Programming Abstractions for Quantum Computing

Jennifer Paykin
jpaykin@galois.com

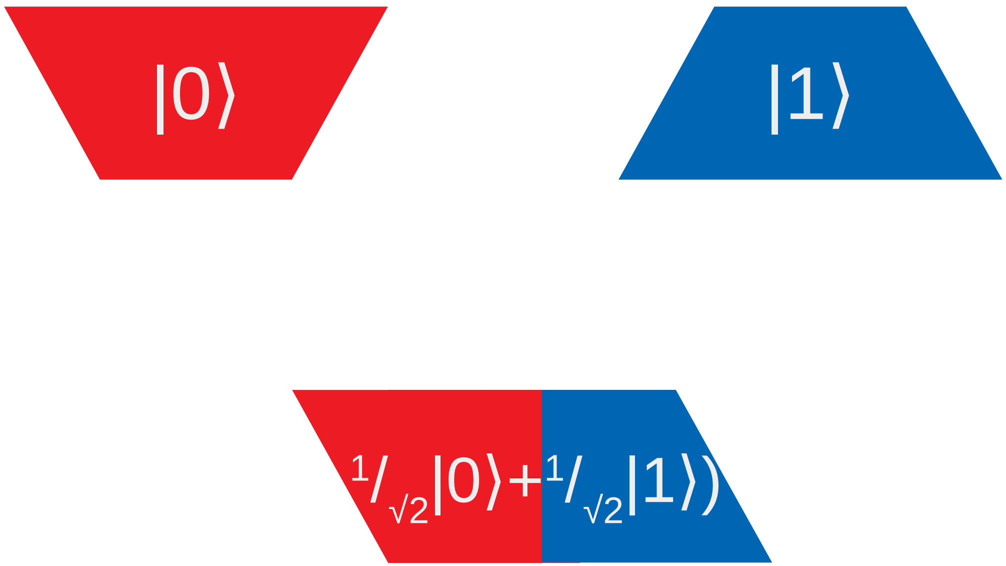
Identify abstractions for solutions to quantum computing problems.

- ☑ New, different, unintuitive computing environment
- ☑ Abstractions could speed up development of useful algorithms and optimizations
- ☑ Influence state-of-the-art systems

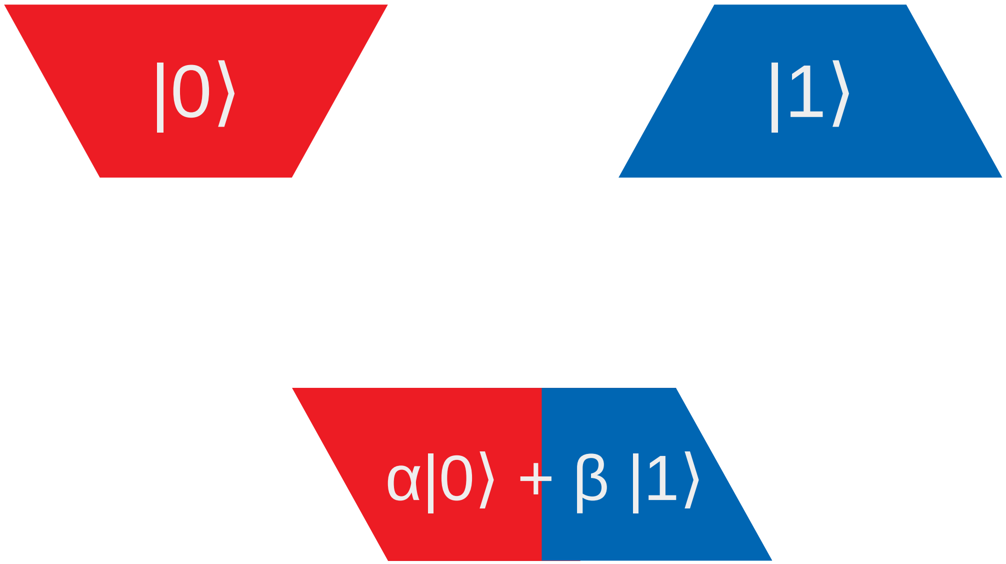
Quantum Computing 101

Qubits

Qubits



Qubits

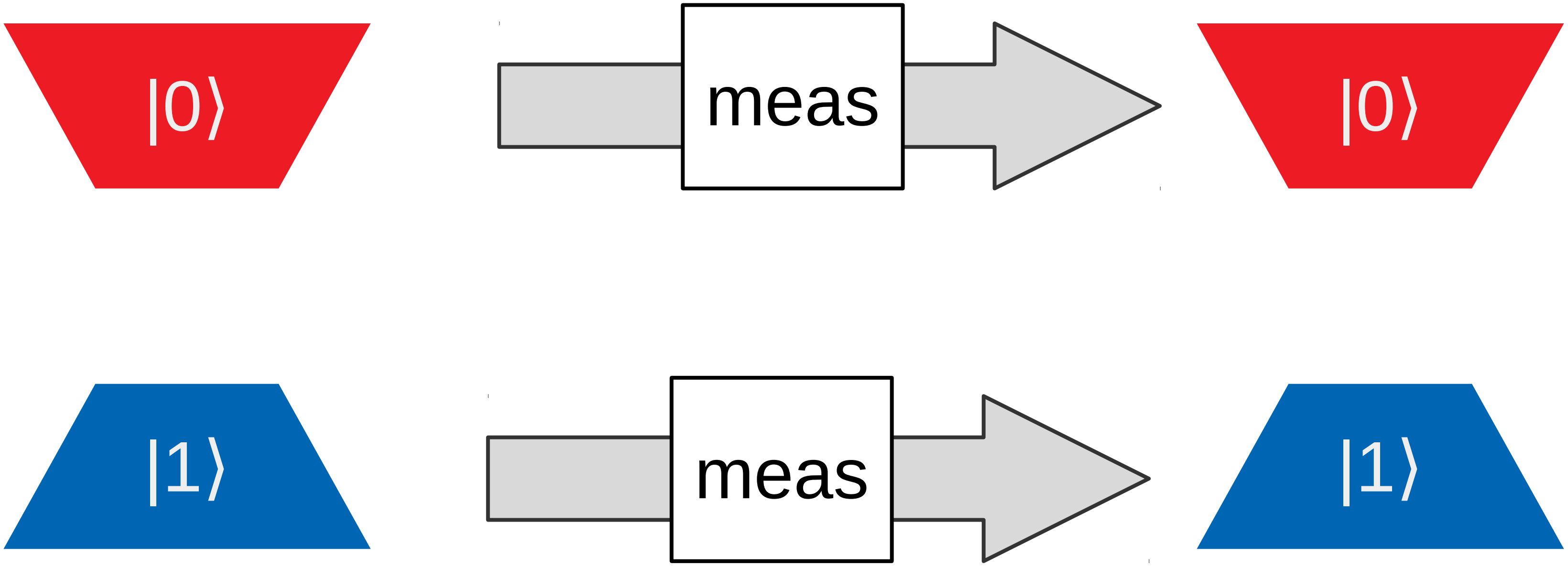


$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

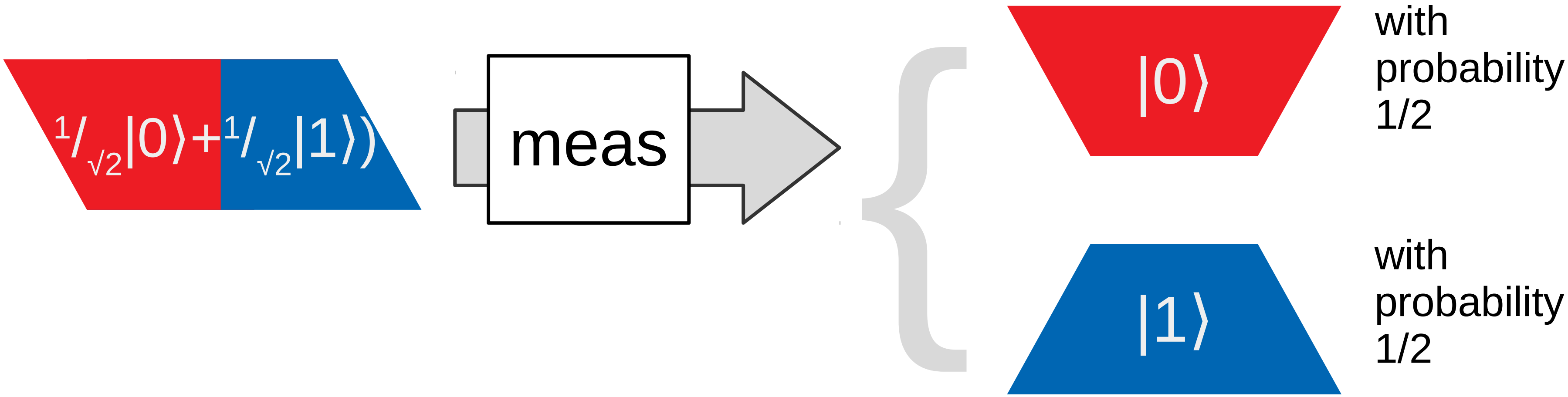
$\alpha, \beta \in \mathbb{C}$

$\alpha^2 + \beta^2 = 1$

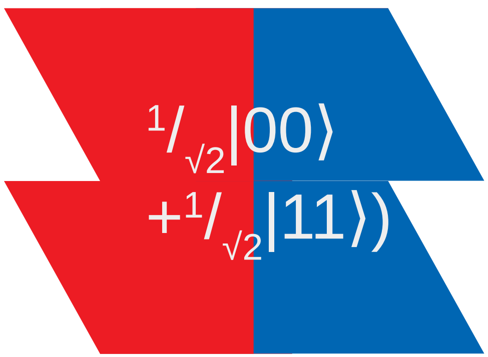
Measurement



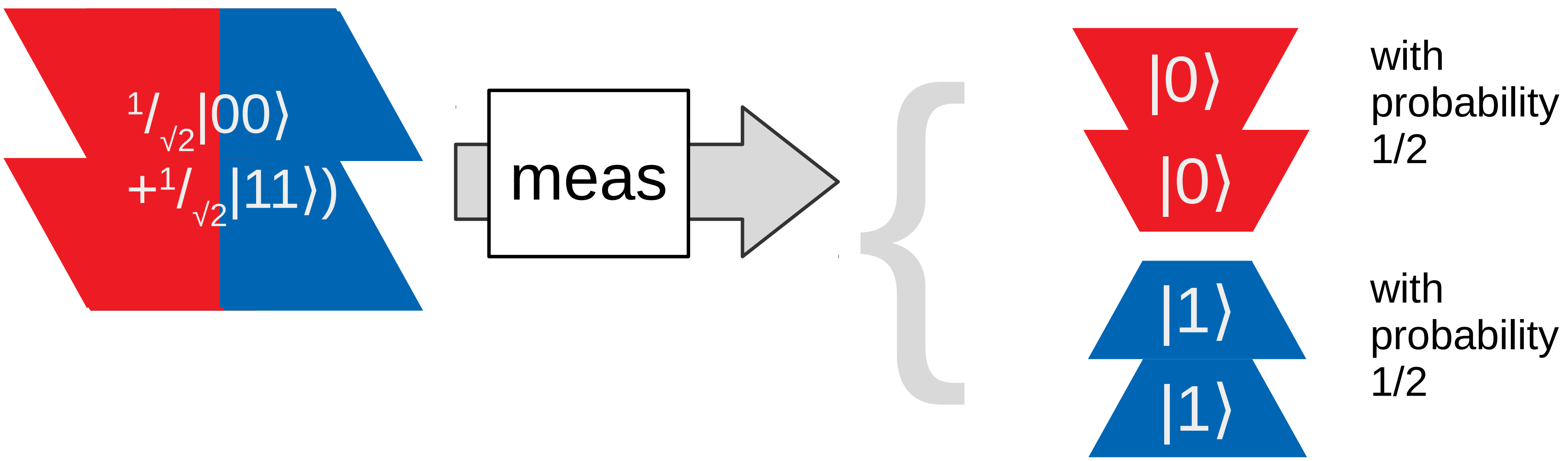
Measurement



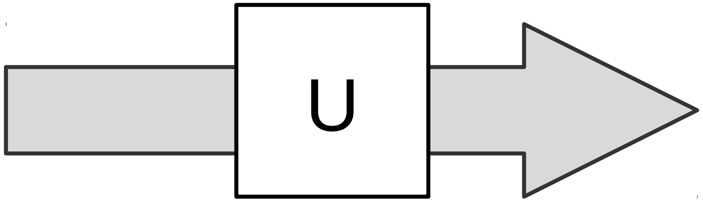
Entanglement



Entanglement



Unitary Matrices



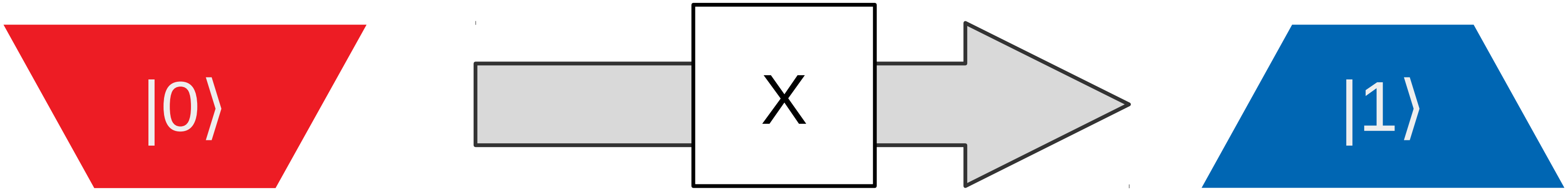
In linear algebra, a complex square matrix U is **unitary** if its conjugate transpose U^* is also its inverse, that is, if

$$U^*U = UU^* = I,$$

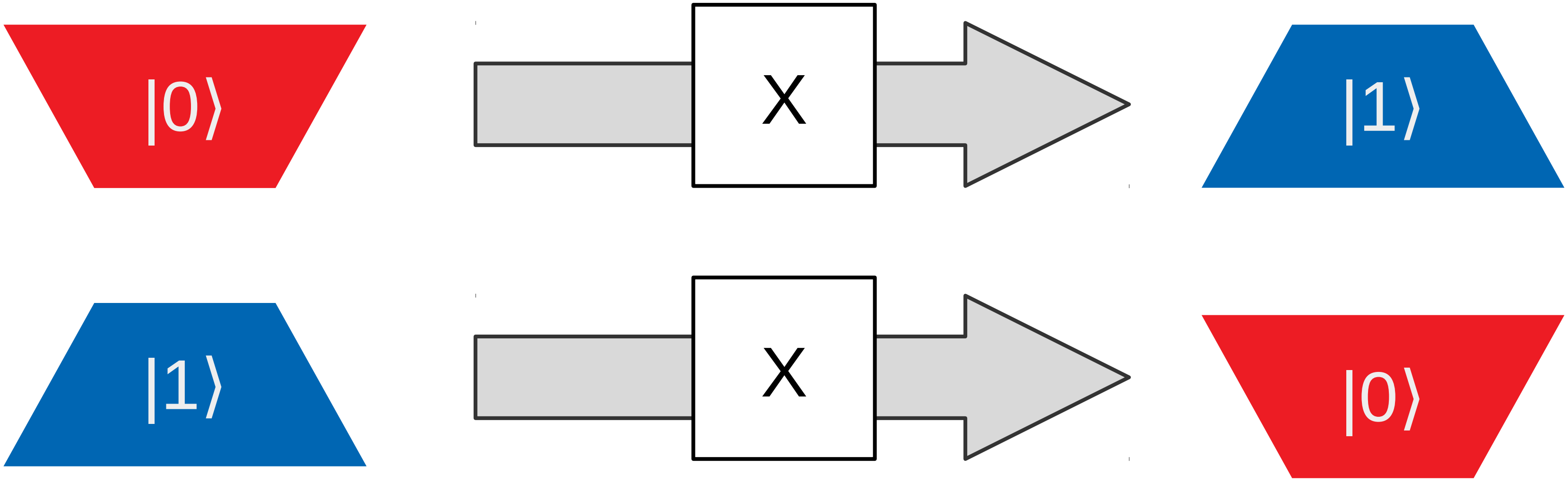
where I is the identity matrix.

Unitary Operations: NOT=X=$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

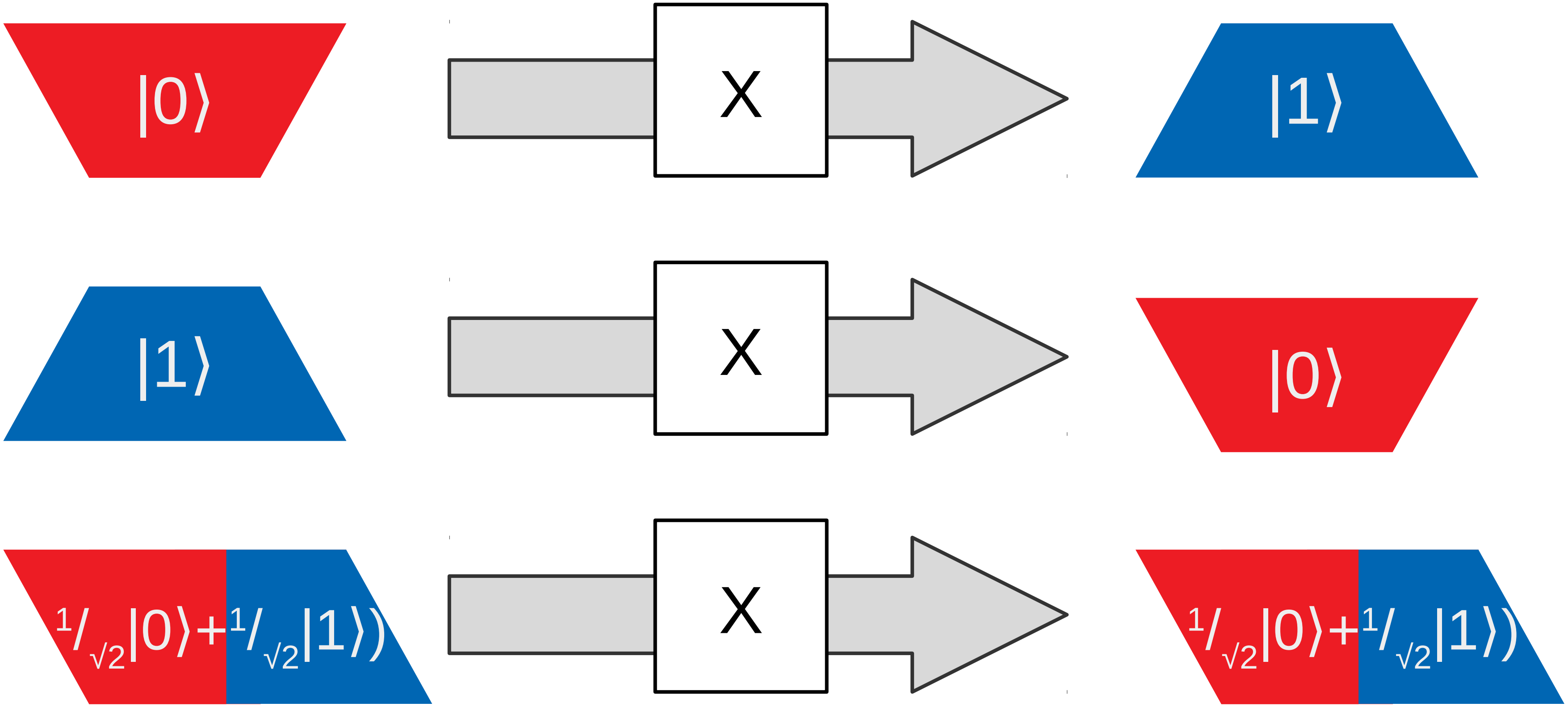
Unitary Operations: NOT=X=$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$



Unitary Operations: NOT=X=$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

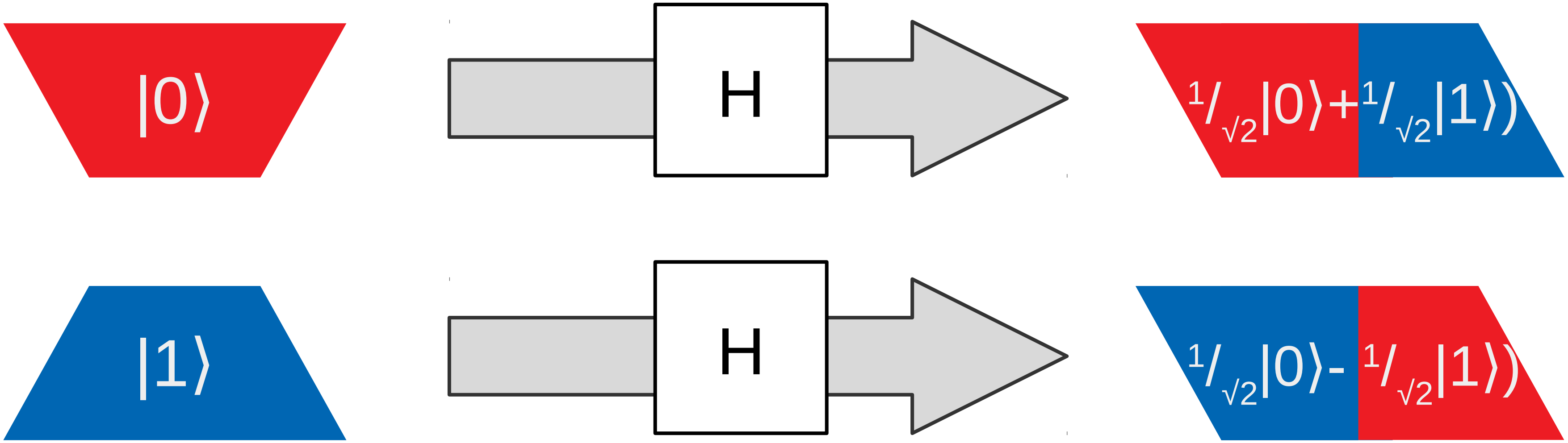


Unitary Operations: NOT=X=$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

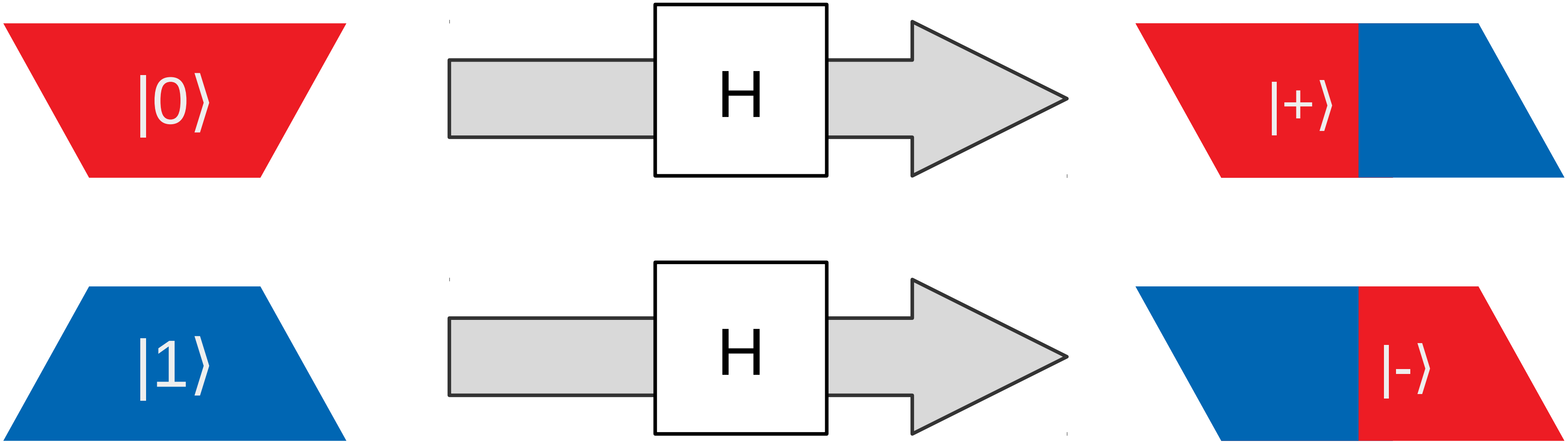


Hadamard = $^{1}/_{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

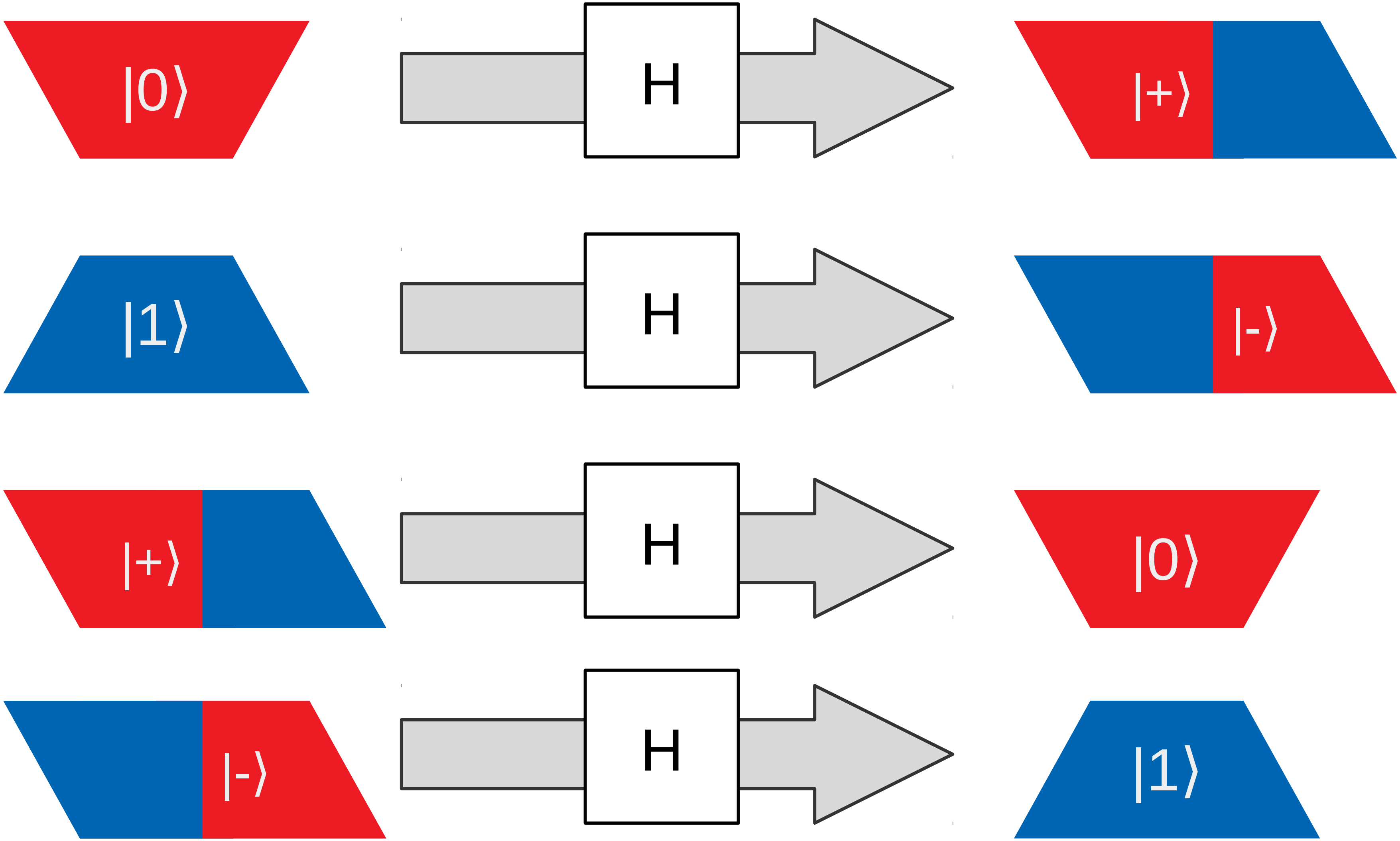
Hadamard = $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$



Hadamard = $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$



Hadamard = $^1/_{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$



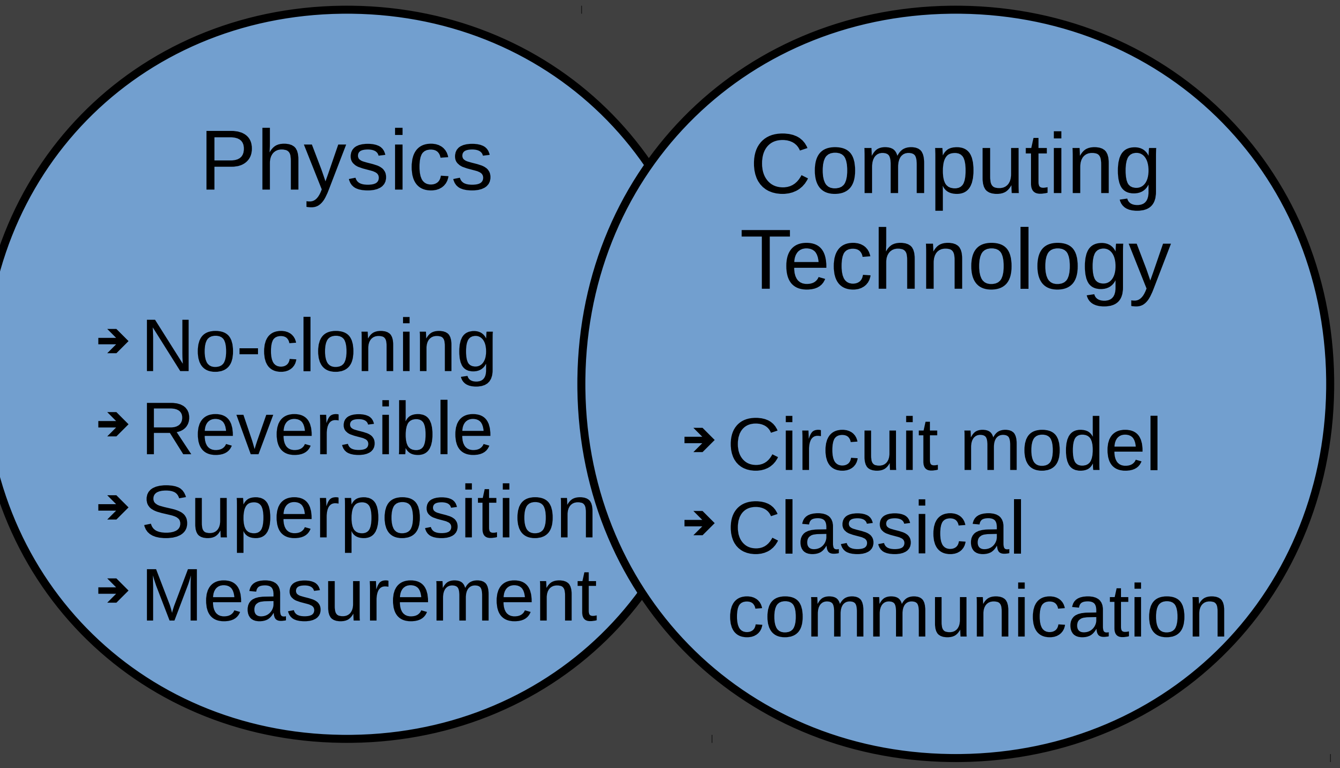
Sources of Abstractions

Sources of Abstractions

Physics

- No-cloning
- Reversible
- Superposition
- Measurement

Sources of Abstractions



Sources of Abstractions

Physics

- No-cloning
- Reversible
- Superposition
- Measurement

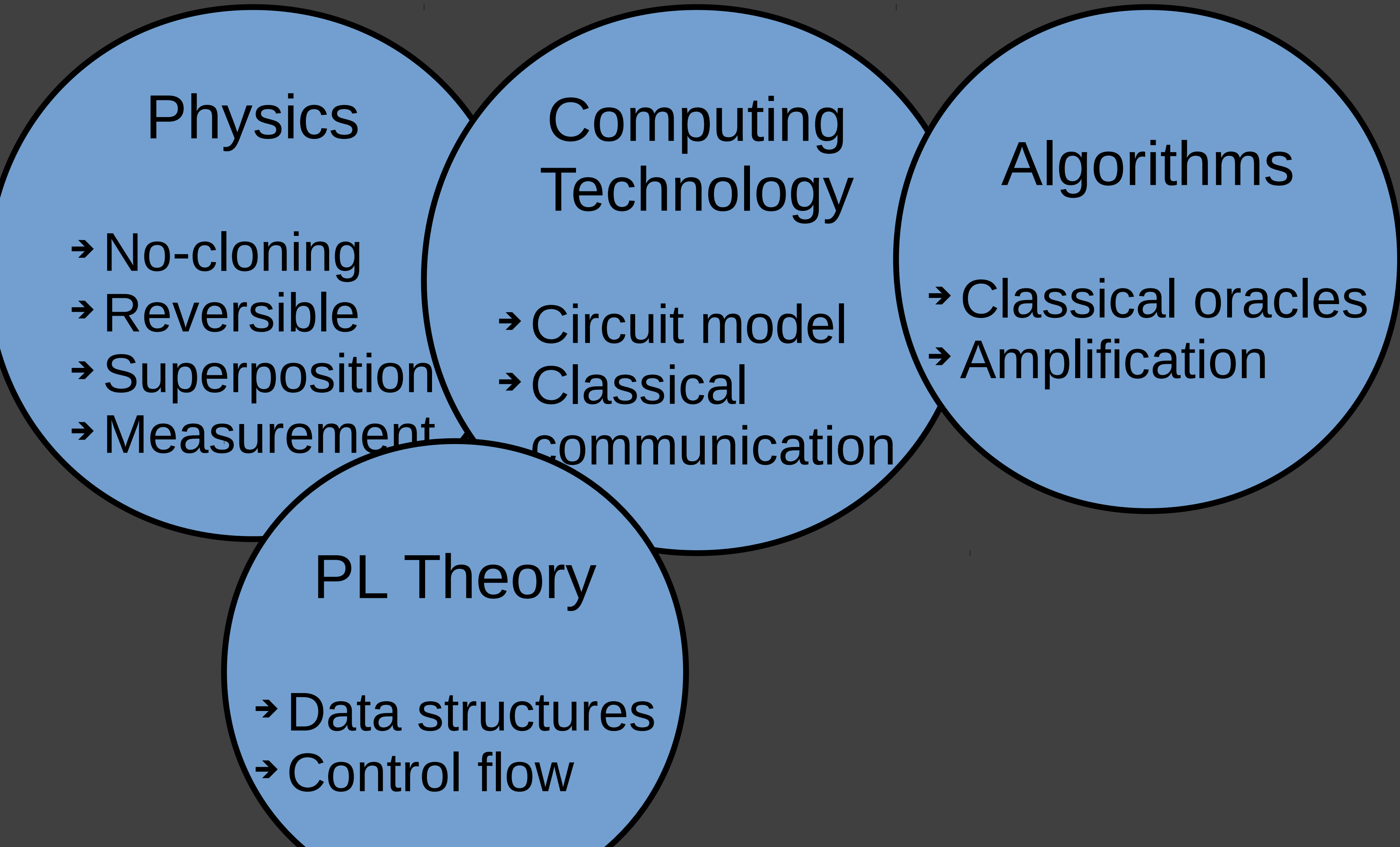
Computing Technology

- Circuit model
- Classical communication

Algorithms

- Classical oracles
- Amplification

Sources of Abstractions



Sources of Abstractions

Physics

- No-cloning
- Reversible
- Superposition
- Measurement

Computing Technology

- Circuit model
- Classical communicati

Algorithms

- Classical oracles
- Amplification

PL Theory

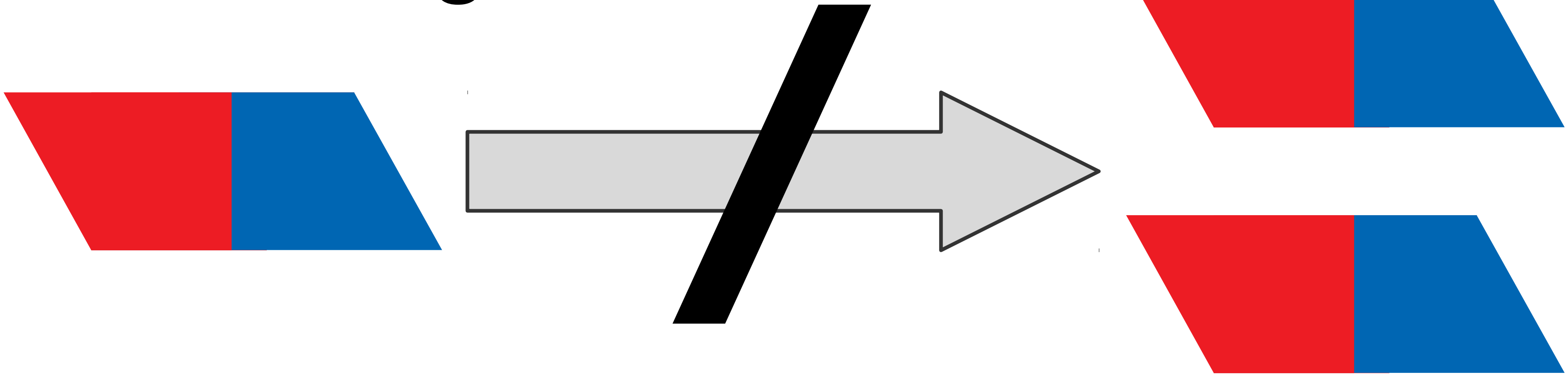
- Data structures
- Control flow

Semantics

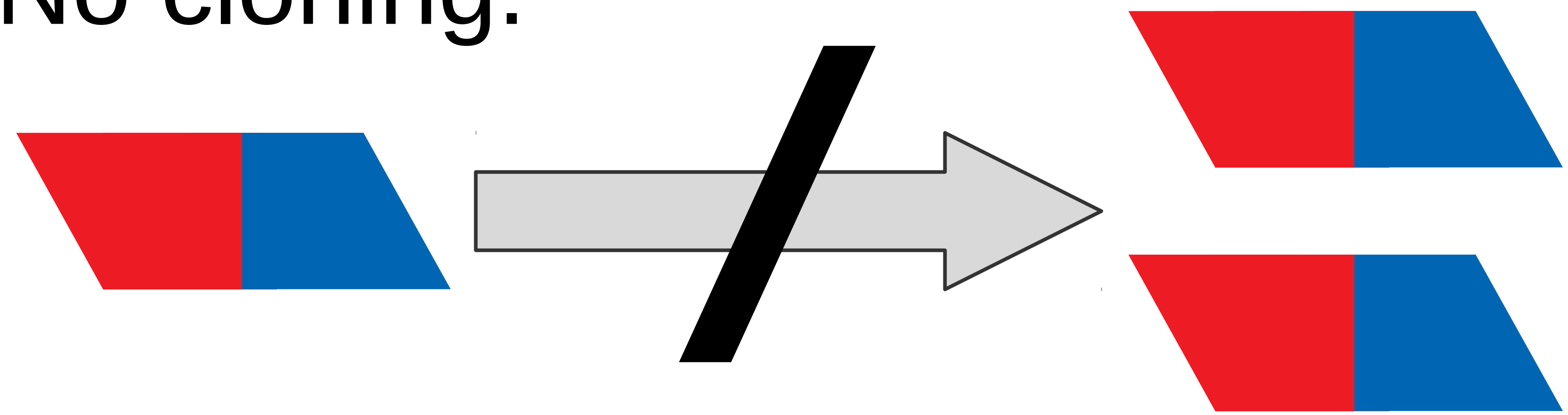
- Quantum CPOs
- String diagrams

Physics Abstraction
No-Cloning

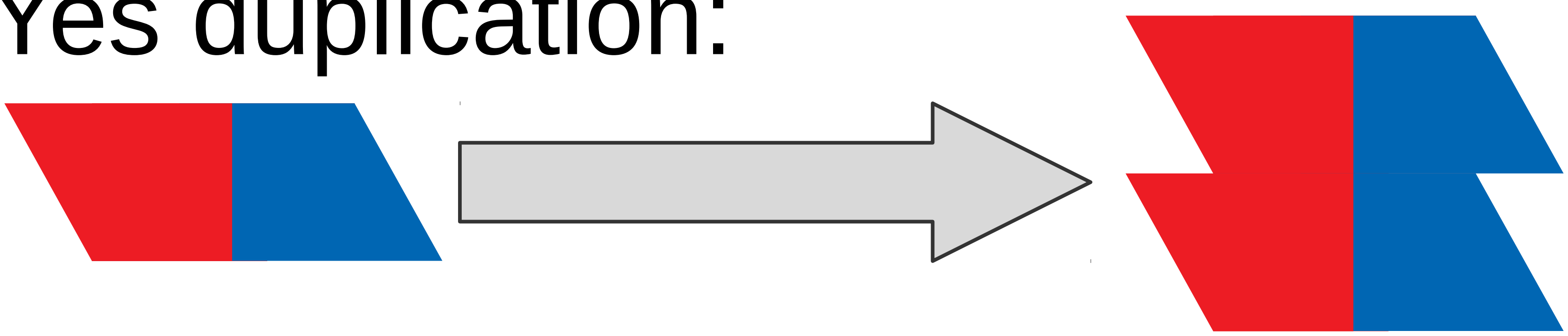
No cloning:



No cloning:



Yes duplication:



QPL: Substructural types

$$
\begin{array}{lccl}
\text{QPL Terms} & P, Q & ::= & \textbf{new bit } b := \mathbf{0} \mid \textbf{new qbit } q := \mathbf{0} \mid \textbf{discard } x \\
 & & \mid & b := \mathbf{0} \mid b := \mathbf{1} \mid q_1, \ldots, q_n \mathrel{*}= S \\
 & & \mid & \textbf{skip} \mid P; Q \\
 & & \mid & \textbf{if } b \textbf{ then } P \textbf{ else } Q \mid \textbf{measure } q \textbf{ then } P \textbf{ else } Q \mid \textbf{while } b \textbf{ do } P \\
 & & \mid & \textbf{proc } X : \Gamma \to \Gamma' \ \{\ P\ \} \textbf{ in } Q \mid y_1, \ldots, y_m = X(x_1, \ldots, x_n)
\end{array}
$$

$$
\frac{}{\Pi \vdash \langle \Gamma \rangle \ \textbf{new qbit } q := \mathbf{0} \ \langle q{:}\textbf{qbit}, \Gamma \rangle}
$$

$$
\frac{}{\Pi \vdash \langle x{:}t, \Gamma \rangle \ \textbf{discard } x \ \langle \Gamma \rangle}
$$

Towards a Quantum Programming Language. Selinger, 2004

Quantum λ calculus

$$\begin{aligned} M, N, P \ ::= \ & c \mid x \mid \lambda x.M \mid MN \mid \\ & \langle M, N \rangle \mid * \mid let\ \langle x, y \rangle = M\ in\ N \mid \\ & inj_l(M) \mid inj_r(M) \mid match\ P\ with\ (x \mapsto M \mid y \mapsto N) \mid \\ & let\ rec\ f\ x = M\ in\ N. \end{aligned}$$

$$\begin{aligned} c ::= \ & \text{new} : () \multimap \text{qubit} \\ \mid \ & \text{meas} : \text{qubit} \multimap \text{bit} \\ \mid \ & U : \text{qubit}^{\otimes n} \multimap \text{qubit}^{\otimes n} \end{aligned}$$

Quantum Lambda Calculus. Selinger and Valiron, 2009

Quantum λ calculus

$$
\begin{aligned}
M, N, P \;::=\; & c \mid x \mid \lambda x.M \mid MN \mid \\
& \langle M, N \rangle \mid * \mid let\ \langle x, y \rangle = M\ in\ N \mid \\
& inj_l(M) \mid inj_r(M) \mid match\ P\ with\ (x \mapsto M \mid y \mapsto N) \mid \\
& let\ rec\ f\ x = M\ in\ N.
\end{aligned}
$$

The Bell experiment can be viewed as the composition

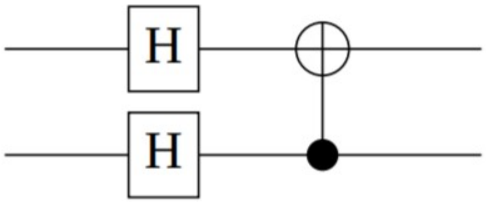
$$\top \xrightarrow{\mathbf{EPR}} qbit \otimes qbit \xrightarrow{f' \otimes f'} (trit \to bit) \otimes (trit \to bit),$$

which produces a term of type $(trit \to bit) \otimes (trit \to bit)$, i.e., a pair $\langle f, g \rangle$ of entangled functions.

Quantum Lambda Calculus. Selinger and Valiron, 2009

Quipper: Circuit generation

```
mycirc :: Qubit -> Qubit -> Circ (Qubit, Qubit)
mycirc a b = do
  a <- hadamard a
  b <- hadamard b
  (a,b) <- controlled_not a b
  return (a,b)
```



Quipper: A Scalable Quantum Programming Language. Green, Lundsdane, Ross, Selinger, and Valiron, 2013

Dependent Types

or

Control Flow

?

PL Abstraction
Dependent Types

Quantum Data Types

Quipper: A Scalable Quantum Programming Language. Green, Lundsdane, Ross, Selinger, and Valiron, 2013

Quantum Data Types

- Qubits, finite tuples of qubits

Quipper: A Scalable Quantum Programming Language. Green, Lundsdane, Ross, Selinger, and Valiron, 2013

Quantum Data Types

- Qubits, finite tuples of qubits
- Lists of qubits
  - Introduced in Quipper
  - Present in most mainstream languages

Quipper: A Scalable Quantum Programming Language. Green, Lundsdane, Ross, Selinger, and Valiron, 2013

Quantum Data Types

- Qubits, finite tuples of qubits
- Lists of qubits
  - Introduced in Quipper
  - Present in most mainstream languages
- Polymorphic lists, trees, algebraic data types

Quipper: A Scalable Quantum Programming Language. Green, Lundsdane, Ross, Selinger, and Valiron, 2013

Classically Dependent Quantum Types

```
Inductive Box w1 w2 : Set := ...
```

```
Definition hadamard_measure : Box Qubit Bit :=
  box_ q ⟹ meas $ _H $ q.
```

QWIRE: A Core Language for Quantum Circuits. Paykin, Rand, Zdancewic, 2017

Classically Dependent Quantum Types

```
Fixpoint NTensor (n : nat) (W : WType) :=
  match n with
  | 0    => One
  | S n' => W ⊗ NTensor n' W
  end.
Infix "⨂" := NTensor (at level 30) : circ_scope.
```

https://github.com/inQWIRE/QWIRE

Classically Dependent Quantum Types

```
Fixpoint inParMany (n : nat) {W W'} (c : Box W W') : Box (n ⨂ W) (n ⨂ W') :=
  match n with
  | 0    => id_circ
  | S n' => inPar c (inParMany n' c)
  end.
```

https://github.com/inQWIRE/QWIRE

Classically Dependent Quantum Types

```
Fixpoint inParMany (n : nat) {W W'} (c : Box W W') : Box (n ⨂ W) (n ⨂ W') :=
  match n with
  | 0    => id_circ
  | S n' => inPar c (inParMany n' c)
  end.
```

```
Definition Deutsch_Jozsa (n : nat) (U : Box (S n ⨂ Qubit) (S n ⨂ Qubit))
                                  : Box One (n ⨂ Bit) :=
  box_ () ⇒
    let_ qs     ← _H #n $ init0 #n $ (());
    let_ q      ← _H $ init1 $ ();
    let_ (q,qs) ← U $ (q,qs);
    let_ ()      ← discard $ meas $q;
    meas #n $ _H #n $ qs.
```

https://github.com/inQWIRE/QWIRE

Shape-Dependent Quantum Types

```
-- length :: List Unit -o Nat
--
-- x : Shape(List Qubit) |- Vec Qubit (length x) : Type
toVec :: ! (x :: List Qubit) -o Vec Qubit (length x)
toVec x = case x of
          Nil -> VNil
          Cons y zs -> VCons y (g' zs)
```

Proto-Quipper-D. Fu and Selinger.

QQTT?
(Quantum Quantitative Type Theory)

```
withAncilla : (Qubit -> List Qubit -> Qubit ⊗ List Qubit) ->
              List Qubit -> List Qubit
withAncilla f ls = let (q,ls') ← f (new 0) ls in
                   -- should be the case that q=|0⟩
                   let _ ← discard q in
                   ls'
```

QQTT?
(Quantum Quantitative Type Theory)

```
data Is0 (q : Qubit) : Type where
  Is0 : Is0 (init 0)

withAncilla : ( (q : Qubit) ⊗ Is0 q -> List Qubit ->
                (q': Qubit) ⊗ Is0 q' ⊗ List Qubit ) ->
              List Qubit -> List Qubit
withAncilla f ls = let (q',pf,ls') = f (init 0,Is0,ls) in
                   -- discard : (q : Qubit) -> Is0 q -> ()
                   let _ ← discard q' pf in
                   ls'
```

Equality??

Algebraic Effects, Linearity, and Quantum Programming Languages. Staton 2015.
A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

Equality??

$$(U_1 \otimes U_2) \# (e_1, e_2) \approx (U_1 \# e_1, U_2 \# e_2) \qquad \text{(U-}\otimes\text{-INTRO)}$$

$$\texttt{let } (x_1, x_2) := (U_1 \otimes U_2) \# e \texttt{ in } e'$$

$$\approx \texttt{let } (y_1, y_2) := e \texttt{ in } e'\{U_1 \# y_1/x_1, U_2 \# y_2/x_2\} \qquad \text{(U-}\otimes\text{-ELIM)}$$

$$U \# (\texttt{let } (x_1, x_2) := e \texttt{ in } e') \approx \texttt{let } (x_1, x_2) := e \texttt{ in } U \# e' \qquad \text{(U-}\otimes\text{-COMM)}$$

$$(U_1 \oplus U_2) \# (\iota_1 e) \approx U_1 \# e \qquad \text{(U-}\oplus\text{-INTRO}_1\text{)}$$

$$(U_1 \oplus U_2) \# (\iota_2 e) \approx U_2 \# e \qquad \text{(U-}\oplus\text{-INTRO}_2\text{)}$$

$$\texttt{case } (U_1 \oplus U_2) \# e \texttt{ of } (\iota_1 x_1 \to e_1 \mid \iota_2 x_2 \to e_2)$$

$$\approx \texttt{case } e \texttt{ of } (\iota_1 y_1 \to e_1\{U_1 \# y_1/x_1\} \mid \iota_2 y_2 \to e_2\{U_2 \# y_2/x_2\}) \qquad \text{(U-}\oplus\text{-ELIM)}$$

$$U \# (\texttt{case } e \texttt{ of } (\iota_1 x_1 \to e_1 \mid \iota_2 x_2 \to e_2))$$

$$\approx \texttt{case } e \texttt{ of } (\iota_1 x_1 \to U \# e_1 \mid \iota_2 x_2 \to U \# e_2) \qquad \text{(U-}\oplus\text{-COMM)}$$

$$U \# (e >! f) \approx e >! \lambda x \to U \# (f x) \qquad \text{(U-LOWER-COMM)}$$

$$U \# e >! \lambda\_.e' \approx e >! \lambda\_.e' \qquad \text{(U-LOWER-ELIM)}$$

$$U \# (V \# e) \approx (U \circ V) \# e \qquad \text{(U-COMPOSE)}$$

$$I \# e \approx e \qquad \text{(U-I)}$$

$$U^\dagger \# U \# e \approx e \qquad \text{(U-}\dagger\text{)}$$

Algebraic Effects, Linearity, and Quantum Programming Languages. Staton 2015.
A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

Equality??

$$X \mathbin{\#} \texttt{init}\ b \approx \texttt{init}(\neg b) \qquad \text{(X-INTRO)}$$

$$\texttt{let}\ !x := \texttt{meas}(X \mathbin{\#} e)\ \texttt{in}\ e' \approx \texttt{let}\ !y := \texttt{meas}\ e\ \texttt{in}\ e'\{\neg y/x\} \qquad \text{(X-ELIM)}$$

$$\texttt{SWAP} \mathbin{\#} (e_1, e_2) \approx (e_2, e_1) \qquad \text{(SWAP-INTRO)}$$

$$\texttt{let}\ (x, y) := \texttt{SWAP} \mathbin{\#} e\ \texttt{in}\ e' \approx \texttt{let}\ (y, x) := e\ \texttt{in}\ e' \qquad \text{(SWAP-ELIM)}$$

$$\texttt{DISTR} \mathbin{\#} (\texttt{init}\ b, e) \approx \texttt{if}\ b\ \texttt{then}\ \iota_2\ e\ \texttt{else}\ \iota_1\ e \qquad \text{(DISTR-INTRO)}$$

$$\texttt{case}(\texttt{DISTR} \mathbin{\#} e)\ \texttt{of}\ (\iota_1 z_1 \to e_1 \mid \iota_2 z_2 \to e_2) \approx \texttt{let}\ (!b, y) := e\ \texttt{in}\ (\texttt{init}\ b, e) \qquad \text{(DISTR-ELIM)}$$

Algebraic Effects, Linearity, and Quantum Programming Languages. Staton 2015.
A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT

- Higher Inductive Types (HITs) use paths to encode equivalence relations or groupoids
  - Groupoid: category where all morphisms are invertible

$$\frac{f : G(\alpha, \beta)}{[f] : [\alpha] = [\beta]}$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT

- Higher Inductive Types (HITs) use paths to encode equivalence relations or groupoids
  - Groupoid: category where all morphisms are invertible
- Path induction holds of HITs
  - i.e. prove theorems with just base case refl
  - Simplify proofs about groupoi

$$\frac{f : G(\alpha, \beta)}{[f] : [\alpha] = [\beta]}$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT

- Higher Inductive Types (HITs) use paths to encode equivalence relations or groupoids
  - Groupoid: category where all morphisms are invertible
- Path induction holds of HITs
  - i.e. prove theorems with just base case refl
  - Simplify proofs about groupoids
- Unitaries form a groupoid

$$\frac{f : G(\alpha, \beta)}{[f] : [\alpha] = [\beta]}$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

- $\mathrm{UMatrix}(\alpha, \beta)$: unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.
- Quantum types: $\mathrm{QType} = \mathrm{FinType}/\mathrm{UMatrix}$.
  - $\mathrm{Qubit} = [\mathrm{Bool}]_{\mathrm{UMatrix}}$
- Unitaries are paths:

$$\frac{U : \mathrm{UMatrix}(\alpha, \beta)}{[U] : [\alpha] = [\beta]}$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

- $H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$, $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- $[H] : \mathsf{Qubit} = \mathsf{Qubit}$, $[X] : \mathsf{Qubit} = \mathsf{Qubit}$
- $[H] \neq [X] \neq 1_{\mathsf{Qubit}}$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.

$(\sigma, \tau : QType \equiv FinType/UMatrix)$

If $e : QExp\ \sigma$, there exists $U \# e : QExp\ \tau$.

(apply the unitary U to e)

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.
$(\sigma, \tau : \mathit{QType} \equiv \mathit{FinType}/\mathit{UMatrix})$

If $e : \mathit{QExp}\ \sigma$, *there exists* $U \# e : \mathit{QExp}\ \tau$.
(apply the unitary U to e)

Proof.

By path induction. Base case for $1_\sigma : \sigma = \sigma$:

$$1_\sigma \# e \equiv e$$

□

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let U *be a unitary transformation* $U : \sigma = \tau$.
($\sigma, \tau : \mathit{QType} \equiv \mathit{FinType}/\mathit{UMatrix}$)

If $e : \mathit{QExp}\ \sigma$, *there exists* $U \# e : \mathit{QExp}\ \tau$.
(apply the unitary U *to* e*)*

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.

$(\sigma, \tau : \mathit{QType} \equiv \mathit{FinType}/\mathit{UMatrix})$

If $e : \mathit{QExp}\ \sigma$, *there exists* $U \# e : \mathit{QExp}\ \tau$.

(apply the unitary U to e)

Proof.

By path induction. Base case for $1_\sigma : \sigma = \sigma$:

$$1_\sigma \# e \equiv e$$

□

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.
(σ, τ : *QType* $\equiv$ *FinType/UMatrix*)

If $e : \mathit{QExp}\ \sigma$, *there exists* $U \# e : \mathit{QExp}\ \tau$.
(*apply the unitary U to e*)

Note

$[H] \# e \neq e$ *because* $[H] \neq 1_{\mathit{Qubit}}$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let $U : \sigma = \tau$ *and* $V : \tau = \rho$ *be unitaries. Then*

$$V \# (U \# e) = (V \circ U) \# e.$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

Let $U : \sigma = \tau$ and $V : \tau = \rho$ be unitaries. Then

$$V \# (U \# e) = (V \circ U) \# e.$$

Proof.

By path induction on V. If $V \equiv 1_\tau$ then

$$LHS = 1_\tau \# (U \# e) = U \# e$$
$$RHS = (1_\tau \circ U) \# e = U \# e$$

□

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Theorem

$[SWAP] \# (e_1, e_2) = (e_2, e_1)$

Proof.

???? □

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Structural equivalence $\sigma \leftrightsquigarrow \tau$:

$$\mathsf{swap}_{X,Y} : X \times Y \to Y \times X$$
$$\mathsf{swap}_{X,Y}(x, y) = (y, x)$$

Lift structural equivalence to unitary:

$$\widehat{\mathsf{swap}}_{\sigma,\tau} : \sigma \otimes \tau = \tau \otimes \sigma$$

such that

$$\widehat{\mathsf{swap}}_{\sigma,\tau} = [\mathsf{SWAP}_{\sigma,\tau}]$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$ *be a structural equivalence. Then*

$$\widehat{f} \# \mathit{init}_\sigma(b) \approx \mathit{init}_\tau(f(b))$$

Partial initialization:

$$\widehat{\mathsf{swap}_{X,Y}} \# (e_1, e_2) \approx \mathsf{swap}(e_1, e_2) = (e_2, e_1)$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

HoTT Quantum λ Calculus

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$ *be a structural equivalence. Then*

$$\widehat{f} \# \mathit{init}_\sigma(b) \approx \mathit{init}_\tau(f(b))$$

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$. *Then:*

$$\mathit{match}\ \widehat{f} \# e\ \mathit{with}\ g \approx \mathit{match}\ e\ \mathit{with}\ (g \circ f)$$

A HoTT Quantum Equational Theory. Paykin and Zdancewic, 2019.

QQTT?

QQTT?

- Quantum λ calculus = deep embedding in HoTT (univalence + groupoid quotients)
  - Would axioms be better in Cubical TT?

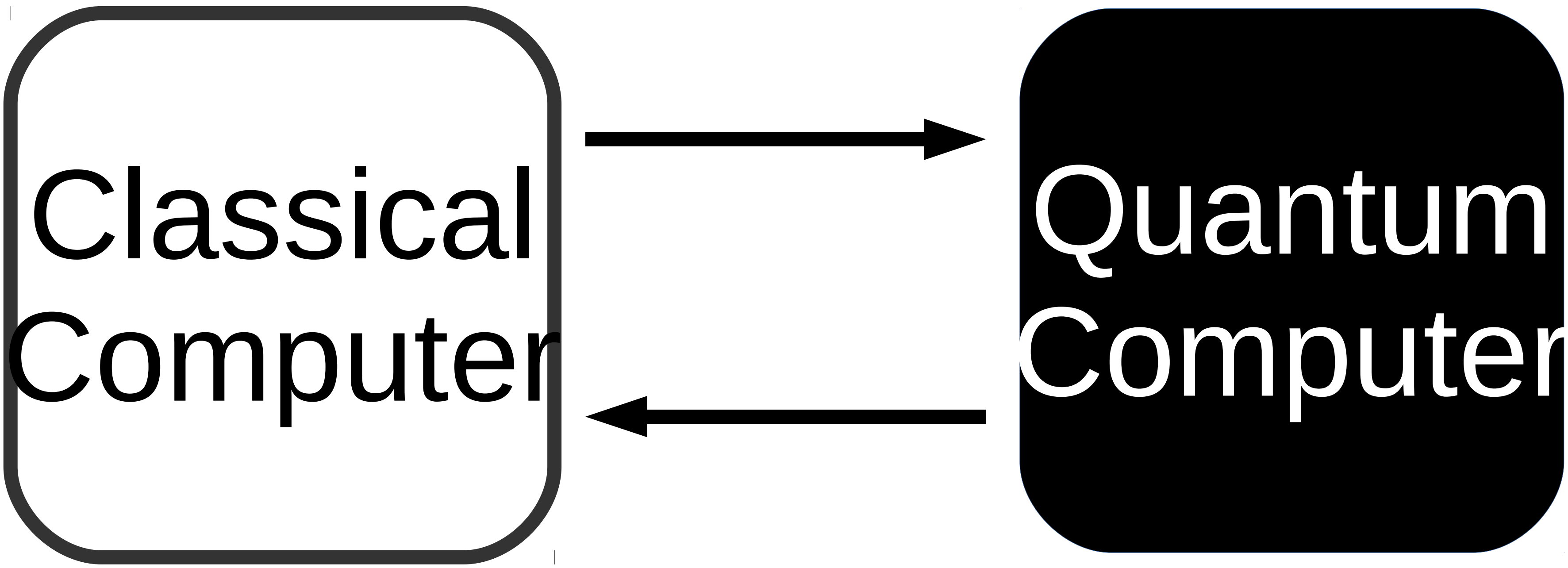
QQTT?

- Quantum λ calculus = deep embedding in HoTT (univalence + groupoid quotients)
  - Would axioms be better in Cubical TT?
- Quantitative HoTT?

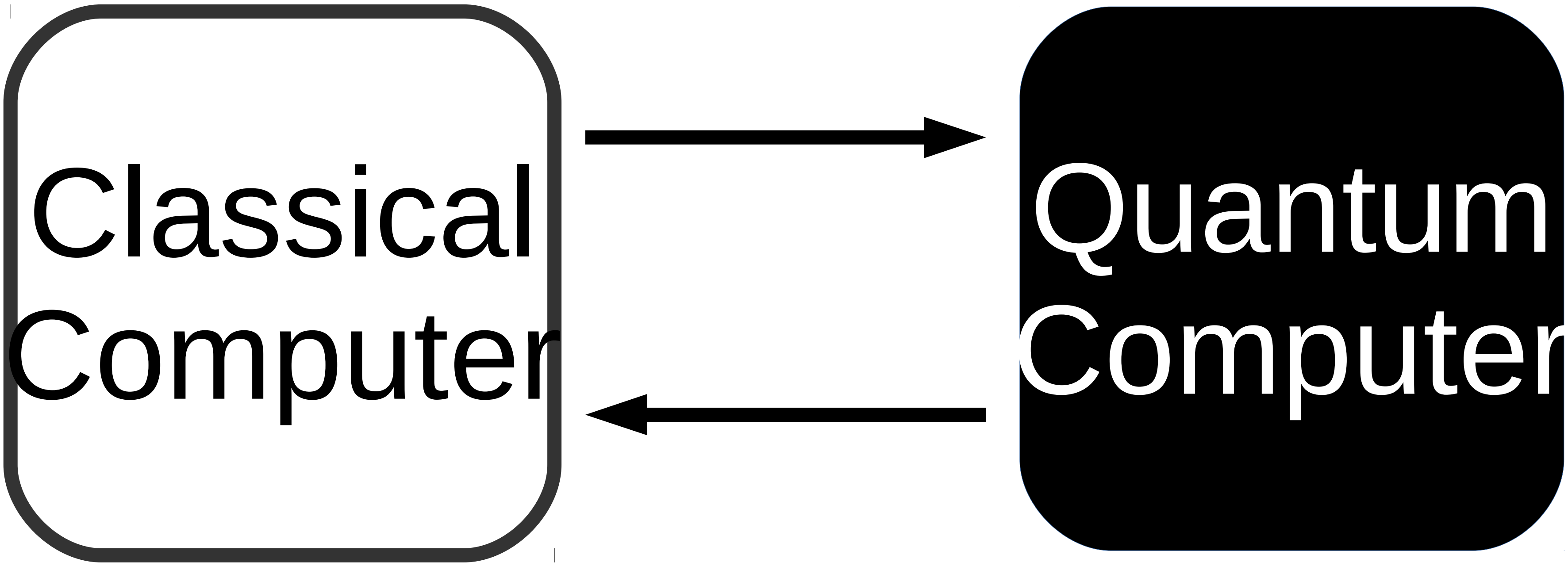
QQTT?

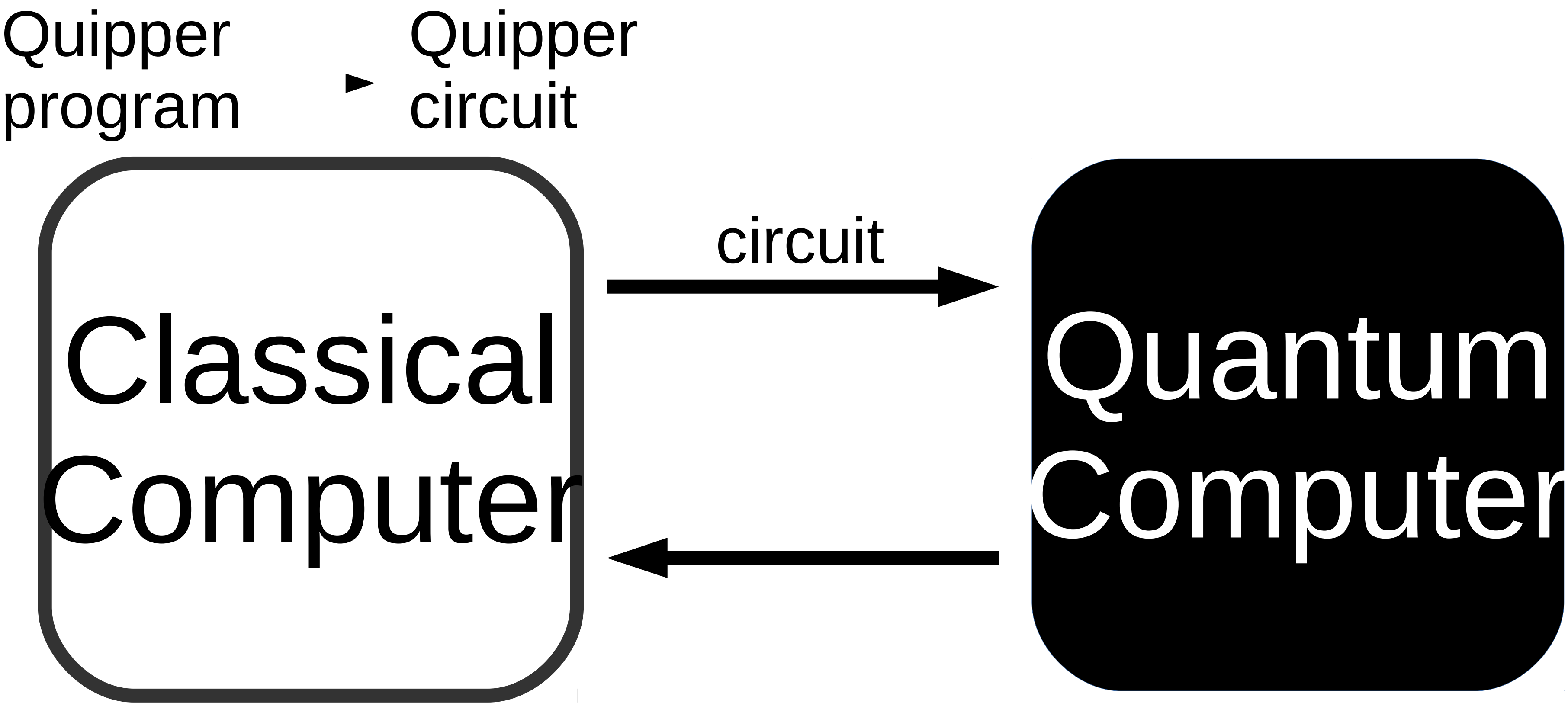
- Quantum λ calculus = deep embedding in HoTT (univalence + groupoid quotients)
  - Would axioms be better in Cubical TT?
- Quantitative HoTT?
- Quantum λ calculus = shallow embedding in QHoTT?

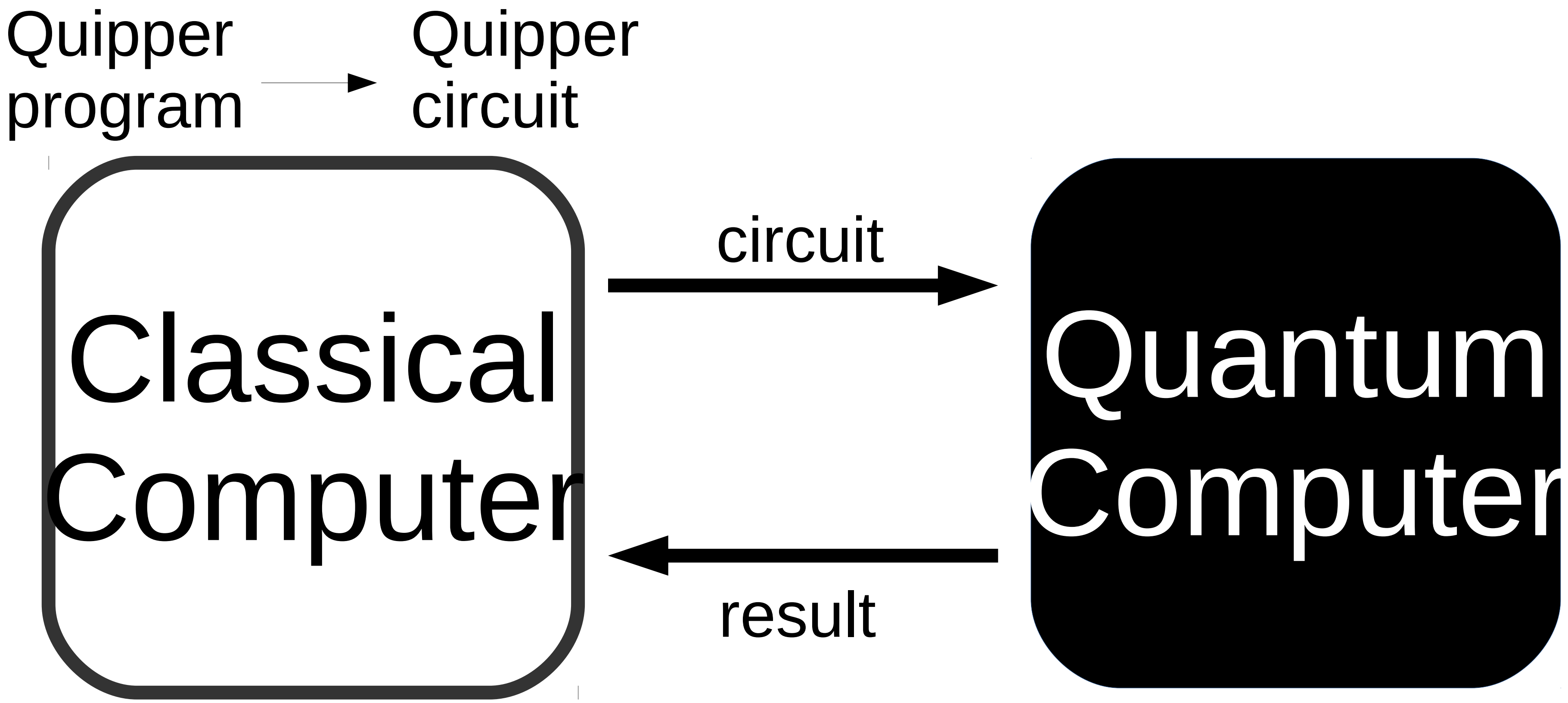
Technology Abstraction
Classical Communication

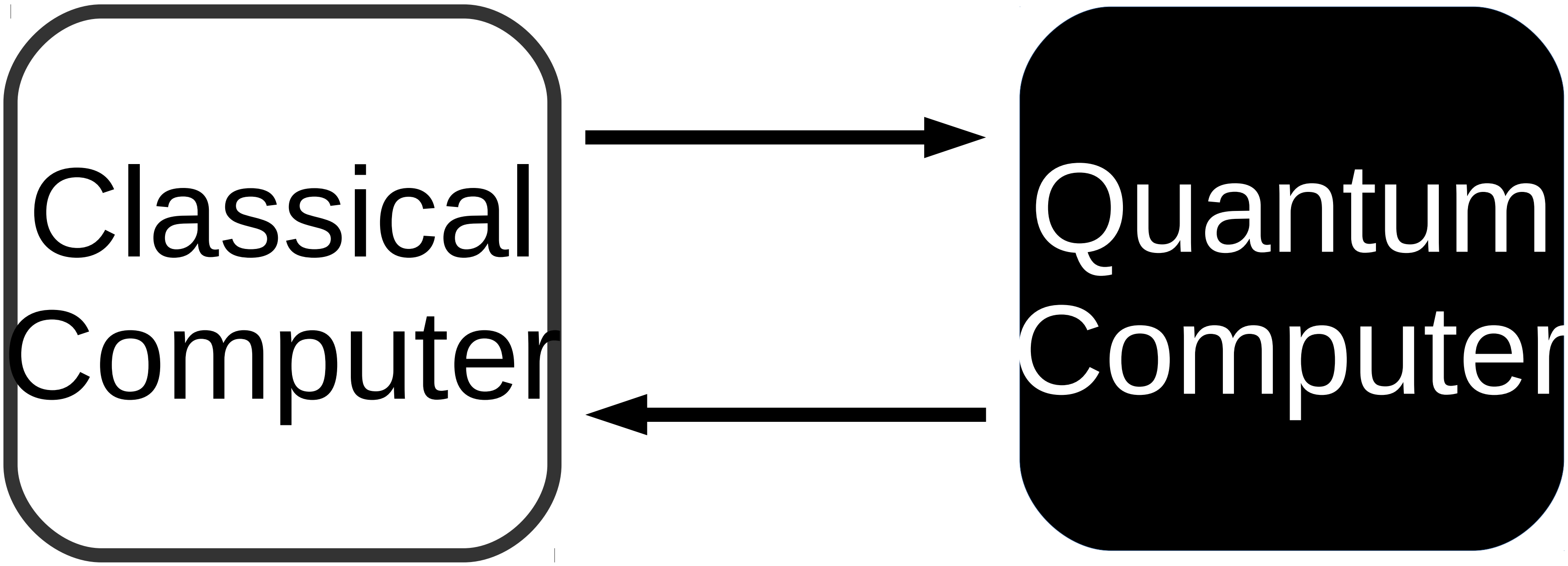

Classical Computer
Quantum Computer

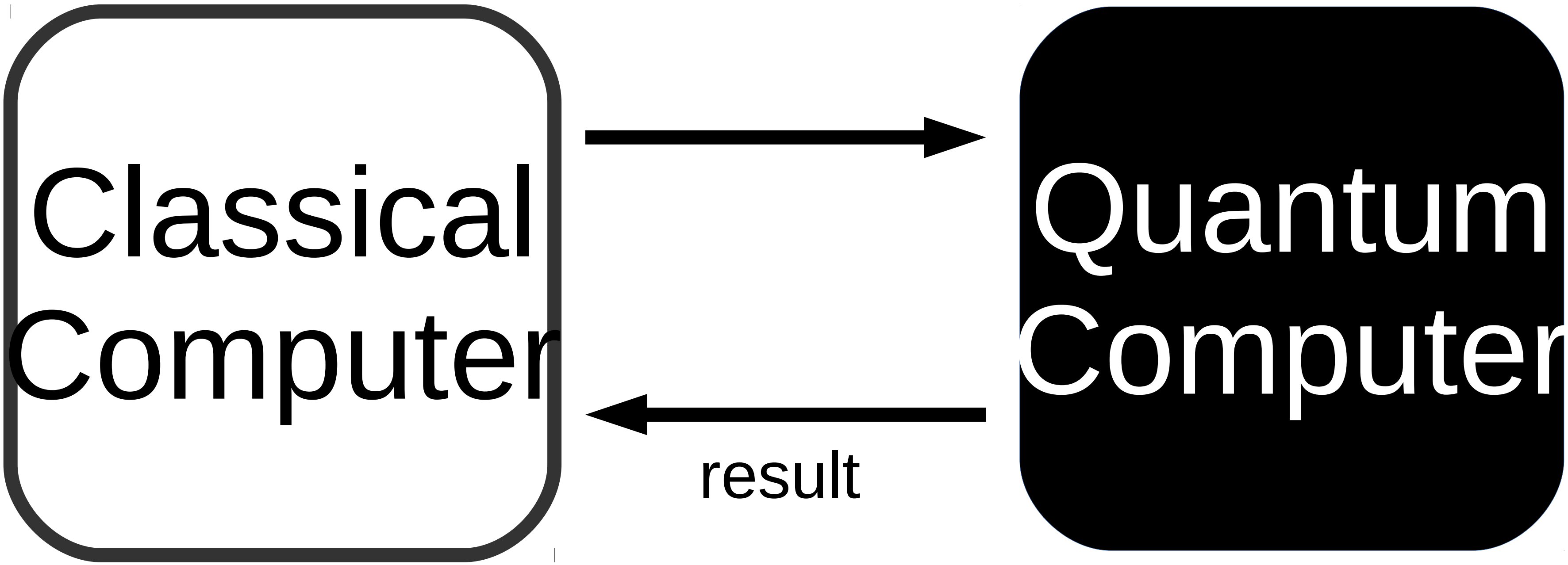
Quipper program → Quipper circuit

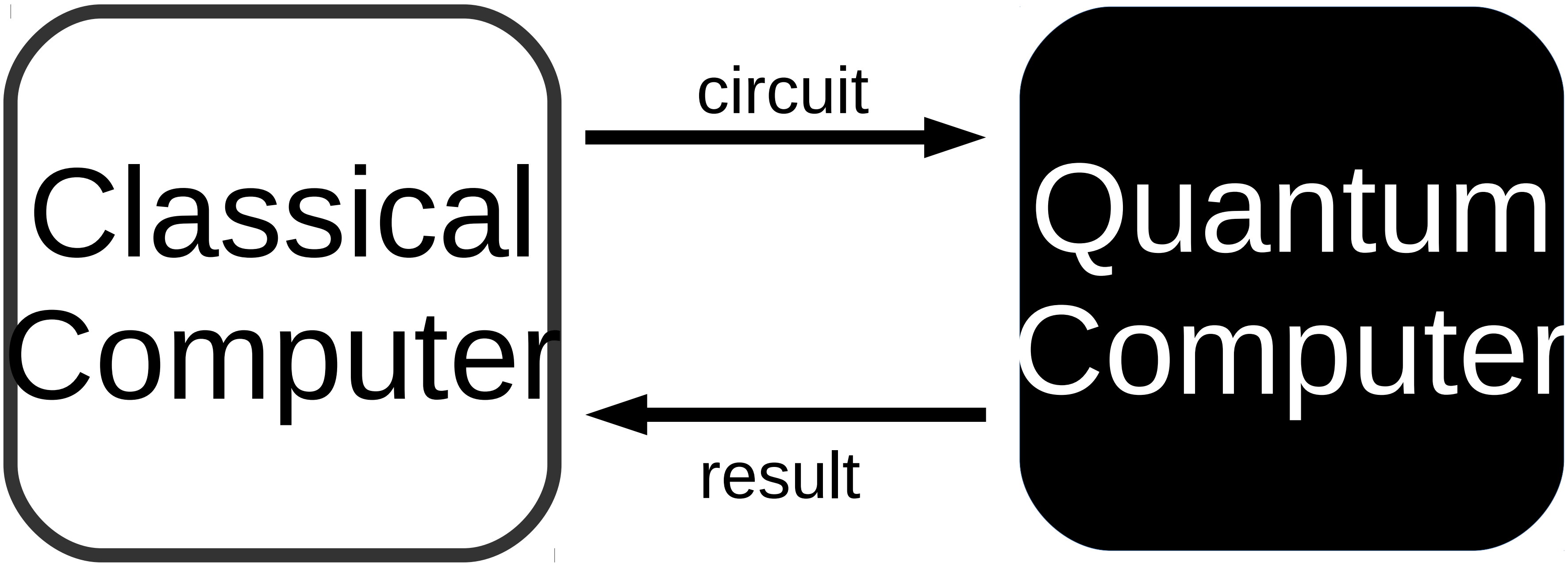


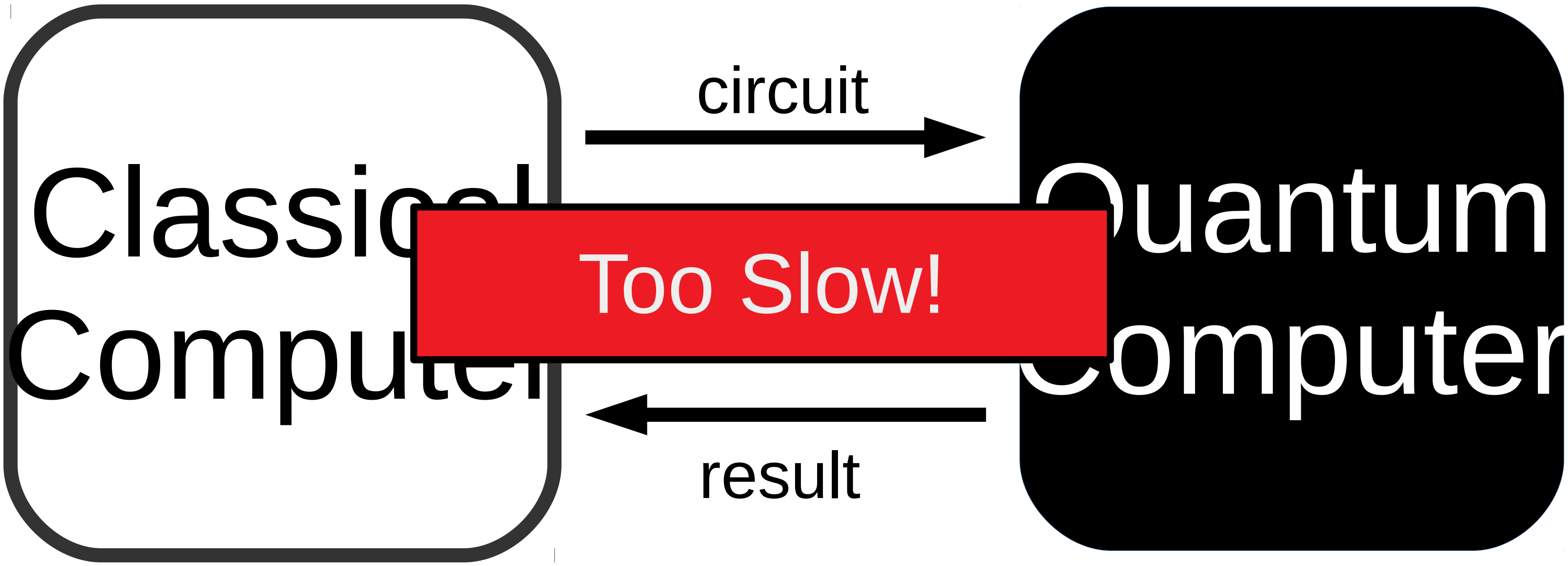

Quipper
program
Quipper
circuit
Classical
Computer
circuit
Quantum
Computer


Quipper
program
Quipper
circuit
Classical
Computer
circuit
result
Quantum
Computer


Classical Computer
Quantum Computer

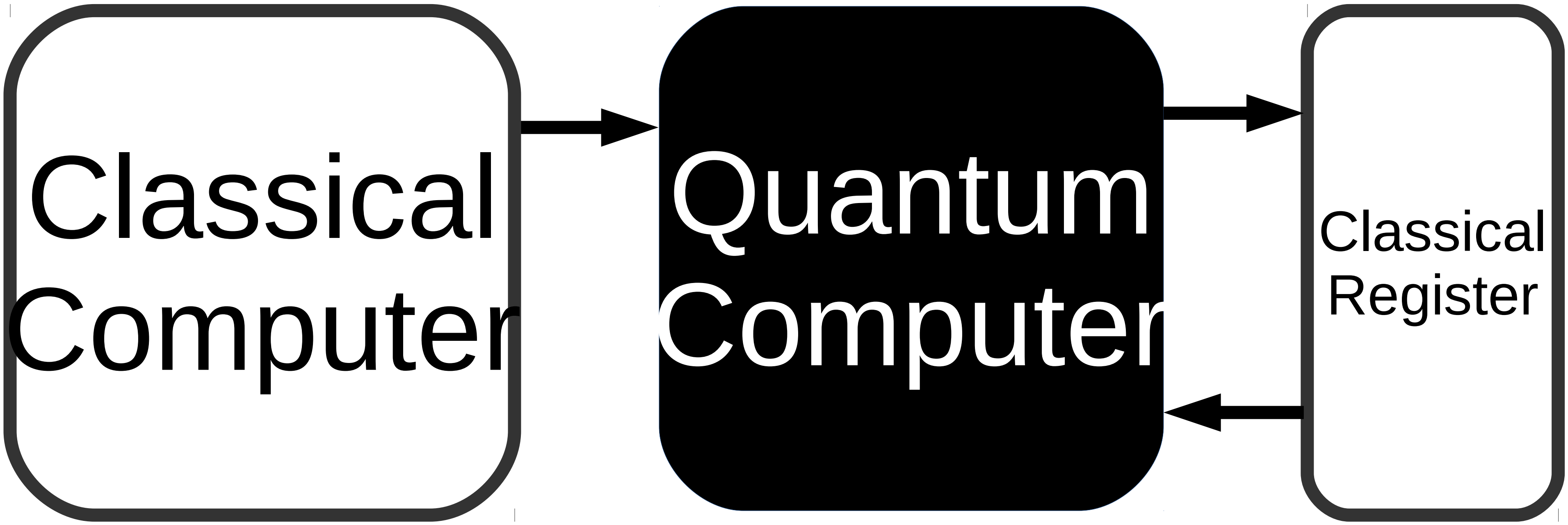

Classical Computer
Quantum Computer
result


Classical Computer
circuit
result
Quantum Computer


circuit
Classical Computer
Too Slow!
Quantum Computer
result

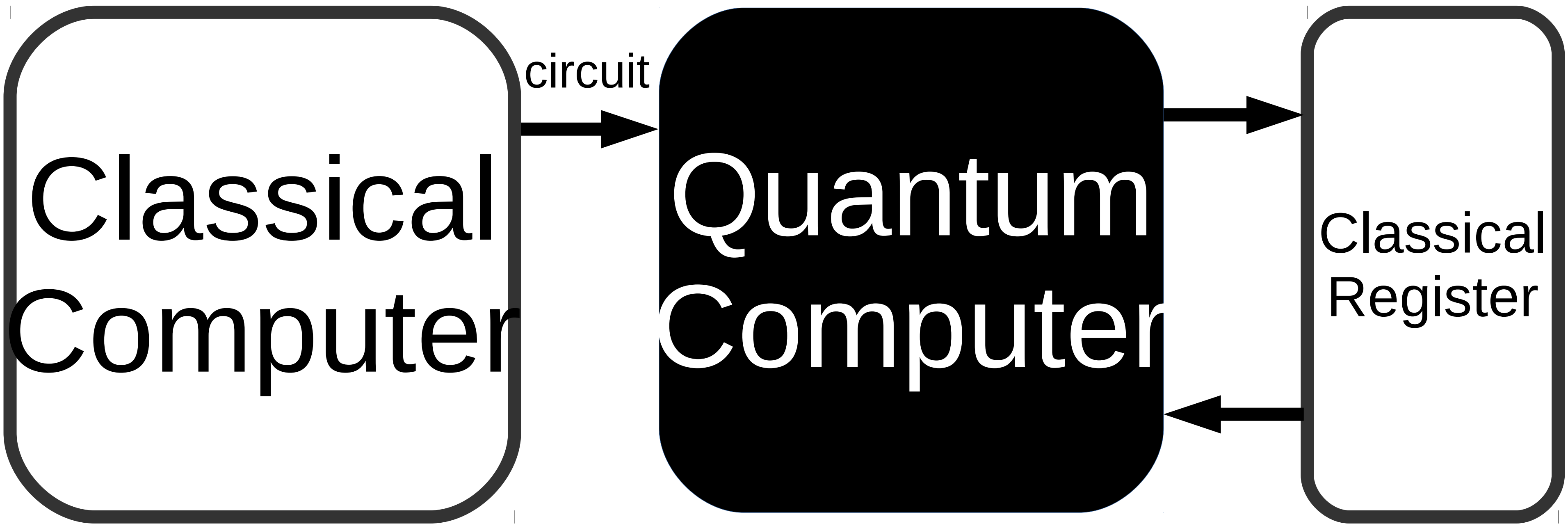
The QRAM Model

Knill, 1996



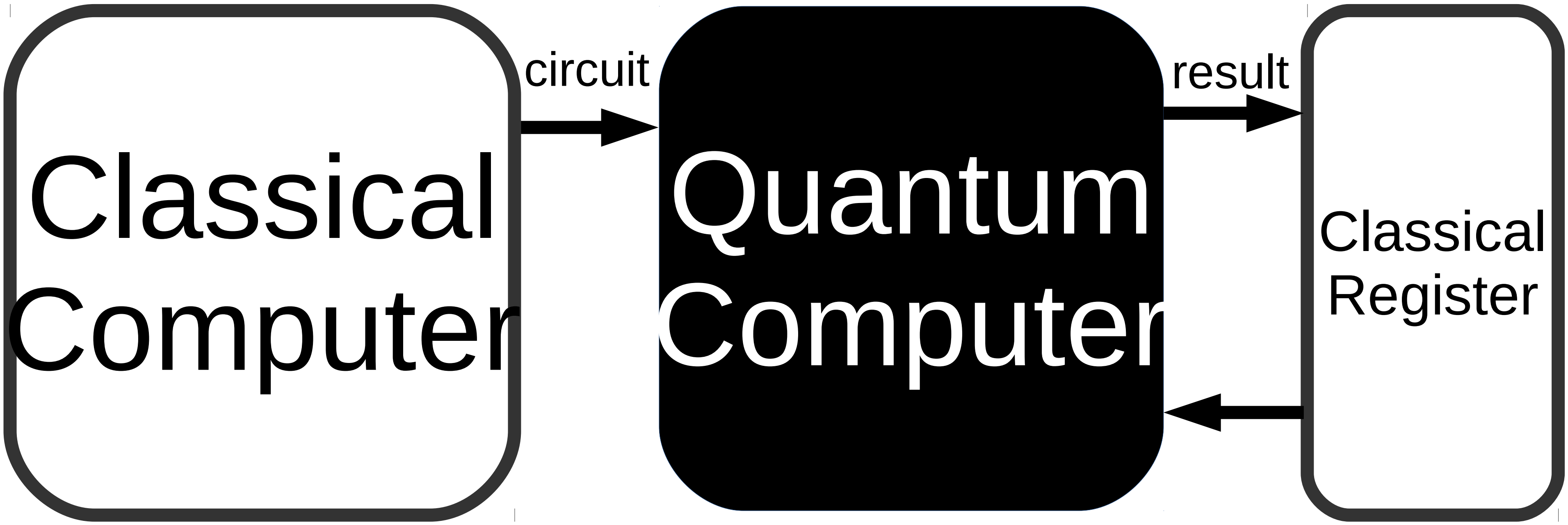
The QRAM Model

Knill, 1996



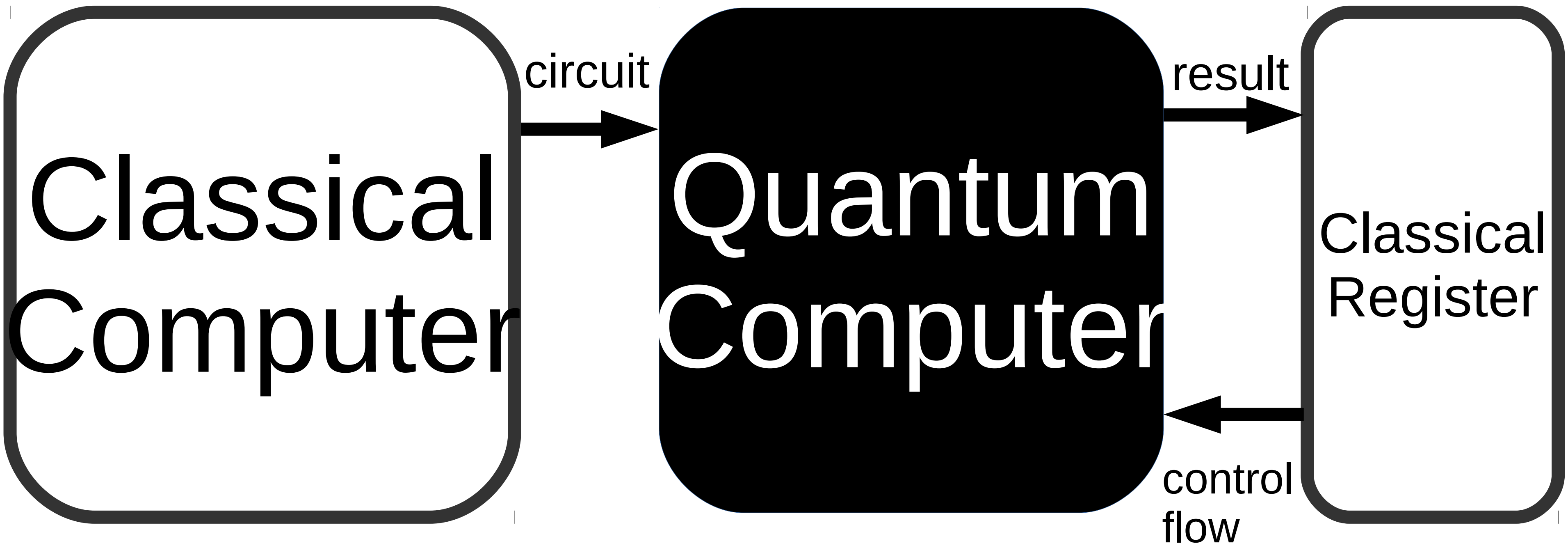
The QRAM Model

Knill, 1996



The QRAM Model

Knill, 1996



PL Abstraction
Control Flow

Yes classical control

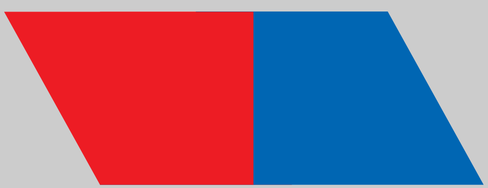
if meas()=1

then ... else ...

Yes classical control

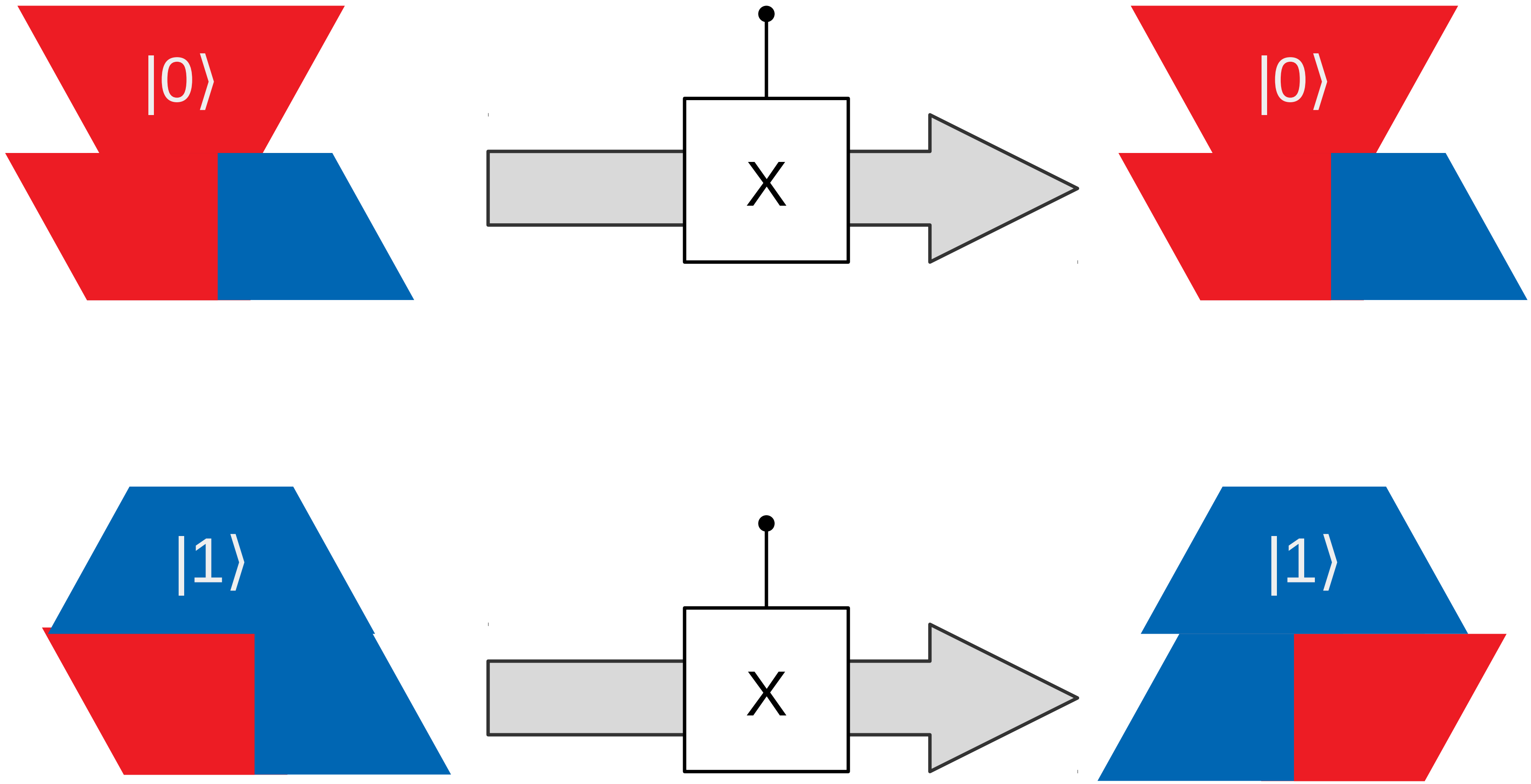
if meas()=1
then ... else ...

No* quantum control

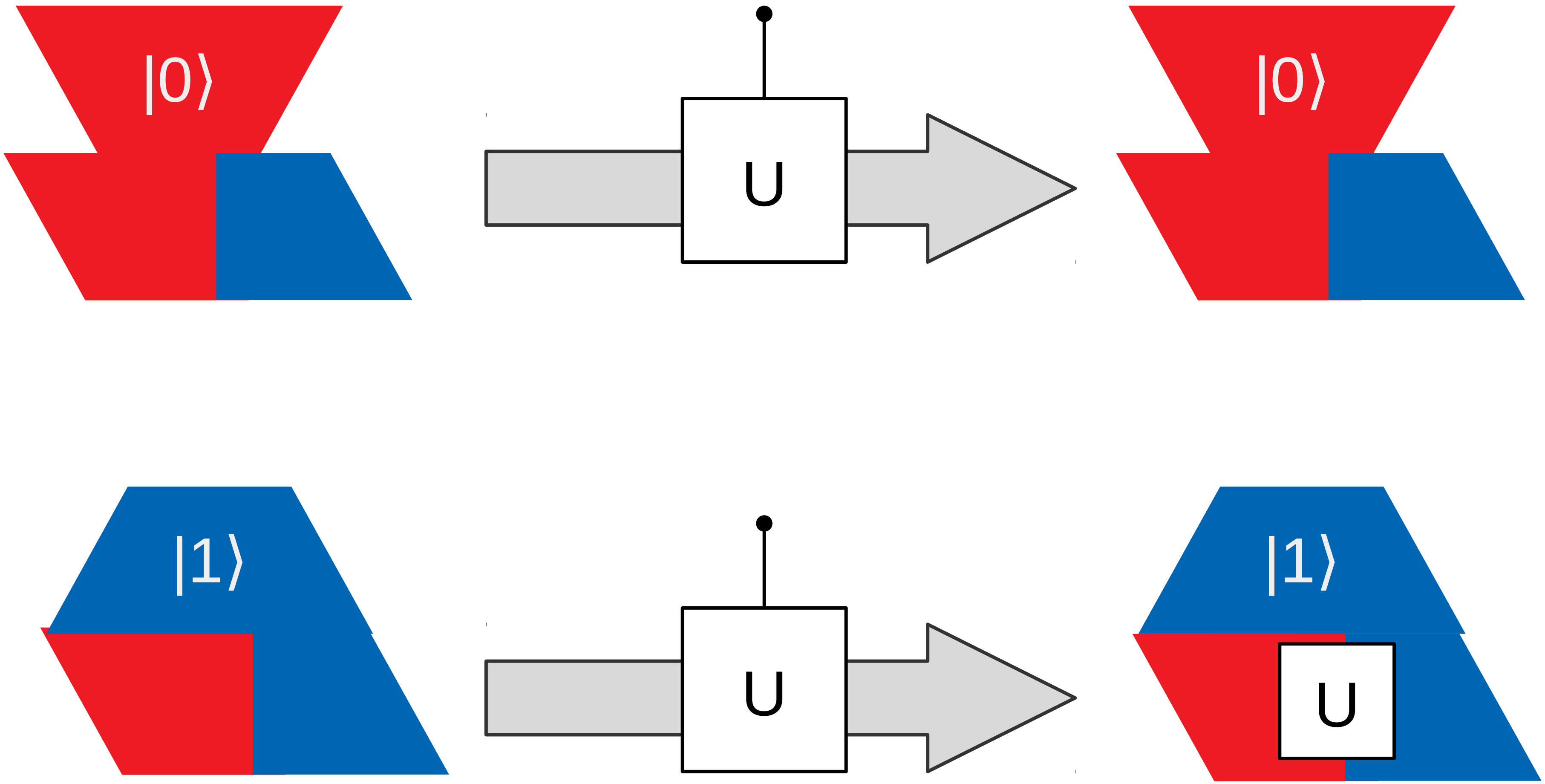
if  = 

then ... else ...

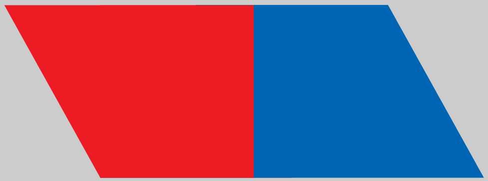
Controlled-NOT



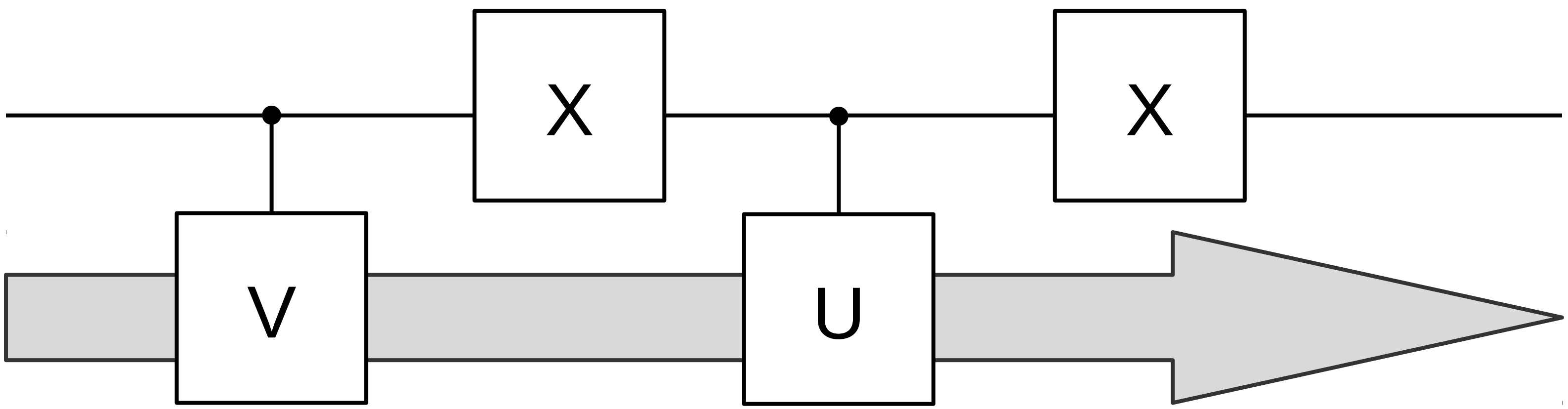
Controlled-U



Quantum unitary control

if  = 

then U else V



Quantum if, take 1

$$
\begin{aligned}
&qnot : \mathbf{Q_2} \multimap \mathbf{Q_2}\\
&qnot\ x = \mathbf{if}^\circ\ x\\
&\qquad\qquad \mathbf{then}\ \mathrm{qfalse}\\
&\qquad\qquad \mathbf{else}\ \mathrm{qtrue}
\end{aligned}
$$

$$
\begin{aligned}
&cnot : \mathbf{Q_2} \multimap \mathbf{Q_2} \multimap \mathbf{Q_2} \otimes \mathbf{Q_2}\\
&cnot\ c\ x = \mathbf{if}^\circ\ c\\
&\qquad\qquad \mathbf{then}\ (\mathrm{qtrue}, qnot\ x)\\
&\qquad\qquad \mathbf{else}\ (\mathrm{qfalse},\ x)
\end{aligned}
$$

$$
\begin{aligned}
&had \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}\\
&had\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ \{(-1)\ \mathrm{qtrue} \mid \mathrm{qfalse}\}\\
&\qquad\qquad\quad \mathbf{else}\ \ \{\mathrm{qtrue} \mid \mathrm{qfalse}\}
\end{aligned}
$$

QML: Quantum data and control. Altenkirch and Grattage, 2005

Quantum if, take q

$$\frac{\begin{array}{l}\Gamma \vdash^{a} c : \sigma \oplus \tau \\ \Delta,\ x : \sigma \vdash^{\circ} t : \rho \\ \Delta,\ y : \tau \vdash^{\circ} u : \rho \quad t \perp u\end{array}}{\Gamma \otimes \Delta \quad \vdash^{a} \quad \texttt{case}^{\circ} \ c \ \texttt{of} \ \{\texttt{inl}\ x \Rightarrow t \mid \texttt{inr}\ y \Rightarrow u\} : \rho} \oplus\text{-elim}^{\circ}$$

QML: Quantum data and control. Altenkirch and Grattage, 2005

Quantum if, take q

$$\dfrac{\begin{array}{l}\Gamma \vdash^a c : \sigma \oplus \tau \\ \Delta,\ x : \sigma \vdash^\circ t : \rho \\ \Delta,\ y : \tau \vdash^\circ u : \rho \quad t \perp u\end{array}}{\begin{array}{l}\Gamma \otimes \Delta \quad \vdash^a \quad \mathtt{case}^\circ \ \ c\ \mathtt{of} \\ \qquad\qquad\qquad \{\mathtt{inl}\ x \Rightarrow t \mid \mathtt{inr}\ y \Rightarrow u\} : \rho\end{array}} \oplus\!-\mathrm{elim}^\circ$$

$$\dfrac{t \perp u \quad \lambda_0^* \kappa_0 = -\lambda_1^* \kappa_1}{\{(\lambda_0)t \mid (\lambda_1)u\} \perp \{(\kappa_0)t \mid (\kappa_1)u\}}\ \mathrm{Osup}$$

QML: Quantum data and control. Altenkirch and Grattage, 2005

Pattern-matching isomorphisms

$$\texttt{not} : \mathbb{B} \leftrightarrow \mathbb{B} = \left(\begin{array}{ccc} \mathtt{ff} & \leftrightarrow & \mathtt{tt} \\ \mathtt{tt} & \leftrightarrow & \mathtt{ff} \end{array} \right),$$

$$\texttt{cnot} : \mathbb{B} \otimes \mathbb{B} \leftrightarrow \mathbb{B} \otimes \mathbb{B} = \left(\begin{array}{ccc} \langle \mathtt{ff}, x \rangle & \leftrightarrow & \langle \mathtt{ff}, x \rangle \\ \langle \mathtt{tt}, \mathtt{ff} \rangle & \leftrightarrow & \langle \mathtt{tt}, \mathtt{tt} \rangle \\ \langle \mathtt{tt}, \mathtt{tt} \rangle & \leftrightarrow & \langle \mathtt{tt}, \mathtt{ff} \rangle \end{array} \right)$$

$$\texttt{Had} : \mathbb{B} \leftrightarrow \mathbb{B}$$

$$\left(\begin{array}{ccc} \mathtt{tt} & \leftrightarrow & \frac{1}{\sqrt{2}}\mathtt{tt} + \frac{1}{\sqrt{2}}\mathtt{ff} \\ \mathtt{ff} & \leftrightarrow & \frac{1}{\sqrt{2}}\mathtt{tt} - \frac{1}{\sqrt{2}}\mathtt{ff} \end{array} \right)$$

From Symmetric Pattern-Matching to Quantum Control. Sabry, Valiron, Vizzotto, 2018.

Quantum if, take 2

$$
\begin{array}{lll}
\mathbf{qif}\ [\overline{q}]: & |1\rangle \rightarrow P_1 \\
\quad\square & |2\rangle \rightarrow P_2 \\
 & \cdots\cdots \\
\quad\square & |n\rangle \rightarrow P_n \\
\mathbf{fiq}
\end{array}
$$

Alternation in Quantum Programming: From Superposition of Data to Superposition of Programs. Ying, Yu, and Feng, 2014.

Quantum if, take 2

$$\begin{aligned}
&\textbf{qif } [\overline{q}] : && |1\rangle \rightarrow P_1 \\
&\quad \square && |2\rangle \rightarrow P_2 \\
& && \cdots\cdots \\
&\quad \square && |n\rangle \rightarrow P_n \\
&\textbf{fiq} &&
\end{aligned}$$

Let $P_1, P_2, ..., P_n$ be a collection of (quantum) programs whose state spaces are the same Hilbert space $\mathcal{H}$. We introduce a new family of quantum variables $\overline{q}$ that do not appear in $P_1, P_2, ..., P_n$.

Alternation in Quantum Programming: From Superposition of Data to Superposition of Programs. Ying, Yu, and Feng, 2014.

Alternation not compositional

$$\begin{aligned}
&\textbf{qif } [\overline{q}] : && |1\rangle \rightarrow P_1 \\
&\quad \square && |2\rangle \rightarrow P_2 \\
& && \cdots\cdots \\
&\quad \square && |n\rangle \rightarrow P_n \\
&\textbf{fiq}
\end{aligned}$$

$$[\![P_1]\!] = [\![P_1']\!] \wedge [\![P_2]\!] = [\![P_2']\!] \not\Rightarrow [\![P_1]\!] \bullet [\![P_2]\!] = [\![P_1']\!] \bullet [\![P_2']\!]$$

Quantum Alternation: Prospects and Problems. Badescu and Panangaden, 2015

Takeaways

Sources of Abstractions

Physics

- No-cloning
- Reversible
- Superposition
- Measurement

Computing Technology

- Circuit model
- Classical communicati

Algorithms

- Classical oracles
- Amplification

PL Theory

- Data structures
- Control flow

Semantics

- Quantum CPOs
- String diagrams

How to motivate engineers?

How to proceed when the abstractions you have are unsatisfactory?