QRAT: The Reachability Analysis Tool for Quantum Programs

Canh Minh Do Kazuhiro Ogata

{canhdo,ogata}@jaist.ac.jp

Japan Advanced Institute of Science and Technology (JAIST)

Physically Presented at 2025 JAIST-Vietnam Research Exchange Meeting (JVREM 2025)

February 27-28, 2025, Japan

About Me

Background:

4/2023–Present	Assistant Professor in Computing Science, JAIST
10/2022	Postdoctoral Researcher, JAIST
9/2022	Ph.D. in Information Science, JAIST
9/2019	M.S. in Information Science, JAIST

Research interests: Formal specification and verification of concurrent/distributed systems for both conventional and emerging technologies.

Contents

Contents

Introduction

- Quantum computing uses the laws of quantum mechanics to solve complex problems beyond the capabilities of classical computing, such as Shor's fast algorithms for integer factoring and Grover's algorithm for searching an unstructured database.
- Several quantum programming languages and platforms are introduced by leading companies: IBM's Qiskit, Google's Cirq, and Microsoft's Q#.
- Due to radically different principles of quantum mechanics, the likelihood of programming errors in quantum programs is even higher compared to classical ones.
- Therefore, ensuring the correctness of quantum programs is crucial in the emerging quantum era.

Reachability Analysis of Quantum Programs

- We introduce QRAT, the first reachability analysis tool for quantum programs, leveraging a state-of-the-art decision diagram developed in MQT Core[1] for quantum computing.
- We use QRAT to confirm the correctness of Quantum Teleportation and Grover's search algorithm to demonstrate its effectiveness and practicality.

	QReach (2024)[2]	QRAT (2025)
System Model	Quantum Markov Chains	Quantum While Programs
Quantum Simulation	CFLOBDD	DD in MQT Core
State Representation	Mixed States	Pure States
Property Interpretation	Hard	Easy
Probability Support	No	Yes (partial)
Usability	Complex	Simple
Output	Closed Subspaces	Reachable States

[1] MQT Core - The Backbone of the Munich Quantum Toolkit (MQT)

[2] Aochu Dai and Mingsheng Ying. "QReach: A Reachability Analysis Tool for Quantum Markov Chains". In: *Computer Aided Verification*. Cham: Springer Nature Switzerland, 2024, pp. 520–532. DOI: 10.1007/978-3-031-65633-0_23.

Contents

Hilbert Spaces

- A Hilbert space $\mathcal{H}$ usually serves as the state space of a quantum system that is a complex vector space equipped with an inner product that satisfies some properties.
- Quantum states in the *n*-qubit systems can be represented by unit complex vectors over an orthonormal basis in 2^n-space $\mathbb{C}^{2^n}$.
- The orthogonal basis called computational basis in the one-qubit system $\mathbb{C}^2$ is the set $\{|0\rangle, |1\rangle\}$ that consists of the column vectors $|0\rangle = (1,0)^T$ and $|1\rangle = (0,1)^T$.
- Any single qubit $|\psi\rangle$ can be expressed as a superposition of $|0\rangle$ and $|1\rangle$ as follows:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = (\alpha, \beta)^T$$

  where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.
- For multiple qubits, the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ of Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ is defined as a vector space consisting of linear combinations of the vectors $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$:

$$|\psi_1\psi_2\rangle = |\psi_1\rangle |\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

- Two or more qubits systems may be in *entangled* states, meaning that quantum states are correlated and inseparable (e.g., EPR states).

Unitary Operators

- Quantum computation is represented by unitary operators (also called quantum gates).
- For example, the Hadamard gate H and Pauli gates X, Y, and Z are quantum gates on the one-qubit system $\mathbb{C}^2$ and are defined as follows:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

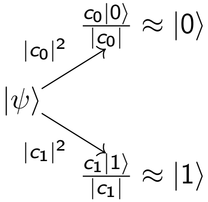
- Two typical quantum gates on the two-qubit systems $\mathbb{C}^4$ are the controlled-X gate (also called the controlled-NOT gate) CX and the swap gate $SWAP$ are defined by

$$CX = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X,$$
$$SWAP = CX(I \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1|)CX,$$

where I denotes the identity matrix of size 2×2.

Measurement

- Measurement is a completely different process from applying quantum gates. Here we roughly explain specific binary projective measurements.
- For the general definition of projective measurement, see the famous textbook of quantum computation and quantum information[3].
- Observe that measurement operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$ are projectors.
- After executing the measurement $\{M_0, M_1\}$, a current state $|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle$ is collapsed into either $\frac{M_0|\psi\rangle}{|c_0|}$ with probability $|c_0|^2$ or into $\frac{M_1|\psi\rangle}{|c_1|}$ with probability $|c_1|^2$.



[3] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

Contents

Syntax of Quantum **While**-Programs

Definition 3.1 (Syntax)

The quantum **while**-programs are defined by the grammar:

$$
\begin{aligned}
S ::= \texttt{skip} \mid \overline{q} := U[\overline{q}] \mid S_1 \,;\, S_2 \\
\mid \textsf{if } M[q] = 1 \textsf{ then } S_1 \textsf{ else } S_2 \textsf{ fi} \\
\mid \textsf{while } M[q] = 1 \textsf{ do } S \textsf{ od}
\end{aligned}
$$

Operational Semantics of Quantum **While**-Programs

(SK) $\langle \texttt{skip}, |\psi\rangle\rangle \to \langle\downarrow, |\psi\rangle\rangle$

(UT) $\langle \overline{q} := U[\overline{q}], |\psi\rangle\rangle \to \langle\downarrow, U\,|\psi\rangle\rangle$

The actual unitary matrix U applies to $|\psi\rangle$ w.r.t $\overline{q}$ on the right-hand side.

(SC) $$\frac{\langle S_1, |\psi\rangle\rangle \to \langle S_1', |\psi'\rangle\rangle}{\langle S_1 \,;\, S_2, |\psi\rangle\rangle \to \langle S_1' \,;\, S_2, |\psi'\rangle\rangle}$$

(IF0) $\langle \text{if } M[q] = 1 \text{ then } S_1 \text{ else } S_2 \text{ fi}, |\psi\rangle\rangle \to \langle S_2, M_0\,|\psi\rangle\rangle$

For result 0, $|\psi\rangle$ is collapsed according to M_0 of measurement $M = \{M_0, M_1\}$.

(IF1) $\langle \text{if } M[q] = 1 \text{ then } S_1 \text{ else } S_2 \text{ fi}, |\psi\rangle\rangle \to \langle S_1, M_1\,|\psi\rangle\rangle$

For result 1, $|\psi\rangle$ is collapsed according to M_1 of measurement $M = \{M_0, M_1\}$.

(L0) $\langle \text{while } M[q] = 1 \text{ do } S \text{ od}, |\psi\rangle\rangle \to \langle\downarrow, M_0\,|\psi\rangle\rangle$

For outcome 0 of measurement $M = \{M_0, M_1\}$.

(L1) $\langle \text{while } M[q] = 1 \text{ do } S \text{ od}, |\psi\rangle\rangle \to \langle S \,;\text{while } M[q] = 1 \text{ do } S \text{ od}, M_1\,|\psi\rangle\rangle$

For outcome 1 of measurement $M = \{M_0, M_1\}$.

Reachability Problem for Quantum Programs

Given a quantum program S and an initial state $|\psi\rangle \in \mathcal{H}_S$, $\langle S', |\psi'\rangle\rangle$ is reachable from $\langle S, |\psi\rangle\rangle$ if and only if there exist configurations $\langle S_1, |\psi_1\rangle\rangle, \ldots, \langle S_n, |\psi_n\rangle\rangle$ for $n \geq 0$ such that

$$\langle S, |\psi\rangle\rangle = \langle S_1, |\psi_1\rangle\rangle \rightarrow \ldots \rightarrow \langle S_n, |\psi_n\rangle\rangle = \langle S', |\psi'\rangle\rangle$$

holds. In the case where $n = 0$, $\langle S, |\psi\rangle\rangle = \langle S', |\psi'\rangle\rangle$ holds.

Contents

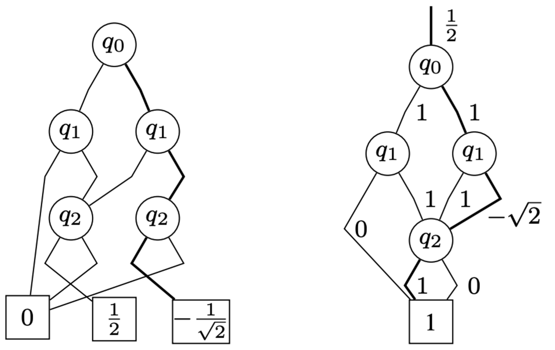
Decision Diagrams

- Classical decision diagrams (DDs) are powerful data structures that efficiently represent and manipulate Boolean functions, enabling the analysis of large state spaces in verification and optimization tasks.
- Recently, several quantum decision diagrams have been introduced to compactly represent quantum states and operations based on which efficient manipulation algorithms are developed.
  - This work used the DD package developed in MQT Core[4] for quantum computing.
  - Moreover, we developed additional features for the DD package, such as on-demand measurements and projections, making it suitable for verification.

[4] MQT Core - The Backbone of the Munich Quantum Toolkit (MQT)

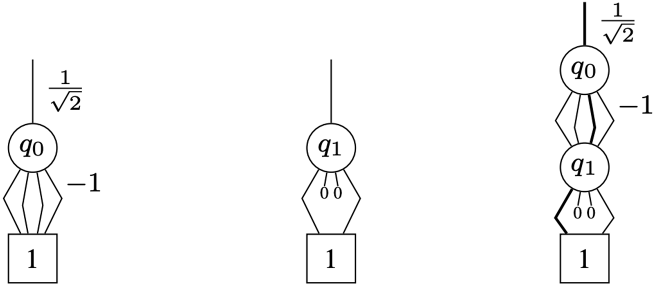
DD Package (MQT Core)

$$|\psi\rangle = \left[0, 0, \frac{1}{2}, 0, \frac{1}{2}, 0, -\frac{1}{\sqrt{2}}, 0\right]^T$$



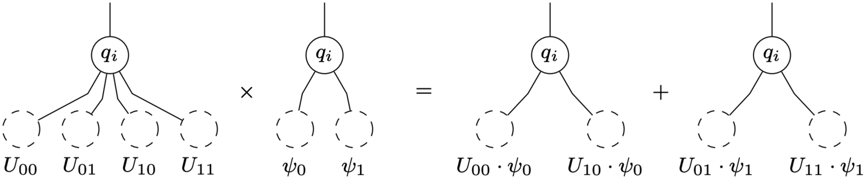
(a) Without edge weights (b) With edge weights

Representation of the state vector

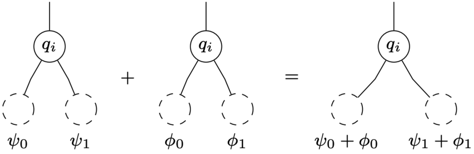


(a) H (b) I_2 (c) $U = H \otimes I_2$

Representation of matrices

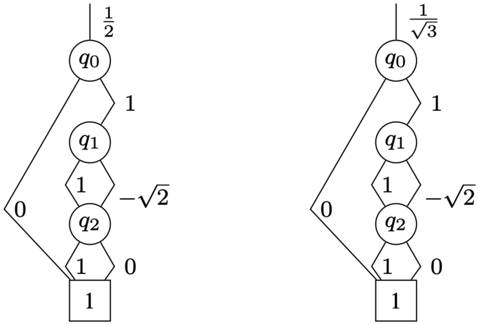


Multiplication of a unitary matrix and a state-vector



Addition of state-vectors

DD Package (MQT Core)

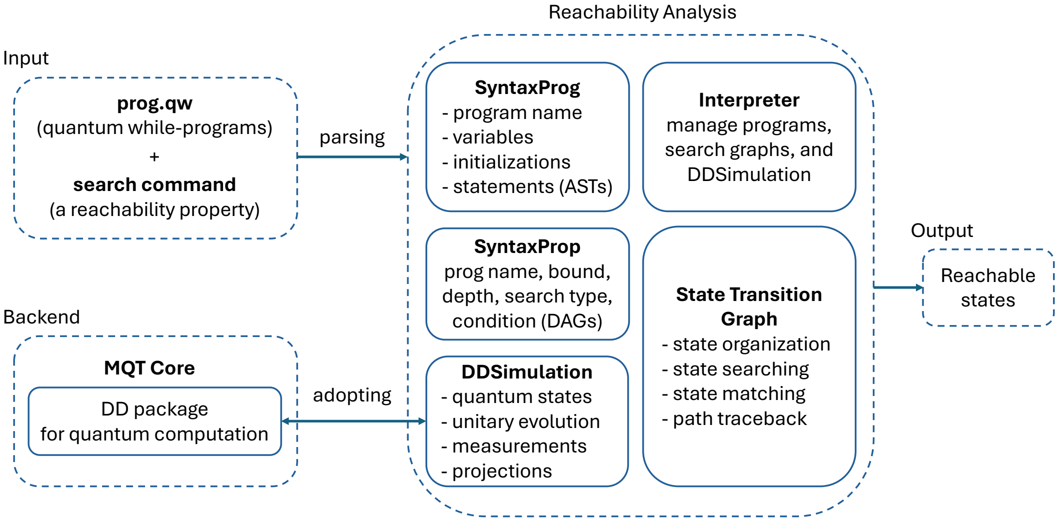


(a) Measure $q_0 = |1\rangle$ (b) Normalize amplitudes

Measurement of qubit q_0

Contents

Architecture of QRAT



Usability of QRAT

- `load <progFile.qw> .`
  Loads the quantum program specified in the `progFile.qw` file. Programs and properties can also be defined together in the same file when necessary.
- `search [bound,depth] in <progName> with <searchType> such that <condition> .`
  Performs a breadth-first search for the program specified by `progName`, aiming to find reachable states that satisfy the given `condition`. The `searchType` parameter determines the type of state transitions to consider:
  - =>1 one state transition step,
  - =>+ one or more state transition steps,
  - =>* zero or more state transition steps,
  - =>! only final states with no further transitions are considered as solutions.

  The optional `bound` argument specifies the maximum number of solutions to find (default is infinity). The optional `depth` argument sets the maximum search depth (default is infinity).
- `show path <stateId> .`
  Displays the path leading to a specific state in the search graph, identified by the number `stateId`.
- `set random seed <number> .`
- `quit .`

Contents

State Orgranization

- We uniquely represent the program counter as a pointer to the next instruction to be executed in the program, and the quantum state as a pointer managed by the ASTs and the DD package, respectively.
- These two pointers are used for hashing and uniquely identifying each state in the search graph.
- A state is thus defined as a tuple of the following components.
    - `stateId` is a unique identifier for the state, assigned incrementally starting from 0 for the initial state.
    - `parentId` is the ID of the parent state and is set to -1 for the initial state.
    - `pc` is the pointer to the program counter.
    - `qs` is the pointer to the quantum state.
    - `nextIds` represents the set of IDs corresponding to successor states.
    - `prob` is the probability associated with transitioning to this state.

Algorithm

Algorithm 1: Reachability analysis algorithm for quantum programs

```
input  : prog – the quantum program,
         property – the reachability property.
output: a finite set of reachable states that satisfy the property.

seenStates.push(buildInitialState(prog))
results ← empty
savedStateId ← 0
while savedStateId < seenStates.size() do
    currState ← seenStates[savedStateId]
    if isEndProg(currState.pc) then
        savedStateId ← savedStateId + 1
        continue
    if isSkipStm(currState.pc) then
        procSkipStm(currState)
    else if isUnitaryStm(currState.pc) then
        procUnitaryStm(currState)
    else if isCondStm(currState.pc) then
        procCondStm(currState)
    else if isWhileStm(currState.pc) then
        procWhileStm(currState)
    savedStateId ← savedStateId + 1
return results

function procCondStm(currState) :
    (qs0, pZero, qs1, pOne) ←
      measureWithProb(currState.qs, extractTargetQubit(currState.pc))
    procBranch(currState, qs0, pZero, currState.pc.getElsePC())
    procBranch(currState, qs1, pOne, currState.pc.getThenPC())
function procBranch(currState, qs, prob, pc) :
    (newState, inCached) ← makeState(currState, qs, prob, pc)
    if not inCached then
        seenStates.push(newState)
        if checkState(newState, property) then
            results ← results ∪ {newState}
```

Property Evaluation

- We define the property $\mathtt{P}(q, |\phi\rangle)$ as atomic propositions in Birkhoff–von Neumann quantum logic to indicate whether $|\psi\rangle_q$ belongs to the subspace spanned by $|\phi\rangle$.
- To evaluate $\mathtt{P}(q, |\phi\rangle)$, we construct the projector $\mathcal{P} = |\phi\rangle\langle\phi|$ and apply it to the quantum state $|\psi\rangle_q$, resulting in

$$\mathcal{P}\,|\psi\rangle_q = |\phi\rangle\langle\phi|\,|\psi\rangle_q = |\psi'\rangle_q\,.$$

  If $|\psi'\rangle_q$ is equivalent to $|\psi\rangle_q$ up to a global phase, the property is satisfied in $|\psi\rangle$; otherwise, it is not.
- The equivalence can be efficiently checked by comparing their pointers, as the DD package uniquely represents quantum states as pointers.

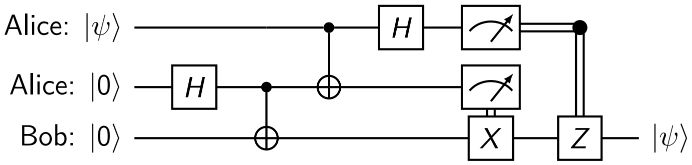
Contents

Case Studies

- We use QRAT to confirm the correctness of the following programs to demonstrate its effectiveness and practicality.
  - **Quantum Teleportation** allows the teleportation of an arbitrary single-qubit state from a sender to a receiver using only three qubits and two classical bits.
  - **Grover Search** provides a quadratic speedup for solving an unstructured search. The algorithm requires $O(\sqrt{N})$ queries to locate the correct solution in a database of size N, compared to $O(N)$ in classical search methods.
- QRAT is implemented in C++ available at `https://github.com/canhminhdo/qrat`.

Quantum Teleportation

- Quantum Teleportation is a quantum communication protocol for teleporting an arbitrary pure state by sending two bits of classical information.
- We want to verify whether Alice correctly teleports an arbitrary unknown quantum state to Bob at the end.



Rechability Analysis of Quantum Teleportation

- We first prepare the Quantum Teleportation program in a file named `teleport.qw`.

```
prog TELEPORT is
var q0, q1, q2 : qubit;
init
    q0 := random; // a random quantum state being teleported
    q1 := |0>;
    q2 := |0>;
begin
    q1 := H[q1];
    q1, q2 := CX[q1, q2];
    q0, q1 := CX[q0, q1];
    q0 := H[q0];
    if M[q1] = 1 then q2 := X[q2]; else skip; fi;
    if M[q0] = 1 then q2 := Z[q2]; else skip; fi;
end
```

Rechability Analysis of Quantum Teleportation

- Run QRAT from the command line.

```
$ qrat
```

- Load the Quantum Teleportation program into QRAT.

```
$ load teleport.qw .
```

- Conduct a reachability analysis for Quantum Teleportation in QRAT.

```
$ search in TELEPORT with =>! such that P(q2, init[q0]) .
```

- Show a path that leads to the target state found from an initial state.

```
$ show path 13 .
```

Experimental results

- We used a MacPro computer with a 2.5 GHz processor, 28 cores, and 1 TB of RAM to conduct experiments.

Program	#Qubits	#Unitary	#Meas.	#States	Time	#Exps.
Teleportation	3	8	2	17	≈ 0	100
Grover	10	1,430	10	5,503	11ms	100
	15	11,973	15	143,012	424ms	100
	20	90,108	20	4,284,369	4.7m	100

Contents

Conclusions and Future Work

- We have introduced QRAT, the first reachability analysis tool for quantum programs, leveraging a state-of-the-art decision diagram developed in MQT Core for quantum computing.
- In future work, in addition to conducting more case studies, we plan to support atomic transitions for sequences of unitary operations in order to reduce the size of the state transition graph and the unique table of quantum states, thereby accelerating the verification process.

Our Recent Work on Quantum System Verification

- C.M. Do, K. Ogata: An Executable Operational Semantics of Quantum Programs and Its Application. *International Symposium on Software Fault Prevention, Verification, and Validation (SFPVV)*, 2024.
- C.M. Do, T. Takagi, K. Ogata: Automated Quantum Protocol Verification Based on Concurrent Dynamic Quantum Logic. *ACM Transactions on Software Engineering and Methodology (ACM TOSEM)*, 2024.
- C.M. Do, K. Ogata: Equivalence Checking of Quantum Circuits Based on Dirac Notation in Maude. *The 15th International Workshop on Rewriting Logic and its Applications (WRLA)*, 2024.
- C.M. Do, K. Ogata: Symbolic Model Checking Quantum Circuits in Maude. *PeerJ Computer Science*, 2024.
- T. Takagi, C.M. Do, K. Ogata: Automated Quantum Program Verification in Dynamic Quantum Logic. *The 5th International Workshop on Dynamic Logic – New Trends and Applications (DaLí)*, 2023.

Thank You!