New Approaches to Simulation and Analysis of Quantum Circuits

Ondřej Lengál

Brno University of Technology, Czech Republic

FI MUNI (Colloquium)

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" $\rightsquigarrow$ 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
- $\rightsquigarrow$ we need to be prepared (computer-aided analysis)

Why Quantum Computation?

Quantum computation:

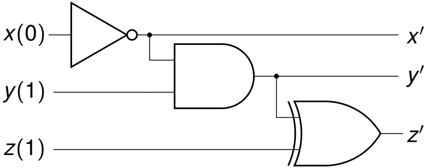
- first proposed by Feynman (1982)
- promises to efficiently solve some problems we don't know how to efficiently solve classically
  - factoring (Shor, 1994): "exponential" $\rightsquigarrow$ polynomial
  - unstructured database search (Grover, 1996): $\mathcal{O}(2^n) \rightsquigarrow \mathcal{O}(\sqrt{2^n})$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

Outline

Short Quantum Introduction

Classical vs. Quantum Circuits — State

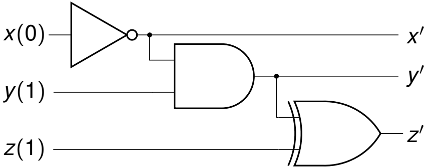
Classical



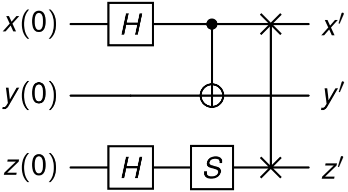
x'	y'	z'	χ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	**1**	**0**	**1**
1	1	1	0

Classical vs. Quantum Circuits — State

Classical



Quantum

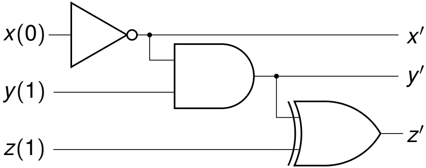


x'	y'	z'	χ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	**1**	**0**	**1**
1	1	1	0

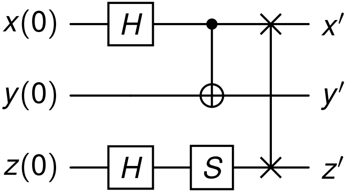
x'	y'	z'	*amp*
0	**0**	**0**	**25 %**
0	0	1	0 %
0	1	0	0 %
0	**1**	**1**	**25 %**
1	**0**	**0**	**25 %**
1	0	1	0 %
1	1	0	0 %
1	**1**	**1**	**25 %**

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

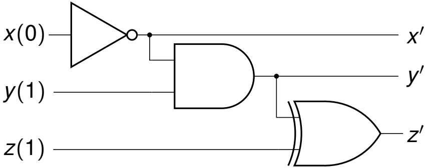
x'	y'	z'	*amp*
0	**0**	**0**	½
0	0	1	0
0	1	0	0
0	**1**	**1**	½
1	**0**	**0**	½ *i*
1	0	1	0
1	1	0	0
1	**1**	**1**	½ *i*

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

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

Classical vs. Quantum Circuits — Gates

Classical

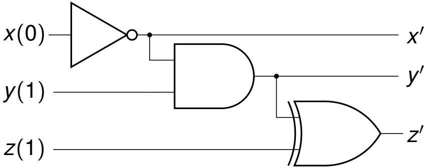


A gate is a truth table

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

Classical vs. Quantum Circuits — Gates

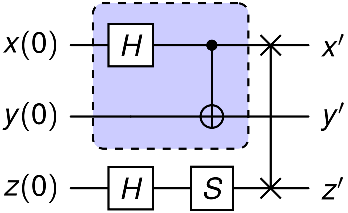
Classical



A gate is a truth table

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

Quantum



A gate is a unitary matrix

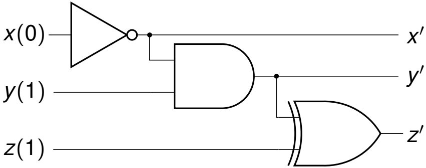
$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}$$

unitary matrix:

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

Classical vs. Quantum Circuits — Gates

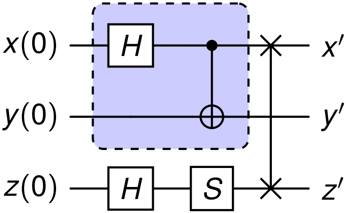
Classical



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

Quantum Circuit Analysis

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 Author*; 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)

$$\overset{\textit{precondition}}{\{Pre\}}\ \underset{\textit{statement}}{S}\ \overset{\textit{postcondition}}{\{Post\}}$$

- *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)

$$\overset{\textit{precondition}}{\{Pre\}}\ \underset{\textit{statement}}{S}\ \overset{\textit{postcondition}}{\{Post\}}$$

- *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:

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

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

Verification of Quantum Circuits

Verification of quantum circuits:

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

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

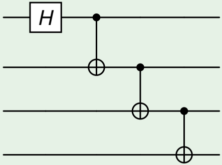
Meaning:

- 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)

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



$\{\frac{1}{\sqrt{2}}|0b_2b_3b_4\rangle \pm \frac{1}{\sqrt{2}}|1\bar{b}_2\bar{b}_3\bar{b}_4\rangle :$
$b_2b_3b_4 \in \{0,1\}^3\}$

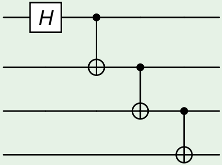
Pre Circuit *Post*

$$Pre = \{|0000\rangle, |0001\rangle, \ldots, |1111\rangle\}$$
$$\text{e.g., } |0010\rangle = [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]^T$$

Verification of Quantum Circuits

Example (GHZ)

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



$\{\frac{1}{\sqrt{2}}|0b_2b_3b_4\rangle \pm \frac{1}{\sqrt{2}}|1\bar{b}_2\bar{b}_3\bar{b}_4\rangle : b_2b_3b_4 \in \{0,1\}^3\}$

Pre Circuit *Post*

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

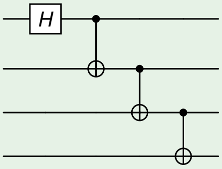
$$\text{e.g., } |0010\rangle = [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]^T$$

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

Verification of Quantum Circuits

Example (GHZ)

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



$\{\frac{1}{\sqrt{2}}|0b_2b_3b_4\rangle \pm \frac{1}{\sqrt{2}}|1\bar{b}_2\bar{b}_3\bar{b}_4\rangle : b_2b_3b_4 \in \{0,1\}^3\}$

Pre — Circuit — *Post*

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

$$\text{e.g., } |0010\rangle = [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]^T$$

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**

$\Rightarrow$

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***

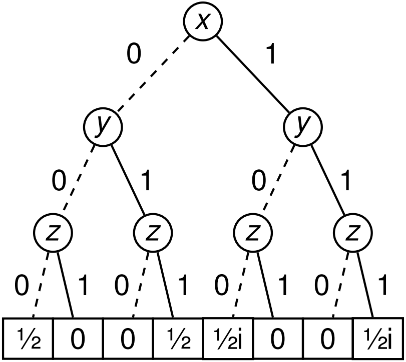
$\Rightarrow$

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***

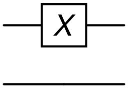
$\Rightarrow$



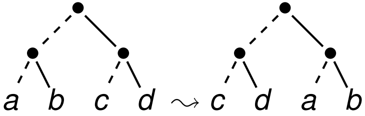
- perfect tree of height n (the number of qubits) $\rightsquigarrow 2^n$ leaves

Quantum Gates are Tree Operations

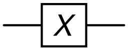
Quantum Gates are Tree Operations



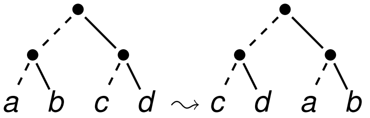
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$

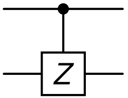


Quantum Gates are Tree Operations

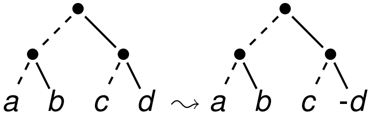


$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$

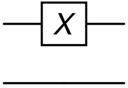




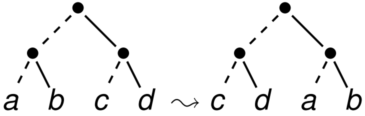
$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

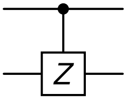


Quantum Gates are Tree Operations

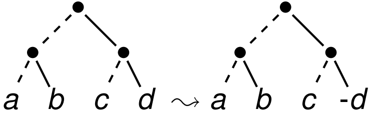


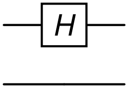
$$X_1 = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{X} \otimes \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{I}$$



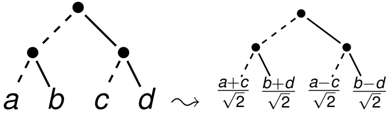


$$CZ_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$





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



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

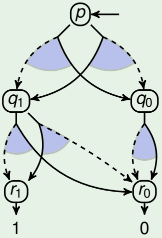
Sets of Quantum States are Sets of Trees

- How to efficiently represent sets of trees?

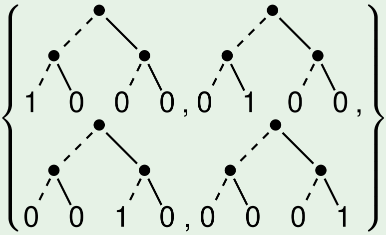
Tree automata!

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

Example



represents the set



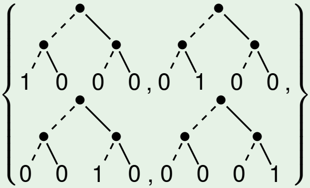
Representing *Pre* and *Post* with Tree Automata

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

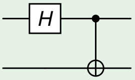
Representing *Pre* and *Post* with Tree Automata

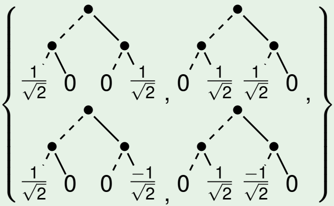
$$\underbrace{\{\mathcal{A}_{Pre}\}}_{\text{precondition}} \quad \underbrace{C}_{\text{circuit}} \quad \underbrace{\{\mathcal{A}_{Post}\}}_{\text{postcondition}}$$

Example (GHZ)



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





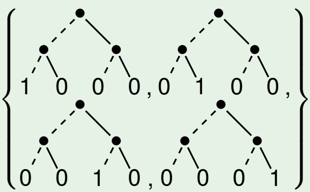
$\mathcal{L}(\mathcal{A}_{Post})$

Representing *Pre* and *Post* with Tree Automata

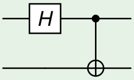
precondition $\{\mathcal{A}_{Pre}\}$ C $\{\mathcal{A}_{Post}\}$ *postcondition*

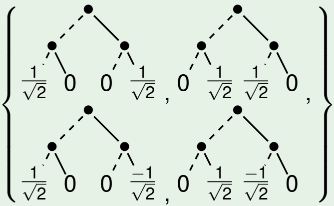
circuit

Example (GHZ)



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





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

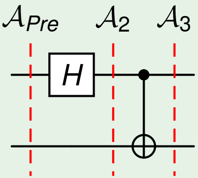
- $\mathcal{A}$'s size can be small
  - e.g., $\mathcal{A}$ for $\{|w\rangle : w \in \{0,1\}^n\}$ needs $\mathcal{O}(n)$ states/transitions

Verification with Tree Automata

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

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

Example



Verification with Tree Automata

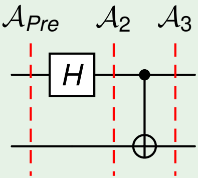
precondition *postcondition*

$$\{\mathcal{A}_{Pre}\} \quad C \quad \{\mathcal{A}_{Post}\}$$

circuit

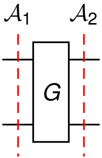
- Run C with $\mathcal{A}_{Pre}$:

Example



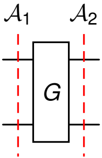
- …and test $\mathcal{L}(\mathcal{A}_3) \subseteq \mathcal{L}(\mathcal{A}_{Post})$
  - (tree automata inclusion is **EXPTIME**-complete)

Abstract Transformers for Quantum Gates



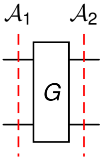
- How to compute $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
  - naively (i.e., one tree by one) — doesn't scale

Abstract Transformers for Quantum Gates



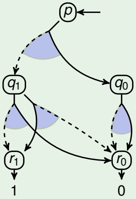
- How to compute $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
  - naively (i.e., one tree by one) — doesn't scale
- $\rightsquigarrow$ abstract transformers
  - specialized automata operations for concrete gates

Abstract Transformers for Quantum Gates

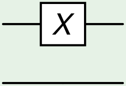


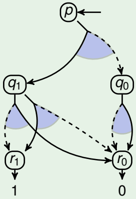
- How to compute $\mathcal{A}_2$ such that $\mathcal{L}(\mathcal{A}_2) = G(\mathcal{L}(\mathcal{A}_1))$ efficiently?
  - naively (i.e., one tree by one) — doesn't scale
- $\leadsto$ abstract transformers
  - specialized automata operations for concrete gates

Example



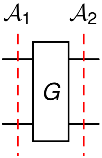
$\{|00\rangle, |01\rangle\}$





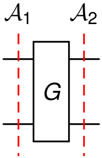
$\{|10\rangle, |11\rangle\}$

Abstract Transformers for Quantum Gates

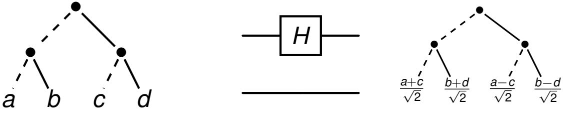


- Supported gate types:
  - (anti-)diagonal: X, Y, Z, S, T, R_z, controls (*CNOT*, *CZ*, *Toffoli*, ...)
    - simple manipulation with automaton: $\mathcal{O}(|\mathcal{A}_1|)$

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



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

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 $\mathcal{A}_{Pre}$.

Quantum Circuit Verification Algorithm

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

- 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

$$\underbrace{\{\mathcal{A}_{Pre}\}}^{\textit{precondition}} \quad \underbrace{C}_{\textit{circuit}} \quad \overbrace{\{\mathcal{A}_{Post}\}}^{\textit{postcondition}}$$

- 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

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

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

Quantum Circuit Verification Algorithm

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

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

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 $\mathcal{L}(\mathcal{A}_C) \subseteq \mathcal{L}(\mathcal{A}_{Post})$.
- Used to verify/find bugs in a number of quantum circuits:
  - Bernstein-Vazirani, Grover (Single/All), MCToffoli, ...
- Scales to up to 40 qubits / 140k gates.
- Found a confirmed bug in QCEC (SOTA equivalence checker).
- Established a connection between **quantum** and **automata**.

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

Symbolic Amplitudes

Introducing Symbolic Amplitudes

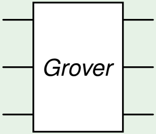
- So far, we only used finite sets of quantum states

Introducing Symbolic Amplitudes

- So far, we only used finite sets of quantum states
- 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:

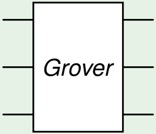
$$h, h', \ell, \ell' \in \mathbb{C} \wedge |h'|^2 \geq |h|^2 \wedge |\ell'|^2 \leq |\ell|^2 \wedge$$
$$|h|^2 \geq |\ell|^2 \wedge |h|^2 + 7|\ell|^2 = 1 \wedge |h'|^2 + 7|\ell'|^2 = 1$$

Introducing Symbolic Amplitudes

- So far, we only used finite sets of quantum states
- 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:

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

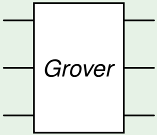
- uncountably many quantum states

Introducing Symbolic Amplitudes

- So far, we only used finite sets of quantum states
- 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 quantum states
- $\rightsquigarrow$ symbolic amplitudes!

Verifying Quantum Circuits using Symbolic Amplitudes

Modifications to the verification algorithm:

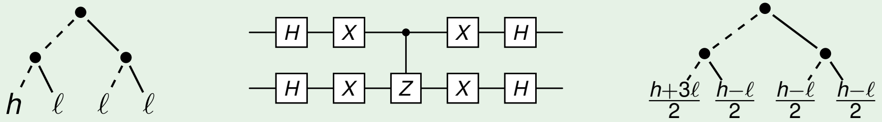
- tree automata $\rightsquigarrow$ symbolic tree automata
  - alphabet contains symbolic values, terms, and predicates

Verifying Quantum Circuits using Symbolic Amplitudes

Modifications to the verification algorithm:

- tree automata $\rightsquigarrow$ symbolic tree automata
  - alphabet contains symbolic values, terms, and predicates
- abstract transformers are symbolic (*à la* symbolic execution):

Example



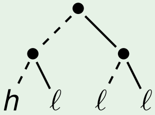
Grover’s diffusion operator

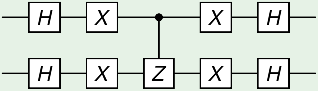
Verifying Quantum Circuits using Symbolic Amplitudes

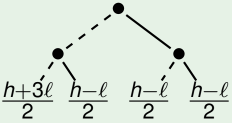
Modifications to the verification algorithm:

- tree automata $\rightsquigarrow$ symbolic 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

Verifying Quantum Circuits using Symbolic Amplitudes

- More expressive specification language
- Properties such as
  - two H gates are identity
  - Bernstein-Vazirani: no imaginary component
  - $\mathit{Grover}_{\mathit{Single}}$: $\Pr(\mathit{Correct}) > 0.9$ ($n = 20$)
  - $\mathit{Grover}_{\mathit{All}}$: $\Pr(\mathit{Correct}) > 0.9$ ($n = 9$)
  - $\mathit{Grover}_{\mathit{Iter}}$: $\Pr(\mathit{Correct})$ increased ($n = 100$)

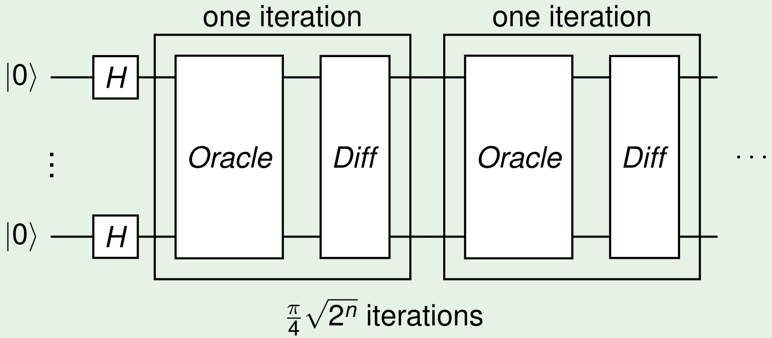
[Chen, Chung, Lengál, Lin, Tsai. AutoQ: An Automata-Based Quantum Circuit Verifier. CAV'23.]

Loop Summarization

Loop Summarization

- Some algorithms use a (fixed #iterations) loop

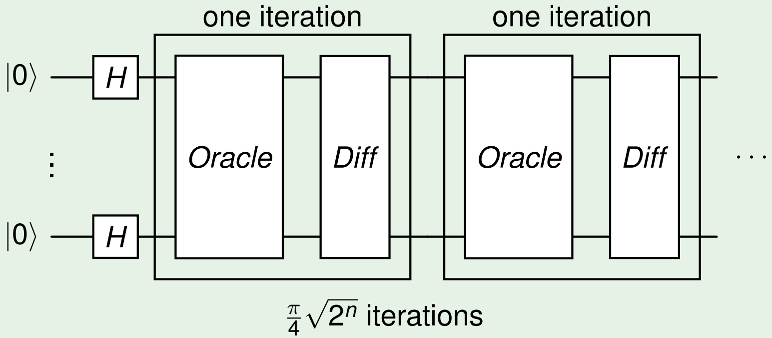
Example (Grover's algorithm)



Loop Summarization

- Some algorithms use a (fixed #iterations) loop

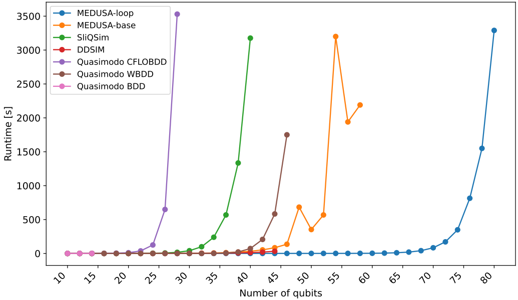
Example (Grover's algorithm)



- one can use symbolic execution (with refinement) to compute the big-step semantics of the loop body
- . . . and then just use that instead of executing the gates

Loop Summarization

- Significant speed-up of simulation of amplitude amplification
  - e.g., Grover's algorithm (below), quantum counting, period finding



- chance for more speed-up (compute the closed form)
- use for analysis (WIP)

[Chen, Chen, Jiang, Jobranová, Lengál. Accelerating Quantum Circuit Simulation with Symbolic Execution and Loop Summarization. ICCAD'24.]

Level-Synchronized Tree Automata

Level-Synchronized Tree Automata (LSTAs)

Problems with the basic TA-based framework:

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

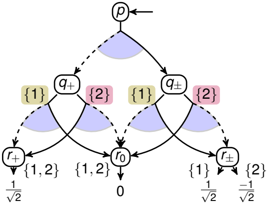
Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

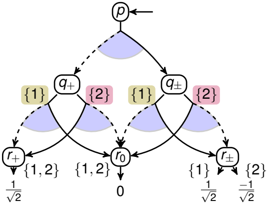
- allow synchronization across subtrees



Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

- allow synchronization across subtrees

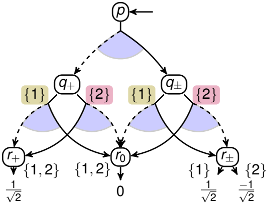


- cost of operations
  - (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
  - general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)

Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

- allow synchronization across subtrees

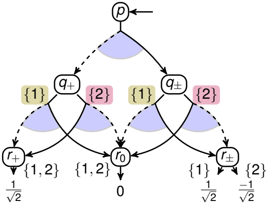


- cost of operations
  - (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
  - general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)
- incomparable to basic TAs
  - cannot express "all trees"

Level-Synchronized Tree Automata (LSTAs)

Level-Synchronized Tree Automata

- allow synchronization across subtrees



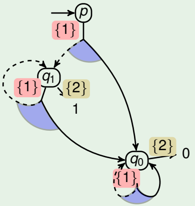
- cost of operations
  - (anti-)diagonal gates: still $\mathcal{O}(|\mathcal{A}|)$
  - general gates: $\mathcal{O}(|\mathcal{A}|^2)$ (improved from $\mathcal{O}(2^{|\mathcal{A}|})$)
- incomparable to basic TAs
  - cannot express "all trees"
- language operations:
  - emptiness: **PSPACE**-complete
 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

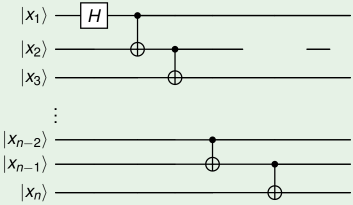
Level-Synchronized Tree Automata (LSTAs)

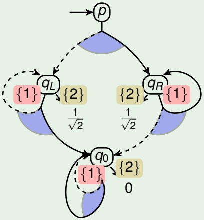
Level-Synchronized Tree Automata

- enable basic parameterized verification

Example (GHZ)







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

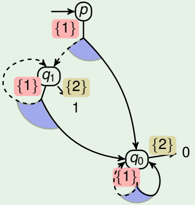
$\{\frac{1}{\sqrt{2}}|0^n\rangle + \frac{1}{\sqrt{2}}|1^n\rangle : n \geq 1\}$

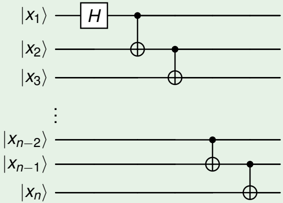
Level-Synchronized Tree Automata (LSTAs)

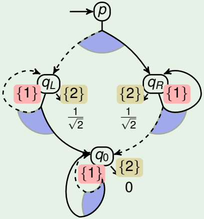
Level-Synchronized Tree Automata

- 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)

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

Verification of Quantum Circuits with Loops

Verification of Quantum Circuits with Loops

- Common structure of quantum programs:

$$\textbf{while}\ \ (M(x_i) = 0)\\ \quad C;$$

repeat-until-success, weakly measured

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

Pre: $\{1\left|0^{n+2}\right\rangle + 0\left|*\right\rangle\}$;
$H_3;\ H_4;\ \ldots;\ H_{n+2}$;
$O_{2,\ldots,(n+2)}; CK_1^2; O_{2,\ldots,(n+2)}$;
Inv: $\{v_{sol1}\left|000^n\right\rangle + v_k\left|000^{n-1}1\right\rangle + \cdots +$
$\quad v_k\left|001^n\right\rangle + v_{sol2}\left|100^n\right\rangle + 0\left|*\right\rangle\}$;
while $M_1 = 0$ **do**
$\quad \{\mathcal{G}_{2,\ldots,(n+2)}; O_{2,\ldots,(n+2)}; CK_1^2; O_{2,\ldots,(n+2)}\}$;
Post: $\{1\left|10s\right\rangle + 0\left|*\right\rangle\}$;

Verification of Quantum Circuits with Loops

- Common structure of quantum programs:

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

  repeat-until-success, weakly measured

- need to extend LSTAs with
  - measurements
  - symbolic values

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

Pre: $\{1\,|0^{n+2}\rangle + 0\,|*\rangle\}$;
$H_3;\ H_4;\ \ldots;\ H_{n+2}$;
$\mathcal{O}_{2,\ldots,(n+2)}; CK_1^2; \mathcal{O}_{2,\ldots,(n+2)}$;
Inv: $\{v_{sol1}\,|000^n\rangle + v_k\,|000^{n-1}1\rangle + \cdots +$
$v_k\,|001^n\rangle + v_{sol2}\,|100^n\rangle + 0\,|*\rangle\}$;
while $M_1 = 0$ **do**
  $\{\mathcal{G}_{2,\ldots,(n+2)}; \mathcal{O}_{2,\ldots,(n+2)}; CK_1^2; \mathcal{O}_{2,\ldots,(n+2)}\}$;
Post: $\{1\,|10s\rangle + 0\,|*\rangle\}$;

Verification of Quantum Circuits with Loops

- Common structure of quantum programs:

$$\textbf{while}\ \ (M(x_i) = 0)\quad C;$$

repeat-until-success, weakly measured

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

```
Pre: {1|0^{n+2}⟩ + 0|∗⟩};
H_3; H_4; ...; H_{n+2};
O_{2,...,(n+2)}; CK_1^2; O_{2,...,(n+2)};
Inv: {v_{sol1}|000^n⟩ + v_k|000^{n-1}1⟩ + ··· +
      v_k|001^n⟩ + v_{sol2}|100^n⟩ + 0|∗⟩};
while M_1 = 0 do
  {G_{2,...,(n+2)}; O_{2,...,(n+2)}; CK_1^2; O_{2,...,(n+2)}};
Post: {1|10s⟩ + 0|∗⟩};
```

- need to extend LSTAs with
  - measurements
  - symbolic values
- managed to verify:
  - weakly-measured Grover's algorithm
  - several repeat-until-success programs

[Chen, Chung, Hsieh, Huang, Lengál, Lin, Tsai. AutoQ 2.0: From Verification of Quantum Circuits to Verification of Quantum Programs. TACAS'25.]

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
  - a promising new formal model: alternating weighted LSTAs
    - can express $H^{\otimes n}$
    - language inclusion seems undecidable
  - a suitable transducer model?

Future Directions

- parameterized verification of more complex circuits
  - a promising new formal model: alternating weighted LSTAs
    - can express $H^{\otimes n}$
    - language inclusion seems undecidable
  - a suitable transducer model?
- a good specification language
  - expressive, user-friendly
  - can compile to LSTAs quickly

Future Directions

- parameterized verification of more complex circuits
  - a promising new formal model: alternating weighted LSTAs
    - can express $H^{\otimes n}$
    - language inclusion seems undecidable
  - a suitable transducer model?
- a good specification language
  - expressive, user-friendly
  - can compile to LSTAs quickly
- support for quantum Fourier transform
  - $\mathcal{O}(2^n)$ amplitude values
  - $\rightsquigarrow$ needs symbolic values for branches

Future Directions

- parameterized verification of more complex circuits
  - a promising new formal model: alternating weighted LSTAs
    - can express $H^{\otimes n}$
    - language inclusion seems undecidable
  - a suitable transducer model?
- a good specification language
  - expressive, user-friendly
  - can compile to LSTAs quickly
- support for quantum Fourier transform
  - $\mathcal{O}(2^n)$ amplitude values
  - $\leadsto$ needs symbolic values for branches
- equivalence checking of parameterized circuits
  - oracle-based circuits
  - dynamic circuits
  - various notions of equivalence

Future Directions

- parameterized verification of more complex circuits
  - a promising new formal model: alternating weighted LSTAs
    - can express $H^{\otimes n}$
    - language inclusion seems undecidable
  - a suitable transducer model?
- a good specification language
  - expressive, user-friendly
  - can compile to LSTAs quickly
- support for quantum Fourier transform
  - $\mathcal{O}(2^n)$ amplitude values
  - $\leadsto$ needs symbolic values for branches
- 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!