Parameterized Verification of Quantum Circuits

Parosh Aziz Abdulla[1,2] Yu-Fang Chen[3] Michal Hečko[4] Lukáš Holík[4,5]
Ondřej Lengál[4] Jyun-Ao Lin[6] Ramanathan S. Thinniyam[1]

[1]Uppsala University, Sweden
[2]Mälardalen University, Sweden
[3]Academia Sinica, Taiwan
[4]Brno University of Technology, Czech Republic
[5]Aalborg University, Denmark
[6]National Taipei University of Technology, Taiwan

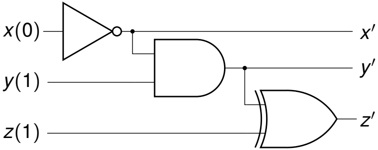
POPL'26

Outline

101 on Quantum Circuits

101 on Quantum Circuits

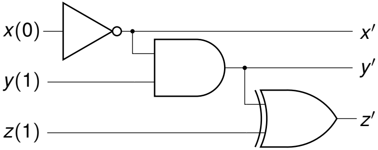
Classical



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

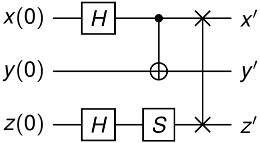
101 on Quantum Circuits

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

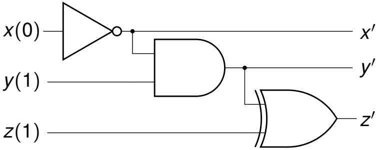
Quantum



x'	y'	z'	*amp*
0	**0**	**0**	**25 %**
0	0	1	0 %
0	1	0	0 %
0	**1**	**1**	**25 %**
1	**0**	**0**	**25 %**
1	0	1	0 %
1	1	0	0 %
1	**1**	**1**	**25 %**

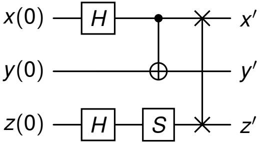
101 on Quantum Circuits

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

Quantum



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***

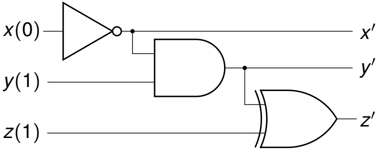
$amp(\vec{x}) \in \mathbb{C}$

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

$\sum_{\vec{x}} \Pr(\vec{x}) = 1$

101 on Quantum Circuits

Classical

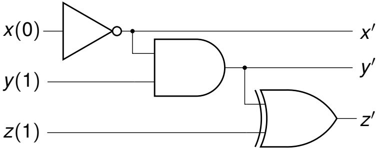


A gate is a truth table

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

101 on Quantum Circuits

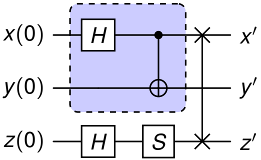
Classical



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

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

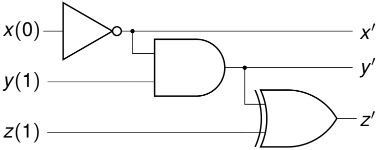
---

unitary matrix:

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

101 on Quantum Circuits

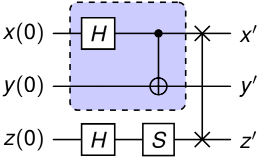
Classical



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, ...

Parameterized Quantum Circuits

- Here we deal with the parameter being size
  - do not confuse with parameters being, e.g., rotation angles

Parameterized Quantum Circuits

- Here we deal with the parameter being size
  - do not confuse with parameters being, e.g., rotation angles
- 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, . . .

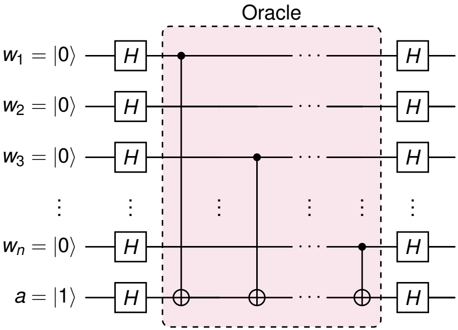


Figure: Bernstein-Vazirani for the secret $(\mathtt{10})^*(\mathtt{1}+\varepsilon)$

Verification of Quantum Circuits

Verification of Quantum Circuits

Our framework:

Verification of Quantum Circuits

Our framework:

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

- *Pre* and *Post* denote sets of **quantum** states
- *C* is a parameterized circuit

Verification of Quantum Circuits

Our framework:

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

- *Pre* and *Post* denote sets of **quantum** states
- *C* is a parameterized circuit

Meaning:

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

General approach

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

Representation

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

Issue: How to (efficiently) represent:

Representation

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

Issue: How to (efficiently) represent:

a) (infinite) sets of quantum states (of various size)

- e.g., $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$

  — all uniform superposition states

Representation

precondition *postcondition*

$$\{Pre\} \quad C \quad \{Post\}$$

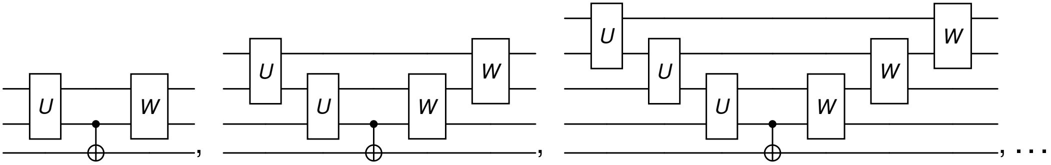
circuit

Issue: How to (efficiently) represent:

a) (infinite) sets of quantum states (of various size)
  - e.g., $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$

    — all uniform superposition states

b) a size-parameterized family of circuits, e.g.,



... in a way that we can test $C(Pre) \overset{?}{\subseteq} Post$?

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

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

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

Question

What automata/transducer models are suitable for representing quantum states and circuits?

Quantum States are Trees

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**

$\Rightarrow$

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**

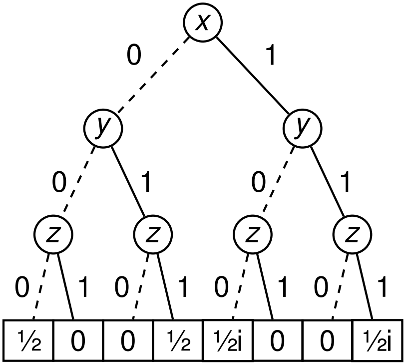
$\Rightarrow$

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**

$\Rightarrow$

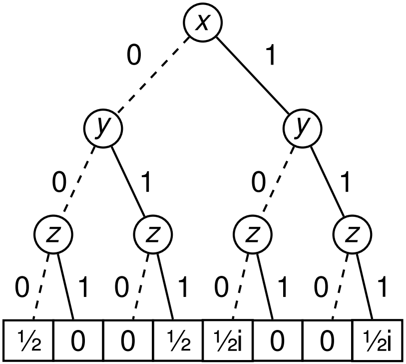


- perfect tree of height n (the number of qubits) $\rightsquigarrow 2^n$ leaves

Quantum 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
- set of states $\rightsquigarrow$ tree automata

Tree Automata for Quantum States

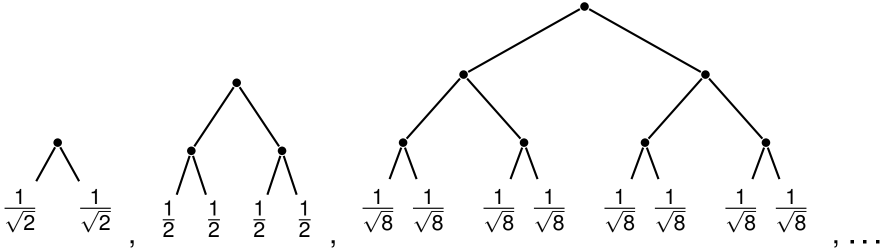
Set of uniform superposition states:

- $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$

Tree Automata for Quantum States

Set of uniform superposition states:

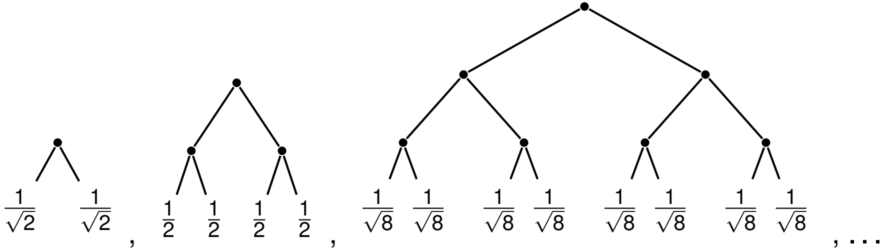
- $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$



Tree Automata for Quantum States

Set of uniform superposition states:

- $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$

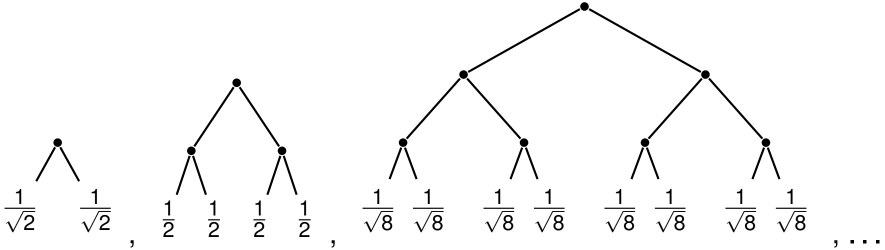


Which tree automata model to use?

Tree Automata for Quantum States

Set of uniform superposition states:

- $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$



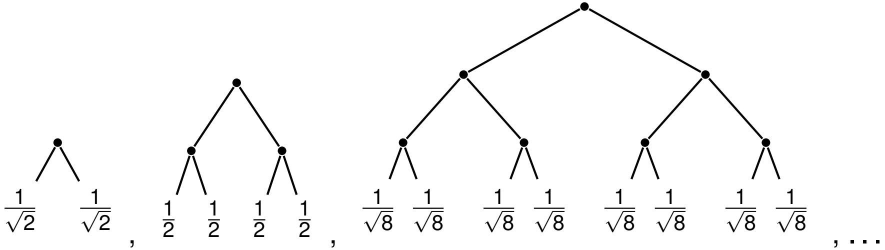
Which tree automata model to use?

- standard tree automata — cannot express perfect trees

Tree Automata for Quantum States

Set of uniform superposition states:

- $\{(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}), (\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}), \ldots\}$



Which tree automata model to use?

- standard tree automata — cannot express perfect trees
- level-synchronized tree automata (LSTAs) [POPL'25] —
  - **can** express perfect trees
  - **cannot** express unboundedly many amplitudes

Synchronized Weighted Tree Automata (SWTAs)

Synchronized Weighted Tree Automata (SWTAs)

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

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

[1] in this talk, we ignore internal labels, i.e., we only consider the structure of trees and leaf values

Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Synchronization — similar to LSTAs [POPL'25]
  - be able to synchronize subtrees on the same level

No synchronization (standard TAs)

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

q is root, u is leaf

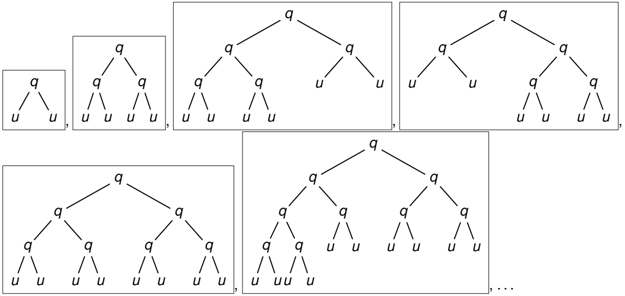
Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Synchronization — similar to LSTAs [POPL'25]
  - be able to synchronize subtrees on the same level

No synchronization (standard TAs)

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

q is root, u is leaf



Synchronized Weighted Tree Automata (SWTAs) — Transitions

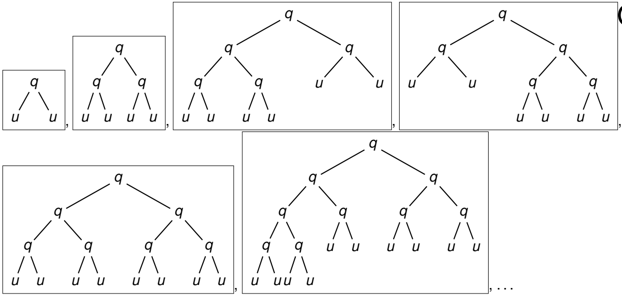
- Synchronization — similar to LSTAs [POPL'25]
  - be able to synchronize subtrees on the same level

No synchronization (standard TAs)

$$q \rightarrow (q, q)$$

$$q \rightarrow (u, u)$$

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!

Synchronized Weighted Tree Automata (SWTAs) — Transitions

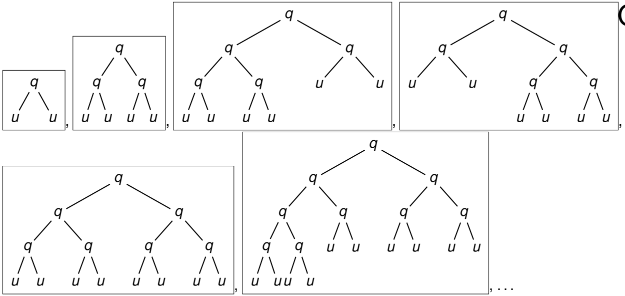
- Synchronization — similar to LSTAs [POPL'25]
  - be able to synchronize subtrees on the same level

No synchronization (standard TAs)

$$q \rightarrow (q, q)$$

$$q \rightarrow (u, u)$$

q is root, u is leaf



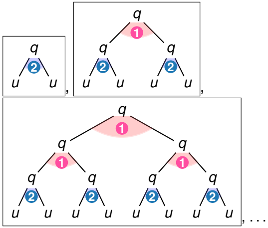
With synchronization

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

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

q is root, u is leaf

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

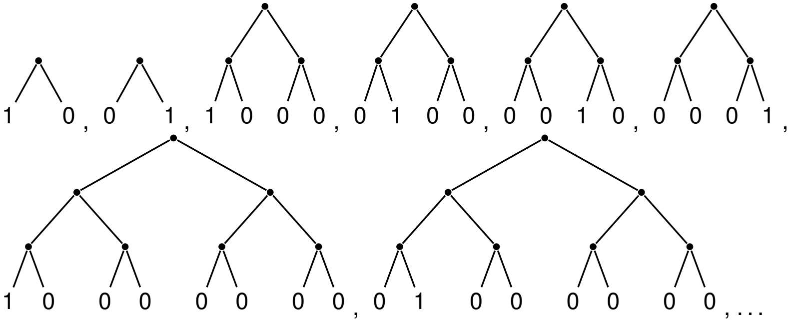


Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Synchronization — similar to LSTAs [POPL'25]

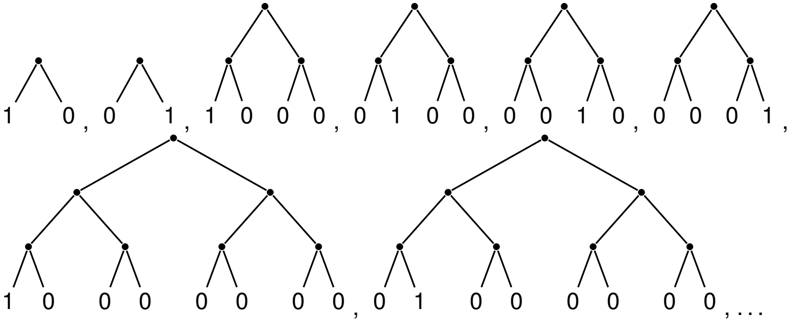
Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Synchronization — similar to LSTAs [POPL'25]
- can express all computational basis states

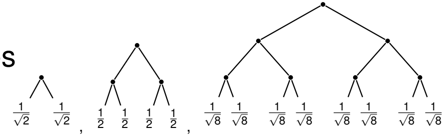


Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Synchronization — similar to LSTAs [POPL'25]
- can express all computational basis states



- but still cannot express all uniform superposition states
  - ∞-many amplitudes



Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Weightedness — states are given weights

Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Weightedness — states are given weights
- Example:

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

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

- p is a leaf state

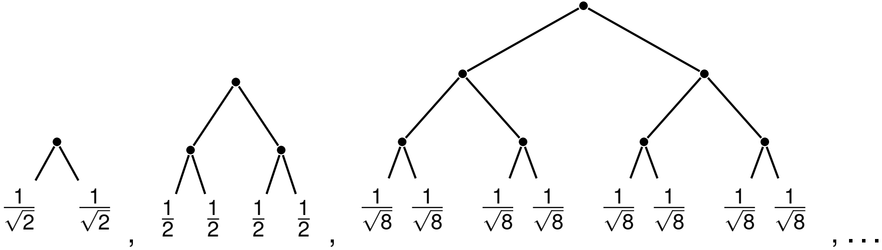
Synchronized Weighted Tree Automata (SWTAs) — Transitions

- Weightedness — states are given weights
- Example:

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

- p is a leaf state

- expresses set of uniform superposition states:



Synchronized Weighted Tree Automata (SWTAs) — Transitions

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 (SWTAs) — Transitions

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):

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 (SWTAs) — Transitions

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):

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

Synchronized Weighted Tree Automata (SWTAs) — Properties

- language union: closed

Synchronized Weighted Tree Automata (SWTAs) — Properties

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

Synchronized Weighted Tree Automata (SWTAs) — 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 (SWTAs) — 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 (SWTAs) — 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 (SWTAs) — 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 (?) $\qquad C(\mathit{Pre}) \overset{?}{\subseteq} \mathit{Post}$

Synchronized Weighted Tree Automata (SWTAs) — Properties

alternative:

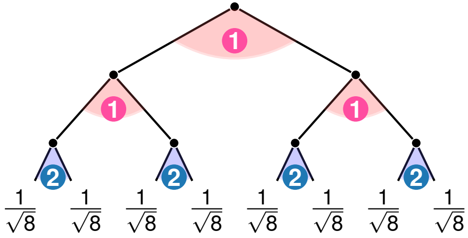
- don't talk about the standard language, but
- $\mathcal{A}$'s tree function $f_{\mathcal{A}} \colon \Omega^* \to \mathbb{T}_{\mathbb{C}}$ $\quad (\Omega = \{❶, \ldots, ⓚ\}, \mathbb{T}_{\mathbb{C}} =$ perfect trees with $\mathbb{C}$ leaves)

Synchronized Weighted Tree Automata (SWTAs) — Properties

alternative:

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

$f_{\mathcal{A}}$(❶❶❷) =

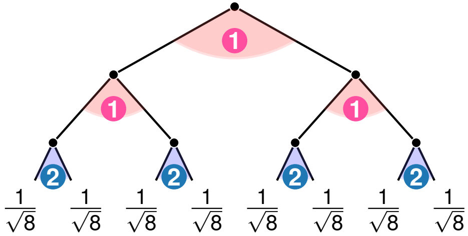


Synchronized Weighted Tree Automata (SWTAs) — Properties

alternative:

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

$f_{\mathcal{A}}($❶❶❷$) =$



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

Synchronized Weighted Tree Automata (SWTAs) — 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 (SWTAs) — 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**

Synchronized Weighted Tree Automata (SWTAs) — 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**
- **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)

Weighted Tree Transducers (WTTs)

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

- $Q = \{q_1, \ldots, q_n\}$ — 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)

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

- $Q = \{q_1, \ldots, q_n\}$ — 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

Weighted Tree Transducers (WTTs)

- capture compactly all fixed-sized gates
  - previous work: specialized procedures
- composition: quadratic
- image computation: quadratic
- equivalence/inclusion: **EXPTIME**-complete

- WTT for fixed-sized QFT: quadratic to #qubits

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

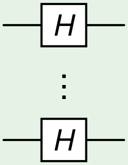
- algorithm to synthesize WTTs from circuit description for some patterns

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

- algorithm to synthesize WTTs from circuit description for some patterns

Example (Hadamard Transform)



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

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

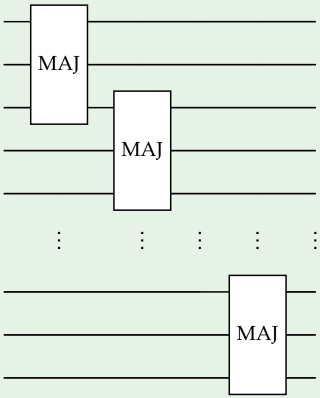
- algorithm to synthesize WTTs from circuit description for some patterns

Weighted Tree Transducers (WTTs)

WTTs for parameterized-size circuits:

- algorithm to synthesize WTTs from circuit description for some patterns

Example (Part of Ripple-Carry Adder)



Evaluation

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

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

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

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

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

Conclusion

- a new automata and transducer model to capture parameterized quantum circuits
- quite expressive, but with decidable important properties allowing fully-automated verification

Conclusion

- a new automata and transducer model to capture parameterized quantum circuits
- quite expressive, but with decidable important properties allowing fully-automated verification

Future work:

- extend with recursion and counters — to enable support of parameterized QFT
- conversion from OpenQASM files
- support for measurement

Conclusion

- a new automata and transducer model to capture parameterized quantum circuits
- quite expressive, but with decidable important properties allowing fully-automated verification

Future work:

- extend with recursion and counters — to enable support of parameterized QFT
- conversion from OpenQASM files
- support for measurement

Thank You!