Automata-based Verification of Quantum Circuits

Ondřej Lengál

Brno University of Technology, Czech Republic

joint work with
Parosh Aziz Abdulla, Yu-Fang Chen, Yo-Ga Chen, Kai-Min Chung, Michal Hečko, Lukáš Holík, Min-Hsiu Hsieh, Wei-Jia Huang, Jyun-Ao Lin, Fang-Yi Lo, Ramanathan S. Thinniyam, Wei-Lun Tsai, Di-De Yen

One FLAT World Seminar (20 May 2026)

Why Quantum Computation?

Quantum computation:

- first proposed by Feynman (1982)
- promises to efficiently solve some problems we don't know how to efficiently solve classically

Why Quantum Computation?

Quantum computation:

- first proposed by Feynman (1982)
- promises to efficiently solve some problems we don't know how to efficiently solve classically
  - factoring (Shor, 1994): "exponential" $\rightsquigarrow$ polynomial
  - unstructured database search (Grover, 1996): $\mathcal{O}(2^n) \rightsquigarrow \mathcal{O}(\sqrt{2^n})$
  - Hamiltonian simulation (simulation of physical processes)

Why Quantum Computation?

Quantum computation:

- first proposed by Feynman (1982)
- promises to efficiently solve some problems we don't know how to efficiently solve classically
  - factoring (Shor, 1994): "exponential" $\rightsquigarrow$ polynomial
  - unstructured database search (Grover, 1996): $\mathcal{O}(2^n) \rightsquigarrow \mathcal{O}(\sqrt{2^n})$
  - Hamiltonian simulation (simulation of physical processes)
- real-world quantum computers are always 10 years away

Why Quantum Computation?

Quantum computation:

- first proposed by Feynman (1982)
- promises to efficiently solve some problems we don't know how to efficiently solve classically
  - factoring (Shor, 1994): "exponential" $\leadsto$ polynomial
  - unstructured database search (Grover, 1996): $\mathcal{O}(2^n) \leadsto \mathcal{O}(\sqrt{2^n})$
  - Hamiltonian simulation (simulation of physical processes)
- real-world quantum computers are always 10 years away
- $\leadsto$ we need to be prepared (computer-aided analysis)

Why Quantum Computation?

Quantum computation:

- first proposed by Feynman (1982)
- promises to efficiently solve some problems we don't know how to efficiently solve classically
  - factoring (Shor, 1994): "exponential" $\leadsto$ polynomial
  - unstructured database search (Grover, 1996): $\mathcal{O}(2^n) \leadsto \mathcal{O}(\sqrt{2^n})$
  - Hamiltonian simulation (simulation of physical processes)
- real-world quantum computers are always 10 years away
- $\leadsto$ we need to be prepared (computer-aided analysis)
- **FUN** and cool community!

Outline

Short Quantum Introduction

Bit vs. Qubit

A bit

$x = 0 \quad \text{or} \quad x = 1$

Bit vs. Qubit

A bit

$x = 0 \quad \text{or} \quad x = 1$

A qubit

$$x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$\alpha, \beta \in \mathbb{C}, \quad \text{s.t.} |\alpha|^2 + |\beta|^2 = 1$$

Bit vs. Qubit

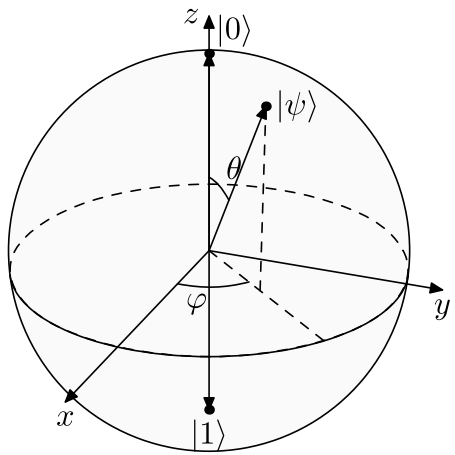
A bit

$x = 0 \quad \text{or} \quad x = 1$

A qubit

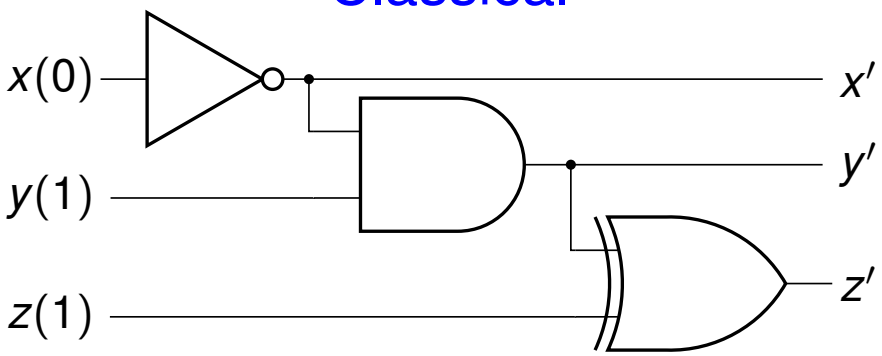
$$x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$\alpha, \beta \in \mathbb{C}, \quad \text{s.t.} |\alpha|^2 + |\beta|^2 = 1$$



Classical vs. Quantum Circuits — State

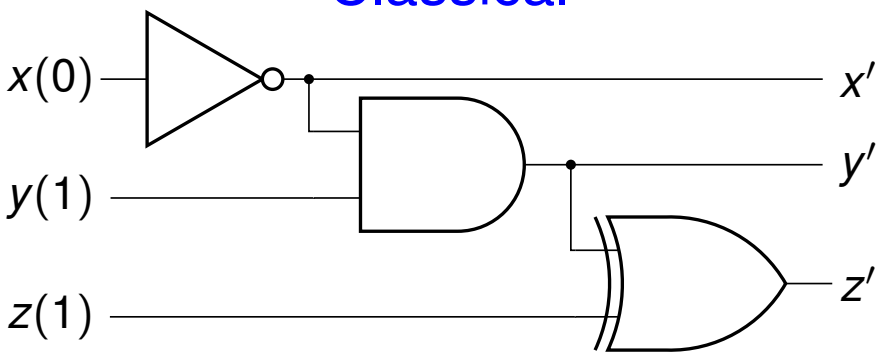
Classical



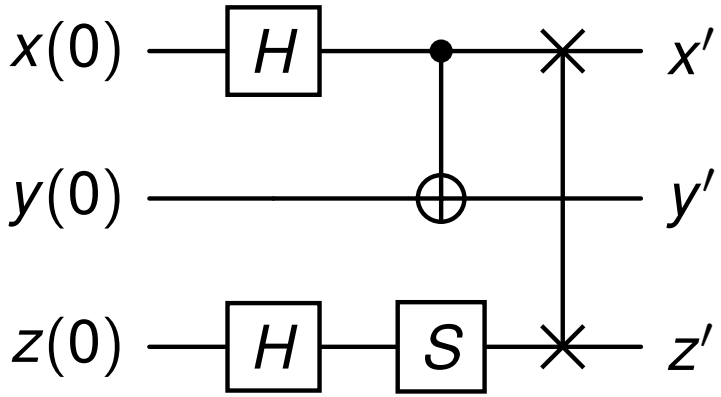
x'	y'	z'	χ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	**1**	**0**	**1**
1	1	1	0

Classical vs. Quantum Circuits — State

Classical



Quantum



x'	y'	z'	χ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	**1**	**0**	**1**
1	1	1	0

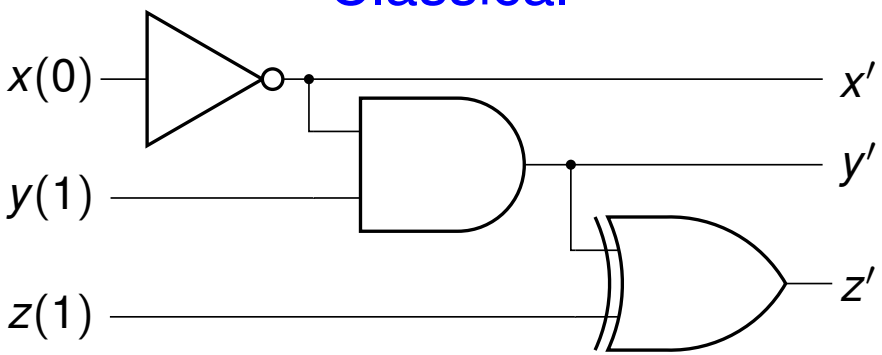
x'	y'	z'	amp
0	**0**	**0**	**½**
0	0	1	0
0	1	0	0
0	**1**	**1**	**½**
1	**0**	**0**	**½ i**
1	0	1	0
1	1	0	0
1	**1**	**1**	**½ i**

$amp(\vec{x}) \in \mathbb{C}$

$\Pr(\vec{x}) = |x|^2$

Classical vs. Quantum Circuits — Gates

Classical

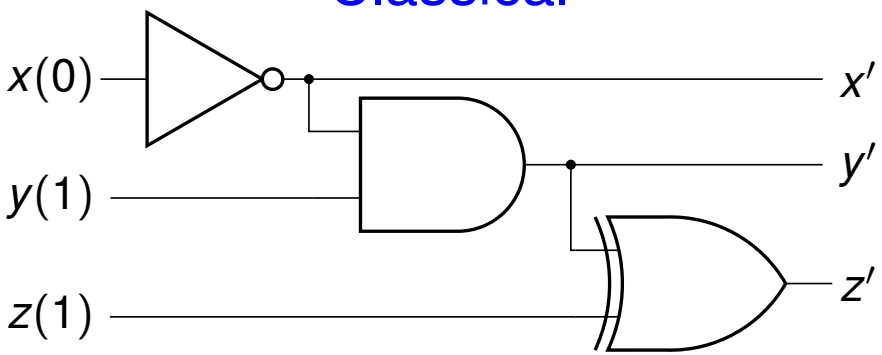


A gate is a truth table

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Classical vs. Quantum Circuits — Gates

Classical



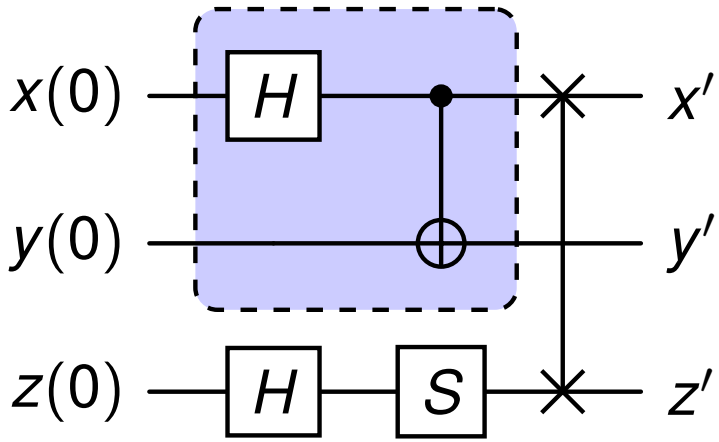
A gate is a truth table

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

unitary matrix:

- conjugate transpose $U^\dagger = U^{-1}$

Quantum

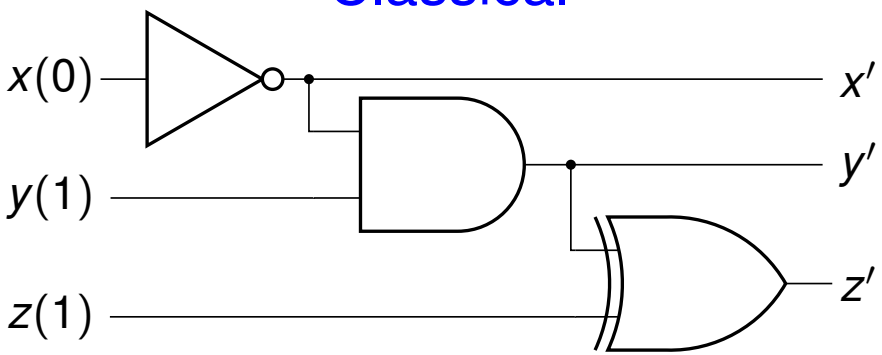


A gate is a unitary matrix

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}$$

Classical vs. Quantum Circuits — Gates

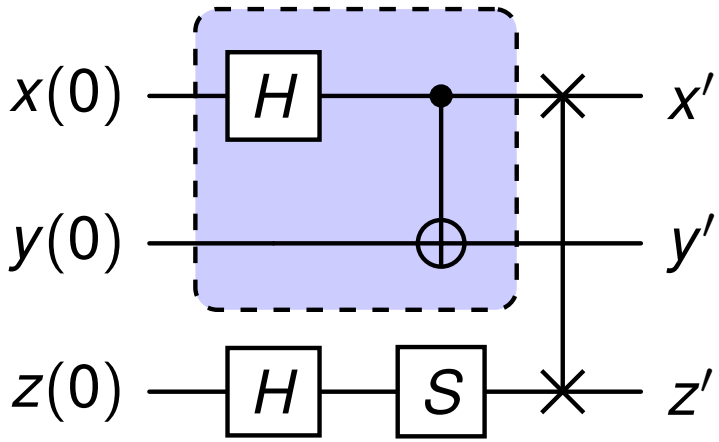
Classical



A gate is a truth table

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Quantum



A gate is a unitary matrix

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}$$

unitary matrix:

- conjugate transpose $U^\dagger = U^{-1}$
- $\rightsquigarrow$ reversibility, norm preservation, no-cloning theorem, ...

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\qquad H|0\rangle =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$
- $|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$
- $|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$ $\quad H|-\rangle =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$
- $|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$ $\quad H|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$
- $|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$ $\quad H|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - \frac{1}{2} \\ \frac{1}{2} + \frac{1}{2} \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$
- $|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$ $\quad H|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - \frac{1}{2} \\ \frac{1}{2} + \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} =$

Short Quantum Circuit Example — Hadamard gate

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$
- $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad H|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$
- $|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ $\quad H|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} + \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$
- $|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$ $\quad H|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - \frac{1}{2} \\ \frac{1}{2} + \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$

Verification of Quantum Circuits

Reasoning over Quantum Circuits

Reasoning over Quantum Circuits

Hard!

Reasoning over Quantum Circuits

Hard!

- exponential size of state representation
- inherently probabilistic (testing is hard!)
- need to deal with complex numbers

Reasoning over Quantum Circuits

Hard!

- exponential size of state representation
- inherently probabilistic (testing is hard!)
- need to deal with complex numbers

Main approaches:

1. state vector simulation (strong: #**P**-complete)
2. equivalence checking (**QMA**-complete)
   - **QMA** = *Quantum Merlin Arthur*; the so-called “quantum NP”
3. (pre/post-condition) verification

Verification of Classical Programs

Verification of classical programs:

- (pre/post-condition based, a.k.a. Floyd-Hoare style)

Verification of Classical Programs

Verification of classical programs:

- (pre/post-condition based, a.k.a. Floyd-Hoare style)

$$\underset{\text{precondition}}{\{Pre\}} \quad \underset{\text{statement}}{S} \quad \underset{\text{postcondition}}{\{Post\}}$$

- *Pre* and *Post* denote sets of program states

Verification of Classical Programs

Verification of classical programs:

- (pre/post-condition based, a.k.a. Floyd-Hoare style)

$$\underbrace{\{Pre\}}_{\textit{precondition}} \quad \underbrace{S}_{\textit{statement}} \quad \underbrace{\{Post\}}_{\textit{postcondition}}$$

- *Pre* and *Post* denote sets of program states

Meaning:

- If S is executed from a state from *Pre*
- and the execution of S terminates,
- then the program state after S terminates is in *Post*.

Verification of Quantum Circuits

Verification of quantum circuits:

Verification of Quantum Circuits

Verification of quantum circuits:

$$\underset{\textit{precondition}}{\{Pre\}} \quad \underset{\textit{circuit}}{C} \quad \underset{\textit{postcondition}}{\{Post\}}$$

- *Pre* and *Post* denote sets of **quantum** states

Verification of Quantum Circuits

Verification of quantum circuits:

$$\underset{\text{precondition}}{\{Pre\}} \quad \underset{\text{circuit}}{C} \quad \underset{\text{postcondition}}{\{Post\}}$$

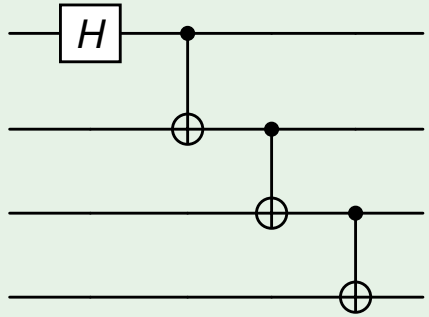
- *Pre* and *Post* denote sets of **quantum** states

Meaning:

- If *C* is executed from a quantum state from *Pre*
- then the quantum state after *C* terminates is in *Post*.
- (termination is implicit)

Verification of Quantum Circuits

Example (GHZ)



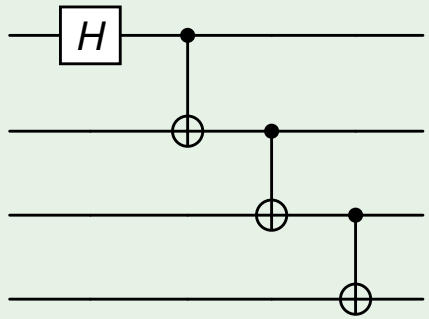
- GHZ (Greenberger–Horne–Zeilinger state)
  - generalization of Bell's state from 2 to arbitrary many qubits
  - maximally entangled qubits

$$|0000\rangle \xrightarrow{GHZ} \frac{|0000\rangle + |1111\rangle}{\sqrt{2}}$$

$$[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]^T \qquad [\tfrac{1}{\sqrt{2}},0,0,0,0,0,0,0,0,0,0,0,0,0,0,\tfrac{1}{\sqrt{2}}]^T$$

Verification of Quantum Circuits

Example (GHZ)



- GHZ (Greenberger–Horne–Zeilinger state)
  - generalization of Bell's state from 2 to arbitrary many qubits
  - maximally entangled qubits

$$|0000\rangle \xrightarrow{GHZ} \frac{|0000\rangle+|1111\rangle}{\sqrt{2}}$$

$$[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]^T \qquad [\tfrac{1}{\sqrt{2}},0,0,0,0,0,0,0,0,0,0,0,0,0,0,\tfrac{1}{\sqrt{2}}]^T$$

$$|0001\rangle \xrightarrow{GHZ} \frac{|0001\rangle+|1110\rangle}{\sqrt{2}}$$

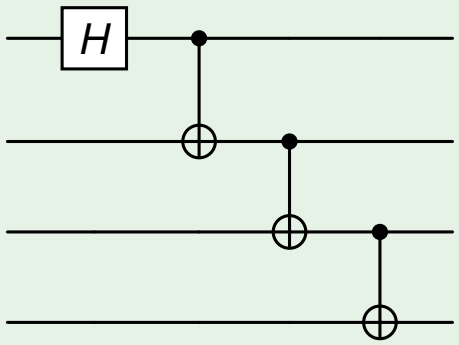
$$[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]^T \qquad [0,\tfrac{1}{\sqrt{2}},0,0,0,0,0,0,0,0,0,0,0,0,\tfrac{1}{\sqrt{2}},0]^T$$

$$\vdots$$

Verification of Quantum Circuits

Example (GHZ)

$\{|w\rangle : w \in \{0,1\}^4\}$



$\{\frac{1}{\sqrt{2}}|0 b_2 b_3 b_4\rangle \pm \frac{1}{\sqrt{2}}|1 \bar{b}_2 \bar{b}_3 \bar{b}_4\rangle : b_2 b_3 b_4 \in \{0,1\}^3\}$

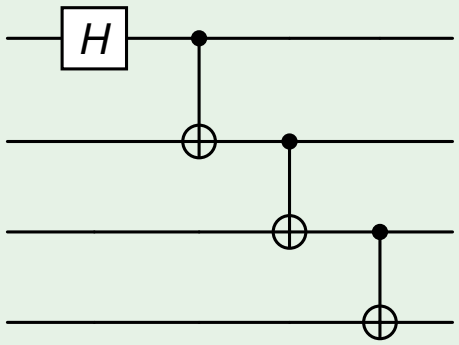
Pre Circuit *Post*

$$Pre = \{|0000\rangle, |0001\rangle, \ldots, |1111\rangle\}$$

Verification of Quantum Circuits

Example (GHZ)

$\{|w\rangle : w \in \{0,1\}^4\}$



$\{\frac{1}{\sqrt{2}}|0 b_2 b_3 b_4\rangle \pm \frac{1}{\sqrt{2}}|1 \bar{b}_2 \bar{b}_3 \bar{b}_4\rangle : b_2 b_3 b_4 \in \{0,1\}^3\}$

Pre Circuit *Post*

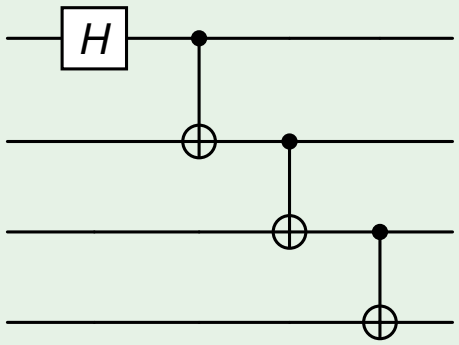
$$Pre = \{|0000\rangle, |0001\rangle, \ldots, |1111\rangle\}$$

How to efficiently represent sets of quantum states *Pre* and *Post*?

Verification of Quantum Circuits

Example (GHZ)

$\{|w\rangle : w \in \{0,1\}^4\}$



$\{\frac{1}{\sqrt{2}}|0 b_2 b_3 b_4\rangle \pm \frac{1}{\sqrt{2}}|1 \bar{b}_2 \bar{b}_3 \bar{b}_4\rangle : b_2 b_3 b_4 \in \{0,1\}^3\}$

Pre Circuit *Post*

$$Pre = \{|0000\rangle, |0001\rangle, \ldots, |1111\rangle\}$$

How to efficiently represent sets of quantum states *Pre* and *Post*?

- naively $\rightsquigarrow$ double exponential size

Quantum States are Trees

Quantum States are Trees

. . . and quantum gates are tree operations

Quantum States are Trees

x	y	z	amp
0	**0**	**0**	**½**
0	0	1	0
0	1	0	0
0	**1**	**1**	**½**
1	**0**	**0**	**½** i
1	0	1	0
1	1	0	0
1	**1**	**1**	**½** i

Quantum States are Trees

x	y	z	*amp*
0	**0**	**0**	**½**
0	0	1	0
0	1	0	0
0	**1**	**1**	**½**
1	**0**	**0**	**½ *i***
1	0	1	0
1	1	0	0
1	**1**	**1**	**½ *i***

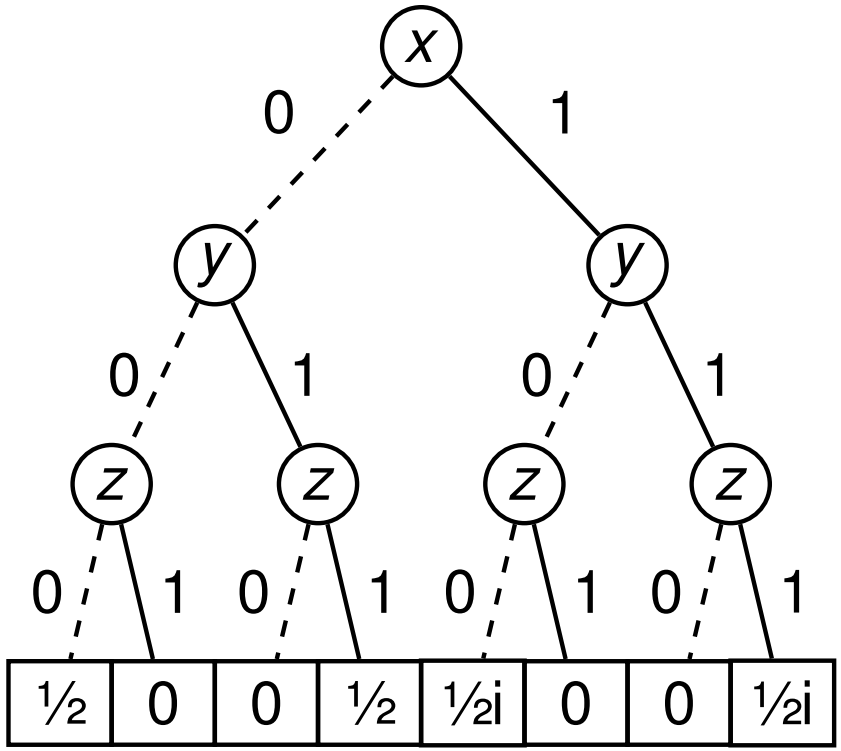
$\Rightarrow$

½	0	0	½	½i	0	0	½i

Quantum States are Trees

x	y	z	*amp*
0	**0**	**0**	**½**
0	0	1	0
0	1	0	0
0	**1**	**1**	**½**
1	**0**	**0**	**½ *i***
1	0	1	0
1	1	0	0
1	**1**	**1**	**½ *i***

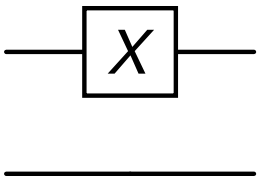
$\Rightarrow$



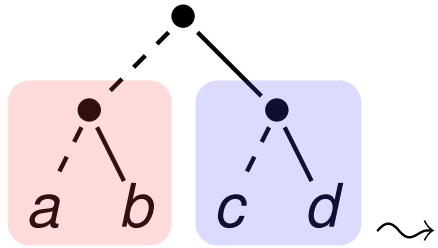
- perfect tree of height n (the number of qubits) $\rightsquigarrow 2^n$ leaves

Quantum Gates are Tree Operations

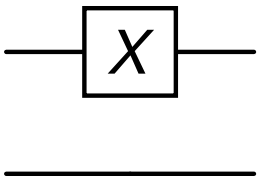
Quantum Gates are Tree Operations



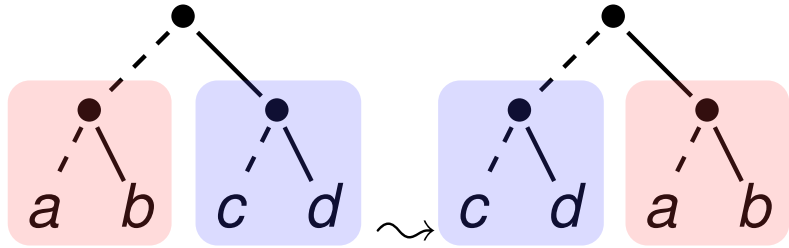
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$



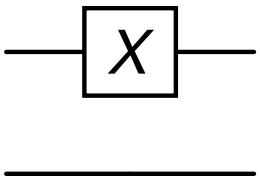
Quantum Gates are Tree Operations



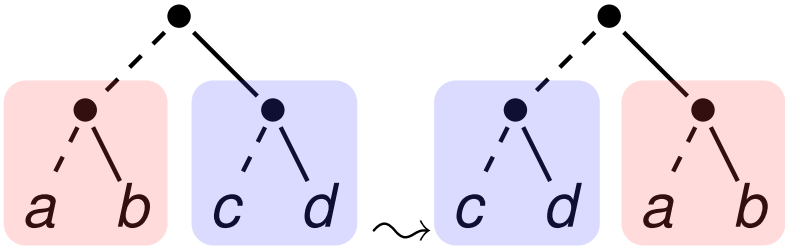
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$

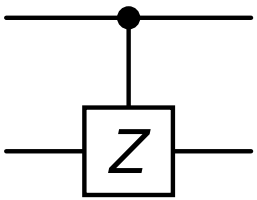


Quantum Gates are Tree Operations

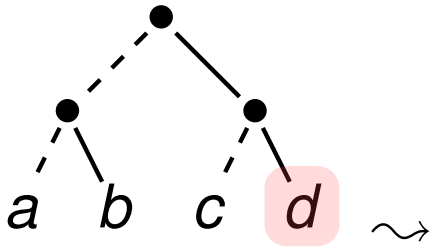


$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$

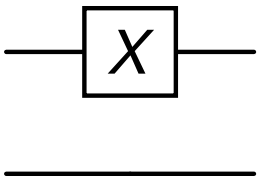




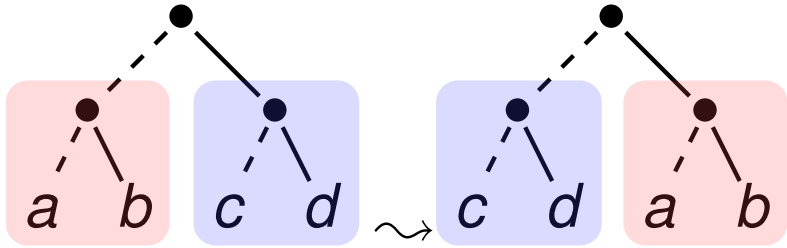
$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

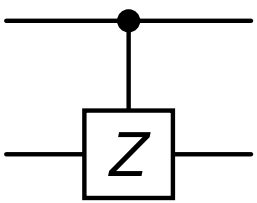


Quantum Gates are Tree Operations

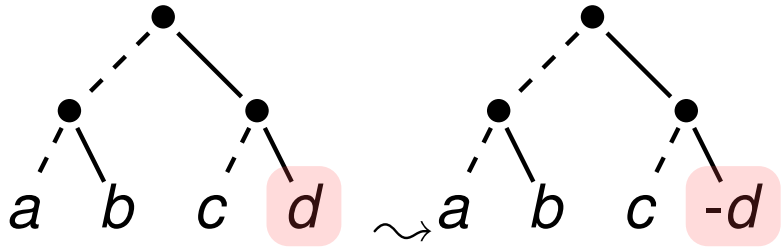


$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$

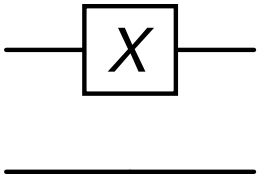




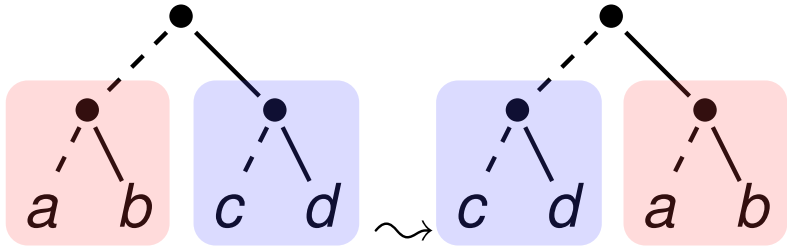
$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

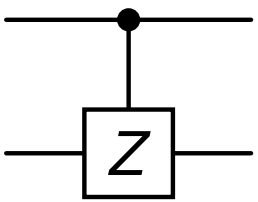


Quantum Gates are Tree Operations

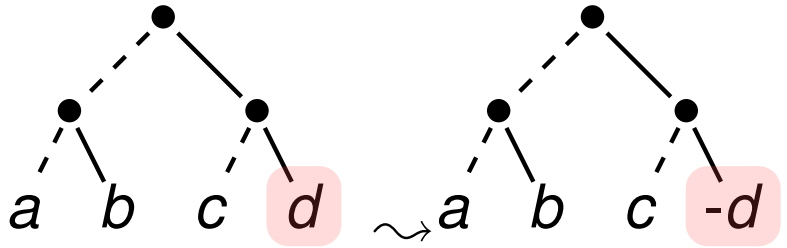


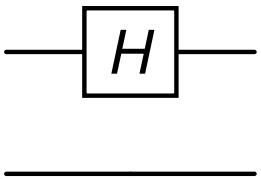
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$



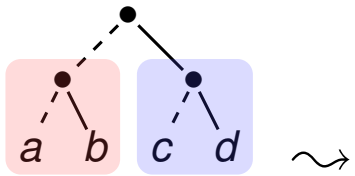


$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

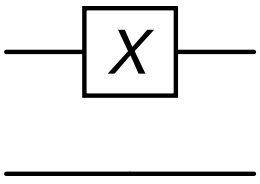




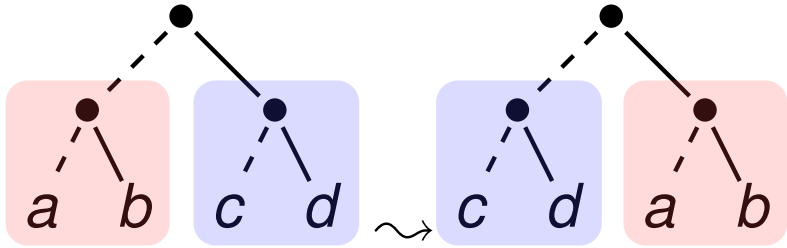
$$H_1 = \overbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}}^{H} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$

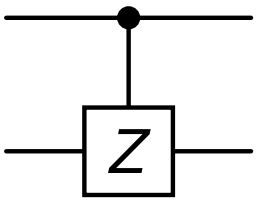


Quantum Gates are Tree Operations

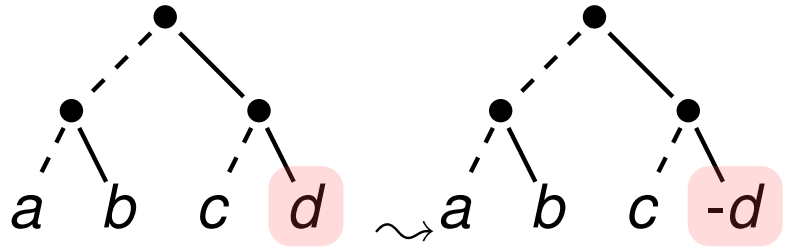


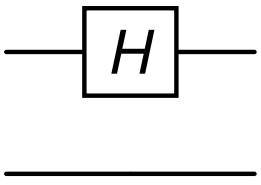
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$



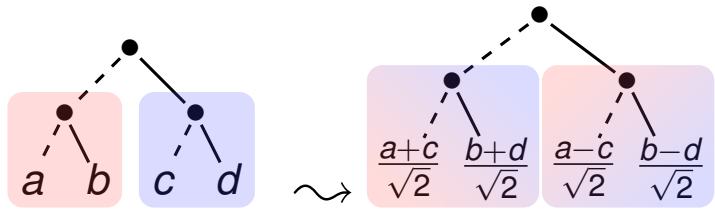


$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$





$$H_1 = \overbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}}^{H} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$



Hadamard gate

Sets of Quantum States are Sets of Trees

- How to efficiently represent sets of trees?

Sets of Quantum States are Sets of Trees

- How to efficiently represent sets of trees?

Tree automata!

Sets of Quantum States are Sets of Trees

- How to efficiently represent sets of trees?

Tree automata!

- tree automata
  - finite-state automata representing sets of finite trees
  - extension of standard finite automata for regular languages

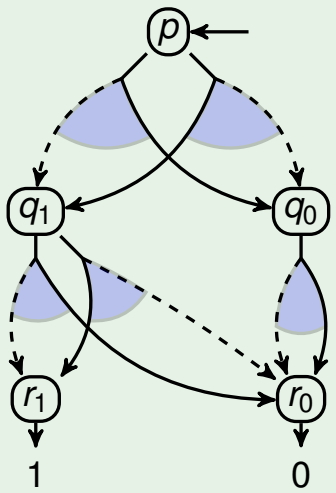
Sets of Quantum States are Sets of Trees

- How to efficiently represent sets of trees?

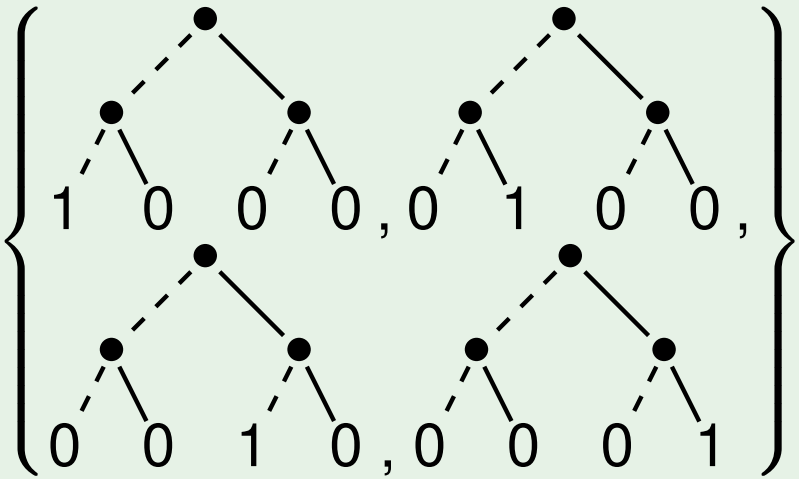
Tree automata!

- tree automata
  - finite-state automata representing sets of finite trees
  - extension of standard finite automata for regular languages

Example



represents the set



Representing *Pre* and *Post* with Tree Automata

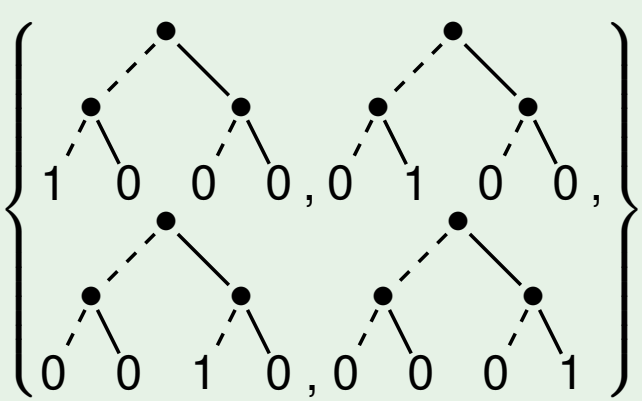
$$\underset{\text{precondition}}{\{\mathcal{A}_{Pre}\}} \quad \underset{\text{circuit}}{C} \quad \underset{\text{postcondition}}{\{\mathcal{A}_{Post}\}}$$

Representing *Pre* and *Post* with Tree Automata

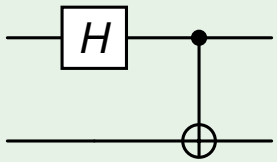
$$\{\mathcal{A}_{Pre}\} \; \underset{\text{circuit}}{C} \; \{\mathcal{A}_{Post}\}$$

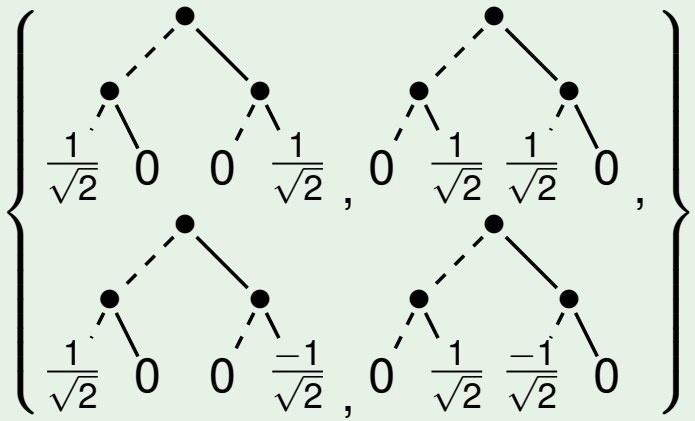
precondition — *postcondition*

Example (GHZ)



$\mathcal{L}(\mathcal{A}_{Pre})$





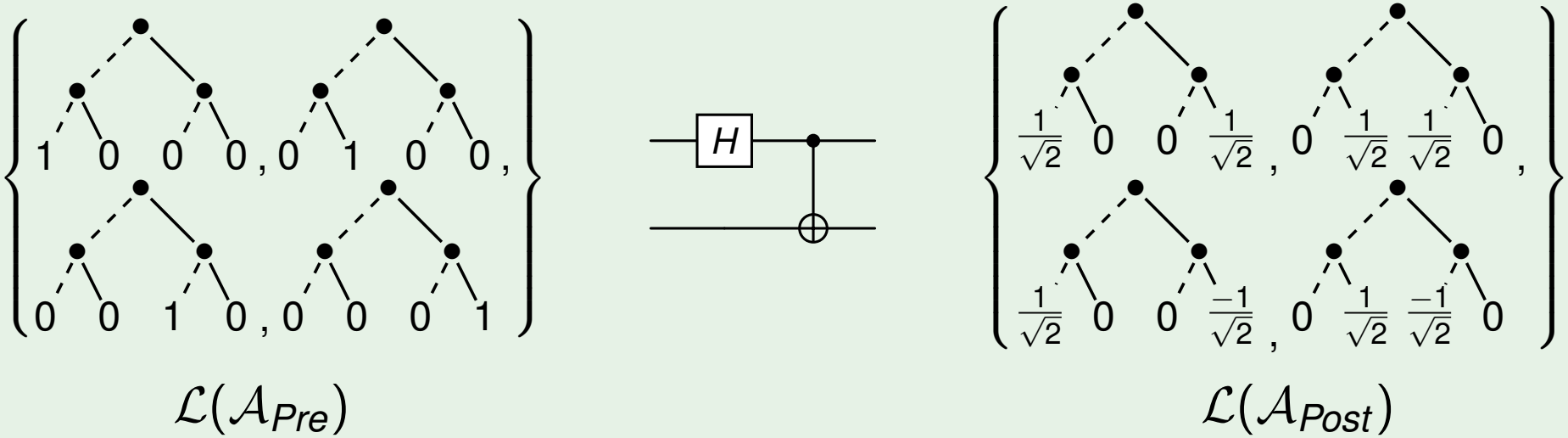
$\mathcal{L}(\mathcal{A}_{Post})$

Representing *Pre* and *Post* with Tree Automata

$$\{\mathcal{A}_{Pre}\} \quad \underset{\text{circuit}}{C} \quad \{\mathcal{A}_{Post}\}$$

precondition *postcondition*

Example (GHZ)



$\mathcal{L}(\mathcal{A}_{Pre})$ $\mathcal{L}(\mathcal{A}_{Post})$

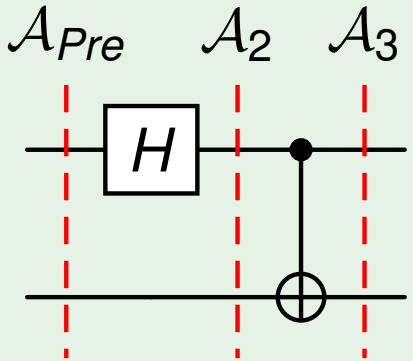
- $\mathcal{A}$'s size can be small
  - e.g., $\mathcal{A}$ for $\{|w\rangle : w \in \{0,1\}^n\}$ needs $\mathcal{O}(n)$ states/transitions

Verification with Tree Automata

$$\underset{\text{precondition}}{\{\mathcal{A}_{Pre}\}} \quad \underset{\text{circuit}}{C} \quad \underset{\text{postcondition}}{\{\mathcal{A}_{Post}\}}$$

- Run C with $\mathcal{A}_{Pre}$:

Example



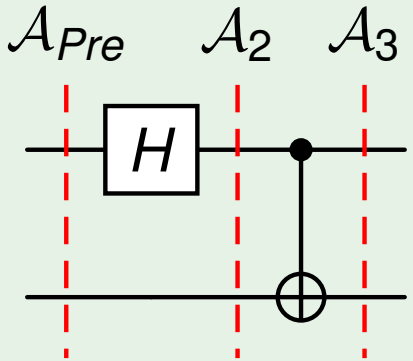
Verification with Tree Automata

$$\{\mathcal{A}_{Pre}\}\ \underset{\text{circuit}}{C}\ \{\mathcal{A}_{Post}\}$$

precondition *postcondition*

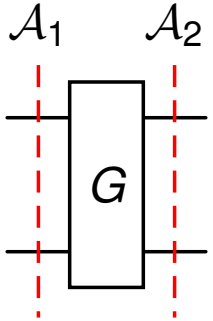
- Run C with $\mathcal{A}_{Pre}$:

Example



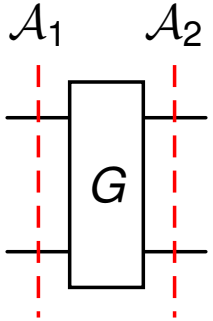
- . . . and test $\mathcal{L}(\mathcal{A}_3) \subseteq \mathcal{L}(\mathcal{A}_{Post})$
  - tree automata inclusion is decidable (**EXPTIME**-complete)

Abstract Transformers for Quantum Gates



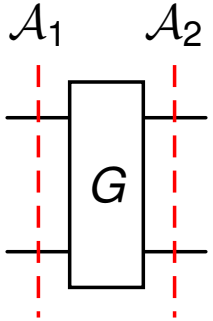
- How to compute $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
  - naively (i.e., one tree by one) — doesn't scale

Abstract Transformers for Quantum Gates



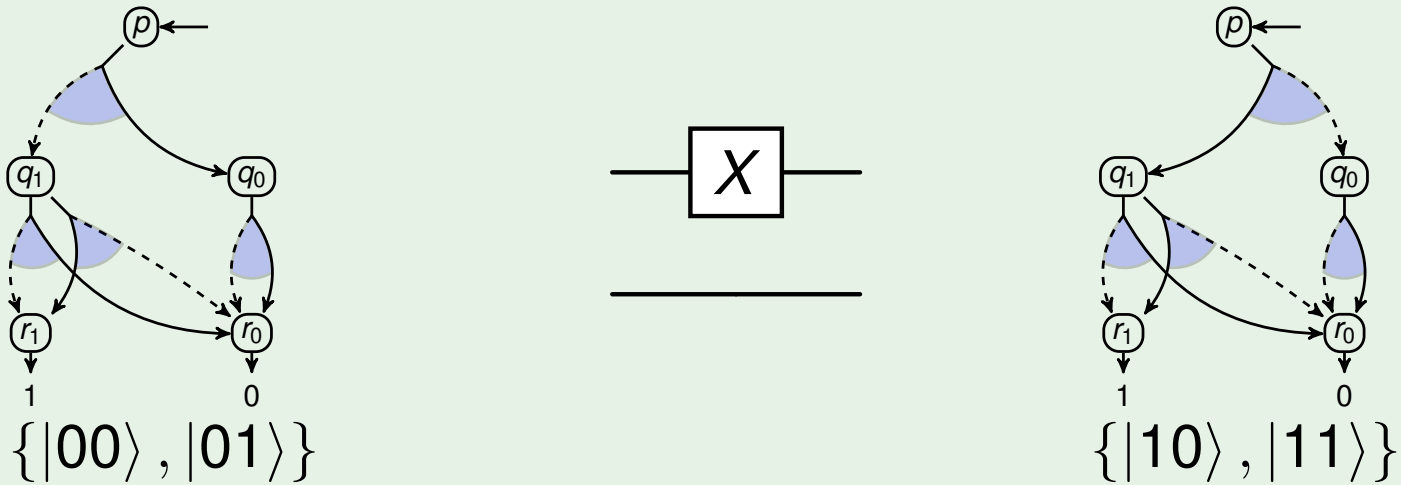
- How to compute $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
  - naively (i.e., one tree by one) — doesn't scale
- $\leadsto$ abstract transformers
  - specialized automata operations for concrete gates

Abstract Transformers for Quantum Gates

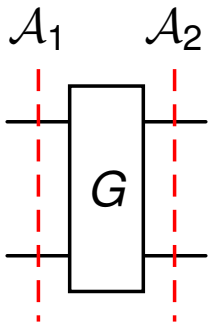


- How to compute $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
  - naively (i.e., one tree by one) — doesn't scale
- $\rightsquigarrow$ abstract transformers
  - specialized automata operations for concrete gates

Example

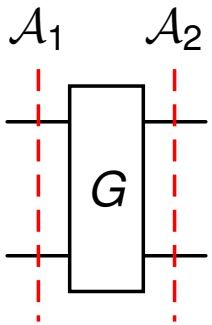


Abstract Transformers for Quantum Gates

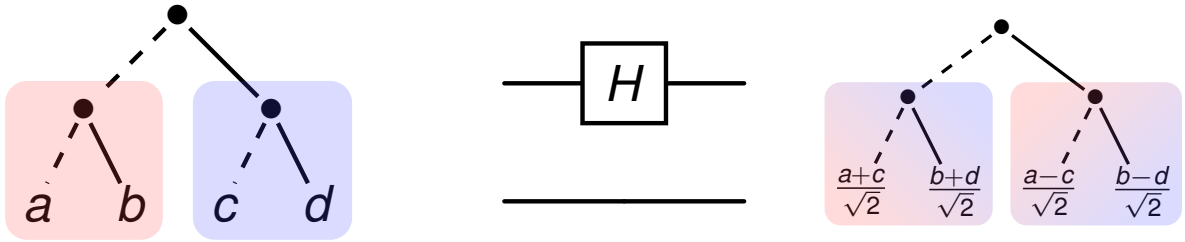


- Supported gate types:
  - (anti-)diagonal: X, Y, Z, S, T, R_z, controls (*CNOT*, *CZ*, *Toffoli*, ...)
    - simple manipulation with automaton: $\mathcal{O}(|\mathcal{A}_1|)$

Abstract Transformers for Quantum Gates



- Supported gate types:
  - (anti-)diagonal: X, Y, Z, S, T, R_z, controls (*CNOT*, *CZ*, *Toffoli*, ...)
    - simple manipulation with automaton: $\mathcal{O}(|\mathcal{A}_1|)$
  - general: $H, R_x, R_y, \ldots$
    - need to synchronize subtrees of the same tree



    - qubit reorder $\rightarrow$ leaf operation $\rightarrow$ qubit reorder: $\mathcal{O}(2^{|\mathcal{A}_1|})$

Quantum Circuit Verification Algorithm

$$\{\mathcal{A}_{Pre}\}\ \underset{circuit}{C}\ \{\mathcal{A}_{Post}\}$$

precondition *postcondition*

- Algorithm:
  1. Start with $\mathcal{A}_{Pre}$.

Quantum Circuit Verification Algorithm

$$\underbrace{\{\mathcal{A}_{Pre}\}}_{\text{precondition}} \quad \underbrace{C}_{\text{circuit}} \quad \underbrace{\{\mathcal{A}_{Post}\}}_{\text{postcondition}}$$

- Algorithm:
  1. Start with $\mathcal{A}_{Pre}$.
  2. Run C on $\mathcal{A}_{Pre}$ using abstract transformers, obtaining $\mathcal{A}_C$.

Quantum Circuit Verification Algorithm

$$\underbrace{\{\mathcal{A}_{Pre}\}}_{\text{precondition}} \quad \underbrace{C}_{\text{circuit}} \quad \underbrace{\{\mathcal{A}_{Post}\}}_{\text{postcondition}}$$

- Algorithm:
  1. Start with $\mathcal{A}_{Pre}$.
  2. Run C on $\mathcal{A}_{Pre}$ using abstract transformers, obtaining $\mathcal{A}_C$.
  3. Test $\mathcal{L}(\mathcal{A}_C) \subseteq \mathcal{L}(\mathcal{A}_{Post})$.

Quantum Circuit Verification Algorithm

$$\{\mathcal{A}_{Pre}\}\ \underbrace{C}_{circuit}\ \{\mathcal{A}_{Post}\}$$

precondition *postcondition*

- Algorithm:
  1. Start with $\mathcal{A}_{Pre}$.
  2. Run C on $\mathcal{A}_{Pre}$ using abstract transformers, obtaining $\mathcal{A}_C$.
  3. Test $\mathcal{L}(\mathcal{A}_C) \subseteq \mathcal{L}(\mathcal{A}_{Post})$.
- Implemented in AutoQ
- Used to verify/find bugs in a number of quantum circuits:
  - Bernstein-Vazirani, Grover (Single/All), MCToffoli,
  - RevLib, Random, FeynmanBench, ...

Quantum Circuit Verification Algorithm

$$\underbrace{\{\mathcal{A}_{Pre}\}}_{\text{precondition}} \quad \underbrace{C}_{\text{circuit}} \quad \underbrace{\{\mathcal{A}_{Post}\}}_{\text{postcondition}}$$

- Algorithm:
  1. Start with $\mathcal{A}_{Pre}$.
  2. Run C on $\mathcal{A}_{Pre}$ using abstract transformers, obtaining $\mathcal{A}_C$.
  3. Test $\mathcal{L}(\mathcal{A}_C) \subseteq \mathcal{L}(\mathcal{A}_{Post})$.
- Implemented in AutoQ
- Used to verify/find bugs in a number of quantum circuits:
  - Bernstein-Vazirani, Grover (Single/All), MCToffoli,
  - RevLib, Random, FeynmanBench, ...
- Scales to up to 40 qubits / 140k gates.
- Found a confirmed bug in QCEC (SOTA equivalence checker).

[Chen, Chung, Lengál, Lin, Tsai, Yen. An Automata-Based Framework for Verification and Bug Hunting in Quantum Circuits. PLDI'23/CACM RH]

New Automata Models

New Automata Models

- Level-Synchronized Tree Automata
  - synchronization across different subtrees

New Automata Models

- Level-Synchronized Tree Automata
  - synchronization across different subtrees
- Synchronized Weighted Tree Automata
  - + weights and alternation

New Automata Models

- Level-Synchronized Tree Automata
  - synchronization across different subtrees
- Synchronized Weighted Tree Automata
  - + weights and alternation
- Weighted Tree Transducers
  - corresponding transducer model

New Automata Models

Problems with standard TAs:

- time complexity of some gates is $\mathcal{O}(2^{|\mathcal{A}|})$

New Automata Models

Problems with standard TAs:

- time complexity of some gates is $\mathcal{O}(2^{|\mathcal{A}|})$
- don't support parameterized verification
  - e.g., cannot express "all perfect binary trees"

New Automata Models

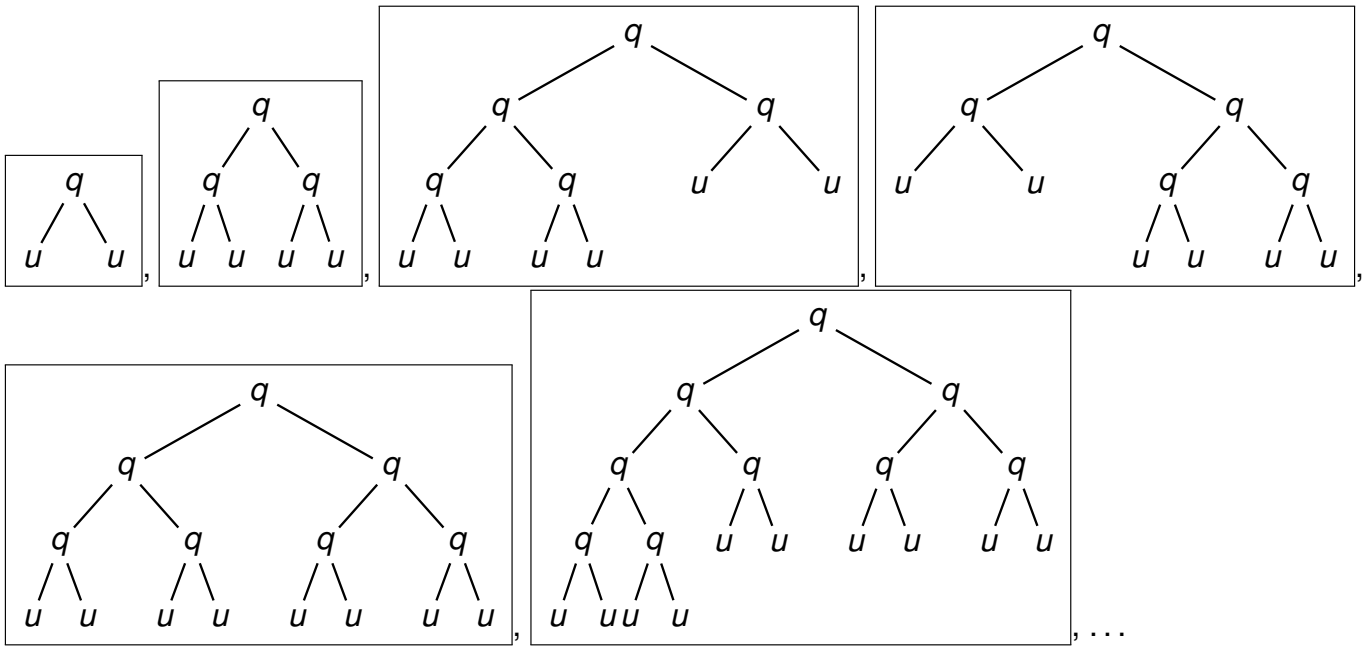
Problems with standard TAs:

- time complexity of some gates is $\mathcal{O}(2^{|\mathcal{A}|})$
- don't support parameterized verification
  - e.g., cannot express "all perfect binary trees"

$$q \to (q, q)$$

$$q \to (u, u)$$

q is root, u is leaf



Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata (LSTAs)

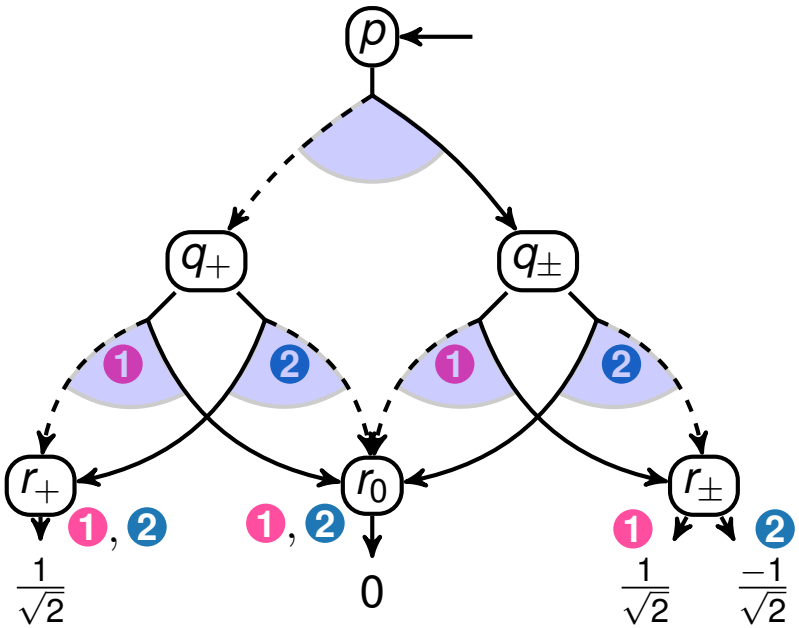
Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata (LSTAs)

- allow synchronization across subtrees rooted at the same level

$q_+ \xrightarrow{1} (r_+, r_0)$ $\quad$ $q_+ \xrightarrow{2} (r_0, r_+)$

$q_\pm \xrightarrow{1} (r_0, r_\pm)$ $\quad$ $q_\pm \xrightarrow{2} (r_\pm, r_0)$



Level-Synchronized Tree Automata (LSTAs)

No synchronization (standard TAs)

$$q \to (q, q)$$

$$q \to (u, u)$$

q is root, u is leaf

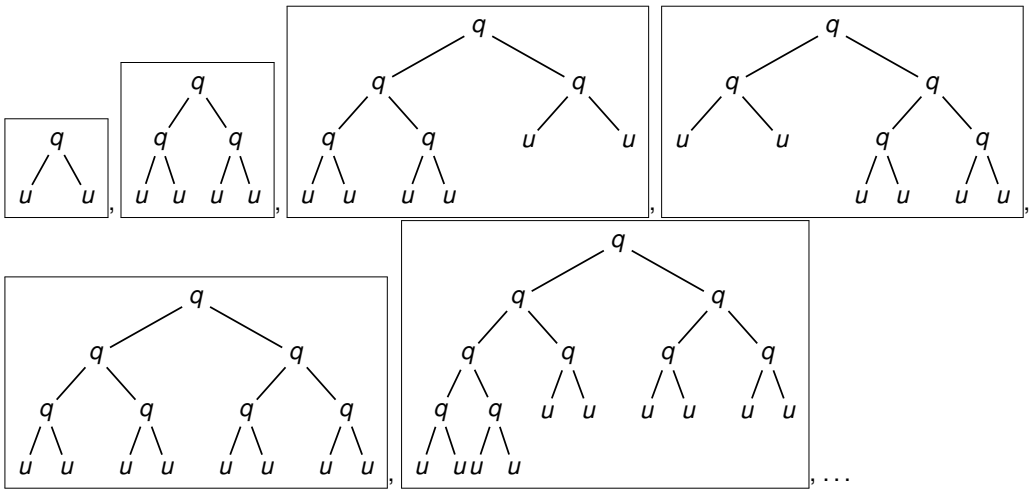
Level-Synchronized Tree Automata (LSTAs)

No synchronization (standard TAs)

$$q \to (q, q)$$

$$q \to (u, u)$$

q is root, u is leaf



Level-Synchronized Tree Automata (LSTAs)

No synchronization (standard TAs)

$$q \to (q, q)$$

$$q \to (u, u)$$

q is root, u is leaf

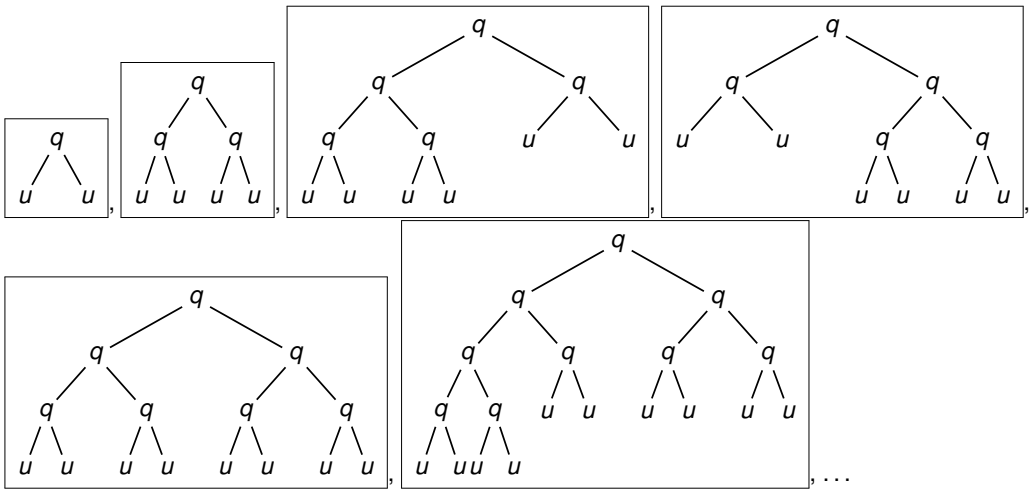
With synchronization

$$q \xrightarrow{1} (q, q)$$

$$q \xrightarrow{2} (u, u)$$

q is root, u is leaf

On every level, transitions with the same colour need to be taken!



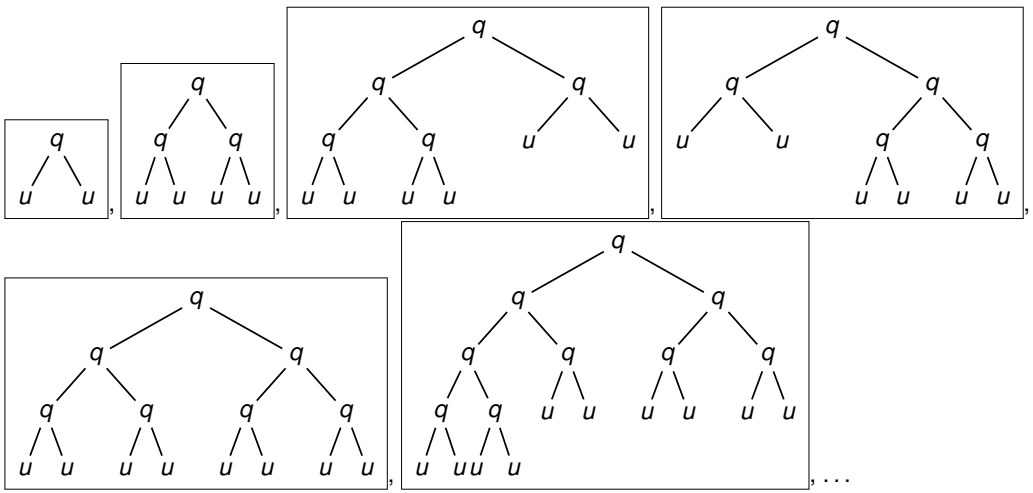
Level-Synchronized Tree Automata (LSTAs)

No synchronization (standard TAs)

$q \to (q, q)$

$q \to (u, u)$

q is root, u is leaf



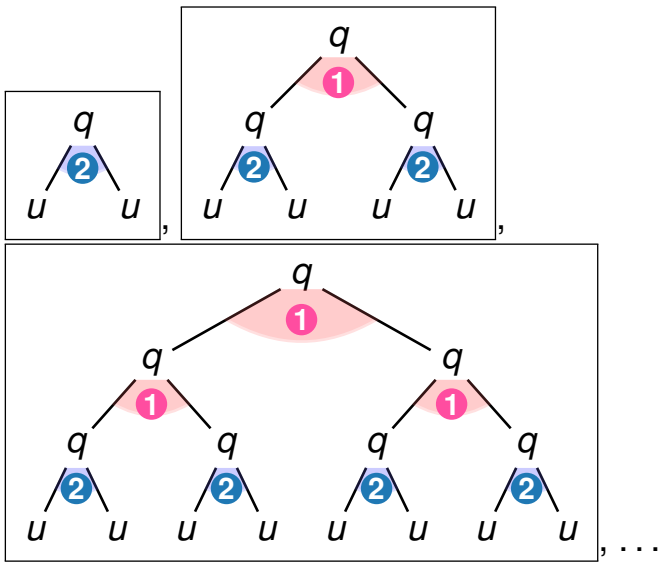
With synchronization

$q \xrightarrow{1} (q, q)$

$q \xrightarrow{2} (u, u)$

q is root, u is leaf

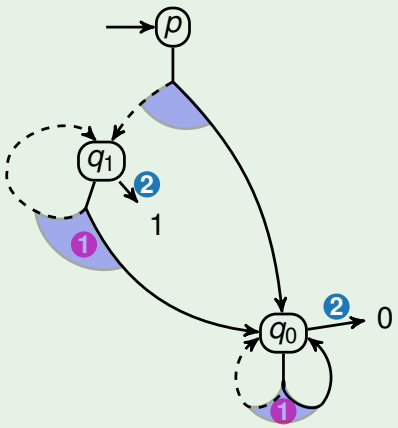
On every level, transitions with the same colour need to be taken!

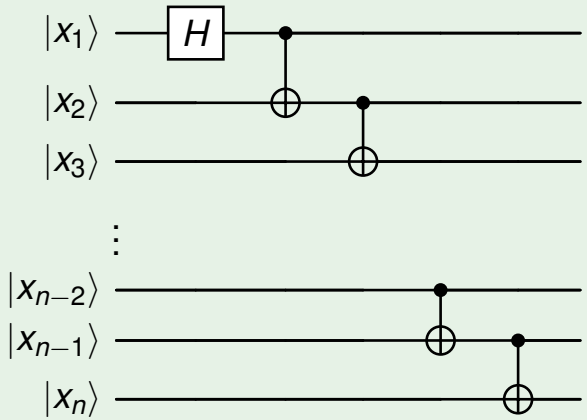


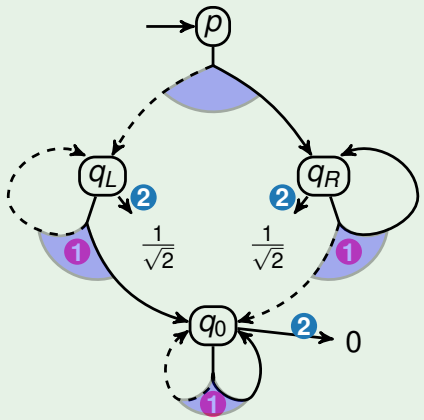
Level-Synchronized Tree Automata (LSTAs)

Enable basic parameterized verification

Example (GHZ)







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

$\{\frac{1}{\sqrt{2}}|0^n\rangle + \frac{1}{\sqrt{2}}|1^n\rangle : n \geq 1\}$

Level-Synchronized Tree Automata (LSTAs)

Provide better scalability

- cost of operations
  - (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
  - general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)

Level-Synchronized Tree Automata (LSTAs)

Provide better scalability

- cost of operations
  - (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
  - general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)
- language operations:
  - emptiness: **PSPACE**-complete
  - inclusion: **PSPACE**-hard, in **EXPSPACE**

Level-Synchronized Tree Automata (LSTAs)

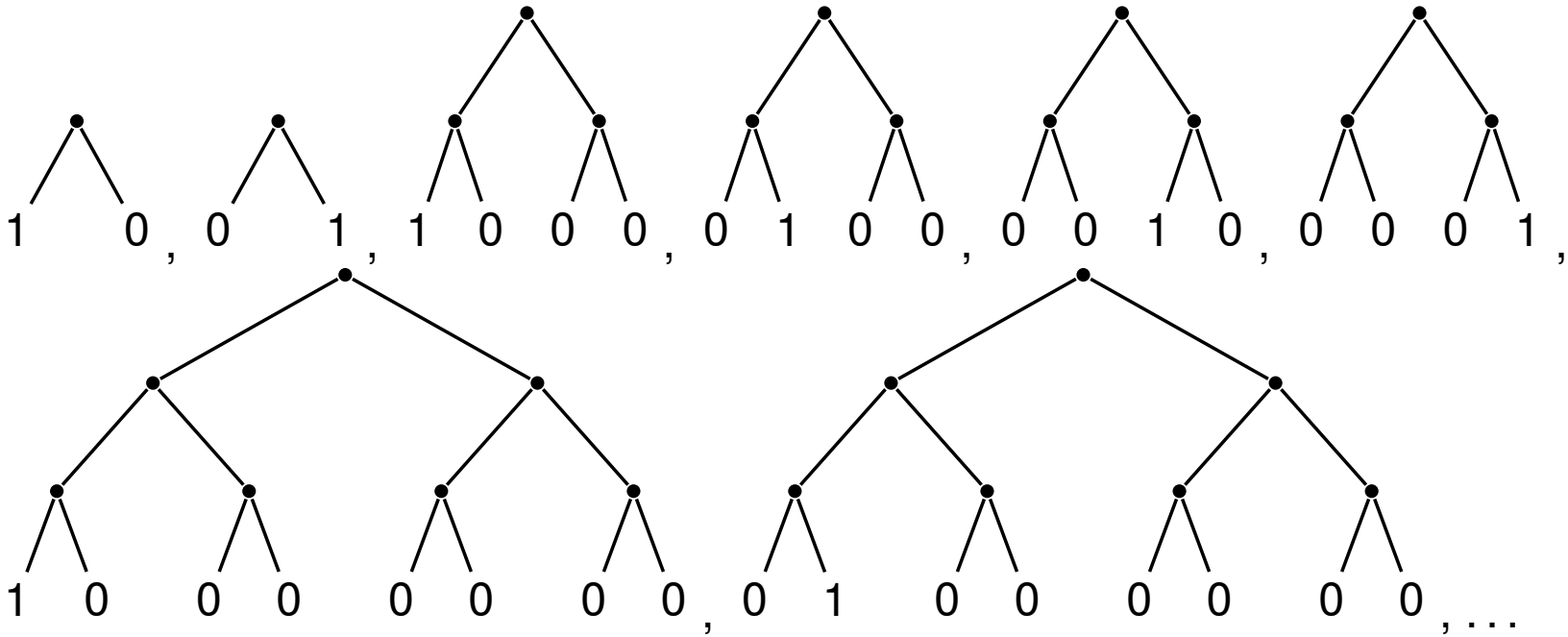
Provide better scalability

- cost of operations
  - (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
  - general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)
- language operations:
  - emptiness: **PSPACE**-complete
  - inclusion: **PSPACE**-hard, in **EXPSPACE**
- real-world speed-ups:
  - 4 min 45 s $\rightsquigarrow$ 0.0 s (BV-All, 27 qbits)
  - $>$5 min $\rightsquigarrow$ 0.6 s (GHZ-All, 128 qbits)
  - $>$5 min $\rightsquigarrow$ 39.3 s (Grover-All, 30 qbits)
  - $>$5 min $\rightsquigarrow$ 17.1 s (Grover-Iter, 200 qbits)

[Abdulla, Chen, Chen, Holík, Lengál, Lin, Lo, Tsai. Verifying Quantum Circuits with Level-Synchronized Tree Automata. POPL'25.]

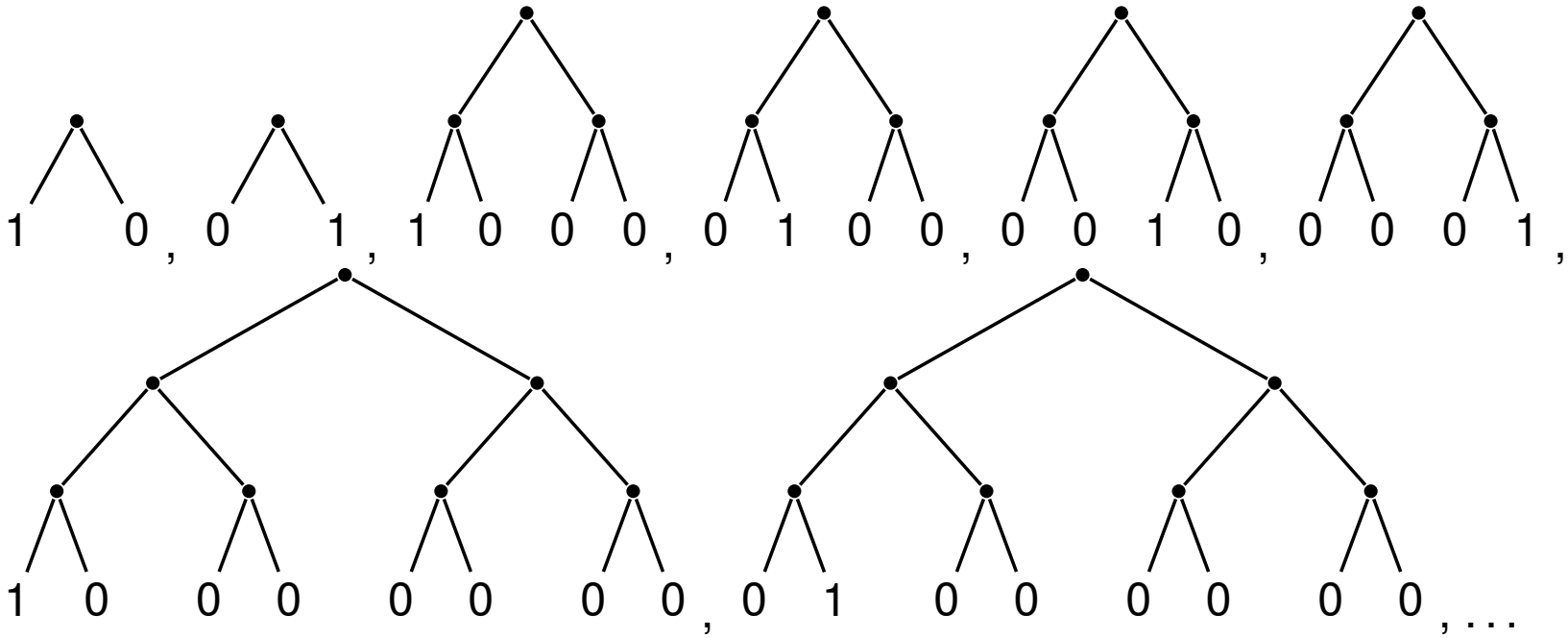
Level-Synchronized Tree Automata (LSTAs)

- LSTAs can express all computational basis states

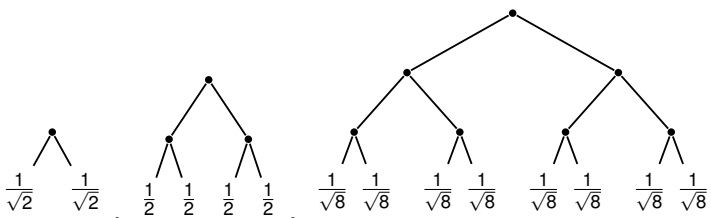


Level-Synchronized Tree Automata (LSTAs)

- LSTAs can express all computational basis states



- but still cannot express all uniform superposition states
  - ∞-many amplitudes
  - important for parameterized verification!



Parameterized Quantum Circuits

- Circuits are parameterized by their size

Parameterized Quantum Circuits

- Circuits are parameterized by their size
- Many standard quantum circuits are parameterized
  - generalized GHZ
  - Bernstein-Vazirani
  - Grover's search
  - error-correction
  - arithmetic circuits
  - quantum counting
  - quantum phase estimation
  - quantum Fourier transform,
  - Shor's algorithm, ...

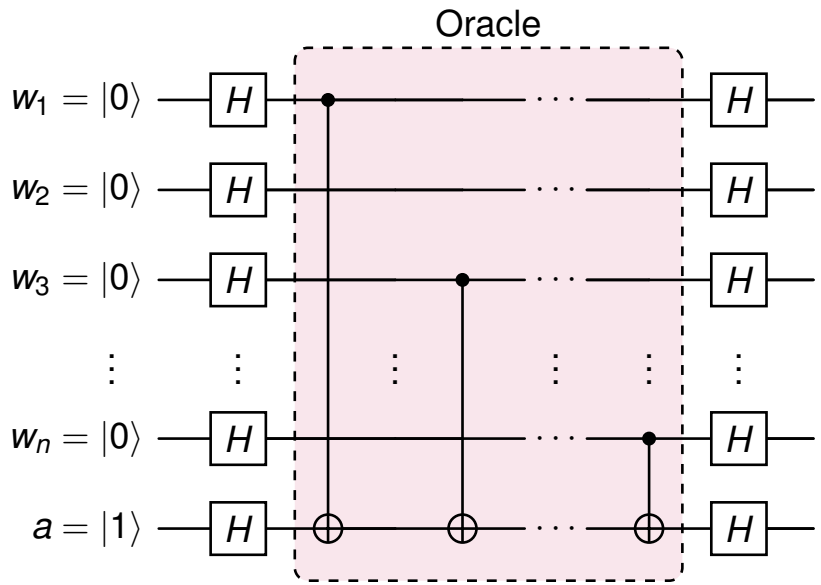


Figure: Bernstein-Vazirani for the secret $(10)^*(1+\varepsilon)$

Synchronized Weighted Tree Automata (SWTAs)

Synchronized Weighted Tree Automata
LSTAs + weights + alternation

[1] we ignore internal labels and only consider the structure of trees and leaf values

Synchronized Weighted Tree Automata (SWTAs)

Synchronized Weighted Tree Automata
LSTAs + weights + alternation

$\mathcal{A} = \langle Q, \Omega, \delta, \mathit{root}, E \rangle$[1]

- $Q = \{q_1, \ldots, q_n\}$ — a finite set of states,
- $\Omega = \{1, \ldots, k\}$ — a finite set of colours (for synchronization),
- δ — a transition function (details later),
- $\mathit{root} \in Q$ — the root state, and
- $E \subseteq Q$ — set of leaf states.

[1] we ignore internal labels and only consider the structure of trees and leaf values

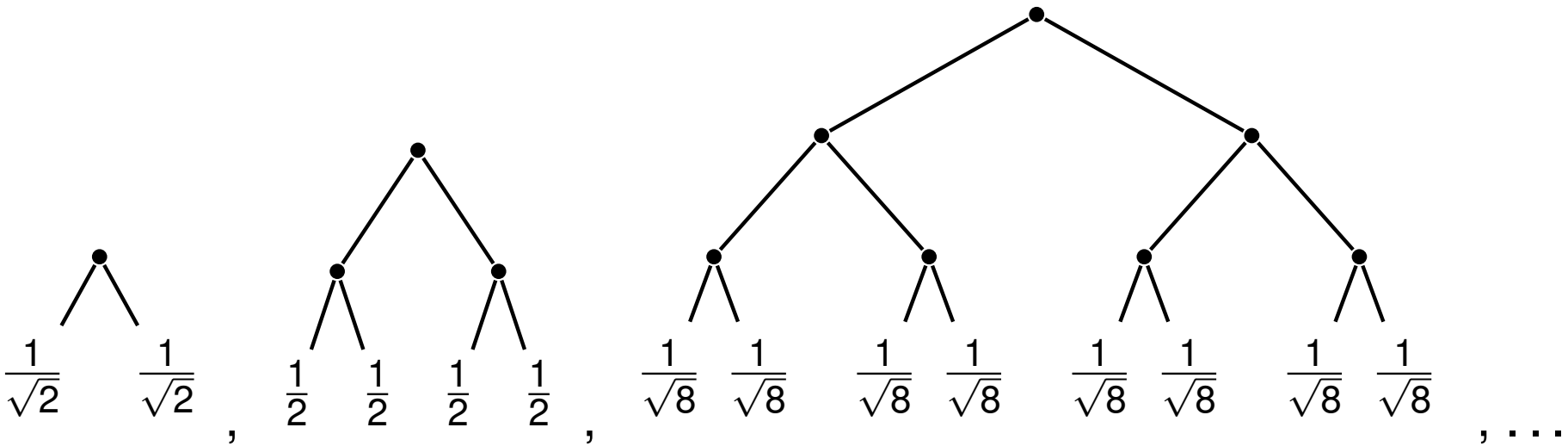
Synchronized Weighted Tree Automata

Example:

$$root \xrightarrow{1} \left(\frac{1}{\sqrt{2}}p, \quad \frac{1}{\sqrt{2}}p\right)$$
$$p \xrightarrow{1} \left(\frac{1}{\sqrt{2}}p, \quad \frac{1}{\sqrt{2}}p\right)$$

- p is a leaf state

- expresses a set of uniform superposition states:



Synchronized Weighted Tree Automata

General form of transitions:

$$u \xrightarrow{1} \left(\tfrac{1}{2}p + \tfrac{1}{\sqrt{8}}q, \quad \tfrac{1}{\sqrt{2}}t - \tfrac{1}{4}z\right)$$

Synchronized Weighted Tree Automata

General form of transitions:

$$u \xrightarrow{1} \left(\tfrac{1}{2}p + \tfrac{1}{\sqrt{8}}q, \quad \tfrac{1}{\sqrt{2}}t - \tfrac{1}{4}z\right)$$

semantics (generator view, inductive):

1. generate trees t_p and t_q from p and q (can be the leaf $\boxed{1}$ if a state is leaf)
2. multiply leaves of t_p by $\frac{1}{2}$ and leaves of t_q by $\frac{1}{\sqrt{8}}$
3. $t_{left} \leftarrow \frac{1}{2}t_p + \frac{1}{\sqrt{8}}t_q$ (structures of t_p and t_q need to match)
4. obtain t_{right} in a similar way
5. construct $t \leftarrow cons(t_{left}, t_{right})$

Synchronized Weighted Tree Automata

General form of transitions:

$$u \xrightarrow{1} \left(\tfrac{1}{2}p + \tfrac{1}{\sqrt{8}}q, \quad \tfrac{1}{\sqrt{2}}t - \tfrac{1}{4}z\right)$$

semantics (generator view, inductive):

1. generate trees t_p and t_q from p and q (can be the leaf $\boxed{1}$ if a state is leaf)
2. multiply leaves of t_p by $\frac{1}{2}$ and leaves of t_q by $\frac{1}{\sqrt{8}}$
3. $t_{left} \leftarrow \frac{1}{2}t_p + \frac{1}{\sqrt{8}}t_q$ (structures of t_p and t_q need to match)
4. obtain t_{right} in a similar way
5. construct $t \leftarrow cons(t_{left}, t_{right})$

language $\mathcal{L}(\mathcal{A})$: the set of trees generated by $\mathcal{A}$

Synchronized Weighted Tree Automata — Properties

- language union: closed

Synchronized Weighted Tree Automata — Properties

- language union: closed
- language complement: not-closed (counting argument)

Synchronized Weighted Tree Automata — Properties

- language union: closed
- language complement: not-closed (counting argument)
- language emptiness: **PSPACE**-complete
  - $\in$ **PSPACE**: subset construction
  - **PSPACE**-hardness: reduction from NFA universality

Synchronized Weighted Tree Automata — Properties

- language union: closed
- language complement: not-closed (counting argument)
- language emptiness: **PSPACE**-complete
  - $\in$ **PSPACE**: subset construction
  - **PSPACE**-hardness: reduction from NFA universality
- intersection non-emptiness: undecidable (reduction from PCP)

Synchronized Weighted Tree Automata — Properties

- language union: closed
- language complement: not-closed (counting argument)
- language emptiness: **PSPACE**-complete
  - $\in$ **PSPACE**: subset construction
  - **PSPACE**-hardness: reduction from NFA universality
- intersection non-emptiness: undecidable (reduction from PCP)
  - language intersection: not-closed (from the above)

Synchronized Weighted Tree Automata — Properties

- language union: closed
- language complement: not-closed (counting argument)
- language emptiness: **PSPACE**-complete
  - $\in$ **PSPACE**: subset construction
  - **PSPACE**-hardness: reduction from NFA universality
- intersection non-emptiness: undecidable (reduction from PCP)
  - language intersection: not-closed (from the above)
- language inclusion/equivalence: undecidable (reduction from emptiness of TM)
  - $\rightsquigarrow$ bad (?) $\qquad C(\mathit{Pre}) \stackrel{?}{\subseteq} \mathit{Post}$

Synchronized Weighted Tree Automata — Properties

alternative:

- don't talk about the standard language, but
- $\mathcal{A}$'s tree function $f_{\mathcal{A}} : \Omega^* \to \mathbb{T}_{\mathbb{C}}$ ($\Omega = \{1, \ldots, k\}$, $\mathbb{T}_{\mathbb{C}} =$ perfect trees with $\mathbb{C}$ leaves)

Synchronized Weighted Tree Automata — Properties

alternative:

- don't talk about the standard language, but
- $\mathcal{A}$'s tree function $f_{\mathcal{A}} : \Omega^* \to \mathbb{T}_{\mathbb{C}}$ $\quad (\Omega = \{1, \ldots, k\}, \mathbb{T}_{\mathbb{C}} =$ perfect trees with $\mathbb{C}$ leaves)
- e.g.,

$$p \xrightarrow{1} \left(\tfrac{1}{\sqrt{2}} p, \quad \tfrac{1}{\sqrt{2}} p\right)$$
$$p \xrightarrow{2} \left(\tfrac{1}{\sqrt{2}} u, \quad \tfrac{1}{\sqrt{2}} u\right)$$

p is root, u is leaf

Synchronized Weighted Tree Automata — Properties

alternative:

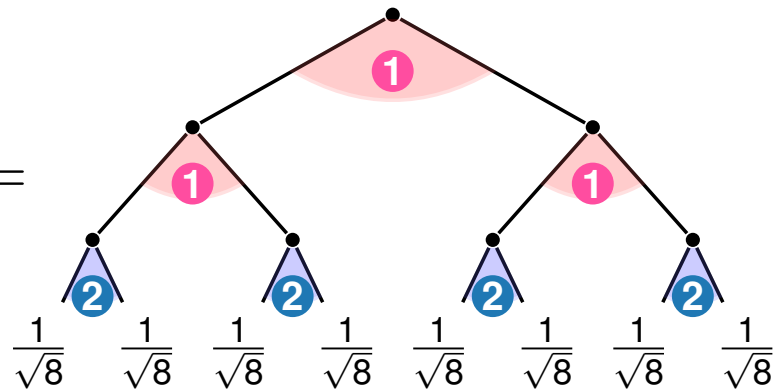
- don't talk about the standard language, but
- $\mathcal{A}$'s tree function $f_{\mathcal{A}} : \Omega^* \to \mathbb{T}_{\mathbb{C}}$ ($\Omega = \{1, \ldots, k\}$, $\mathbb{T}_{\mathbb{C}} =$ perfect trees with $\mathbb{C}$ leaves)
- e.g.,

$p \xrightarrow{1} (\frac{1}{\sqrt{2}}p, \frac{1}{\sqrt{2}}p)$

$p \xrightarrow{2} (\frac{1}{\sqrt{2}}u, \frac{1}{\sqrt{2}}u)$

p is root, u is leaf

$f_{\mathcal{A}}(112) =$



Synchronized Weighted Tree Automata — Properties

alternative:

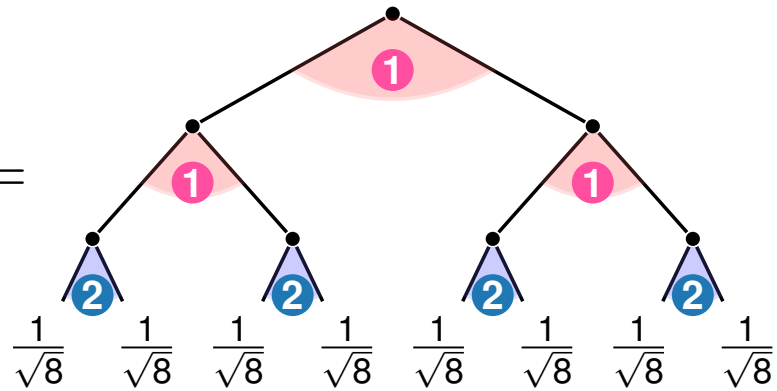
- don't talk about the standard language, but
- $\mathcal{A}$'s tree function $f_{\mathcal{A}} : \Omega^* \to \mathbb{T}_{\mathbb{C}}$ ($\Omega = \{1, \ldots, k\}$, $\mathbb{T}_{\mathbb{C}} =$ perfect trees with $\mathbb{C}$ leaves)
- e.g.,

$p \xrightarrow{1} \left(\frac{1}{\sqrt{2}}p, \quad \frac{1}{\sqrt{2}}p\right)$

$p \xrightarrow{2} \left(\frac{1}{\sqrt{2}}u, \quad \frac{1}{\sqrt{2}}u\right)$

p is root, u is leaf

$f_{\mathcal{A}}(112) =$



- very useful!
  - $\rightsquigarrow$ each computational basis state tree can map to one string in Ω^*: $Pre \leftrightarrow \Omega^*$
  - $C(Pre)$ preserves tree function: e.g., $f_{Pre}(112) \xrightarrow{C} f_{C(Pre)}(112)$
  - allows relational specification, circuit equivalence checking

Synchronized Weighted Tree Automata — Properties

- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

Synchronized Weighted Tree Automata — Properties

- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

- **PSPACE**-hard and in **EXPTIME** (conj.: **EXPTIME**-complete)

Synchronized Weighted Tree Automata — Properties

- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

- **PSPACE**-hard and in **EXPTIME** (conj.: **EXPTIME**-complete)
- **algorithm** for $f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}}$:
  1. test $dom(f_{\mathcal{A}}) = dom(f_{\mathcal{B}})$; if no, **fail**

Synchronized Weighted Tree Automata — Properties

- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

- **PSPACE**-hard and in **EXPTIME** (conj.: **EXPTIME**-complete)
- **algorithm** for $f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}}$:
  1. test $dom(f_{\mathcal{A}}) = dom(f_{\mathcal{B}})$; if no, **fail**
  2. compute SWTA $\mathcal{C}$ s.t. $f_{\mathcal{C}} = f_{\mathcal{A}} - f_{\mathcal{B}}$

Synchronized Weighted Tree Automata — Properties

- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

- **PSPACE**-hard and in **EXPTIME** (conj.: **EXPTIME**-complete)
- **algorithm** for $f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}}$:
  1. test $dom(f_{\mathcal{A}}) = dom(f_{\mathcal{B}})$; if no, **fail**
  2. compute SWTA $\mathcal{C}$ s.t. $f_{\mathcal{C}} = f_{\mathcal{A}} - f_{\mathcal{B}}$
  3. $f_{\mathcal{A}} = f_{\mathcal{B}}$ iff $\mathcal{C}$ only generates 0-trees

Synchronized Weighted Tree Automata — Properties

- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

- **PSPACE**-hard and in **EXPTIME** (conj.: **EXPTIME**-complete)
- **algorithm** for $f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}}$:
  1. test $dom(f_{\mathcal{A}}) = dom(f_{\mathcal{B}})$; if no, **fail**
  2. compute SWTA $\mathcal{C}$ s.t. $f_{\mathcal{C}} = f_{\mathcal{A}} - f_{\mathcal{B}}$
  3. $f_{\mathcal{A}} = f_{\mathcal{B}}$ iff $\mathcal{C}$ only generates 0-trees
  4. transform $\mathcal{C}$ into a linear transition system $\mathcal{S}$

Synchronized Weighted Tree Automata — Properties

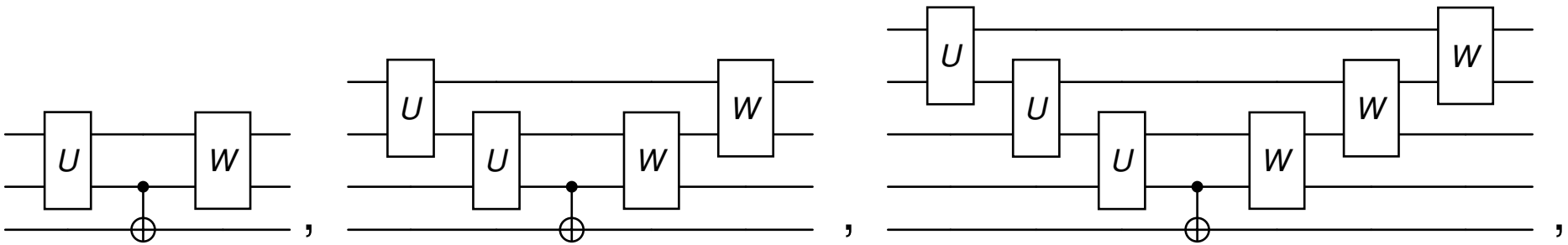
- tree function equivalence/inclusion of $\mathcal{A}$ and $\mathcal{B}$

$$f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}} \qquad\qquad f_{\mathcal{A}} \stackrel{?}{\subseteq} f_{\mathcal{B}}$$

- **PSPACE**-hard and in **EXPTIME** (conj.: **EXPTIME**-complete)
- **algorithm** for $f_{\mathcal{A}} \stackrel{?}{=} f_{\mathcal{B}}$:
  1. test $dom(f_{\mathcal{A}}) = dom(f_{\mathcal{B}})$; if no, **fail**
  2. compute SWTA $\mathcal{C}$ s.t. $f_{\mathcal{C}} = f_{\mathcal{A}} - f_{\mathcal{B}}$
  3. $f_{\mathcal{A}} = f_{\mathcal{B}}$ iff $\mathcal{C}$ only generates 0-trees
  4. transform $\mathcal{C}$ into a linear transition system $\mathcal{S}$
  5. run Karr's algorithm to compute for every state in $\mathcal{S}$ the vector space of all linear relations, check 0-dim subspace for interesting states
     - M. Karr. Affine Relationships Among Variables of a Program. Acta Inf., 6:133–151, 1976.

Weighted Tree Transducers (WTTs)

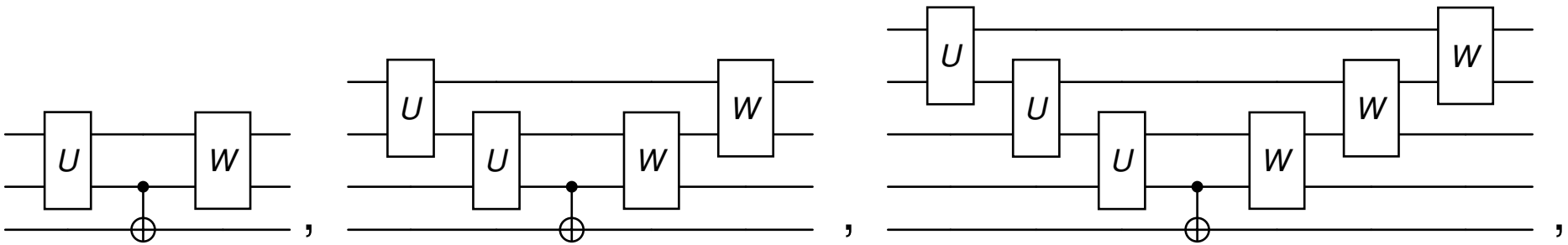
Issue: How to (efficiently) represent size-parameterized families of circuits, e.g.,



...

Weighted Tree Transducers (WTTs)

Issue: How to (efficiently) represent size-parameterized families of circuits, e.g.,



...

- previous approach (specialized automata operations) doesn't work any more

Regular Model Checking

- popular approach to parameterized verification of classical systems

$$C(Pre) \stackrel{?}{\subseteq} Post$$

- *Pre*, *Post* — represented by automata (of various kinds)
- C — represented by a transducer

Regular Model Checking

- popular approach to parameterized verification of classical systems

$$C(Pre) \stackrel{?}{\subseteq} Post$$

- Pre, $Post$ — represented by automata (of various kinds)
- C — represented by a transducer
- $C(Pre)$ — obtained by image computation
- $\stackrel{?}{\subseteq}$ — language inclusion check

Regular Model Checking

- popular approach to parameterized verification of classical systems

$$C(Pre) \stackrel{?}{\subseteq} Post$$

- Pre, $Post$ — represented by automata (of various kinds)
- C — represented by a transducer
- $C(Pre)$ — obtained by image computation
- $\stackrel{?}{\subseteq}$ — language inclusion check

Goal

We need a transducer model for SWTAs!

Weighted Tree Transducers (WTTs)

Weighted Tree Transducers (WTTs)
represent quantum gates

Weighted Tree Transducers (WTTs)

Weighted Tree Transducers (WTTs)
represent quantum gates

$\mathcal{T} = \langle Q, \delta, root, E \rangle$

- $Q = \{q_1, \ldots, q_n\}$ — a finite set of states,
- δ — a transition function
- $root \in Q$ — the root state, and
- $E \subseteq Q$ — set of leaf states.

Weighted Tree Transducers (WTTs)

Weighted Tree Transducers (WTTs)
represent quantum gates

$\mathcal{T} = \langle Q, \delta, root, E \rangle$

- $Q = \{q_1, \ldots, q_n\}$ — a finite set of states,
- δ — a transition function
- $root \in Q$ — the root state, and
- $E \subseteq Q$ — set of leaf states.

transitions:

$$q \to \left(\tfrac{1}{2} p(\mathbf{L}) + \tfrac{1}{\sqrt{8}} u(\mathbf{R}), \quad \tfrac{1}{\sqrt{2}} t(\mathbf{L}) - \tfrac{1}{4} z(\mathbf{R})\right)$$

- **L** and **R** denote the input **L**eft and **R**ight sub-trees

Weighted Tree Transducers (WTTs)

- capture compactly all fixed-sized gates
  - previous work: specialized procedures
- composition: quadratic
- image computation: quadratic
- equivalence/inclusion: **EXPTIME**-complete
  - $\in$ **EXPTIME**: similar construction as for tree func. equiv. of SWTAs
  - **EXPTIME**-hardness: reduction from TA intersection nonemptiness

Weighted Tree Transducers (WTTs)

- capture compactly all fixed-sized gates
  - previous work: specialized procedures
- composition: quadratic
- image computation: quadratic
- equivalence/inclusion: **EXPTIME**-complete
  - $\in$ **EXPTIME**: similar construction as for tree func. equiv. of SWTAs
  - **EXPTIME**-hardness: reduction from TA intersection nonemptiness

- WTT for fixed-sized QFT: quadratic to #qubits
  - QFT: Quantum Fourier Transform

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

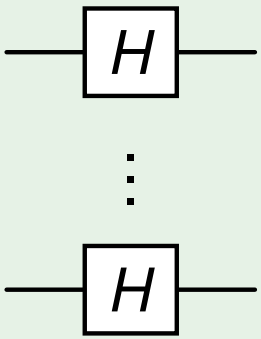
- algorithm to synthesize WTTs from description for some patterns

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

- algorithm to synthesize WTTs from description for some patterns

Example (Hadamard Transform)



$$q \to \left(\tfrac{1}{\sqrt{2}} q(\mathbf{L}) + \tfrac{1}{\sqrt{2}} q(\mathbf{R}),\quad \tfrac{1}{\sqrt{2}} q(\mathbf{L}) - \tfrac{1}{\sqrt{2}} q(\mathbf{R})\right)$$

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

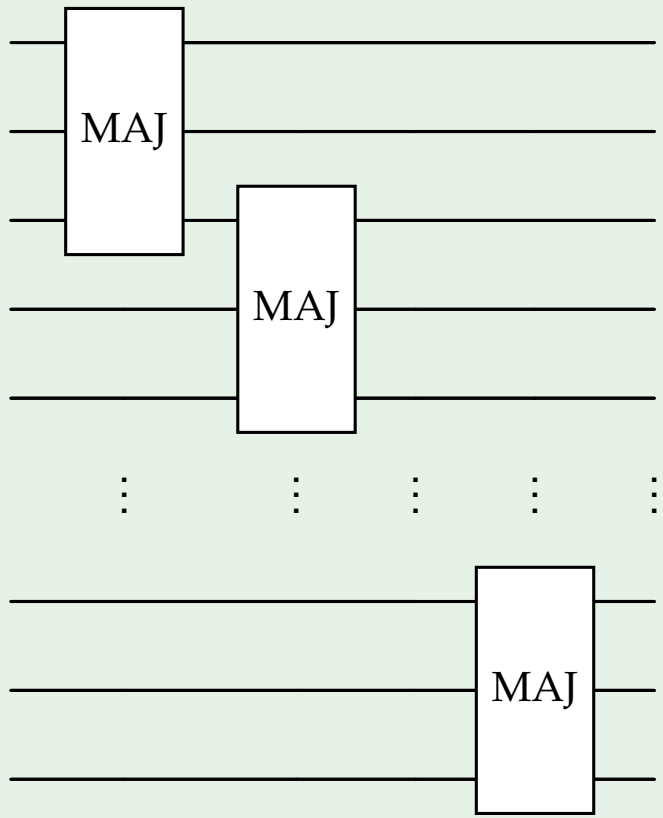
- algorithm to synthesize WTTs from description for some patterns

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

- algorithm to synthesize WTTs from description for some patterns

Example (Part of Ripple-Carry Adder)



Experiments

Implementation: prototype AUTOQ-PARA

Benchmarks:

Pre/*Post*-condition verification

- BV — Bernstein-Vazirani's algorithm with a fixed secret
- Adder — carry-ripple adder
- QECC — syndrom extraction of a repetition code

Experiments

Implementation: prototype AUTOQ-PARA

Benchmarks:

Pre/*Post*-condition verification

- BV — Bernstein-Vazirani's algorithm with a fixed secret
- Adder — carry-ripple adder
- QECC — syndrom extraction of a repetition code

circuit	time
BV	0.014 s
Adder	11.007 s
QECC	0.314 s
Grover	0.088 s
Ham. sim.	0.663 s

Experiments

Implementation: prototype AUTOQ-PARA

Benchmarks:

Pre/*Post*-condition verification

- BV — Bernstein-Vazirani's algorithm with a fixed secret
- Adder — carry-ripple adder
- QECC — syndrom extraction of a repetition code

circuit	time
BV	0.014 s
Adder	11.007 s
QECC	0.314 s
Grover	0.088 s
Ham. sim.	0.663 s

Equivalence testing

- Grover — one Grover iter., w/ vs. w/o multi-control gates
- Ham. sim. — Hamiltonian simulation of a *1D Heisenberg spin chain*, basic vs. optimized version

[Abdulla, Chen, Hečko, Holík, Lengál, Lin, Thinniyam. Parameterized Verification of Quantum Circuits. POPL'26.]

Takeaways and Future Directions

Takeaways

Quantum ♡ Automata

Takeaways

Quantum ♡ Automata

- opportunities for new useful formal models

Takeaways

Quantum ♡ Automata

- opportunities for new useful formal models
- a lot of fun!

Future Directions

- parameterized verification of more complex circuits
  - support for recursion
  - support for parameterized rotation gates
  - $\leadsto$ enable verification of QFT, quantum counting, QPE, ...

Future Directions

- parameterized verification of more complex circuits
  - support for recursion
  - support for parameterized rotation gates
  - $\leadsto$ enable verification of QFT, quantum counting, QPE, ...
- a good specification language
  - expressive, user-friendly
  - can compile to automata quickly
  - (one attempt accepted for CAV'26)

Future Directions

- parameterized verification of more complex circuits
  - support for recursion
  - support for parameterized rotation gates
  - $\rightsquigarrow$ enable verification of QFT, quantum counting, QPE, ...
- a good specification language
  - expressive, user-friendly
  - can compile to automata quickly
  - (one attempt accepted for CAV'26)
- equivalence checking of parameterized circuits
  - oracle-based circuits
  - dynamic circuits
  - various notions of equivalence

Future Directions

- parameterized verification of more complex circuits
  - support for recursion
  - support for parameterized rotation gates
  - $\rightsquigarrow$ enable verification of QFT, quantum counting, QPE, ...
- a good specification language
  - expressive, user-friendly
  - can compile to automata quickly
  - (one attempt accepted for CAV'26)
- equivalence checking of parameterized circuits
  - oracle-based circuits
  - dynamic circuits
  - various notions of equivalence
- How to represent quantum circuits efficiently?
  - algebra over trees? logic?

Future Directions

- parameterized verification of more complex circuits
  - support for recursion
  - support for parameterized rotation gates
  - $\rightsquigarrow$ enable verification of QFT, quantum counting, QPE, ...
- a good specification language
  - expressive, user-friendly
  - can compile to automata quickly
  - (one attempt accepted for CAV'26)
- equivalence checking of parameterized circuits
  - oracle-based circuits
  - dynamic circuits
  - various notions of equivalence
- How to represent quantum circuits efficiently?
  - algebra over trees? logic?

Thank you!