HPCA 2025 Tutorial

Topic 3. MorphQPV: Exploiting Isomorphism in Quantum Programs to Facilitate Confident Verification

ACES Lab

JanusQ Cloud

Speaker: Siwei Tan

College of Computer Science and Technology
Zhejiang University (ZJU)

https://janusq.github.io/HPCA_2025_Tutorial/

Outline of Presentation

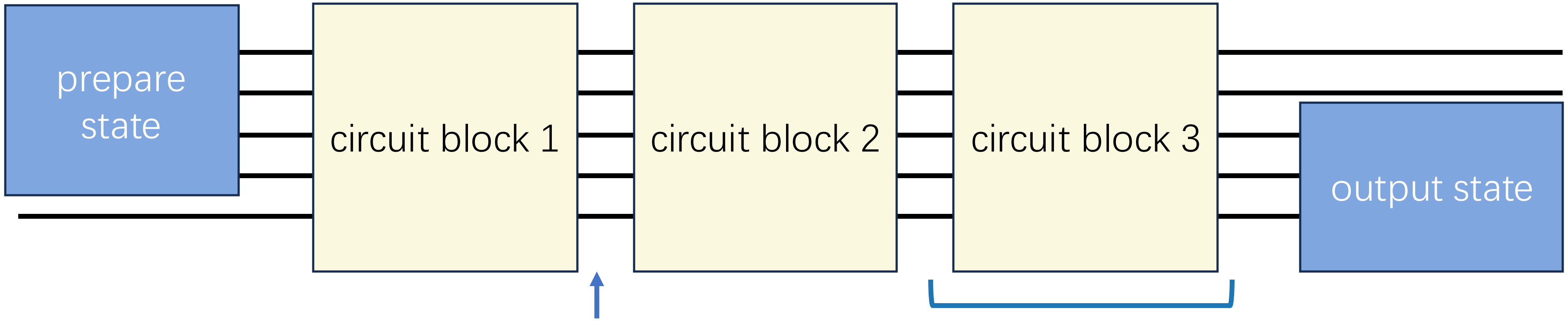
- **Background and Challenges**
- Overview of MorphQPV
- Assertion Statement and Validation
- Experiment
- API of MorphQPV

Ensure Program Is Correct?

Quantum program verification is **a fundamental theoretical method** for reliable quantum computing, which aims to ensure that quantum programs have correct behavior and achieve desired results during execution.

Process to check the program is correct

1. Check the relationship between the states in each stage
2. There may be mid-measurement or feedback



√ *1. check the runtime states is correct*

√ *2. check the relation between states is correct*

Described as **an assertion**, which is defined as a predicate. The predicate is expected to be true if there are no bugs.

Ensure Program Is Correct?

Quantum program verification is **a fundamental theoretical method** for reliable quantum computing, which aims to ensure that quantum programs have correct behavior and achieve desired results during execution.

Process to check the program is correct

1. Check the relationship between the states in each stage
2. There may be mid-measurement or feedback

$$assert\ a \geq 0$$

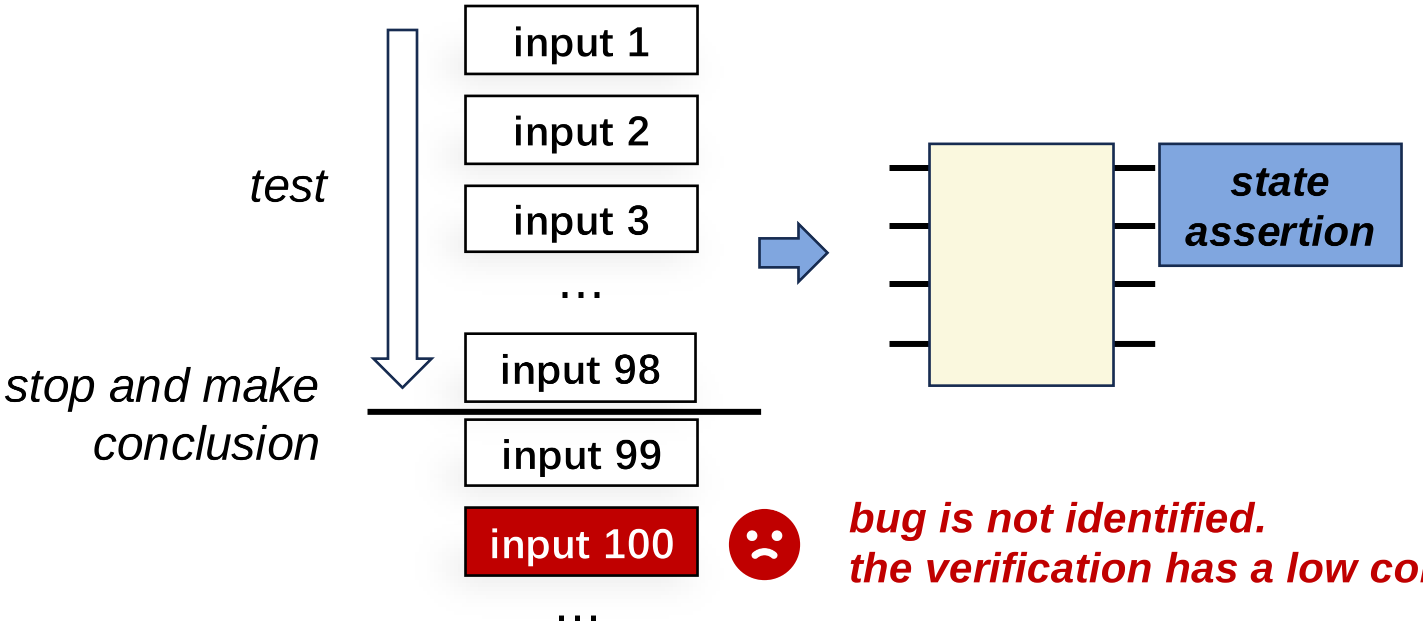
$$b = \sqrt{a}$$

Quantum assertion is similar to the classical assertion, while the verified qubits usually stay in the superposition state.

Confidence of the Verification

ZHEJIANG UNIVERSITY

Confidence of the verification: The probability that the correctness holds for all inputs.



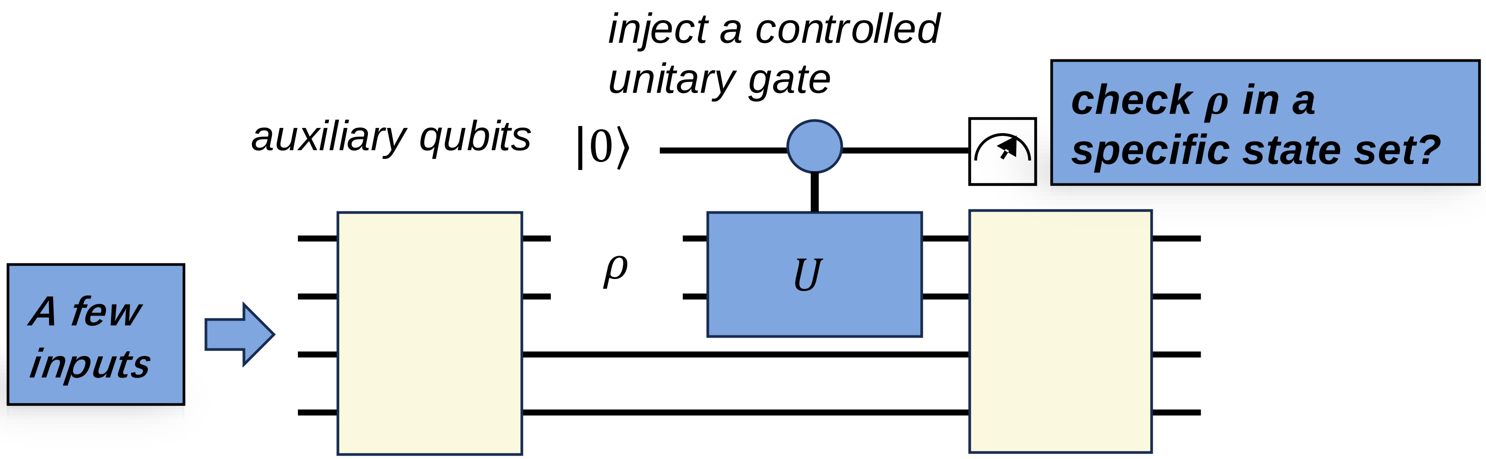
bug is not identified.
the verification has a low confidence

The ability to generalize the test inputs to the whole space is necessary to ensure high confidence

Limitation of Current Quantum Assertion Methods

	Statement	Analysis method	Input coverage	Theorem
A good assertion validation method	Specify the relation between states	Efficiently characterize the relation	The whole input space	1. Complexity is up-bounded 2. Theorem guarantee to high confidence

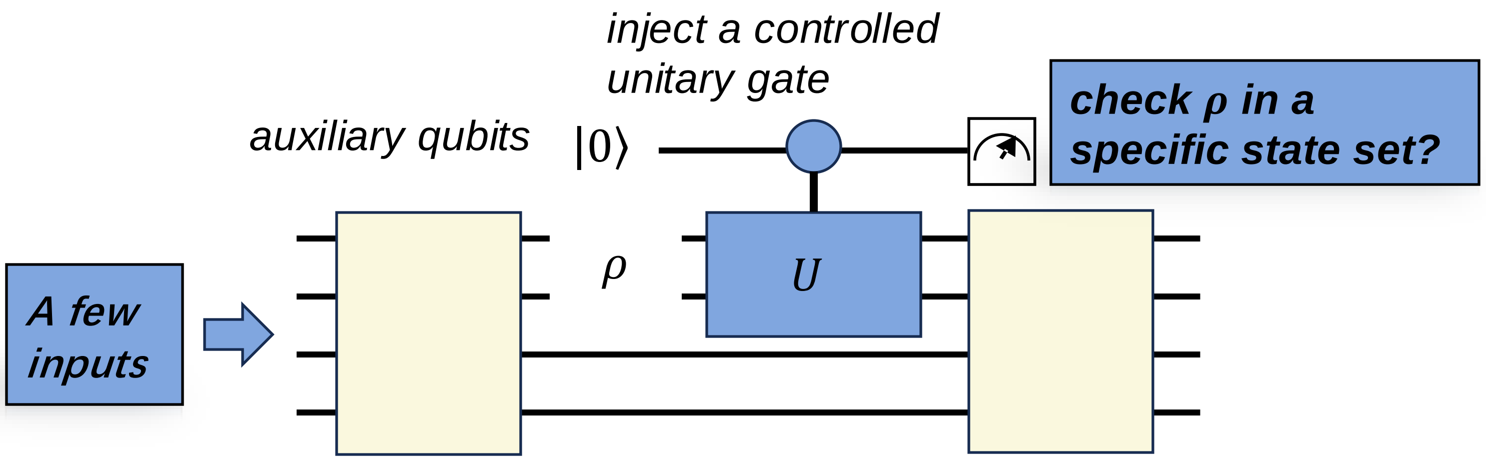
An example **Ji Liu, et al. Quantum circuits for dynamic runtime assertions in quantum computation. ASPLOS, 2020**



Limitation of Current Quantum Assertion Methods

	Statement	Analysis method	Input coverage	Theorem
A good assertion validation method	Specify the relation between states	Efficiently characterize the relation	The whole input space	1. Complexity is up-bounded 2. Theorem guarantee to high confidence

An example **Ji Liu, et al. Quantum circuits for dynamic runtime assertions in quantum computation. ASPLOS, 2020**



Expressiveness?

Only support state in state equal operation.

Efficient?

Numerous gates to synthesize unitary gates.

High confidence?

Cover a few inputs.
Lack confidence guarantee.

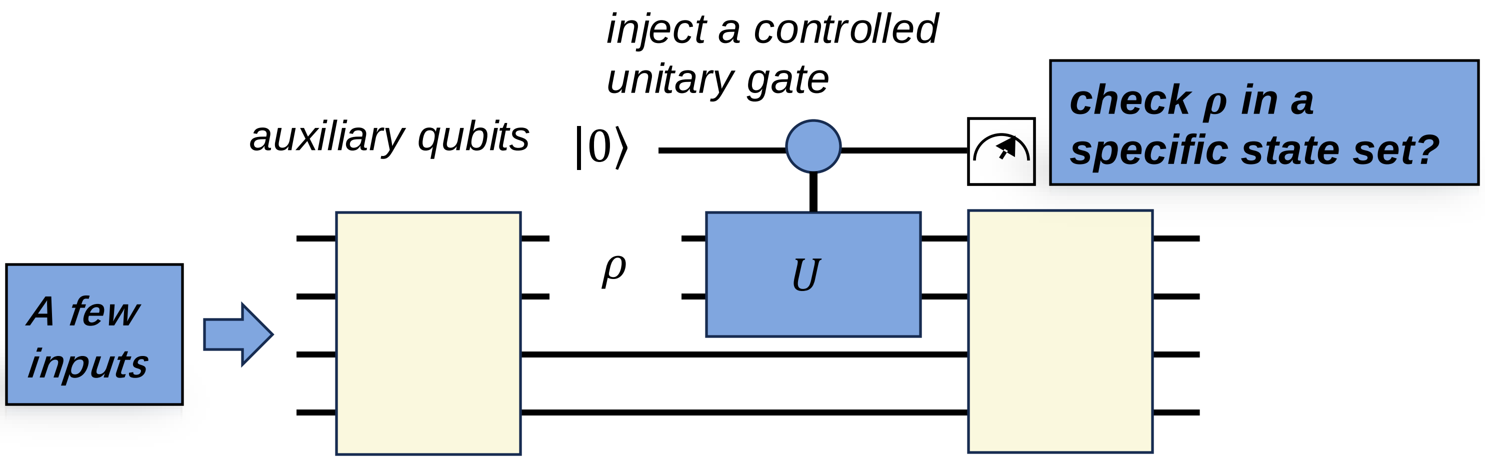
Limitation of Current Quantum Assertion Methods

	Statement	Analysis method	Input coverage	Theorem
A good assertion validation method	Specify the relation between states	Efficiently characterize the relation	The whole input space	1. Complexity is up-bounded 2. Theorem guarantee to high confidence

Similar problems in current works, including
Liu, et al. OOPSLA 2020,
Huang, et al. ISCA, 2019

An example

Ji Liu, et al. Quantum circuits for dynamic runtime assertio



Expressiveness?

Only support state in state equal operation.

Efficient?

Numerous gates to synthesize unitary gates.

High confidence?

Cover a few inputs.
Lack confidence guarantee.

Reason of Limitations

Quantum collapse leads to loss of information.

N-qubit state

=

$$\begin{bmatrix} 0.2 & 0.3i & \cdots & 0.0 & 0.0 \\ 0.1 & 0.3 & & 0.3 & 0.0 \\ \vdots & & \ddots & \vdots & \\ 0.4 & 0.0 & & 0.1 & 0.5 \\ 0.1 & 0.2 & \cdots & 0.0 & 0.0 \end{bmatrix}$$

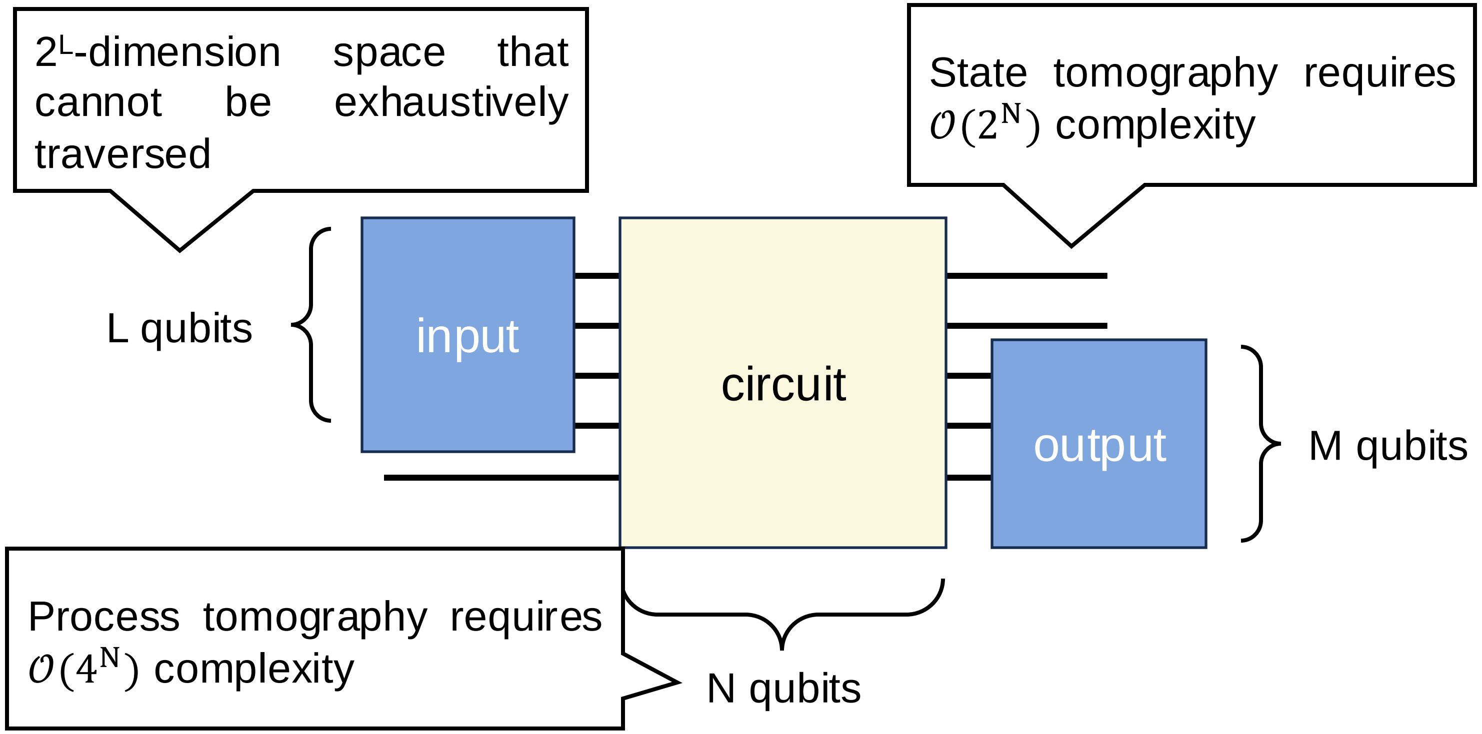
$2^N \times 2^N$ -size density matrix

Only diagonal elements are obtained by measurements

Lack an efficient method to characterize the mapping from input to output

Necessary to ensure high confidence

Necessary to achieve high expressiveness



Alternative approach but has high complexity (N > L, N > M)

Outline of Presentation

- Background and Challenges
- **Overview of MorphQPV**
- Assertion Statement and Validation
- Experiment
- API of MorphQPV

MorphQPV Overview

write a quantum program

```
input q[0,1];
x q[2,3,4];
cz q[1],q[4];
t1 q[1,2];
x q[2,3,4];
h q[1];
t2 q[0];
```

label states by a tracepoint pragma

Object: enable input-independent assertion

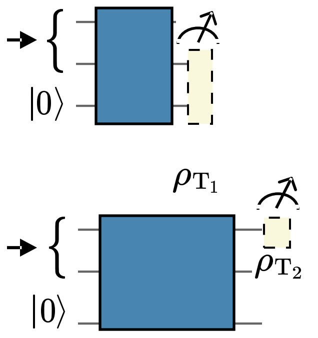
Object: minimize the number of program execution

Object: apply global search to counter example

Step 1. assertion statement

assume
$P_1(\rho_{T_1})$
$P_2(\rho_{T_2})$
guarantee
$P_3(\rho_{T_1}, \rho_{T_2})$

Step 2. program characterization



$\rho_{T_1} = f_1(\rho_{in})$

$\rho_{T_2} = f_2(\rho_{in})$

Step 3. assertion validation

$$\begin{aligned} \underset{\rho_{in}}{\text{maximize}}\ & P_3(f_1(\rho_{in}), f_2(\rho_{in})), \\ \text{subject to}\ & P_1(f_1(\rho_{in})) \le 0, \\ & P_2(f_2(\rho_{in})) \le 0, \end{aligned}$$

$max \le 0$

yes: correct & confidence

no: counter-example

compare

specify the ***expected relation*** *between the tracepoint states*

characterize the ***natural relation*** *by building approximation functions*

Object: guide the allocation of computation resource

MorphQPV Overview

write a quantum program

```
input q[0,1];
x q[2,3,4];
cz q[1],q[4];
t1 q[1,2];
x q[2,3,4];
h q[1];
t2 q[0];
```

label states by a tracepoint pragma

MorphQPV Overview

浙江大學 ZHEJIANG UNIVERSITY

write a quantum program

```
input q[0,1];
x q[2,3,4];
cz q[1],q[4];
t1 q[1,2];
x q[2,3,4];
h q[1];
t2 q[0];
```

label states by a tracepoint pragma

Object: enable input-independent assertion

Step 1. assertion statement

assume
$P_1(\rho_{T_1})$
$P_2(\rho_{T_2})$
guarantee
$P_3(\rho_{T_1}, \rho_{T_2})$

specify the ***expected relation*** *between the tracepoint states*

MorphQPV Overview

write a quantum program

```
input q[0,1];
x q[2,3,4];
cz q[1],q[4];
t1 q[1,2];
x q[2,3,4];
h q[1];
t2 q[0];
```

label states by a tracepoint pragma

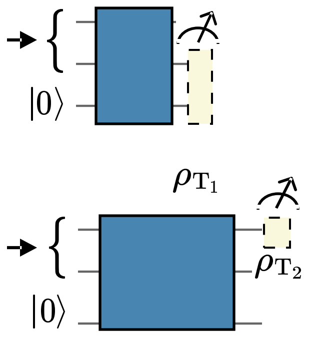
Object: enable input-independent assertion

Object: minimize the number of program execution

Step 1. assertion statement

assume
$P_1(\rho_{T_1})$
$P_2(\rho_{T_2})$
guarantee
$P_3(\rho_{T_1}, \rho_{T_2})$

Step 2. program characterization



$\rho_{T_1} = f_1(\rho_{in})$

$\rho_{T_2} = f_2(\rho_{in})$

*specify the **expected relation** between the tracepoint states*

*characterize the **natural relation** by building approximation functions*

MorphQPV Overview

write a quantum program

```
input q[0,1];
x q[2,3,4];
cz q[1],q[4];
t1 q[1,2];
x q[2,3,4];
h q[1];
t2 q[0];
```

label states by a tracepoint pragma

Object: enable input-independent assertion

Object: minimize the number of program execution

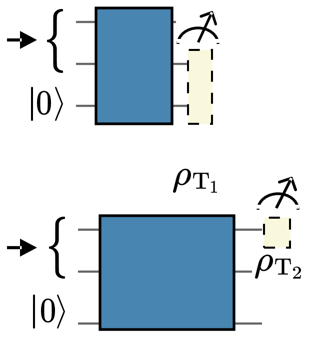
Object: apply global search to counter example

Step 1. assertion statement

assume
$P_1(\rho_{T_1})$
$P_2(\rho_{T_2})$
guarantee
$P_3(\rho_{T_1}, \rho_{T_2})$

specify the ***expected relation*** *between the tracepoint states*

Step 2. program characterization



$\rho_{T_1} = f_1(\rho_{in})$

$\rho_{T_2} = f_2(\rho_{in})$

characterize the ***natural relation*** *by building approximation functions*

compare

Step 3. assertion validation

$$\max_{\rho_{in}} P_3(f_1(\rho_{in}), f_2(\rho_{in})),$$
$$\text{subject to } P_1(f_1(\rho_{in})) \le 0,$$
$$P_2(f_2(\rho_{in})) \le 0,$$

$max \le 0$

yes → correct & confidence

no → counter-example

Object: guide the allocation of computation resource

Outline of Presentation

- Background and Challenges
- Overview of MorphQPV
- **Assertion Statement and Validation**
- Experiment
- API of MorphQPV

Assertion Statement

ZHEJIANG UNIVERSITY

1. Label the asserted state by tracepoint pragma

```
input q[0,1];
x q[2,3,4];
cz q[1],q[4];
t1 q[1,2];
x q[2,3,4];
h q[1];
t2 q[0];
```

input q[qubits]

t index q[qubits]

2. Use assume-guarantee assertion to specify the relation between states

assume:

$$P_1(\rho_{T1}) \text{ , } P_2(\rho_{T2})$$

guarantee:

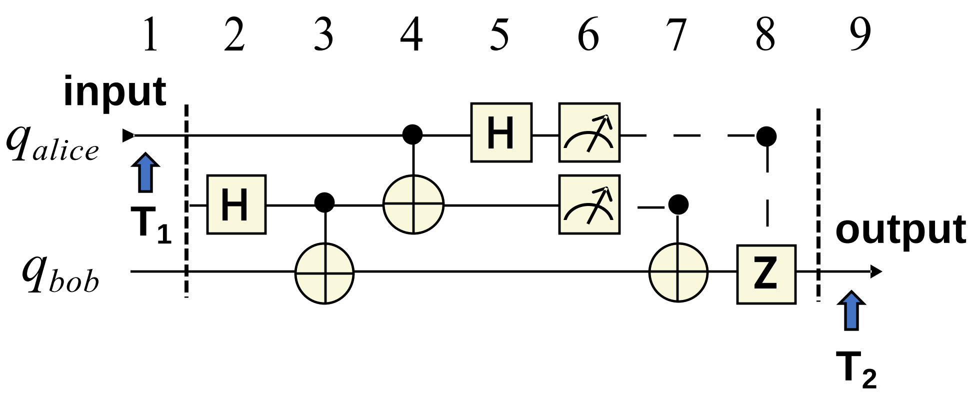
$$P_3(\rho_{T1}, \rho_{T2})$$

If $P_1 \leq 0$ and $P_2 \leq 0$ then $P_3 \leq 0$
⇒ assertion is correct

or

If $P_1 \leq 0$ and $P_2 \leq 0$ then $P_3 > 0$
⇒ assertion fails

An example: Quantum teleportation



input state should equal output state

assume:

$$P_1(\rho_{T1}) = \|\rho_{T1}\rho_{T1}{}^{\dagger} - \rho_{T1}\| \text{ ,}$$

$$P_2(\rho_{T2}) = \|\rho_{T2}\rho_{T2}{}^{\dagger} - \rho_{T2}\|,$$

guarantee:

$$P_3(\rho_{T1}, \rho_{T2}) = \|\rho_{T1} - \rho_{T2}\|$$

Isomorphism-based Characterization

Isomorphism

A structure-preserving mapping $\mathbb{R}_x \to \mathbb{R}_y$ between two spaces of the same type that can be retraced by an inverse mapping.

Example of isomorphism

$$x + 1 = y$$

inverse

$$y - 1 = x$$

Quantum evolution is isomorphism

$$U\rho U^{-1} = \rho'$$

inverse

$$U^{-1}\rho' U = \rho$$

Feature of isomorphism

additivity: $\mathrm{f}(u + v) = f(u) + f(v)$

homogeneity: $\mathrm{f}(c\ u) = c\ f(u)$

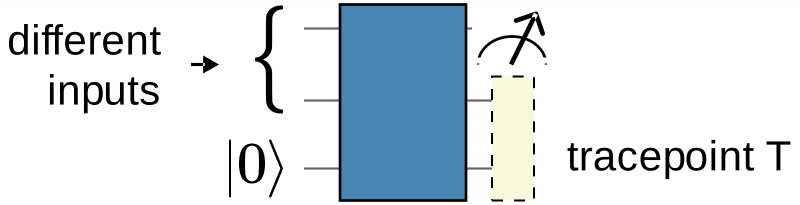
$\mathrm{f}(\sum_i c_i u_i) = \sum_i c_i\ f(\mathrm{u}_i)$

also has the feature

inspire us to generalize the information obtained from individual input into a broader input space

Isomorphism-based Characterization

Step 1: sample inputs



Inputs are orthogonal and prepared by the Clifford group.

Based *on "Sergey Bravyi and Dmitri Maslov. Hadamard-free circuits expose the structure of the clifford group. IEEE Transactions on Information Theory, 2021".*

Record inputs and state at tracepoint as $\langle \sigma_{\mathrm{input,i}}, \sigma_{\mathrm{T,i}} \rangle$ pairs.

Input state	Tracepoint state
$\sigma_{\mathrm{input},1}$	$\sigma_{\mathrm{T},1}$
$\sigma_{\mathrm{input},2}$	$\sigma_{\mathrm{T},2}$
…	
$\sigma_{\mathrm{input},N_{\mathrm{sample}}}$	$\sigma_{\mathrm{T},N_{\mathrm{sample}}}$

Obtained by tomography

Step 2: construct approximation function $f(\rho_{\mathrm{input}}) = \rho_{\mathrm{T}}$

The function is computed in two steps:

1. For input ρ_{input}, it first approximates the ρ_{input} to the linear combination of sampled inputs $\sigma_{\mathrm{inpu,i}}$

$$\rho_{\mathrm{input}} = \sum_i \alpha_i \, \sigma_{\mathrm{input,i}}$$

$\{\alpha_i\}$ is real parameters.

2. It then outputs tracepoint state:

$$\rho_{\mathrm{T}} = \sum_i \alpha_i \, \sigma_{\mathrm{T,i}}$$

Based on the additivity and homogeneity of isomorphism

$$\mathrm{f}(\textstyle\sum_i c_i u_i) = \sum_i c_i \, f(\mathrm{u}_i)$$

Accuracy of Characterization

ZHEJIANG UNIVERSITY

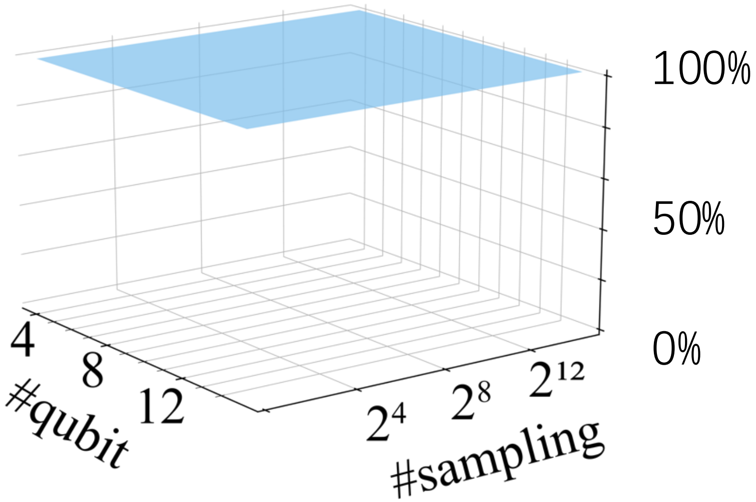
Theorem 1 (Approximation Accuracy)

- Case1: For inputs that can be accurately represented by linear combination of sampled inputs, the accuracy is 100%.
- Case2: For inputs with eigenstates that cannot be represented the linear combination of sampled inputs, the average accuracy is $\frac{N_{\text{sample}}}{2^{N_{input}}} \times 100\%$

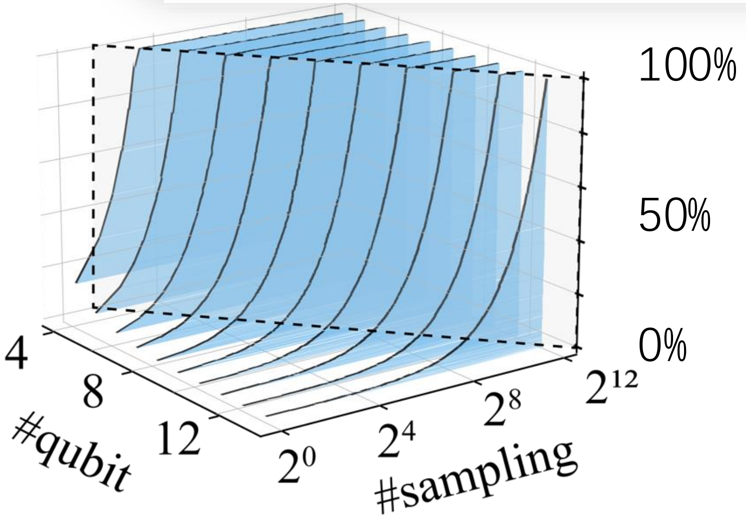
Example: Approximation accuracy in the quantum teleportation programs with different number of qubits and sampled inputs.

The approximation result is the same as the tomograph result, when the accuracy is 100%, .

Case 1



Case 2



Assertion Validation

Assertion

assume:

$P_1(\rho_{T1})$, $P_2(\rho_{T2})$

assume-guarantee:

$P_3(\rho_{T1}, \rho_{T2})$

Maximization problem

$$\max_{\rho_{\text{input}}} P_3(\rho_{T1}, \rho_{T2}),$$

subject to $P_1(\rho_{T1}) \leq 0,$

$P_2(\rho_{T2}) \leq 0.$

Maximization problem

$$\max_{\{\alpha_i\}} P_3\big(f_1(\{\alpha_i\}), f_2(\{\alpha_i\})\big),$$

subject to $P_1\big(f_1(\{\alpha_i\})\big) \leq 0,$

$P_2\big(f_2(\{\alpha_i\})\big) \leq 0.$

Solver by solver, e.g. quadratic programming

Validation

if $\max P_3 \leq 0$:

assertion is true

else:

assertion is false

Characterization

$f_1(\rho_{\text{input}}) = \rho_{\text{T1}}$

$f_2(\rho_{\text{input}}) = \rho_{\text{T2}}$

$f_1(\{\alpha_i\}) = \rho_{\text{T1}}$

$f_2(\{\alpha_i\}) = \rho_{\text{T2}}$

Approximation parameters

Assertion Validation

Assertion

assume:

$$P_1(\rho_{T1})\ ,\ P_2(\rho_{T2})$$

assume-guarantee:

$$P_3(\rho_{T1}, \rho_{T2})$$

Maximization problem

$$\max_{\rho_{\text{input}}}\ P_3(\rho_{T1}, \rho_{T2}),$$

subject to $P_1(\rho_{T1}) \leq 0,$

$$P_2(\rho_{T2}) \leq 0.$$

Assertion Validation

Assertion

assume:

$$P_1(\rho_{T1}) \;,\; P_2(\rho_{T2})$$

assume-guarantee:

$$P_3(\rho_{T1}, \rho_{T2})$$

Maximization problem

$$\max_{\rho_{\text{input}}} \; P_3(\rho_{T1}, \rho_{T2}),$$

subject to $P_1(\rho_{T1}) \leq 0,$

$$P_2(\rho_{T2}) \leq 0.$$

Characterization

$$f_1(\rho_{\text{input}}) = \rho_{\text{T1}}$$
$$f_2(\rho_{\text{input}}) = \rho_{\text{T2}}$$

$$f_1(\{\alpha_i\}) = \rho_{\text{T1}}$$
$$f_2(\{\alpha_i\}) = \rho_{\text{T2}}$$

Approximation parameters

Assertion Validation

Assertion

assume:

$P_1(\rho_{T1})$, $P_2(\rho_{T2})$

assume-guarantee:

$P_3(\rho_{T1}, \rho_{T2})$

Maximization problem

$$\max_{\rho_{\text{input}}} P_3(\rho_{T1}, \rho_{T2}),$$

subject to $P_1(\rho_{T1}) \leq 0,$

$P_2(\rho_{T2}) \leq 0.$

Maximization problem

$$\max_{\{\alpha_i\}} P_3\big(f_1(\{\alpha_i\}), f_2(\{\alpha_i\})\big),$$

subject to $P_1\big(f_1(\{\alpha_i\})\big) \leq 0,$

$P_2\big(f_2(\{\alpha_i\})\big) \leq 0.$

Solver by solver, e.g. quadratic programming

Validation

if $\max P_3 \leq 0$:

assertion is true

else:

assertion is false

Characterization

$f_1(\rho_{\text{input}}) = \rho_{\text{T1}}$

$f_2(\rho_{\text{input}}) = \rho_{\text{T2}}$

$f_1(\{\alpha_i\}) = \rho_{\text{T1}}$

$f_2(\{\alpha_i\}) = \rho_{\text{T2}}$

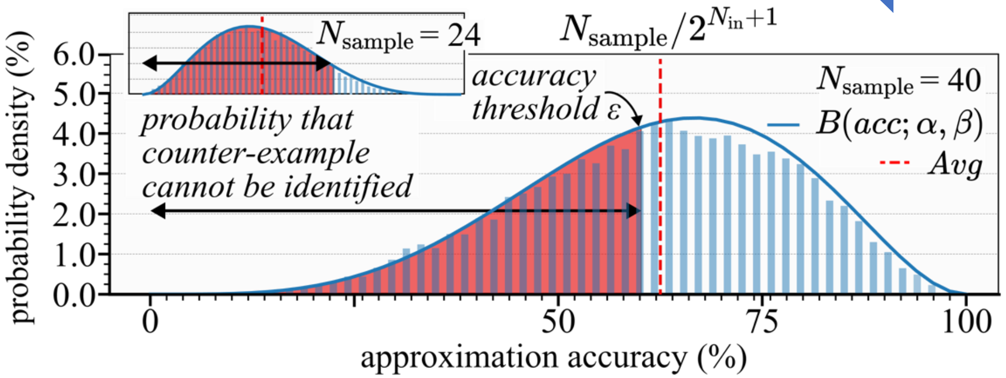
Approximation parameters

Confidence of Validation

confidence = P(the correctness holds for all inputs) = 1 – P(counter-example exists but is not identified)

P(counter-example exists but is not identified) = $\mathrm{P}(\text{accuracy of counter} - \text{example} < \boxed{\epsilon})$

accuracy threshold to discriminate error



$$\mathrm{P}(\text{accuracy } < \epsilon) = \int_0^{\epsilon} \mathrm{B}(x; \beta_1, \beta_2)$$

Accuracies follow Beta distribution $B(\beta_1, \beta_2)$
β_1, β_2 can obtained by fitting some test inputs

Theorem 2 (Confidence)

When the program only has one counter-example

lower-bound

$$\text{confidence} = 1 - \mathrm{P}(\text{accuracy } < \epsilon)$$

When the program only has N_{c-e} counter-examples

$$\text{confidence} = 1 - \mathrm{P}(\text{accuracy } < \epsilon)^{N_{c-e}}$$

Accuracy and confidence linearly increase as the number of sampled inputs grows

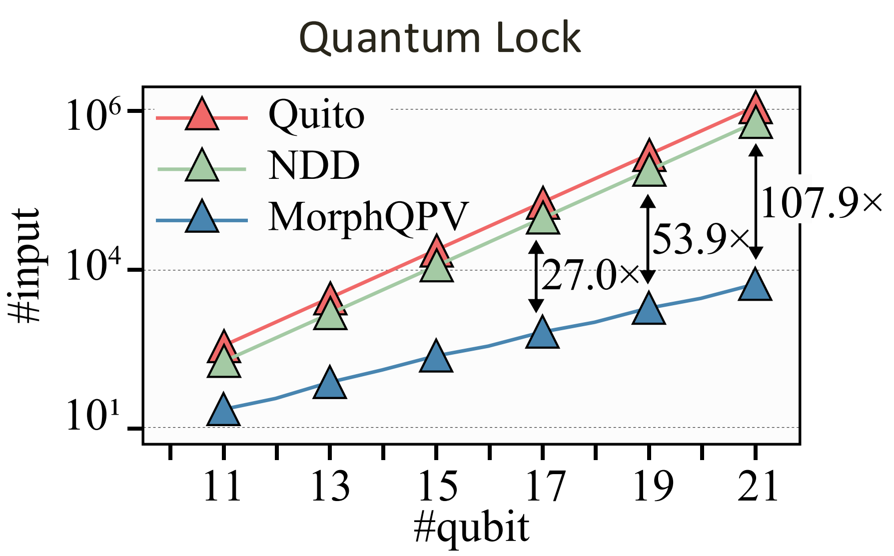
Outline of Presentation

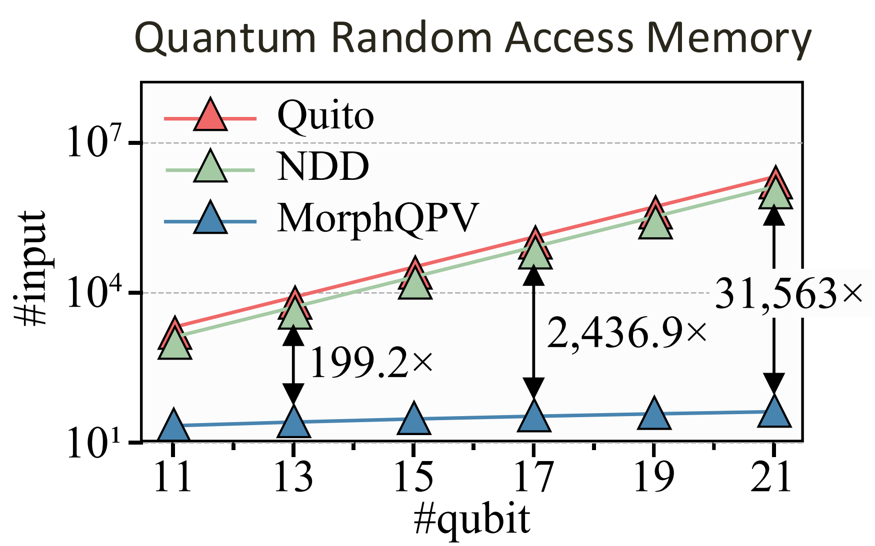
ZHEJIANG UNIVERSITY

- Background and Challenges
- Overview of MorphQPV
- Assertion Statement and Validation
- **Experiment**
- API of MorphQPV

Comparison to Prior Works

	Huang, et al. ISCA'19	Li,et al. OOPSLA'20	Liu, et al. ASPLOS'20	Feng, et al. ASPLOS'23	**MorphQPV**
Verified Object	Probability distribution	Mixed state	Mixed state	Mixed state	Mixed state & Evolution
Comparison	Part	Equal & In	Equal & In	Equal & In	Full
Interpretability	Part	No	No	No	Full
Feedback	No	No	No	No	Full



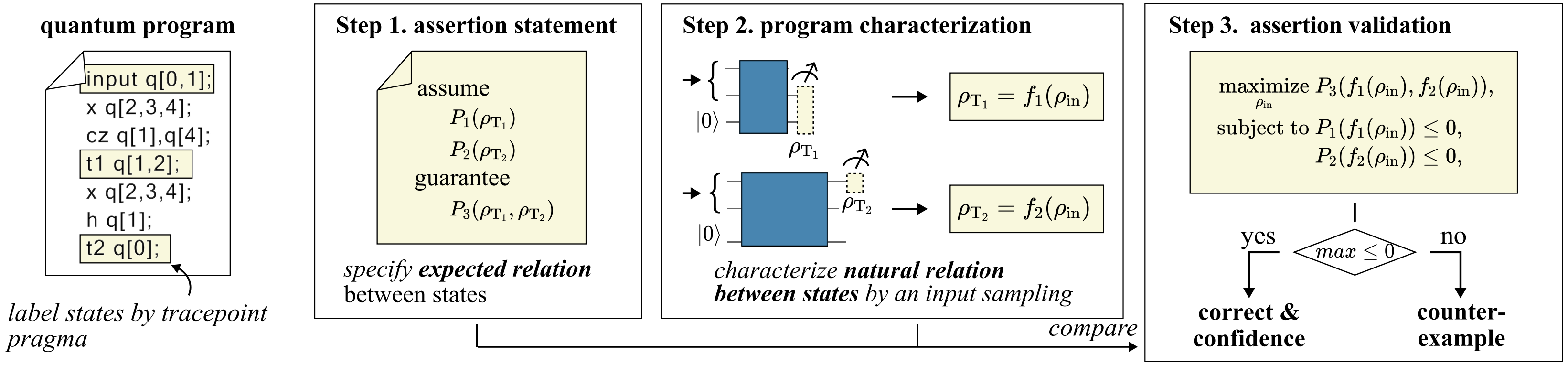


The number of test inputs in debugging the programs with different number of qubits

Conclusion

1. Limitations of prior assertion works: low confidence, low expressiveness, and high overhead
2. Three-step verification of MorphQPV: statement, characterization, and validation
3. Two theorems: upper-bound complexity of verification and lower-bound of confidence
4. Contents that are not mentioned in the presentation:
   - Proof of theorems。
   - Further optimization to minimize the overhead。
   - detailed comparison with prior works.

Please refer to the paper.



Outline of Presentation

- Background and Challenges
- Overview of MorphQPV
- Assertion Statement and Validation
- Experiment
- **API of MorphQPV**

API of MorphQPV

File:
- JanusQ/examples/ipynb/3_1_verify_quantum_program.ipynb
- https://janusq.github.io/tutorials/demo/ 3_1_verify_quantum_program

configure solver

assertion statement and validation in quantum circuit

```
from janusq.verification.morphqpv import MorphQC,Config
from janusq.verification.morphqpv import IsPure,Equal,NotEqual

myconfig = Config()
myconfig.solver = 'sgd'

with MorphQC(config=myconfig) as morphQC:
    morphQC.x([1,3])
    morphQC.y([0,1,2])
    for i in range(4):
        morphQC.cnot([i, i+1])
    morphQC.s([0,2,4])
    morphQC.add_tracepoint(2,4)
    morphQC.assume(1,IsPure())
    ...
    morphQC.guarantee([1,2],Equal())
    morphQC.guarantee([0,1],NotEqual())
```

Thanks for listening

MorphQPV: Exploiting Isomorphism in Quantum Programs to Facilitate Confident Verification

Siwei Tan*, Debin Xiang*, Liqiang Lu†, Junlin Lu, Qiuping Jiang,
Mingshuai Chen, and Jianwei Yin†