AN AUTOMATA-BASED APPROACH FOR SCALABLE VERIFICATION OF QUANTUM CIRCUITS

Yu-Fang Chen¹, Kai-Min Chung¹, Ondrej Lengal², Jyun-Ao Lin¹, Wei-Lun Tsai¹, Di-De Yen¹

Academia Sinica, Taiwan¹, Brno University of Technology, Czech Republic²

QPL 2023, Paris
Outline

- Motivation
- Quantum Circuit Verification
- Evaluation
Why Quantum Circuit Verification?

- Increasing complexity of circuits
- Infeasibility of testing
- Challenge of probabilistic features
State-of-the-Art

We focus on **fully** automatic approaches:

- **Quantum Simulation**
  - low coverage

- **Quantum Abstraction Interpretation**
  - cannot catch bugs

- **Quantum Model Checking**
  - only work for small examples

- **Circuit Equivalence Checking**
  - inflexible for user custom properties
Outline

- Minimal Quantum Background and Motivation
- Quantum Circuit Verification with Pre and Post-Conditions
- Evaluation
Tree as a Quantum State

- A 3-bit quantum state
Quantum Gate and Tree Transformation

- An example of apply X gate (negation) on qubit $x_1$. 
Classical Hoare triple

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

\[
\{P\} \ S \ \{Q\}
\]

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

- For any predicates \( P \) and \( Q \) and any program \( S \),

\[
\{ P \} \ S \ \{ Q \}
\]

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

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

\[ \{P\} \ C \ \{Q\} \]

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

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

{P} C {Q}
Quantum Circuit Verification

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

From automata theory:
Set of words $\rightarrow$ Regular language (Finite automata)
Set of trees $\rightarrow$ Regular tree language (Tree automata)
Tree Automata Encoding of Quantum States

$q \xrightarrow{x_1} (q_1, q_0)$

$q_0 \xrightarrow{x_2} (q_2, q_2)$

$q_1 \xrightarrow{x_2} (q_3, q_2)$

$q_2 \xrightarrow{0} ()$

$q_3 \xrightarrow{1} ()$

Figure 1: The TA of $|00\rangle$. 
Tree Automata Encoding of Quantum States

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

\[ \{ |000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle \} \]
TA as Compact Representation of Quantum States

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

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

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

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

says that if $C$ is started in (a state satisfying) $P$, then it terminates in $Q$. 
Two Approaches for TA Gate Operations

- **Permutation-based approach:**
  - Faster, but works for a smaller set of gates.
  - Done by directly modifying TA transitions.

- **Composition-based approach**
  - Slower, but complete for universal quantum computing.
Example of an Operation: X gate on qubit 2.
Example of an Operation: Z, S, T gates on qubit 1.

- Multiply the right subtree of $x_1$ with some constant $c$. 
Two Approaches for TA Gate Operations

- Permutation-based approach:
  - Faster, but works for a smaller set of gates.
  - Done by directly modifying TA transitions.

- Composition-based approach
  - Slower, but complete for universal quantum computing.
1. Projection ($\overline{T_{x_2}}$ & $T_{x_2}$)
1. Projection \((T_{x_2} & T_{x_2})\)
2. Restriction \((B_{x_2} \cdot \& \ B_{\overline{x_2}} \cdot )\)

\[ B_{x_2} \cdot T_{\overline{x_2}} \]

\[
\begin{pmatrix}
0 & 0 \\
0 & 1
\end{pmatrix}
\quad
\begin{pmatrix}
1 & 0 \\
0 & 0
\end{pmatrix}
\]
3. Binary Operation ($+ \& -$)

\[ B_{x_2} \cdot T_{\overline{x_2}} + B_{\overline{x_2}} \cdot T_{x_2} \]
4. Constant Multiplication ($c \cdot$)

\[ (-1) \cdot T \]
Tree Operations

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

<table>
<thead>
<tr>
<th>Gate</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_t$</td>
<td>$B_{X_t} \cdot T_{X_t} + B_{X_t'} \cdot T_{X_t}$</td>
</tr>
<tr>
<td>$Y_t$</td>
<td>$\omega^2 \cdot (B_{X_t} \cdot T_{X_t} - B_{X_t'} \cdot T_{X_t})$</td>
</tr>
<tr>
<td>$Z_t$</td>
<td>$B_{X_t} \cdot T - B_{X_t} \cdot T$</td>
</tr>
<tr>
<td>$H_t$</td>
<td>$(T_{X_t} + B_{X_t} \cdot T_{X_t} - B_{X_t'} \cdot T) / \sqrt{2}$</td>
</tr>
<tr>
<td>$S_t$</td>
<td>$B_{X_t} \cdot T + \omega^2 \cdot B_{X_t'} \cdot T$</td>
</tr>
<tr>
<td>$T_t$</td>
<td>$B_{X_t} \cdot T + \omega \cdot B_{X_t'} \cdot T$</td>
</tr>
<tr>
<td>$Rx(\frac{\pi}{2})_t$</td>
<td>$(T - \omega^2 \cdot (B_{X_t} \cdot T_{X_t} + B_{X_t'} \cdot T_{X_t})) / \sqrt{2}$</td>
</tr>
<tr>
<td>$Ry(\frac{\pi}{2})_t$</td>
<td>$(T_{X_t} + B_{X_t} \cdot T - B_{X_t'} \cdot T_{X_t}) / \sqrt{2}$</td>
</tr>
<tr>
<td>$CNOT^c_t$</td>
<td>$B_{X_c} \cdot T + B_{X_c} \cdot B_{X_t} \cdot T_{X_t} + B_{X_c} \cdot B_{X_t} \cdot T_{X_t}$</td>
</tr>
<tr>
<td>$CZ^c_t$</td>
<td>$B_{X_c} \cdot T + B_{X_t} \cdot T - B_{X_c} \cdot B_{X_t} \cdot T - B_{X_c} \cdot B_{X_t} \cdot T$</td>
</tr>
<tr>
<td>$Toffoli^c_t$</td>
<td>$B_{X_c} \cdot T + B_{X_c} \cdot T - B_{X_c} \cdot B_{X_c'} \cdot T + B_{X_t} \cdot B_{X_c} \cdot B_{X_c'} \cdot T_{X_t} + B_{X_t} \cdot B_{X_c} \cdot B_{X_c'} \cdot T_{X_t}$</td>
</tr>
<tr>
<td>$Fredkin^c_{t,t'}$</td>
<td>$B_{X_c} \cdot T + B_{X_t} \cdot B_{X_t'} \cdot B_{X_c} \cdot T + B_{X_t} \cdot B_{X_t'} \cdot B_{X_c} \cdot T_{X_t} + B_{X_c} \cdot T_{X_t} + B_{X_t} \cdot B_{X_t'} \cdot B_{X_c} \cdot T_{X_t} + B_{X_c} \cdot T_{X_t} + B_{X_t} \cdot B_{X_t'} \cdot B_{X_c} \cdot T_{X_t} + B_{X_c} \cdot T_{X_t} + B_{X_t} \cdot B_{X_t'} \cdot B_{X_c} \cdot T_{X_t}$</td>
</tr>
</tbody>
</table>
X gate operating on qubit 2

\[ B_{x_2} \cdot T_{x_2} + B_{\overline{x_2}} \cdot T_{\overline{x_2}} \]
Lift these operations to TA

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

- **Hoare Triple**: \{P\} C \{Q\}, ideally
  - P, Q: we use regular tree language as predicates
  - C: a sequence of quantum gates

- **Our approach**
  - Compute C(P), the set of states after executing C.
  - Test if C(P) \subseteq Q (via standard TA algorithms).
Overview

Tree automata

```
 Gol
 q[1];
 cx q[0],q[1];
 t q[0];
 t q[1];
 t q[2];
tdg q[3];
 cx q[0],q[1];
 cx q[2],q[3];
 cx q[3],q[0];
```
Overview

P

P1

Q

h q[3];
cx q[2], q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0], q[1];
cx q[2], q[3];
cx q[3], q[0];
Overview

P^2

P

Q

```
h q[3];
cx q[2], q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0], q[1];
cx q[2], q[3];
cx q[3], q[0];
```
Overview

\[ P \]

\[ P^3 \]

\[ Q \]

```cpp
h q[3];
cx q[2], q[3];
t q[0];
t q[1];
t q[2];
tdg q[3];
cx q[0], q[1];
cx q[2], q[3];
cx q[3], q[0];
```
Overview

\[ P^{10} \subseteq Q \]

(via standard TA algorithms)
Outline

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

- BV, Grover-Sing: $|P| = 1$
- MCToffoli, Grover-All: $|P| \gg 1$
Experiment – Bug Hunting

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

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

#### AutoQ, Feynman, Qcec

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#q</th>
<th>#G</th>
<th>Time</th>
<th>Bug?</th>
<th>Time</th>
<th>Bug?</th>
<th>Time</th>
<th>Bug?</th>
</tr>
</thead>
<tbody>
<tr>
<td>csun_mux_9</td>
<td>30</td>
<td>141</td>
<td>0.5s</td>
<td>T</td>
<td>6.0s</td>
<td>—</td>
<td>1m1s</td>
<td>F</td>
</tr>
<tr>
<td>gf2_10_mult</td>
<td>30</td>
<td>348</td>
<td>1.5s</td>
<td>T</td>
<td>0.5s</td>
<td>—</td>
<td>1.6s</td>
<td>—</td>
</tr>
<tr>
<td>gf2_16_mult</td>
<td>48</td>
<td>876</td>
<td>9.3s</td>
<td>T</td>
<td>3.6s</td>
<td>—</td>
<td>1m25s</td>
<td>T</td>
</tr>
<tr>
<td>gf2_32_mult</td>
<td>96</td>
<td>3,323</td>
<td>2m0s</td>
<td>T</td>
<td>51s</td>
<td>—</td>
<td>2m52s</td>
<td>T</td>
</tr>
<tr>
<td>ham15-high</td>
<td>20</td>
<td>1,799</td>
<td>7.5s</td>
<td>T</td>
<td>3m50s</td>
<td>—</td>
<td>5.6s</td>
<td>T</td>
</tr>
<tr>
<td>mod_adder_1024</td>
<td>28</td>
<td>1,436</td>
<td>10s</td>
<td>T</td>
<td>8.7s</td>
<td>—</td>
<td>1m2s</td>
<td>T</td>
</tr>
<tr>
<td>hwb10</td>
<td>16</td>
<td>31,765</td>
<td>1m49s</td>
<td>T</td>
<td>timeout</td>
<td>50.6s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>hwb11</td>
<td>15</td>
<td>87,790</td>
<td>4m31s</td>
<td>T</td>
<td>timeout</td>
<td>48.8s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>hwb12</td>
<td>20</td>
<td>171,483</td>
<td>13m43s</td>
<td>T</td>
<td>timeout</td>
<td>1m30s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>hwb8</td>
<td>12</td>
<td>6,447</td>
<td>16s</td>
<td>T</td>
<td>timeout</td>
<td>43.6s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>qcla_adder_10</td>
<td>36</td>
<td>182</td>
<td>1.7s</td>
<td>T</td>
<td>1.0s</td>
<td>—</td>
<td>1m5s</td>
<td>F</td>
</tr>
<tr>
<td>qcla_mod_7</td>
<td>26</td>
<td>295</td>
<td>2.0s</td>
<td>T</td>
<td>1m28s</td>
<td>—</td>
<td>59.0s</td>
<td>F</td>
</tr>
</tbody>
</table>

#### Random

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#q</th>
<th>#G</th>
<th>Time</th>
<th>Bug?</th>
<th>Time</th>
<th>Bug?</th>
<th>Time</th>
<th>Bug?</th>
</tr>
</thead>
<tbody>
<tr>
<td>35a</td>
<td>35</td>
<td>106</td>
<td>2.6s</td>
<td>T</td>
<td>0.2s</td>
<td>—</td>
<td>1m4s</td>
<td>F</td>
</tr>
<tr>
<td>35b</td>
<td>35</td>
<td>106</td>
<td>1.3s</td>
<td>T</td>
<td>0.1s</td>
<td>T</td>
<td>1m5s</td>
<td>F</td>
</tr>
<tr>
<td>35c</td>
<td>35</td>
<td>106</td>
<td>1.1s</td>
<td>T</td>
<td>0.1s</td>
<td>T</td>
<td>1m7s</td>
<td>T</td>
</tr>
<tr>
<td>35d</td>
<td>35</td>
<td>106</td>
<td>1.1s</td>
<td>T</td>
<td>0.1s</td>
<td>T</td>
<td>1m5s</td>
<td>T</td>
</tr>
<tr>
<td>35e</td>
<td>35</td>
<td>106</td>
<td>1.0s</td>
<td>T</td>
<td>0.1s</td>
<td>—</td>
<td>1m4s</td>
<td>T</td>
</tr>
<tr>
<td>35f</td>
<td>35</td>
<td>106</td>
<td>2.0s</td>
<td>T</td>
<td>0.2s</td>
<td>T</td>
<td>1m5s</td>
<td>F</td>
</tr>
<tr>
<td>35g</td>
<td>35</td>
<td>106</td>
<td>1.0s</td>
<td>T</td>
<td>0.2s</td>
<td>—</td>
<td>1m7s</td>
<td>T</td>
</tr>
<tr>
<td>35h</td>
<td>35</td>
<td>106</td>
<td>1.2s</td>
<td>T</td>
<td>0.2s</td>
<td>—</td>
<td>2.0s</td>
<td>—</td>
</tr>
<tr>
<td>35i</td>
<td>35</td>
<td>106</td>
<td>1.2s</td>
<td>T</td>
<td>0.3s</td>
<td>T</td>
<td>1m6s</td>
<td>T</td>
</tr>
<tr>
<td>35j</td>
<td>35</td>
<td>106</td>
<td>1.4s</td>
<td>T</td>
<td>0.2s</td>
<td>—</td>
<td>1m6s</td>
<td>F</td>
</tr>
<tr>
<td>add16_174</td>
<td>49</td>
<td>65</td>
<td>3.1s</td>
<td>T</td>
<td>timeout</td>
<td>1m16s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>add2_183</td>
<td>97</td>
<td>129</td>
<td>25s</td>
<td>T</td>
<td>timeout</td>
<td>2m2s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>add32_184</td>
<td>193</td>
<td>257</td>
<td>2m8s</td>
<td>T</td>
<td>timeout</td>
<td>2.6s</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>avg8_325</td>
<td>320</td>
<td>1,758</td>
<td>22m42s</td>
<td>T</td>
<td>timeout</td>
<td>2.2s</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>bw_291</td>
<td>87</td>
<td>308</td>
<td>11s</td>
<td>T</td>
<td>15s</td>
<td>T</td>
<td>2m10s</td>
<td>T</td>
</tr>
<tr>
<td>cycle10_293</td>
<td>39</td>
<td>79</td>
<td>0.5s</td>
<td>T</td>
<td>0.5s</td>
<td>T</td>
<td>1m8s</td>
<td>T</td>
</tr>
<tr>
<td>e64_bdd_295</td>
<td>195</td>
<td>388</td>
<td>50s</td>
<td>T</td>
<td>timeout</td>
<td>1.5s</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>en3p_296</td>
<td>206</td>
<td>648</td>
<td>2m5s</td>
<td>T</td>
<td>1m36s</td>
<td>T</td>
<td>2.6s</td>
<td>—</td>
</tr>
<tr>
<td>haml5_298</td>
<td>45</td>
<td>154</td>
<td>1.3s</td>
<td>T</td>
<td>0.9s</td>
<td>T</td>
<td>1m15s</td>
<td>T</td>
</tr>
<tr>
<td>mod5adder_306</td>
<td>32</td>
<td>97</td>
<td>0.8s</td>
<td>T</td>
<td>1.1s</td>
<td>T</td>
<td>1m2s</td>
<td>T</td>
</tr>
<tr>
<td>rd84_313</td>
<td>34</td>
<td>105</td>
<td>0.9s</td>
<td>T</td>
<td>1.5s</td>
<td>T</td>
<td>1m5s</td>
<td>T</td>
</tr>
</tbody>
</table>

#### RevLib

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#q</th>
<th>#G</th>
<th>Time</th>
<th>Bug?</th>
<th>Time</th>
<th>Bug?</th>
<th>Time</th>
<th>Bug?</th>
</tr>
</thead>
<tbody>
<tr>
<td>urr1_149</td>
<td>9</td>
<td>11,555</td>
<td>40s</td>
<td>T</td>
<td>timeout</td>
<td>45.2s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>urr2_152</td>
<td>8</td>
<td>5,031</td>
<td>14s</td>
<td>T</td>
<td>21m39s</td>
<td>T</td>
<td>36.4s</td>
<td>T</td>
</tr>
<tr>
<td>urr3_155</td>
<td>10</td>
<td>26,469</td>
<td>1m29s</td>
<td>T</td>
<td>timeout</td>
<td>45.1s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>urr4_187</td>
<td>11</td>
<td>32,005</td>
<td>2m10s</td>
<td>T</td>
<td>timeout</td>
<td>46.6s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>urr5_158</td>
<td>9</td>
<td>10,277</td>
<td>26s</td>
<td>T</td>
<td>timeout</td>
<td>38.2s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>urr6_160</td>
<td>15</td>
<td>10,741</td>
<td>1m6s</td>
<td>T</td>
<td>timeout</td>
<td>48.4s</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>hwb6_301</td>
<td>46</td>
<td>160</td>
<td>2.5s</td>
<td>T</td>
<td>2.7s</td>
<td>T</td>
<td>1m13s</td>
<td>T</td>
</tr>
<tr>
<td>hwb7_302</td>
<td>73</td>
<td>282</td>
<td>10s</td>
<td>T</td>
<td>1.5s</td>
<td>T</td>
<td>1m39s</td>
<td>T</td>
</tr>
<tr>
<td>hwb8_303</td>
<td>112</td>
<td>450</td>
<td>34s</td>
<td>T</td>
<td>43s</td>
<td>T</td>
<td>2m25s</td>
<td>T</td>
</tr>
<tr>
<td>hwb9_304</td>
<td>170</td>
<td>700</td>
<td>1m48s</td>
<td>T</td>
<td>2m29s</td>
<td>T</td>
<td>2.0s</td>
<td>—</td>
</tr>
</tbody>
</table>

**GitHub Repositories:**

- [meamy/feynman](https://github.com/meamy/feynman)
- [cda-tum/qcec](https://github.com/cda-tum/qcec)
Extension: Symbolic Tree

A Symbolic Tree (State), s

The state after executing C, C(s)
Predicate Tree as Specifications for Relational Properties
Experiment – Symbolic

\( x \in \{a, b, c, \ldots, h\} \) is a complex number and \( |x|^2 \) is the probability.
Summary:
• Interesting link between automata and quantum computing.
• So far only basic TA has been tried, there are many more possibilities.
• The main reason for efficiency is the compact structure of TA.