Quantum automaton implementation. Circuit optimization

Aliya Khadieva

University of Latvia
QWorld Association

2022/04/27

Areas of work

Automata models
Affine finite automata
Quantum finite automata

Affine finite automata

Affine finite automata - Generalization of QFA

- *SUBSETSUM* problem is verified by an integer-valued Affine Nondeterministic Automaton such that every member is accepted with probability 1 and every non-member is accepted with probability at most $\frac{1}{2t+1}$ for some $t \in \mathbb{Z}^+$.
- Every unary language $L \subseteq \{a\}^*$ is verified by an Affine Nondeterministic Automaton with error bound 0.155.

Khadieva, A., Yakaryılmaz, A. (2021, October). Affine automata verifiers. In International Conference on Unconventional Computation and Natural Computation (pp. 84-100). Springer, Cham.

Quantum finite automata

Existing Results on Quantum finite automata

Existing Results

- Let p be a prime
- $MOD_p = \{a^j \mid j \text{ is divisible by } p\}$

Existing Results

- Let p be a prime
- $MOD_p = \{a^j \mid j \text{ is divisible by } p\}$

Any 1-way *DFA* recognizing MOD_p has at least p states.

There is much more efficient *QFA*

Existing Results

- Let p be a prime
- $MOD_p = \{a^j \mid j \text{ is divisible by } p\}$

Any 1-way *DFA* recognizing MOD_p has at least p states.

There is much more efficient *QFA*

Ambainis, Freivalds. 1998

MOD_p can be recognized by a *QFA* with $O(\log p)$ states.
Big-O constant depends on required probability of correct answer.
For $x \in MOD_p$ the answer is always correct with probability 1.

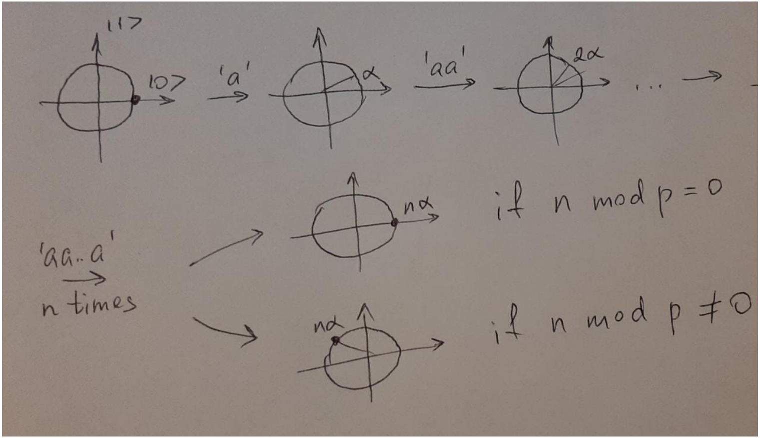
Ambainis, Nahimovs, 2008

For any $\epsilon > 0$, there is a *QFA* with $4\frac{\log 2p}{\epsilon}$ states recognizing MOD_p with probability at least $1 - \epsilon$.

QFA construction

Initially, $|q_0 q_1 \dots q_{\log d}\rangle|q_{target}\rangle = |000..0\rangle|0\rangle$
On the left end-marker,
$|000..0\rangle|0\rangle \rightarrow H^{\log d} \otimes I \rightarrow = \frac{1}{\sqrt{d}}(|0\rangle + |1\rangle + \dots + |d\rangle)|0\rangle$
The automaton U reading an input symbol rotates $|q_{target}\rangle$ on angles $\alpha_i = \frac{2\pi k_i}{p}$ for each $i \in \{0...d\}$, where i corresponds to the state of the quantum register.

QFA construction



For each i, $\alpha_i = \frac{2\pi k_i}{p}$

Recognition

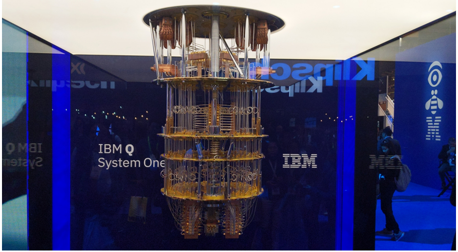
Claim 1

If the input word is a^j and j is divisible by p, then the automaton accepts with probability 1.

Claim 2

If the input word is a^j and j is NOT divisible by p, then the automaton accepts with probability
$\frac{1}{d^2}\left(\cos\frac{2\pi k_1 j}{p} + \cos\frac{2\pi k_2 j}{p} + \cdots + \cos\frac{2\pi k_d j}{p}\right)^2$

QFA implementation



Qiskit

QFA implementation

Let $k_1, \cdots, k_d$ be a sequence of integers, where $d = c \log p$

d states for determining d transformations

for each 'a' $|q_{target}\rangle$ is rotated on angles

state 0	state 1	....	state d
$\frac{2\pi k_0}{p}$	$\frac{2\pi k_1}{p}$		$\frac{2\pi k_d}{p}$

QFA implementation

Let $k_1, \cdots, k_d$ be a sequence of integers, where $d = c \log p$

d states for determining d transformations

for each 'a' $|q_{target}\rangle$ is rotated on angles

state 0	state 1		state d
$\frac{2\pi k_0}{p}$	$\frac{2\pi k_1}{p}$	....	$\frac{2\pi k_d}{p}$

How to differentiate these states ?

QFA implementation

Let $k_1, \cdots, k_d$ be a sequence of integers, where $d = c \log p$

	state 0	state 1		state d
d states for determining d transformations				
for each 'a' $\|q_{target}\rangle$ is rotated on angles	$\frac{2\pi k_0}{p}$	$\frac{2\pi k_1}{p}$	....	$\frac{2\pi k_d}{p}$

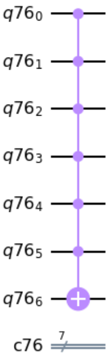
How to differentiate these states?

Transform the state i to get a state $|1..11\rangle$ and apply control rotation on an angle α_i

- $|000\rangle \xrightarrow{X \otimes X \otimes X} |111\rangle$
- $|010\rangle \xrightarrow{X \otimes X \otimes X} |101\rangle \neq |111\rangle$
- $|011\rangle \xrightarrow{X \otimes X \otimes X} |100\rangle \neq |111\rangle$

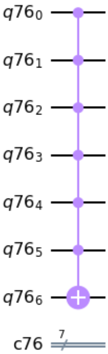
QFA implementation

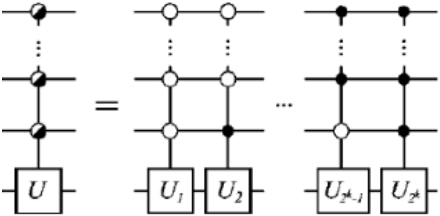
The implementation of the multi-qubit-controlled rotation is VERY expensive !



QFA implementation

The implementation of the multi-qubit-controlled rotation is VERY expensive !

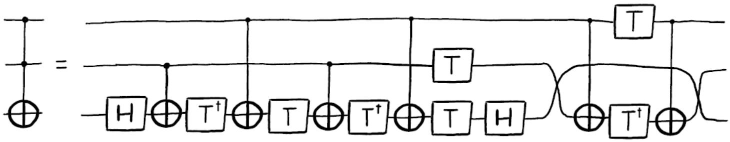




MCXGate

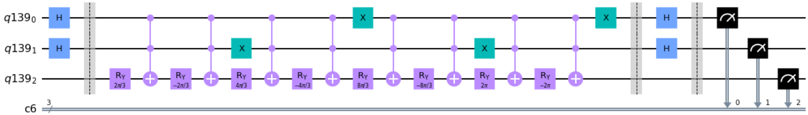
- MXGate() - recursively implemented multi-qubit-controlled cnot operation
- An amount of cnot-gates of the circuit with n control qubits
- $S_{cx}(n) = 4 * S_{cx}(n/2) = 4 * (4 * (\cdots 4 * S_{cx}(2)) \cdots) = 4^{\log n - 1} * 6 \approx n^2$ cnot operations
- denote the computational complexity of a cnot operation with one controller by $S_{cx}(1)$
- $S_{cx}(2) = 6 * S_{cx}(1)$
- $S_{cx}(3) = 14 * S_{cx}(1)$

Toffoli gate decomposition



$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$

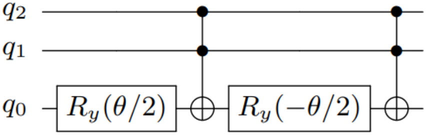
QFA implementation



Grey codes :

000		000
001		001
010		011
011	$\rightarrow$	010
100		110
101		111
110		101
111		100

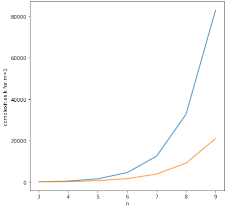
controlled $R_y(\theta)$ decomposition



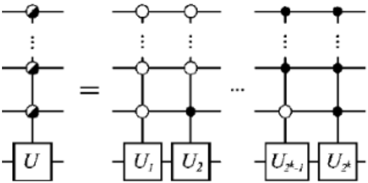
$$\Sigma(n) = 2^n S_{rot}(n) = 2^n * 2 * S_{cx}(n) \approx 2^{n+1} n^2 S_{cx}(1)$$

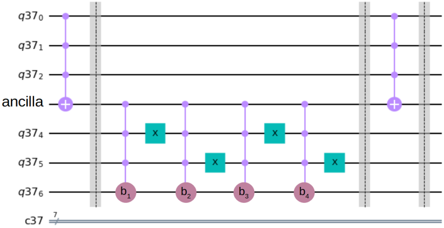
Circuit optimization. Ancilla qubit

- k is a number of 'active' qubits
- $\Sigma(n,k) = 2^{n-k}(2S_{cx}(n-k) + 2^k S_{rot}(k+1)) \approx 2^{n-k}(2(n-k)^2 + 2\cdot 2^k \cdot (k+1)^2) = 2^{n-k+1}(n-k)^2 + 2^{n+1}(k+1)^2$

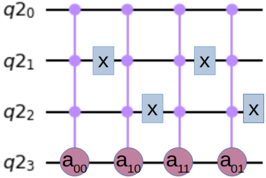


$$\Sigma(n,k) << \Sigma_{old}(n)$$

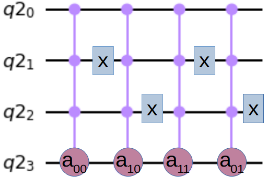


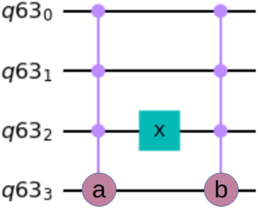


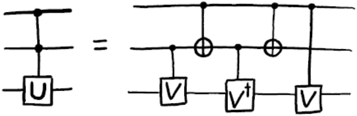
Circuit optimization.Subcircuit decomposition



Circuit optimization.Subcircuit decomposition



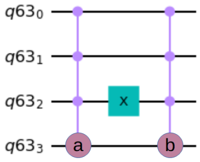




U is rotation on some γ and V is rotation on $\gamma/2$
$U = V^2$

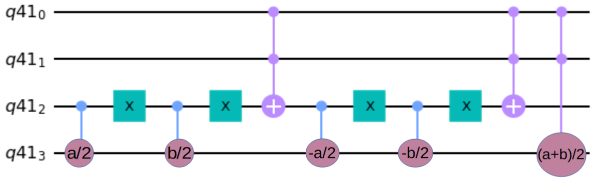
Circuit optimization.Subcircuit decomposition

For 2 angles
$S(3) = 2S_{rot}(3) = 4S_{cx}(3) = 56 \cdot S_{cx}(1)$



$S(2) = 24 \cdot S_{cx}(1)$

$S'(3) = 4S_{rot}(1) + 2S_{cx}(2) + S_{rot}(2) = 32 \cdot S_{cx}(1)$

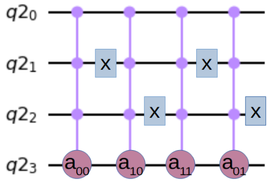


$S'(2) = 12 \cdot S_{cx}(1)$

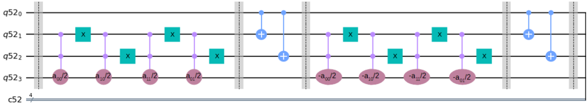
Circuit optimization.Subcircuit decomposition

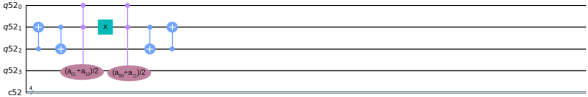
For 4 angles

$$S(3) = 4S_{rot}(3) =$$
$$= 112 \cdot S_{cx}(1)$$



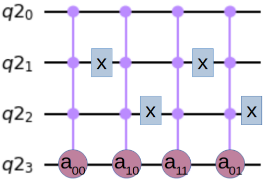
$$S'(3) = 8S_{rot}(2) + 8S_{cx}(1) + 2S_{rot}(2) =$$
$$= 128 \cdot S_{cx}(1)$$



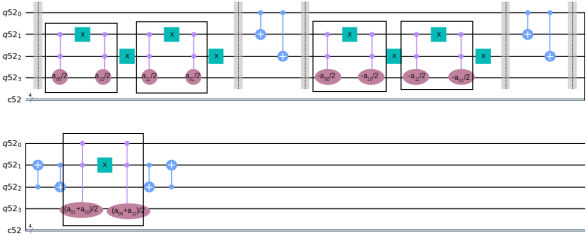


Circuit optimization.Subcircuit decomposition

For 4 angles

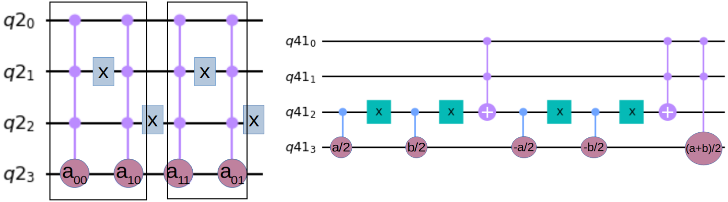


$S''(3) = 68 \cdot S_{cx}(1)$



Circuit optimization.Subcircuit decomposition

For 4 angles $S(3) = 2 \cdot S_{2angles}(3) = 64 \cdot S_{cx}(1)$



- If 2^{n-1} angles and n controllers, then
- $S(n) = 2^n(1 + (n-1)^2) \cdot S_{cx}(1)$
- The whole circuit complexity is
- $\Sigma_{new}(n,k) = 2^{n-k+1}(n-k)^2 + 2^{n+1}(1+k^2)$,
- when $\Sigma_{old}(n,k) \approx 2^{n-k+1}(n-k)^2 + 2^{n+1}(k+1)^2$
- $\Sigma_{old}(n,k) - \Sigma_{new}(n,k) = 2k \cdot 2^{n+1}$

Thank you !