

Automata-Based Fully Automated Analysis of Quantum Programs with AutoQ

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

VQC'25

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)

$$\begin{array}{ccccc} \textit{precondition} & & & & \textit{postcondition} \\ \{Pre\} & S & \{Post\} \\ & \textit{statement} & \end{array}$$

- *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)

$$\begin{array}{ccccc} \textit{precondition} & & & & \textit{postcondition} \\ \{Pre\} & S & \{Post\} \\ & \textit{statement} & \end{array}$$

- *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:

$$\begin{array}{ccccc} \textit{precondition} & & & & \textit{postcondition} \\ \{Pre\} & & C & & \{Post\} \\ & & \textit{circuit} & & \end{array}$$

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

Verification of Quantum Circuits

Verification of **quantum circuits**:

$$\begin{array}{ccc} \textit{precondition} & & \textit{postcondition} \\ \{Pre\} & C & \{Post\} \\ & \textit{circuit} & \end{array}$$

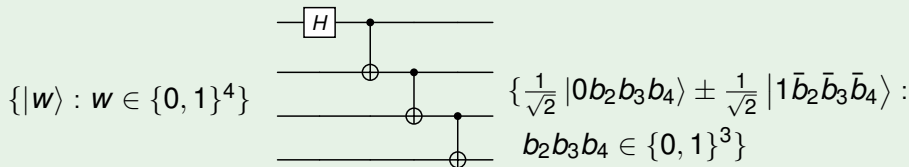
- *Pre* and *Post* denote **sets of quantum states**

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)



Pre

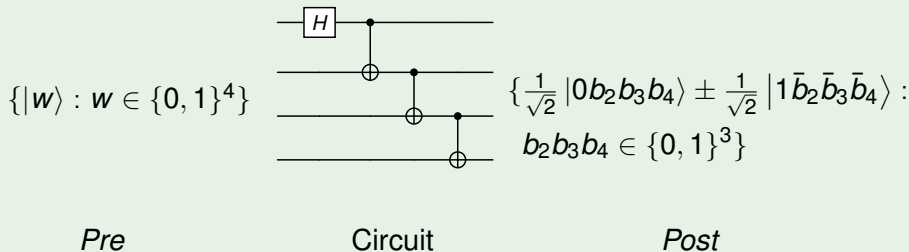
Circuit

Post

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

Verification of Quantum Circuits

Example (GHZ)

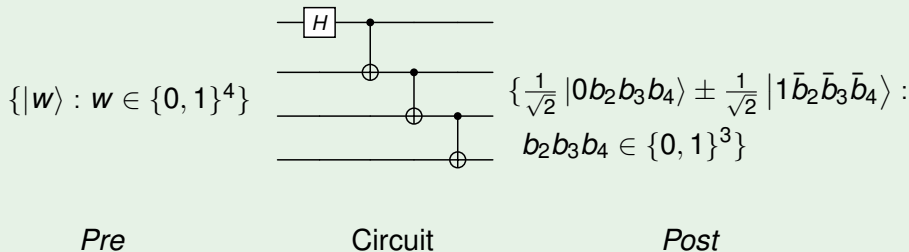


$$Pre = \{|0000\rangle, |0001\rangle, \dots, |1111\rangle\}$$

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

Verification of Quantum Circuits

Example (GHZ)



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

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

■ naively \leadsto 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	$\frac{1}{2}$
0	0	1	0
0	1	0	0
0	1	1	$\frac{1}{2}$
1	0	0	$\frac{1}{2}i$
1	0	1	0
1	1	0	0
1	1	1	$\frac{1}{2}i$



Quantum States are Trees

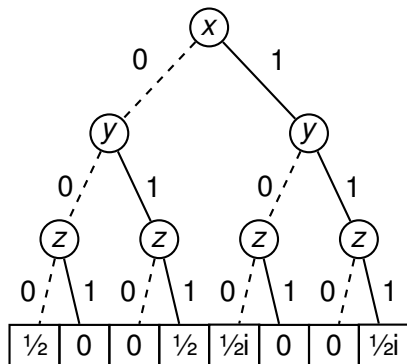
x	y	z	amp
0	0	0	$\frac{1}{2}$
0	0	1	0
0	1	0	0
0	1	1	$\frac{1}{2}$
1	0	0	$\frac{1}{2}i$
1	0	1	0
1	1	0	0
1	1	1	$\frac{1}{2}i$



$\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{2}i$	0	0	$\frac{1}{2}i$
---------------	---	---	---------------	----------------	---	---	----------------

Quantum States are Trees

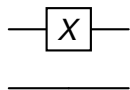
x	y	z	amp
0	0	0	$\frac{1}{2}$
0	0	1	0
0	1	0	0
0	1	1	$\frac{1}{2}$
1	0	0	$\frac{1}{2}i$
1	0	1	0
1	1	0	0
1	1	1	$\frac{1}{2}i$



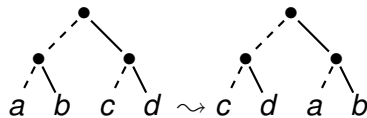
- perfect tree of height n (the number of qubits) $\leadsto 2^n$ leaves

Quantum Gates are Tree Operations

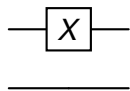
Quantum Gates are Tree Operations



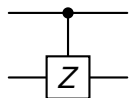
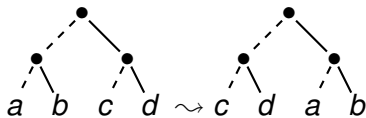
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^X \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^I$$



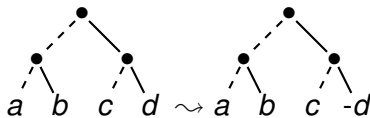
Quantum Gates are Tree Operations



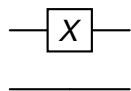
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^X \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^I$$



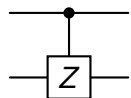
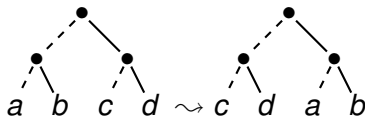
$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$



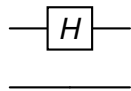
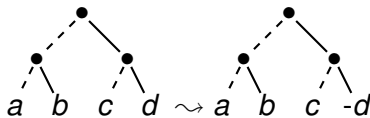
Quantum Gates are Tree Operations



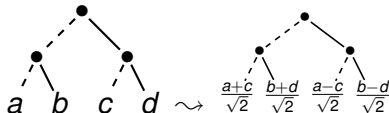
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^X \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^I$$



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



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



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

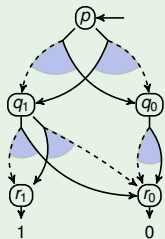
- How to efficiently represent sets of trees?

Tree automata!

- tree automata

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

Example



represents the set

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} = \left\{ \begin{array}{cc} \begin{array}{c} \bullet \\ \swarrow \quad \searrow \\ \bullet \quad \bullet \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ 1 \quad 0 \quad 0 \quad 0 \end{array}, & \begin{array}{c} \bullet \\ \swarrow \quad \searrow \\ \bullet \quad \bullet \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ 0 \quad 1 \quad 0 \quad 0 \end{array}, \\ \begin{array}{c} \bullet \\ \swarrow \quad \searrow \\ \bullet \quad \bullet \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ 0 \quad 0 \quad 1 \quad 0 \end{array}, & \begin{array}{c} \bullet \\ \swarrow \quad \searrow \\ \bullet \quad \bullet \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ 0 \quad 0 \quad 0 \quad 1 \end{array} \end{array} \right\}$$

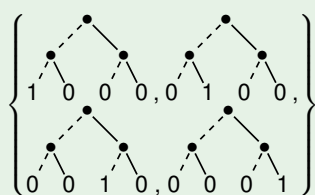
Representing *Pre* and *Post* with Tree Automata

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

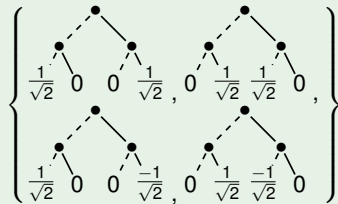
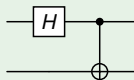
Representing *Pre* and *Post* with Tree Automata

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

Example (GHZ)



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

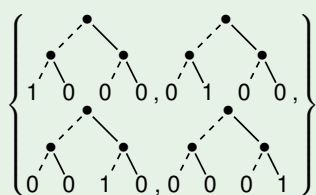


$\mathcal{L}(\mathcal{A}_{Post})$

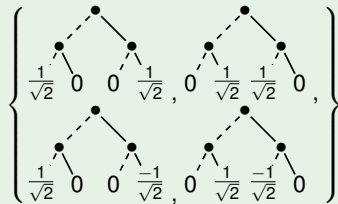
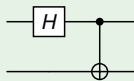
Representing *Pre* and *Post* with Tree Automata

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

Example (GHZ)



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



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

■ \mathcal{A} 's size can be small

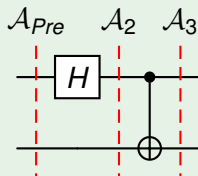
► e.g., \mathcal{A} for $\{|w\rangle : w \in \{0, 1\}^n\}$ needs $\mathcal{O}(n)$ states/transitions

Verification with Tree Automata

$$\begin{array}{ccc} \textit{precondition} & & \textit{postcondition} \\ \{\mathcal{A}_{Pre}\} & C & \{\mathcal{A}_{Post}\} \\ & \textit{circuit} & \end{array}$$

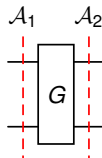
- Run C with \mathcal{A}_{Pre} :

Example



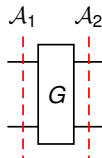
- ... and test $\mathcal{L}(\mathcal{A}_3) \subseteq \mathcal{L}(\mathcal{A}_{Post})$
 - (standard tree automata inclusion is **EXPTIME**-complete)

Abstract Transformers for Quantum Gates



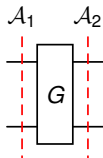
- How to compute \mathcal{A}_2 such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
 - naively (i.e., one tree by one) — doesn't scale

Abstract Transformers for Quantum Gates



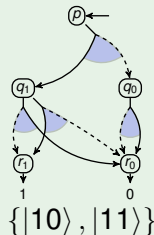
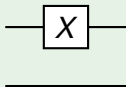
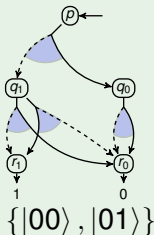
- How to compute \mathcal{A}_2 such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
 - ▶ naively (i.e., one tree by one) — doesn't scale
- \leadsto **abstract transformers**
 - ▶ specialized automata operations for concrete gates

Abstract Transformers for Quantum Gates

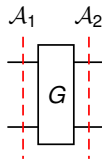


- How to compute \mathcal{A}_2 such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
 - ▶ naively (i.e., one tree by one) — doesn't scale
- \leadsto **abstract transformers**
 - ▶ specialized automata operations for concrete gates

Example



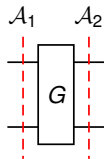
Abstract Transformers for Quantum Gates



■ Supported gate types:

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

Abstract Transformers for Quantum Gates

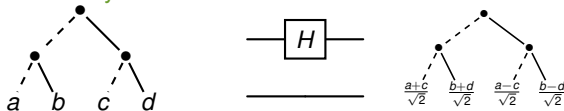


■ Supported gate types:

- ▶ **(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, \dots

- need to **synchronize** subtrees of the same tree



- **standard** tree automata: $\mathcal{O}(2^{|\mathcal{A}_1|})$
- **level-synchronized** tree automata: $\mathcal{O}(|\mathcal{A}_1|^2)$

Quantum Circuit Verification Algorithm

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

- Algorithm:

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

Quantum Circuit Verification Algorithm

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

■ 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

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

■ Algorithm:

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

Level-Synchronized Tree Automata

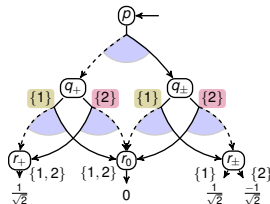
Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

- allow synchronization across subtrees

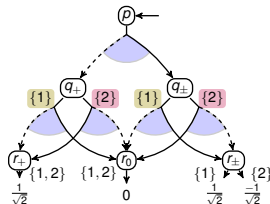


- cost of operations
 - ▶ (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
 - ▶ general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)

Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

- allow synchronization across subtrees

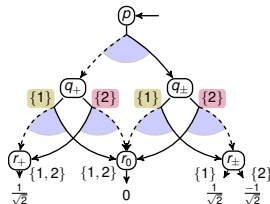


- cost of operations
 - ▶ (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
 - ▶ general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)
- incomparable to basic TAs
 - ▶ cannot express “all trees”

Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

- allow synchronization across subtrees



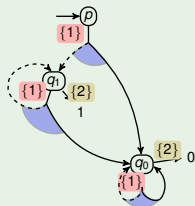
- cost of operations
 - ▶ (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
 - ▶ general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)
- incomparable to basic TAs
 - ▶ cannot express “all trees”
- language operations:
 - ▶ **emptiness**: **PSPACE**-complete
 - ▶ **inclusion/equivalence**: **PSPACE**-hard, in **EXSPACE**

Level-Synchronized Tree Automata (LSTAs)

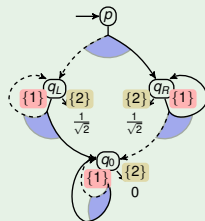
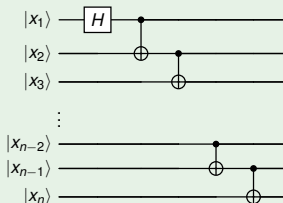
Level-Synchronized Tree Automata

- enable basic parameterized verification

Example (GHZ)



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



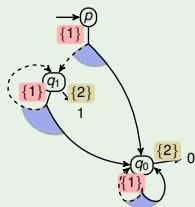
$$\{\frac{1}{\sqrt{2}} |0^n\rangle + \frac{1}{\sqrt{2}} |1^n\rangle : n \geq 1\}$$

Level-Synchronized Tree Automata (LSTAs)

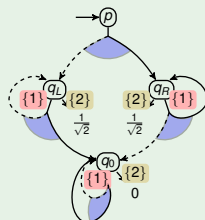
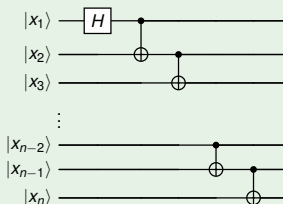
Level-Synchronized Tree Automata

- enable basic parameterized verification

Example (GHZ)



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



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

- GHZ, fermionic unitary evolution (single/double fermionic excitation)

Weighted Level-Synchronized Tree Automata

Weighted Level-Synchronized Tree Automata

Weighted Level-Synchronized Tree Automata

transitions:

- tree automata: $q \rightarrow a(q_1, q_2)$
- level-synchronized TAs: $q \xrightarrow{1} a(q_1, q_2)$
- weighted LSTAs: $q \xrightarrow{1} a(\frac{1}{\sqrt{2}}q_1 + \frac{1}{\sqrt{2}}q_2, \frac{1}{\sqrt{2}}q_1 - \frac{1}{\sqrt{2}}q_2)$

Weighted Level-Synchronized Tree Automata

Weighted Level-Synchronized Tree Automata

transitions:

- tree automata: $q \rightarrow a(q_1, q_2)$
- level-synchronized TAs: $q \xrightarrow{1} a(q_1, q_2)$
- weighted LSTAs: $q \xrightarrow{1} a(\frac{1}{\sqrt{2}}q_1 + \frac{1}{\sqrt{2}}q_2, \frac{1}{\sqrt{2}}q_1 - \frac{1}{\sqrt{2}}q_2)$

weighted LSTAs:

- language operations:
 - ▶ emptiness: **PSPACE**-complete
 - ▶ inclusion/equivalence: undecidable
 - ▶ coloured equivalence: **PSPACE**-hard, in **EXSPACE**
 - usable for testing equivalence
 - sometimes usable for pre/post-verification

Weighted Level-Synchronized Tree Automata

Weighted Level-Synchronized Tree Automata

transitions:

- tree automata: $q \rightarrow a(q_1, q_2)$
- level-synchronized TAs: $q \xrightarrow{1} a(q_1, q_2)$
- weighted LSTAs: $q \xrightarrow{1} a(\frac{1}{\sqrt{2}}q_1 + \frac{1}{\sqrt{2}}q_2, \frac{1}{\sqrt{2}}q_1 - \frac{1}{\sqrt{2}}q_2)$

weighted LSTAs:

- language operations:
 - ▶ emptiness: **PSPACE**-complete
 - ▶ inclusion/equivalence: undecidable
 - ▶ coloured equivalence: **PSPACE**-hard, in **EXSPACE**
 - usable for testing equivalence
 - sometimes usable for pre/post-verification

+ weighted tree transducers (instead of specialized algorithms)

- regular model checking-like verification algorithm
- support for parameterized verification

What can we verify?

- (parameterized versions of) Bernstein-Vazirani, multi-control Toffoli
- Grover's algorithm:
 - ▶ pre/post with precise sets of quantum states
 - ▶ single/all oracles
 - ▶ $P(\text{solution}) > 0.9$ (symbolic TAs)
 - ▶ one iteration increases probability (symbolic TAs)
 - ▶ equivalence of parameterized one loop (WLSTAs)
 - ▶ weakly-measured version (symbolic TAs + measurements)
- repeat until success circuits
- circuits from **RevLib**, **Feynman**, **Random**
- parameterized GHZ (LSTAs), arithmetic circuits (WLSTAs), basic QECCs (WLSTAs)
- parameterized fermionic unitary evolution (LSTAs), Hamiltonian simulation (WLSTAs)

Takeaways and Future Directions

Quantum Automata

Future Directions

- a good **specification language**
 - ▶ expressive, user-friendly
 - ▶ can compile to (*)TAs quickly
- parameterized verification of circuits with QFT
- How to represent quantum circuits efficiently?
 - ▶ algebra over trees? logic?

Future Directions

- a good **specification language**
 - ▶ expressive, user-friendly
 - ▶ can compile to (*)TAs quickly
- parameterized verification of circuits with QFT
- How to represent quantum circuits efficiently?
 - ▶ algebra over trees? logic?

Thank you!

References

- Chen, Chung, Lengál, Lin, Tsai, Yen. An Automata-Based Framework for Verification and Bug Hunting in Quantum Circuits. PLDI'23.
- Chen, Chung, Lengál, Lin, Tsai. AutoQ: An Automata-Based Quantum Circuit Verifier. CAV'23.
- Abdulla, Chen, Chen, Holík, Lengál, Lin, Lo, Tsai. Verifying Quantum Circuits with Level-Synchronized Tree Automata. POPL'25.
- Chen, Chung, Hsieh, Huang, Lengál, Lin, Tsai. AutoQ 2.0: From Verification of Quantum Circuits to Verification of Quantum Programs. TACAS'25.

Symbolic Amplitudes

Introducing Symbolic Amplitudes

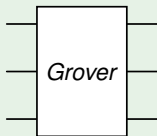
- So far, we only used **finite** numbers of **amplitudes**

Introducing Symbolic Amplitudes

- So far, we only used **finite** numbers of **amplitudes**
- But what about verifying a property like this?

Example

$$\{h|000\rangle + \ell|w\rangle : \\ w \in \{0,1\}^3 \setminus \{000\}\}$$



$$\{h'|000\rangle + \ell'|w\rangle : \\ w \in \{0,1\}^3 \setminus \{000\}\}$$

global constraint:

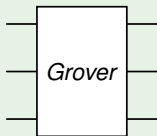
$$h, h', \ell, \ell' \in \mathbb{C} \wedge |h'|^2 \geq |h|^2 \wedge |\ell'|^2 \leq |\ell|^2 \wedge \\ |h|^2 \geq |\ell|^2 \wedge |h|^2 + 7|\ell|^2 = 1 \wedge |h'|^2 + 7|\ell'|^2 = 1$$

Introducing Symbolic Amplitudes

- So far, we only used **finite** numbers of **amplitudes**
- But what about verifying a property like this?

Example

$$\{h|000\rangle + \ell|w\rangle : \\ w \in \{0,1\}^3 \setminus \{000\}\}$$



$$\{h'|000\rangle + \ell'|w\rangle : \\ w \in \{0,1\}^3 \setminus \{000\}\}$$

global constraint:

$$h, h', \ell, \ell' \in \mathbb{C} \wedge |h'|^2 \geq |h|^2 \wedge |\ell'|^2 \leq |\ell|^2 \wedge \\ |h|^2 \geq |\ell|^2 \wedge |h|^2 + 7|\ell|^2 = 1 \wedge |h'|^2 + 7|\ell'|^2 = 1$$

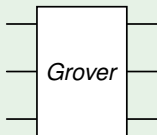
- uncountably many amplitudes
- **uncountably many quantum states**

Introducing Symbolic Amplitudes

- So far, we only used **finite** numbers of **amplitudes**
- But what about verifying a property like this?

Example

$$\{h|000\rangle + \ell|w\rangle : \\ w \in \{0,1\}^3 \setminus \{000\}\}$$



$$\{h'|000\rangle + \ell'|w\rangle : \\ w \in \{0,1\}^3 \setminus \{000\}\}$$

global constraint:

$$h, h', \ell, \ell' \in \mathbb{C} \wedge |h'|^2 \geq |h|^2 \wedge |\ell'|^2 \leq |\ell|^2 \wedge \\ |h|^2 \geq |\ell|^2 \wedge |h|^2 + 7|\ell|^2 = 1 \wedge |h'|^2 + 7|\ell'|^2 = 1$$

- uncountably many amplitudes
- **uncountably many quantum states**
- \leadsto **symbolic amplitudes!**

Verifying Quantum Circuits using Symbolic Amplitudes

Modifications to the verification algorithm:

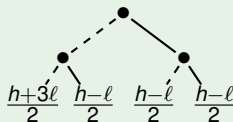
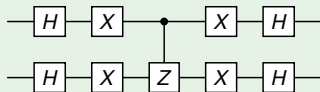
- (L-S) tree automata \leadsto **symbolic** (L-S) tree automata
 - ▶ alphabet contains **symbolic values**, terms, and **predicates**

Verifying Quantum Circuits using Symbolic Amplitudes

Modifications to the verification algorithm:

- (L-S) tree automata \leadsto **symbolic** (L-S) tree automata
 - ▶ alphabet contains **symbolic values**, terms, and **predicates**
- abstract transformers are **symbolic** (*à la* symbolic execution):

Example



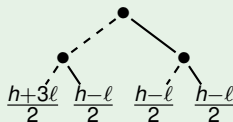
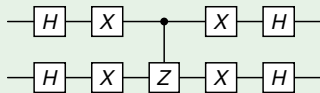
Grover's diffusion operator

Verifying Quantum Circuits using Symbolic Amplitudes

Modifications to the verification algorithm:

- (L-S) tree automata \leadsto **symbolic** (L-S) tree automata
 - ▶ alphabet contains **symbolic values**, terms, and **predicates**
- abstract transformers are **symbolic** (*à la* symbolic execution):

Example



Grover's diffusion operator

- modified **language inclusion test**

Verification of Quantum Circuits with Loops

- Common structure of quantum programs:

while ($M(x_i) = 0$)
 C ;

- repeat-until-success, weakly measured Grover

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

```
1 Pre:  $\{1 \mid |0^{n+2}\rangle + 0 \mid *\rangle\}$ ;  
2  $H_3; H_4; \dots; H_{n+2}$ ;  
3  $O_{2,\dots,(n+2)}; CK_1^2; O_{2,\dots,(n+2)}$ ;  
4 Inv:  $\{v_{sol1} \mid 000^n\rangle + v_k \mid 000^{n-1}1\rangle + \dots +$   
5      $v_k \mid 001^n\rangle + v_{sol2} \mid 100^n\rangle + 0 \mid *\rangle\}$ ;  
6 while  $M_1 = 0$  do  
7 |  $\{\mathcal{G}_{2,\dots,(n+2)}; O_{2,\dots,(n+2)}; CK_1^2; O_{2,\dots,(n+2)}\}$ ;  
8 Post:  $\{1 \mid 10s\rangle + 0 \mid *\rangle\}$ ;
```

Verification of Quantum Circuits with Loops

- Common structure of quantum programs:

while ($M(x_i) = 0$)
 C ;

- repeat-until-success, weakly measured Grover
- **symbolic** (L-S)TAs + measurements

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

```
1 Pre:  $\{1 | 0^{n+2}\rangle + 0 | *\rangle\}$ ;  
2  $H_3; H_4; \dots; H_{n+2}$ ;  
3  $O_{2,\dots,(n+2)}; CK_1^2; O_{2,\dots,(n+2)}$ ;  
4 Inv:  $\{v_{sol1} | 000^n\rangle + v_k | 000^{n-1}1\rangle + \dots +$   
5      $v_k | 001^n\rangle + v_{sol2} | 100^n\rangle + 0 | *\rangle\}$ ;  
6 while  $M_1 = 0$  do  
7 |  $\{\mathcal{G}_{2,\dots,(n+2)}; O_{2,\dots,(n+2)}; CK_1^2; O_{2,\dots,(n+2)}\}$ ;  
8 Post:  $\{1 | 10s\rangle + 0 | *\rangle\}$ ;
```
