Projection-based Runtime Assertions for Testing and Debugging Quantum Programs

Gushu Li[1], Li Zhou[2], Nengkun Yu[3], Yufei Ding[1], Mingsheng Ying[3], and Yuan Xie[1]

[1] University of California, Santa Barbara, USA

[2] Max Planck Institute for Security and Privacy, Germany

[3] University of Technology, Sydney, Australia

Bio

- Name: Gushu Li
- 4-th Year Ph.D. student at ECE Department, UCSB
- Advisors: Yuan Xie, Yufei Ding
- Research interests: quantum computing, applications, programming language, compiler optimization, hard-ware architecture design, etc.

A short story behind this project

- FCRC 2019 in Phoenix, AZ, Jun 2019 (every 4 years)

A short story behind this project

- Yipeng presented his assertion paper at ISCA 2019 (part of FCRC)

  Huang, Y., & Martonosi, M. (2019, June). **Statistical assertions for validating patterns and finding bugs in quantum programs**. In *Proceedings of the 46th International Symposium on Computer Architecture* (pp. 541-553).

  The first quantum program assertion for testing to my best.

A short story behind this project

- Yipeng presented his assertion paper at ISCA 2019 (part of FCRC)
- I met Yipeng during the conference.

A short story behind this project

- Yipeng presented his assertion paper at ISCA 2019 (part of FCRC)
- I met Yipeng during the conference.
- Li Zhou (the co-primary author) presented at PLDI 2019. He works on programming language theory for quantum computing.

A short story behind this project

- Yipeng presented his assertion paper at ISCA 2019 (part of FCRC).
- I met Yipeng during the conference.
- Li Zhou (the co-primary author) presented at PLDI 2019. He works on programming language theory for quantum computing.

- Li and I spent one afternoon together and formulated most of the ideas in this paper.

A short story behind this project

- Later that day, I summarized the major ideas and email our advisors.
- Then this project is formally initiated.

Gushu Li <gushuli@ece.u... Mon, Jun 24, 2019, 11:12 PM
to Li, Yufei, Mingsheng

Dear Li,

It was nice talking to you today. The notes are attached based on today's discussion.

We will propose a runtime assertion scheme for debugging on a quantum computer.
The following three major contributions are expected:

A short story behind this project

- Later that day, I summarized the major ideas and email our advisors.
- Then this project is formally initiated.

- First submitted to PLDI 2020
  - It got rejected because I made some mistakes. I worked mostly on compiler optimization and hardware architecture before this project. It was my first time to submit to a PL conference.
- Then OOPSLA 2020
  - I learnt some basics and read more about PL. We spent another month incorporating reviews' feedback and rewriting most of the paper.

Outline

- Motivation
- Preliminary
- Assertion design (theoretical foundations)
- Assertion implementation (practical implementation issues)
- Some examples
- Summary

To improve the program reliability

- In this study, we target quantum program runtime testing and debugging on quantum computers.

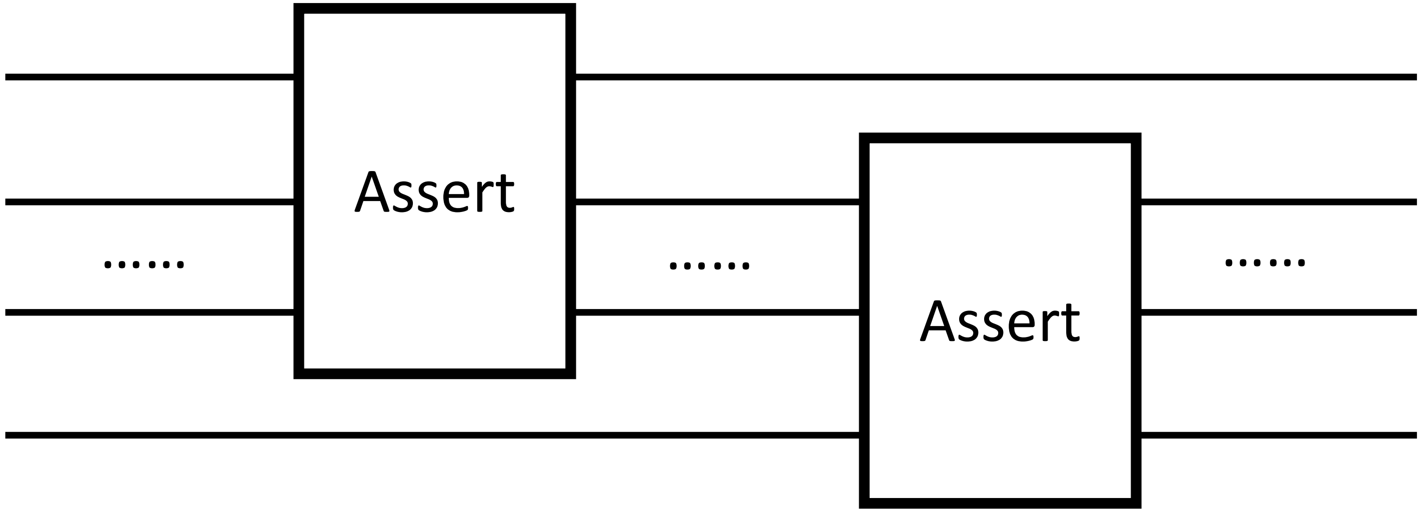
To improve the program reliability

- In this study, we target quantum program runtime testing and debugging on quantum computers.

- Compared with verification
  - Verification may be able to guarantee the correctness, but it is usually very expensive.
  - Testing is cheaper. It can reveal some errors but not guarantee the correctness.
  - Still quite useful.

What is an assertion?

- A predicate on some program variables at a place in the program
- If the predicate is satisfied, pass.
- If not, abort.

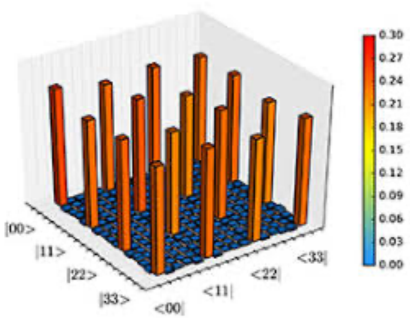


Test a quantum program state

Test a quantum program state

- Learn from quantum information processing?

Quantum State Tomography:

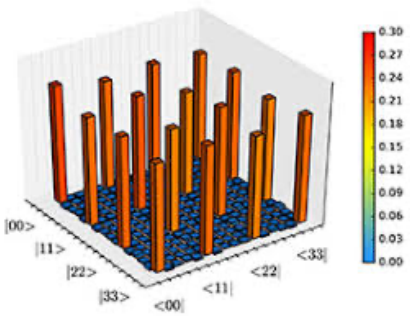


Yes, QST can fully characterize any state ρ, but it is too expensive. Repeat state preparation and measurement many times (exponential).

Test a quantum program state

- Some computer science approaches?

Quantum State Tomography:



Yes, QST can fully characterize any state ρ, but it is too expensive. Repeat state preparation and measurement many times (exponential).

Assertions for state property checking

Expressive power: describe a quantum state property using classical languages

$$|1010 \ldots 0101\rangle$$
$$|++ + \cdots + +\rangle$$
$$|000 \ldots 00\rangle + |111 \ldots 11\rangle$$

Another implicit inefficiency: measurement in the assertion checking may destroy the tested state.

Test a quantum program state

- We hope to have a new approach
  - Strong expressive power:
    The language should naturally describe complex quantum state properties
  - Efficient checking:
    The predicate satisfactory can be checked within few executions

- And after careful consideration, we select **Projections**

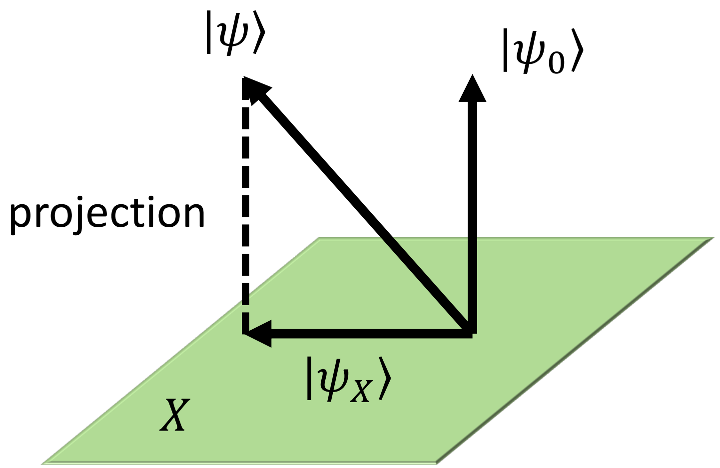
Preliminary

- Quantum computing basics
  - Qubit: q
  - Hilbert space: H, zero operator: 0_H, identity operator: I_H
  - State vector: $|\psi\rangle$, density operator: ρ (more general)
  - Unitary transformation (quantum gate): $U: |\psi\rangle \mapsto U|\psi\rangle, \rho \mapsto U\rho U^\dagger$
  - Measurement $M = \{M_m\}$
    - We will only discuss about projective measurement which is supported on most hardware platforms.

Projection: definition and property

- Definition: for each closed subspace X of a Hilbert space H, we can define a projection P_X
  - For every $|\psi\rangle \in H$, we have $|\psi\rangle = |\psi_X\rangle + |\psi_0\rangle$, with $|\psi_X\rangle \in X$ and $|\psi_0\rangle \in X^\perp$.
    ($X^\perp$ is the orthocomplement of X)

  Then, $P_X: H \mapsto X$ is defined by $P_X|\psi\rangle = |\psi_X\rangle$ for every $|\psi\rangle \in H$.



Projection: definition and property

- Definition: for each closed subspace X of a Hilbert space H, we can define a projection P_X
  - For every $|\psi\rangle \in H$, we have $|\psi\rangle = |\psi_X\rangle + |\psi_0\rangle$, with $|\psi_X\rangle \in X$ and $|\psi_0\rangle \in X^\perp$.
    ($X^\perp$ is the orthocomplement of X)
    Then, $P_X: H \mapsto X$ is defined by $P_X|\psi\rangle = |\psi_X\rangle$ for every $|\psi\rangle \in H$.
- Property:
  - One-to-one correspondence between the closed subspaces of H and the projections in it. (We denote P_X as P and do not distinguish a projection and its corresponding subspace)
  - When $|\psi\rangle$ (or ρ) is in P, we have $P|\psi\rangle = |\psi\rangle$ (or $P\rho P = \rho$).

Projective measurement

- Definition:
  - A quantum measurement in which all the measurement operators are projections
  - $M = \{P_m\}, \sum P_m = I_H, P_m P_n = \begin{cases} P_m, & m = n \\ 0_H, & m \neq n \end{cases}$
- After a projective measurement
  - With probability $p(m) = \langle\psi|P_m^\dagger P_m|\psi\rangle$ (resp. $tr(P_m\rho)$), the state is changed to $\frac{P_m|\psi\rangle}{\sqrt{\langle\psi|P_m^\dagger P_m|\psi\rangle}}$ $\left(\frac{P_m\rho P_m^\dagger}{tr(P_m\rho)}\right)$, $\sum p(m) = 1$
  - A projective measurement will apply projection operators on a state

Projective measurement

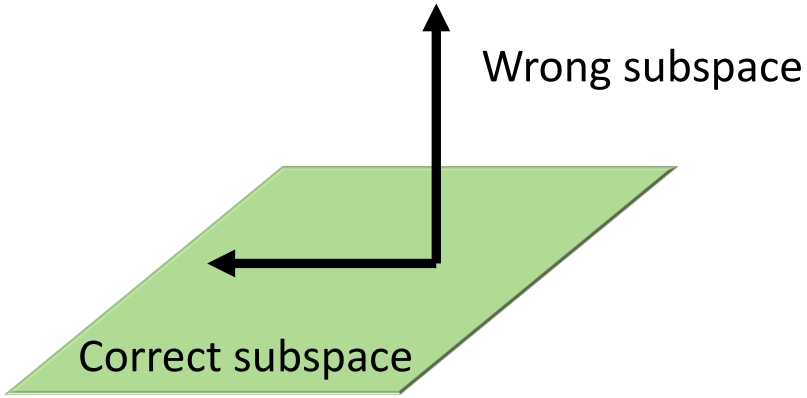
- When a projective measurement will not change the measured state?
  - Recall that when $|\psi\rangle$ (or ρ) is in P, we have $P|\psi\rangle = |\psi\rangle$ (or $P\rho P = \rho$).

- For a projective measurement $M = \{P_m\}$, when $|\psi\rangle$ (or ρ) is in P_m, the state after the measurement is still $|\psi\rangle$ (or ρ) with probability 1.

Why using projections

- 1. Projections can naturally represent predicates

Projection-based Predicates: Suppose P is a projection on a Hilbert space H, a state ρ is said to satisfy a projection P (written $\rho \vDash P$) if $\mathrm{supp}(\rho) \subseteq X$.
($\mathrm{supp}(\rho)$ is the subspace spanned by the eigenvectors of ρ with non-zero eigenvalues)
Note that: $\rho \vDash P \Longrightarrow P\rho = \rho$



Why using projections

- 1. Projections can naturally represent predicates
- 2. Can be checked upon projective measurements

For projection P, we can define: $M_P = \{M_{true} = P, M_{false} = I - P\}$
The measurement outcome will tell us whether the tested state ρ is in P .

There are some practical issues when physically implementing the constructed measurement. We will discuss them later.

Benefits of projections

- 1. Strong expressive power
  - Much more expressive compared with classical language description
  - Projections can have different ranks (different dimensions of the subspaces)
    - e.g., $P = |00\rangle\langle 00|$, the correct state must be the $|00\rangle$ state.
    - $P = |00\rangle\langle 00| + |11\rangle\langle 11|$, the correct state can be the linear combination of $|00\rangle$ and $|11\rangle$
    - Specify subspaces of any dimensions

Benefits of projections

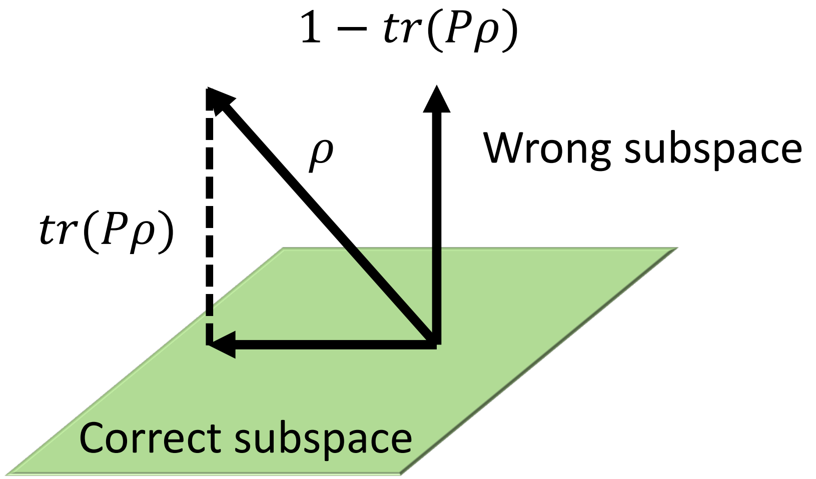
- 1. Strong expressive power
  - Much more expressive compared with classical language description
  - Projections can have different ranks (different dimensions of the subspaces)
    - e.g., $P = |00\rangle\langle 00|$, the correct state must be the $|00\rangle$ state.
    - $P = |00\rangle\langle 00| + |11\rangle\langle 11|$, the correct state can be the linear combination of $|00\rangle$ and $|11\rangle$
    - Specify subspaces of any dimensions
  - Indistinguishable states, $\rho = \alpha|00\rangle\langle 00| + \beta|11\rangle\langle 11|$
    - Because we are not doing tomography
    - And this will allow efficient checking

Benefits of projections

- 1. Strong expressive power
- 2. Efficient checking
  - Recall that $P\rho P = \rho$ when $\rho \vDash P$
  - Therefore in the measurement $M_P = \{M_{true} = P, M_{false} = I - P\}$
    - If $\rho \vDash P$, we will always have outcome 'true' and the state is not changed in the measurement
    - If $\rho \nvDash P$ and we have outcome 'false', we will know that the state is wrong.
    - If $\rho \nvDash P$ and we have outcome 'true', the state ρ' after the measurement will satisfy P because the projective measurement maps ρ back to the correct subspace.
    - (<u>This allows runtime checking. We can run multiple assertions in one execution. A passed assertion will not affect the following execution.</u>)

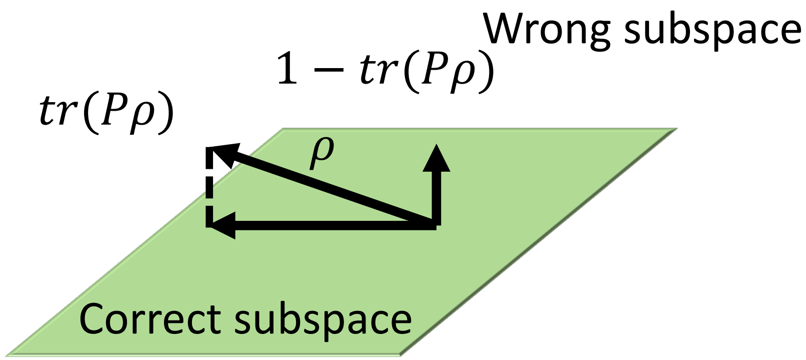
Benefits of projections

- 1. Strong expressive power
- 2. Efficient checking
  - How many executions do we need?
  - For k execution, the probability of not observing 'false' outcome is $tr(P\rho)^k$, it exponentially.



Benefits of projections

- 1. Strong expressive power
- 2. Efficient checking
  - How many executions do we need?
  - What if $1 - tr(P\rho)$ is small?
  - The ‘error’ is not severe in such cases
  - Because the semantics of quantum programs are **trace-nonincreasing** quantum operations.



Benefits of projections

- 1. Strong expressive power
- 2. Efficient checking
  - How many executions do we need?
  - What if $1 - tr(P\rho)$ is small?
  - The 'error' is not severe in such cases
  - Because the semantics of quantum programs are **trace-nonincreasing** quantum operations

Correct state ρ $\longrightarrow$ Program S $\longrightarrow$ Final state ρ_f

Small-error state ρ' $\longrightarrow$ Program S $\longrightarrow$ Final state ρ'_f

$D(\rho, \rho') < \epsilon$
D is the trace distance
$\longrightarrow$
We have $D(\rho_f, \rho'_f) < \epsilon$
The final state is also only slightly affected.

Benefits of projections

$D(\rho_f, \rho'_f) = 0$ iff $\rho_f = \rho'_f$

When the trace distance between two states are small, their measurement output distributions will always be 'similar' no matter how you measure them

- 1. Strong expressive power
- 2. Efficient checking
  - How many executions do we need?
  - What if $1 - tr(P\rho)$ is small?
  - The 'error' is not severe in such cases
  - Because the semantics of quantum programs are **trace-nonincreasing** quantum operations

Correct state ρ → Program S → Final state ρ_f

Small-error state ρ' → Program S → Final state ρ'_f

$D(\rho, \rho') < \epsilon$
D is the trace distance → We have $D(\rho_f, \rho'_f) < \epsilon$
The final state is also only slightly affected.

Projection-based assertion primitive

- $assert(\bar{q}; P)$, P is a projection, $\bar{q}$ is a set of qubits
- The semantics:
  - Construct a projective measurement $M_P = \{M_{true} = P, M_{false} = I - P\}$
  - If the measurement outcome is 'true', continue
  - If the measurement outcome is 'false', abort and report the termination location.

Multiple assertions per execution

- Divide the program into segments, and then inject assertions.
- $\mathrm{S}_1; assert(\bar{q}_1; P_1); \mathrm{S}_2; assert(\bar{q}_2; P_2) \ldots; \mathrm{S}_l; assert(\bar{q}_l; P_l)$

- We quantitatively evaluate the *statistical properties* of checking such a program with multiple assertions.
- Also approximate projection-based assertion.
- Details are available in the paper.

Practical implementation issues

- $M_P = \{M_{true} = P, M_{false} = I - P\}$
- This constructed measurement may not be directly executable on a quantum computer.
- There are two constraints.

Practical constraints

- 1. Limited measurement basis
  - Most physical quantum computers only support measurement along the computational basis
  - $M_P = \{M_{true} = |0\rangle\langle 0|, M_{false} = |1\rangle\langle 1|\}$,
  - $M_P = \{M_{true} = |+\rangle\langle +|, M_{false} = |-\rangle\langle -|\}$

Practical constraints

- 1. Limited measurement basis
  - Most physical quantum computers only support measurement along the computational basis
  - $M_P = \{M_{true} = |0\rangle\langle 0|, M_{false} = |1\rangle\langle 1|\}$, 😇
  - $M_P = \{M_{true} = |+\rangle\langle +|, M_{false} = |-\rangle\langle -|\}$ 😡
- 2. Dimension mismatch
  - 2^n-dimensional space for n qubits
  - Measure one qubit, the space is reduced by half, 2^{n-1}-dimensional space
  - We can only measure an integer number of qubits
  - Only support rank $P \in \{2^{n-1}, 2^{n-2}, \ldots, 2^0\}$
  - $P = |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 11|$ 😡

Transformation techniques

- We first solve the measurement basis problem
- Add a unitary transformation of adjust the measurement basis

Proposition: For projection P with $\mathrm{rank}\ P = 2^m$, there exists a unitary transformation U_P such that:

$$U_P P U_P^\dagger \;\; = \; Q_{q_1} \otimes \; Q_{q_2} \otimes \cdots \otimes \; Q_{q_n} = \otimes_{i=1}^n Q_{q_i} \triangleq Q_P$$

where $Q_{q_i} \in \{|0\rangle\langle 0|, |1\rangle\langle 1|, I\}$

- Apply U_P, check for Q_P , then apply $U_P^\dagger$

Transformation techniques

- We first solve the measurement basis problem
- An example:

$$P = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\frac{1}{\sqrt{2}}(\langle 00| + \langle 11|) = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

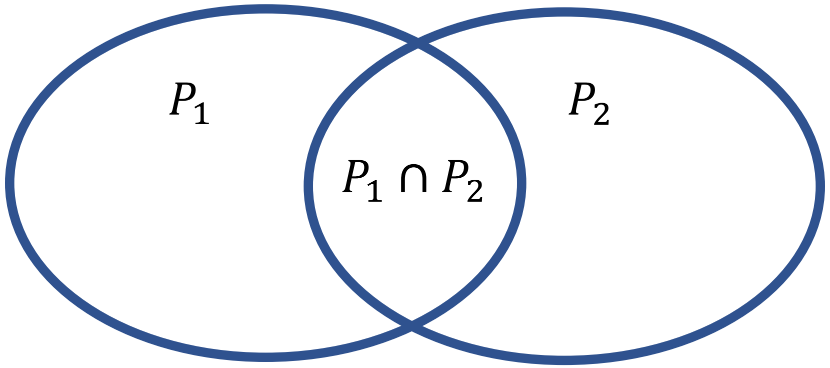
$$U_P P U_P^\dagger = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & \frac{-1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix} \cdot P \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = Q_P$$

One CNOT gate and one H gate

Transformation techniques

- We then solve the dimension mismatch problem
- Combine two projections to implement a projection with small dimension
  - Intersections of subspaces are still subspaces

Proposition: For projection P with $\mathrm{rank}\, P < 2^{n-1}$, there exists $P_1, P_2, \ldots, P_l$ satisfying $\mathrm{rank}\, P_i = 2^{n_i}$, n_is are integers and $P = P_1 \cap P_2 \cap \ldots \cap P_l$



Transformation techniques

- We then solve the dimension mismatch problem
- Combine two projections to implement a projection with small dimension
  - Intersections of subspaces are still subspaces

Proposition: For projection P with $\mathrm{rank}\, P < 2^{n-1}$, there exists $P_1, P_2, \ldots, P_l$ satisfying $\mathrm{rank}\, P_i = 2^{n_i}, n_i$s are integers and $P = P_1 \cap P_2 \cap \ldots \cap P_l$

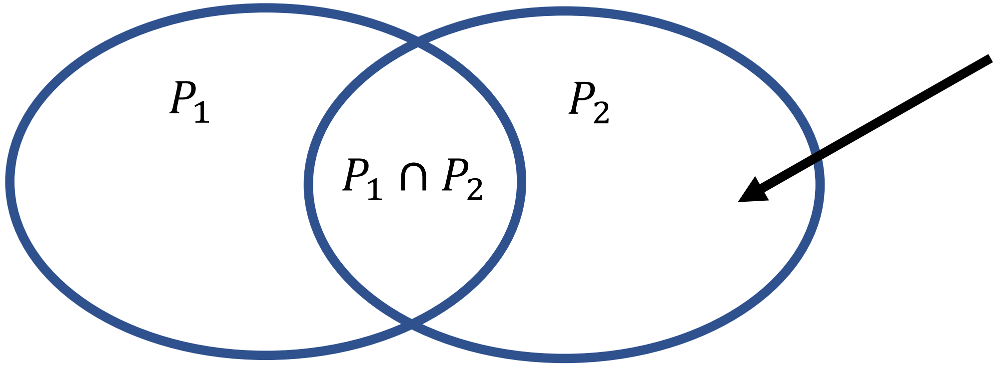
- $assert(\bar{q}; P)$ is now transformed to

$$assert(\bar{q}; P_1); assert(\bar{q}; P_2); \ldots; assert(\bar{q}; P_l)$$

- There are no dimension mismatch for these sub-assertions.

Transformation techniques

- What about rank $P > 2^{n-1}$?
  - Can we use disjunction $P = P_1 \cup P_2$?
  - Mathematically it is OK but it may lose checking efficiency



When a state is in P_2 but not in P_1, checking P_1 for it will destroy the state.

(Recall when projective measurement will not change the measured state)

  - We should not use disjunction

Transformation techniques

- What about rank $P > 2^{n-1}$?
  - We always have rank $P < 2^n = 2^{(n+1)-1}$
  - rank $P < 2^{n-1}$

Let $n := n + 1$

Transformation techniques

- What about $\mathrm{rank}\, P > 2^{n-1}$?
- Introduce an auxiliary qubit $a = |0\rangle$
- We now have $n + 1$ qubits, but we can check for P
- $\mathrm{rank}(|0\rangle_a\langle 0| \otimes P)$ = $\mathrm{rank}\, P < 2^n$
- $assert(\bar{q}; P)$ is now transformed to

$$assert(a, \bar{q}; |0\rangle_a\langle 0| \otimes P);$$

How to apply those techniques

- For $assert(\bar{q}; P)$
- If $\mathrm{rank}\, P > 2^{n-1}$, introduce an auxiliary qubit, implement

$$assert(a, \bar{q}; |0\rangle_a\langle 0| \otimes P);$$

- If $\mathrm{rank}\, P \notin \{2^{n-1}, 2^{n-2}, \dots, 2^0\}$, find an array of sub-assertions
- Apply unitary transformations to implement the assertions

What about automation?

- We need to manipulate operator of size 2^n
- Decomposing a large unitary into single-qubit gates and two-qubit gates is hard

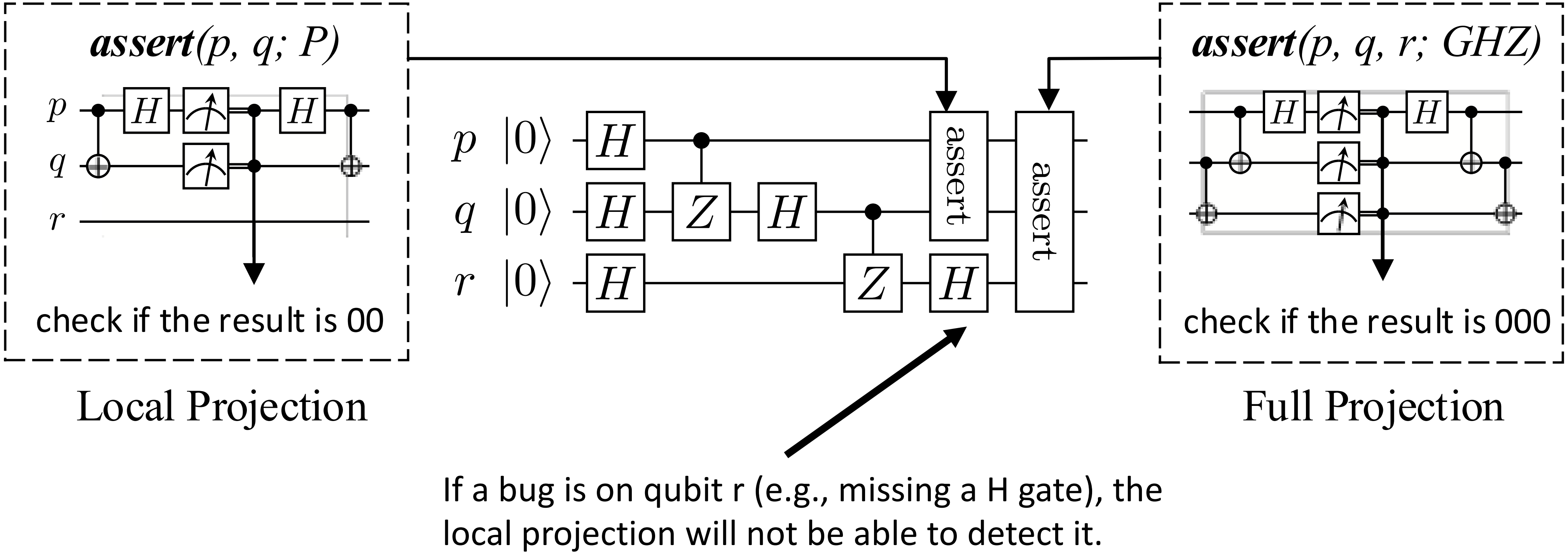
- How can we find simple implementations that are manageable by classical computers?

Local projection

- We can trade in some checking accuracy by only observing part of the entire system.
- **Key idea**: by taking the partial trace on the projection, we can find a ***sound*** approximation of the original assertion with P
- The local projection assertions will be applied a smaller number of qubits and the operators are smaller.

Local projection

- An example:



What projections we should use?

- This question is not about the assertion itself.
- We expect that the programmers should understand their programs.
- An example:
  - Chemistry simulation: simulating two electrons on four orbitals
  - $|0011\rangle$ is a two-electron state, $|0111\rangle$ is a three-electron state (not valid, the number of particles should not change in a chemical process)
  - A valid solution should be in the subspace: $span(\{|0011\rangle, |0101\rangle, |0110\rangle, |1001\rangle, |1010\rangle, |1100\rangle\})$, a 6-dimensional subspace in a 16-dimensional Hilbert space
- Projection-based assertions depend on the programs

Take-home message

- **Fundamental difficulty**: a quantum state has a very large size ($O(\exp(n))$), but one measurement can only probe limited information ($O(n)$) and may destroy the tested state.

- **Key:** Checking/Testing a state $\neq$ Knowing everything about the state
  - The difference between classical and quantum
  - In projection-based assertion checking, we only know about whether the tested state is a subspace. And we know about the target subspace.

- A '**more quantum**' assertion design for quantum PL

Summary

- We propose to check quantum program states with projections.
- A sweet point between expressive power and checking efficiency.
- We consider some practical constraints and propose several transformation techniques to make the assertions executable.

- More examples (assertion examples, numerical simulations) and formal descriptions can be found in the paper.
- https://arxiv.org/abs/1911.12855 (proofs available)

Q&A

- Thank you!