Proving Quantum Programs Correct

Kesha Hietala
University of Maryland
kesha@cs.umd.edu

Robert Rand
University of Chicago
rand@uchicago.edu

Shih-Han Hung
University of Maryland
shung@umd.edu

Liyi Li
University of Maryland
liyili2@umd.edu

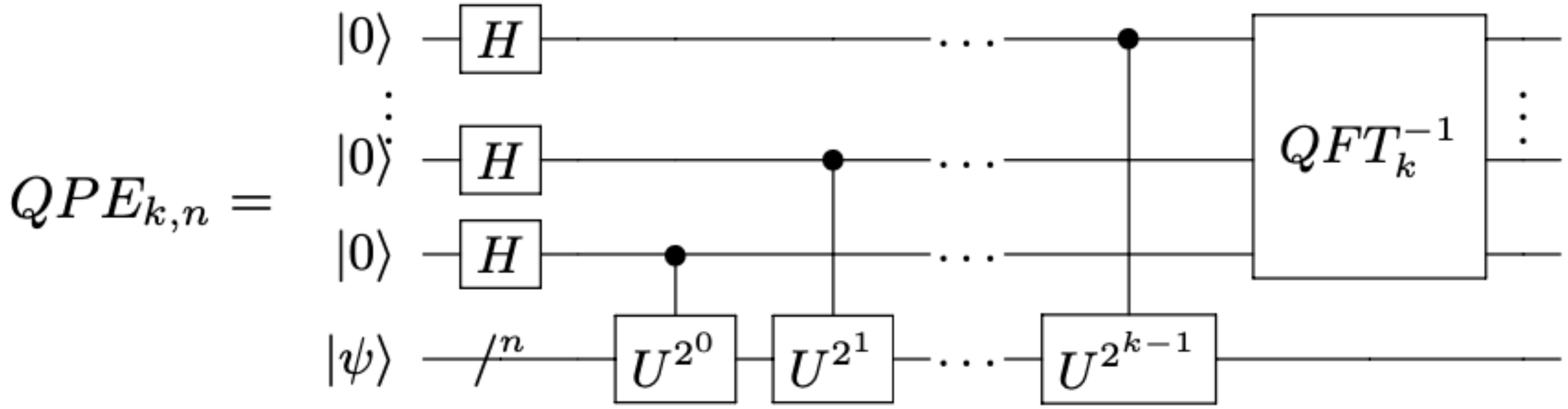
Michael Hicks
University of Maryland
mwh@cs.umd.edu

ITP 2021

Image from https://www.ibm.com/quantum-computing/

Writing Quantum Programs is Hard

- Quantum indeterminacy ⇒ quantum programs are **probabilistic**
- Quantum programs are written as **circuits**



- Quantum programs use **new primitives**
  - E.g. “prepare a uniform superposition”, “perform a Fourier transform”

Writing (Correct) Quantum Programs is Hard

- In general
  - What is “correct?” Answer may be approximate
  - Breakpoints break things (opening the box kills the cat)
  - Simulating quantum programs is intractable

- In the near term
  - Computing resources (e.g., qubits) are scarce
  - Execution is error prone

Formal verification can help!

SQIR

- SQIR is a **S**imple **Q**uantum **I**ntermediate **R**epresentation for expressing quantum circuits + libraries for reasoning about quantum programs in the *Coq Proof Assistant*

- Presented as the intermediate representation of a verified compiler (à la CompCert) at POPL 2021 (arxiv:1912.02250)

- Our ITP paper looks at using SQIR as a *source* language for verified quantum programming

- Code available at github.com/inQWIRE/SQIR

This Talk

- **Intro to Quantum**
- SQIR Syntax & Semantics
- Proof Engineering
- Results

Qubits

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$|0\rangle \qquad\qquad\qquad |1\rangle$$

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

Superposition: Qubits can be in multiple states (0 or 1) at once

Measurement

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

$$|\alpha|^2 \qquad |\beta|^2$$

$$\begin{pmatrix}1\\ 0\end{pmatrix} \qquad \begin{pmatrix}0\\ 1\end{pmatrix}$$

$$|0\rangle \qquad |1\rangle$$

Measurement: Looking at a qubit probabilistically turns it into a bit.

Measurement

$|+\rangle$ state

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$\left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2} \qquad \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$$

$$\begin{pmatrix}1\\ 0\end{pmatrix} \qquad \begin{pmatrix}0\\ 1\end{pmatrix}$$

$|0\rangle$ $\qquad$ $|1\rangle$

Measurement: Looking at a qubit probabilistically turns it into a bit.

Operators

A unitary operator transforms, or *evolves*, a state

$$H \; |0\rangle = |+\rangle$$

$$H \; |+\rangle = |0\rangle$$

This is the *Hadamard* operator, H

(which is its own inverse)

Operators

Operators are represented as *unitary matrixes*

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix}$$

Measurement 2.0

$$\begin{tikzcd} & \begin{pmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{pmatrix} \arrow[dl, bend right, "\frac{1}{2}"'] \arrow[dr, bend left, "\frac{1}{2}"] & \\ \begin{pmatrix}1\\0\\0\\0\end{pmatrix} & & \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \end{tikzcd}$$

Measurement 2.0

$$\begin{tikzcd}[column sep=small] & \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} \arrow[dl, bend right, "\frac{1}{2}"'] \arrow[dr, bend left, "\frac{1}{2}"] & \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} & & \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ |00\rangle & & |11\rangle \end{tikzcd}$$

Entanglement

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \qquad\qquad \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad\qquad ? \otimes ?$$

Entangled qubits are not probabilistically independent — they cannot be decomposed. Connection at a distance!

Multi-Qubit Unitaries

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\text{CNOT}\,|+0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

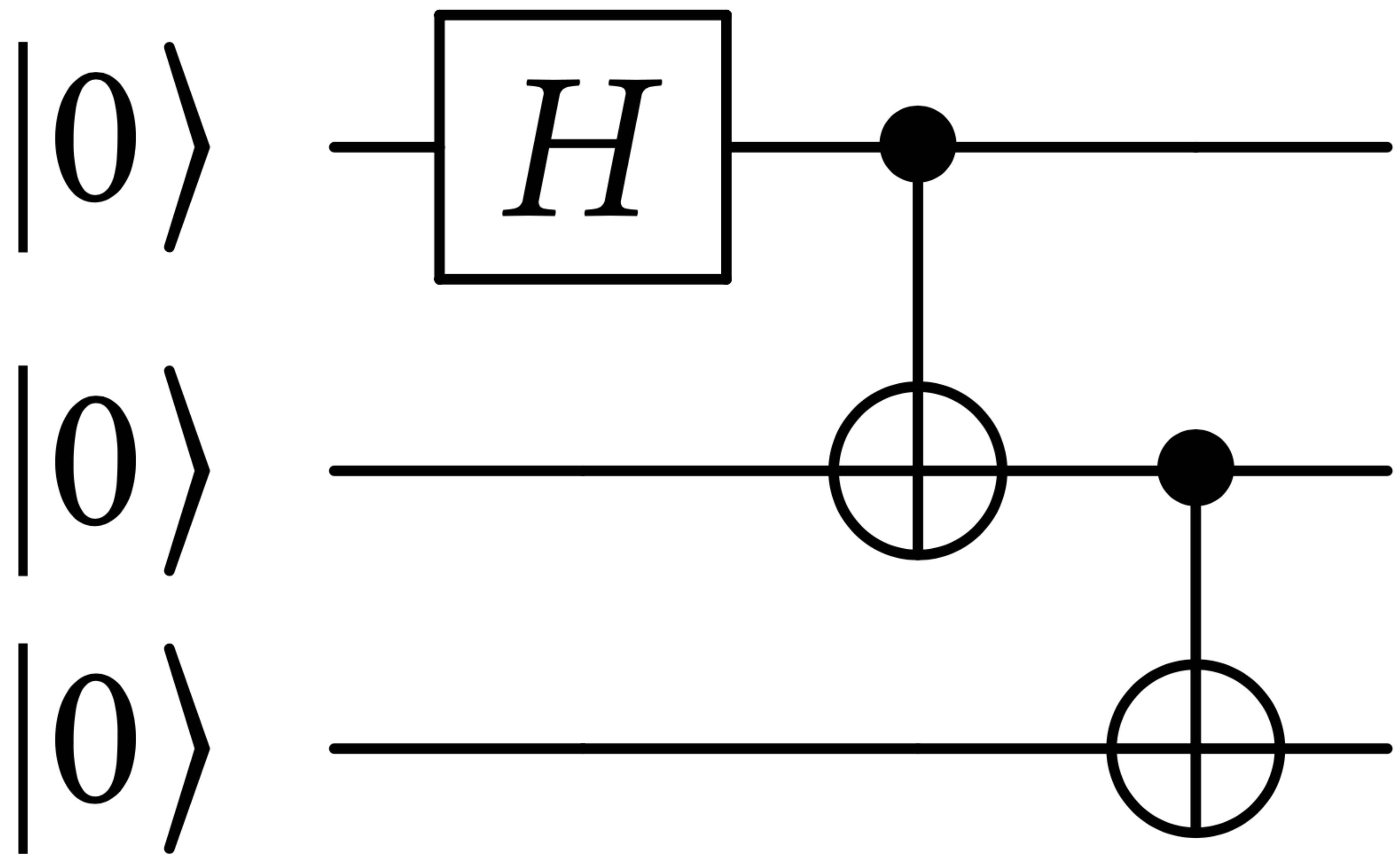
General Quantum States

- So far we have seen *pure states*
  - *E.g.* $|0\rangle, |1\rangle, |+\rangle$
- A *mixed state* is a (classical) probability distribution over pure states
  - E.g. $\begin{cases} |0\rangle & \text{with probability 1/2} \\ |1\rangle & \text{with probability 1/2} \end{cases}$
- Density matrices allow us to describe both pure and mixed states

$$\rho = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad \rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

Circuits

Quantum programs are often written as circuits




```
H 0
CNOT 0 1
CNOT 1 2
```

This Talk

- Intro to Quantum
- **SQIR Syntax & Semantics**
- Proof Engineering
- Results

Unitary SQIR

- Semantics parameterized by *gate set G* and *dimension d of a global register*

$$U \;::=\; U_1;\; U_2 \mid G\; q \mid G\; q_1\; q_2$$

E.g. $apply_1(X, q, d) = I_{2^q} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes I_{2^{(d-q-1)}}$

- The denotation (semantics) of *U* is a $2^d \times 2^d$ unitary matrix

$$[\![U_1;\; U_2]\!]_d = [\![U_2]\!]_d \times [\![U_1]\!]_d$$

$$[\![G_1\; q]\!]_d = \begin{cases} apply_1(G_1,\; q,\; d) & \text{well-typed} \\ 0_{2^d} & \text{otherwise} \end{cases}$$

$q < d$

$$[\![G_2\; q_1\; q_2]\!]_d = \begin{cases} apply_2(G_2,\; q_1,\; q_2,\; d) & \text{well-typed} \\ 0_{2^d} & \text{otherwise} \end{cases}$$

$q_1 < d \;\wedge\; q_2 < d \;\wedge\; q_1 \neq q_2$

Non-Unitary SQIR

- Semantics parameterized by *gate set G* and *dimension d of a global register*

$$P \ ::= \ \mathtt{skip} \mid P_1;\ P_2 \mid U \mid \mathtt{meas}\ q\ P_1\ P_2$$

- The denotation of *P* is a function over $2^d \times 2^d$ density matrices

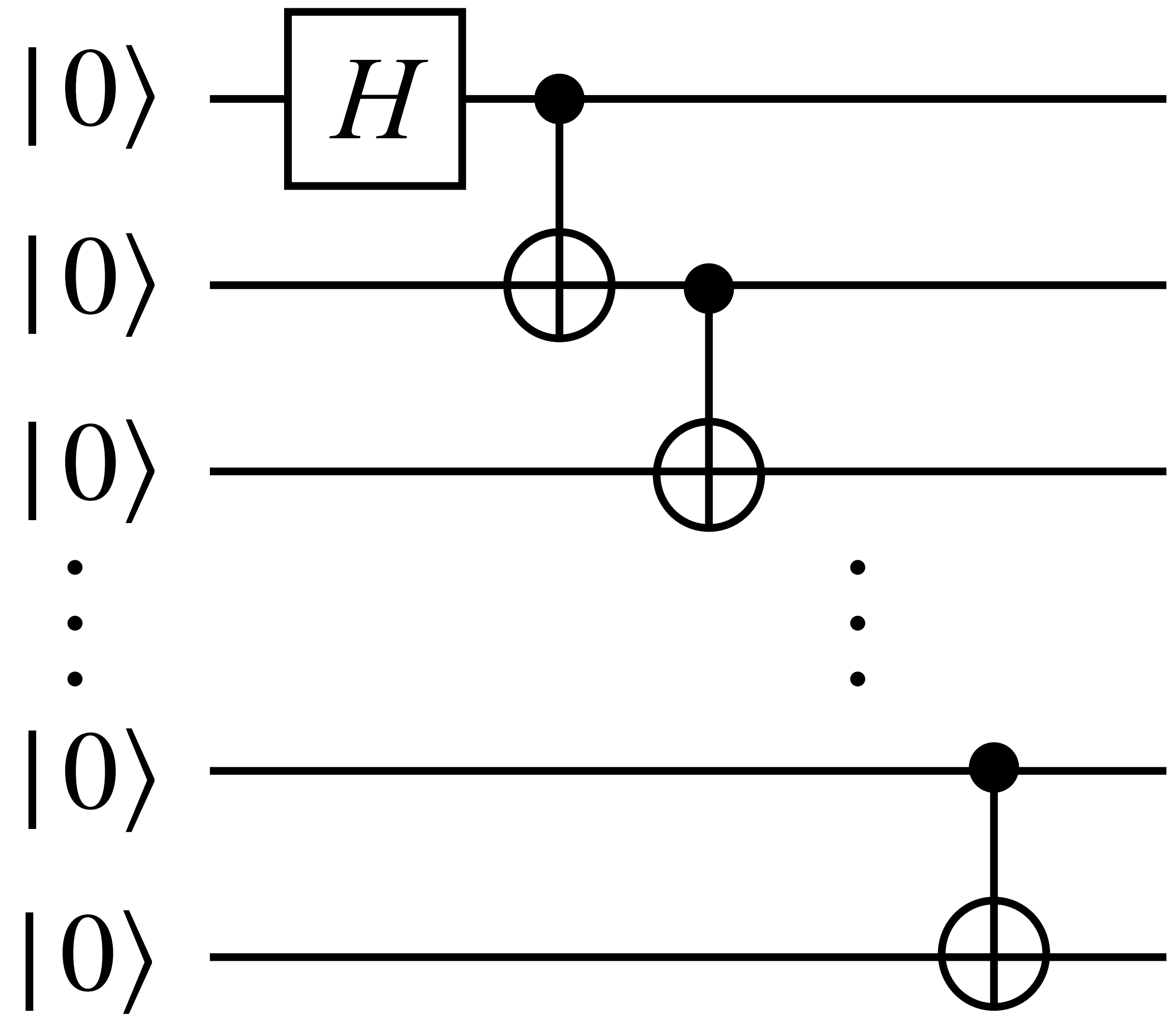
$$\begin{aligned}
\{\!|\mathsf{skip}|\!\}_d(\rho) &= \rho \\
\{\!|P_1;\ P_2|\!\}_d(\rho) &= (\{\!|P_2|\!\}_d \circ \{\!|P_1|\!\}_d)(\rho) \\
\{\!|U|\!\}_d(\rho) &= [\![U]\!]_d \times \rho \times [\![U]\!]_d^\dagger \\
\{\!|\mathsf{meas}\ q\ P_1\ P_2|\!\}_d(\rho) &= \{\!|P_2|\!\}_d(|0\rangle_q\langle 0| \times \rho \times |0\rangle_q\langle 0|) \\
&\quad + \{\!|P_1|\!\}_d(|1\rangle_q\langle 1| \times \rho \times |1\rangle_q\langle 1|)
\end{aligned}$$

Standard semantics; also used in QHL[1] and QWIRE[2]

[1] Ying. *Floyd-Hoare logic for quantum programs*. TOPLAS 2012.
[2] Paykin et al. *QWIRE: A core language for quantum circuits*. POPL 2017.

SQIR Metaprogramming

- SQIR programs just express circuits. We can express parameterized circuit families using Coq as a meta programming language




```
Fixpoint ghz (n : ℕ) : ucom base n :=
  match n with
  | 0 ⇒ SKIP
  | 1 ⇒ H 0
  | S n' ⇒ ghz n'; CNOT (n'-1) n'
  end.
```

- The `ghz` Coq function returns a SQIR program (of type `ucom base n`) whose semantics is the `n`-qubit GHZ state

Proofs of Correctness in Coq

- We might like to prove that evaluating `ghz n` on $|0\rangle^{\otimes n}$ produces $|GHZ^n\rangle$
  - where $|GHZ^n\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n})$

```
Definition GHZ (n : ℕ) : Vector (2 ^ n) :=
  match n with
  | 0     ⇒ I 1
  | S n' ⇒ 1/√2 * |0⟩^⊗n + 1/√2 * |1⟩^⊗n
  end.

Lemma ghz_correct : ∀ n : ℕ,
  n > 0 → ⟦ghz n⟧_n × |0⟩^⊗n = GHZ n.
Proof.
...
Qed.
```

This Talk

- Intro to Quantum
- SQIR Syntax & Semantics
- **Proof Engineering** ⟵ the focus of our paper
- Results

SQIR Design Highlights

- Reference **qubits using concrete indices** (`CNOT (n-1) n` vs. `CNOT x y`)
  - Semantics just maps to the proper column/row in the matrix
  - Disjointness is *syntactic*; important for well-formedness
- **Separate the unitary core from the full language** with measurement
  - Unitary matrix semantics simpler than *density matrix* formulation (but can use the latter when needed)
  - Allows representing quantum state using a vector, which enables better automation
- See our paper for more!

Vector States

- *apply₁* and *apply₂* become unwieldy for expressions with many qubits

$$apply_1(X, q, d) = I_{2^q} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes I_{2^{(d-q-1)}}$$

$$apply_2(CNOT, q_1, q_2, d) = \begin{cases} I_{2^{q_1}} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes I_{2^{q_2-q_1-1}} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes I_{2^{d-q_2-1}} \; + \; I_{2^{q_1}} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes I_{2^{d-q_1-1}} & \text{for } \; q_1 < q_2 \\ I_{2^{q_2}} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes I_{2^{q_1-q_2-1}} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes I_{2^{d-q_1-1}} \; + \; I_{2^{q_2}} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes I_{2^{d-q_2-1}} & \text{for } \; q_2 < q_1 \end{cases}$$

  - We provide automation for simplifying products of *apply* terms to *grid normal form*
  - But the normalized terms can be quite large & have many cases to account for different orderings of qubit arguments

Vector States

- It's simpler to describe a unitary gate by its effect on *basis vectors*

$$\texttt{X a}\colon \ |\ldots x \ldots\rangle \mapsto |\ldots (\neg x) \ldots\rangle$$

$$\texttt{CNOT a b}\colon \ |\ldots x \ldots y \ldots\rangle \mapsto |\ldots x \ldots (x \oplus y) \ldots\rangle$$

- Basis vectors alone aren't enough to represent all quantum states
  $\rightarrow$ we provide a construct for describing sums over vectors

- Measurement is not unitary
  $\rightarrow$ we provide *measurement predicates* like `probability_of_outcome`

Related Work

- QWIRE *[Rand et al., QPL 2017]*
  - Implemented in Coq
  - Used to verify simple randomness generation circuits and small examples
- QBRICKS *[Chareton et al., ESOP 2021]*
  - Implemented in Why3
  - Used to verify Grover's algorithm and Quantum Phase Estimation
- Quantum Hoare Logic (QHL) *[Liu et al., CAV 2019]*
  - Implemented in Isabelle/HOL
  - Used to verify Grover's algorithm

Related Work

	QWIRE	QBRICKS	QHL	SQIR
Uses concrete indices		✓	✓	✓
Special support for unitary programs		✓		✓
General support for measurement	✓		✓	✓

SQIR is *flexible*, supporting multiple semantics and approaches to proof

This Talk

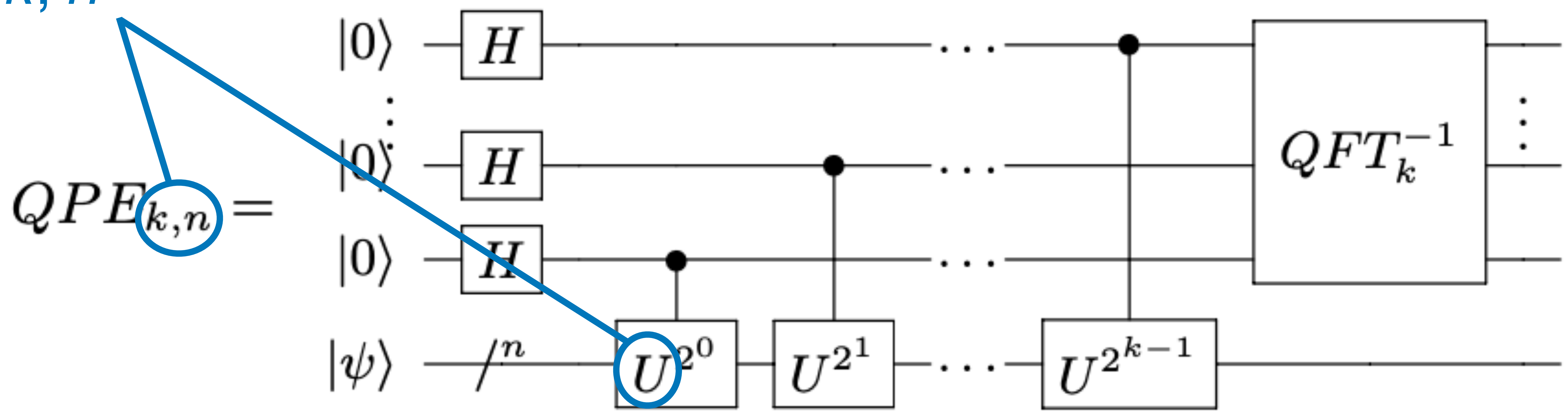
- Intro to Quantum
- SQIR Syntax & Semantics
- Proof Engineering
- **Results**

Proofs so Far

- We have formally verified several source programs correct
  - Quantum teleportation / superdense coding
  - GHZ state preparation
  - Deutsch-Jozsa algorithm
  - Simon's algorithm
  - Grover's search algorithm
  - Quantum phase estimation
- These proofs constitute about 3.5k lines of Coq (core of SQIR is 3.9k)
- Our specifications and proofs follow the standard textbook arguments

Example: QPE

parameterized by *U*, *k*, *n*



- **Q**uantum **P**hase **E**stimation: given a circuit implementing some unitary U and an state $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, find θ
  - The key "quantum" part of Shor's factoring algorithm
  - The most sophisticated quantum algorithm verified by any current tool
- The SQIR implementation is 40 lines and the proof is 1000 lines
  - Proof completed in two person-weeks

Example: QPE

- Correctness property in the case where θ can be represented using exactly *k* bits (call this representation *z*):

```
Lemma QPE_correct_simplified: ∀ k n (u : ucom base n) z (ψ : Vector 2^n),
  n > 0 → k > 1 → uc_well_typed u → WF_Matrix ψ →
  let θ := z / 2^k in
  ⟦u⟧_n × ψ = e^(2πiθ) * ψ →
  ⟦QPE k n u⟧_(k+n) × (|0⟩^k ⊗ ψ) = |z⟩ ⊗ ψ.
```

- Conclusion says that the running `QPE` on the input $|00\ldots0\rangle \otimes |\psi\rangle$ produces *z* in the first *k* bits

Example: QPE

- If θ can not be exactly expressed using *k* bits, we get an approximation within $\frac{1}{2^{k+1}}$ of the true value with probability at least $\frac{4}{\pi^2} \approx 0.41$

```
Lemma QPE_semantics_full : ∀ k n (u : ucom base n) z (ψ : Vector 2^n) (δ : R),
  n > 0 → k > 1 → uc_well_typed u → Pure_State_Vector ψ →
  -1 / 2^(k+1) ≤ δ < 1 / 2^(k+1) → δ ≠ 0 →
  let θ := z / 2^k + δ in
  [[u]]_n × ψ = e^(2πiθ) * ψ →
  prob_partial_meas |z⟩ ([[QPE k n u]]_(k+n) × (|0⟩^k ⊗ ψ)) ≥ 4 / π^2.
```

Future Directions

- Extract verified SQIR programs to executable OpenQASM circuits
  - Requires careful thought about gate sets and the implementation of "control" and "adjoint" functions to produce reasonably efficient code
- Verify near-term quantum algorithms
  - Requires better handling for *approximate* algorithms
  - May need to account for errors → requires density matrices
- Higher-level abstractions for describing quantum programs and specifications?

Conclusions

- Formal verification for quantum programs is a recent area of interest
  - Recent work includes QWIRE, QBRICKS, QHL
  - SQIR is one of the most successful examples to date
  - This is an open field!
- GitHub repository: github.com/inQWIRE/SQIR

Pull requests welcome!

- Full version of the ITP paper: arxiv:2010.01240
- POPL 2021 paper on optimizing SQIR programs: arxiv:1912.02250