AutoQ 2.0: From Verification of Quantum Circuits to Verification of Quantum Programs

Yu-Fang Chen[1], Kai-Min Chung[1], Min-Hsiu Hsieh[2], Wei-Jia Huang[2], Ondřej Lengál[3], Jyun-Ao Lin[4], **Wei-Lun Tsai[1]**

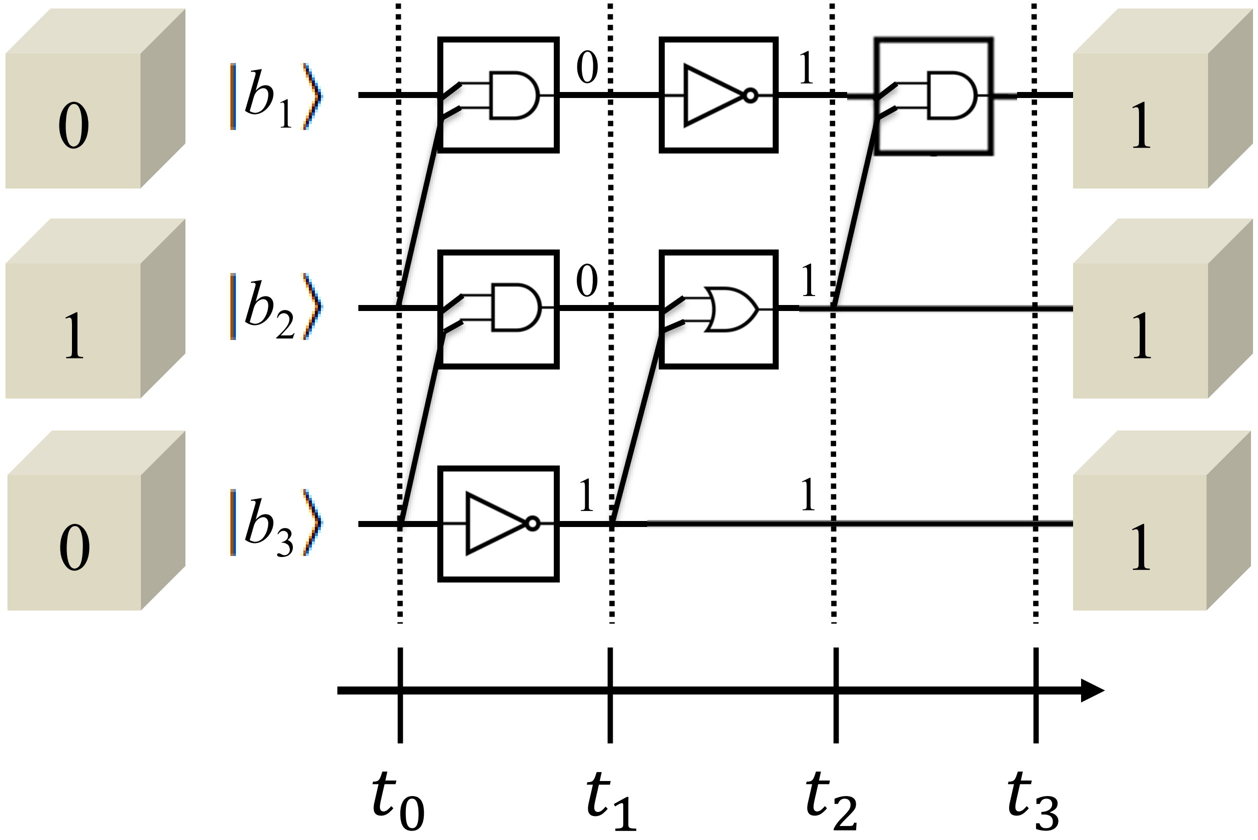
[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan,
[2] Hon Hai Quantum Computing Research Center, Taipei, Taiwan,
[3] Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic,
[4] Innovation Frontier Institute of Research for Science and Technology & Department of Computer Science and Information Engineering,
National Taipei University of Technology, Taipei, Taiwan

Outline

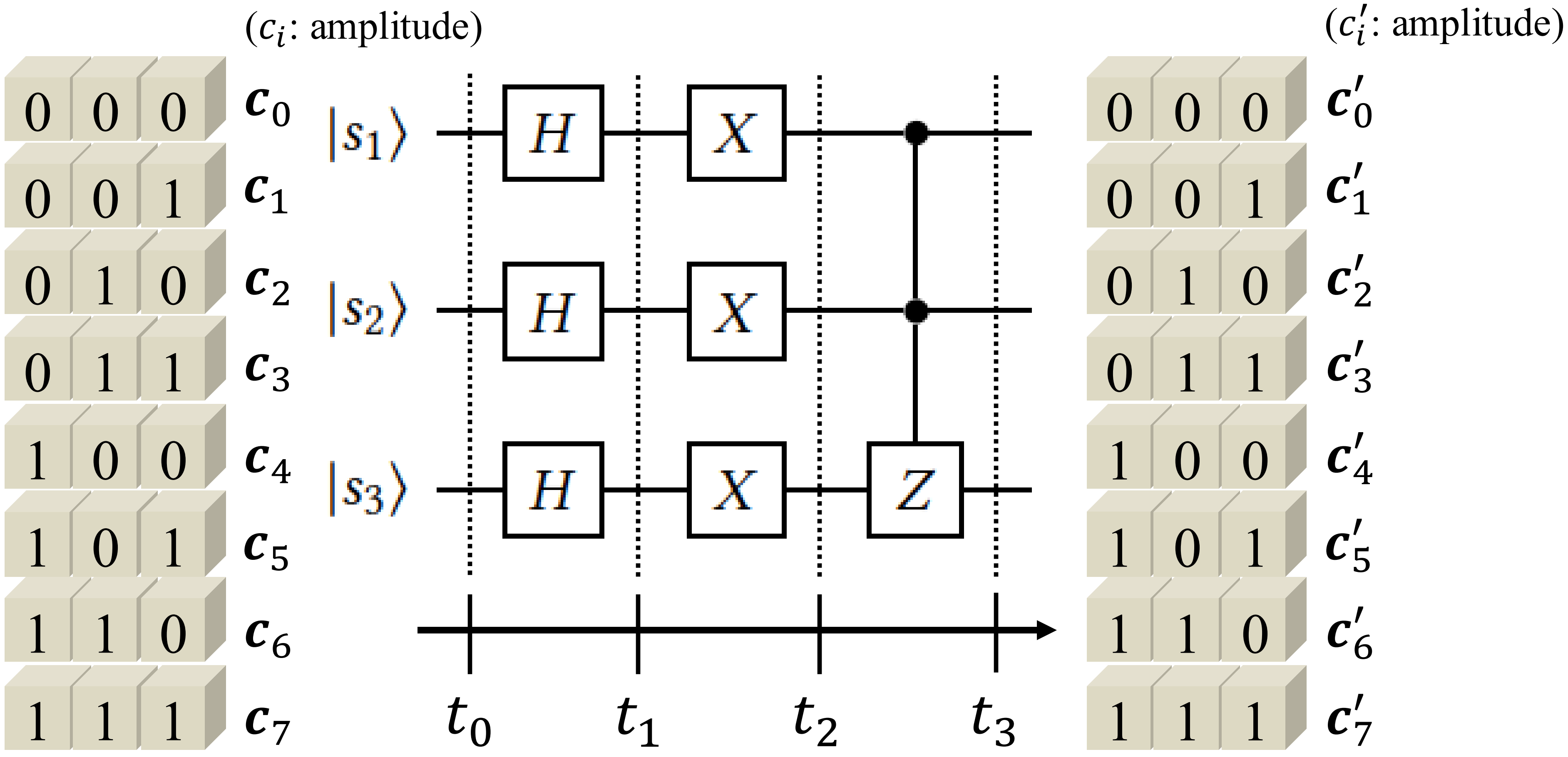
- ☐ Basics of Quantum Computing
- ☐ AutoQ 1.0: A Quantum (Circuit) Verification Framework
- ☐ AutoQ 2.0: From Quantum Circuits to Quantum Programs
- ☐ Possible Improvement and Summary

Basics of Quantum Computing

A Classical Circuit

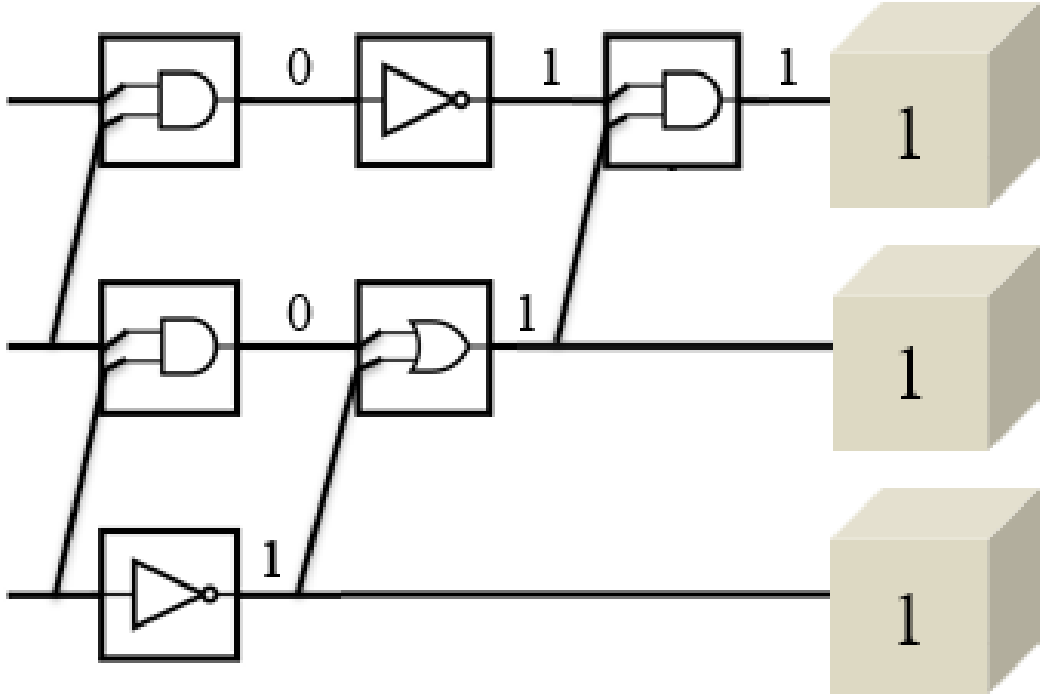


Generalized to a Quantum Circuit

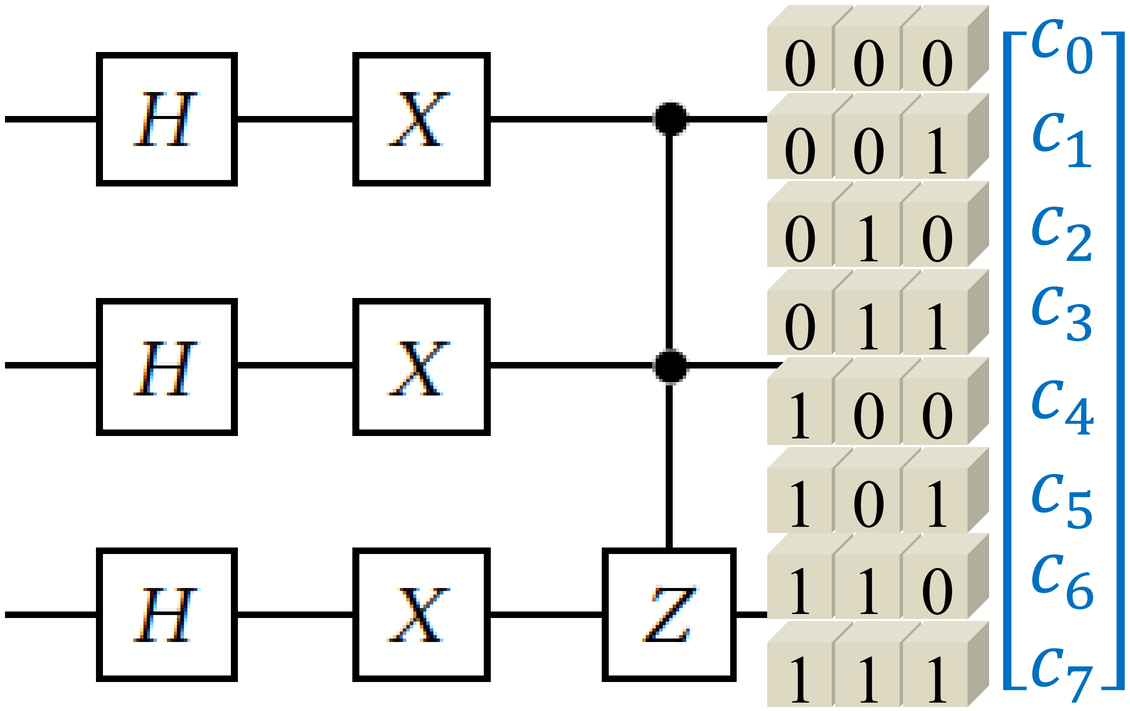


Dirac notation: $c_0|000\rangle + c_1|001\rangle + c_2|010\rangle + c_3|011\rangle + c_4|100\rangle + c_5|101\rangle + c_6|110\rangle + c_7|111\rangle$

Classical States vs Quantum States



which is a binary string of length 3

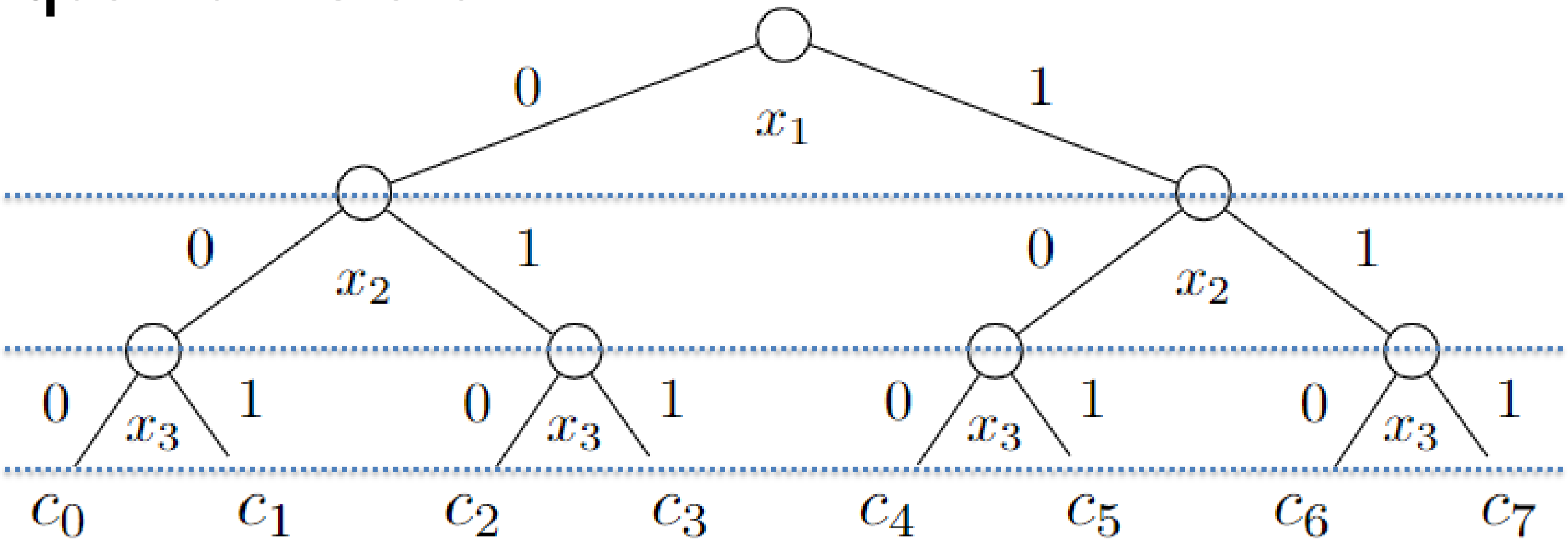


which we call a "**state vector**" of dimension 2^3

(serving the role of an element in a vector/Hilbert space)

From Decision Trees' Perspectives

- Motivation: To utilize the succinct tree-like model in POPL 2025.
- A 3-bit quantum state.



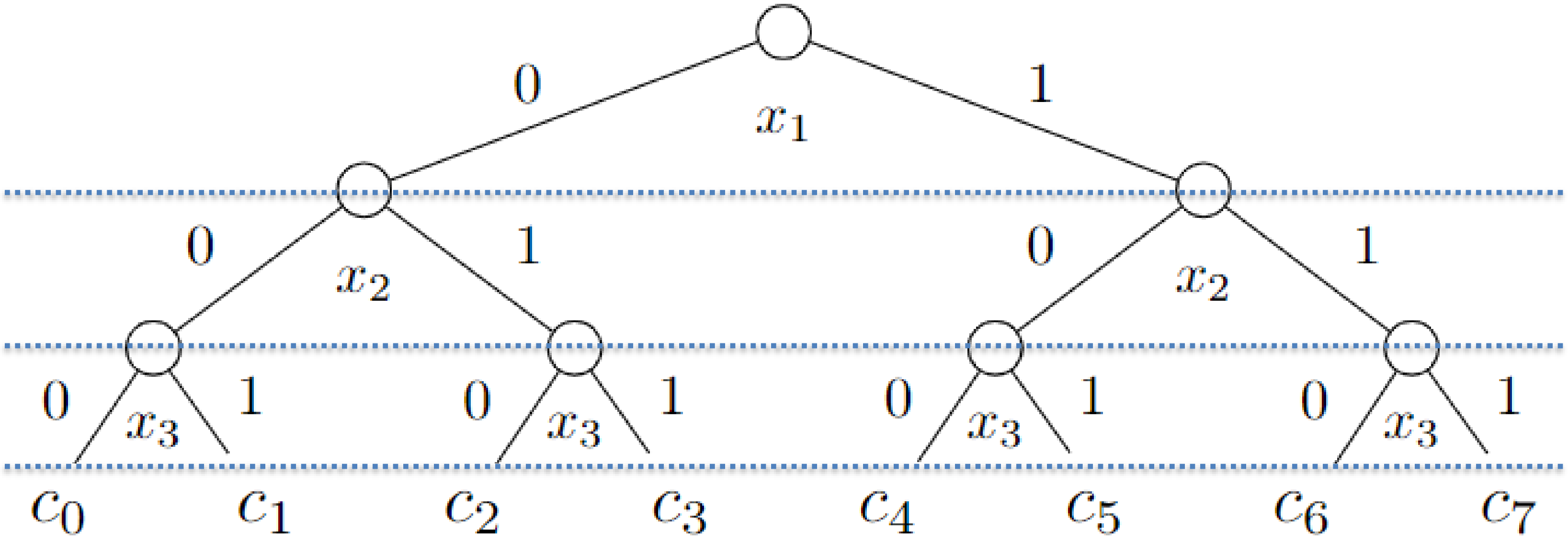
$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

Dirac notation: $c_0|000\rangle + c_1|001\rangle + c_2|010\rangle + c_3|011\rangle + c_4|100\rangle + c_5|101\rangle + c_6|110\rangle + c_7|111\rangle$

From Decision Trees' Perspectives

- A 3-bit quantum state.

Message 1: A quantum state is a decision tree.

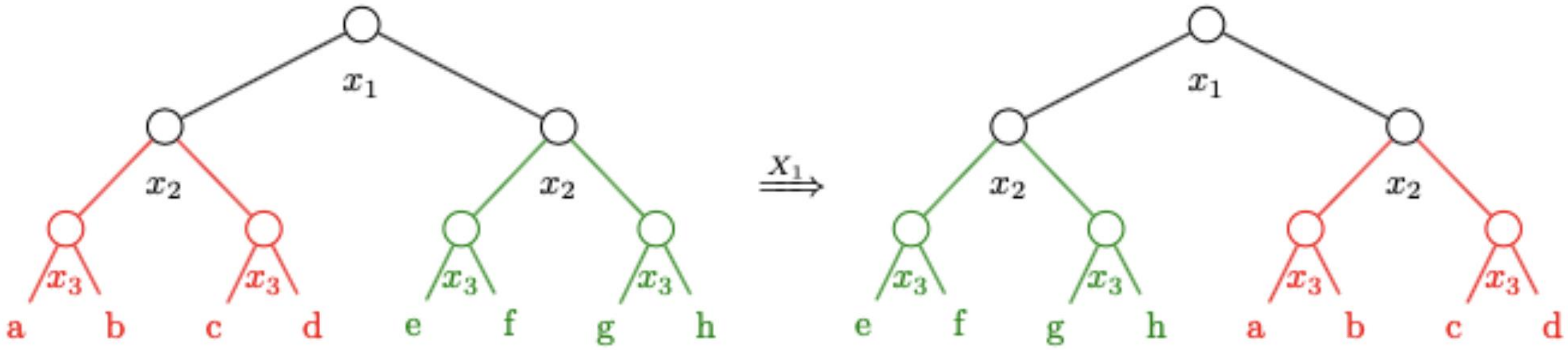


$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

Dirac notation: $c_0|000\rangle + c_1|001\rangle + c_2|010\rangle + c_3|011\rangle + c_4|100\rangle + c_5|101\rangle + c_6|110\rangle + c_7|111\rangle$

Quantum Gates = Tree Transformations

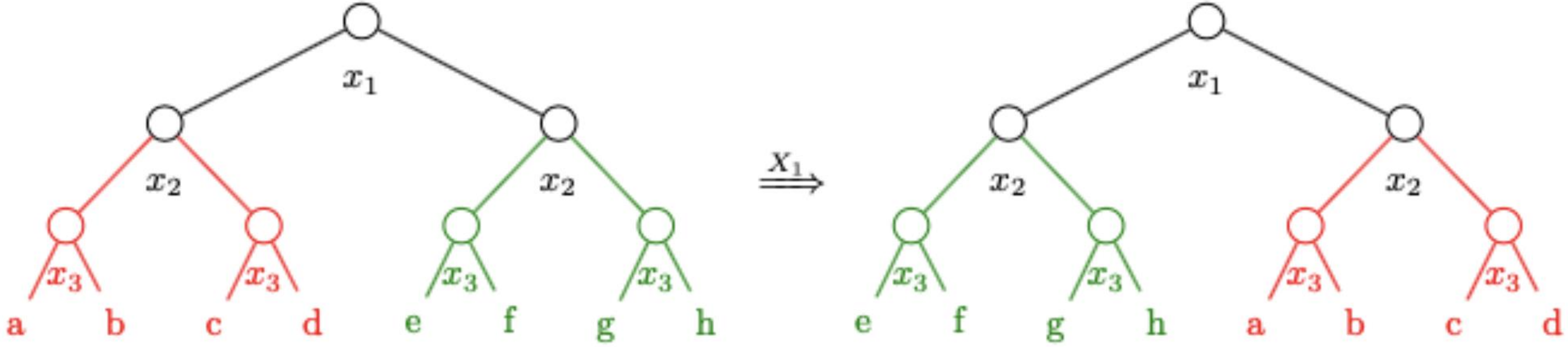
- An example of applying an X (negation) gate on qubit x_1.



Quantum Gates = Tree Transformations

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

Message 2: A quantum gate is just a tree transformation.



Supported Quantum Gates

TABLE I
QUANTUM GATES SUPPORTED IN THIS WORK.

Gate	Symbol	Matrix
Pauli-X (X)	$\oplus$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -\imath \\ \imath & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S)	S	$\begin{bmatrix} 1 & 0 \\ 0 & \imath \end{bmatrix}$
T	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{\imath\pi/4} \end{bmatrix}$
Rx($\frac{\pi}{2}$)	$R_x(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\imath \\ -\imath & 1 \end{bmatrix}$
Ry($\frac{\pi}{2}$)	$R_y(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$
Controlled-NOT (CNOT)	$\bullet$ / $\oplus$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Z (CZ)	$\bullet$ / Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Toffoli	$\bullet$ / $\bullet$ / $\oplus$	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

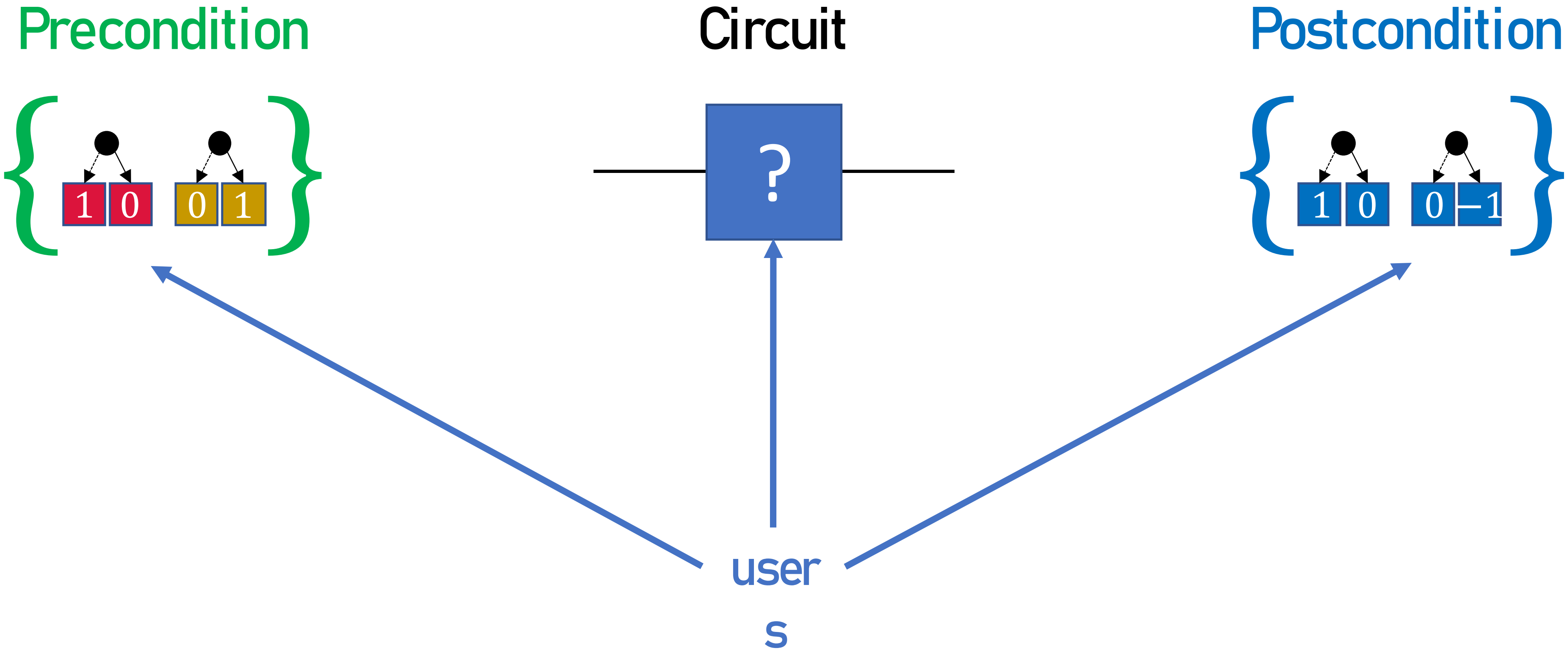
https://github.com/fmlab-iis/AutoQ/

AutoQ 1.0

Automata-based Quantum (Circuit) Verification

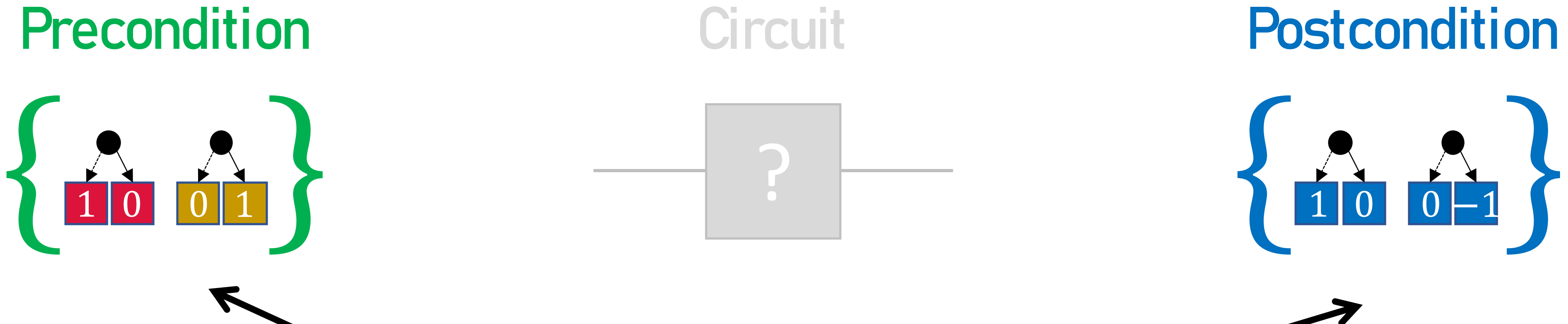
AutoQ 1.0 Automata-based Quantum Verification

A Hoare triple would be like …



AutoQ 1.0 Automata-based Quantum Verification

A Hoare triple would be like ...



Set of quantum states

- The definition of validness?

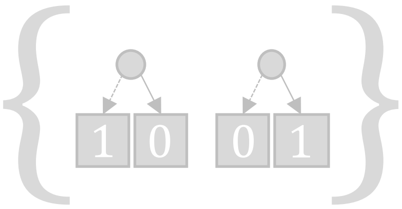
AutoQ 1.0 Automata-based Quantum Verification

TABLE I
QUANTUM GATES SUPPORTED IN THIS WORK.

Gate	Symbol	Matrix
Pauli-X (X)	$\oplus$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -\imath \\ \imath & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S)	S	$\begin{bmatrix} 1 & 0 \\ 0 & \imath \end{bmatrix}$
T	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{\imath\pi/4} \end{bmatrix}$
Rx($\frac{\pi}{2}$)	$R_x(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\imath \\ -\imath & 1 \end{bmatrix}$
Ry($\frac{\pi}{2}$)	$R_y(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$
Controlled-NOT (CNOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Toffoli		$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$

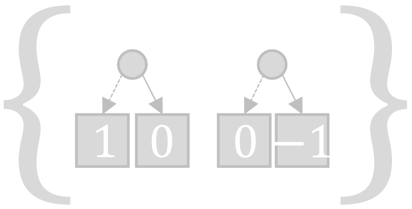
A Hoare triple would be like ...

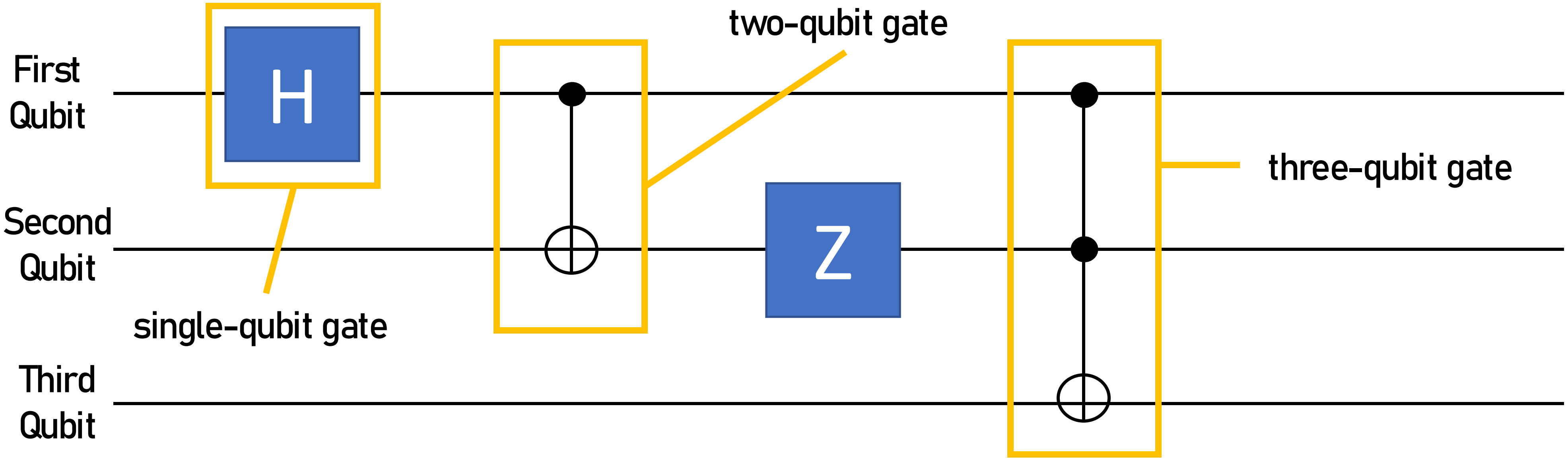
Precondition



Circuit

Postcondition

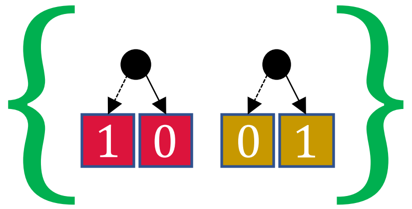




AutoQ 1.0 Automata-based Quantum Verification

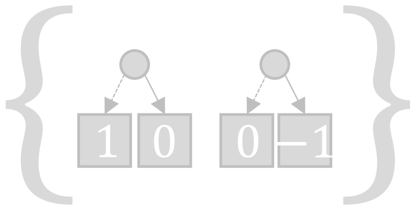
A Hoare triple would be like ...

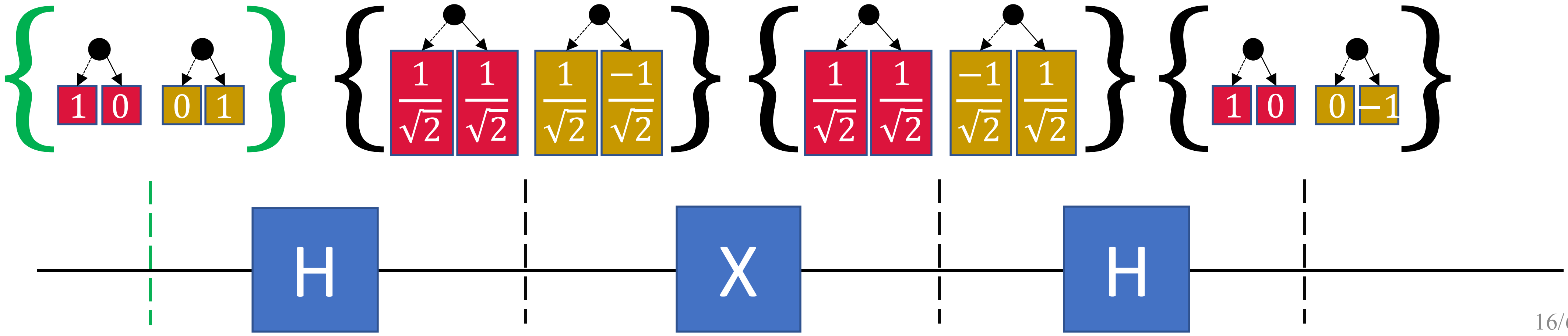
Precondition



Circuit

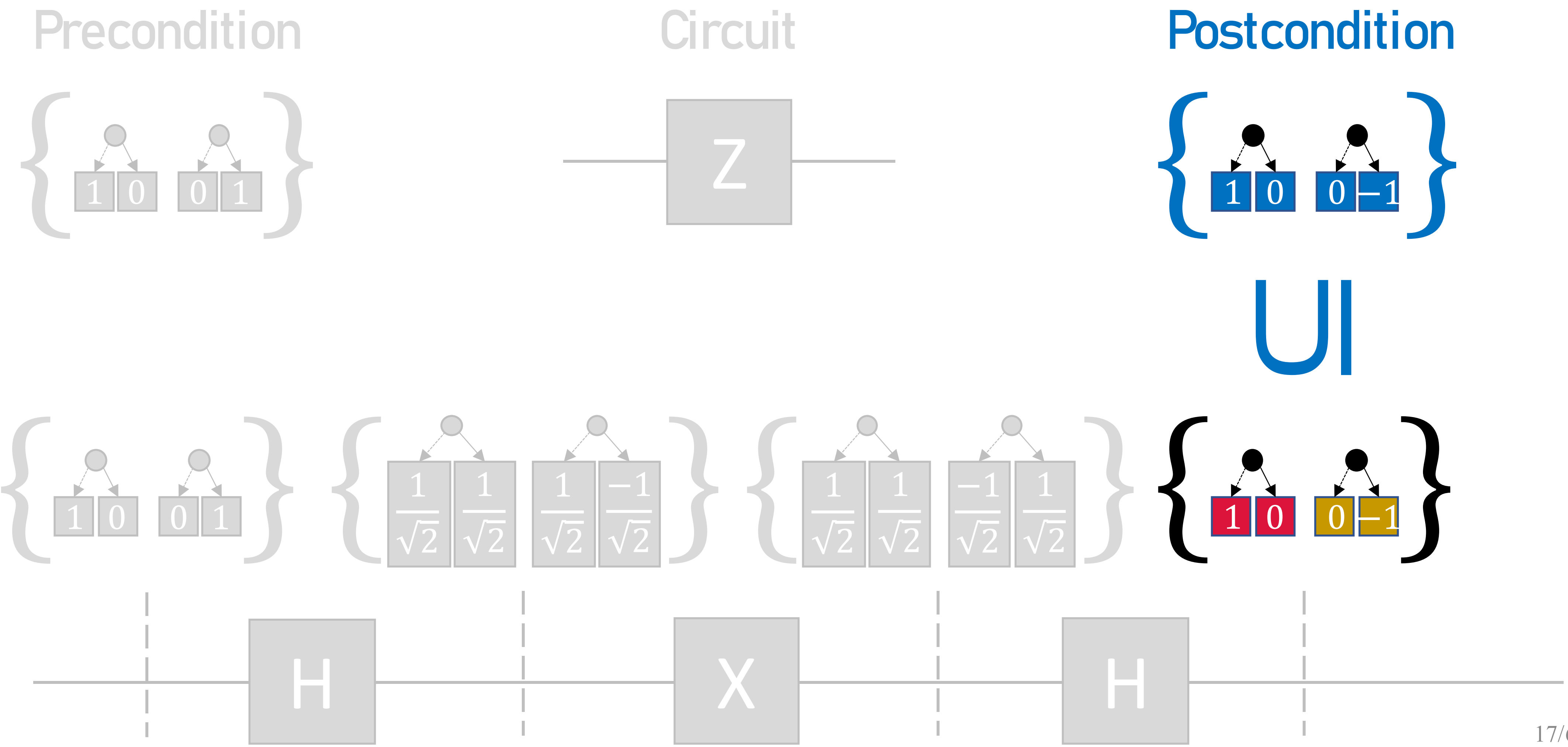
Postcondition





AutoQ 1.0 Automata-based Quantum Verification

A Hoare triple is valid if ...



Implementation

1. A set of quantum states … but

how?

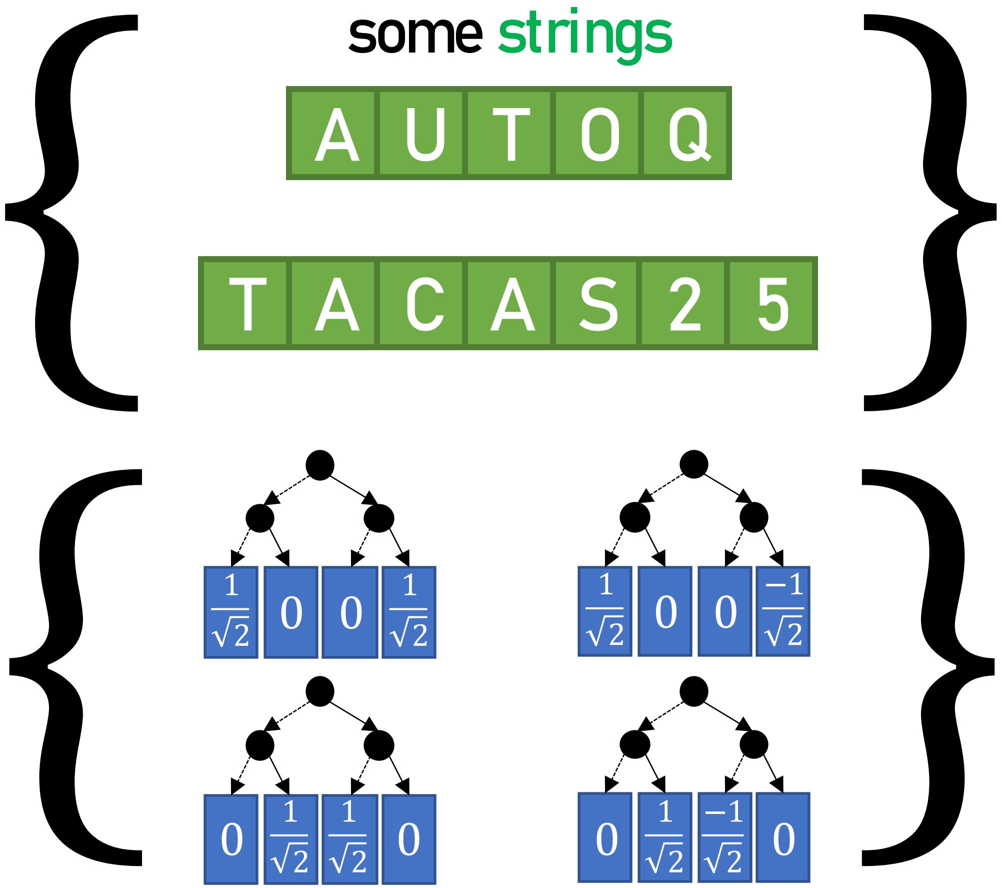
$\{$Precondition$\}$ Circuit $\{$Postcondition$\}$

A set of quantum states … but how?

{Precondition} Circuit {Postcondition}

(Word)

Automata



A set of quantum states … but how?

{Precondition} Circuit {Postcondition}

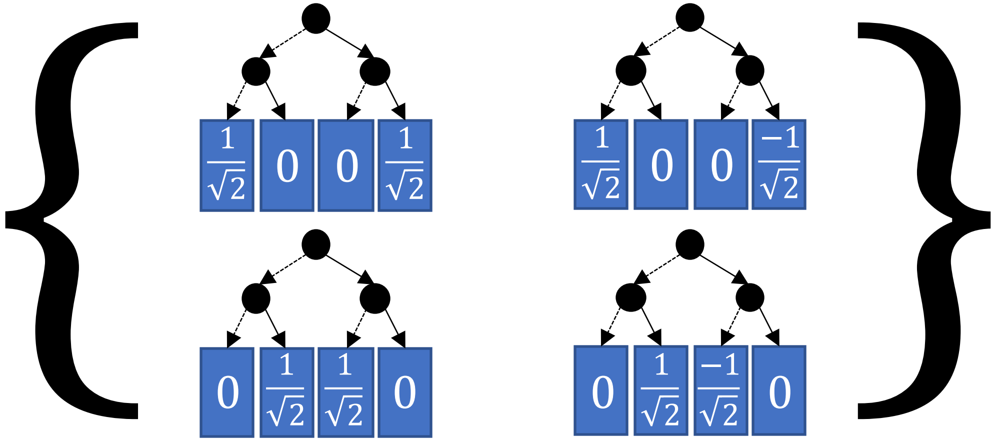
(Word)
Automata

some strings

A U T O Q

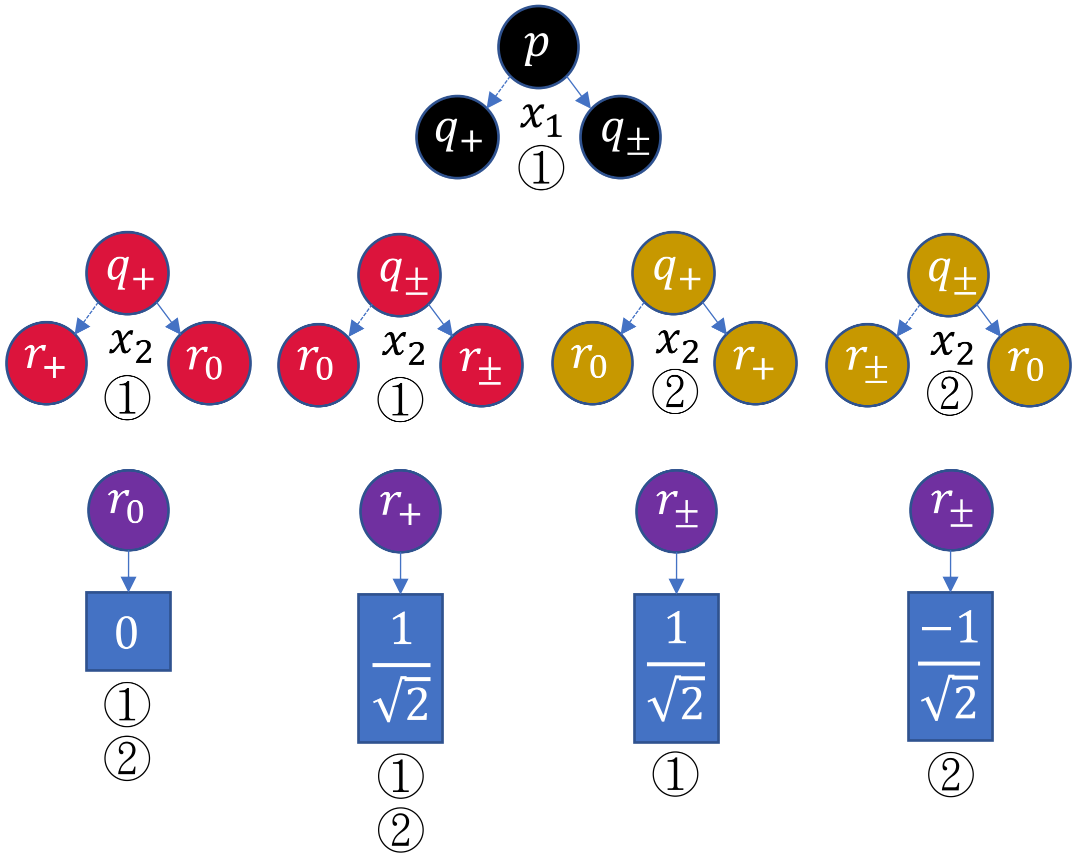
T A C A S 2 5

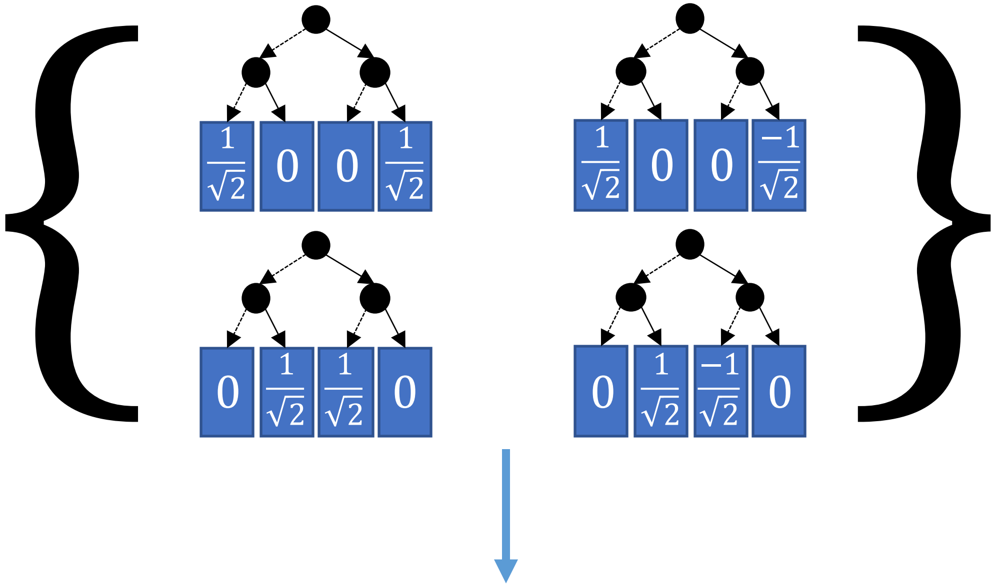
Level-Synchronized
Tree Automata
(Automata)



A set of quantum states … but how?

{Precondition} Circuit {Postcondition}



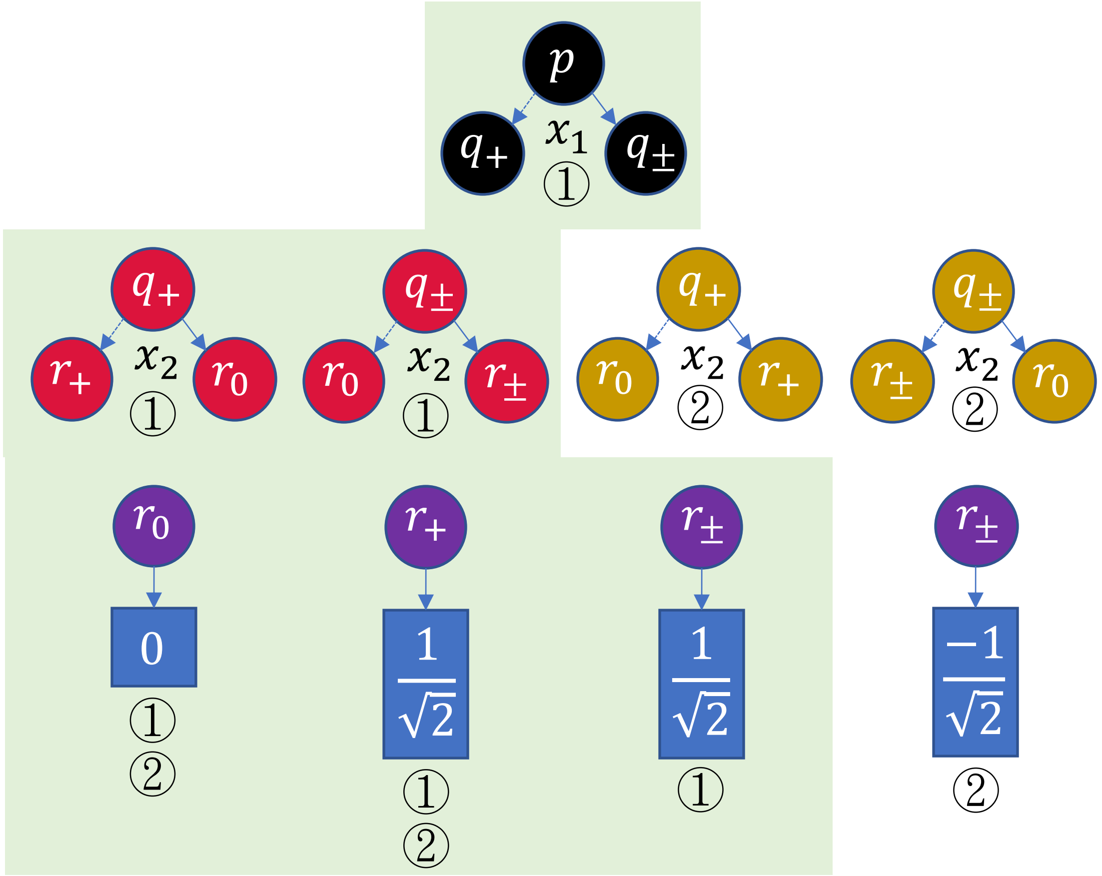


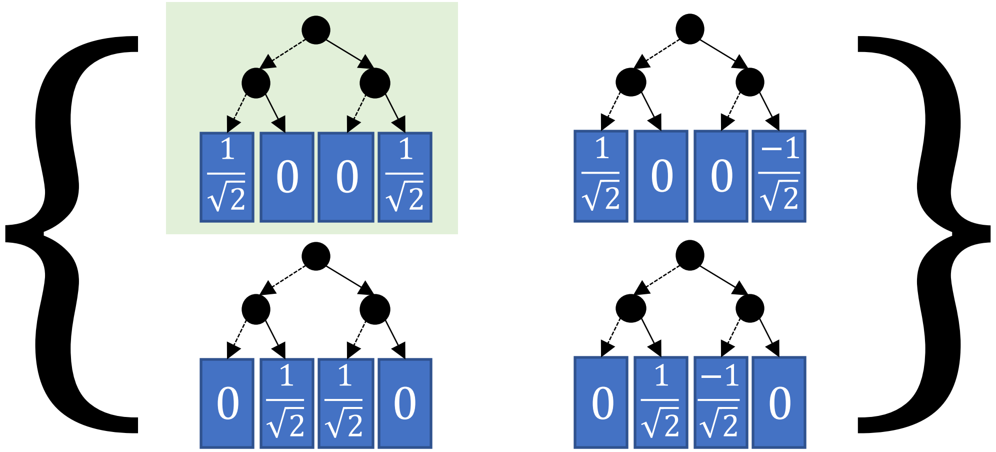
Level-Synchronized Tree Automata

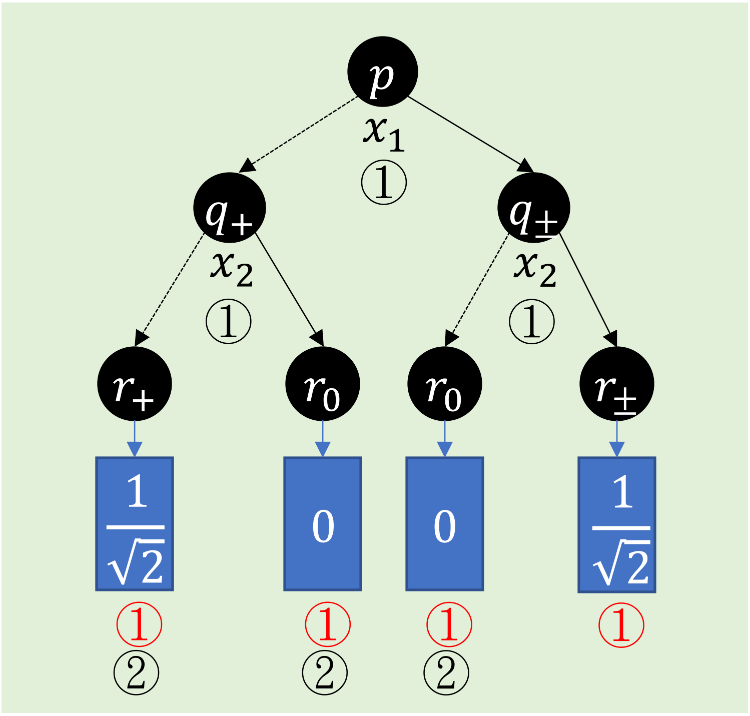
- How does such an automaton accept a tree?

A set of quantum states … but how?

{Precondition} Circuit {Postcondition}

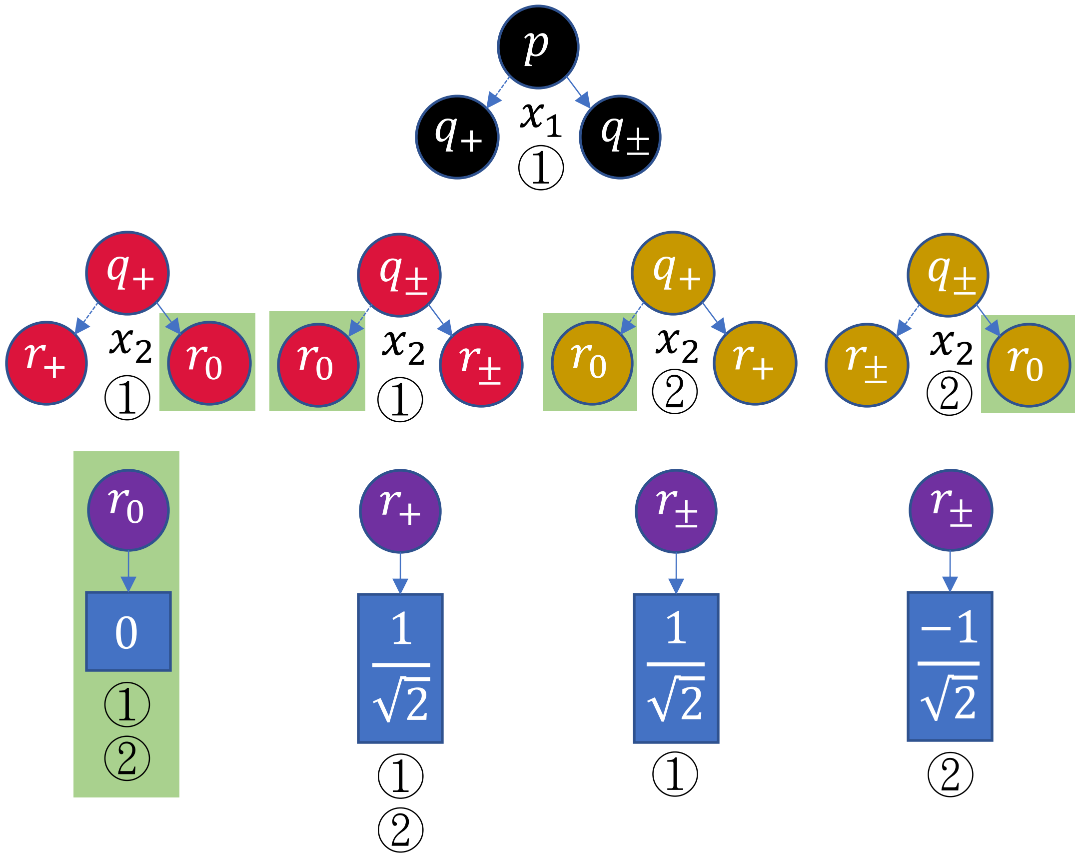


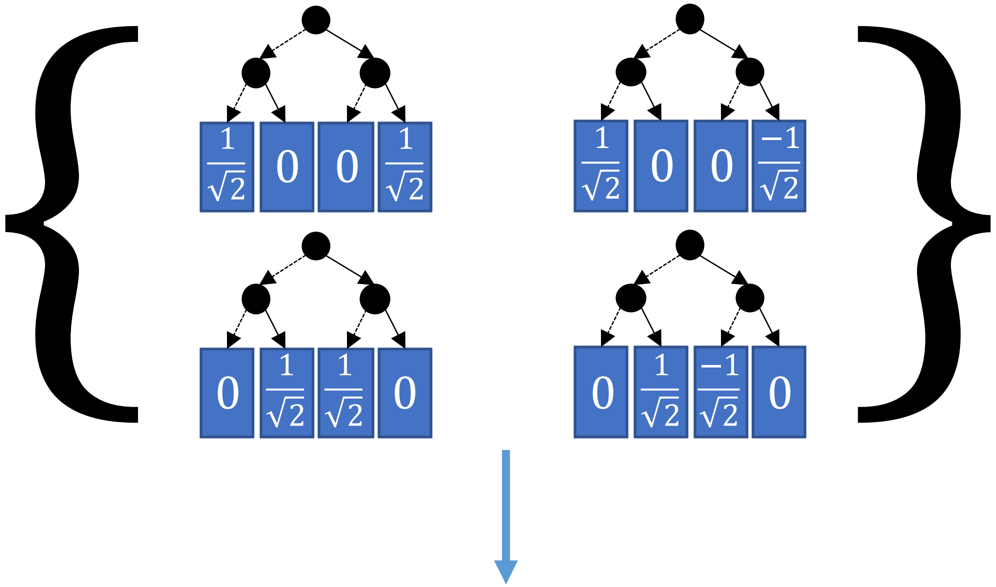




A set of quantum states … but how?

{Precondition} Circuit {Postcondition}



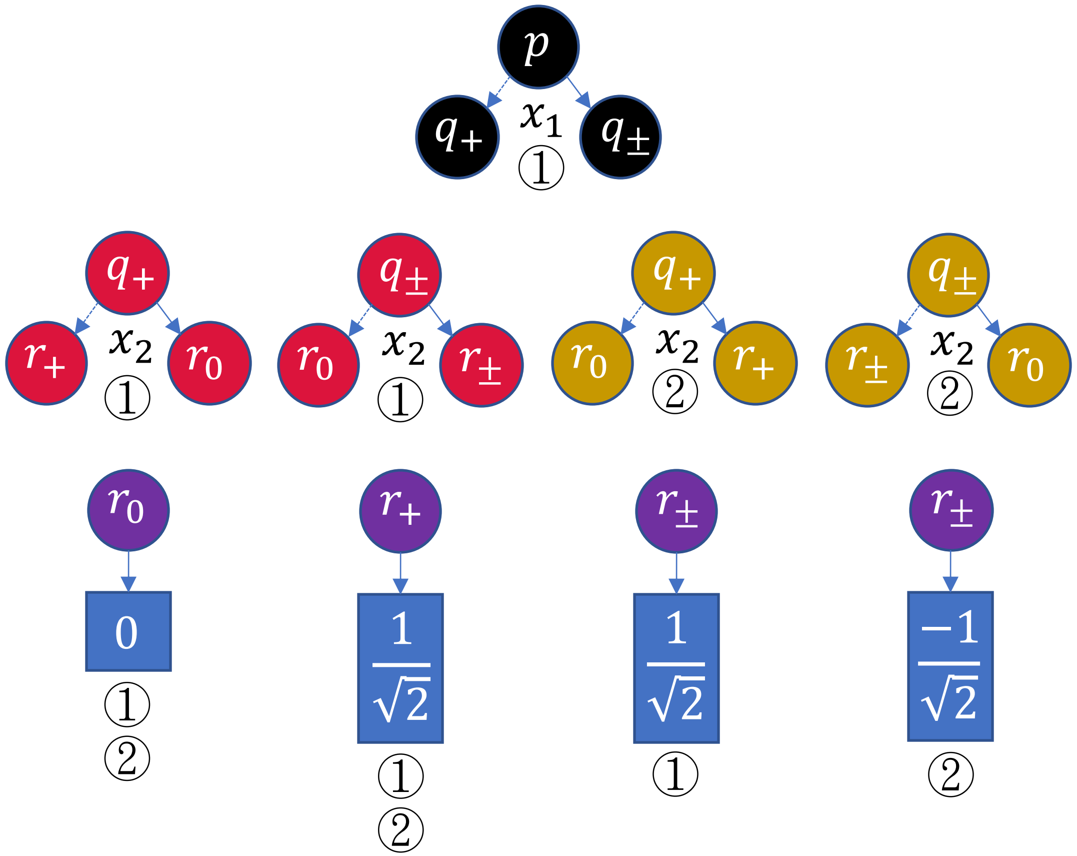


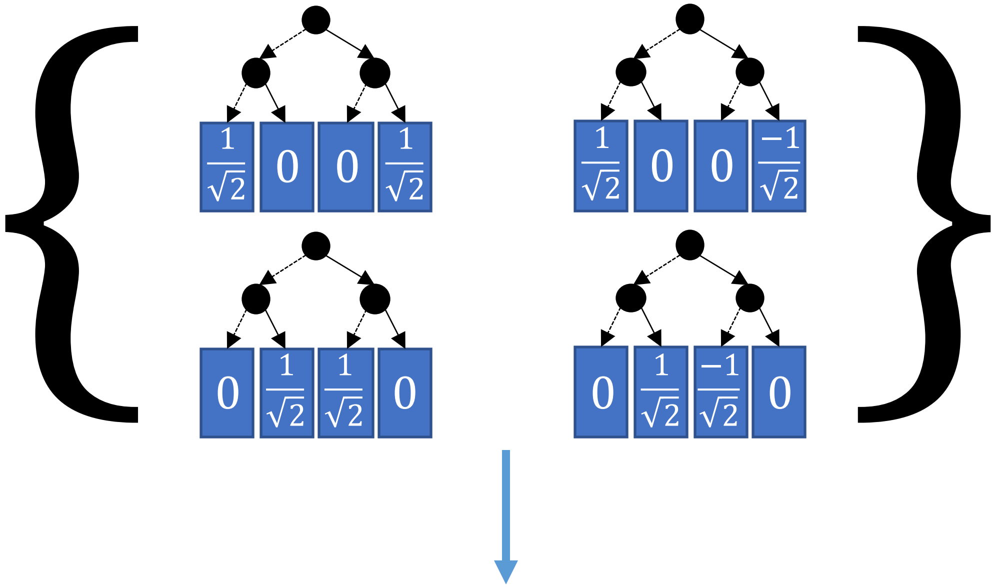
Level-Synchronized Tree Automata

Important: The same structure in many subtrees will be merged into one or more transitions for succinctness!

A set of quantum states … but how?

{Precondition} Circuit {Postcondition}





Level-Synchronized Tree Automata

RESEARCH-ARTICLE | OPEN ACCESS |

Verifying Quantum Circuits with Level-Synchronized Tree Automata

Authors: Parosh Aziz Abdulla, Yo-Ga Chen, Yu-Fang Chen, Lukáš Holík, Ondřej Lengál, Jyun-Ao Lin, Fang-Yi Lo, Wei-Lun Tsai | Authors Info & Claims

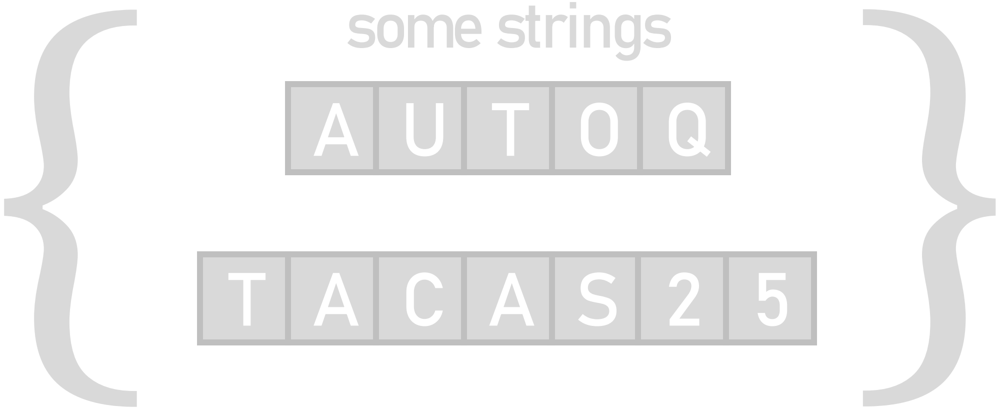
https://dl.acm.org/doi/10.1145/3704869

{Precondition} Circuit {Postcondition}

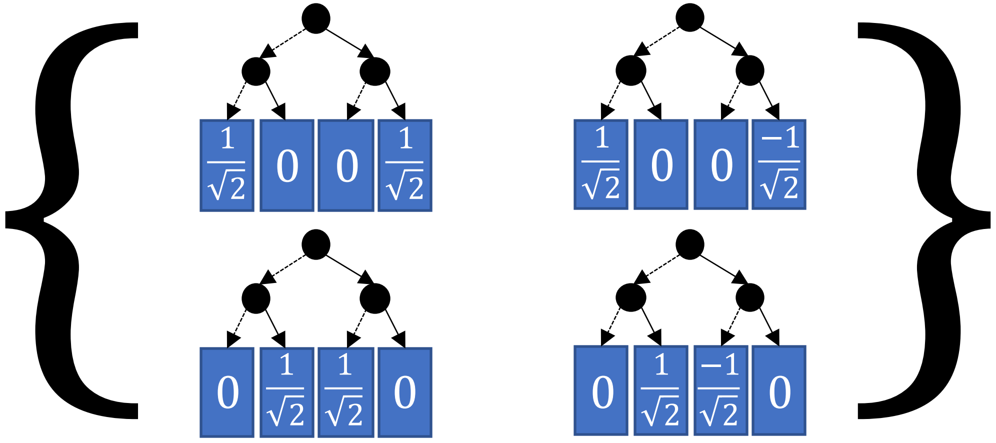
A set of quantum states … but how?

(Word)

Automata ←



Level-Synchronized Tree Automata $\mathbb{A}$ recognizes $L(\mathbb{A}) =$



2. Derive the set of quantum states after executing the circuit ...

$\{\text{Precondition}\}$ Circuit $\{\text{Postcondition}\}$

Algorithm for common quantum gates

{Precondition} **Circuit** {Postcondition}

Level-Synchronized Tree Automaton

$\mathbb{A}$

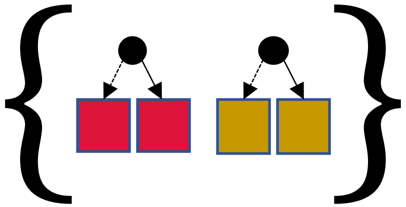
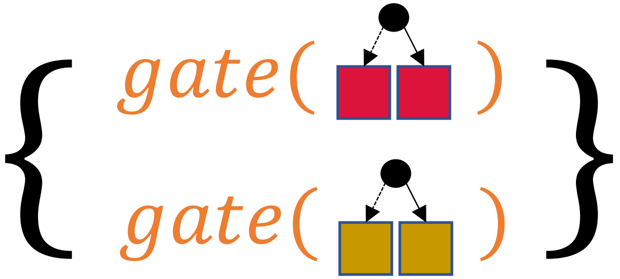


TABLE I
QUANTUM GATES SUPPORTED IN THIS WORK.

Gate	Symbol	Matrix
Pauli-X (X)	$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix}0 & -\imath\\\imath & 0\end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S)	S	$\begin{bmatrix}1 & 0\\0 & \imath\end{bmatrix}$
T	T	$\begin{bmatrix}1 & 0\\0 & e^{\imath\pi/4}\end{bmatrix}$
Rx($\frac{\pi}{2}$)	$R_x(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -\imath\\-\imath & 1\end{bmatrix}$
Ry($\frac{\pi}{2}$)	$R_y(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\1 & 1\end{bmatrix}$
Controlled-NOT (CNOT)	$\bullet$ / $\oplus$	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$
Controlled-Z (CZ)	$\bullet$ / Z	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & -1\end{bmatrix}$
Toffoli	$\bullet$ / $\bullet$ / $\oplus$	$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

Level-Synchronized Tree Automaton

$gate(\mathbb{A})$



Algorithm for common quantum gates

{Precondition} **Circuit** {Postcondition}

Level-Synchronized Tree Automaton

$\mathbb{A}$

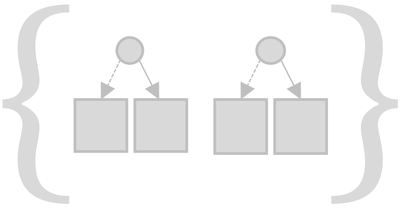
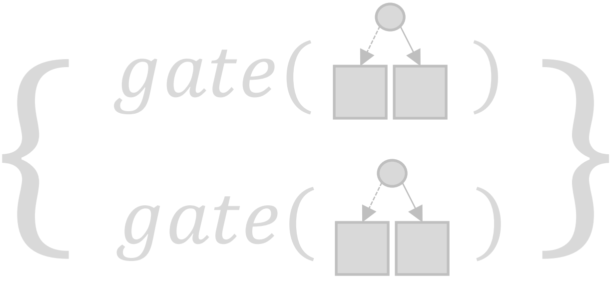


TABLE I
QUANTUM GATES SUPPORTED IN THIS WORK.

Gate	Symbol	Matrix
Pauli-X (X)	$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix}0 & -\imath\\\imath & 0\end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S)	S	$\begin{bmatrix}1 & 0\\0 & \imath\end{bmatrix}$
T	T	$\begin{bmatrix}1 & 0\\0 & e^{\imath\pi/4}\end{bmatrix}$
Rx($\frac{\pi}{2}$)	$R_x(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -\imath\\-\imath & 1\end{bmatrix}$
Ry($\frac{\pi}{2}$)	$R_y(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\1 & 1\end{bmatrix}$
Controlled-NOT (CNOT)	$\bullet$ / $\oplus$	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled-Z (CZ)	$\bullet$ / Z	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
Toffoli	$\bullet$ / $\bullet$ / $\oplus$	$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

Level-Synchronized Tree Automaton

$gate(\mathbb{A})$



Algorithm for common quantum gates

{Precondition} **Circuit** {Postcondition}

Level-Synchronized Tree Automaton

$\mathbb{A}$

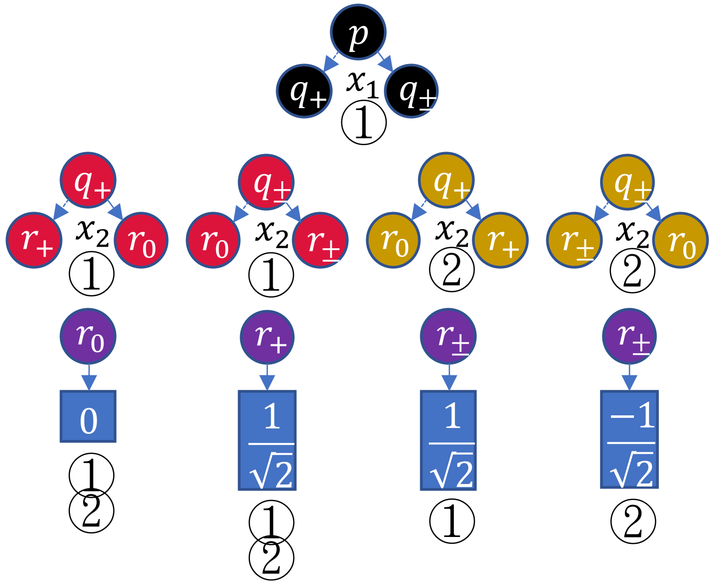
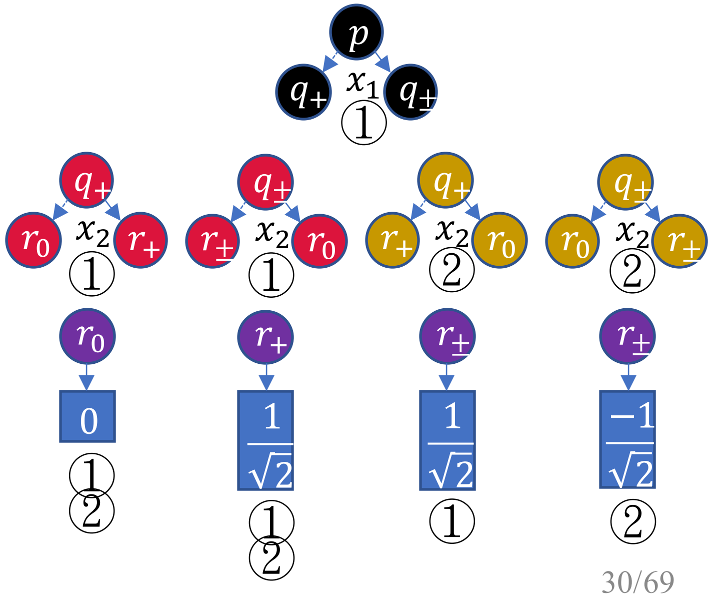


TABLE I
QUANTUM GATES SUPPORTED IN THIS WORK.

Gate	Symbol	Matrix
Pauli-X (X)	$\oplus$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -\imath \\ \imath & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S)	S	$\begin{bmatrix} 1 & 0 \\ 0 & \imath \end{bmatrix}$
T	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{\imath\pi/4} \end{bmatrix}$
Rx($\frac{\pi}{2}$)	$R_x(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\imath \\ -\imath & 1 \end{bmatrix}$
Ry($\frac{\pi}{2}$)	$R_y(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$
Controlled-NOT (CNOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Z (CZ)	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

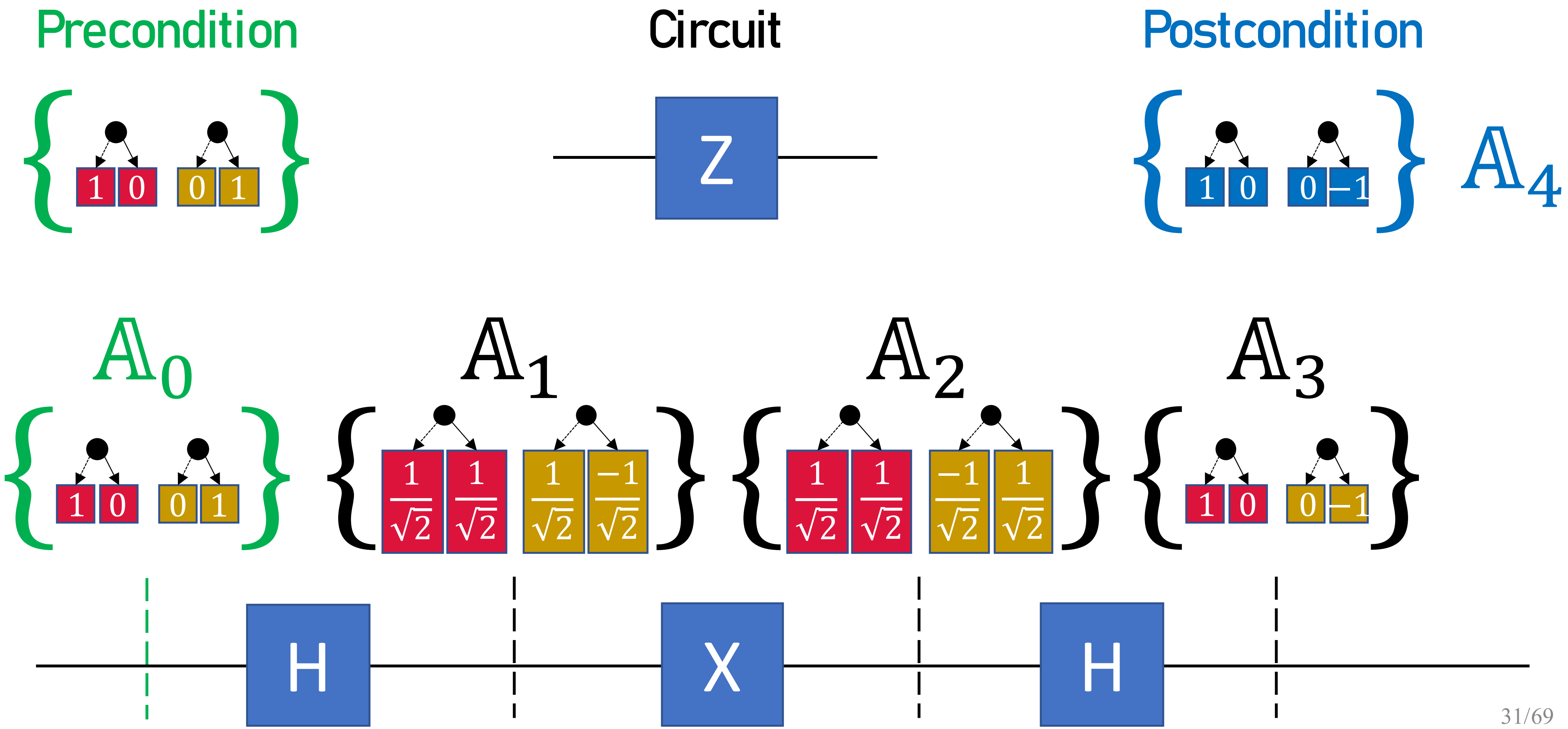
Level-Synchronized Tree Automaton

$X_2(\mathbb{A})$



AutoQ 1.0 Automata-based Quantum Verification

A Hoare triple would be like ...



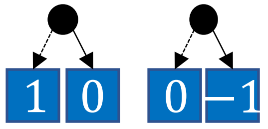
3. Verify a Hoare triple ...

{Precondition} Circuit {Postcondition}

To verify a Hoare triple …

{Precondition} Circuit {Postcondition}

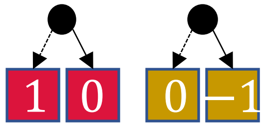
Postcondition

$$L(\mathbb{A}_{post}) = L(\mathbb{A}_4) = \Big\{$$ 

$$\Big\}$$

language inclusion checking of level-synchronized tree automata ?

$\cup\!\vert$

$$L(Circuit(\mathbb{A}_{pre})) = L(\mathbb{A}_3) = \Big\{$$ 

$$\Big\}$$

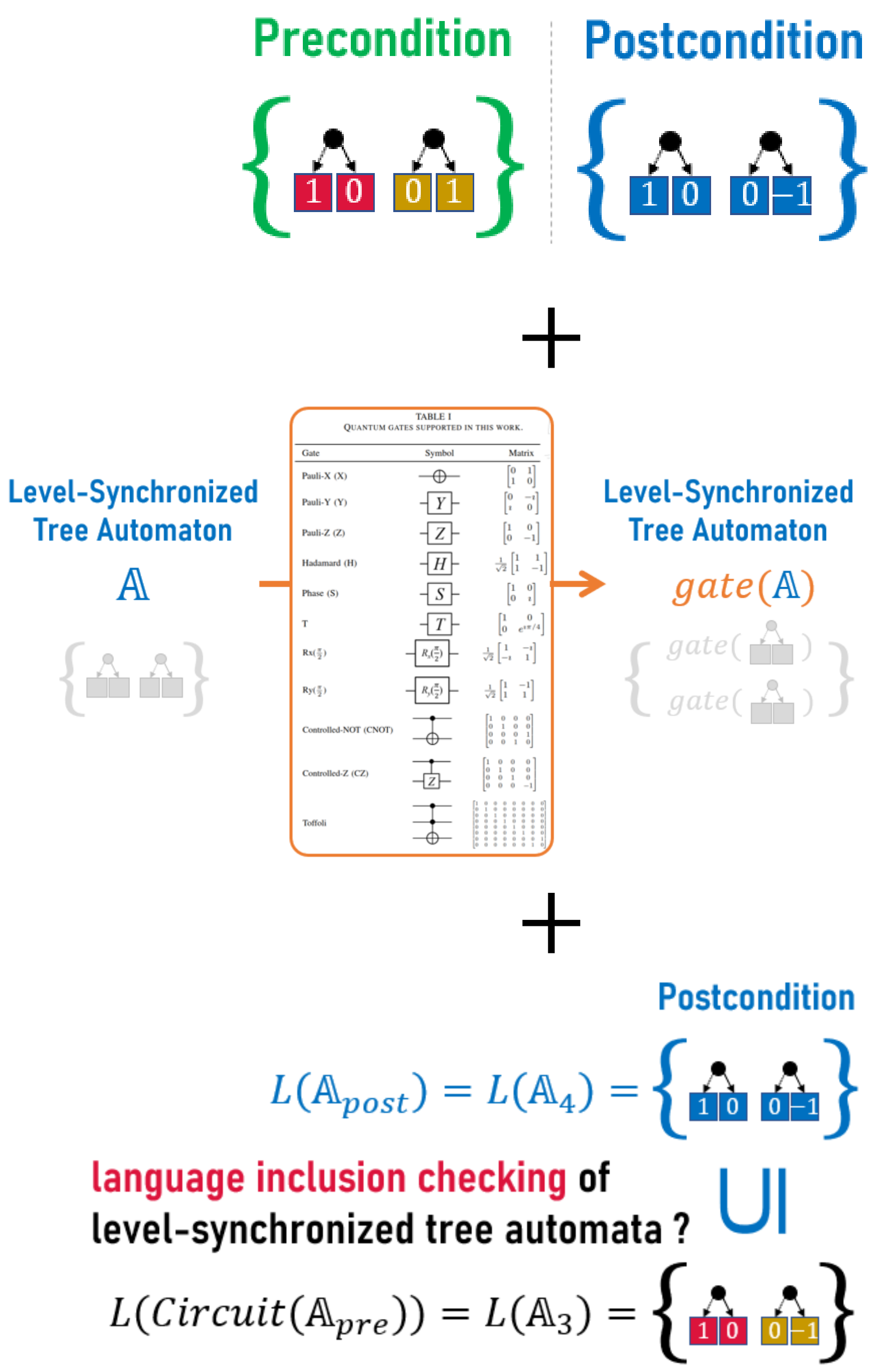


TABLE I
QUANTUM GATES SUPPORTED IN THIS WORK.

Gate	Symbol	Matrix
Pauli-X (X)	$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S)	S	$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
T	T	$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Rx($\frac{\pi}{2}$)	$R_x(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -i\\-i & 1\end{bmatrix}$
Ry($\frac{\pi}{2}$)	$R_y(\frac{\pi}{2})$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\1 & 1\end{bmatrix}$
Controlled-NOT (CNOT)	CNOT	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled-Z (CZ)	CZ	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
Toffoli	Toffoli	$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

AutoQ is fully automated!

AutoQ 2.0:
From Quantum Circuits to Quantum Programs

Quantum Circuits (AutoQ 1.0)

1. quantum gates

➡

Quantum Programs (AutoQ 2.0)

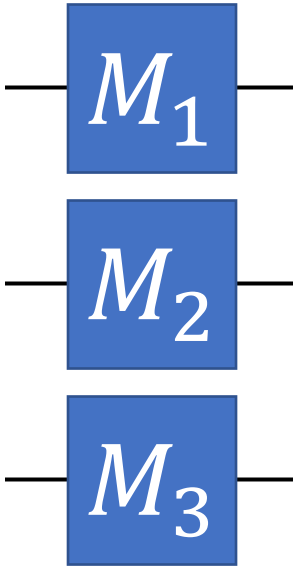
1. quantum gates
2. branches
3. loops

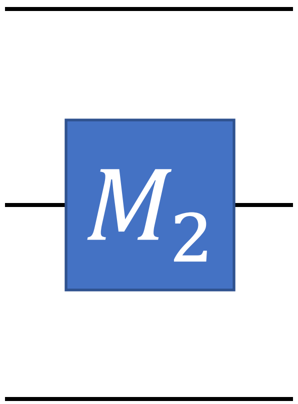
Why measurement?

- if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$
- while $(M_q = b)$ do $\{P\}$

Measurement

- Case 1 – Measure all qubits together.
- Case 2 – Measure only one qubit.
  - if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$
  - while $(M_q = b)$ do $\{P\}$

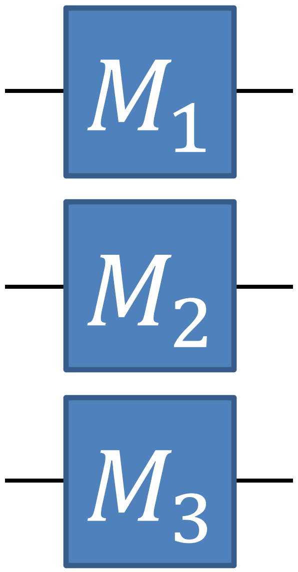




Measurement – All Qubits Together

(c_i: amplitude) (p_i: probability)

0	0	0	$\lvert c_0 \rvert^2 = 5\%$
0	0	1	$\lvert c_1 \rvert^2 = 7\%$
0	1	0	$\lvert c_2 \rvert^2 = 1\%$
0	1	1	$\lvert c_3 \rvert^2 = 4\%$
1	0	0	$\lvert c_4 \rvert^2 = 6\%$
1	0	1	$\lvert c_5 \rvert^2 = 2\%$
1	1	0	$\lvert c_6 \rvert^2 = 5\%$
1	1	1	$\lvert c_7 \rvert^2 = 70\%$

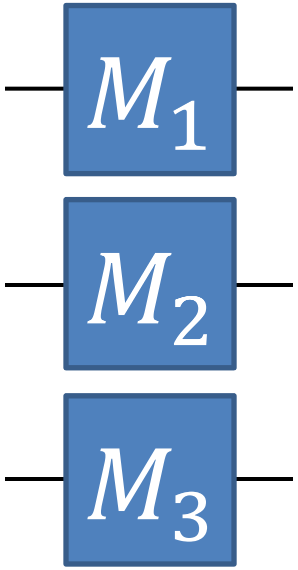


- All possible outcomes $= \{|000\rangle, |001\rangle, \ldots, |110\rangle, |111\rangle\}$
- P(outcome $= |i\rangle) = |c_i|^2$
- The resulting state $= |i\rangle$.

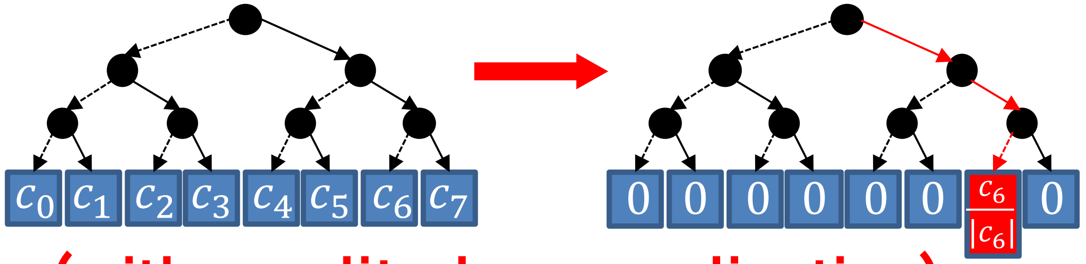
Measurement – All Qubits Together

(c_i: amplitude) (p_i: probability)

0	0	0	$\lvert c_0\rvert^2 = 5\%$
0	0	1	$\lvert c_1\rvert^2 = 7\%$
0	1	0	$\lvert c_2\rvert^2 = 1\%$
0	1	1	$\lvert c_3\rvert^2 = 4\%$
1	0	0	$\lvert c_4\rvert^2 = 6\%$
1	0	1	$\lvert c_5\rvert^2 = 2\%$
1	1	0	$\lvert c_6\rvert^2 = 5\%$
1	1	1	$\lvert c_7\rvert^2 = 70\%$



- All possible outcomes = $\{|000\rangle, |001\rangle, \ldots, |110\rangle, |111\rangle\}$
- P(outcome = $|i\rangle$) = $|c_i|^2$
- The resulting state = $|i\rangle$.



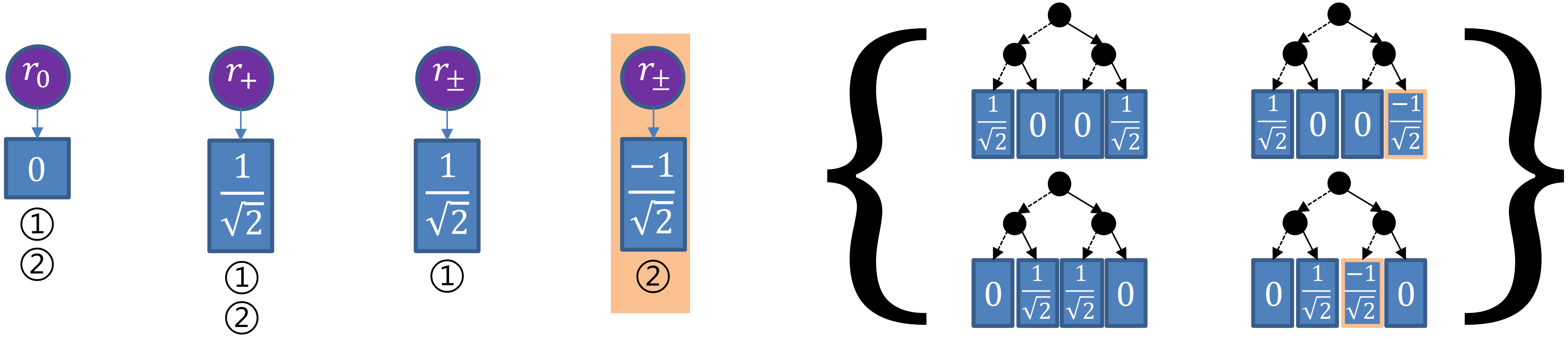
(with amplitude normalization)

Normalize the amplitudes such that the summation of all possible outcomes' probabilities is still 1.

Contribution 1 – No Amplitude Normalization

No amplitude normalization in the implementation because:

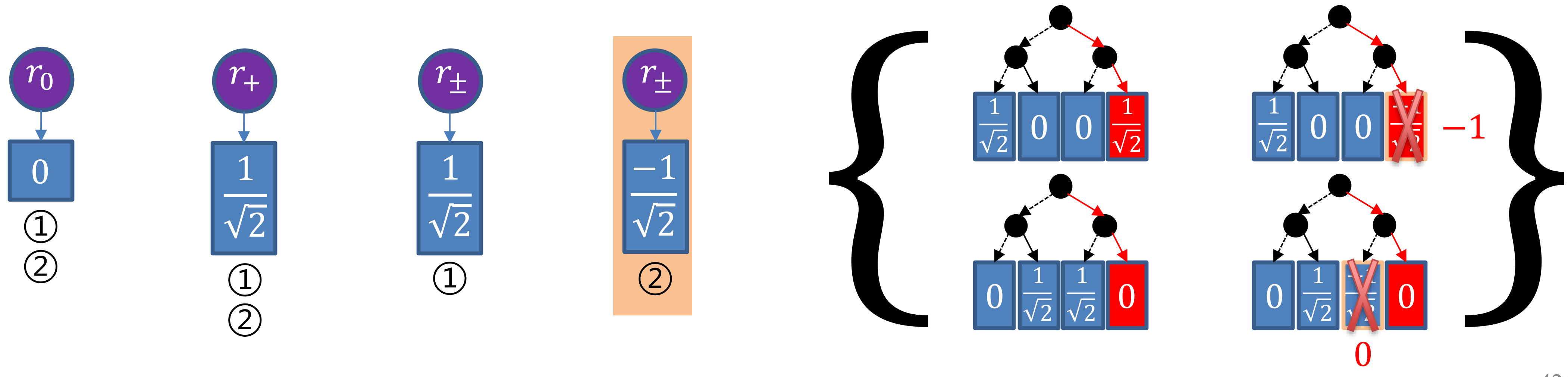
1. Level-synchronized tree automata merge the same amplitude in different trees into one or more amplitude transitions. After a quantum measurement, one amplitude in different trees may have different scaling factors. We don't even know what trees a particular transition belongs to, so it is infeasible to identify all scaling factors of an amplitude transition.



Contribution 1 – No Amplitude Normalization

No amplitude normalization in the implementation because:

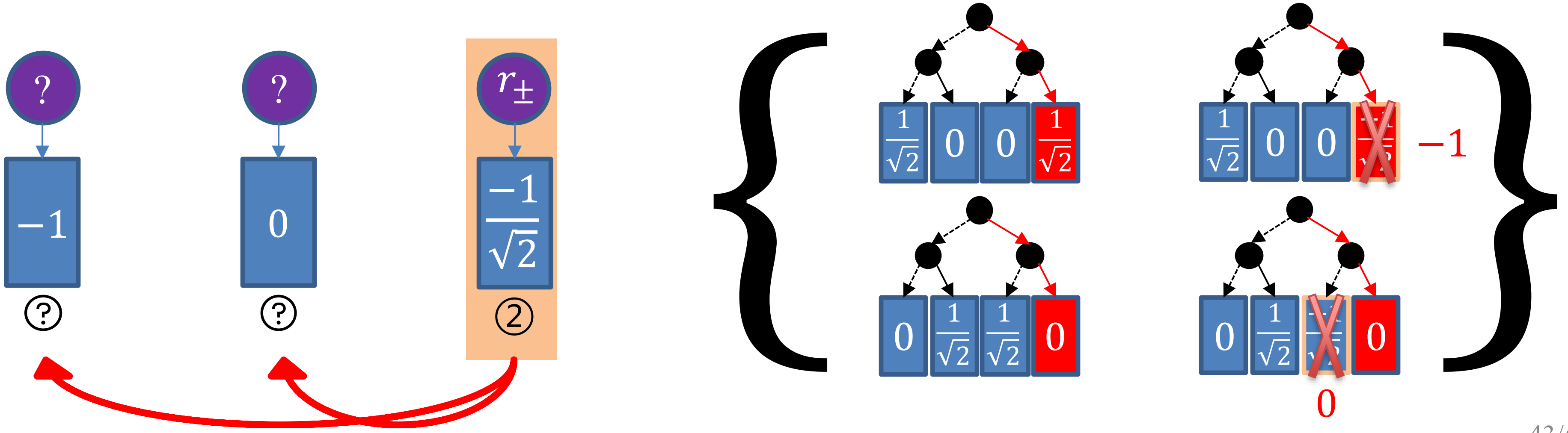
1. Level-synchronized tree automata merge the same amplitude in different trees into one or more amplitude transitions. After a quantum measurement, one amplitude in different trees may have different scaling factors. We don't even know what trees a particular transition belongs to, so <u>it is infeasible to identify all scaling factors of an amplitude transition.</u>



Contribution 1 – No Amplitude Normalization

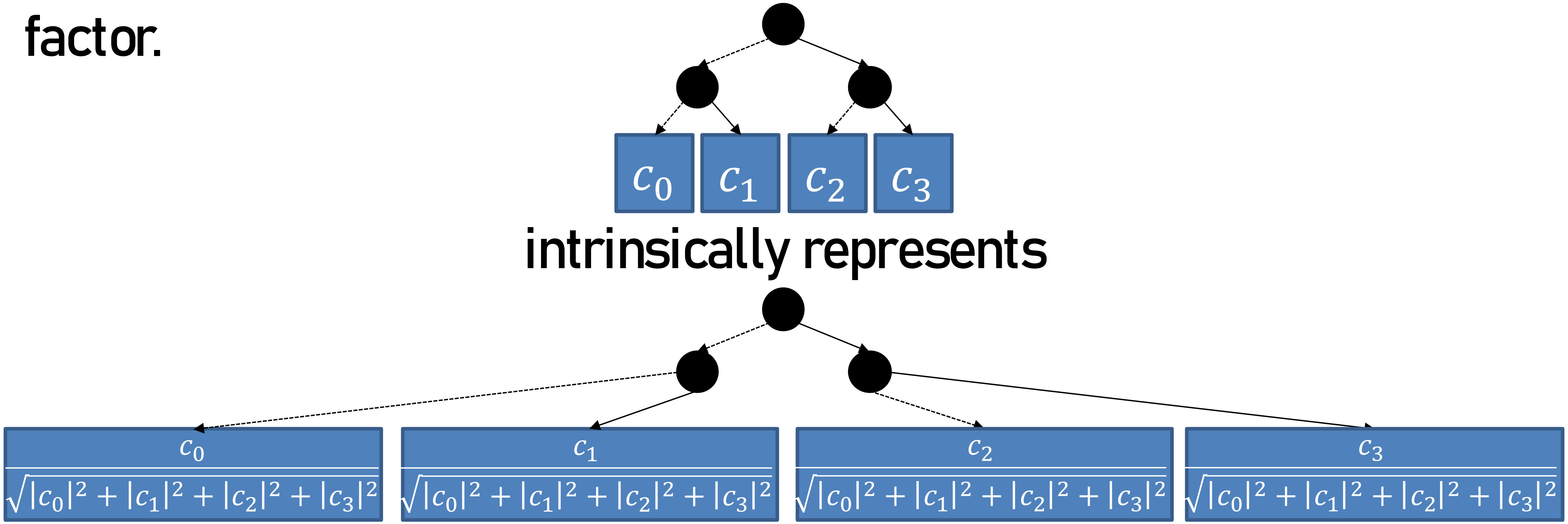
No amplitude normalization in the implementation because:

2. Even if this mechanism is feasible to implement, the resulting automaton may lack the succinctness because one amplitude transition may be transformed into many pieces with different scaling factors. Different amplitudes cannot be merged, so <u>the size of automata may explode, drastically degrading the performance.</u>



Contribution 1 – No Amplitude Normalization

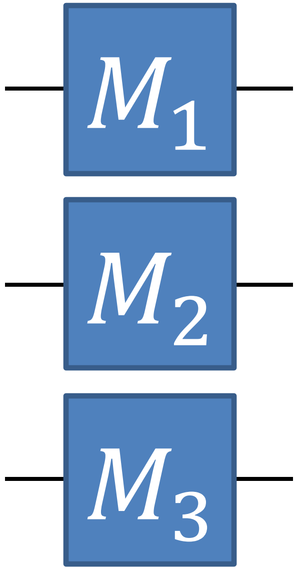
No amplitude normalization in the implementation still works because each non-scaled tree intrinsically represents a normalized valid tree (the sum of probabilities is 1) with the unique positive real factor.



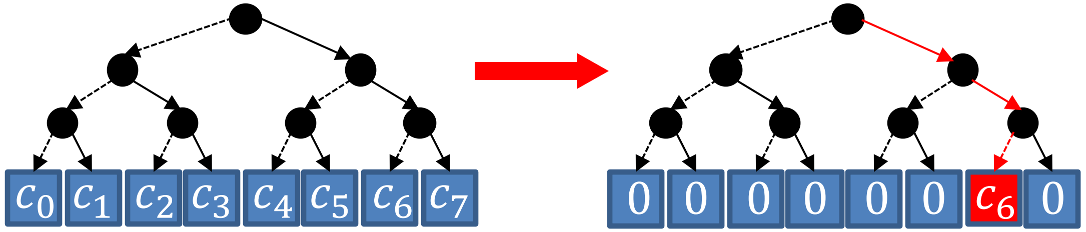
Measurement – All Qubits Together

(c_i: amplitude) (p_i: probability)

0 0 0	$\|c_0\|^2 = 5\%$
0 0 1	$\|c_1\|^2 = 7\%$
0 1 0	$\|c_2\|^2 = 1\%$
0 1 1	$\|c_3\|^2 = 4\%$
1 0 0	$\|c_4\|^2 = 6\%$
1 0 1	$\|c_5\|^2 = 2\%$
1 1 0	$\|c_6\|^2 = 5\%$
1 1 1	$\|c_7\|^2 = 70\%$



- All possible outcomes = $\{|000\rangle, |001\rangle, \ldots, |110\rangle, |111\rangle\}$
- P(outcome = $|i\rangle$) = $|c_i|^2$
- The resulting state = $|i\rangle$.

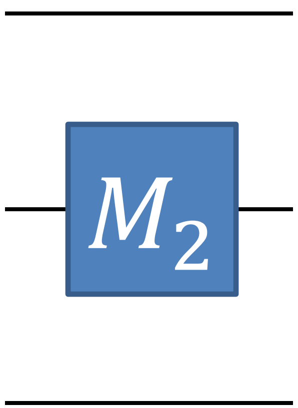


(no amplitude normalization)

Measurement – Only One Qubit

(c_i: amplitude) (p_i: probability)

0	0	0	$|c_0|^2 = 5\%$
0	0	1	$|c_1|^2 = 7\%$
0	1	0	$|c_2|^2 = 1\%$
0	1	1	$|c_3|^2 = 4\%$
1	0	0	$|c_4|^2 = 6\%$
1	0	1	$|c_5|^2 = 2\%$
1	1	0	$|c_6|^2 = 5\%$
1	1	1	$|c_7|^2 = 70\%$

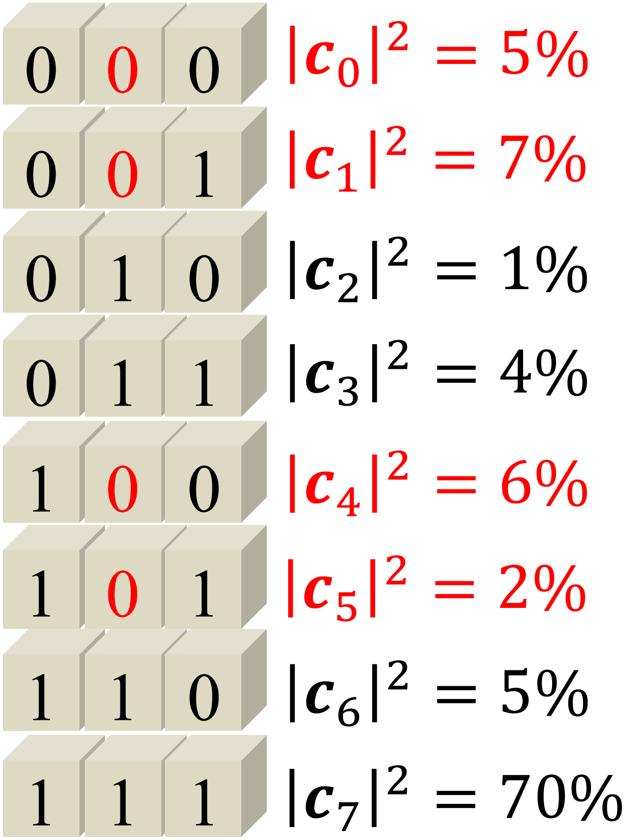


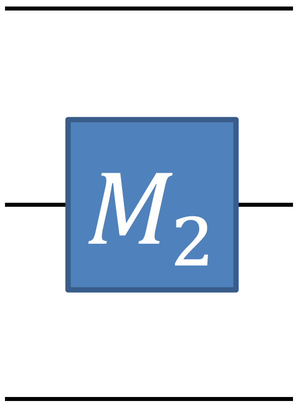
- All possible outcomes $= \{|0\rangle_2, |1\rangle_2\}$
- P(outcome $= |i\rangle_2$) $= \sum_{x\in\{0,1\}, y\in\{0,1\}}$ P(outcome $= |xiy\rangle$)
- The resulting state $= \dfrac{\sum_{x\in\{0,1\}, y\in\{0,1\}} c_{xiy}|xiy\rangle}{\sqrt{\text{P(outcome} = |i\rangle_2)}}$.

(normalization)

Measurement – Only One Qubit

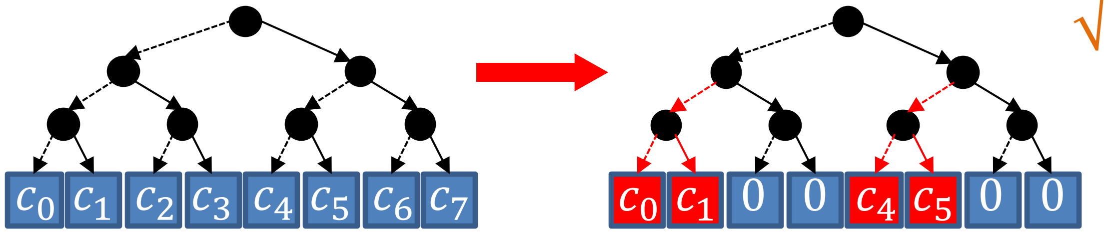
(c_i: amplitude) (p_i: probability)





- All possible outcomes $= \{|0\rangle_2, |1\rangle_2\}$
- P(outcome $= |i\rangle_2$) $= \sum_{x\in\{0,1\},\, y\in\{0,1\}}$ P(outcome $= |xiy\rangle$)
- The resulting state $= \dfrac{\sum_{x\in\{0,1\},\, y\in\{0,1\}} c_{xiy}|xiy\rangle}{\sqrt{\text{P(outcome} = |i\rangle_2)}}$.

(normalization)



(no amplitude normalization)

Execution Path Decided by Measurement

- if ($M_q = b$) then $\{P_1\}$ else $\{P_2\}$
- while ($M_q = b$) do $\{P\}$

if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

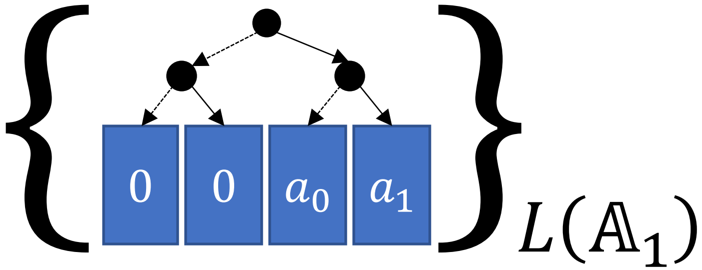
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

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

Post: $\{a_0 |10\rangle + a_1 |11\rangle,$

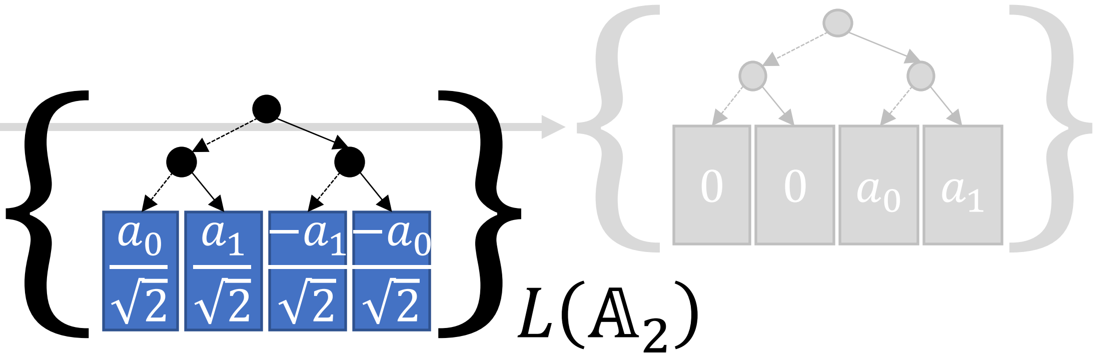
$\quad -a_0 |11\rangle - a_1 |10\rangle\}$;



if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

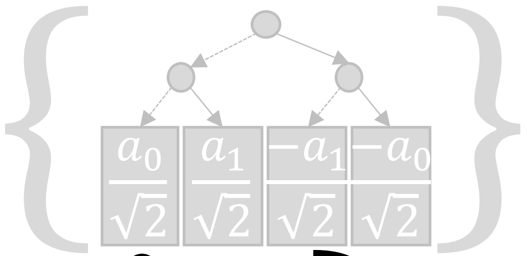
1 Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
2 $H_1; CX_2^1$;
3 **if** $M_1 = 0$ **then** $\{X_1\}$;
4 Post: $\{a_0 |10\rangle + a_1 |11\rangle$,
5 $-a_0 |11\rangle - a_1 |10\rangle\}$;



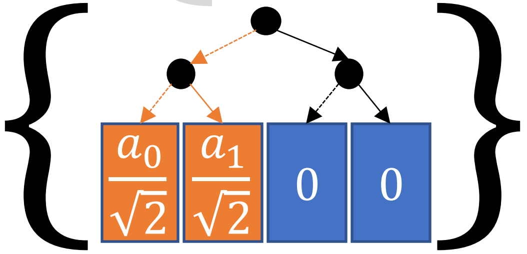
if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

1 Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
2 $H_1 ; CX_2^1$;
3 **if** $M_1 = 0$ **then** $\{X_1\}$;
4 Post: $\{a_0 |10\rangle + a_1 |11\rangle$,
5 $-a_0 |11\rangle - a_1 |10\rangle\}$;



if $(M_1 = 0)$
then {
X_1
}
else {
}



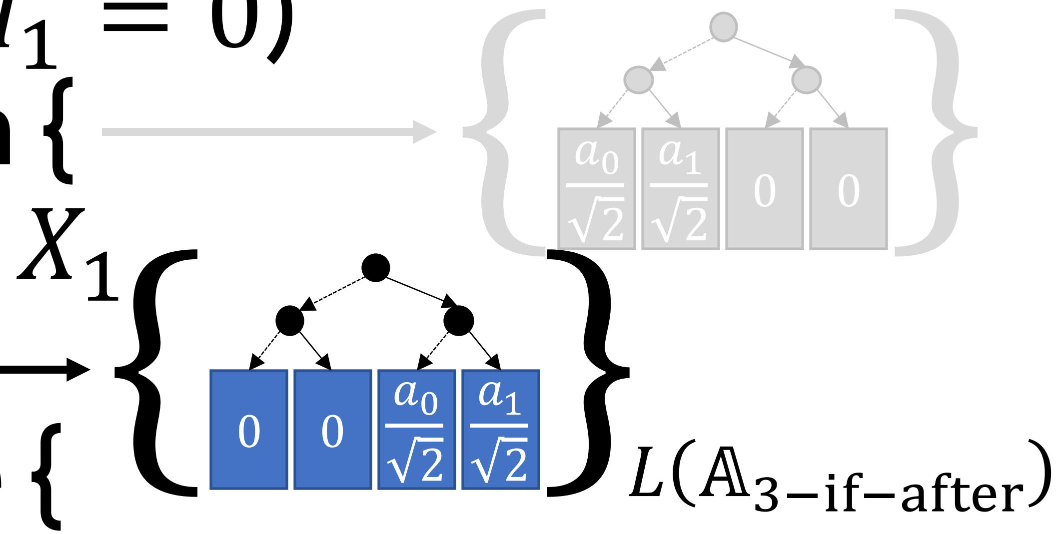
$L(\mathbb{A}_{3-\text{if}-\text{before}})$

if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

Pre: $\{a_0\,|10\rangle + a_1\,|11\rangle\}$;
$H_1; CX_2^1$;
if $M_1 = 0$ **then** $\{X_1\}$;
Post: $\{a_0\,|10\rangle + a_1\,|11\rangle\,,$
$-a_0\,|11\rangle - a_1\,|10\rangle\}$;

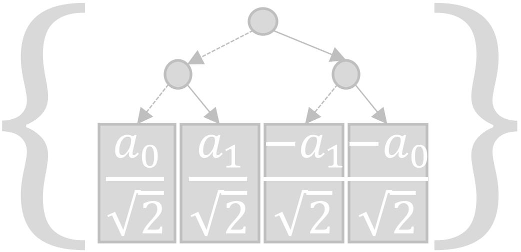
if $(M_1 = 0)$
then {
}
else {
}



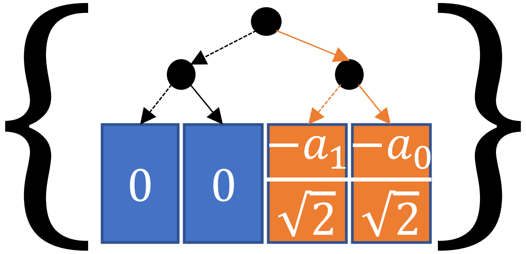
if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
$H_1; CX_2^1$;
if $M_1 = 0$ **then** $\{X_1\}$;
Post: $\{a_0 |10\rangle + a_1 |11\rangle$,
$-a_0 |11\rangle - a_1 |10\rangle\}$;



if $(M_1 = 0)$
then {
X_1
}
else {
}



$L(\mathbb{A}_{3-\text{else}})$

if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

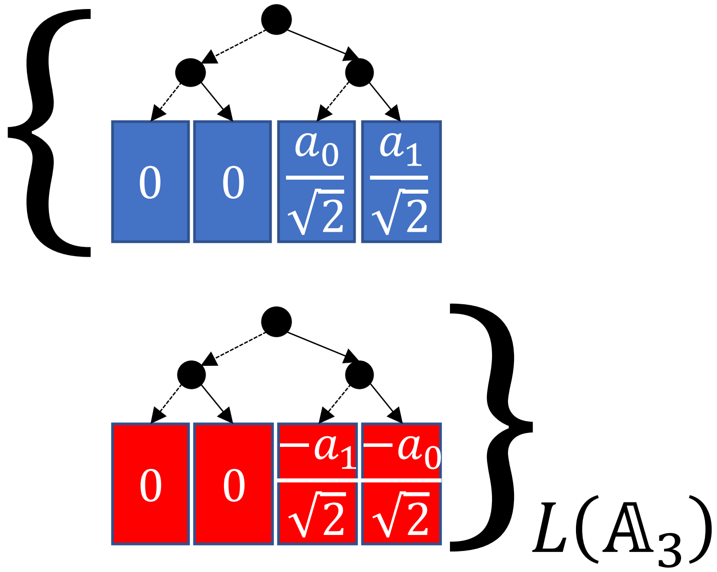
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

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

Post: $\{a_0 |10\rangle + a_1 |11\rangle$,

$-a_0 |11\rangle - a_1 |10\rangle\}$;

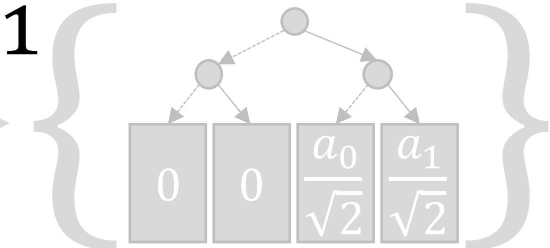


if $(M_1 = 0)$

then {

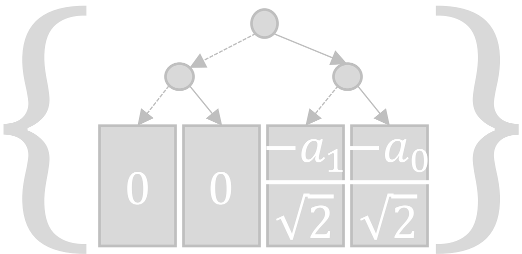
X_1

}



else {

}



if $(M_q = b)$ then $\{P_1\}$ else $\{P_2\}$

Algorithm 1: "$-X_2$" if $M_1 = 1$

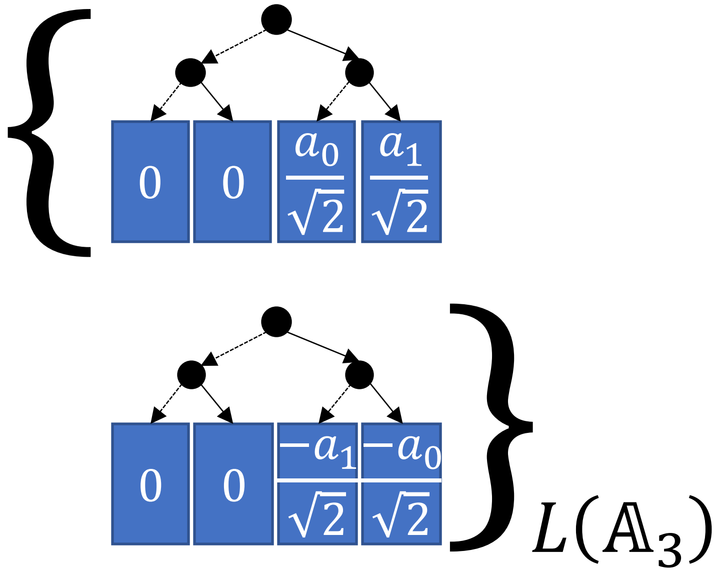
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

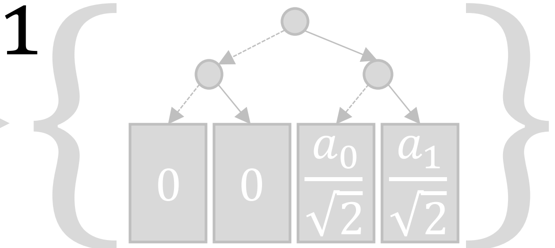
if $M_1 = 0$ **then** $\{X_1\}$;

Post: $\{a_0 |10\rangle + a_1 |11\rangle,$

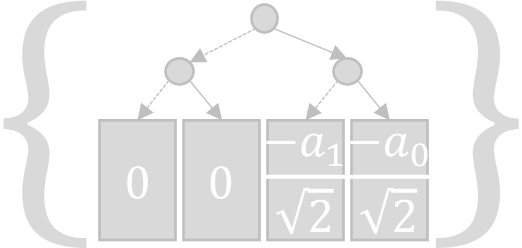
$-a_0 |11\rangle - a_1 |10\rangle\}$;



if $(M_1 = 0)$
then {
X_1
}

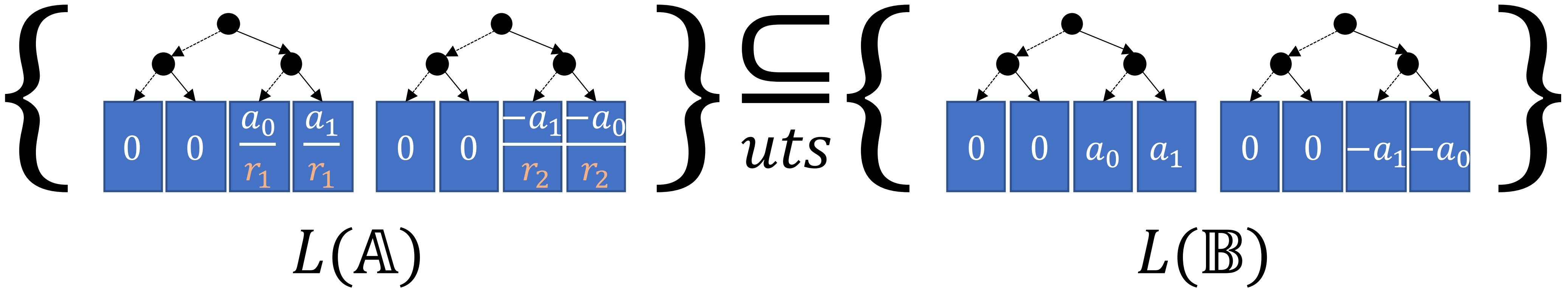


else {
}



Contribution 2 - Up-to-scaling Inclusion Checking

1. Definition: $L(\mathbb{A}) \subseteq L(\mathbb{B})$ if for each tree $q_{\mathbb{A}}$ in $L(\mathbb{A})$, there is another tree $q_{\mathbb{B}}$ in $L(\mathbb{B})$ such that $q_{\mathbb{A}} = r \cdot q_{\mathbb{B}}$ for some real number $r > 0$.



2. See the paper for the implementation detail.

https://arxiv.org/abs/2411.09121

while $(M_q = b)$ do $\{P\}$

Algorithm 2: "$-X_2$"

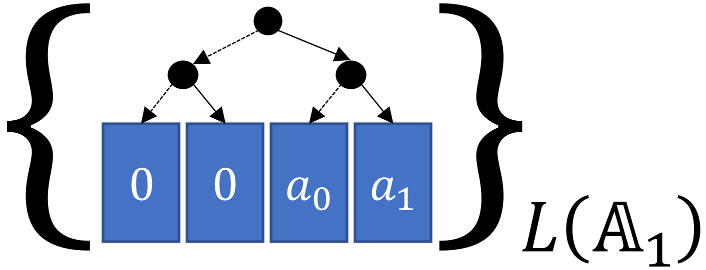
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

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

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

Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;



Algorithm 1: "$-X_2$" if $M_1 = 1$

Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

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

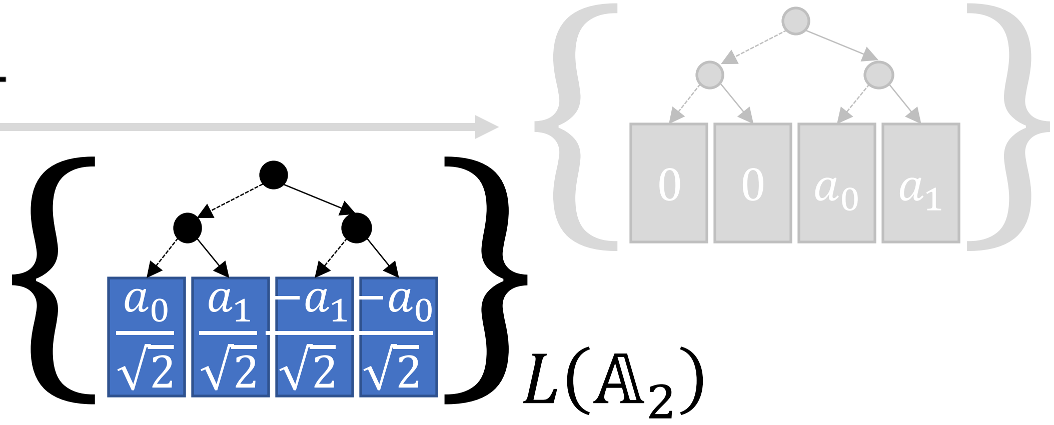
Post: $\{a_0 |10\rangle + a_1 |11\rangle$,

$-a_0 |11\rangle - a_1 |10\rangle\}$;

while $(M_q = b)$ do $\{P\}$

Algorithm 2: “$-X_2$”

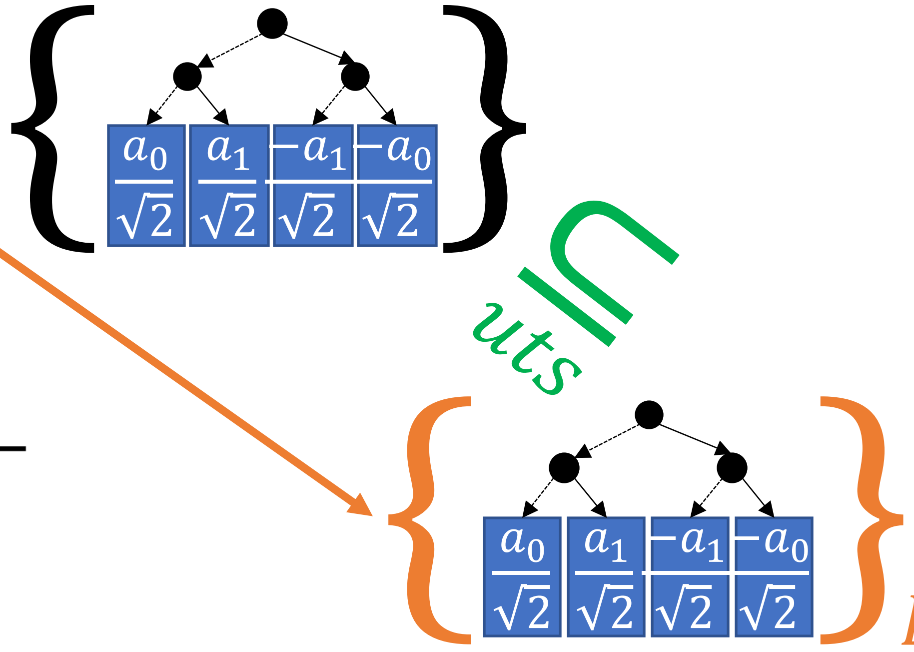
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
$H_1; CX_2^1$;
Inv: $\{\frac{a_0}{\sqrt{2}} |00\rangle - \frac{a_0}{\sqrt{2}} |11\rangle + \frac{a_1}{\sqrt{2}} |01\rangle - \frac{a_1}{\sqrt{2}} |10\rangle\}$;
while $M_1 = 0$ **do** $\{X_1; H_1; CX_2^1\}$;
Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;



while $(M_q = b)$ do $\{P\}$

Algorithm 2: “$-X_2$”

Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
$H_1; CX_2^1$;
Inv: $\{\frac{a_0}{\sqrt{2}} |00\rangle - \frac{a_0}{\sqrt{2}} |11\rangle + \frac{a_1}{\sqrt{2}} |01\rangle - \frac{a_1}{\sqrt{2}} |10\rangle\}$;
while $M_1 = 0$ do $\{X_1; H_1; CX_2^1\}$;
Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;



$L(\mathbb{A}_3)$

Step 1. Guess the loop invariant.

Step 2. Verify that the entry set is contained in the loop invariant.

while $(M_q = b)$ do $\{P\}$

Algorithm 2: "$-X_2$"

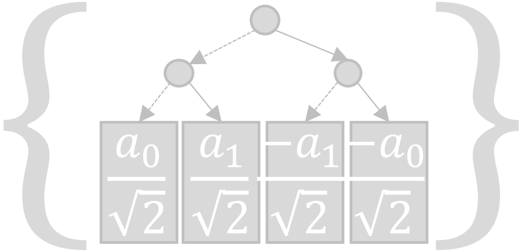
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

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

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

Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;

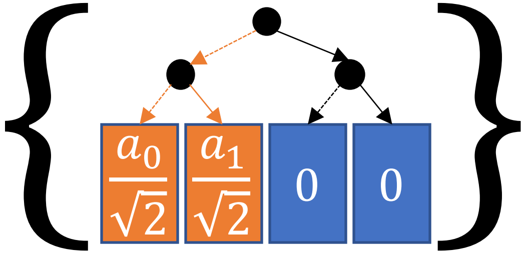


while $(M_1 = 0)$

do {

$X_1; H_1; C_1X_2$

}

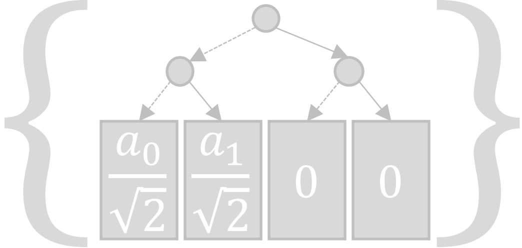


$L(\mathbb{A}_{4-\text{before}})$

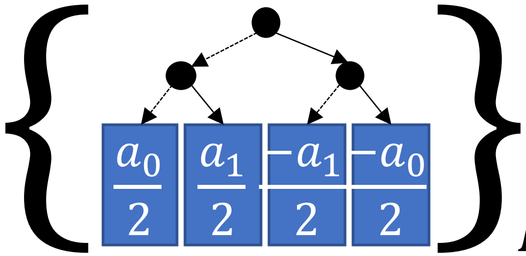
while $(M_q = b)$ do $\{P\}$

Algorithm 2: "$-X_2$"

Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
$H_1; CX_2^1$;
Inv: $\{\frac{a_0}{\sqrt{2}} |00\rangle - \frac{a_0}{\sqrt{2}} |11\rangle + \frac{a_1}{\sqrt{2}} |01\rangle - \frac{a_1}{\sqrt{2}} |10\rangle\}$;
while $M_1 = 0$ **do** $\{X_1; H_1; CX_2^1\}$;
Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;



while $(M_1 = 0)$
do {
$X_1; H_1; C_1X_2$
}

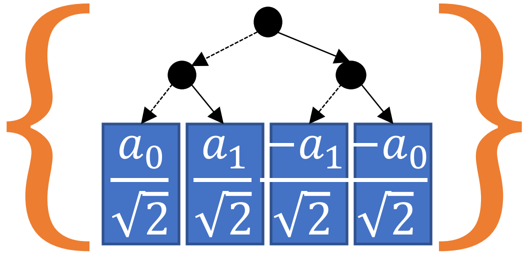


$L(\mathbb{A}_{4-\text{after}})$

while $(M_q = b)$ do $\{P\}$

Algorithm 2: "$-X_2$"

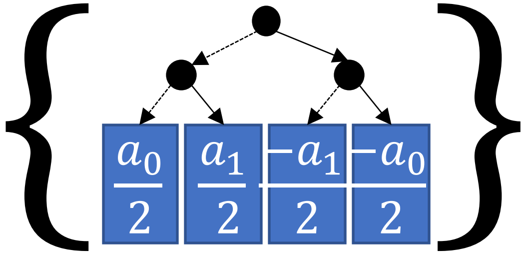
Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;
$H_1; CX_2^1$;
Inv: $\{\frac{a_0}{\sqrt{2}} |00\rangle - \frac{a_0}{\sqrt{2}} |11\rangle + \frac{a_1}{\sqrt{2}} |01\rangle - \frac{a_1}{\sqrt{2}} |10\rangle\}$;
while $M_1 = 0$ **do** $\{X_1; H_1; CX_2^1\}$;
Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;

→ 

$\subseteq_{uts}$

while $(M_1 = 0)$
do {
$X_1; H_1; C_1X_2$
}

Step 3. Verify that the exit set is also contained in the loop invariant.



←

while $(M_q = b)$ do $\{P\}$

Algorithm 2: "$-X_2$"

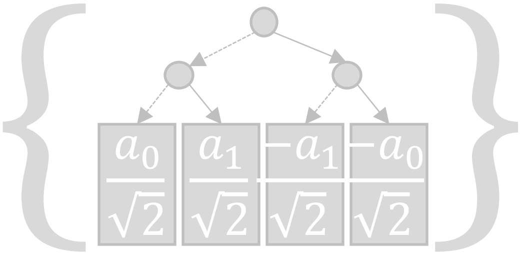
1 Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

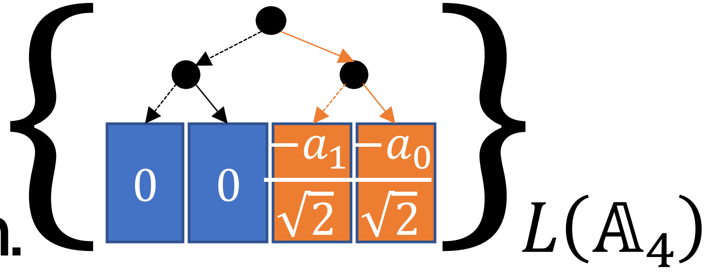
2 $H_1; CX_2^1$;

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

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

5 Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;





The resulting set after Line 4 is exactly the postcondition.

This algorithm essentially applies the negative X gate on the 2nd qubit.

Quantum Algorithm Examples

Table 1. Results of verifying some real-world examples with AutoQ 2.0. The number x in WMGrover(x) indicates that the number of items to be searched is 2^x.

Weakly Measured Grover's Search [6]						*Repeat-Until-Success* [41]					
program	**qubits**	**gates**	**result**	**time**	**memory**	**program**	**qubits**	**gates**	**result**	**time**	**memory**
WMGrover (03)	7	50	OK	0.0s	42MB	$(2X + \sqrt{2}Y + Z)/\sqrt{7}$	2	29	OK	0.0s	7MB
WMGrover (10)	21	169	OK	0.2s	42MB	$(I + i\sqrt{2}X)/\sqrt{3}$	2	17	OK	0.0s	7MB
WMGrover (20)	41	339	OK	0.8s	42MB	$(I + 2iZ)/\sqrt{5}$	2	27	OK	0.0s	6MB
WMGrover (30)	61	509	OK	2.3s	43MB	$(3I + 2iZ)/\sqrt{13}$	2	43	OK	0.0s	7MB
WMGrover (40)	81	679	OK	5.4s	43MB	$(4I + iZ)/\sqrt{17}$	2	77	OK	0.0s	6MB
WMGrover (50)	101	849	OK	11s	44MB	$(5I + 2iZ)/\sqrt{29}$	2	69	OK	0.0s	7MB

- Weakly measured Grover's search: no need to know the number of iterations

Quantum Algorithm Examples

Table 1. Results of verifying some real-world examples with AutoQ 2.0. The number x in WMGrover(x) indicates that the number of items to be searched is 2^x.

Weakly Measured Grover's Search [6]						*Repeat-Until-Success* [41]					
program	**qubits**	**gates**	**result**	**time**	**memory**	**program**	**qubits**	**gates**	**result**	**time**	**memory**
WMGrover (03)	7	50	OK	0.0s	42MB	$(2X+\sqrt{2}Y+Z)/\sqrt{7}$	2	29	OK	0.0s	7MB
WMGrover (10)	21	169	OK	0.2s	42MB	$(I+i\sqrt{2}X)/\sqrt{3}$	2	17	OK	0.0s	7MB
WMGrover (20)	41	339	OK	0.8s	42MB	$(I+2iZ)/\sqrt{5}$	2	27	OK	0.0s	6MB
WMGrover (30)	61	509	OK	2.3s	43MB	$(3I+2iZ)/\sqrt{13}$	2	43	OK	0.0s	7MB
WMGrover (40)	81	679	OK	5.4s	43MB	$(4I+iZ)/\sqrt{17}$	2	77	OK	0.0s	6MB
WMGrover (50)	101	849	OK	11s	44MB	$(5I+2iZ)/\sqrt{29}$	2	69	OK	0.0s	7MB

- Repeat-until-success: implement quantum gates with while-loops.

Algorithm 2: "$-X_2$"

Pre: $\{a_0\,|10\rangle + a_1\,|11\rangle\}$;

$H_1; CX_2^1$;

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

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

Post: $\{-a_0\,|11\rangle - a_1\,|10\rangle\}$;

Quantum Algorithm Examples

Table 1. Results of verifying some real-world examples with AutoQ 2.0. The number x in WMGrover(x) indicates that the number of items to be searched is 2^x.

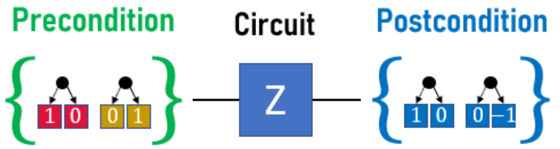
Weakly Measured Grover's Search [6]						*Repeat-Until-Success* [41]					
program	**qubits**	**gates**	**result**	**time**	**memory**	**program**	**qubits**	**gates**	**result**	**time**	**memory**
WMGrover (03)	7	50	OK	0.0s	42MB	$(2X + \sqrt{2}Y + Z)/\sqrt{7}$	2	29	OK	0.0s	7MB
WMGrover (10)	21	169	OK	0.2s	42MB	$(I + i\sqrt{2}X)/\sqrt{3}$	2	17	OK	0.0s	7MB
WMGrover (20)	41	339	OK	0.8s	42MB	$(I + 2iZ)/\sqrt{5}$	2	27	OK	0.0s	6MB
WMGrover (30)	61	509	OK	2.3s	43MB	$(3I + 2iZ)/\sqrt{13}$	2	43	OK	0.0s	7MB
WMGrover (40)	81	679	OK	5.4s	43MB	$(4I + iZ)/\sqrt{17}$	2	77	OK	0.0s	6MB
WMGrover (50)	101	849	OK	11s	44MB	$(5I + 2iZ)/\sqrt{29}$	2	69	OK	0.0s	7MB

Possible Improvement

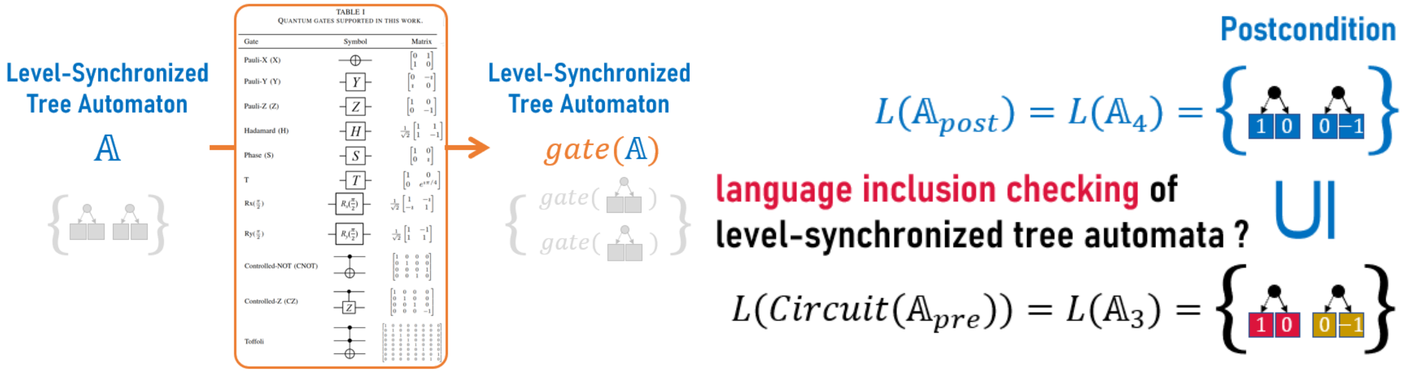
- Automatic synthesis of loop invariants.

Takeaways: What You Should Know

- We propose a quantum verification framework with Hoare triples.



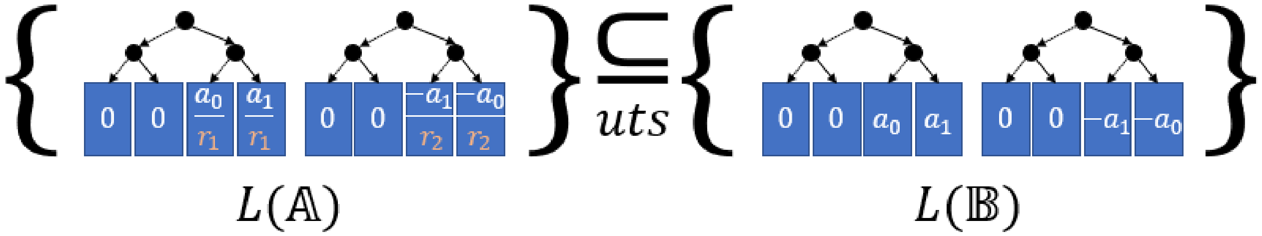
- We use level-synchronized tree automata to encode sets of quantum states and fully automate the framework.



- We extend the framework to additionally support branches and loops, which constitute quantum programs.

if $(M_q = b)$ **then** $\{P_1\}$ **else** $\{P_2\}$ **while** $(M_q = b)$ **do** $\{P\}$

- We bypass the need of normalizing the amplitudes and successfully develop the up-to-scaling inclusion checking algorithm.



THE END

Quantum Algorithm Examples

- Weakly Measured Grover's Search → no need for the number of iterations

Algorithm 6: A Weakly Measured Version of Grover's algorithm (solution $s = 0^n$)

Pre: $\{1\,|0^{n+2}\rangle + 0\,|*\rangle\}$;

$H_3;\ H_4;\ \ldots;\ H_{n+2}$;

$\mathcal{O}_{2,\ldots,(n+2)};\ CK_1^2;\ \mathcal{O}_{2,\ldots,(n+2)}$;

Inv: $\{v_{sol1}\,|000^n\rangle + v_k\,|000^{n-1}1\rangle + \cdots +$

$v_k\,|001^n\rangle + v_{sol2}\,|100^n\rangle + 0\,|*\rangle\}$;

while $M_1 = 0$ **do**

$\quad \{\mathcal{G}_{2,\ldots,(n+2)};\ \mathcal{O}_{2,\ldots,(n+2)};\ CK_1^2;\ \mathcal{O}_{2,\ldots,(n+2)}\}$;

Post: $\{1\,|10s\rangle + 0\,|*\rangle\}$;

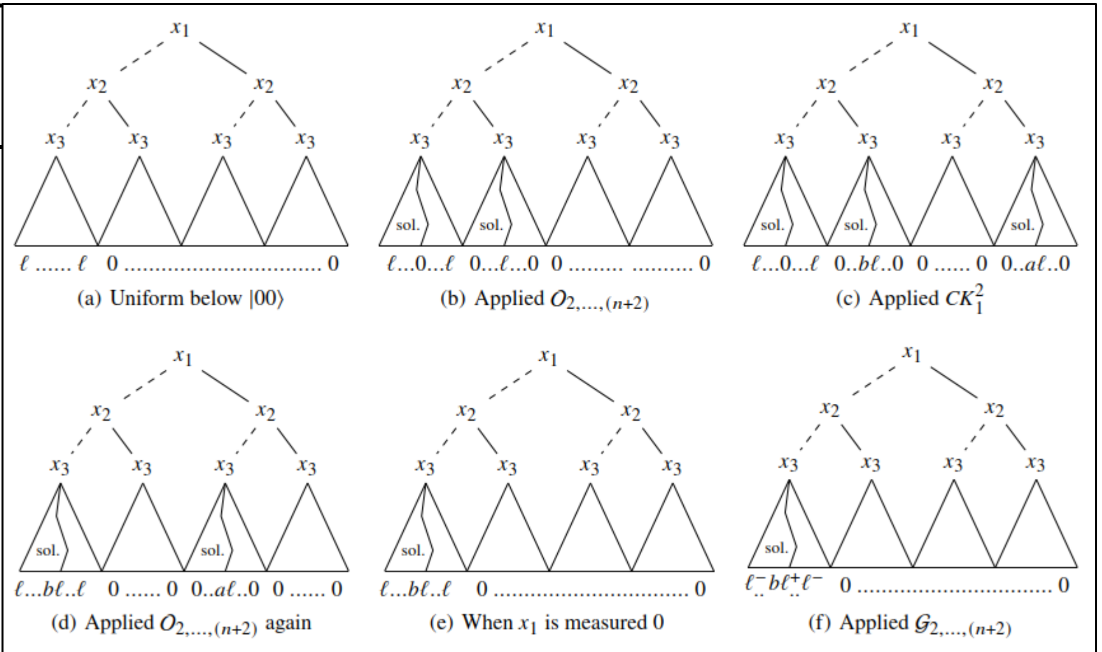


Fig. 7. Intermediate states of Algorithm 6

Quantum Algorithm Examples

- Repeat-Until-Success

Algorithm 2: "$-X_2$"

Pre: $\{a_0 |10\rangle + a_1 |11\rangle\}$;

$H_1; CX_2^1$;

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

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

Post: $\{-a_0 |11\rangle - a_1 |10\rangle\}$;