中央研究院
ACADEMIA SINICA

AN **AUTOMATA-BASED** APPROACH FOR SCALABLE VERIFICATION OF QUANTUM CIRCUITS

Yu-Fang Chen[1], Kai-Min Chung[1], Ondrej Lengal[2], **Jyun-Ao Lin[1]**, Wei-Lun Tsai[1], Di-De Yen[1]

Academia Sinica, Taiwan[1] , Brno University of Technology, Czech Republic[2]

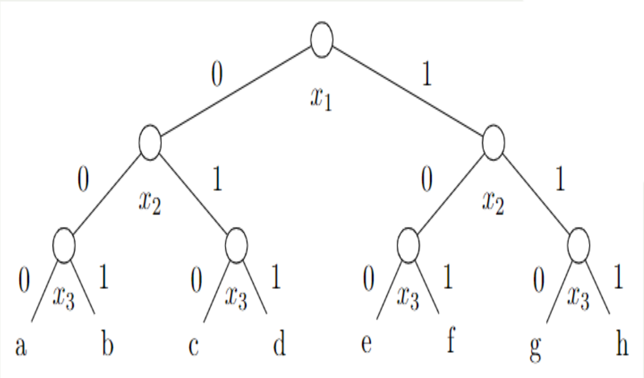
QPL 2023, Paris

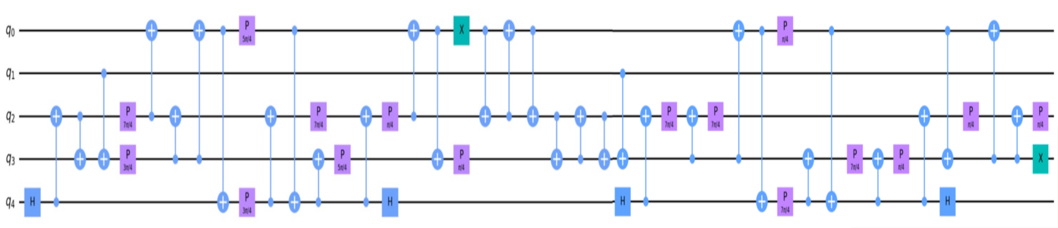
Outline

- **Motivation**
- Quantum Circuit Verification
- Evaluation

Why Quantum Circuit Verification?

- Increasing complexity of circuits
- Infeasibility of testing
- Challenge of probabilistic features





State-of-the-Art

We focus on **fully** automatic approaches:

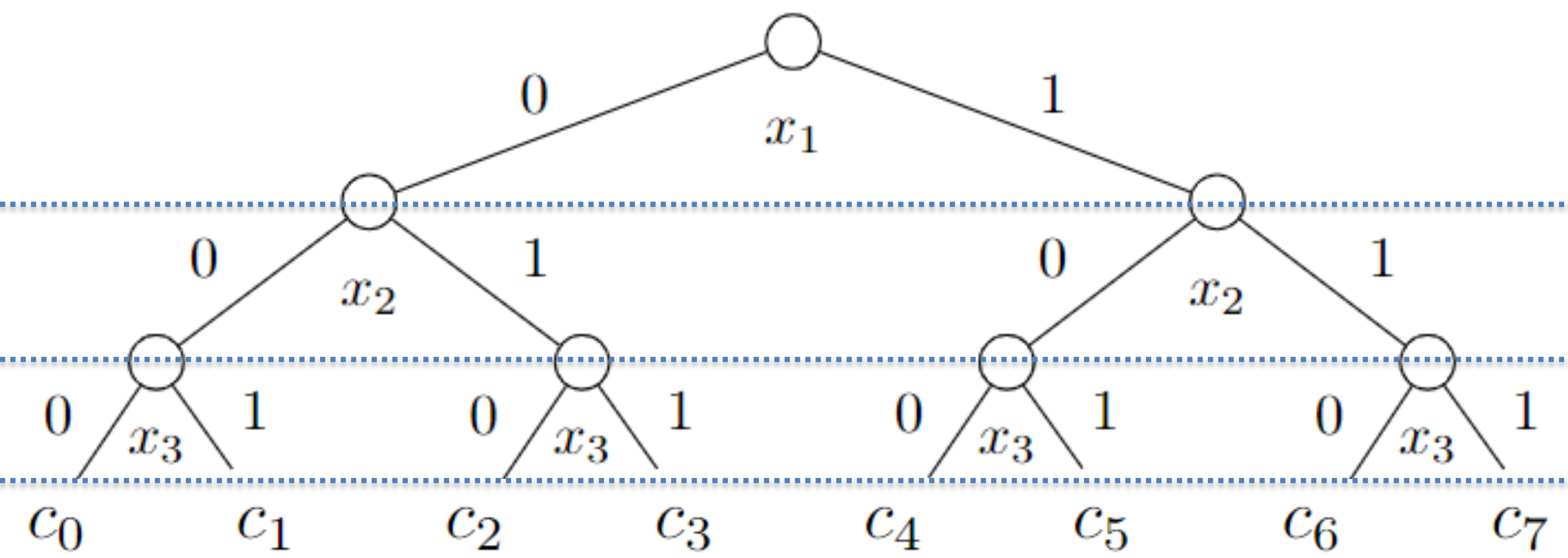
- **Quantum Simulation**
  - low coverage
- **Quantum Abstraction Interpretation**
  - cannot catch bugs
- **Quantum Model Checking**
  - only work for small examples
- **Circuit Equivalence Checking**
  - inflexible for user custom properties

Outline

- Minimal Quantum Background and Motivation
- **Quantum Circuit Verification with Pre and Post-Conditions**
- Evaluation

Tree as a Quantum State

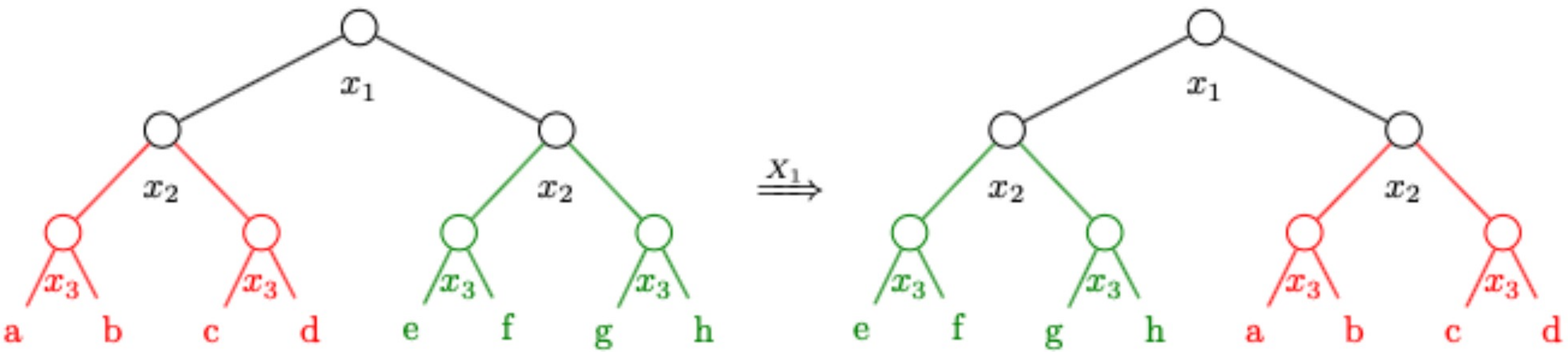
- A 3-bit quantum state



$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} \begin{array}{c} |000\rangle \\ |001\rangle \\ |010\rangle \\ |011\rangle \\ |100\rangle \\ |101\rangle \\ |110\rangle \\ |111\rangle \end{array}$$

Quantum Gate and Tree Transformation

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



Classical Hoare triple

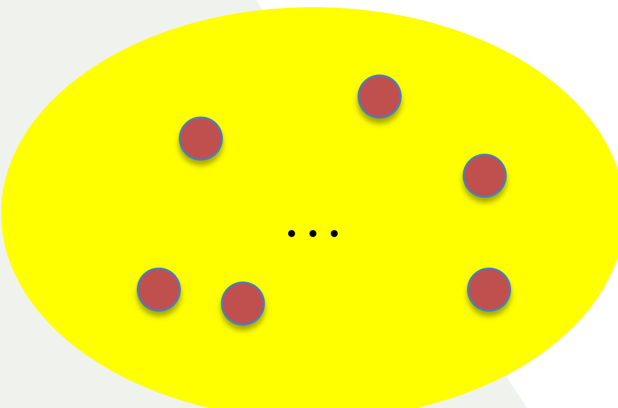
- For any predicates P and Q and any program S,

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

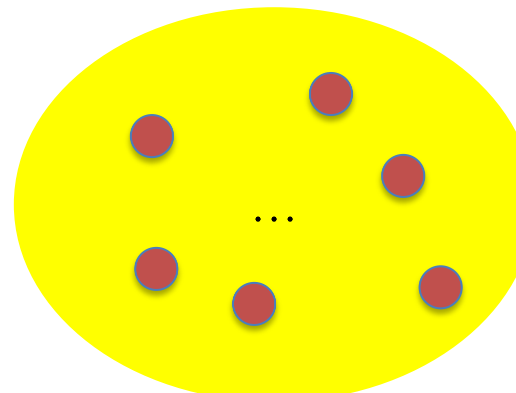
says that if S is started in (a state satisfying) P, then it terminates in Q.

Classical Hoare triple

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



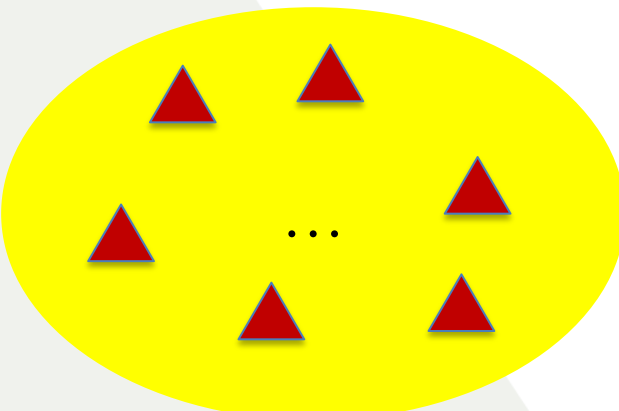
$$\{P\}\ S\ \{Q\}$$



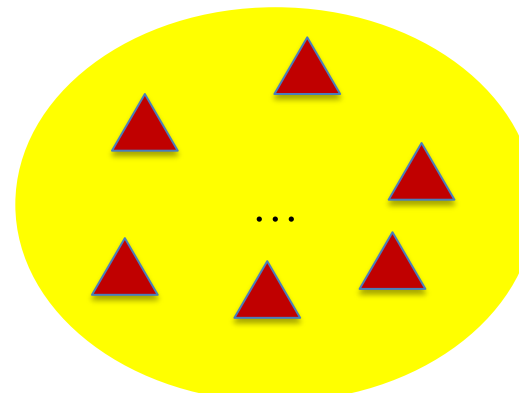
says that if S is started in (a state satisfying) P, then it terminates in Q.

Quantum Circuit Verification

- For any predicates P and Q and any circuit C,



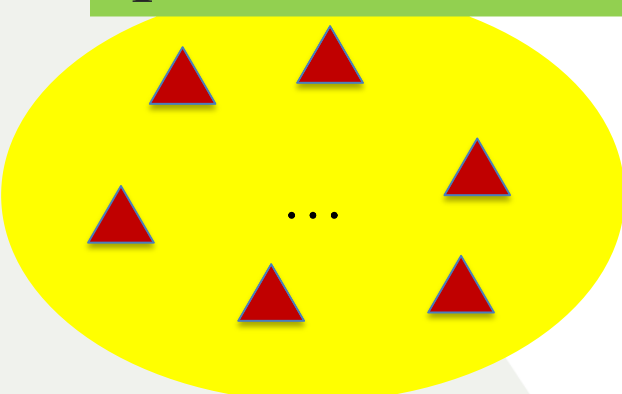
$$\{P\}\ C\ \{Q\}$$



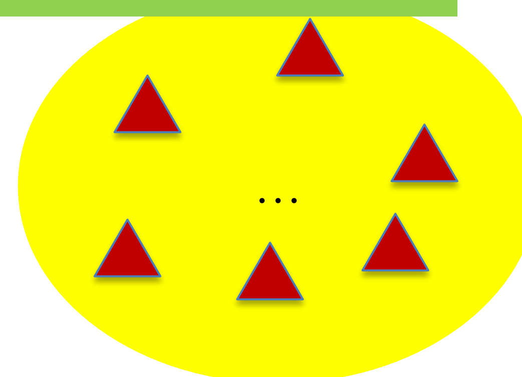
says that if C is started in (a state satisfying) P, then it terminates in Q.

Quantum Circuit Verification

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

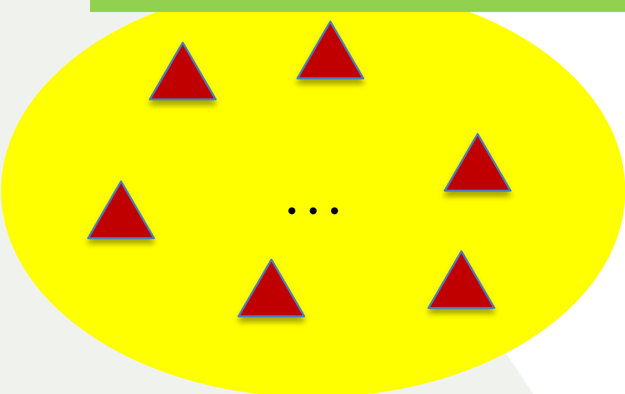


$\{P\}\ C\ \{Q\}$

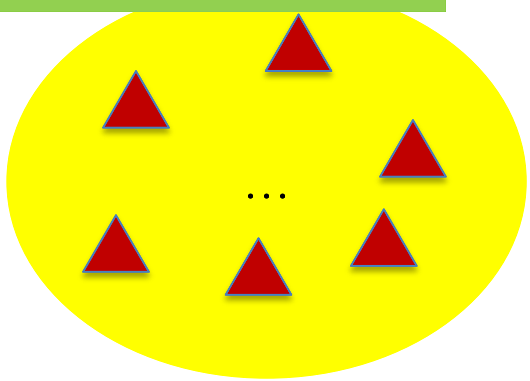


Quantum Circuit Verification

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



$\{P\}\ C\ \{Q\}$



From automata theory:

Set of words ➔ Regular language (Finite automata)

Set of trees ➔ Regular tree language (Tree automata)

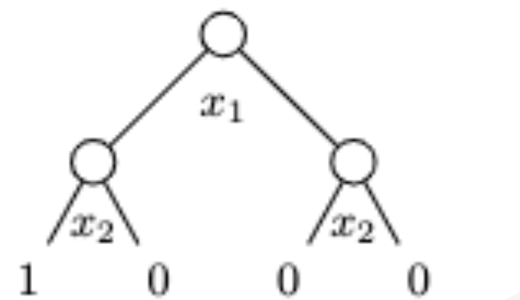
Tree Automata Encoding of Quantum States

$q - \boxed{x_1} \rightarrow (q_1, q_0)$

$q_0 - \boxed{x_2} \rightarrow (q_2, q_2)$ $\qquad$ $q_2 - \boxed{0} \rightarrow ()$

$q_1 - \boxed{x_2} \rightarrow (q_3, q_2)$ $\qquad$ $q_3 - \boxed{1} \rightarrow ()$

Figure 1: The TA of $|00\rangle$.



Tree Automata Encoding of Quantum States

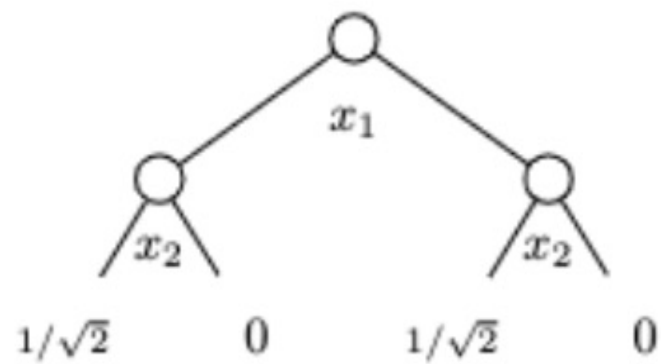
$q \text{—}\boxed{x_1}\!\to (q_0, q_0)$

$q_0 \text{—}\boxed{x_2}\!\to (q_1, q_2)$

$q_1 \text{—}\boxed{1/\sqrt{2}}\!\to ()$

$q_2 \text{—}\boxed{0}\!\to ()$

Figure 2: The TA of $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$



Tree Automata Encoding of Quantum States

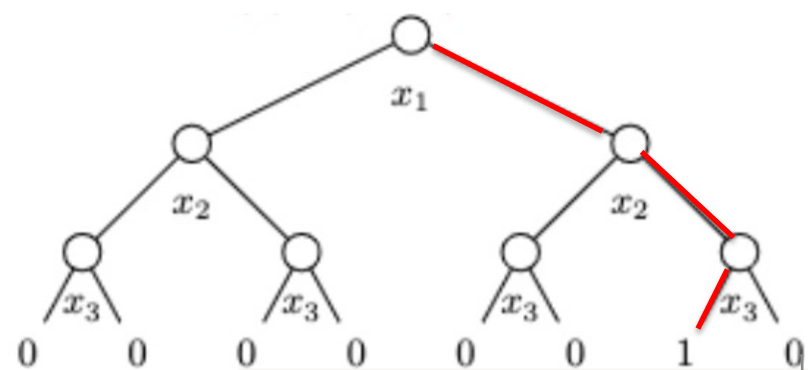
$$q \xrightarrow{\boxed{x_1}} (q_0^1, q_1^1) \qquad q_1^1 \xrightarrow{\boxed{x_2}} (q_0^2, q_1^2) \qquad q_1^2 \xrightarrow{\boxed{x_3}} (q_0, q_1) \qquad q_0 \xrightarrow{\boxed{0}} ()$$

$$q \xrightarrow{\boxed{x_1}} (q_1^1, q_0^1) \qquad q_1^1 \xrightarrow{\boxed{x_2}} (q_1^2, q_0^2) \qquad q_1^2 \xrightarrow{\boxed{x_3}} (q_1, q_0) \qquad q_1 \xrightarrow{\boxed{1}} ()$$

$$q_0^1 \xrightarrow{\boxed{x_2}} (q_0^2, q_0^2) \qquad q_0^2 \xrightarrow{\boxed{x_3}} (q_0, q_0)$$

Figure 3: The TA of all 3-qubit basis states.

- $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle,$ $|100\rangle, |101\rangle,$ $|110\rangle$$, |111\rangle\}$



TA as Compact Representation of Quantum States

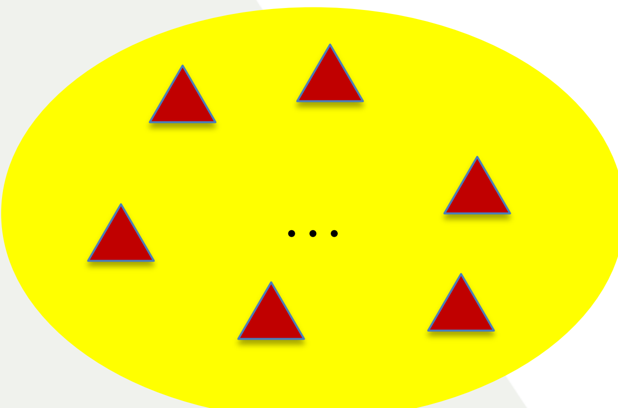
- This TA accepts all 2^n basis states.
  # of transitions: $3n+1$

$$
\begin{array}{lllll}
q \xrightarrow{x_1} (q_0^1, q_1^1) & q_1^1 \xrightarrow{x_2} (q_0^2, q_1^2) & \dots & q_1^{n-1} \xrightarrow{x_n} (q_0, q_1) & q_0 \xrightarrow{\mathbf{0}} () \\
q \xrightarrow{x_1} (q_1^1, q_0^1) & q_1^1 \xrightarrow{x_2} (q_1^2, q_0^2) & \dots & q_1^{n-1} \xrightarrow{x_n} (q_1, q_0) & q_1 \xrightarrow{\mathbf{1}} () \\
 & q_0^1 \xrightarrow{x_2} (q_0^2, q_0^2) & \dots & q_0^{n-1} \xrightarrow{x_n} (q_0, q_0) &
\end{array}
$$

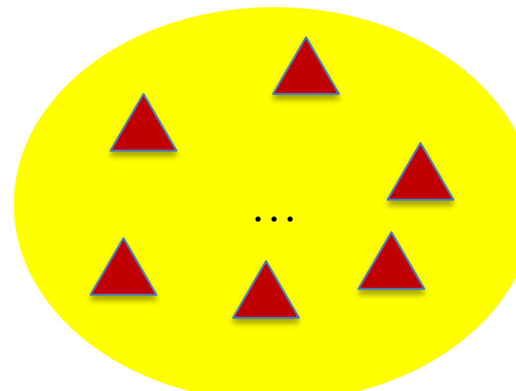
Figure 4: The TA of all 3-qubit basis states.

Quantum Circuit Verification

- For any predicates P and Q and any circuit C,



{P} C {Q}

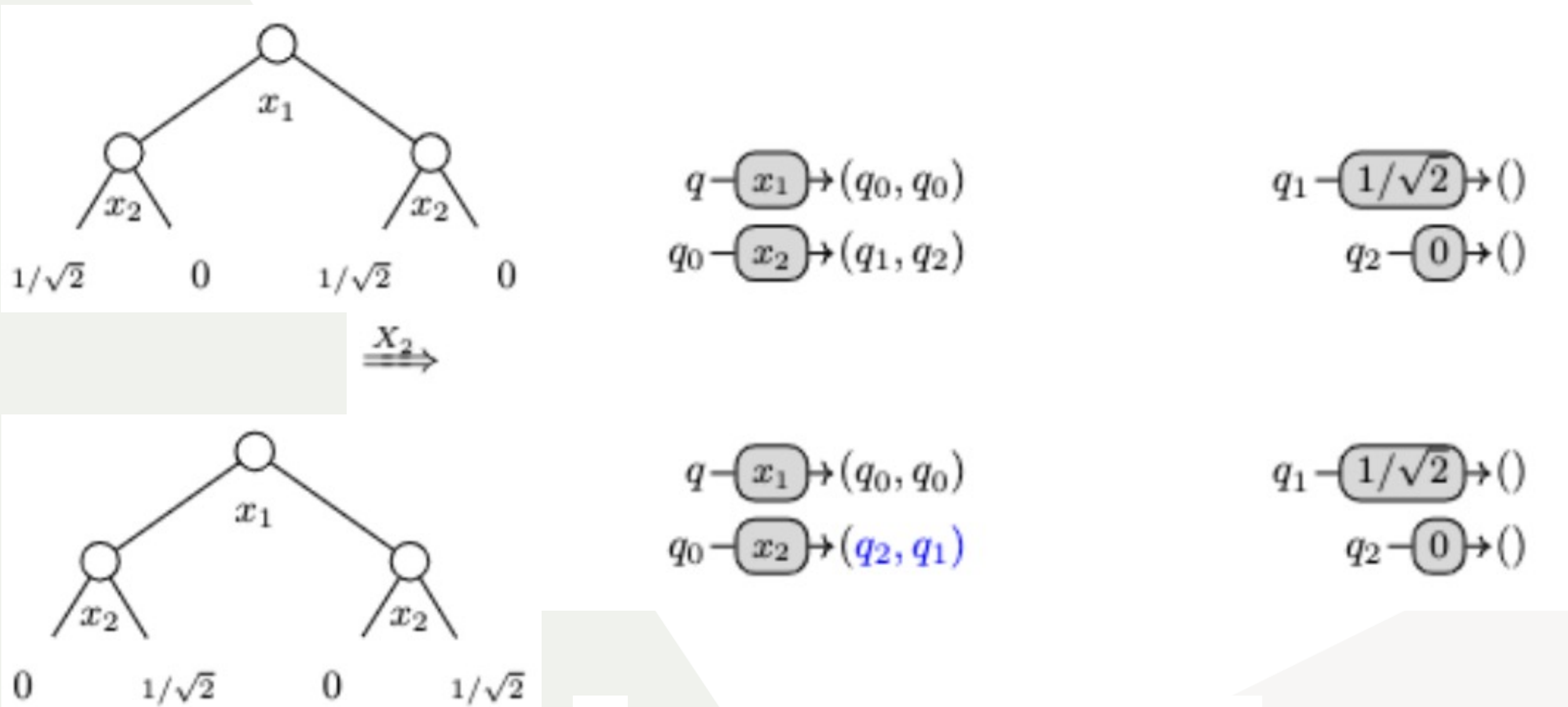


says that if C is started in (a state satisfying) P, then it terminates in Q.

Two Approaches for TA Gate Operations

- Permutation-based approach:
  - Faster, but works for a smaller set of gates.
  - Done by directly modifying TA transitions.
- Composition-based approach
  - Slower, but complete for universal quantum computing.

Example of an Operation: X gate on qubit 2.



Example of an Operation: Z, S, T gates on qubit 1.

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

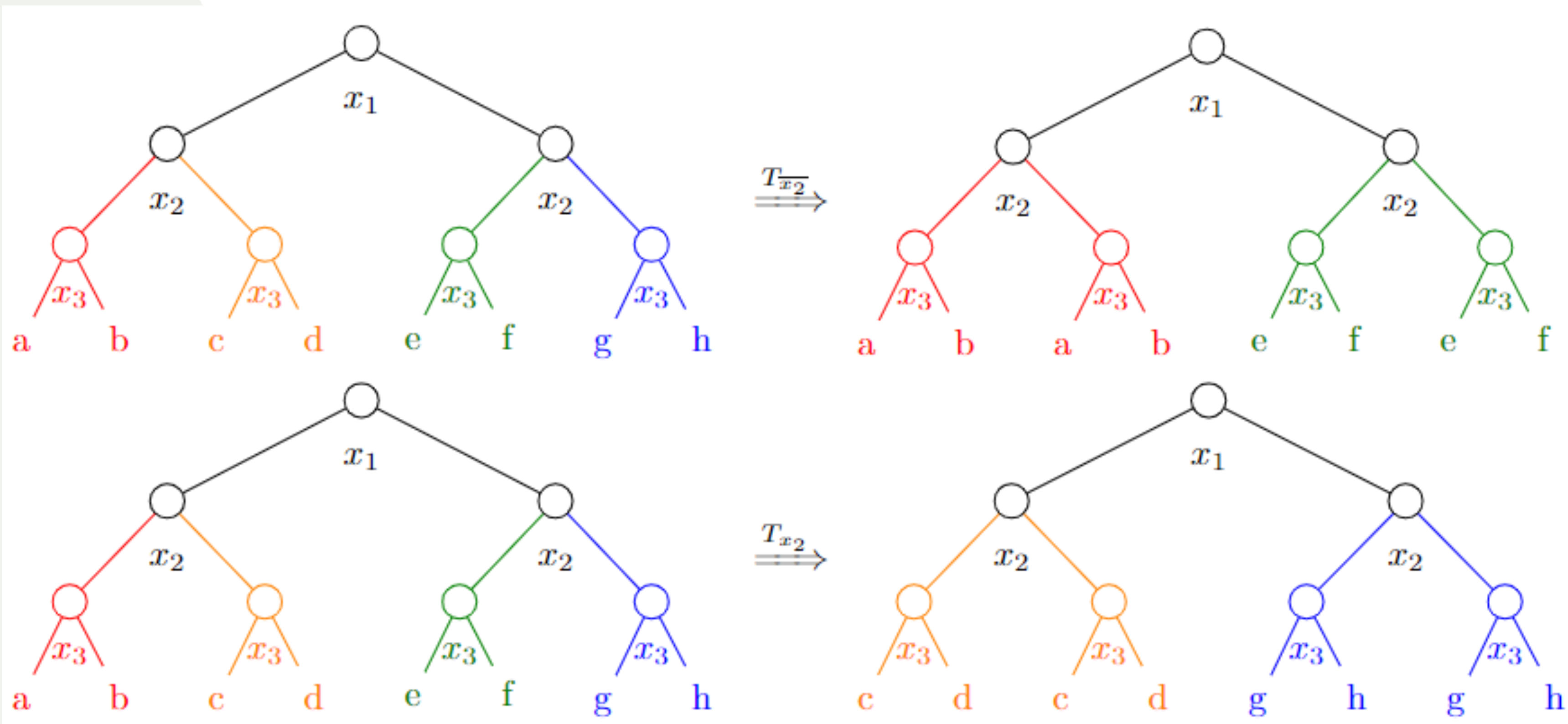
$$
\begin{array}{llll}
q \xrightarrow{\boxed{x_1}} (q_0^1, q_1^1) & q_1^1 \xrightarrow{\boxed{x_2}} (q_0^2, q_1^2) & q_1^2 \xrightarrow{\boxed{x_3}} (q_0, q_1) & q_0 \xrightarrow{\boxed{0}} () \\
q \xrightarrow{\boxed{x_1}} (q_1^1, q_0^1) & q_1^1 \xrightarrow{\boxed{x_2}} (q_1^2, q_0^2) & q_1^2 \xrightarrow{\boxed{x_3}} (q_1, q_0) & q_1 \xrightarrow{\boxed{1}} () \\
 & q_0^1 \xrightarrow{\boxed{x_2}} (q_0^2, q_0^2) & q_0^2 \xrightarrow{\boxed{x_3}} (q_0, q_0) &
\end{array}
$$

$$
\begin{array}{llll}
q \xrightarrow{\boxed{x_1}} (q_0^1, q_1^1) & q_1^1 \xrightarrow{\boxed{x_2}} (q_0^2, q_1^2) & q_1^2 \xrightarrow{\boxed{x_3}} (q_0, q_1) & q_0 \xrightarrow{\boxed{c \times 0}} () \\
q \xrightarrow{\boxed{x_1}} (q_1^1, q_0^1) & q_1^1 \xrightarrow{\boxed{x_2}} (q_1^2, q_0^2) & q_1^2 \xrightarrow{\boxed{x_3}} (q_1, q_0) & q_1 \xrightarrow{\boxed{c \times 1}} () \\
 & q_0^1 \xrightarrow{\boxed{x_2}} (q_0^2, q_0^2) & q_0^2 \xrightarrow{\boxed{x_3}} (q_0, q_0) &
\end{array}
$$

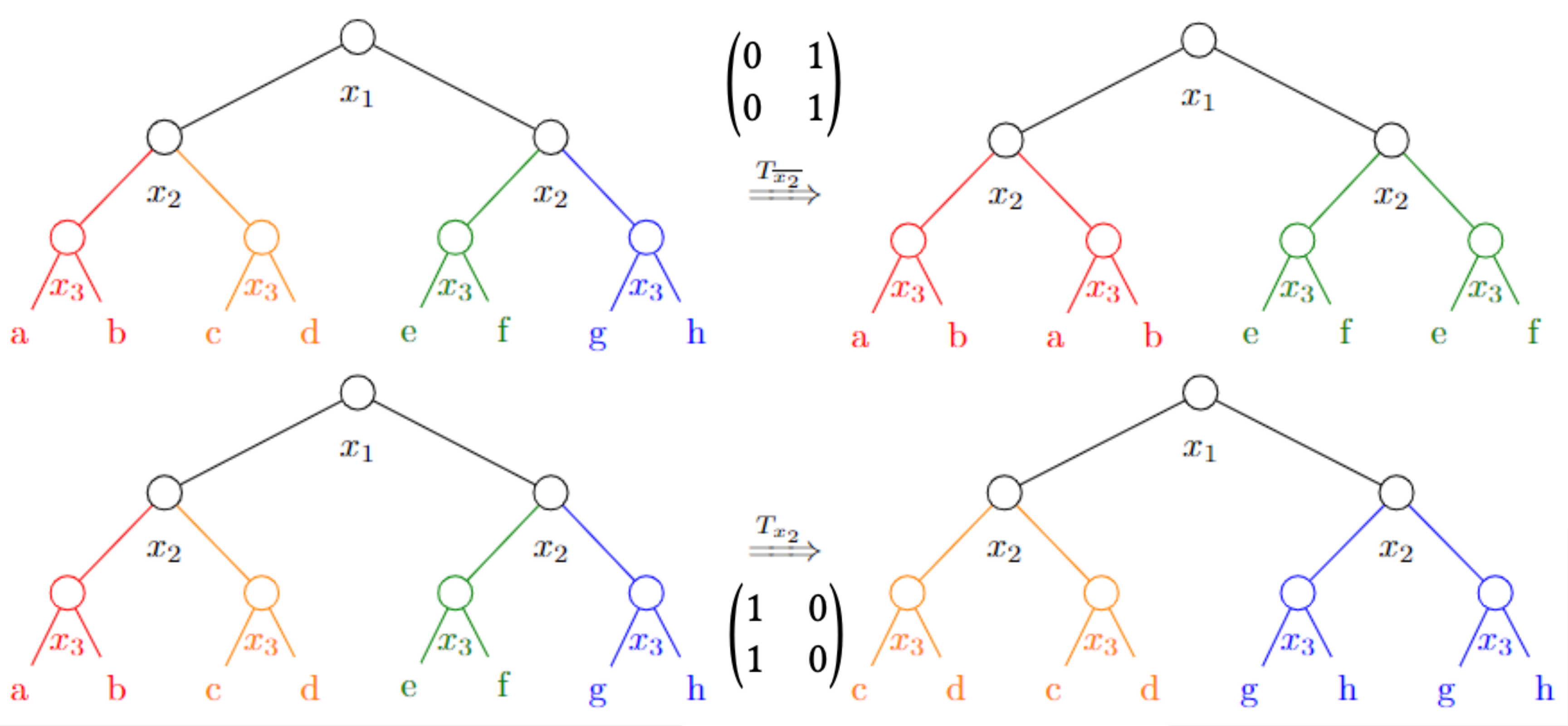
Two Approaches for TA Gate Operations

- Permutation-based approach:
  - Faster, but works for a smaller set of gates.
  - Done by directly modifying TA transitions.
- Composition-based approach
  - Slower, but complete for universal quantum computing.

1. Projection ($T_{\overline{x_2}}$ & T_{x_2})



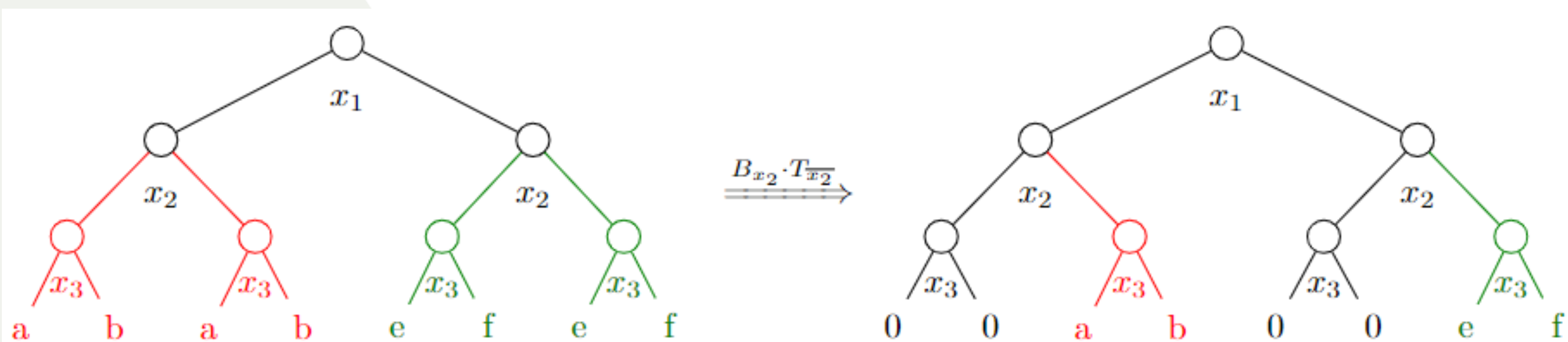
1. Projection ($T_{\overline{x_2}}$ & T_{x_2})



2. Restriction ($B_{x_2}\cdot$ & $B_{\overline{x_2}}\cdot$)

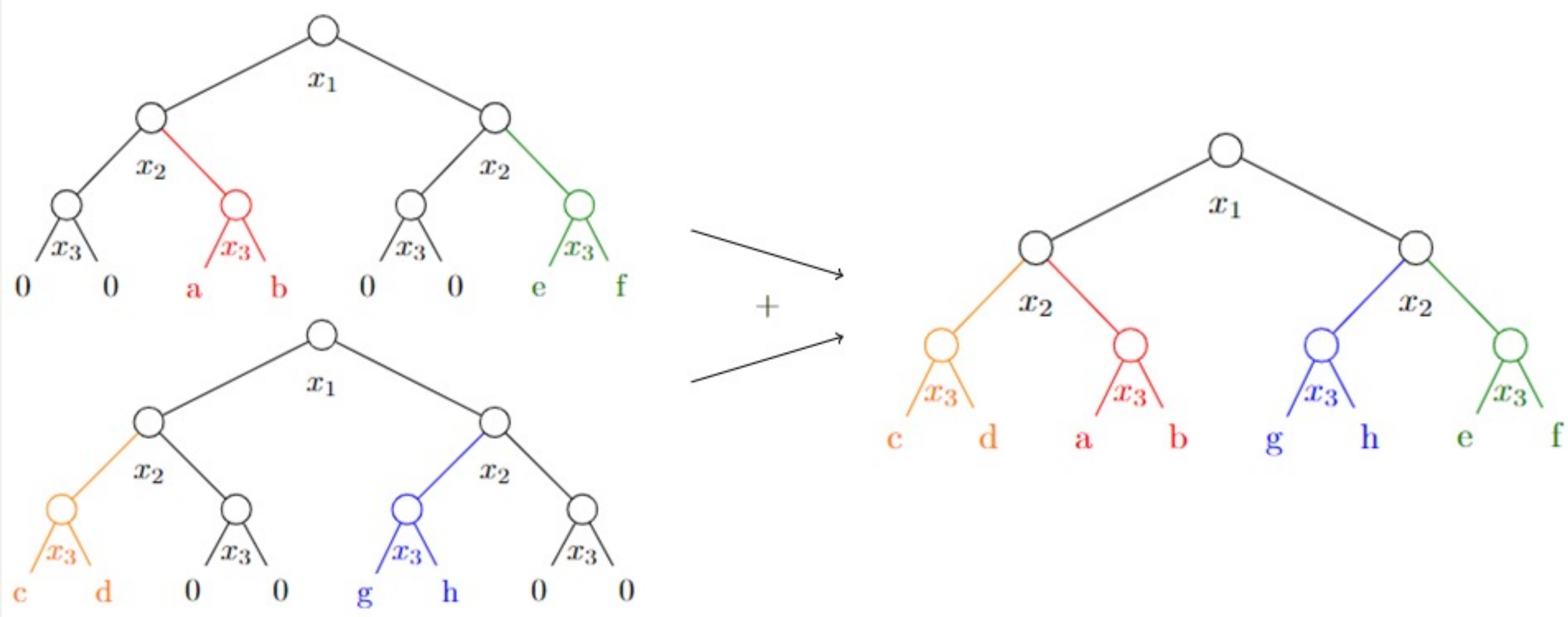
$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

- $B_{x_2}\cdot T_{\overline{x_2}}$



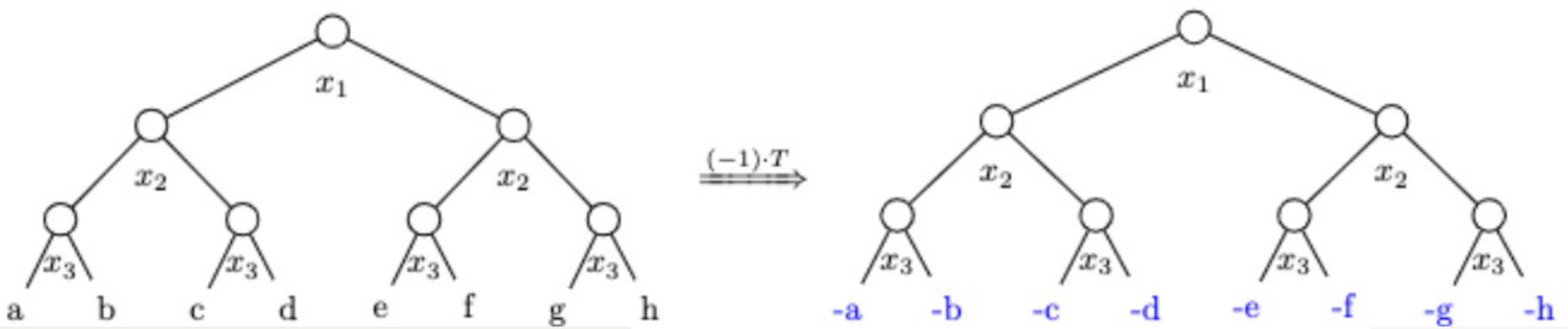
3. Binary Operation (+ & –)

- $B_{x_2} \cdot T_{\overline{x_2}} + B_{\overline{x_2}} \cdot T_{x_2}$



4. Constant Multiplication (c ·)

- $(-1) \cdot T$



Tree Operations

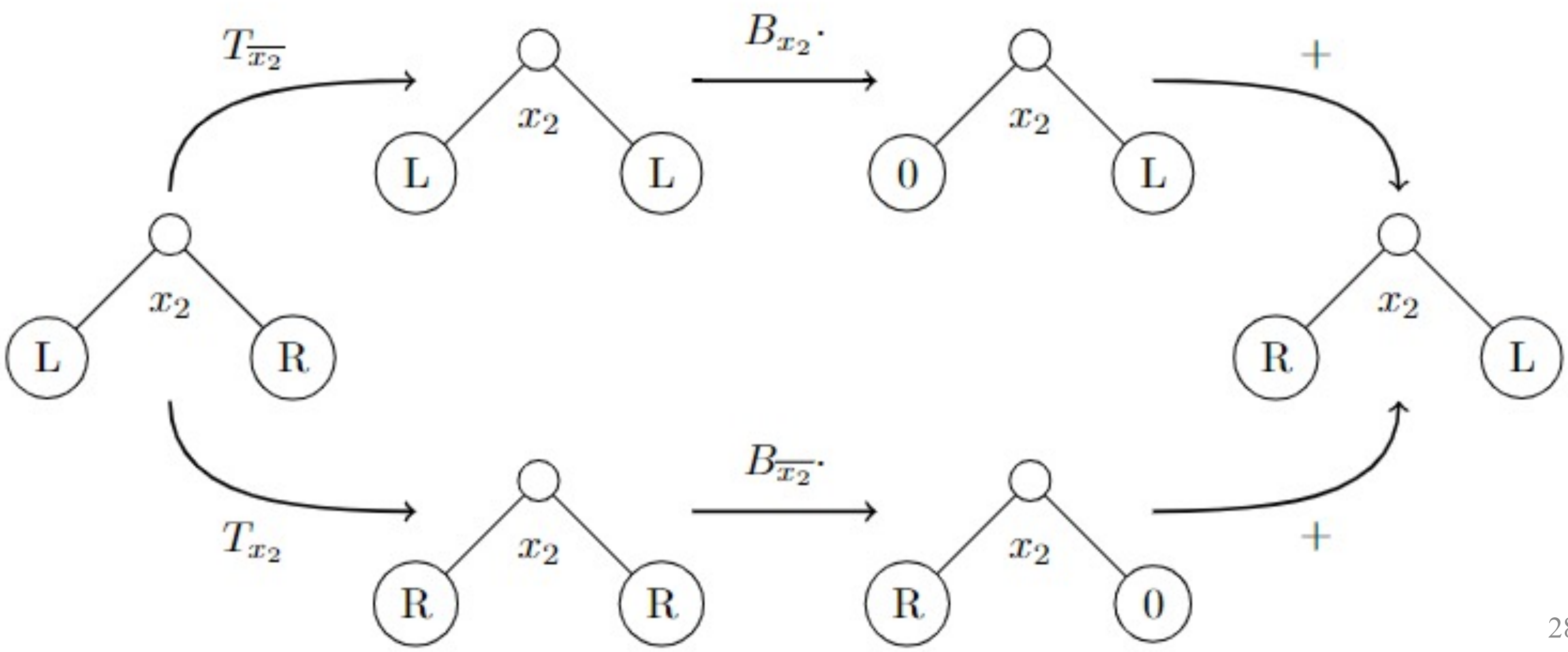
Table 1. Symbolic update formulae for the considered quantum gates. Notice that in all cases x_c and x_c' denote the two control bits, and x_t and x_t' denote the two target bits, if they exist.

Gate	Update
X_t	$\boxed{B_{x_t} \cdot} T_{\overline{x_t}} + B_{\overline{x_t}} \cdot \boxed{T_{x_t}}$
Y_t	$\omega^2 \cdot (B_{x_t} \cdot T_{\overline{x_t}} - B_{\overline{x_t}} \cdot T_{x_t})$
Z_t	$B_{\overline{x_t}} \cdot T \boxed{-} B_{x_t} \cdot T$
H_t	$(T_{\overline{x_t}} + B_{\overline{x_t}} \cdot T_{x_t} - B_{x_t} \cdot T)/\sqrt{2}$
S_t	$B_{\overline{x_t}} \cdot T + \omega^2 \cdot B_{x_t} \cdot T$
T_t	$B_{\overline{x_t}} \cdot T + \boxed{\omega \cdot} B_{x_t} \cdot T$
$\mathrm{Rx}(\frac{\pi}{2})_t$	$(T - \omega^2 \cdot (B_{x_t} \cdot T_{\overline{x_t}} + B_{\overline{x_t}} \cdot T_{x_t}))/\sqrt{2}$
$\mathrm{Ry}(\frac{\pi}{2})_t$	$(T_{\overline{x_t}} + B_{x_t} \cdot T - B_{\overline{x_t}} \cdot T_{x_t})/\sqrt{2}$
CNOT_t^c	$B_{\overline{x_c}} \cdot T + B_{x_c} \cdot B_{\overline{x_t}} \cdot T_{x_t} + B_{x_c} \cdot B_{x_t} \cdot T_{\overline{x_t}}$
CZ_t^c	$B_{\overline{x_c}} \cdot T + B_{\overline{x_t}} \cdot T - B_{\overline{x_c}} \cdot B_{\overline{x_t}} \cdot T - B_{x_c} \cdot B_{x_t} \cdot T$
$\mathrm{Toffoli}_t^{c,c'}$	$B_{\overline{x_c}} \cdot T + B_{\overline{x_{c'}}} \cdot T - B_{\overline{x_c}} \cdot B_{\overline{x_{c'}}} \cdot T + B_{x_t} \cdot B_{x_c} \cdot B_{x_{c'}} \cdot T_{\overline{x_t}} + B_{\overline{x_t}} \cdot B_{x_c} \cdot B_{x_{c'}} \cdot T_{x_t}$
$\mathrm{Fredkin}_{t,t'}^c$	$B_{\overline{x_c}} \cdot T + B_{x_t} \cdot B_{x_{t'}} \cdot B_{x_c} \cdot T + B_{\overline{x_t}} \cdot B_{\overline{x_{t'}}} \cdot B_{x_c} \cdot T + B_{x_t} \cdot B_{\overline{x_{t'}}} \cdot B_{x_c} \cdot T_{\overline{x_t} x_{t'}} + B_{\overline{x_t}} \cdot B_{x_{t'}} \cdot B_{x_c} \cdot T_{x_t \overline{x_{t'}}}$

Yuan-Hung Tsai, Jie-Hong R. Jiang, and Chiao-Shan Jhang. 2021. Bit-Slicing the Hilbert Space: Scaling Up Accurate Quantum Circuit Simulation. In *58th ACM/IEEE Design Automation Conference, DAC 2021, San Francisco, CA, USA, December 5-9, 2021*. IEEE, 439–444. https://doi.org/10.1109/DAC18074.2021.9586191

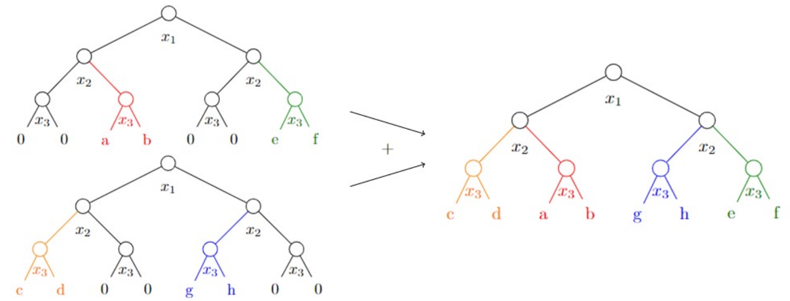
X gate operating on qubit 2

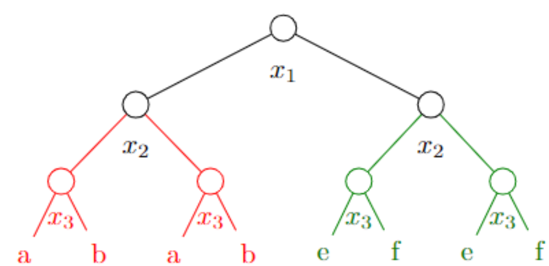
- $B_{x_2} \cdot T_{\overline{x_2}} + B_{\overline{x_2}} \cdot T_{x_2}$



Lift these operations to TA

- Constant Multiplication
- Restriction
- Binary Operation
- Projection
  - need to say the same left and right subtrees.





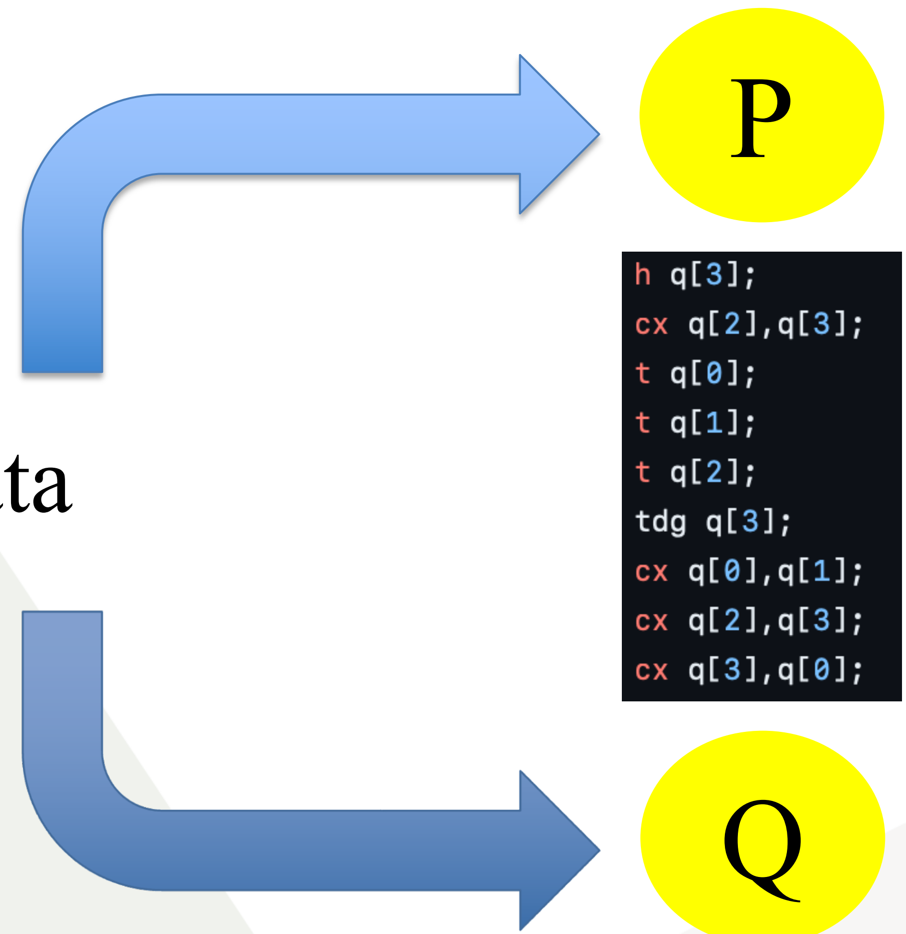
Quantum Circuit Verification

- Hoare Triple: {P} C {Q}, ideally
  - P, Q: we use regular tree language as predicates
  - C: a sequence of quantum gates
- Our approach
  - Compute C(P), the set of states after executing C.
  - Test if C(P) ⊆ Q (via standard TA algorithms).

Yu-Fang Chen, Kai-Min Chung, Ondřej Lengál, Jyun-Ao Lin, Wei-Lun Tsai, Di-De Yen.
An Automata-based Framework for Verification and Bug Hunting in Quantum Circuits
PLDI 2023 (Distinguished paper award) and a presentation at **QPL 2023**

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^1

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

中央研究院
ACADEMIA SINICA

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)

Outline

- Minimal Quantum Background and Motivation
- Quantum Circuit Verification in the Hoare-Style
- **Evaluation**

Experiment - Verification

- BV, Grover-Sing: $|P| = 1$
- MCToffoli, Grover-All: $|P| \gg 1$

	n	#q	#G	AutoQ-Hybrid analysis	AutoQ-Hybrid =
BV	95	96	241	6.0s	0.0s
	96	97	243	5.9s	0.0s
	97	98	246	6.3s	0.0s
	98	99	248	6.5s	0.0s
	99	100	251	6.7s	0.0s
Grover-Sing	12	24	5,215	11s	0.0s
	14	28	12,217	31s	0.0s
	16	32	28,159	1m29s	0.0s
	18	36	63,537	4m1s	0.0s
	20	40	141,527	10m56s	0.0s
MCToffoli	8	16	15	0.0s	0.0s
	10	20	19	0.0s	0.0s
	12	24	23	0.0s	0.0s
	14	28	27	0.1s	0.0s
	16	32	31	0.2s	0.0s
Grover-All	6	18	357	3.3s	0.0s
	7	21	552	10s	0.0s
	8	24	939	39s	0.1s
	9	27	1,492	2m17s	0.4s
	10	30	2,433	9m48s	2.1s

Experiment – Bug Hunting

- Create C_{bug} by appending one random gate at a random qubit to the end of C_{ori}.

- Verify if $\{P\}\ C_{bug}\ \{C_{ori}(P)\}$ is an invalid Hoare triple, where P starts from an arbitrary basis state and gradually includes also other basis states until AutoQ finds a bug.

中央研究院
ACADEMIA SINICA

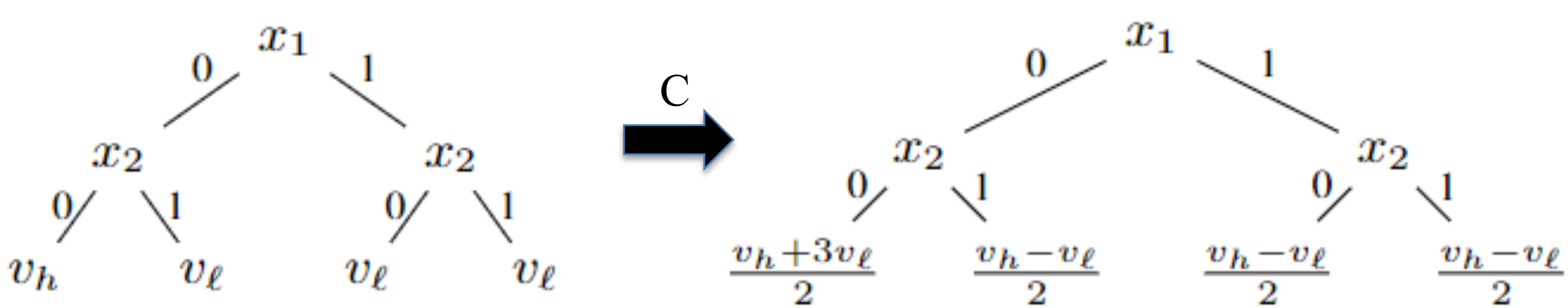
Experiment – Bug Hunting

	circuit	#q	#G	AutoQ time	AutoQ bug?	Feynman time	Feynman bug?	Qcec time	Qcec bug?
FeynmanBench	csum_mux_9	30	141	0.5s	T	6.0s	—	1m1s	F
	gf2^10_mult	30	348	1.5s	T	0.5s	—	1.6s	—
	gf2^16_mult	48	876	9.3s	T	3.6s	—	1m25s	T
	gf2^32_mult	96	3,323	2m0s	T	51s	—	2m52s	T
	ham15-high	20	1,799	7.5s	T	3m50s	—	56.8s	T
	mod_adder_1024	28	1,436	10s	T	8.7s	—	1m2s	T
Random	35a	35	106	2.6s	T	0.2s	—	1m4s	F
	35b	35	106	1.3s	T	0.1s	T	1m5s	F
	35c	35	106	1.1s	T	0.1s	T	1m7s	T
	35d	35	106	1.1s	T	0.1s	T	1m5s	T
	35e	35	106	1.0s	T	0.1s	—	1m4s	T
	35f	35	106	2.0s	T	0.2s	T	1m5s	F
	35g	35	106	1.0s	T	0.2s	—	1m7s	T
	35h	35	106	1.2s	T	0.2s	—	2.0s	—
	35i	35	106	1.2s	T	0.3s	T	1m6s	T
	35j	35	106	1.4s	T	0.2s	—	1m6s	F
RevLib	add16_174	49	65	3.1s	T	timeout		1m16s	T
	add32_183	97	129	25s	T	timeout		2m2s	T
	add64_184	193	257	2m8s	T	timeout		2.0s	—
	avg8_325	320	1,758	22m42s	T	timeout		2.2s	—
	bw_291	87	308	11s	T	15s	T	2m10s	T
	cycle10_293	39	79	0.5s	T	0.5s	T	1m10s	T
	e64-bdd_295	195	388	50s	T	timeout		1.5s	—
	ex5p_296	206	648	2m5s	T	1m36s	T	2.0s	—
	ham15_298	45	154	1.3s	T	0.9s	T	1m15s	T
	mod5adder_306	32	97	0.8s	T	1.1s	T	1m2s	T
	rd84_313	34	105	0.9s	T	1.5s	T	1m5s	T

circuit	#q	#G	AutoQ time	AutoQ bug?	Feynman time	Feynman bug?	Qcec time	Qcec bug?
hwb10	16	31,765	1m49s	T	timeout		50.6s	T
hwb11	15	87,790	4m31s	T	timeout		48.8s	T
hwb12	20	171,483	13m43s	T	timeout		1m30s	T
hwb8	12	6,447	16s	T	timeout		43.6s	T
qcla_adder_10	36	182	1.7s	T	1.0s	—	1m5s	F
qcla_mod_7	26	295	2.0s	T	1m28s	—	59.0s	F
70a	70	211	21s	T	1.3s	—	1m 42s	T
70b	70	211	15s	T	0.7s	T	1m41s	T
70c	70	211	10s	T	0.8s	—	1m37s	T
70d	70	211	timeout		0.9s	T	1m39s	T
70e	70	211	24s	T	0.8s	—	1m36s	T
70f	70	211	31s	T	0.8s	T	1m34s	F
70g	70	211	16m8s	T	1.2s	—	1m47s	T
70h	70	211	15s	T	1.3s	—	1m42s	T
70i	70	211	18s	T	1.0s	—	1m47s	T
70j	70	211	1m37s	T	1.2s	—	1m49s	T
urf1_149	9	11,555	40s	T	timeout		45.2s	T
urf2_152	8	5,031	14s	T	21m39s	T	36.4s	T
urf3_155	10	26,469	1m29s	T	timeout		45.1s	T
urf4_187	11	32,005	2m10s	T	timeout		46.6s	T
urf5_158	9	10,277	26s	T	timeout		38.2s	T
urf6_160	15	10,741	1m6s	T	timeout		48.4s	T
hwb6_301	46	160	2.5s	T	2.7s	T	1m13s	T
hwb7_302	73	282	10s	T	15s	T	1m39s	T
hwb8_303	112	450	34s	T	43s	T	2m25s	T
hwb9_304	170	700	1m48s	T	2m29s	T	2.0s	—

https://github.com/meamy/feynman
https://github.com/cda-tum/qcec

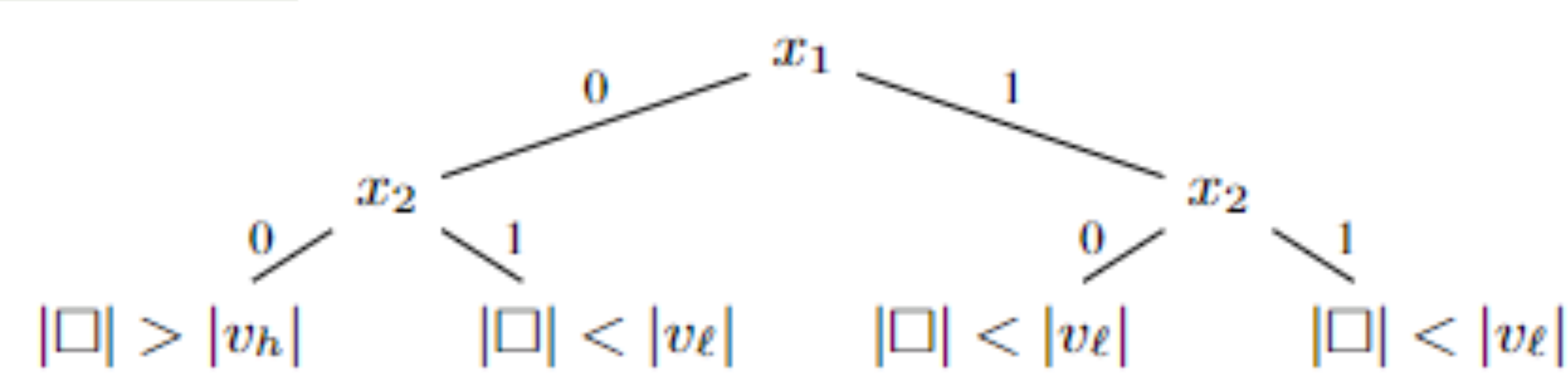
Extension: Symbolic Tree



A Symbolic Tree (State), s

The state after executing C, C(s)

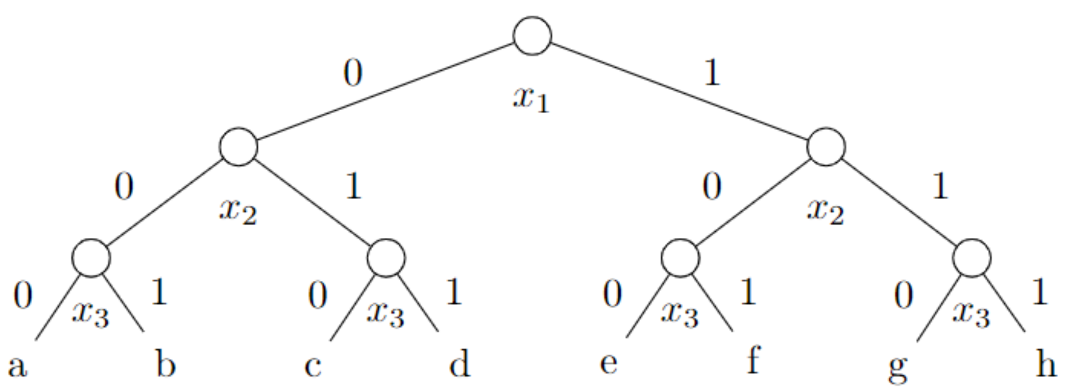
Predicate Tree as Specifications for **Relational Properties**



Yu-Fang Chen, Kai-Min Chung, Ondřej Lengál, Jyun-Ao Lin, Wei-Lun Tsai.
AutoQ: An Automata-based Quantum Circuit Verifier.
CAV 2023

Experiment – Symbolic

circuit	qubits	gates	property	result	time
H^2	1	2	$H^2 = I$	OK	0.22s
H^2 (bug)	1	2	$H^2 = I$	Bug	0.17s
BV(2)	2	6	ψ_{Im}	OK	0.11s
BV(2) (bug)	2	6	ψ_{Im}	Bug	0.15s
BV(100)	100	251	ψ_{Im}	OK	10.90s
BV(1,000)	1,000	2,500	ψ_{Im}	OK	198m28s
$\text{Grover}_{\text{All}}(3)$	9	64	P(Correct) > 0.9	OK	0.40s
$\text{Grover}_{\text{All}}(8)$	24	939	P(Correct) > 0.9	OK	3m18s
$\text{Grover}_{\text{All}}(9)$	27	1,492	P(Correct) > 0.9	OK	25m16s
$\text{Grover}_{\text{Single}}(3)$	6	54	P(Correct) > 0.9	OK	0.34s
$\text{Grover}_{\text{Single}}(16)$	32	28,159	P(Correct) > 0.9	OK	2m21s
$\text{Grover}_{\text{Single}}(18)$	36	63,537	P(Correct) > 0.9	OK	6m37s
$\text{Grover}_{\text{Single}}(20)$	40	141,527	P(Correct) > 0.9	OK	19m57s
$\text{Grover}_{\text{Iter}}(2)$	3	13	P(Correct) Increased	OK	0.40s
$\text{Grover}_{\text{Iter}}(18)$	36	157	P(Correct) Increased	OK	1.95s
$\text{Grover}_{\text{Iter}}(50)$	100	445	P(Correct) Increased	OK	47.76s
$\text{Grover}_{\text{Iter}}(75)$	150	671	P(Correct) Increased	OK	3m29s
$\text{Grover}_{\text{Iter}}(100)$	200	895	P(Correct) Increased	OK	10m53s



x ∈{a, b, c, …, h} is complex number and $|x|^2$ is the probability

Thank You

Summary:

- Interesting link between automata and quantum computing.
- So far only basic TA has been tried, there are many more possibilities.
- The main reason for efficiency is the compact structure of TA.