中央研究院
ACADEMIA SINICA

QUANTUM PROGRAM VERIFICATION

AN AUTOMATA-BASED APPROACH

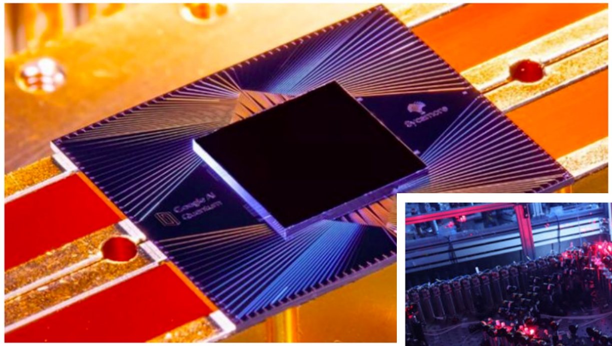
Yu-Fang Chen (陳郁方)
Institute of Information Science, Academia Sinica

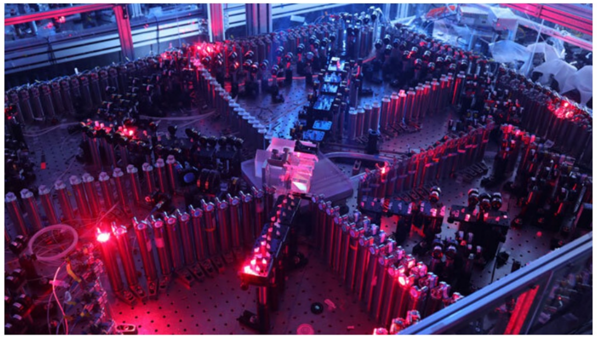
Joint work with Parosh Aziz Abdulla, Yo-Ga Chen, Kai-Min Chung, Lukáš Holík, Ondrej Lengal, Fang-Yi Lo, Jyun-Ao Lin, Wei-Lun Tsai

Sponsors: NSTC 國家科學及技術委員會 National Science and Technology Council; AFOSR Air Force Office of Scientific Research; FOXCONN

Why Quantum Computing Is Important?

- Promises:
  - Solve conventional unsolvable problems.
  - Example: break cryptography
- Algorithms for solving practical problems are under fast developing:
  - Machine learning
  - Optimization
  - Quantum chemistry





The quantum computer Jiuzhang manipulates light via a complex arrangement of optical devices (shown).
HANSEN ZHONG

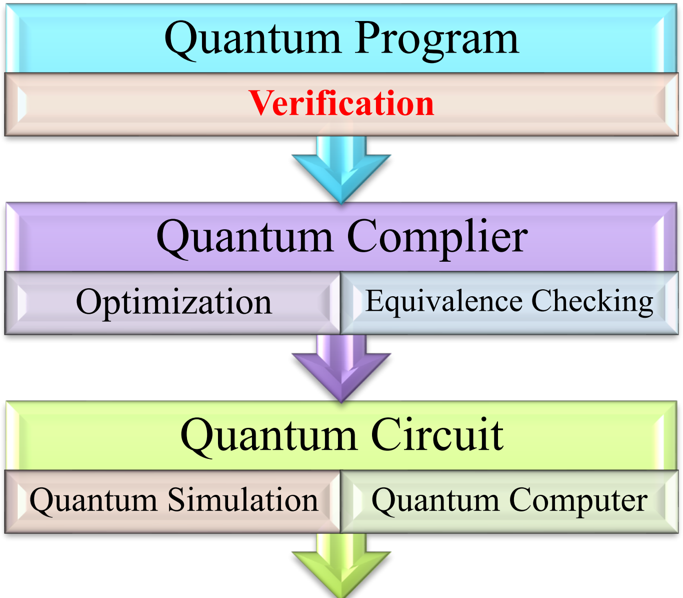
Quantum Software Stack

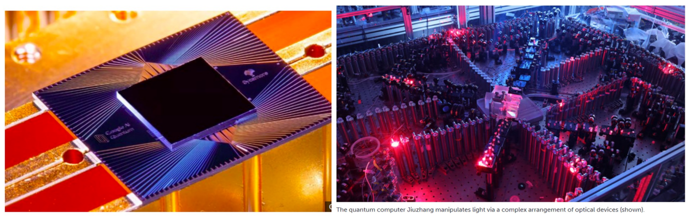
ACADEMIA SINICA 中央研究院

Q# Qiskit Cirq

OpenQASM

- Quantum computers are not standalone; they require classical software support.
- Classical software handles tasks like control flow, algorithm design, and data preprocessing.
- The synergy between quantum and classical systems is essential for potential applications.





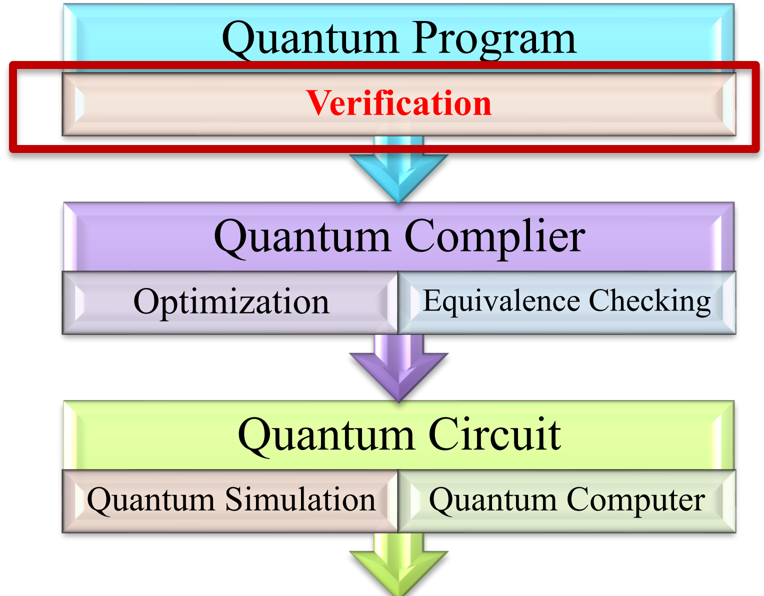
IBM	$U1^{\theta}, U2^{\theta_1,\theta_2}, U3^{\theta_1,\theta_2,\theta_3}, CX$
Rigetti	$R_x^{\pi}, R_x^{\pi/2}, R_x^{-\pi/2}, R_z^{\theta}, CZ$
Ion	$R_x^{\theta}, R_y^{\theta}, R_z^{\theta}, R_{xx}^{\theta}$

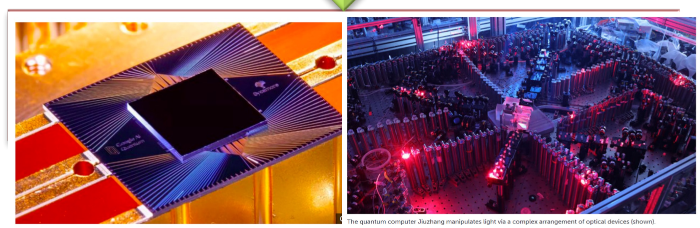
Quantum Software Stack

中央研究院 ACADEMIA SINICA

- Quantum computers are not standalone; they require classical software support.
- Classical software handles tasks like control flow, algorithm design, and data preprocessing.
- The synergy between quantum and classical systems is essential for potential applications.

Q# Qiskit Cirq OpenQASM





Rigetti	$R_x^{\pi}, R_x^{\pi/2}, R_x^{-\pi/2}, R_z^{\theta}, \mathrm{CZ}$
Ion	$R_x^{\theta}, R_y^{\theta}, R_z^{\theta}, R_{xx}^{\theta}$
IBM	$\mathrm{U1}^{\theta}, \mathrm{U2}^{\theta_1,\theta_2}, \mathrm{U3}^{\theta_1,\theta_2,\theta_3}, \mathrm{CX}$

Quantum Software Correctness

中央研究院 ACADEMIA SINICA

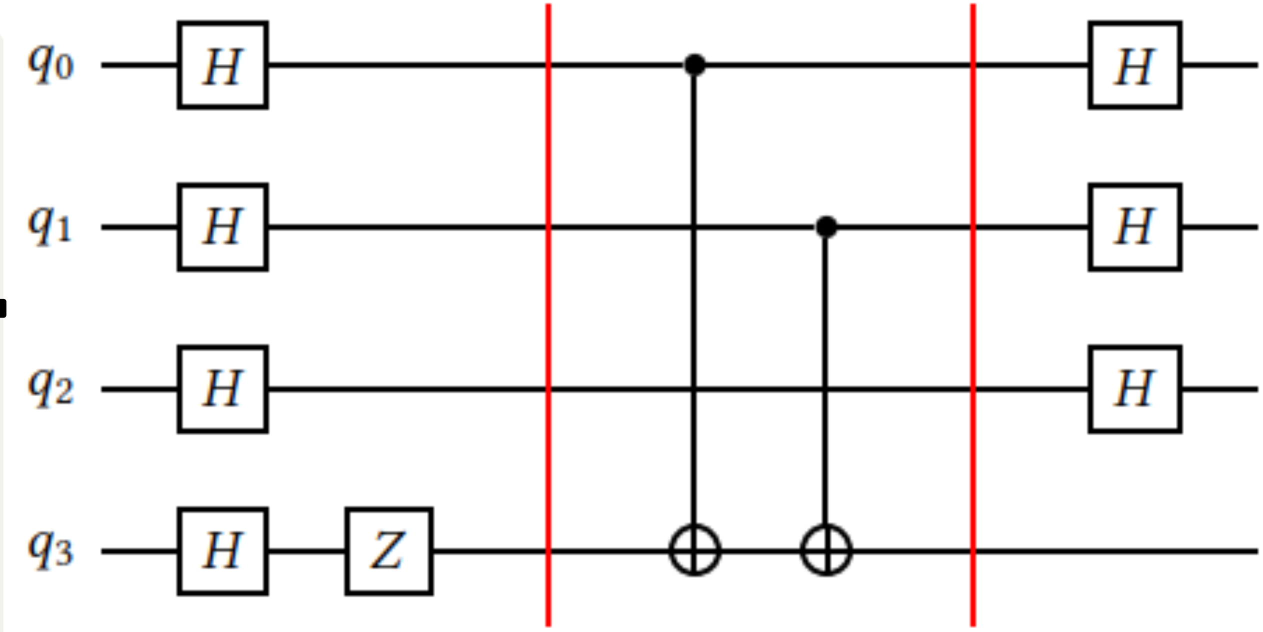
- **Challenges in Software Development**
  - Software complexity grows, making correctness harder to ensure.
  - Debugging costs exceed half of software development expenses.

- **Quantum Software Development**
  - Traditional methods struggle due to quantum's probabilistic nature.
  - Quantum states collapse upon observation, hampering traditional testing.

- **Formal Verification**
  - Provides a highly effective means of ensuring the quality of quantum software.

Verification via Examples

- Pre: The default initial state.

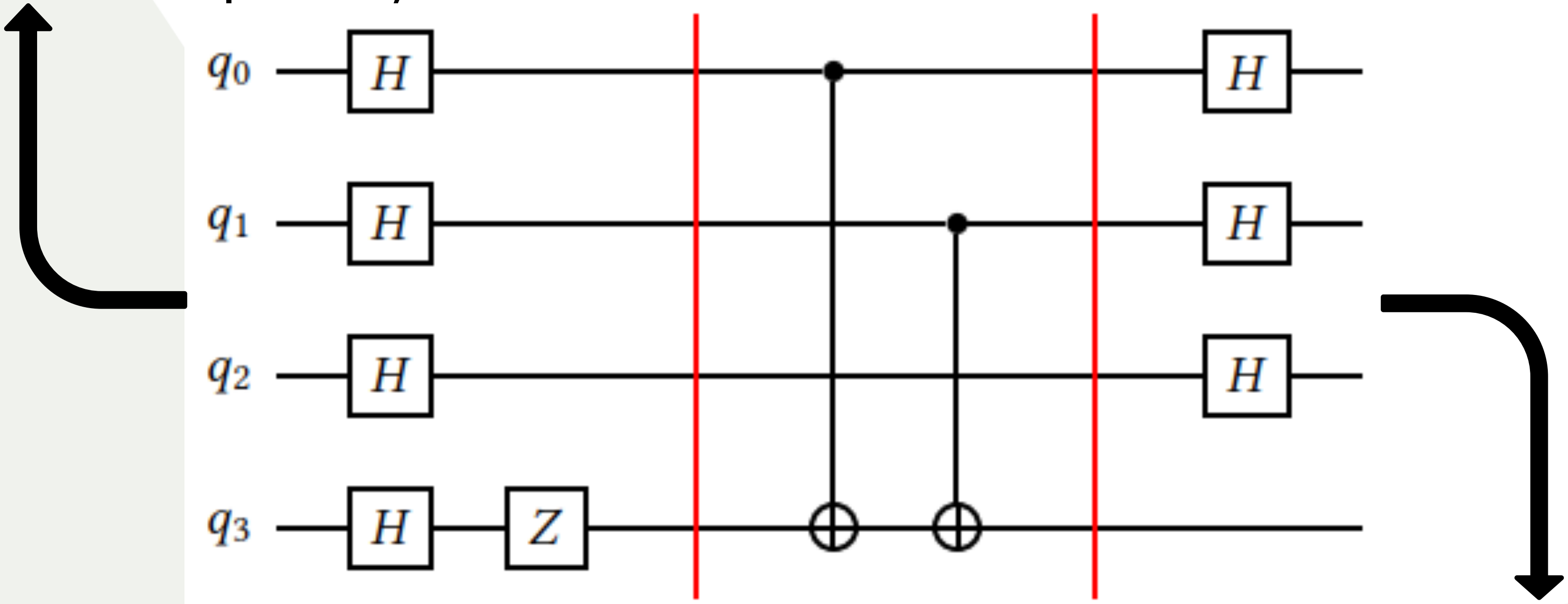


- Post: Found the hidden string (110 in this case).

Bernstein Vazirani Algorithm

Verification via Examples

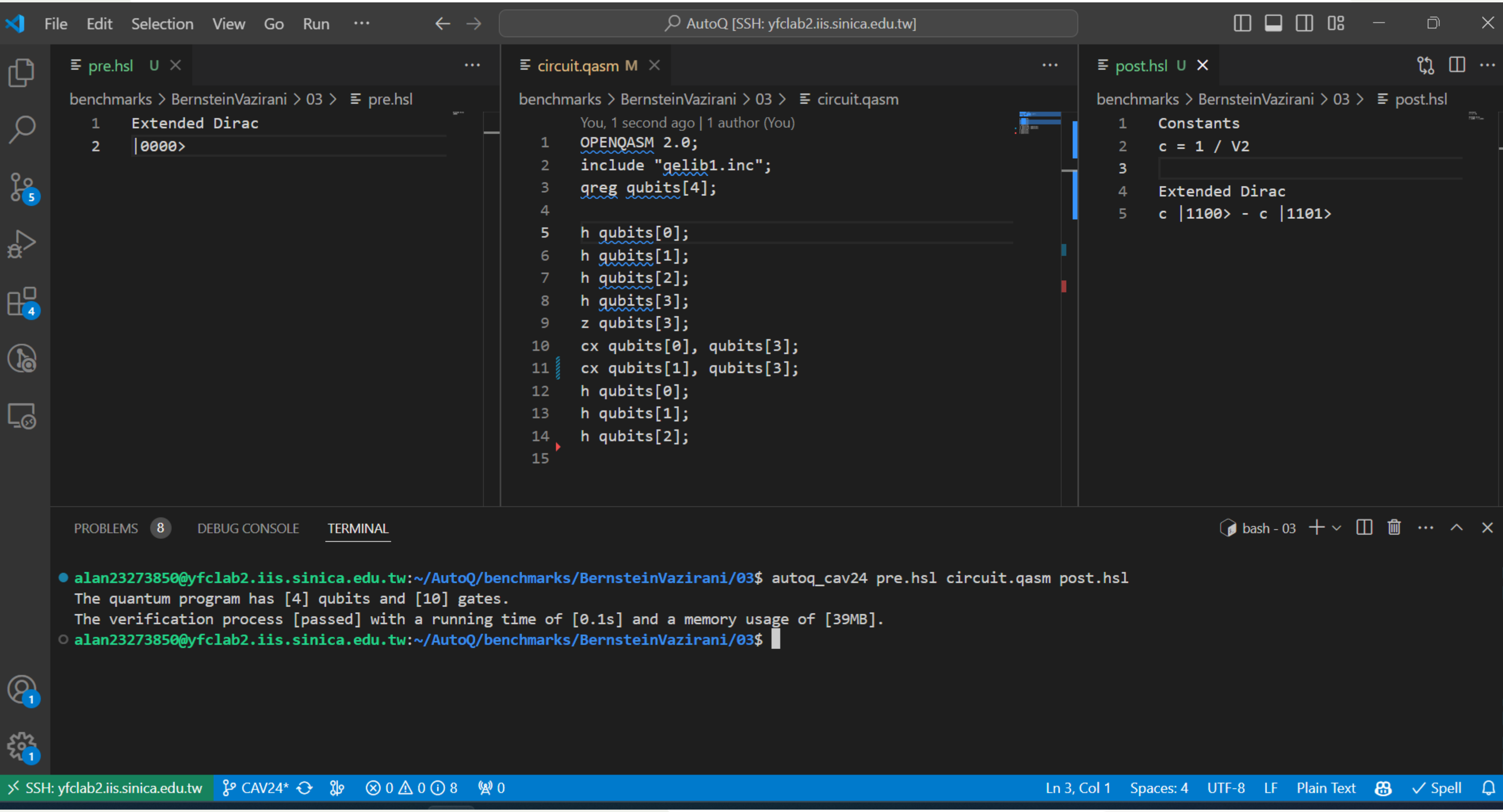
- Pre: $|0000\rangle$.



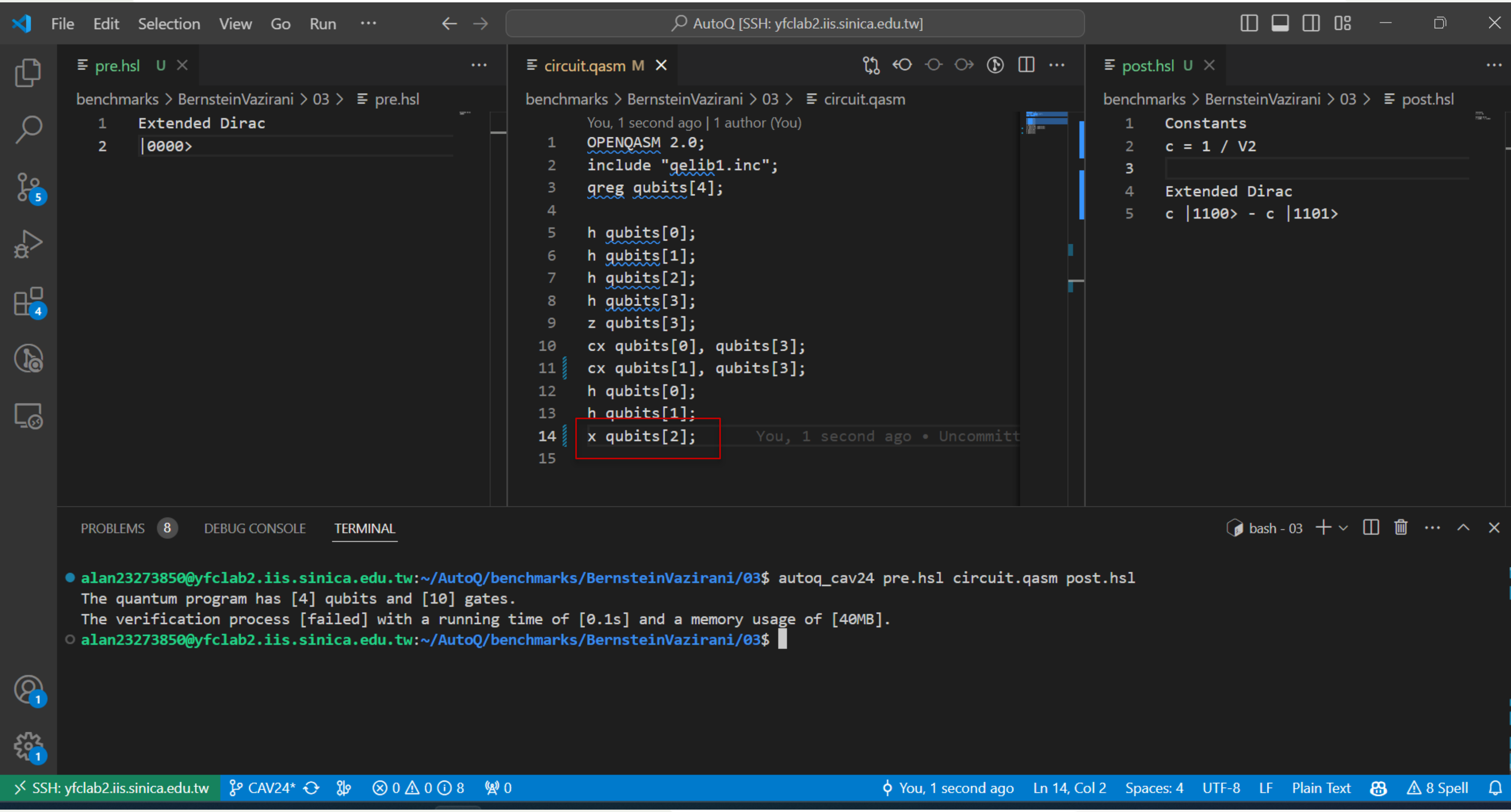
- Post: $|110-\rangle$.

Bernstein Vazirani Algorithm

Screenshots of AutoQ



Screenshots of AutoQ

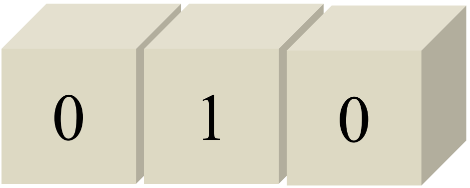


Outline

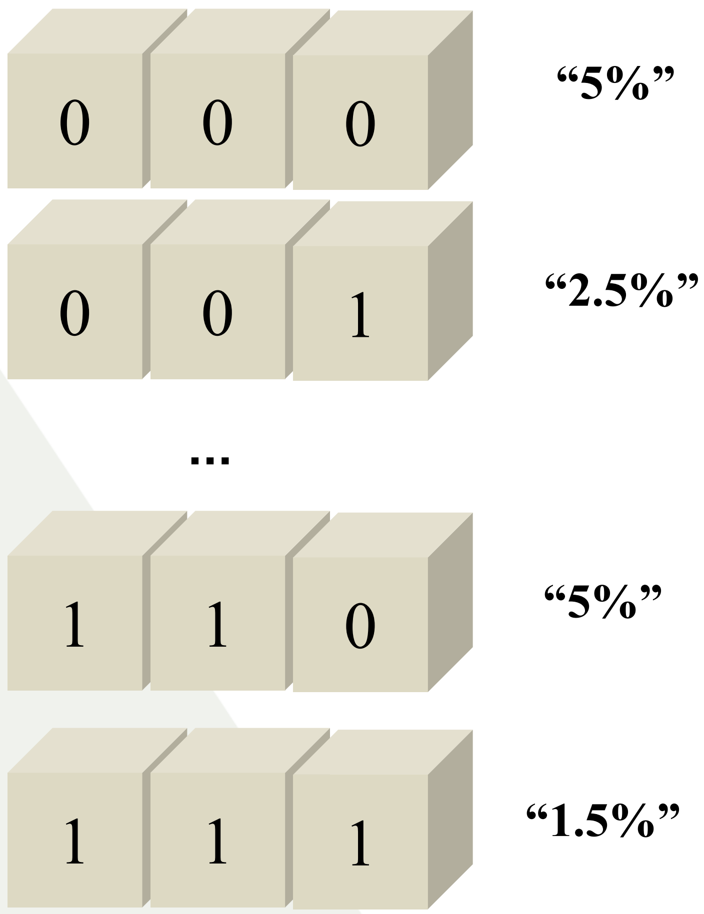
- **Quantum Background**
- Quantum Circuit Verification
- Quantum Program Verification

A 3-Bit Classical State

中央研究院
ACADEMIA SINICA

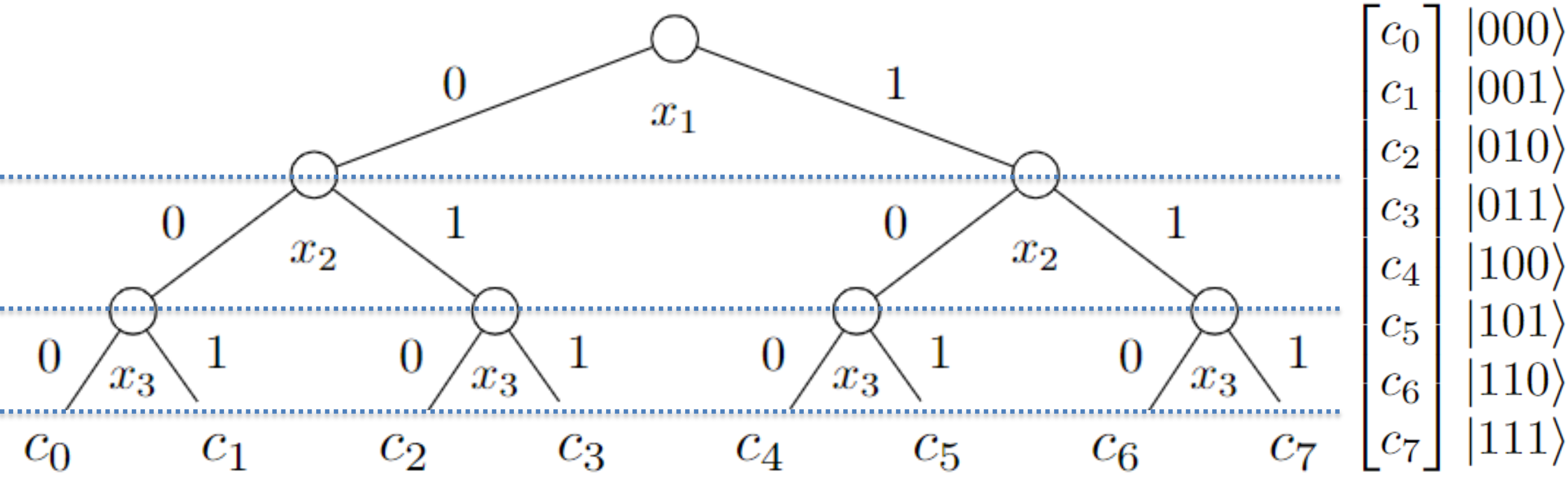


A 3-Qubit Quantum State



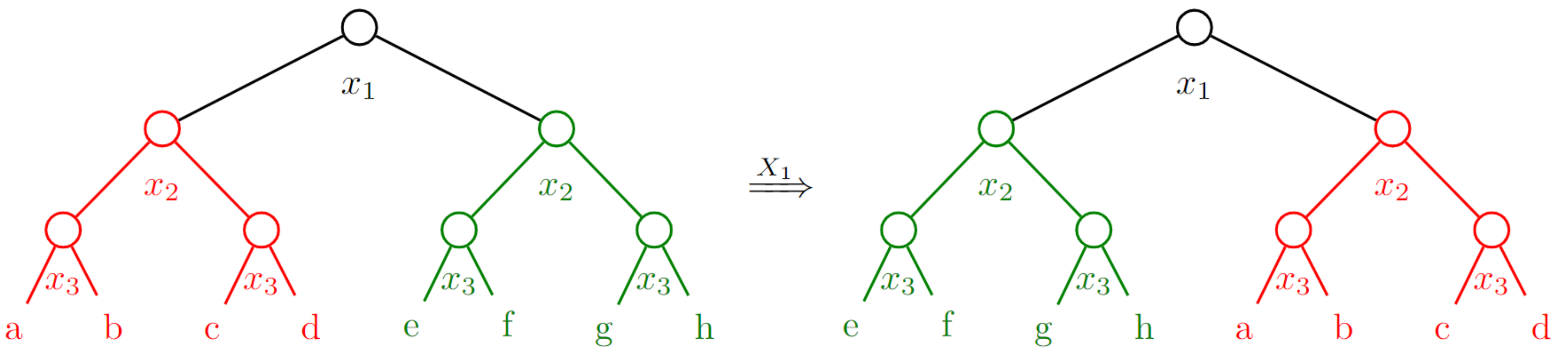
Tree as a Quantum State

- A 3-bit quantum state



Quantum Gate and Tree Transformation

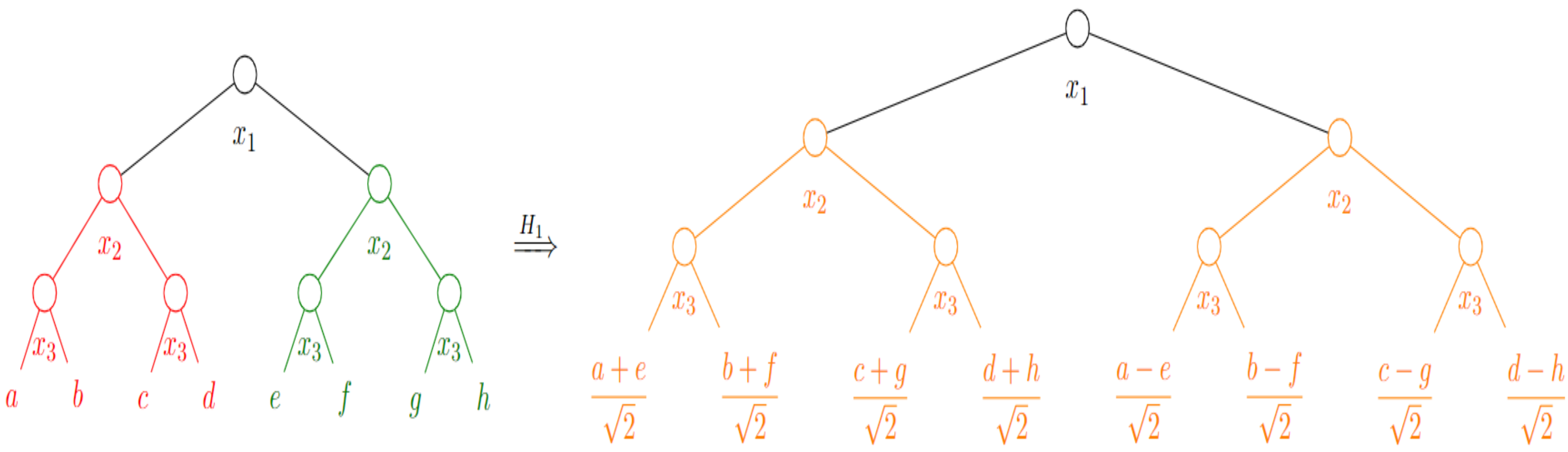
- An example of apply X gate (negation) on qubit x_1.



Quantum Gate and Tree Transformation

ACADEMIA SINICA
中央研究院

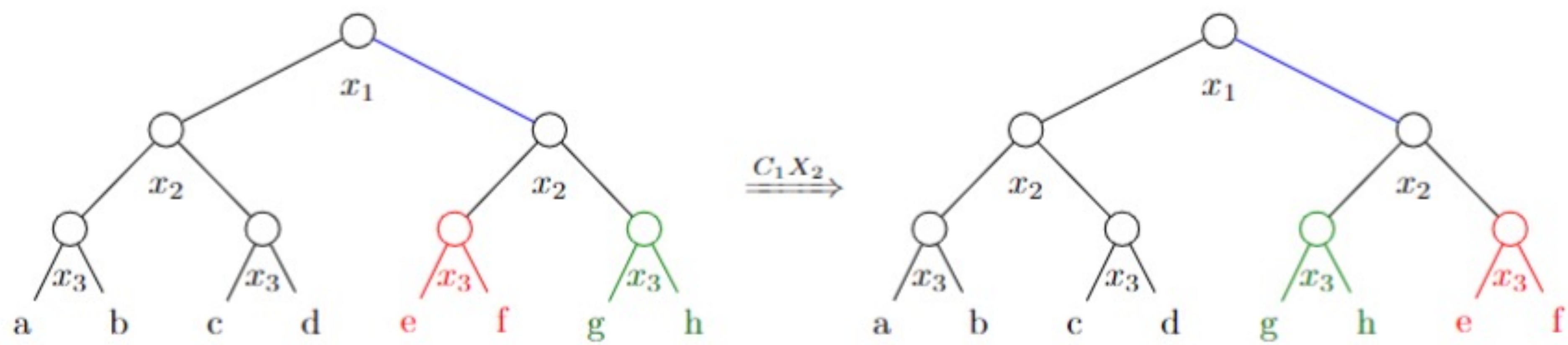
- An example of apply H gate on qubit x_1.



$|a|^2 + |b|^2 + \ldots + |h|^2 = 1$

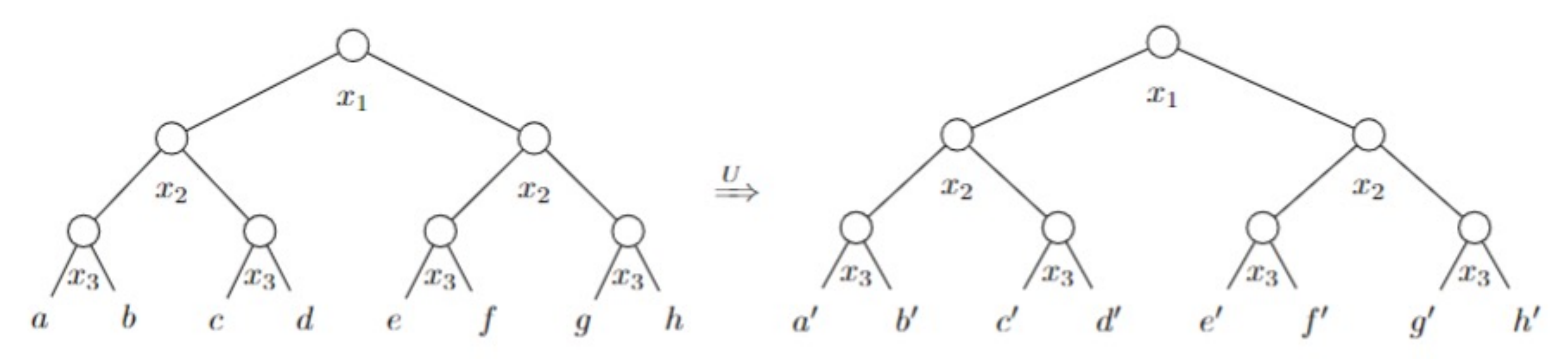
Quantum Gate and Tree Transformation

- An example of apply CX gate on control qubit x_1 and target qubit x_2.

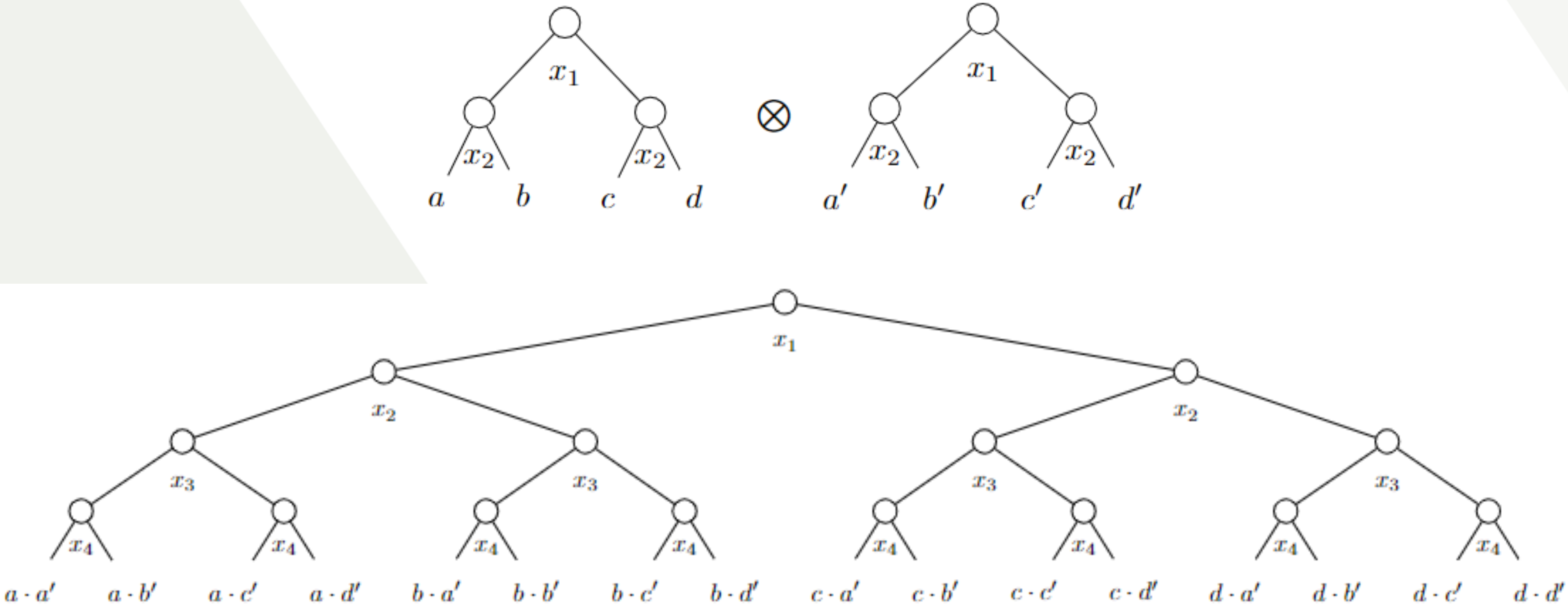


Quantum Parallelism

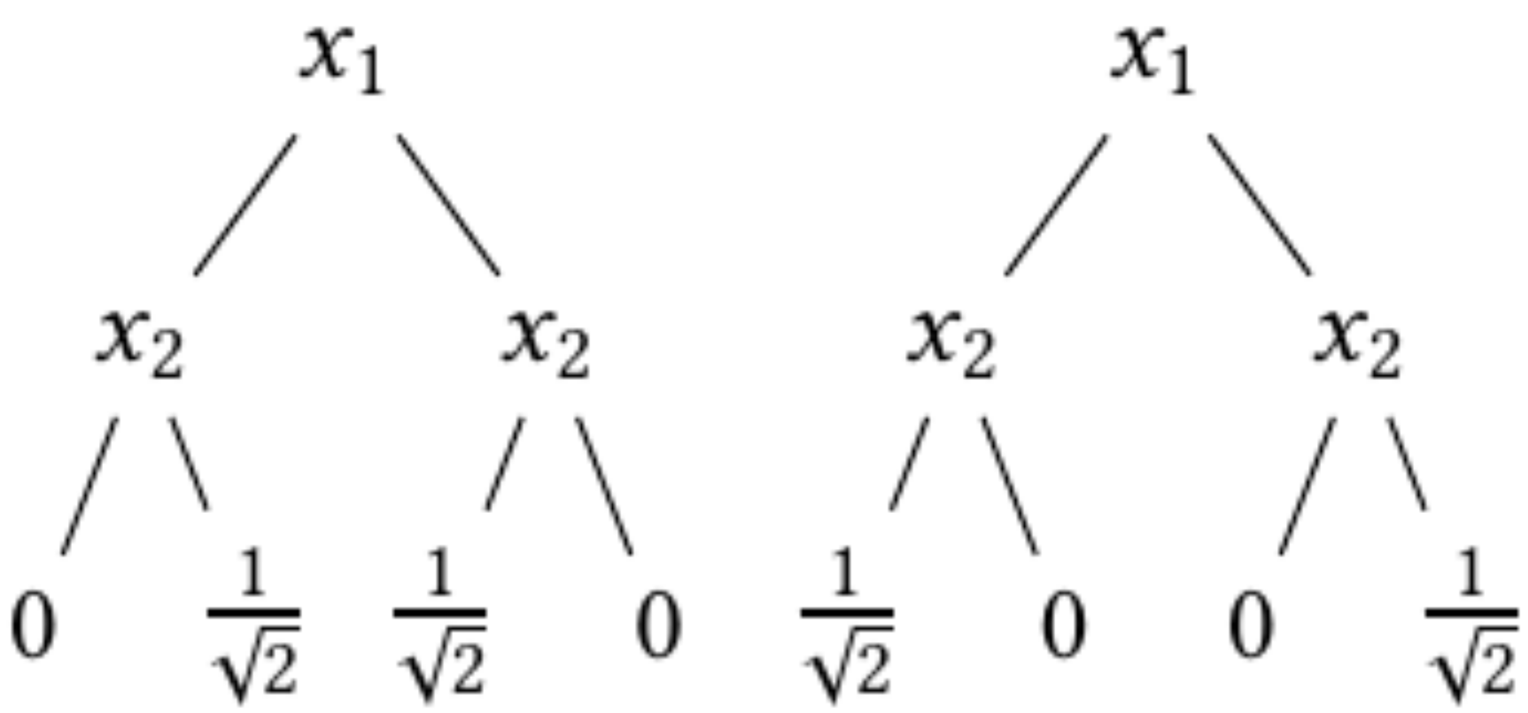
- One gate updates an exponential number of classical states



Tensor Product



Entanglement



$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

Quantum Circuit

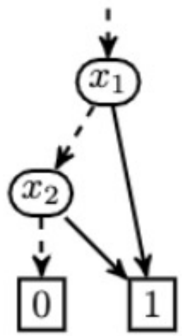
- A sequence of quantum gates viewed as tree transformations

```
x q[0];
x q[1];
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

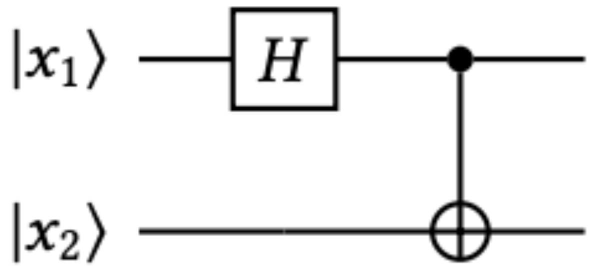
```
cx q[1],q[2];
cx q[0],q[1];
cx q[2],q[3];
tdg q[0];
tdg q[1];
tdg q[2];
t q[3];
cx q[0],q[1];
cx q[2],q[3];
s q[3];
cx q[3],q[0];
h q[3];
```

```
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
cx q[1],q[2];
cx q[0],q[1];
cx q[2],q[3];
```

Quantum Simulation

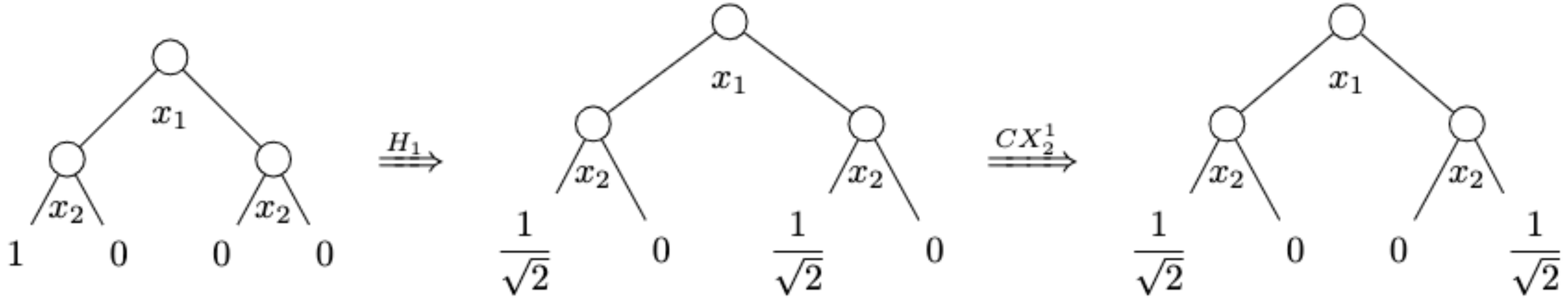


Multi-Terminal Binary Decision Diagram (MTBDD)

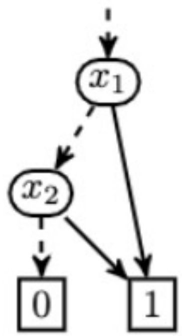


The EPR circuit

Compress



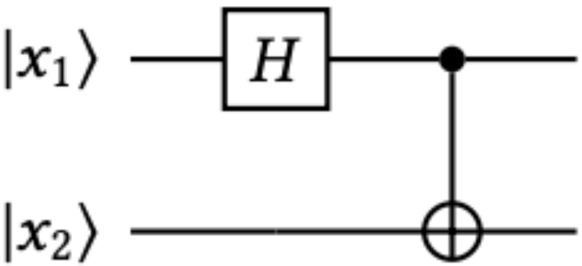
Quantum Simulation



Multi-Terminal Binary Decision Diagram (MTBDD)

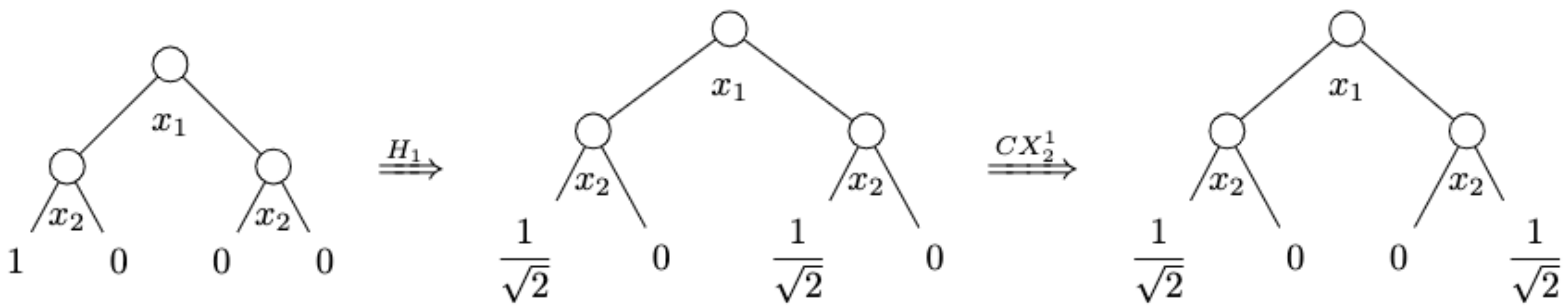
In classical verification
- BDDs encodes a set of states

In quantum, it encodes one state
- how to encode a set of state?



The EPR circuit

Compress



Outline

中央研究院
ACADEMIA SINICA

- Quantum Background
- **Quantum Circuit Verification**
- Quantum Program Verification

Classical Hoare triple

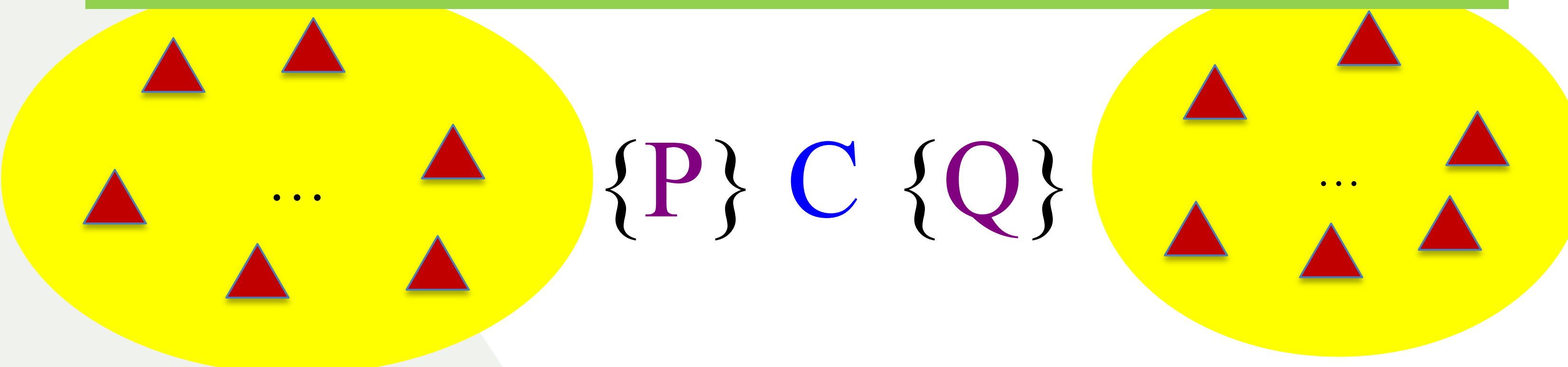
- For any predicates P and Q and any program S,

Precondition $\{P\}\ S\ \{Q\}$ Postcondition

says that if S is started in (a state satisfying) P, then it terminates in Q.

Quantum Circuit Verification

Need a symbolic representation of a set of quantum states (trees).



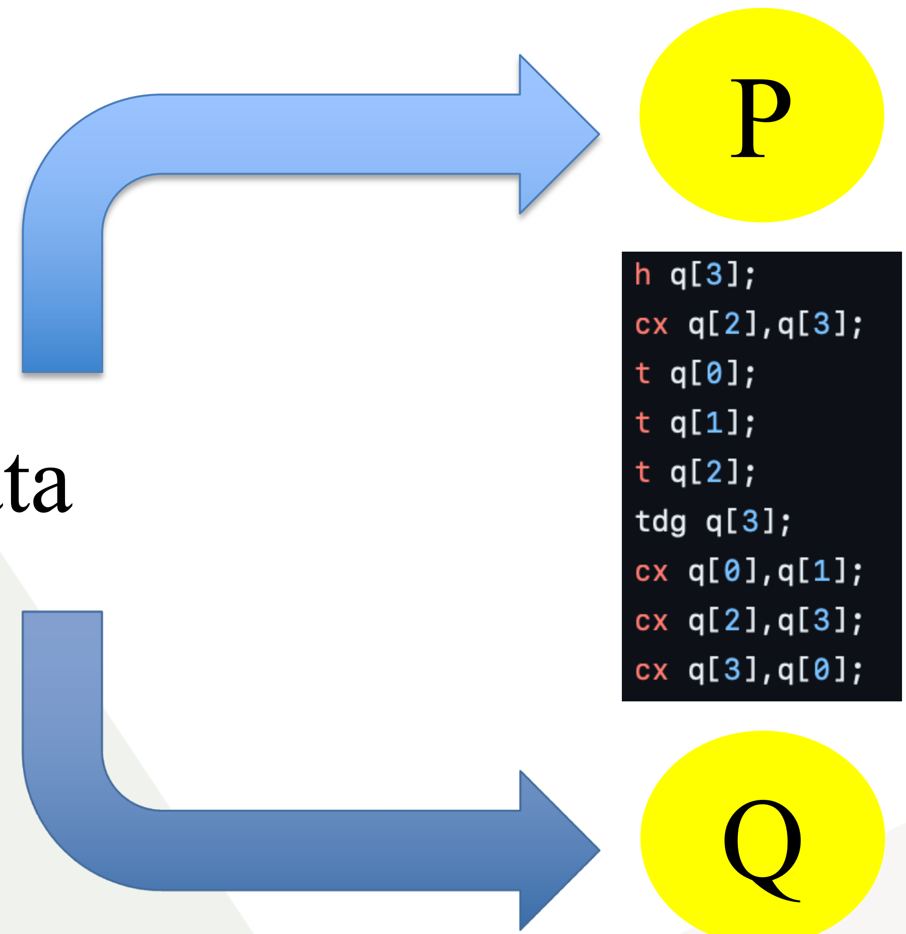
From automata theory:

Set of words ➔ Regular language (Finite automata)

Set of trees ➔ Regular tree language (Tree automata)

Overview

Tree automata




```
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

Overview

P

P^1

```
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

Q

Overview

P^2

P

```
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

Q

Overview

P

P^3

```
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

Q

Overview

P

```
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

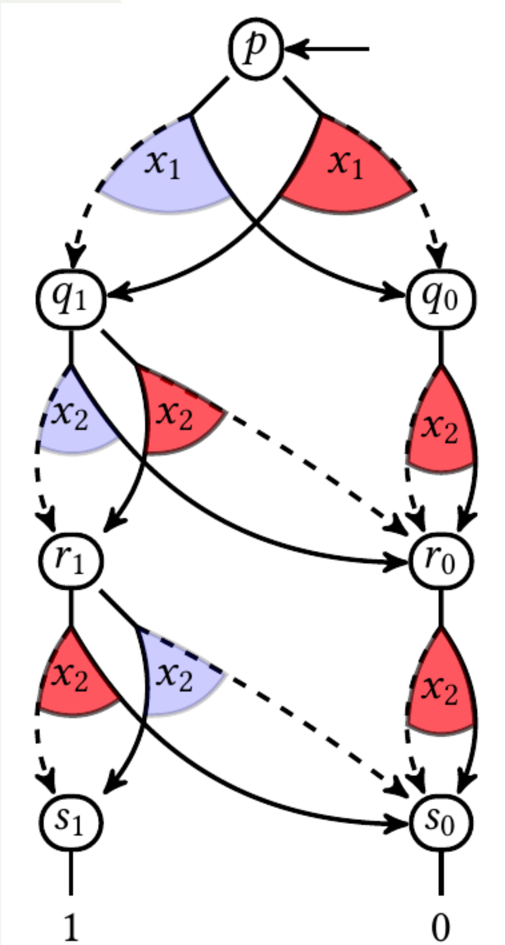
$P^{10} \subseteq Q$

(via standard TA algorithms)

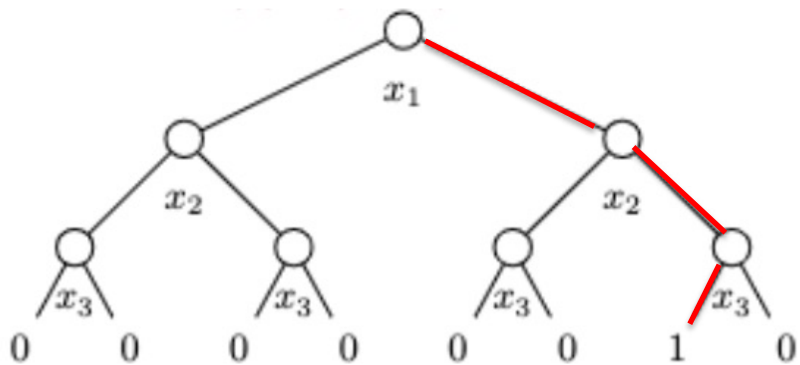
Three Components for Symbolic Verification

1. **Symbolic representation of sets of states**
2. Algorithm to compute the post image
3. Algorithm to check containment

Examples: Tree Automata Encoding of Quantum States

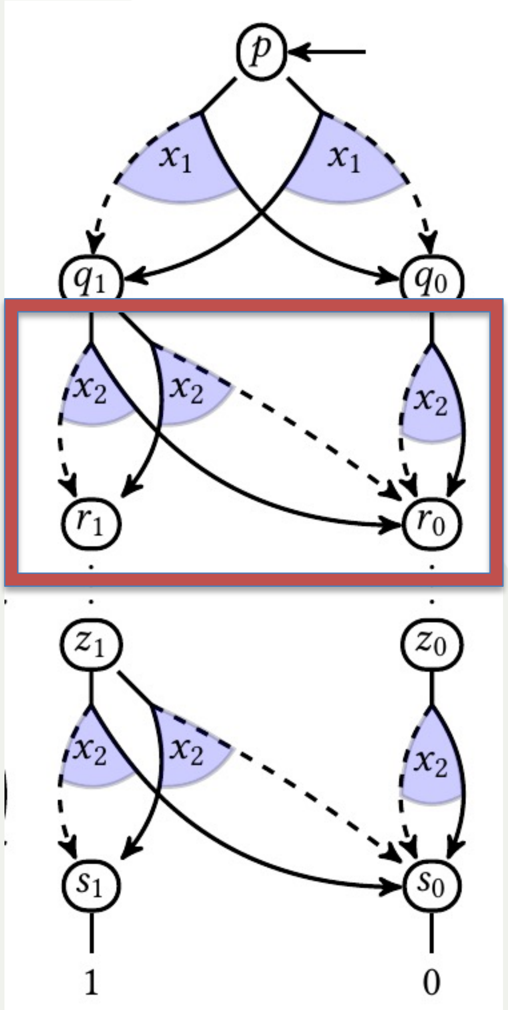


- This TA accepts all 3-qubit basis quantum states $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$



BDD + non-deterministic branching

TA as Compact Representation of Quantum States

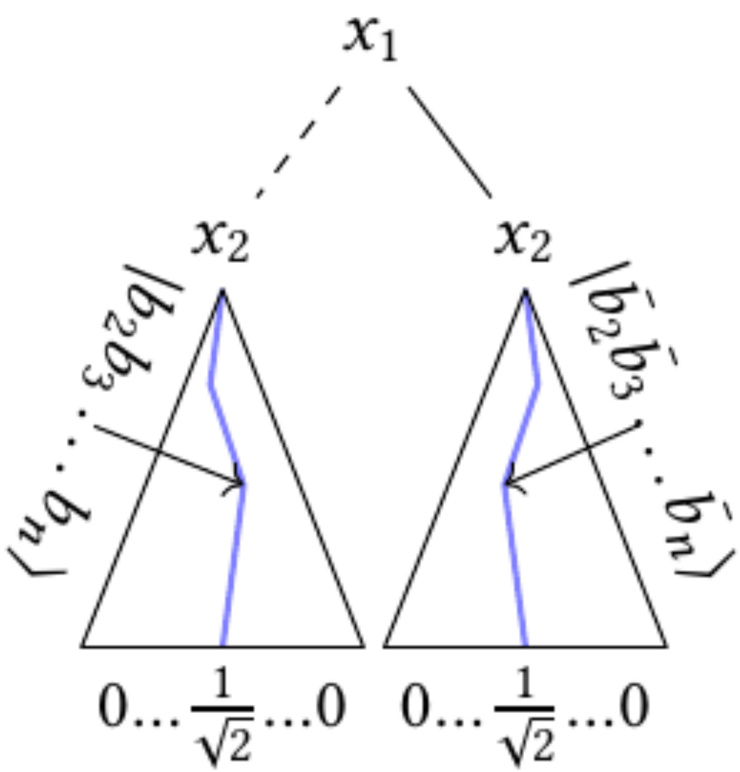


- This TA accepts all 2^n basis states.
- # of transitions: $3n+1$

Why it can be some compact? Merge shared structures.

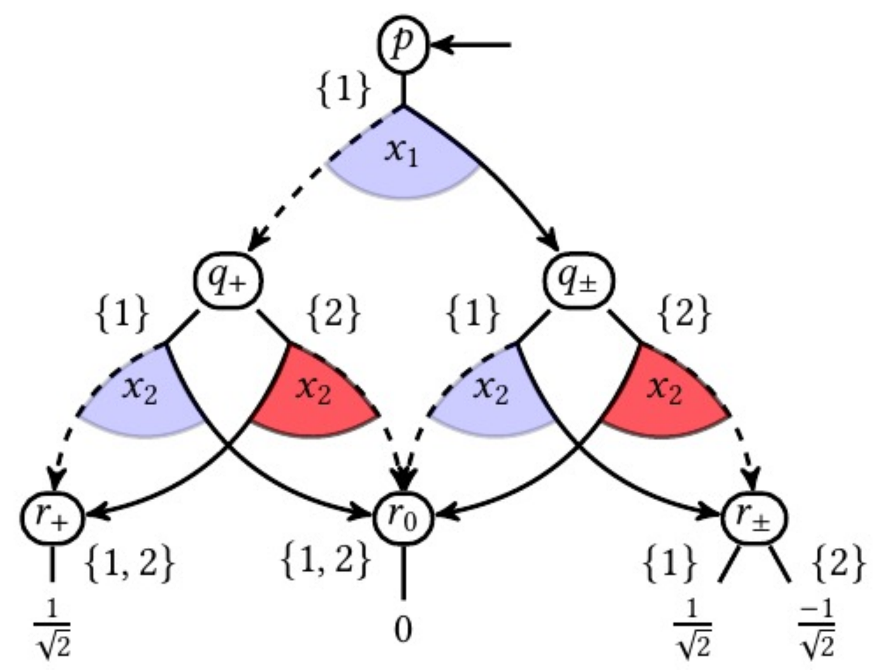
How about this

$$\{\tfrac{1}{\sqrt{2}}(|0b_2b_3\ldots b_n\rangle + |1\bar{b}_2\bar{b}_3\ldots\bar{b}_n\rangle) \mid b_2b_3\ldots b_n \in \mathbb{B}^{n-1}\}$$



- A common pattern of reachable set of quantum states.
- Need at least 2^{n-1} root transitions.

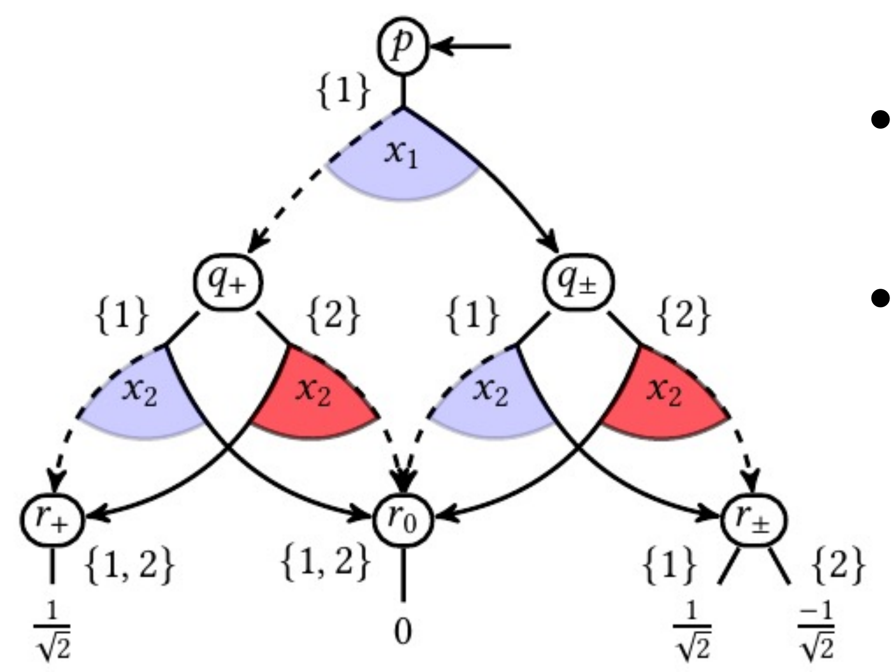
Level Synchronized Tree Automata (under submission)



- Transitions are labeled with "choices" {1},{2}, or {1,2}
- A run only allows transitions with common choice at the same level.
- Choices of transitions from one state must be disjoint.

$$\{\frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle), \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle) \mid \pm \in \{+,-\}\}$$

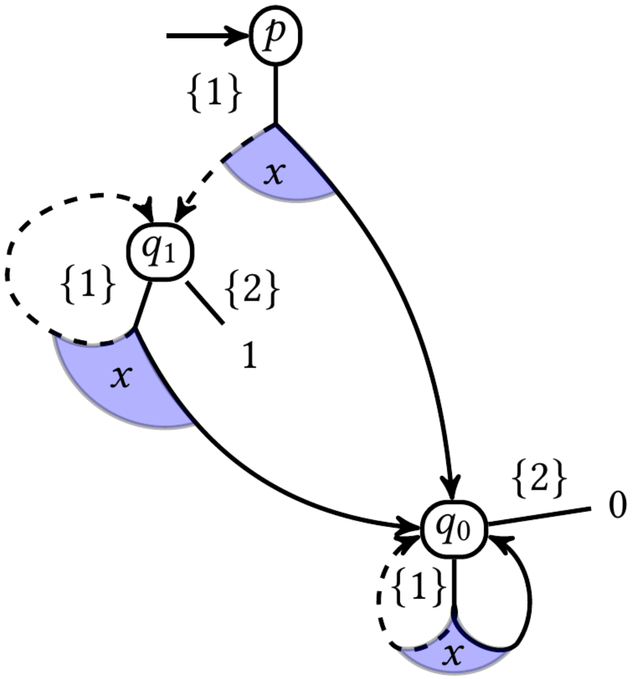
Level Synchronized Tree Automata (under submission)



- Incomparable expressiveness compared with standard TA.
- Language inclusion is decidable.

Level Synchronized Tree Automata (under submission)

中央研究院 ACADEMIA SINICA



$$\{|0^n\rangle \mid n \geq 1\}$$

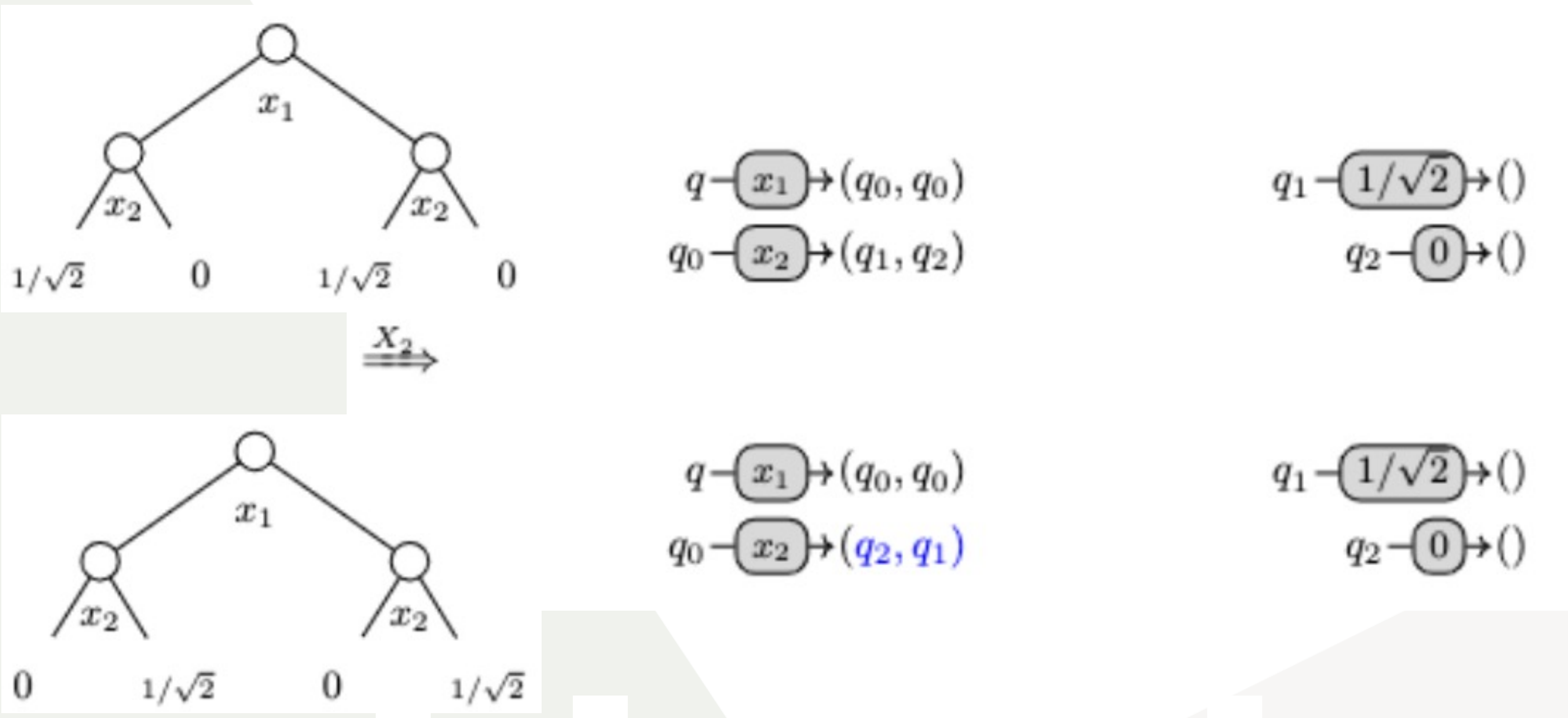
Take **choice 1** to make a next level or **choice 2** to leaves

BDD + non-deterministic branching + cycle

Three Components for Symbolic Verification

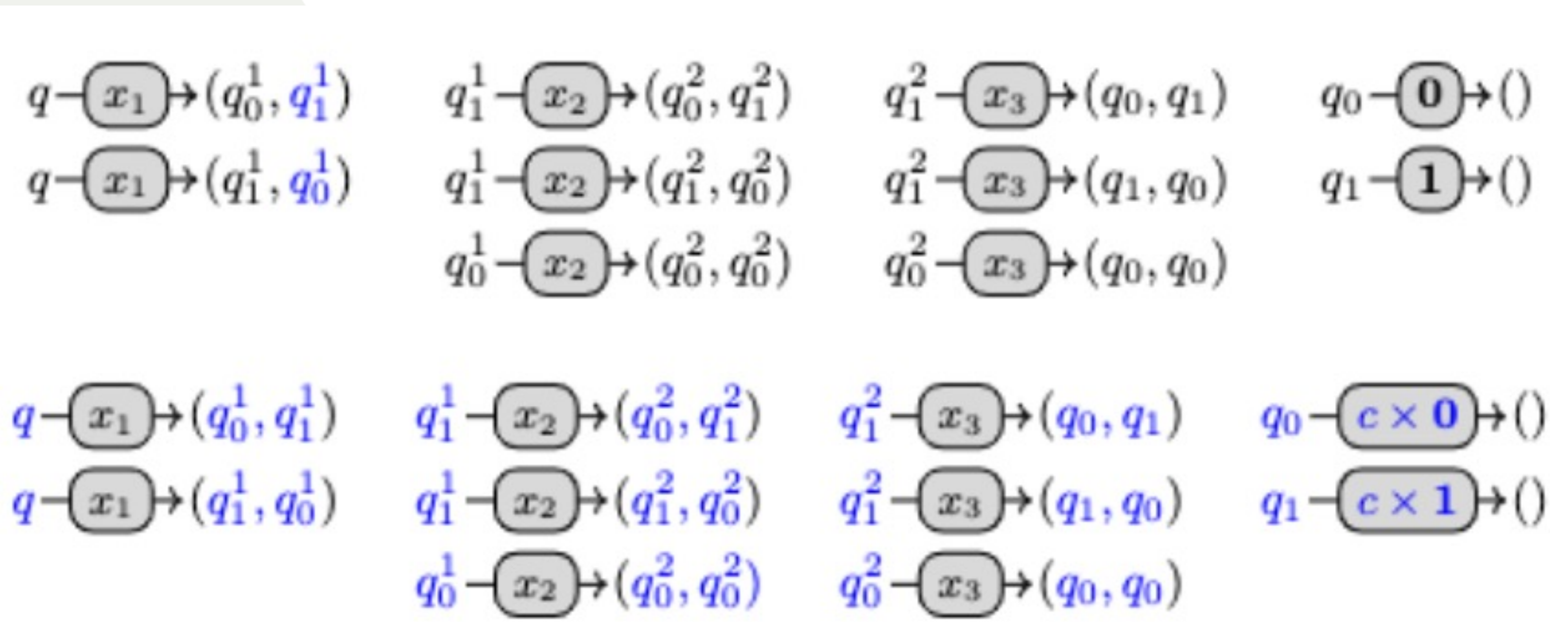
1. Symbolic representation of sets of states
2. **Algorithm to compute the post image**
3. Algorithm to check containment

Examples of Gate Operations:
X gate on qubit 2.



Example of Gate Operations:
Z, S, T gates on qubit 1.

- Multiply the right subtree of x_1 with some constant c.

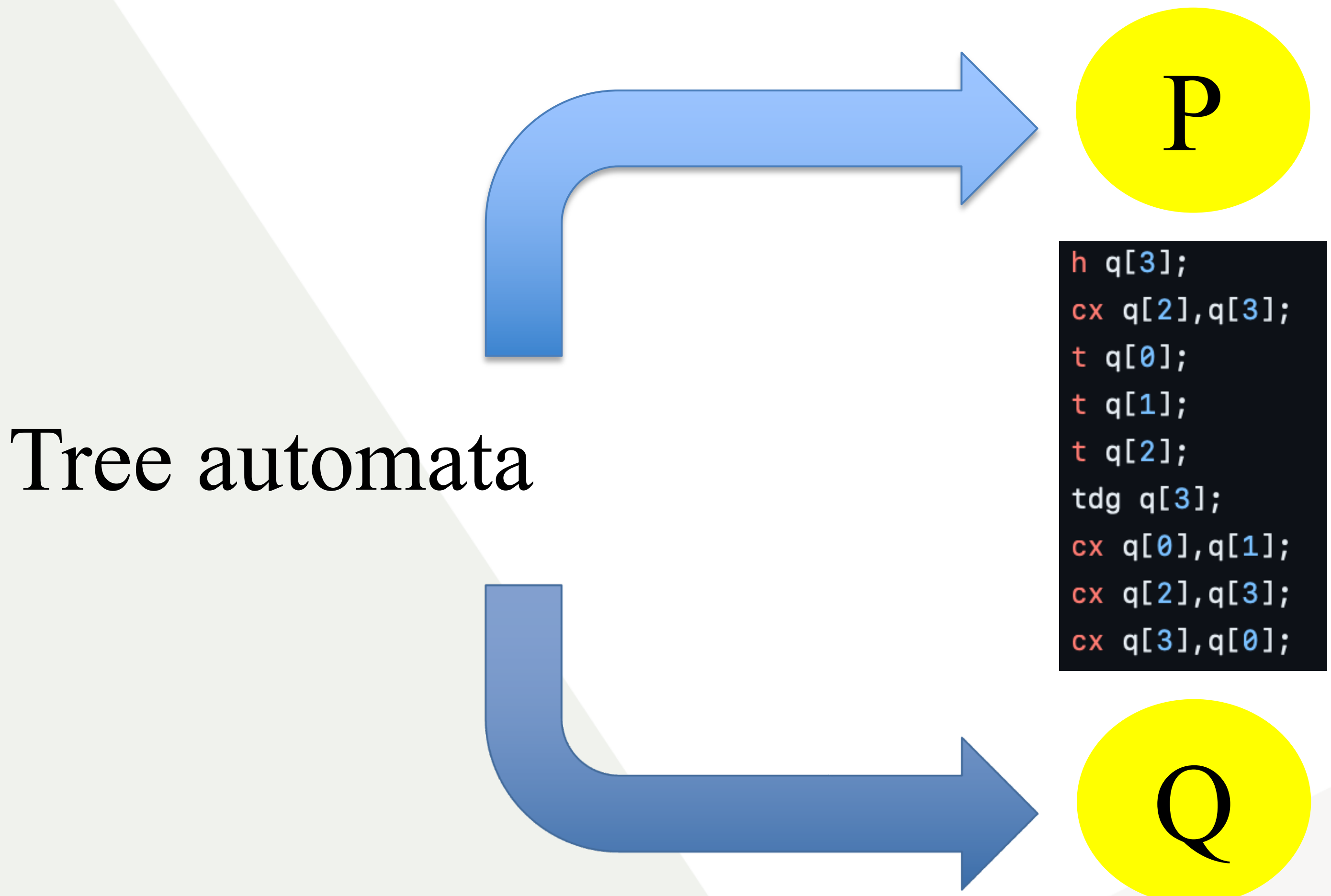


中央研究院
ACADEMIA SINICA

Three Components for Symbolic Verification

1. Symbolic representation of sets of states
2. Algorithm to compute the post image
3. **Algorithm to check containment**

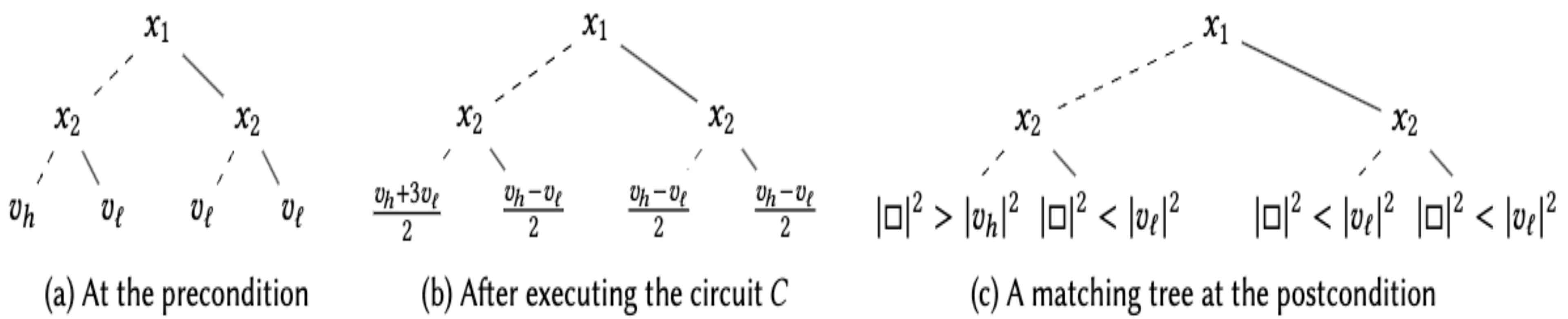
Recap: Automata-based quantum circuit verification




```
h q[3];
cx q[2],q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0],q[1];
cx q[2],q[3];
cx q[3],q[0];
```

Symbolic Extension

- Note this is different from symbolic automata



Yu-Fang Chen, Kai-Min Chung, Ondřej Lengál, Jyun-Ao Lin, Wei-Lun Tsai,
AUTOQ: An Automata-Based Quantum Circuit Verifier (CAV 2023)

Outline

中央研究院
ACADEMIA SINICA

- Quantum Background
- Quantum Circuit Verification
- **Quantum Program Verification (on-going)**

Quantum Programs

ACADEMIA SINICA

Quantum program =

Quantum circuit + Conditional statement + Loop

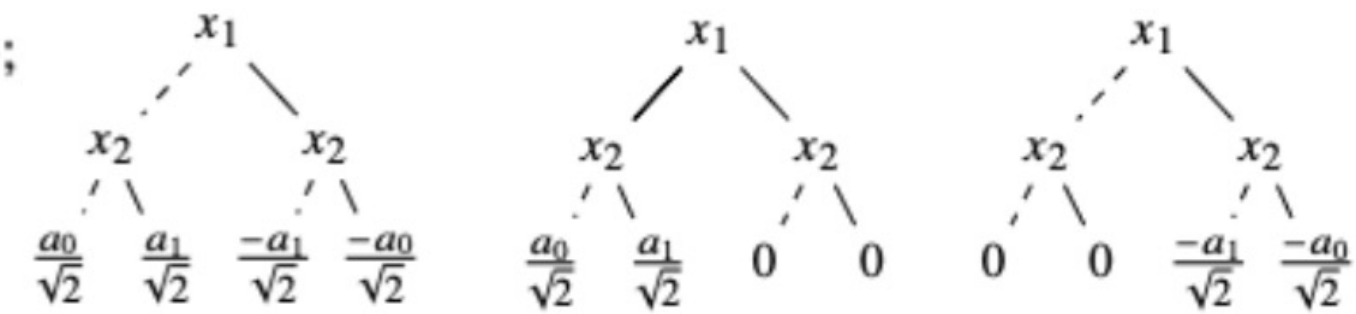
$H_1; CX_2^1;$

$\textbf{if } M_1 = 0 \textbf{ then } \{X_1\};$

$\textbf{while } M_1 = 0 \textbf{ do } \{X_1; H_1; CX_2^1\};$

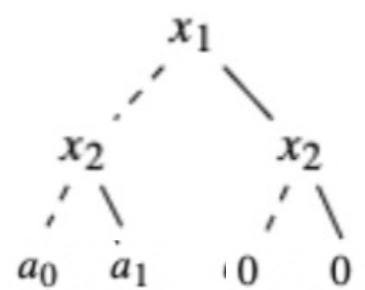
Conditional Statement

$H_1; CX_2^1;$
if $M_1 = 0$ then $\{X_1\}$;



(b) Applied $H_1; CX_2^1$ (c) $M_1 = 0$ (d) $M_1 = 1$

Normalize?



Loop Statement

Algorithm 2: "$-X_2$"

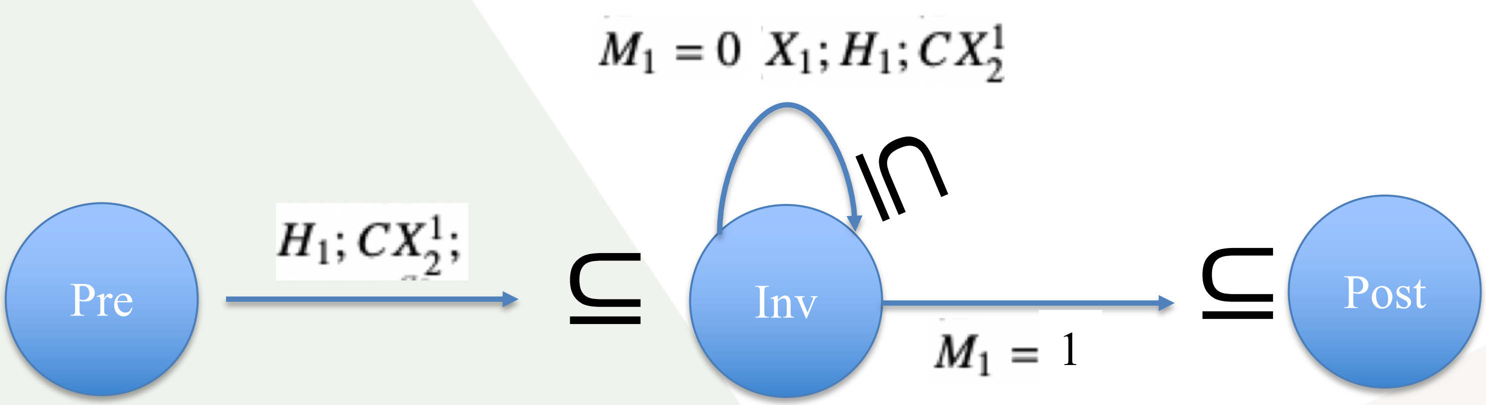
Pre: $\{(a_0 |10\rangle + a_1 |11\rangle + 0 |*\rangle)\}$;

$H_1; CX_2^1$;

Inv: $\{\frac{a_0}{\sqrt{2}} |00\rangle + \frac{a_1}{\sqrt{2}} |01\rangle - \frac{a_1}{\sqrt{2}} |10\rangle - \frac{a_0}{\sqrt{2}} |11\rangle\}$;

while $M_1 = 0$ **do** $\{X_1; H_1; CX_2^1\}$;

Post: $\{(-a_1 |10\rangle - a_0 |11\rangle + 0 |*\rangle)\}$;

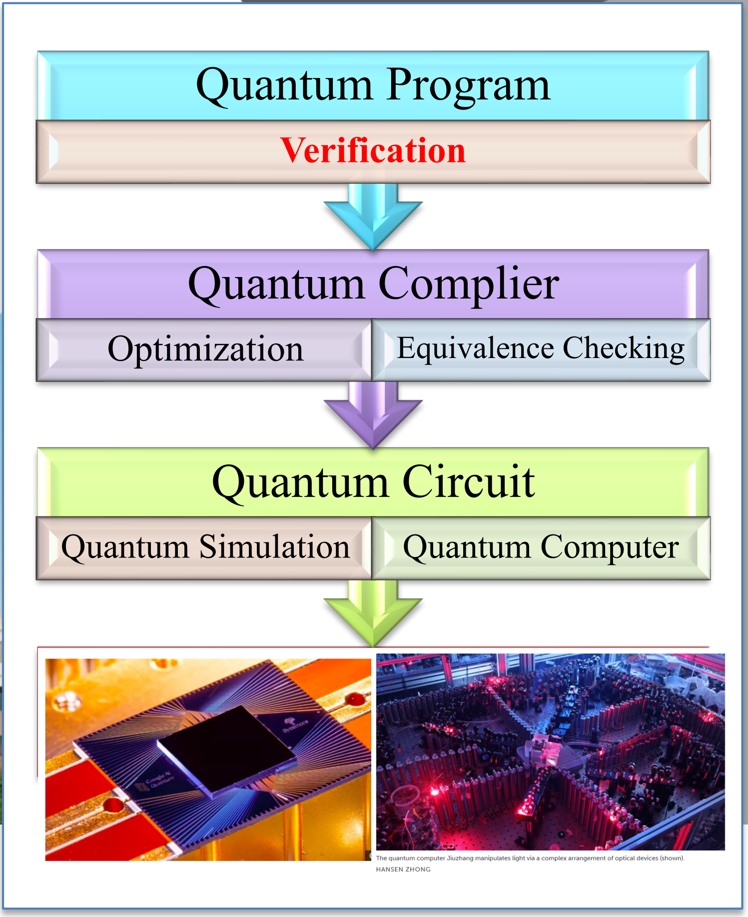


Reference

中央研究院
ACADEMIA SINICA

- An Automata-Based Framework for Verification and Bug Hunting in Quantum Circuits (PLDI 2023) https://dl.acm.org/doi/10.1145/3591270
- AUTOQ: An Automata-Based Quantum Circuit Verifier (CAV 2023) https://link.springer.com/chapter/10.1007/978-3-031-37709-9_7
- Verifying Quantum Circuits with Level-Synchronized Tree Automata (under submission)
- AutoQ 2.0: From Verification of Quantum Circuits 2 to Verification of Quantum Programs (working draft)

Thank You



Summary:

- Quantum computers are not standalone; they require classical software support.
- Formal verification is a promising approach for ensuring quantum software quality.
- Plenty of new research problems.