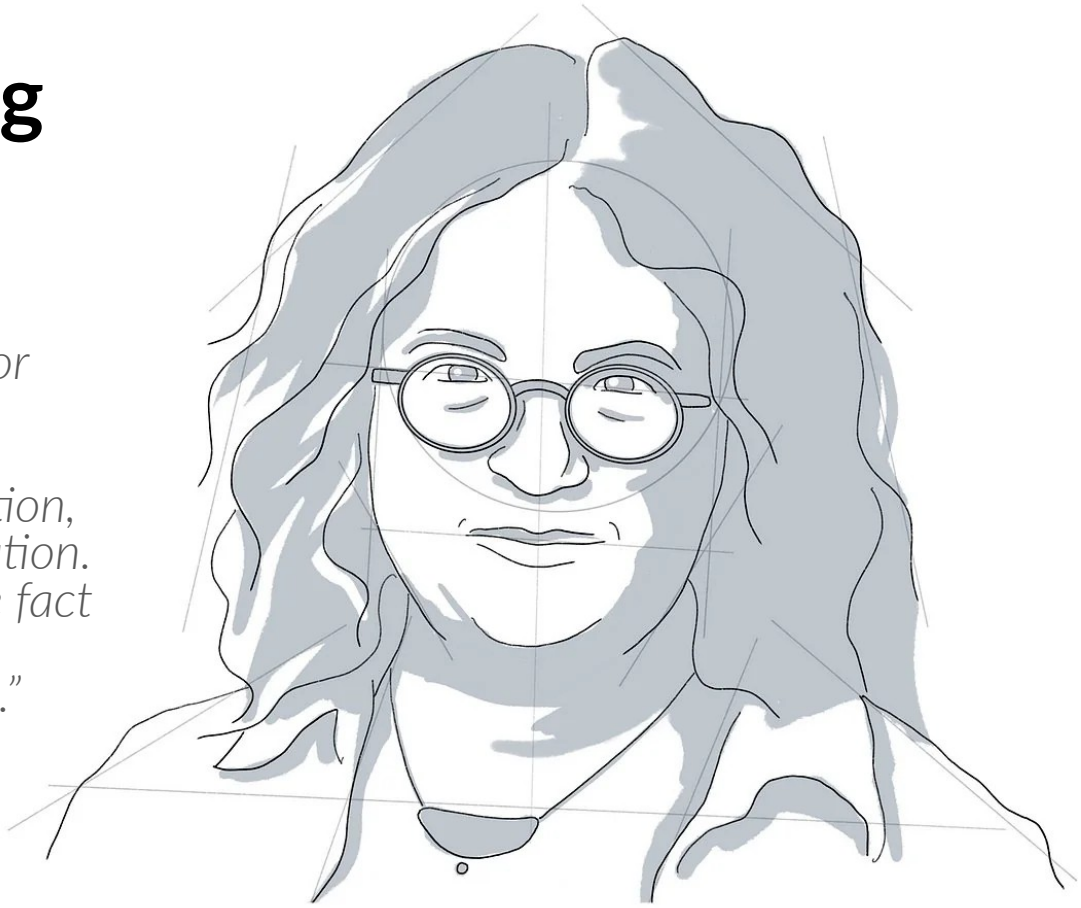
Robin Kaarsgaard  University of Southern Denmark

Neil Julien Ross  Dalhousie University

Amr Sabry  Indiana University Bloomington

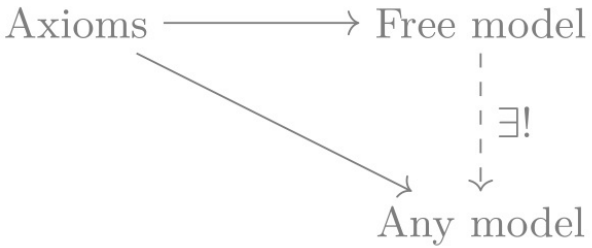
Quantum computing as a completion

“It’s really something that is special for quantum computation because it’s somehow ‘complete’ — quantum computation is some kind of completion, mathematically, of classical computation. I think of this as maybe similar to the fact that the complex numbers are an algebraic closure of the real numbers.”



Universality

Characterise property up to isomorphism by behaviour rather than construction



PNAS RESEARCH ARTICLE | COMPUTER SCIENCES OPEN ACCESS

Free quantum computing

Jacques Carette[a,1], Chris Heunen[b,1,2], Robin Kaarsgaard[c,1], Neil J. Ross[d,1], and Amr Sabry[a,1]

Edited by Peter Shor, Massachusetts Institute of Technology, Cambridge, MA; received May 1, 2025; accepted January 8, 2026

Quantum computing improves substantially on known classical algorithms for various important problems, but the nature of the relationship between quantum and classical computing is not yet fully understood. This relationship can be clarified by free models, that add to classical computing just enough physical principles to represent quantum computing and no more. Here, we develop an axiomatization of quantum computing that replaces the standard continuous postulates with a small number of discrete equations, as well as a free model that replaces the standard linear-algebraic model with a category-theoretical one. The axioms and model are based on reversible classical computing, isolate quantum advantage in the ability to take certain well-behaved square roots, and link to various quantum computing hardware platforms. This approach allows combinatorial optimization, including brute force computer search, to optimize quantum computations. The free model may be interpreted as a programming language for quantum computers, that has the same expressivity and computational universality as the standard model, but additionally allows automated verification and reasoning.

reversible computing | axiomatization | free model | category theory

Significance

Quantum computing holds great promise, but its foundations and the source of its advantages remain conceptually obscure. We develop a framework that contains no extraneous mathematical assumptions and clearly separates what is truly quantum from what is just classical computing in disguise. Instead of relying on the infinite precision of continuous complex numbers, this symbolic approach uses a small finite number of discrete building blocks that reflect physical implementations. Unlike traditional models, this model supports purely symbolic combinatorial reasoning, enabling the use of powerful classical computer science techniques. This framework is just as effective as traditional ones, and offers a rigorous, simpler foundation to understand and engineer quantum computation.

Quantum computing improves substantially on known classical algorithms for certain problems (1) including prime factorization (2), boson sampling (3), and Hamiltonian simulation (4). The nature of the relationship between classical and quantum computing is not yet fully understood. The literature only identifies several notions that do not explain the difference on their own (5), including superposition (6), entanglement (7), nonlocality (8), contextuality (9), and interference (10). The advantage of quantum computing over classical computing is often considered quantitatively: just how much advantage does a specific quantum algorithm have over classical ones (11)? Here, we consider it qualitatively: does a model of computation allow algorithms that have advantage over classical ones, or does it not?

The question is difficult partly because quantum computing is often implicitly confined to a standard model, that uses complex linear maps (12). For a meaningful comparison, both classical and quantum computing must be situated within a larger landscape of models. We use as models bipermutative categories, the most permissive notion of model of computation possible. Within this framework, we identify a unified free model of quantum computing that can express all quantum algorithms with advantage over classical ones.

To explain the free property of a model by way of example, consider combinatorics. The natural numbers can be completed with negatives in multiple ways: $\mathbb{N}$ embeds into $\mathbb{Z}$ while preserving addition as $n \mapsto n$, but also as $n \mapsto -n$ or $n \mapsto 2n$. The fact that these are completions with negatives means that an additive inverse for the number 1 is adjoined, and entails that all these embeddings are in fact the same up to an unimportant global scalar. The fact that it is the free completion means that any other completion has to encompass it. The real numbers also contain $\mathbb{N}$ and allow subtraction, but additionally have many other properties that are superfluous for counting. Any addition-preserving embedding $\mathbb{N} \to \mathbb{R}$ factors uniquely through the free model $\mathbb{Z}$ (of natural numbers with negatives). In this sense, the integers are the smallest model, devoid of extraneous assumptions or constraints, of natural numbers with negatives. This is exactly what being a free model captures. See Fig. 1. Generally, free models are unique: they are completely determined by their defining properties (such as having negatives) up to a unique isomorphism (such as an unimportant global scalar).

This article explains the relationship between classical and quantum computing in a similar way. Like adjoining -1 to the natural numbers, we adjoin a small number of generators with physical significance to reversible classical computing and show that reversible quantum computing arises as the free completion. Like the real numbers, any other model of reversible quantum computing, including the standard model of complex unitary matrices, must factor uniquely through this free one. Conversely,

Author affiliations: [a]Department of Computing and Software, McMaster University, Hamilton, ON L8S 4K1, Canada; [b]School of Informatics, University of Edinburgh, Edinburgh EH9 1DF, United Kingdom; [c]Department of Mathematics and Computer Science, Centre for Quantum Mathematics, Centre for Formal Methods and Future Computing, University of Southern Denmark, Odense 5230, Denmark; [d]Department of Mathematics and Statistics, Dalhousie University, Halifax, NS B3H 4R2, Canada; and [e]Department of Computer Science, Indiana University, Bloomington, IN 47408

Author contributions: J.C., C.H., R.K., N.J.R., and A.S. designed research; performed research; contributed new reagents/analytic tools; analyzed data; and wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission.

[1]J.C., C.H., R.K., N.J.R., and A.S. contributed equally to this work.

[2]To whom correspondence may be addressed. Email: chris.heunen@ed.ac.uk.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2510881123/-/DCSupplemental.

Published February 17, 2026.

One rig to control them all

Chris Heunen
University of Edinburgh, United Kingdom

Robin Kaarsgaard
University of Southern Denmark, Denmark

Louis Lemonnier
University of Edinburgh, United Kingdom

Abstract

Controlled commands—computations whose execution depends on a separate input—play a central role in reversible Boolean circuits and quantum circuits. However, existing formalisms typically treat control only implicitly, entangled with other aspects of computation. From a semantic perspective, control is most naturally expressed in semisimple rig categories, which—unlike standard circuit models such as props—support both parallel and conditional composition.

We present a construction that freely adjoins an explicit syntactic notion of control to a circuit theory specified as a suitable prop, subject to eight universally quantified equations. Our main result is that these equations are sound and complete for the intended semantics of control: the resulting theory satisfies a universal property, identifying it exactly as the circuit subtheory of the free semisimple rig completion. The proof combines coherence for rig categories with a new method based on induction over Gray codes.

We illustrate the usefulness of the framework by showing that it simplifies several existing sound and complete axiomatisations of quantum circuits, isolating a small and conceptually clean set of generators and equations. In addition, the same equations yield a sound and complete axiomatisation of the multiply controlled Toffoli gate set, that is universal for reversible Boolean circuits.

2012 ACM Subject Classification Theory of computation → Control primitives; Theory of computation → Categorical semantics; Theory of computation → Quantum computation theory

Keywords and phrases Quantum control, rig categories, complete equational theories

Digital Object Identifier 10.4230/LIPIcs.LICS.2026.3

Acknowledgements The authors would like to thank the members of the quantum programming group, specifically Wang Fang, and of the category theory group at Edinburgh for their support and feedback. We extend our thanks to Robert Booth, Kostia Chardonnet, Noé Delorme, Nicolas Heurtel, Simon Perdrix, and Peter Selinger for comments and fruitful discussions that helped us improve the paper. We also thank the anonymous reviewers for their comments and suggestions. This research was funded by the Engineering and Physical Sciences Research Council (EPSRC) under project EP/X025551/1 "Rubber DUQ: Flexible Dynamic Universal Quantum programming". Robin Kaarsgaard was supported by Sapere Aude: DFF-Research Leader grant 5251-00024B.

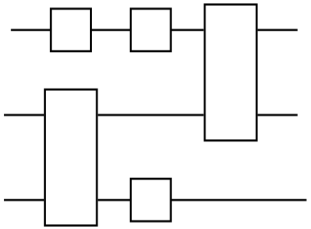
1 Introduction

Many computations contain *controlled* commands, that is, commands that are executed depending on the value of some memory cell. By *control* we mean the aspects of a computation that govern these dependencies.[1] Typically, the controlled command acts on one part of memory, and the controlling memory cell resides in another part of memory. To be more precise, for example, consider controlled negation in reversible Boolean circuits: the target

[1] We do not mean 'control theory' as used in e.g. engineering [22].

41st Annual Symposium on Logic in Computer Science (LICS 2026).
Editors: Claudia Faggian and Joost-Pieter Katoen; Article No. 3; pp. 3:1–3:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Circuits



Circuits

morphism in monoidal category (**C**,⊕,0) =
string diagram with wires Ob(**C**) and gates Mor(**C**)

Boolean circuits
Bool / **Perm**

Wires are 2={0,1} / **n**
Gates are bijections
Parallel by cartesian product

$$2 \xrightarrow{X} 2 \qquad 2 \times 2 \xrightarrow{CX} 2 \times 2$$
$$0 \mapsto 1 \qquad 00 \mapsto 00$$
$$1 \mapsto 0 \qquad 01 \mapsto 01$$
$$10 \mapsto 11$$
$$11 \mapsto 10$$

Quantum circuits
Unitary

Wires are $\mathbb{C}^2$ / $\mathbb{C}^n$
Gates are unitary matrices
Parallel by tensor product

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{pmatrix}$$

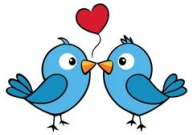
Presentations

- *PROP* is a strict symmetric monoidal category $(\mathbf{C}, \oplus, 0)$ where objects are $n \in \mathbb{N}$ and $m \oplus n = m + n$

- It is *presented* by *generators* $\{g_i \in \mathbf{C}\}_i$

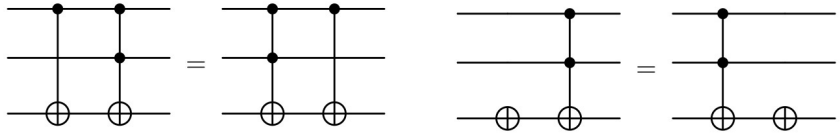
  and *relations* {equations}

  if any $f \in \mathbf{C}$ is a circuit of g_i's and two circuits represent the same morphism iff they are equal by equations

Fundamental theorem of reversible computing:

- **Bool** = < X, CX, C^nX
  | $X^2 = 1$, $CX^2 = 1$, $C^nX^2 = 1$, ... >



- Full & faithful / Sound & complete:
  f=g in **Bool** ⇔ f=g by equations

What about quantum circuits?

Unitary $\supseteq$ $\langle$ X, CX, CCX, H, T $\mid$ $X^2 = 1$, $CX^2 = 1$, $CCX^2 = 1$, … $\rangle$

$\supseteq$ $\langle$ X, CX, CCX, H $\mid$ $X^2 = 1$, $CX^2 = 1$, $CCX^2 = 1$, … $\rangle$ dense

But syntactic relations unknown or unwieldy

Controlled Circuits

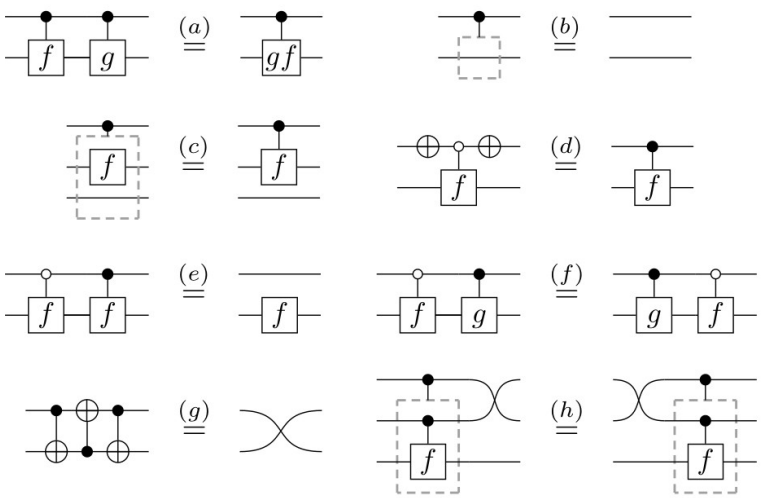
Crops

- *Controllable PROP (CROP)* is a PROP (**P**,⊕,0) of endomorphisms with fixed x : 1 → 1

- Given **P**, build **cP** by generators and relations

  f : n → n from **P**

  C^b f : 1+n → 1+n for f from **P**



Fundamental theorem of reversible computing revisited:

$$\mathbf{Bool} = \mathbf{c}\langle\ X : 1 \to 1 \ \mid X^2 = 1\ \rangle$$

Similar result for **Unitary**?

Matrices

“Eighty percent of mathematics is linear algebra”

*Both **Perm** and **Unitary** have more structure than ⊗*

Rig category: Categorification of natural numbers

$$\frac{f\colon A \to B \qquad g\colon B \to C}{g \circ f\colon A \to C}$$

$$\frac{f\colon A \to B \qquad f'\colon A' \to B'}{f \otimes f'\colon A \otimes A' \to B \otimes B'}$$

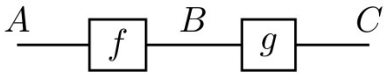
$$\frac{f\colon A \to B \qquad f'\colon A' \to B'}{f \oplus f'\colon A \oplus A' \to B \oplus B'}$$

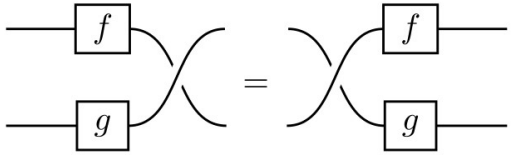
$$\frac{}{\mathrm{id}\colon A \to A}$$

$$(h \circ g) \circ f = h \circ (g \circ f)$$
$$\mathrm{id} \circ f = f = f \circ \mathrm{id}$$

$$(A \otimes B) \otimes C \simeq A \otimes (B \otimes C)$$
$$I \otimes A \simeq A \simeq A \otimes I$$
$$A \otimes B \simeq B \otimes A$$

$$(A \oplus B) \oplus C \simeq A \oplus (B \oplus C)$$
$$O \oplus A \simeq A \simeq A \oplus O$$
$$A \oplus B \simeq B \oplus A$$





$$A \otimes (B \oplus C) \simeq (A \otimes B) \oplus (A \otimes C)$$
$$A \otimes O \simeq O$$

Bipermutative category: Strict version

Permutations and rig categories

- **Perm** is the initial rig category

- **Unitary** is a rig category with ⊕ direct sum

$$f \oplus g = \begin{pmatrix} f & 0 \\ 0 & g \end{pmatrix}$$

- Classical reversible computing sits inside quantum reversible computing via permutation matrices: there is a unique rig functor **Perm** → **Unitary**

- Note: ⊕ is not a biproduct

$$A \underset{\binom{1}{0}}{\overset{(1\ 0)}{\rightleftarrows}} A \oplus B \underset{\binom{0}{1}}{\overset{(0\ 1)}{\rightleftarrows}} B$$

Programming with Permutations

$$
\begin{aligned}
b &::= \mathbb{0} \mid \mathbb{1} \mid b + b \mid b \times b && \text{(value types)}\\
t &::= b \leftrightarrow b && \text{(combinator types)}\\
iso &::= id \mid swap^{+} \mid assocr^{+} \mid assocl^{+} \mid unite^{+}l \mid uniti^{+}l \mid absorbl \mid factorzr && \text{(isomorphisms)}\\
&\quad \mid swap^{\times} \mid assocr^{\times} \mid assocl^{\times} \mid unite^{\times}l \mid uniti^{\times}l \mid dist \mid factor\\
c &::= iso \mid c \fatsemi c \mid c + c \mid c \times c && \text{(combinators)}
\end{aligned}
$$

$$
\begin{array}{rcrcll}
id & : & b & \leftrightarrow & b & : id\\
swap^{+} & : & b_1 + b_2 & \leftrightarrow & b_2 + b_1 & : swap^{+}\\
assocr^{+} & : & (b_1 + b_2) + b_3 & \leftrightarrow & b_1 + (b_2 + b_3) & : assocl^{+}\\
unite^{+}l & : & \mathbb{0} + b & \leftrightarrow & b & : uniti^{+}l\\
swap^{\times} & : & b_1 \times b_2 & \leftrightarrow & b_2 \times b_1 & : swap^{\times}\\
assocr^{\times} & : & (b_1 \times b_2) \times b_3 & \leftrightarrow & b_1 \times (b_2 \times b_3) & : assocl^{\times}\\
unite^{\times}l & : & \mathbb{1} \times b & \leftrightarrow & b & : uniti^{\times}l\\
dist & : & (b_1 + b_2) \times b_3 & \leftrightarrow & (b_1 \times b_3) + (b_2 \times b_3) & : factor\\
absorbl & : & b \times \mathbb{0} & \leftrightarrow & \mathbb{0} & : factorzr
\end{array}
$$

$$
\frac{c_1 : b_1 \leftrightarrow b_2 \quad c_2 : b_2 \leftrightarrow b_3}{c_1 \fatsemi c_2 : b_1 \leftrightarrow b_3}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 + c_2 : b_1 + b_2 \leftrightarrow b_3 + b_4}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 \times c_2 : b_1 \times b_2 \leftrightarrow b_3 \times b_4}
$$

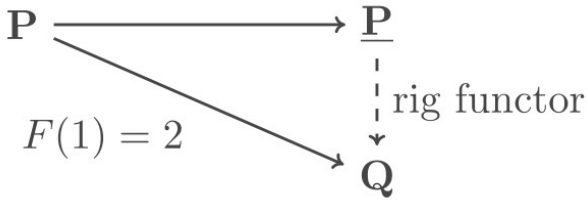
$$
\begin{aligned}
\textsc{ctrl}\ c &= dist \fatsemi id + (id \times c) \fatsemi factor\\
1 : \mathbb{1} \leftrightarrow \mathbb{1} &= id\\
\textsc{x} : \mathbb{2} \leftrightarrow \mathbb{2} &= swap^{+}\\
\textsc{cx} : \mathbb{2} \times \mathbb{2} \leftrightarrow \mathbb{2} \times \mathbb{2} &= \textsc{ctrl}\ swap^{+}\\
\textsc{ccx} : \mathbb{2} \times \mathbb{2} \times \mathbb{2} \leftrightarrow \mathbb{2} \times \mathbb{2} \times \mathbb{2} &= \textsc{ctrl}\ \textsc{cx}
\end{aligned}
$$

Matrix Completion

Kronecker product

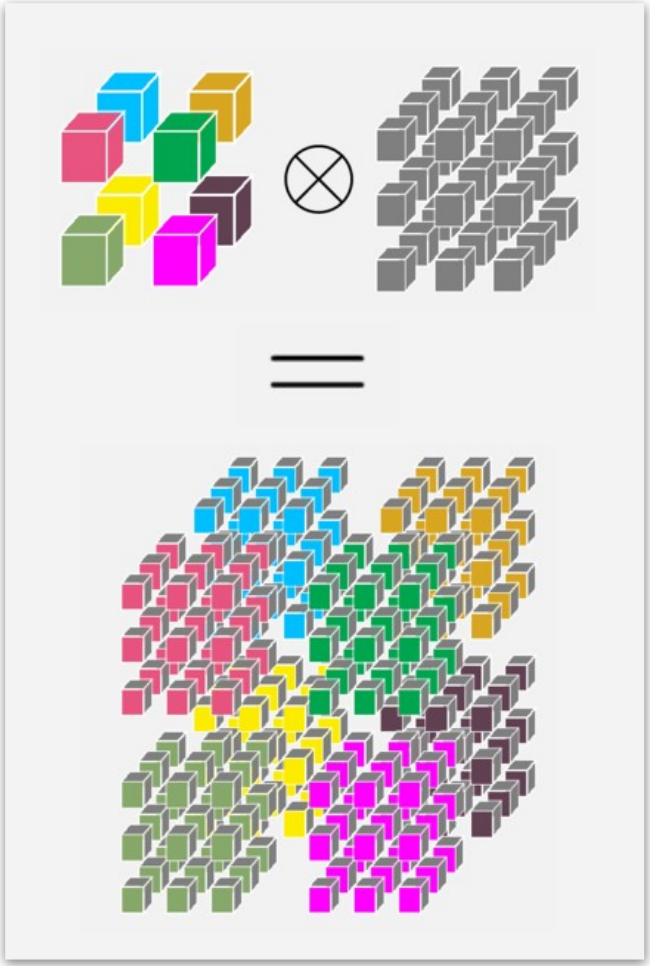
- Given prop (**P**,⊕,0), define
  m ⊗ n = mn on objects
  f ⊗ g = (f ⊗ 1) (1 ⊗ g) on morphisms
  f ⊗ 1 = f ⊕ ··· ⊕ f
  force bifunctoriality (f ⊗ 1) (1 ⊗ g) = (1 ⊗ g) (f ⊗ 1)

- This gives the free strict semisimple rig category.



- Variation: for crops, additional relation x = $\text{swap}_{1,1}$

- Example: **Bool** = < X >



Matrix = Control

Matrix = Control

Theorem: $cP = \overline{P}$

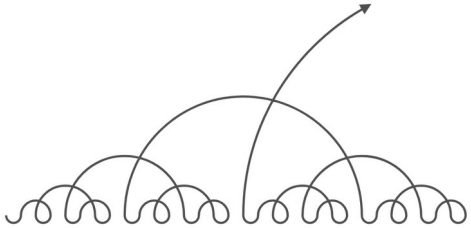
$n \longmapsto 2^n$

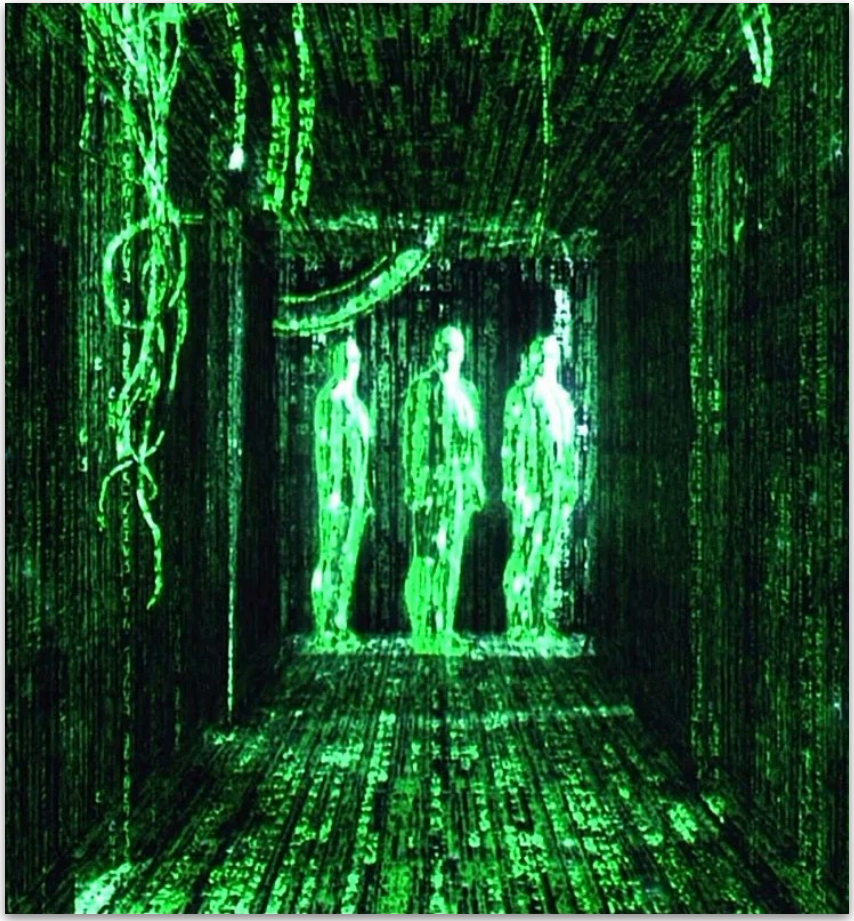
$f \longmapsto f$

$\longmapsto 1 \oplus f$

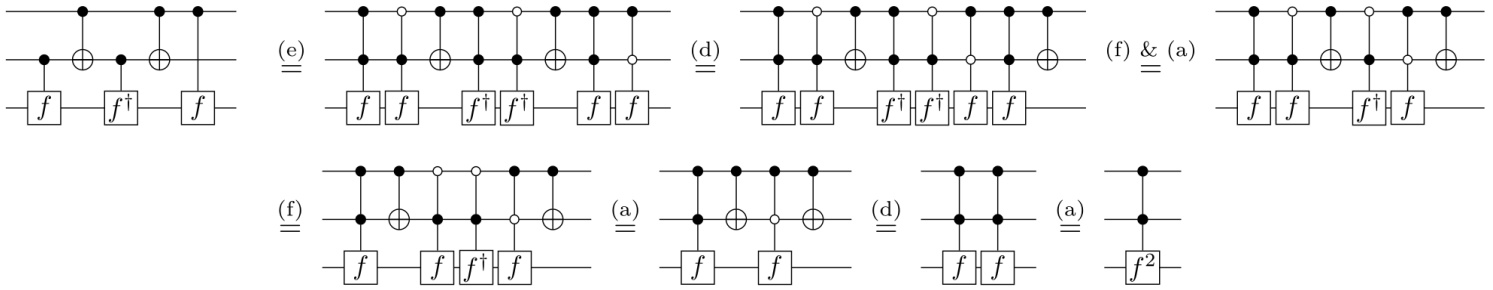
$\longmapsto$ uses Gray codes

Binary	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

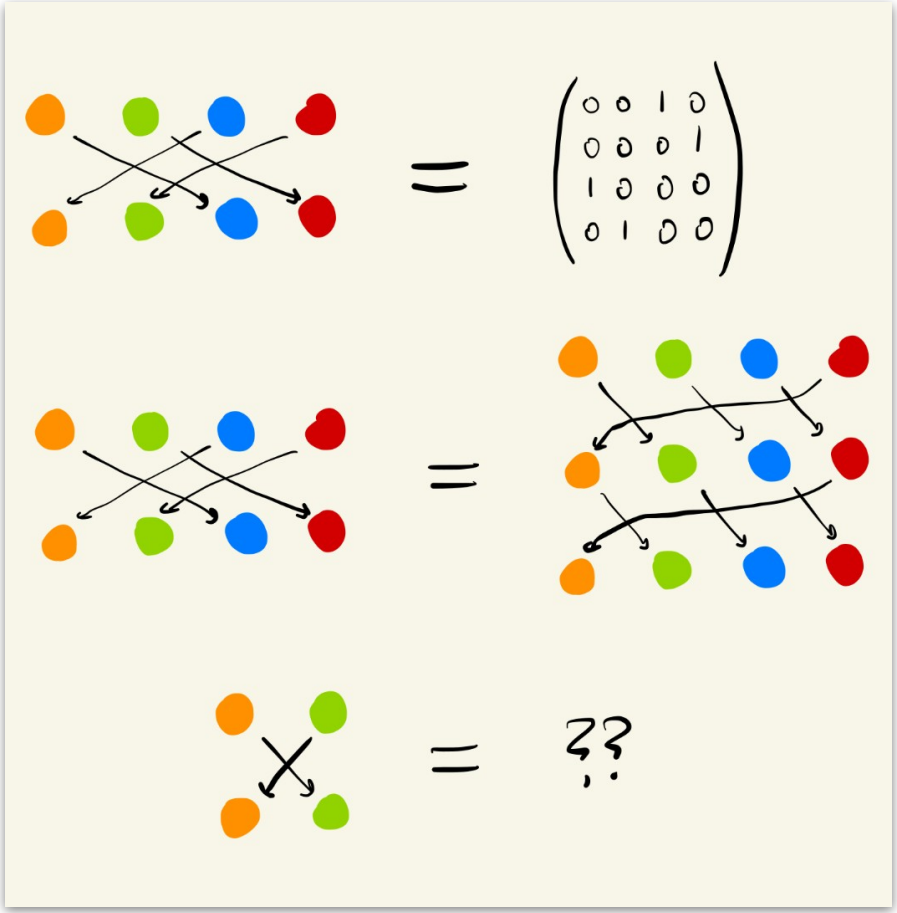




Optimisation



Square Roots



Square roots

Not all reversible classical programs in **Bool** have square roots.

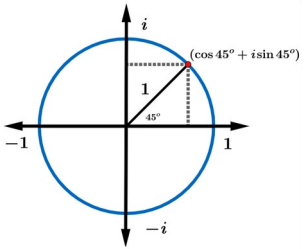
But all quantum programs in **Unitary** do!

Can we regard reversible quantum computing as a completion of reversible classical computing?

A Few Square Roots

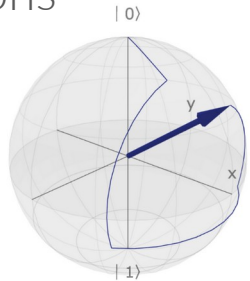
Add two generators

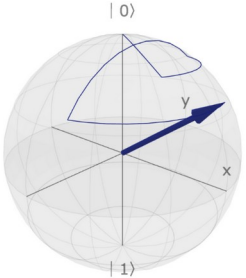
$\omega : I \to I$
$V : I \oplus I \to I \oplus I$



and three relations

V^2 = swap
ω^8 = id
VSV = SVS





where $S = 1 \oplus \omega^2$

Gates

Unitary matrices form a model with

$\omega = e^{2\pi i/8}$ $\quad V = \frac{1+i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$

In addition to classical reversible gates can now model *phase gates*

$T = id \oplus \omega$ $\quad S = id \oplus \omega^2$ $\quad Z = id \oplus \omega^4$

and the *Hadamard gate*

$H = \omega^7 V S V$

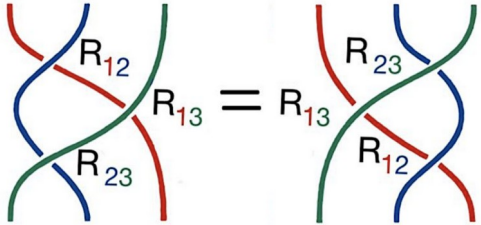
Whence the equations?

Unitaries describe discrete time evolution
In continuous time, $f=e^{itH}$ for Hamiltonian H

Square root of evolution is simply evolution for half a discrete time step: $e^{i(t/2)H}\ e^{i(t/2)H} = e^{itH}$

Borne out in quantum hardware: superconducting and trapped-ion

Third equation: anyonic quantum computing



Free Quantum Computing

Definition: **Q** = < $\omega : 1 \to 1, V : 2 \to 2$ |
$\omega^8 = 1, \quad V^2 = X, \quad S\,V\,S = V\,S\,V$ >

Proposition: **Q** exists.

Theorem (universality): **Unitary**$\supseteq$ **Q** dense.
In fact **Q = Unitary(** $\mathbb{Z}[\frac{1}{2}, e^{2\pi i/8}]$ **)**.

Theorem (soundness, completeness):
f=g in **Q** iff f=g in **Unitary**. (*)

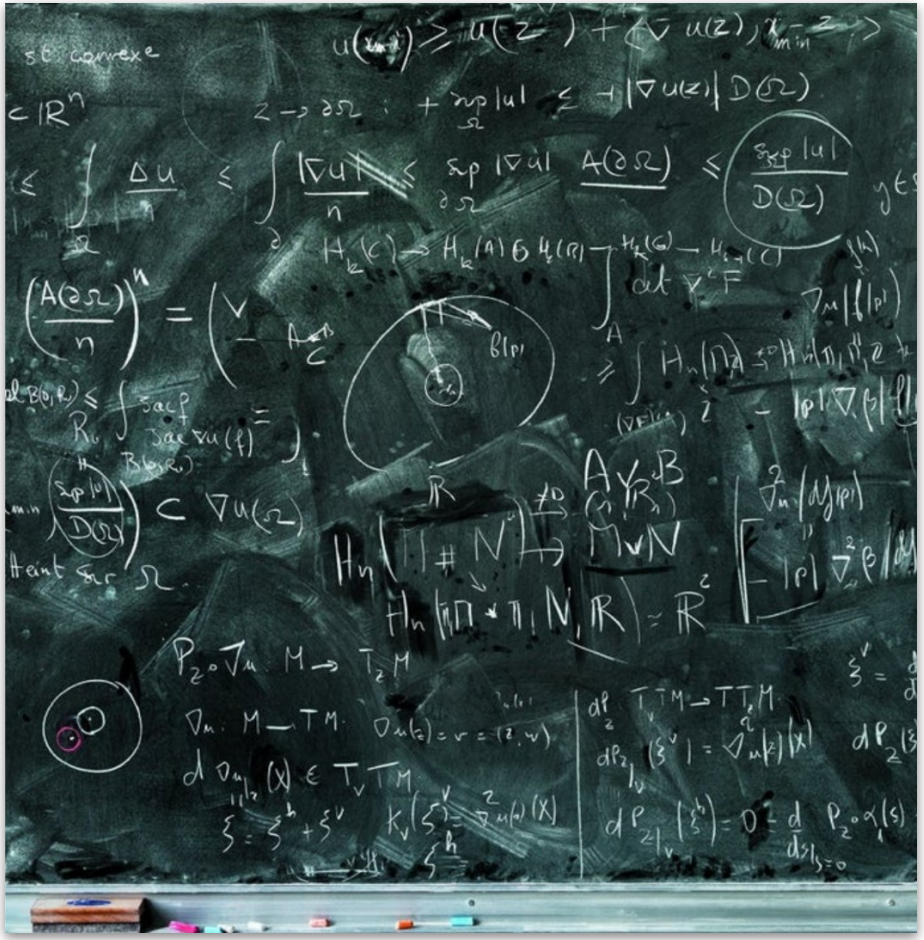
(*) $f \oplus f' = g \oplus g' \Rightarrow f = g$, not needed when we add measurement

Equational reasoning

Can be mechanised! Example:

$$
\begin{aligned}
[\![S;S]\!] &= [\![S]\!] \circ [\![S]\!] \\
&= (\mathrm{id} \oplus \omega^2) \circ (\mathrm{id} \oplus \omega^2) \\
&= (\mathrm{id} \circ \mathrm{id}) \oplus (\omega^2 \circ \omega^2) \\
&= \mathrm{id} \oplus \omega^4 \\
&= [\![Z]\!]
\end{aligned}
$$

Detailed proofs in papers,
substantial part formalised in Agda



Conclusion

Universal Quantum Programming Language

- Universal
- Fully abstract
- Finite equational presentation
- Equations natural
- Rig operations axiomatise control
- Square roots axiomatise quantum

References

- “Free Quantum Computing”
  Proceedings of the National Academy of Sciences, 2026
- “One Rig to Control Them All”
  ACM/IEEE Logic in Computer Science, 2026
- “Hadamard-Π: Equational quantum programming”
  ACM Principles of Programming Languages, 2026
- “With a Few Square Roots, Quantum Computing is as easy as Pi”
  ACM Principles of Programming Languages, 2024
- “Universal Properties of Quantum Maps”,
  Quantum Physics and Logic, 2023
- “Quantum Information Effects”
  ACM Principles of Programming Languages, 2022

Measurement

- **Theorem:** classical computation
  = classical reversible computation
  + information effects

- Can copy and delete classical bits with
  erase : b ⇝ 1
  create : 1 ⇝ b

- **Idea:** model quantum measurement similarly

  measure : qubit → qubit

  $\text{measure}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}$

Garbage and Heap

- Given rig category **P**, make new one **mP** with same objects but

  $$\mathbf{mP(A, B) = P(A \oplus H, B \otimes G) / \sim}$$

  where f ~ (1 x g) f (1 + h)

- **Theorem**: universal way to make a rig category coaffine (O initial, I terminal)
- **Theorem**: **mUnitary = CPTP**
- Practically, can compile dynamic program with measurements anywhere into static one with measurements at the end

It’s just a phase

- Many (most?) quantum algorithms come down to eigenspace decomposition and eigenvector manipulation
- Lift to primitive
- Universal

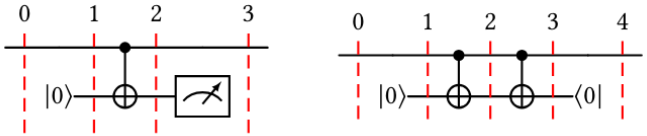
$$\begin{array}{lll} \mathsf{Z} := \text{if let } |1\rangle \text{ then } \mathrm{Ph}(\pi) & & \mathsf{X} := \text{if let } |-\rangle \text{ then } \mathrm{Ph}(\pi) \\ \mathsf{T} := \text{if let } |1\rangle \text{ then } \mathrm{Ph}(\pi/4) & & \mathsf{Y} := \text{if let } \mathsf{S} \cdot |-\rangle \text{ then } \mathrm{Ph}(\pi) \\ \mathsf{H} := \text{if let } \mathsf{Y}^{1/4} \cdot |1\rangle \text{ then } \mathrm{Ph}(\pi) & & \mathsf{CX} := \text{if let } |1\rangle \otimes \mathrm{id}_1 \text{ then } \mathsf{X} \end{array}$$

- Grover now one-liner:

$$\text{if let } |\omega_1\rangle \otimes \cdots \otimes |\omega_n\rangle \text{ then } \mathrm{Ph}(\pi)$$

More ancillae, more problems

- Cannot reuse dirty qubits

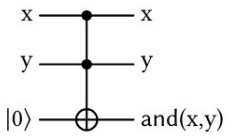


- Rust type system: ownership, borrowing

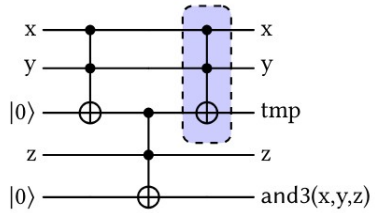
```rust
fn main() {
  let x = [1,2,3];
  let y = f(x); // value of x is moved
  print!("{x},{y}"); // so x lost access
}
fn f<T>(x : T) -> T {
  x
}
```

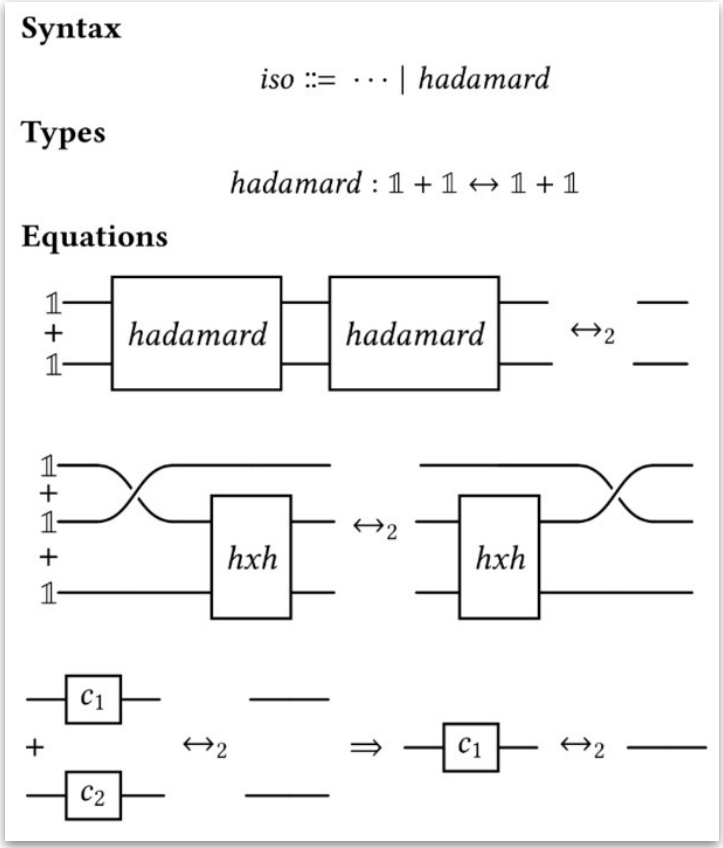
```rust
fn main() {
  let x = [1,2,3];
  let y = f(&x); // x is borrowed
  print!("{x},{y}"); // x can be read
}
fn f<T>(x : T) -> T {
  x
}
```

- Compiler automatically inserts uncomputation




```rust
fn and3<'a>(
  x:&'a qbit,
  y:&'a qbit,
  z:&'a qbit
) -> #'a qbit {
  let tmp = and(x,y);
  let ref = &tmp;
  and(ref,z)
}
```





Programming with equations

- Word problem decidable
- Normalisation by evaluation
- Fewer generators than Π
- **Theorem (completeness):** $[\![f]\!]=[\![g]\!]$ iff $(\!|f|\!)=(\!|g|\!)$

Orthogonal group presentations

Proposition: there is finite presentation for unitary matrices $O_n(\mathbb{Z}[1/\sqrt{2}])$

$$(-1)_{[a]}^2 \approx \varepsilon \qquad H_{[a,b]}^2 \approx \varepsilon$$

$$X_{[a,b]}^2 \approx \varepsilon$$

$$(-1)_{[a]}H_{[b,c]} \approx H_{[b,c]}(-1)_{[a]}$$

$$X_{[a,b]}H_{[c,d]} \approx H_{[c,d]}X_{[a,b]}$$

$$(-1)_{[a]}(-1)_{[b]} \approx (-1)_{[b]}(-1)_{[a]} \qquad H_{[a,b]}H_{[c,d]} \approx H_{[c,d]}H_{[a,b]}$$

$$(-1)_{[a]}X_{[b,c]} \approx X_{[b,c]}(-1)_{[a]}$$

$$X_{[a,b]}X_{[c,d]} \approx X_{[c,d]}X_{[a,b]} \qquad H_{[b,c]}X_{[a,b]} \approx X_{[a,b]}H_{[a,c]}$$

$$H_{[a,c]}X_{[b,c]} \approx X_{[b,c]}H_{[a,b]}$$

$$(-1)_{[a]}X_{[a,b]} \approx X_{[a,b]}(-1)_{[b]}$$

$$X_{[b,c]}X_{[a,b]} \approx X_{[a,b]}X_{[a,c]} \qquad (-1)_{[a]}(-1)_{[b]}H_{[a,b]} \approx H_{[a,b]}(-1)_{[a]}(-1)_{[b]}$$

$$X_{[a,c]}X_{[b,c]} \approx X_{[b,c]}X_{[a,b]} \qquad (-1)_{[b]}H_{[a,b]} \approx H_{[a,b]}X_{[a,b]}$$

$$X_{[b,c]}H_{[a,b]}X_{[a,c]}H_{[a,d]}H_{[a,b]}X_{[a,c]}H_{[a,d]} \approx H_{[a,b]}X_{[a,c]}H_{[a,d]}H_{[a,b]}X_{[a,c]}H_{[a,d]}X_{[c,d]}$$