Contents lists available at ScienceDirect

Information Processing Letters

journal homepage: www.elsevier.com/locate/ipl

Branching bisimulation semantics for quantum processes

Hao Wu [a], Qizhe Yang [b,*], Huan Long [a]

[a] BASICS, Shanghai Jiao Tong University, China
[b] Shanghai Normal University, China

ARTICLE INFO

Keywords:
Formal semantics
Quantum process algebra
Branching bisimulation
Verification

ABSTRACT

The qCCS model proposed by Feng et al. provides a powerful framework to describe and reason about quantum communication systems that could be entangled with the environment. However, they only studied weak bisimulation semantics. In this paper we propose a new branching bisimilarity for qCCS and show that it is a congruence. The new bisimilarity is based on the concept of ϵ-tree and preserves the branching structure of concurrent processes where both quantum and classical components are allowed. Furthermore, we present a polynomial time equivalence checking algorithm for the ground version of our branching bisimilarity.

1. Introduction

Quantum computers have been proven capable of achieving exponential speed-up for certain classical computational problems, which is known as quantum supremacy [9]. With rapid development of quantum hardware, such as IBM's unveiling the 433-qubit Osprey processor in 2022 and more recently Google's 70-qubit Sycamore processor, the research of quantum information has garnered increasing interest. However, the laws of quantum mechanics, such as the no-cloning of quantum information [26], present profound challenges to almost all fields of computer science, including quantum programming language, quantum model checking etc. One well-known example is the verification of the correctness and safety of quantum communication protocols. The classical theories, methods, and technologies are not directly applicable to quantum systems.

Fortunately, over the years researchers have proposed many powerful frameworks for describing and verifying properties of classical concurrent communication systems. Process algebra has been especially successful in building rigorous mathematical languages and techniques for formally modeling classical concurrent systems. As a continuing of such success, various quantum process algebras have been proposed for formal analysis of quantum systems and verification of quantum communication protocols. One of the most fundamental theoretical research problems in quantum process algebra is to find a suitable quantum generalization of the notion of bisimulation in classical process algebra. In practical terms, an equally significant question arises concerning the efficient verification of bisimulation in quantum systems. Generally what we would desire is a full-fledged quantum process algebra equipped with proper behavioral equivalence on quantum processes that can be verified efficiently. In this paper we propose *quantum branching bisimulation* for the quantum CCS model, which abstracts from internal activity while at the same time preserves the branching structure of quantum processes in a strong sense. Congruence property and equivalence checking algorithm for quantum branching bisimulation are also given.

1.1. Related work

In this part we discuss some related work that we view as most relevant to ours, especially from both bisimulation relation and verification algorithm aspects. We then give a brief introduction on our new quantum branching bisimulation.

Jorrand and Lalire [16] defined a language QPAlg (Quantum Process Algebra) by extending a CCS-like process algebra with several quantum primitives in 2004. Later a branching bisimilarity that preserves the branching structure of process graphs was defined in [17]. However, this bisimilarity is *not* a congruence as it is not preserved by the parallel composition operator. Additionally, their setting does not consider the state change of quantum systems caused by quantum operations. Here quantum operations formalize the possible transformations that a quantum system may undergo, including *unitary transformations* and *quantum measurements* [21]. Gay and Nagarajan [12] defined a language CQP (Communicating Quantum Processes), which is obtained from the π-calculus [19,20] by adding primitives for unitary transformations and quantum measurements. They established a type system to ensure the legality of ownership of qubits. A full probabilistic branching bisimulation for CQP was proposed by Davidson [3] and shown

* Corresponding author.
E-mail addresses: wuhao_seiee@sjtu.edu.cn (H. Wu), qzyang@shnu.edu.cn (Q. Yang), longhuan@sjtu.edu.cn (H. Long).

https://doi.org/10.1016/j.ipl.2024.106492
Received 28 September 2023; Received in revised form 8 March 2024; Accepted 8 March 2024

Available online 16 March 2024

to be a congruence. However, neither QPAlg nor CQP can effectively describe the entangled communication in quantum systems, where entangled communication refers to the phenomenon that two participant quantum systems in a communication process may share a pair of qubits in the entangled state.

To overcome this problem, Feng et al. [7] then proposed a language named *quantum CCS* (i.e., qCCS) for quantum concurrent systems. The language qCCS is the quantum extension of classical value-passing CCS [14,15]. By carefully designing the semantics of quantum-input (quantum-output resp.), qCCS provides a general framework to describe quantum communication systems which could be entangled with the environment. They also studied bisimulation for finite quantum processes and proved a limited congruence property for the bisimilarity, in which some constraints are put on the participating processes when the parallel composition operator is involved. Later in [28] the same authors studied a purely quantum version of qCCS in which no classical data is explicitly involved. They established the strong bisimulation semantics for such a model and showed that the induced bisimilarity is a congruence with respect to the parallel composition operator. In [8], the authors introduce a new notion of weak probabilistic bisimilarity for the general qCCS model where both classical and quantum data are involved. They also prove that it is preserved by all process operators, including parallel composition and recursive definitions. The bisimilarity in [8] distinguishes quantum-input from other actions, and to match a quantum-input, the application of super-operator on evolving processes should be considered. Such requirement is somewhat complicated and unnatural. Then Deng and Feng in [5] proposed an open bisimulation for quantum processes, in which the bisimulation condition and the closure under super-operator application are made to be two separated requirements. They further showed that the induced open bisimilarity is a congruence relation and has a modal characterization. Later in [6], Feng et al. propose the symbolic bisimulation for qCCS and show its coincidence with the open bisimulation in [5] when strong bisimulation is considered.

Apart from the discussion about proper equivalence relations for quantum processes, efficient equivalence checking algorithm is of great importance in practice. Given two quantum processes P and S, equivalence checking decides whether P and S are related by the selected equivalence relation. For quantum models, Qin et al. have investigated the equivalence checking problem for the ground (weak) bisimilarity [22]. Inspired by [24], they reduce the problem of finding a matching weak transition to a linear programming problem that can be solved in polynomial time and give a polynomial time algorithm for the equivalence checking problem. In [29], Zhang et al. gives a polynomial time equivalence checking algorithm for the branching bisimilarity in RCCS model by exploiting the classical split-refinement method.

As we have just shown, most of the previous bisimulations proposed for qCCS model are either strong (weak resp.) bisimulation or not congruent. They are not exactly what we will need in practice. On the one hand, strong bisimulation requires that every internal action must be bisimulated for bisimilar processes. Such requirement is usually too strict to practical use. On the other hand, weak bisimulation ignores all internal actions and only requires that each external actions should be matched. It turns out that weak bisimulation does not preserve the branching structure of a process, i.e., two weak bisimilar processes may go through non-bisimilar intermediate states. Such deficiency can cause many problems in practical applications and analysis [13,10]. As an example, consider two quantum programs P_1 and P_2 which are supposed to implement some quantum protocol specification (formalized as process S). To say that P_i ($i \in \{1, 2\}$) implements S correctly, P_i should simulate every state of S precisely without introducing any unwanted states as by-product. In classical concurrent systems, one such proper relation is the famous branching bisimilarity [25]. In [1], Basten then proved that the branching bisimilarity is indeed an equivalence. When it comes to the probabilistic setting, Castiglioni et al. [2] proposed a probabilistic branching bisimilarity for divergence-free probabilistic processes and showed that it is an equivalence. Compared with strong (weak resp.) bisimulation, branching bisimulation ignores redundant simulation of deterministic steps while still preserving the branching structure of the considered processes. Thus branching bisimulation is often more suitable for being chosen as the criteria for evaluating equivalence between specifications and implementations. Such difference is crucial for compositional concurrent processes as the environment can interact with intermediate processes freely. It is then natural to study quantum branching bisimulation. In our opinion, one possible obstacle for the lack of such work in the past could be that most of the previous quantum bisimulation are based on the conception of probabilistic bisimulation proposed by Segala et al. [23], where they use the notion of probability distribution on states to characterize probabilistic transition. This strategy becomes highly involved if one tries to directly extend it to branching bisimulation. Recently, a branching bisimilarity equivalence [11] has been defined for randomized process models based on the concept of ϵ-tree and has been shown to be a congruence relation, which makes the study of quantum branching bisimulation feasible. In this paper, we extend the general approach to qCCS model in combination with the notion of open bisimulation. We propose a novel quantum branching bisimulation. Then follow the methodology in [29], we also give an efficient equivalence checking algorithm for it.

In short, in this paper we focus on solving two fundamental problems for quantum system verification. We first establish a new equivalence with good algebraic property, and then give an efficient equivalence checking algorithm for it.

1.2. *Contribution*

The main contributions of this paper are twofold.

1. We propose a new branching bisimulation for the full qCCS model, which is proved to be a congruence relation. In particular, we use the ϵ-tree technique which is model-independent.
2. We present a polynomial time equivalence checking algorithm for the ground branching bisimilarity.

1.3. *Organization*

The structure of the paper is as follows. Section 2 recalls the syntax and semantics of qCCS model. Section 3 introduces our branching bisimilarity for qCCS and shows that it is a congruence relation. Section 4 gives the polynomial equivalence checking algorithm for the ground branching bisimilarity. Section 5 includes some concluding remarks.

2. Quantum CCS

In this paper, we will mainly follow the quantum CCS model proposed by Feng et al. in [8], with one exception that we only allow guarded nondeterministic choice rather than general summation, which is necessary for the congruence result. In qCCS, we use Real to denote the set of real-valued classical data and Qbt to denote the set of quantum data (qubits). Accordingly, we assume two countable sets of variables: *cVar* for the set of classical variables, ranged over by $x, y, \cdots$, and *qVar* for the set of quantum variables, ranged over by $q, r, \cdots$. The set of real-valued expressions is denoted by *Exp*, ranged over by e. We also assume two types of channels in qCCS: *cChan* for classical channels, ranged over by $c, d, \cdots$, and *qChan* for quantum channels, ranged over by $\underline{c}, \underline{d}, \cdots$. Then the set of all channels is denoted as $Chan = cChan \cup qChan$. A relabeling function f is an injective function on $Chan$ with $f(cChan) \subseteq cChan$ and $f(qChan) \subseteq qChan$. A set of distinct quantum variables $\{q_1, \cdots, q_n\}$ is often abbreviate as $\widetilde{q}$, when the number n is unimportant or clear from the context.

Now we present the syntax of qCCS.

Tau: $$\frac{}{\langle \tau.P,\rho\rangle \xrightarrow{\tau} \langle P,\rho\rangle}$$

C-Inp: $$\frac{}{\langle c?x.P,\rho\rangle \xrightarrow{c?v} \langle P\{v/x\},\rho\rangle} \quad v \in \mathsf{Real}$$

C-Outp: $$\frac{}{\langle c!e.P,\rho\rangle \xrightarrow{c!v} \langle P,\rho\rangle} \quad v = [\![e]\!]$$

C-Com: $$\frac{\langle P_1,\rho\rangle \xrightarrow{c?v} \langle P_1',\rho\rangle \quad \langle P_2,\rho\rangle \xrightarrow{c!v} \langle P_2',\rho\rangle}{\langle P_1 \| P_2,\rho\rangle \xrightarrow{\tau} \langle P_1' \| P_2',\rho\rangle}$$

Q-Inp: $$\frac{}{\langle \underline{c}?q.P,\rho\rangle \xrightarrow{\underline{c}?r} \langle P\{r/q\},\rho\rangle} \quad r \notin qv(\underline{c}?q.P)$$

Q-Outp: $$\frac{}{\langle \underline{c}!q.P,\rho\rangle \xrightarrow{\underline{c}!q} \langle P,\rho\rangle}$$

Q-Com: $$\frac{\langle P_1,\rho\rangle \xrightarrow{\underline{c}?r} \langle P_1',\rho\rangle \quad \langle P_2,\rho\rangle \xrightarrow{\underline{c}!r} \langle P_2',\rho\rangle}{\langle P_1 \| P_2,\rho\rangle \xrightarrow{\tau} \langle P_1' \| P_2',\rho\rangle}$$

Oper: $$\frac{}{\langle U[\widetilde{q}].P,\rho\rangle \xrightarrow{\tau} \langle P, U_{\widetilde{q}}(\rho)\rangle}$$

Meas: $$\frac{}{\langle M[\widetilde{q};x].P,\rho\rangle \xrightarrow{p_i\tau} \langle P\{\lambda_i/x\}, E^i_{\widetilde{q}}\rho E^i_{\widetilde{q}}/p_i\rangle}$$
where M has the spectrum decomposition $M = \sum_{i\in I}\lambda_i E^i$ and $p_i = \mathrm{tr}(E^i_{\widetilde{q}})/\mathrm{tr}(\rho)$.

Int: $$\frac{\langle P_1,\rho\rangle \xrightarrow{\ell} \langle P_1',\rho'\rangle}{\langle P_1 \| P_2,\rho\rangle \xrightarrow{\ell} \langle P_1' \| P_2,\rho'\rangle} \quad bv(\ell) \cap qv(P_2) = \emptyset$$

Sum: $$\frac{\langle \lambda_i.P_i,\rho\rangle \xrightarrow{\ell} \langle P_i',\rho'\rangle}{\langle \sum_{i\in I}\lambda_i.P_i,\rho\rangle \xrightarrow{\ell} \langle P_1',\rho'\rangle}$$

Rel: $$\frac{\langle P,\rho\rangle \xrightarrow{\ell} \langle P',\rho'\rangle}{\langle P[f],\rho\rangle \xrightarrow{f(\ell)} \langle P'[f],\rho'\rangle}$$

Res: $$\frac{\langle P,\rho\rangle \xrightarrow{\ell} \langle P',\rho'\rangle}{\langle P\backslash L,\rho\rangle \xrightarrow{\ell} \langle P'\backslash L,\rho'\rangle} \quad cn(\ell) \cap L = \emptyset$$

Cho: $$\frac{\langle P,\rho\rangle \xrightarrow{\ell} \langle P',\rho'\rangle \quad [\![b]\!]=\mathrm{true}}{\langle \mathbf{if}\ b\ \mathbf{then}\ P,\rho\rangle \xrightarrow{\ell} \langle P',\rho'\rangle}$$

Def: $$\frac{\langle P\{\widetilde{r}/\widetilde{q}\},\rho\rangle \xrightarrow{\ell} \langle P',\rho'\rangle \quad A(\widetilde{q}) \overset{\mathrm{def}}{=} P}{\langle A(\widetilde{r}),\rho\rangle \xrightarrow{\ell} \langle P',\rho'\rangle}$$

Fig. 1. Operational semantics of qCCS.

Definition 2.1 *(Quantum process [8]).* The set of qCCS processes $qProc$ and the free quantum variable function $qv : qProc \to 2^{qVar}$ are defined inductively by the following rules:

(1) $\mathbf{nil} \in qProc$, and $qv(\mathbf{nil}) = \emptyset$;
(2) $\mathsf{A}(\widetilde{q}) \in qProc$, and $qv(\mathsf{A}(\widetilde{q})) = \widetilde{q}$;
(3) $\tau.P \in qProc$, and $qv(\tau.P) = qv(P)$;
(4) $c?x.P \in qProc$, and $qv(c?x.P) = qv(P)$;
(5) $c!e.P \in qProc$, and $qv(c!e.P) = qv(P)$;
(6) $\underline{c}?q.P \in qProc$, and $qv(\underline{c}?q.P) = qv(P) - \{q\}$;
(7) If $q \notin qv(P)$ then $\underline{c}!q.P \in qProc$, and $qv(\underline{c}!q.P) = qv(P) \cup \{q\}$;
(8) $U[\widetilde{q}].P \in qProc$, and $qv(U[\widetilde{q}].P) = qv(P) \cup \widetilde{q}$;
(9) $M[\widetilde{q};x].P \in qProc$, and $qv(M[\widetilde{q};x].P) = qv(P) \cup \widetilde{q}$;
(10) $\sum_{i\in I}\lambda_i.P_i \in qProc$, and $qv(\tau.P) = \bigcup_{i\in I} qv(\lambda_i.P_i)$;
(11) If $qv(P) \cap qv(Q) = \emptyset$ then $P \parallel Q \in qProc$, and $qv(P \parallel Q) = qv(P) \cup qv(Q)$;
(12) $P[f] \in qProc$, and $qv(P[f]) = qv(P)$;
(13) $P\backslash L \in qProc$, and $qv(P\backslash L) = qv(P)$;
(14) $\mathbf{if}\ b\ \mathbf{then}\ P \in qProc$, and $qv(\mathbf{if}\ b\ \mathbf{then}\ P) = qv(P)$,

where $P, Q \in qProc$, $\lambda_i \in \{\tau, c?x, c!e, \underline{c}?q, \underline{c}!q, U[\widetilde{q}], M[\widetilde{q};x]\}$, f is a relabeling function, $L \subseteq Chan$, and b is a boolean expression.

Most constructors are standard as in the classical CCS [18]. We only briefly explain some new constructors in quantum scenario: $\underline{c}?q$ ($\underline{c}!q$ resp.) stands for the action of quantum-input (quantum-output resp.) a qubit via quantum channel $\underline{c}$; $U[\widetilde{q}]$ denotes the action of applying a *trace-preserving super-operator* [8] U on the qubits $\widetilde{q}$; $M[\widetilde{q};x]$ denote the action of performing the measurement on qubits $\widetilde{q}$ according to the observable M, where the classical variable x is used to store the measurement result; $\mathsf{A}(\widetilde{q})$ is a process constant defined by the equation $\mathsf{A}(\widetilde{q}) \overset{\mathrm{def}}{=} P$, where $P \in qProc$ with $qv(P) \subseteq \widetilde{q}$. The notion of *free* and *bound* classical variables has their usual meanings as in classical CCS. Only note that the variable x appears in quantum measurement operator $M[\widetilde{q};x]$ is bound. Given a qCCS process P, if it contains no free classical variable (i.e., $fv(P) = \emptyset$), then we call it a *closed quantum process*.

Here we introduce some unitary operators that we will use later. The Hadamard operator H is a single-qubit operation that maps the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The four Pauli operators σ^0, σ^1, σ^2 and σ^3 form a basis for the real vector space of 2×2 Hermitian matrices. Under the computational basis, these operators can be defined as follows:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \sigma^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma^2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \sigma^3 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

Note that all unitary transformations are trace-preserving super-operators [8], including the Hadamard operator and the Pauli operators.

A few more notations are necessary for presenting the operational semantics of qCCS. Given any quantum variable $q \in qVar$, its associating 2-dimensional Hilbert space is denoted by $\mathcal{H}_q$. Given a nonempty subset $S \subseteq qVar$, we use $\mathcal{H}_S$ to denote the tensor product space $\bigotimes_{q\in S}\mathcal{H}_q$ and use $\mathcal{H}_{\overline{S}}$ to denote the space $\bigotimes_{q\notin S}\mathcal{H}_q$. The state space of a system that contains all quantum variables is then denoted as $\mathcal{H} = \bigotimes_{q\in qVar}\mathcal{H}_q$.

Given a closed quantum process P and a density operator ρ on the Hilbert space $\mathcal{H}_S$ (where S is a *finite* set with $qv(P) \subseteq S$), we call the pair $\langle P, \rho\rangle$ a *configuration*. Let Con be the set of all configurations, ranged over by $A, B, C, \cdots$. Given an equivalence $\mathcal{E}$ on Con, we use $Con/\mathcal{E}$ to denote the set of equivalence classes of Con under $\mathcal{E}$. We will write $A\ \mathcal{E}\ B$ for $(A, B) \in \mathcal{E}$, and use $[A]_{\mathcal{E}}$ to represent the equivalence class containing A. The set of nondeterministic actions Act_n takes the following form:

$$\{\tau\} \cup \{c?v, c!v \mid c \in cChan, v \in \mathsf{Real}\}$$
$$\cup \{\underline{c}?r, \underline{c}!r \mid \underline{c} \in qChan, r \in qVar\}.$$

The set of probabilistic actions is $Act_p = \{p\tau \mid 0 < p < 1\}$. Then the set of all possible actions is $Act = Act_n \cup Act_p$. We write ℓ for elements of Act.

The operational semantics of qCCS is given in Fig. 1, where $\ell \in Act$. Here the semantics of quantum measurement is given as a collection of (probabilistic) silent transitions, which helps establish the congruence result in the following sections.

Most of the rules in Fig. 1 are straightforward. We only explain the measurement rule (i.e., **Meas**). It characterizes that, after measurement the configuration will evolve into different configurations with corresponding probabilities depending on different measurement outcomes. Notice that similar to [7], the rule for unitary transformation (i.e., **Oper**) and measurement on quantum systems (i.e., **Meas**) are considered as (probabilistic) silent actions performed by the quantum systems. One can refer to [7] for more explanations.

3. Branching bisimulation between quantum processes

As mentioned earlier, [8] proposes the strong and weak bisimulation for qCCS as tools to capture the idea that a quantum process approximately implements its specification. However as in real specification, such approximation is essentially branching bisimulation as there will be no silent actions in specification. As a matter of fact, in concurrency society, branching bisimulation is generally regarded as the finest practical equivalence [25] of the whole linear time-branching time spectrum. It is then natural to build the branching bisimulation for qCCS and study its algebraic properties. In this section, we will propose a branching bisimilarity for qCCS model and then prove its congruence property.

3.1. *Quantum branching bisimulation*

In Fig. 1 we have introduced probabilistic silent transitions for quantum measurement as the rule of **Meas**. Thus to build branching bisimulation for qCCS, we can take the techniques in [11] and then facilitate the relating argument. We start with the definition of ϵ-tree in [11], which is a convenient technical gadget for our work.

Definition 3.1 *(ϵ-tree [11]).* Let $\mathcal{E}$ be an equivalence on Con and $A \in Con$ be a configuration. An ϵ-tree $t^A_{\mathcal{E}}$ of A with regard to $\mathcal{E}$ is a labeled tree such that the following statements hold.

- Each node of $t^A_{\mathcal{E}}$ is labeled by an element of Con and each edge is labeled by an element of $(0,1]$. The root of $t^A_{\mathcal{E}}$ is labeled by A.
- All the labels of the nodes of $t^A_{\mathcal{E}}$ are in $[A]_{\mathcal{E}}$.
- If a node labeled B has only one child B', then the edge from B to B' is labeled 1 and $B \xrightarrow{\tau} B'$.
- If a node labeled B has k children $B_1, \cdots, B_k$ and each edge from B to B_i is labeled p_i, then $\{p_i\}_{i\in[k]}$ is a probability distribution and $B \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]} B_i$.

Here $B \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]} B_i$ is the collective silent transition defined in [11], which means that B can perform a silent transition to $\{B_i\}_{i\in[k]}$ with corresponding probability $\{p_i\}_{i\in[k]}$. Intuitively, ϵ-tree is a generalization of the classical state-preserving internal action sequence. For a better understanding of ϵ-tree, consider the following example.

Example 3.2. To the process:

$$\mathsf{A}(q) \overset{\text{def}}{=} M[q;x].\mathbf{if}\ x=0\ \mathbf{then}\ H[q].c!0.\mathbf{nil}$$
$$\mathbf{else}\ H[q].\sigma^2[q].A(q).$$

Consider any density operator $\rho \in \mathcal{D}(\mathcal{H}_{\overline{\{q\}}})$. Let $\rho_0 = [|0\rangle]_q \otimes \rho$, $\rho_1 = [|1\rangle]_q \otimes \rho$, $\rho_2 = \left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]_q \otimes \rho$, and $\rho_3 = \left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]_q \otimes \rho$. Here we use the symbol $[\Psi]_q$ to emphasize that the given state Ψ is actually associated with the quantum variable q. Then consider any equivalence $\mathcal{E}$ satisfying that $[\langle \mathsf{A}(q),\rho_3\rangle]_{\mathcal{E}} = [\langle H[q].c!0.\mathbf{nil},\rho_0\rangle]_{\mathcal{E}} = [\langle c!0.\mathbf{nil},\rho_3\rangle]_{\mathcal{E}} = [\langle H[q].\sigma^2[q].\mathsf{A}(q),\rho_1\rangle]_{\mathcal{E}} = [\langle \sigma^2[q].\mathsf{A}(q),\rho_2\rangle]_{\mathcal{E}}$. The infinite ϵ-tree of $\langle \mathsf{A}(q),\rho_3\rangle$ is given in Fig. 2.

A *branch* π in an ϵ-tree is either a finite path from the root to a leaf or an infinite path from the root. For a finite branch π, we write

$$\begin{tikzcd}
& \langle \mathsf{A}(q),\rho_3\rangle \arrow[dl, "1/2"'] \arrow[dr, "1/2"] & \\
\langle H[q].c!0.\mathbf{nil},\rho_0\rangle \arrow[d, "1"] & & \langle H[q].\sigma^2[q].\mathsf{A}(q),\rho_1\rangle \arrow[d, "1"] \\
\langle c!0.\mathbf{nil},\rho_3\rangle & & \langle \sigma^2[q].\mathsf{A}(q),\rho_2\rangle \arrow[d, "1"] \\
& & \langle \mathsf{A}(q),\rho_3\rangle \\
& \cdot\ \cdot\ \cdot & \cdot\ \cdot\ \cdot
\end{tikzcd}$$

Fig. 2. The ϵ-tree for $\langle \mathsf{A}(q),\rho_3\rangle$.

$|\pi|$ for its length and use $\pi(i)$ to denote the label of its i-th edge. The probability $\mathsf{P}(\pi)$ of a finite branch π is then defined as $\prod_{i\le|\pi|}\pi(i)$. The *convergence probability* $\mathsf{P}^c(t^A_{\mathcal{E}})$ is defined as follows, which is intuitively the probability of the portion of finite branches in $t^A_{\mathcal{E}}$.

$$\mathsf{P}^c(t^A_{\mathcal{E}}) \overset{\text{def}}{=} \lim_{k\to\infty}\Big(\sum\{\mathsf{P}(\pi) \mid \pi \text{ is a finite branch in } t^A_{\mathcal{E}}$$
$$\text{such that } |\pi| \le k\}\Big).$$

An ϵ-tree $t^A_{\mathcal{E}}$ is *regular* if $\mathsf{P}^c(t^A_{\mathcal{E}}) = 1$.

Based on the ϵ-tree, two types of transitions are introduced in Definition 3.3 (Definition 3.4 resp.), which are used to characterize state-changing non-probabilistic actions (probabilistic actions resp.) that should be bisimulated explicitly.

Definition 3.3 *(ℓ-transition [11]).* Suppose $\mathcal{B} \in Con/\mathcal{E}$ and $\neg(\ell = \tau \wedge \mathcal{B} = [A]_{\mathcal{E}})$. We say there is an ℓ-transition from A to $\mathcal{B}$ with regard to $\mathcal{E}$, written $A \rightsquigarrow_{\mathcal{E}}\xrightarrow{\ell} \mathcal{B}$, if there exists a regular ϵ-tree $t^A_{\mathcal{E}}$ such that, $L \xrightarrow{\ell} L' \in \mathcal{B}$ for every leaf L of $t^A_{\mathcal{E}}$.

Intuitively, ℓ-transition is a generalization of the transition $\Rightarrow_{\mathcal{E}}\xrightarrow{\ell}$ in the classical CCS model, where the state-preserving internal action sequence $\Rightarrow_{\mathcal{E}}$ is replaced by a regular ϵ-tree, and the action $\xrightarrow{\ell}$ is either an external action or a state-changing internal action.

We still need to formalize the simulation of probabilistic actions. Suppose $L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]} L_i$ and some L_i falls into an equivalence class $\mathcal{B} \ne [L]_{\mathcal{E}}$. Define

$$\mathsf{P}\left(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}\right) \overset{\text{def}}{=} \sum_{i\in[k]} \left\{p_i \mid L \xrightarrow{p_i\tau} L_i \in \mathcal{B}\right\}.$$

The normalized probability is defined as the conditional probability of leaving $[L]_{\mathcal{E}}$ to $\mathcal{B}$, i.e.,

$$\mathsf{P}_{\mathcal{E}}\left(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}\right)$$
$$\overset{\text{def}}{=} \mathsf{P}\left(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}\right) \Big/ \left(1 - \mathsf{P}\left(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} [L]_{\mathcal{E}}\right)\right).$$

Intuitively, normalized probability characterizes the conditional probability of transferring into a new equivalent class with respect to equivalence $\mathcal{E}$ via one step of probabilistic silent actions (i.e., the collective silent transition). Next we promote this conception to ϵ-tree by considering every leaf and then get the so-called p-transition.

Definition 3.4 *(p-transition [11]).* Suppose $\mathcal{B} \in Con/\mathcal{E}$ and $\mathcal{B} \ne [A]_{\mathcal{E}}$. We say there is a p-transition from A to $\mathcal{B}$ with regard to $\mathcal{E}$, written

$A \rightsquigarrow_{\mathcal{E}} \xrightarrow{p} \mathcal{B}$, if there exists a regular ϵ-tree $t^A_{\mathcal{E}}$ satisfying that $L \xrightarrow{\coprod_{i\in[k]} p_i \tau} \coprod_{i\in[k]} L_i$ and the normalized probability $\mathsf{P}_{\mathcal{E}}\left(L \xrightarrow{\coprod_{i\in[k]} p_i \tau} \mathcal{B}\right) = p$ for every leaf L of $t^A_{\mathcal{E}}$.

Intuitively, p-transition is the counterpart of ℓ-transition for probabilistic actions, which requires that after going through a regular ϵ-tree, every evolved configuration can arrive at a new equivalence class with the same normalized probability p.

Suppose $\langle P, \rho\rangle$ is a configuration, the partial trace of ρ with respect to $qv(P)$ can be defined as $\mathrm{tr}_{qv(P)}(\rho)$, which is exactly the reduced quantum state of ρ on the environment of system P. Now we first define the ground branching bisimulation and bisimilarity for qCCS.

Definition 3.5 *(Ground branching bisimulation).* An equivalence relation $\mathcal{R} \subseteq Con \times Con$ is called a ground branching bisimulation if for any $\langle P, \rho\rangle, \langle Q, \sigma\rangle \in Con$, $\langle P, \rho\rangle\ \mathcal{R}\ \langle Q, \sigma\rangle$ implies that

1. $qv(P) = qv(Q)$, $\mathrm{tr}_{qv(P)}(\rho) = \mathrm{tr}_{qv(Q)}(\sigma)$;
2. If $\langle P, \rho\rangle \rightsquigarrow_{\mathcal{R}} \xrightarrow{\ell} C \in Con/\mathcal{R}$ such that $\neg(\ell = \tau \wedge C = [\langle P, \rho\rangle]_{\mathcal{R}})$, then $\langle Q, \sigma\rangle \rightsquigarrow_{\mathcal{R}} \xrightarrow{\ell} C$;
3. If $\langle P, \rho\rangle \rightsquigarrow_{\mathcal{R}} \xrightarrow{p} C \in Con/\mathcal{R}$ such that $C \neq [\langle P, \rho\rangle]_{\mathcal{R}}$, then $\langle Q, \sigma\rangle \rightsquigarrow_{\mathcal{R}} \xrightarrow{p} C$.

The largest ground branching bisimulation, which is guaranteed to exist using standard arguments [8], is called *ground branching bisimilarity* and is denoted by $\simeq_{gb}$.

In the above definition, the term *ground* is used to emphasize that we do not consider super-operator application when matching a quantum input action. In contrast, the *weak bisimilarity* proposed in [8] separates quantum input from other actions and considers the effects of super-operators in quantum input clause.

Following [4], we are then ready to introduce the notion of *closeness under super-operator application.*

Definition 3.6 *([4]).* An relation $\mathcal{R} \subseteq Con \times Con$ is closed under super-operator application if $\langle P, \rho\rangle\ \mathcal{R}\ \langle Q, \sigma\rangle$ implies that $\langle P, U(\rho)\rangle\ \mathcal{R}\ \langle Q, U(\sigma)\rangle$ for any super-operator U acting on $\mathcal{H}_{\overline{qv(P)}}$, where $\mathcal{H}_{\overline{qv(P)}}$ stands for the tensor product space $\bigotimes_{q \notin qv(P)} \mathcal{H}_q$.

Now we can formalize branching bisimulation in the quantum scenario.

Definition 3.7 *(Branching bisimulation).* An equivalence relation $\mathcal{R} \subseteq Con \times Con$ is called a branching bisimulation if it satisfies that

1. $\mathcal{R}$ is a ground branching bisimulation;
2. $\mathcal{R}$ is closed under all super-operator application.

Then two quantum configurations $\langle P, \rho\rangle$ and $\langle Q, \sigma\rangle$ are called branching bisimilar, denoted by $\langle P, \rho\rangle \simeq \langle Q, \sigma\rangle$, if there exists a branching bisimulation $\mathcal{R}$ satisfying that $\langle P, \rho\rangle\ \mathcal{R}\ \langle Q, \sigma\rangle$.

We can also lift the branching bisimulation relation from configurations to processes. Note in Definition 3.8 we reuse the $\simeq$ notation as its meaning should be clear from the context.

Definition 3.8. Two quantum processes P and Q are branching bisimilar, denoted as $P \simeq Q$, if for any quantum state ρ and any indexed set $\widetilde{v}$ of classical values, $\langle P\{\widetilde{v}/\widetilde{x}\}, \rho\rangle \simeq \langle Q\{\widetilde{v}/\widetilde{x}\}, \rho\rangle$. Here $\widetilde{x}$ is the set of free classical variables contained in P and Q.

We give two examples for further explanations about the new equivalence relations. Example 3.9 briefly compare our new branching bisimulation and the ones in the literature. Example 3.10 shows how to model classical quantum algorithm by $\simeq$.

Example 3.9. Let $U = \sigma^1 H$. Suppose $P = H[q].\mathbf{nil} + U[q].\mathbf{nil} + M_{0,1}[q; x].\mathbf{nil}$ and $Q = H[q].\mathbf{nil} + U[q].\mathbf{nil}$. Let $\rho = \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right]_q \otimes \rho'$, where $\rho' \in \mathcal{D}(\mathcal{H}_{\overline{\{q\}}})$.

- According to [7], $\langle P, \rho\rangle$ and $\langle Q, \rho\rangle$ are weak bisimilar. Particularly, $\langle Q, \rho\rangle$ can simulate the action $M_{0,1}[q; x]$ of $\langle P, \rho\rangle$ by choosing its actions $H[q]$ and $U[q]$ with respective probabilities one half.
- According to the probabilistic branching bisimulation given in [17], configurations $\langle \mathbf{nil}, [0]_q \otimes \rho'\rangle$ and $\langle \mathbf{nil}, [1]_q \otimes \rho'\rangle$ are regarded as equal, which then implies that $\langle P, \rho\rangle$ and $\langle Q, \rho\rangle$ are bisimilar as well. Yet their equivalence holds by a completely different reason: state changes of contexts caused by quantum operations are ignored there.
- In our framework, $\langle P, \rho\rangle$ and $\langle Q, \rho\rangle$ are not branching bisimilar to each other. Particularly, the p-transition $\langle P, \rho\rangle \rightsquigarrow_{\simeq} \xrightarrow{1/2} [\langle \mathbf{nil}, [0]_q \otimes \rho'\rangle]_{\simeq}$ cannot be simulated by $\langle Q, \rho\rangle$. It should be noted that, among these three equivalences, our branching bisimilarity is the only one that can both preserve the branching structures of the evolving configurations and keep track of changes of quantum states.

Example 3.10. Given a black box function (an oracle) O which delivers the result of the transformation $O\,|x\rangle\,|q\rangle = |x\rangle\,|q \oplus f(x)\rangle$, where $f(x) = 0$ for all $0 \le x < 2^n$ except a unique x_0, for which $f(x_0) = 1$. Let $|\varphi\rangle$ be the equal superposition state $\frac{1}{2^{n-1}} \sum_{x=0}^{2^n-1} |x\rangle$. *Grover's algorithm* [21] can output x_0 with probability $\mathcal{O}(1)$ in only $\mathcal{O}(\sqrt{2^n})$ steps. Therefore by repeating Grover's algorithm a constant number of times we can find x_0 with probability 1. This procedure can be described by the following qCCS-process.

$$GS = H^{\otimes n}[q_1, \cdots, q_n].\sigma^1[q].H[q].\{GI[q_1, \cdots, q_n, q]\}^R.$$
$$M[q_1, \cdots, q_n; x].Set_0[q_1, \cdots, q_n, q].$$
$$\textbf{if } f(x) = 1 \textbf{ then } d!x.\mathbf{nil} \textbf{ else } GS$$

where $GI = (2\,|\varphi\rangle\langle\varphi| - I)O$ is the Grover iteration operator and $R \approx \lceil \pi\sqrt{2^n}/4 \rceil$ is the Grover iteration repeating times. Let $\rho = [|0\rangle^{\otimes n}]_{q_1, \cdots, q_n} \otimes [|0\rangle]_q \otimes \rho'$, where $\rho' \in \mathcal{D}(\mathcal{H}_{\overline{\{q_1, \cdots, q_n, q\}}})$. Then there exists an ϵ-tree for configuration $\langle GS, \rho\rangle$ such that $\langle GS, \rho\rangle \rightsquigarrow_{\simeq} \xrightarrow{d!x_0} [\langle \mathbf{nil}, \rho\rangle]_{\simeq}$.

3.2. Congruence property

Congruence is one of the most desired properties for a relation, especially for compositional systems. In this section, we will show that the branching bisimulation proposed in Section 3.1 is indeed a congruence. The congruence proof follows a similar strategy as in [11] and [27]. However, branching bisimilarity in the quantum setting has additional requirements, including equal partial trace and closeness under super-operator application, thereby making establishing such a congruence result much more complicated. We start by showing that $\simeq$ is an equivalence relation.

Theorem 3.11. *$\simeq$ is an equivalence and it is the largest branching bisimulation on Con.*

Proof. Suppose $\{\mathcal{R}_i\}_{i\in I}$ is a collection of branching bisimulation on Con, we can show that the equivalence closure of $\bigcup_{i\in I} \mathcal{R}_i$ is also a branching bisimulation. The proof is similar to the one of Proposition 4.2 in [11]. □

We then proceed to show that the bisimilarity $\simeq$ for configurations is preserved by all static constructors.

Lemma 3.12. *If $\langle P,\rho\rangle \simeq \langle Q,\sigma\rangle$ then*

(1) $\langle P \parallel R,\rho\rangle \simeq \langle Q \parallel R,\sigma\rangle$;
(2) $\langle P[f],\rho\rangle \simeq \langle Q[f],\sigma\rangle$;
(3) $\langle P\backslash L,\rho\rangle \simeq \langle Q\backslash L,\sigma\rangle$;
(4) $\langle \textbf{if } b \textbf{ then } P,\rho\rangle \simeq \langle \textbf{if } b \textbf{ then } Q,\sigma\rangle$.

Proof. We only give the detail of item (1) as an example, as the other cases are simpler or easier.

Let $\mathcal{S} \overset{\text{def}}{=} \{(\langle P \parallel R,\rho\rangle, \langle Q \parallel R,\sigma\rangle) \mid \langle P,\rho\rangle \simeq \langle Q,\sigma\rangle\}$ and $\mathcal{R} \overset{\text{def}}{=} (\mathcal{S} \cup \approx)^*$. It will be enough to show that $\mathcal{R}$ is a branching bisimulation. Suppose $\langle P \parallel R,\rho\rangle\ \mathcal{S}\ \langle Q \parallel R,\sigma\rangle$ where $\langle P,\rho\rangle \simeq \langle Q,\sigma\rangle$. By the definition of $\simeq$ we have $qv(P) = qv(Q)$ and $\mathrm{tr}_{qv(P)}(\rho) = \mathrm{tr}_{qv(Q)}(\sigma)$. Thus $qv(P \parallel R) = qv(Q \parallel R)$ and $\mathrm{tr}_{qv(P\parallel R)}(\rho) = \mathrm{tr}_{qv(Q\parallel R)}(\sigma)$ hold.

According to Definition 3.7, we also have $\langle P,U(\rho)\rangle \simeq \langle Q,U(\sigma)\rangle$ for any super-operator U acting on $\mathcal{H}_{\overline{qv(P)}}$, which implies that $\langle P \parallel R,U(\rho)\rangle\ \mathcal{S}\ \langle Q \parallel R,U(\sigma)\rangle$ for any super-operator U acting on $\mathcal{H}_{\overline{qv(P)}}$. Hence $\mathcal{S}$ is closed under super-operator application. According to Theorem 3.11, we have $\simeq$ is closed under super-operator application, thus $\mathcal{R}$ is also closed under super-operator application. Then we show that $\mathcal{R}$ is a ground branching bisimulation. According to Definition 3.5, it boils down to showing that whenever $\langle P \parallel R,\rho\rangle\ \mathcal{S}\ \langle Q \parallel R,\sigma\rangle$, the following two requirements hold:

i. If $\langle P \parallel R,\rho\rangle \rightsquigarrow_{\mathcal{R}}\xrightarrow{\ell} C \in Con/\mathcal{R}$ and $\neg(\ell = \tau \wedge C = [\langle P \parallel R,\rho\rangle]_{\mathcal{R}})$, then $\langle Q \parallel R,\sigma\rangle \rightsquigarrow_{\mathcal{R}}\xrightarrow{\ell} C$;
ii. If $\langle P \parallel R,\rho\rangle \rightsquigarrow_{\mathcal{R}}\xrightarrow{p} C \in Con/\mathcal{R}$ such that $C \neq [\langle P \parallel R,\rho\rangle]_{\mathcal{R}}$, then $\langle Q \parallel R,\sigma\rangle \rightsquigarrow_{\mathcal{R}}\xrightarrow{p} C$.

Below we only give detailed proof for the first statement. Proof for the second one is similar and hence omitted here.

Consider an ℓ-transition $\langle P \parallel R,\rho\rangle \rightsquigarrow_{\mathcal{R}}\xrightarrow{\ell} C$. It consists of a regular ϵ-tree $t_{\langle P\parallel R,\rho\rangle}$ of $\langle P \parallel R,\rho\rangle$ with regard to $\mathcal{R}$ and, for every leaf L of $t_{\langle P\parallel R,\rho\rangle}$, a transition $L \xrightarrow{\ell} L' \in C$. We will construct an ℓ-transition $\langle Q \parallel R,\rho\rangle \rightsquigarrow_{\mathcal{R}}\xrightarrow{\ell} C$ by induction on the structure of $t_{\langle P\parallel R,\rho\rangle}$. It is carried out by a careful case analysis about the immediate transitions that $\langle P \parallel R,\rho\rangle$ can perform.

- The root of $t_{\langle P\parallel R,\rho\rangle}$ can perform the internal transition $\xrightarrow{\tau}$. Three subcases are possible.
  i. This transition is caused by $\mathcal{R}$ solely, that is, $\langle R,\rho\rangle \xrightarrow{\tau} \langle R',U'(\rho)\rangle$ and $\langle P \parallel R,\rho\rangle \xrightarrow{\tau} \langle P \parallel R',U'(\rho)\rangle$ for some R' and super-operator U'. Then $\langle Q \parallel R,\sigma\rangle \xrightarrow{\tau} \langle Q \parallel R',U'(\sigma)\rangle$. As $\simeq$ is closed under super-operator application, we have $\langle P,U'(\rho)\rangle \simeq \langle Q,U'(\sigma)\rangle$, which follows that $\langle P \parallel R',U'(\rho)\rangle\ \mathcal{S}\ \langle Q \parallel R',U'(\sigma)\rangle$.
  ii. This transition is caused by P solely, that is, $\langle P,\rho\rangle \xrightarrow{\tau} \langle P',\rho'\rangle$ and $\langle P \parallel R,\rho\rangle \xrightarrow{\tau} \langle P' \parallel R,\rho'\rangle$ for some P' and ρ'. If $\langle P',\rho'\rangle \simeq \langle P,\rho\rangle$, then $\langle P' \parallel R,\rho'\rangle\ \mathcal{S}\ \langle Q \parallel R,\sigma\rangle$. If $\langle P',\rho'\rangle \not\simeq \langle P,\rho\rangle$, then $\langle Q,\sigma\rangle \rightsquigarrow_{\simeq}\xrightarrow{\tau} [\langle P',\rho'\rangle]_{\simeq}$. For every leaf $\langle Q'',\sigma''\rangle$ in the regular ϵ-tree of $\langle Q,\sigma\rangle$, there exists a configuration $\langle Q',\sigma'\rangle$ such that $\langle Q'',\sigma''\rangle \xrightarrow{\tau} \langle Q',\sigma'\rangle \in [\langle P',\rho'\rangle]_{\simeq}$. We then have $\langle Q'' \parallel R,\sigma''\rangle\ \mathcal{S}\ \langle Q \parallel R,\sigma\rangle\ \mathcal{S}\ \langle P \parallel R,\rho\rangle\ \mathcal{R}\ \langle P' \parallel R,\rho'\rangle\ \mathcal{S}\ \langle Q' \parallel R,\sigma'\rangle$. Thus $\langle Q \parallel R,\sigma\rangle$, $\langle Q'',\sigma''\rangle$ and $\langle Q' \parallel R,\sigma'\rangle$ are related by $\mathcal{R}$. We can then continue to construct an ϵ-tree for $\langle Q' \parallel R,\sigma'\rangle$ by induction on the structure of the ϵ-tree of $\langle P' \parallel R,\rho'\rangle$.
  iii. This transition is induced by an interaction between quantum process P and R. Then w.l.o.g., we can assume that

$$\langle P,\rho\rangle \xrightarrow{c?q} \langle P',\rho\rangle,\ \langle R,\rho\rangle \xrightarrow{c!q} \langle R',\rho\rangle,$$

  and $\langle P \parallel R,\rho\rangle \xrightarrow{\tau} \langle P' \parallel R',\rho\rangle$. Then $\langle R,\eta\rangle \xrightarrow{c!q} \langle R',\eta\rangle$ for any $\eta \in \mathcal{D}(\mathcal{H})$. From the assumption that $\langle P,\rho\rangle \simeq \langle Q,\sigma\rangle$, we have $\langle Q,\sigma\rangle \rightsquigarrow_{\simeq}\xrightarrow{c?q} [\langle P',\rho\rangle]_{\simeq}$. For every leaf $\langle Q'',\sigma''\rangle$ in the regular ϵ-tree of $\langle Q,\sigma\rangle$, there exists some Q' such that $\langle Q'',\sigma''\rangle \xrightarrow{c?q} \langle Q',\sigma''\rangle \in [\langle P',\rho'\rangle]_{\simeq}$. These together give us $\langle Q'' \parallel R,\sigma''\rangle \xrightarrow{\tau} \langle Q' \parallel R',\sigma''\rangle$ and $\langle P' \parallel R',\rho\rangle\ \mathcal{S}\ \langle Q' \parallel R',\sigma''\rangle$.
- The root of $t_{\langle P\parallel R,\rho\rangle}$ can perform the probabilistic transition $\xrightarrow{\coprod_{i\in[k]} p_i\tau}$. Two subcases are possible.
  i. This transition is caused by R solely, that is, $\langle R,\rho\rangle \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]}\langle R_i,U_i(\rho)\rangle$ and

$$\langle P \parallel R,\rho\rangle \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]}\langle P \parallel R_i,U_i(\rho)\rangle$$

  for some super-operators U_i. Then we have $\langle Q \parallel R,\sigma\rangle \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]}\langle Q \parallel R_i,U_i(\sigma)\rangle$. As $\simeq$ is closed under super-operator application, we have $\langle P,U_i(\rho)\rangle \simeq \langle Q,U_i(\sigma)\rangle$ for all $i \in [k]$, and then $\langle P \parallel R_i,U_i(\rho)\rangle\ \mathcal{S}\ \langle Q \parallel R_i,U_i(\sigma)\rangle$ by the definition of $\mathcal{S}$.
  ii. This transition is caused by P solely, that is, $\langle P,\rho\rangle \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]}\langle P_i,\rho_i\rangle$ and

$$\langle P \parallel R,\rho\rangle \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]}\langle P_i \parallel R,\rho_i\rangle.$$

  There are two cases. In the first case $\langle P_i,\rho_i\rangle \simeq \langle P,\rho\rangle$ for all $i \in [k]$, then $\langle P_i \parallel R,\rho_i\rangle\ \mathcal{S}\ \langle Q \parallel R,\sigma\rangle$ for all $i \in [k]$. In the second case, without loss of generality, we assume $\langle P_1,\rho_1\rangle \not\simeq \langle P,\rho\rangle$. Let $r = \mathsf{P}_{\simeq}\left(\langle P,\rho\rangle \xrightarrow{\coprod_{i\in[k]} p_i\tau} [\langle P_1,\rho_1\rangle]_{\simeq}\right)$. As $\langle P,\rho\rangle \simeq \langle Q,\sigma\rangle$, we have $\langle Q,\sigma\rangle \rightsquigarrow_{\simeq}\xrightarrow{r} [\langle P_1,\rho_1\rangle]_{\simeq}$. For every leaf $\langle Q'',\sigma''\rangle$ in the regular ϵ-tree of $\langle Q,\sigma\rangle$, $\langle Q'',\sigma''\rangle \xrightarrow{\coprod_{j\in[k']} q_j\tau} \coprod_{j\in[k']}\langle Q_j,\sigma_j\rangle$ such that $\mathsf{P}_{\simeq}\left(\langle Q,\sigma\rangle \xrightarrow{\coprod_{j\in[k']} q_j\tau} [\langle P_1,\rho_1\rangle]_{\simeq}\right) = r$. For these $\langle Q_j,\sigma_j\rangle \in [\langle P_1,\rho_1\rangle]_{\simeq}$, we see that $\langle P_1 \parallel R,\rho_1\rangle\ \mathcal{S}\ \langle Q_j \parallel R,\sigma_j\rangle$. We then continue to construct an ϵ-tree for $\langle Q_j \parallel R,\sigma_j\rangle$ by induction on the structure of the ϵ-tree of $\langle P_i \parallel R,\rho_1\rangle$.
- The root of $t_{\langle P\parallel R,\rho\rangle}$ can perform the external transition $\xrightarrow{\ell}$. This is similar to the first case and thus omitted here. □

Similar to the classical value-passing CCS, as a relation between quantum processes, $\simeq$ is preserved by all constructors of qCCS.

Theorem 3.13 *(Congruence property). If $P \simeq Q$ and $P_i \simeq Q_i$ for all $i \in I$ then*

(1) $\sum_{i\in I} \lambda_i.P_i \simeq \sum_{i\in I} \lambda_i.Q_i$,
where $\lambda_i \in \{\tau, c?x, c!e, \underline{c}?q, \underline{c}!q, U[\tilde{q}], M[\tilde{q};x]\}$;
(2) $P \parallel R \simeq Q \parallel R$;
(3) $P[f] \simeq Q[f]$;
(4) $P\backslash L \simeq Q\backslash L$;
(5) $\textbf{if } b \textbf{ then } P \simeq \textbf{if } b \textbf{ then } Q$.

Proof. The proof for (1) is similar to Theorem 38 of [7], and the others are direct from Lemma 1. □

4. Equivalence checking algorithm

Efficient algorithms for checking whether an implementation realizes the specification faithfully or whether two implementations are equivalent to each other are quite important in practice. So far, all verification algorithms on qCCS (with semantics given in Fig. 1) in the literature are restricted to ground version [22,6]. The reason is that verification for closeness under super-operator application is generally infeasible due to the infinity of all possible super-operators. In this section, we will build a polynomial algorithm which checks ground branching bisimilarity for quantum processes.

4.1. Description of the algorithm

Before giving the description of the algorithm, we first introduce some necessary notations and conceptions. Given a non-empty set $S \subseteq Con$, a *partition* $\mathcal{X}$ of S is a collection of pairwise disjoint subsets of R whose union is equal to S. Given a partition $\mathcal{X}$ of set S and a configuration $A \in S$, we use $\mathcal{E}_{\mathcal{X}}$ to denote the equivalence relation induced by $\mathcal{X}$ and $[A]_{\mathcal{X}}$ to denote the equivalence class containing A. Suppose $\mathcal{X}_1$ and $\mathcal{X}_2$ are two partitions of some set S, if for each $C \in \mathcal{X}_2$ there exists some $C' \in \mathcal{X}_1$ such that $C \subseteq C'$, then we say that $\mathcal{X}_2$ is *finer* than $\mathcal{X}_1$ (or equivalently, $\mathcal{X}_1$ is *coarser* than $\mathcal{X}_2$). Given a configuration $A \in Con$, the set of configurations reachable from A is denoted by R_A.

Since branching bisimilarity [25] is undecidable for the full CCS model with parallel composition and restriction operators, the general equivalence checking problem for ground branching bisimilarity on the full qCCS model is also undecidable. Thus in this section, we will focus on finite-state systems. Therefore the set R_A is assumed to be finite to any quantum process A. In this case, an ϵ-tree can be characterized by a finite directed graph where nodes of the same label are merged, just as the following definition describes.

Definition 4.1 *(ϵ-graph [29]).* An ϵ-graph $G^A_{\mathcal{E}}$ of A with regard to an equivalence relation $\mathcal{E}$ is a weighted directed graph formed by merging nodes of the same label in an ϵ-tree $t^A_{\mathcal{E}}$ into one node. A node in $G^A_{\mathcal{E}}$ is called a sink node if its out-degree is 0. We denote the set of all sink nodes of $G^A_{\mathcal{E}}$ by $sink(G^A_{\mathcal{E}})$.

Based on the ϵ-graph, we present our equivalence checking algorithm for ground branching bisimilarity: **GroundBranBisim**(A, B).

GroundBranBisim(A, B) Equivalence Checking Algorithm.

```
Input: Two configurations $A$, $B$
Output: Whether $A \simeq_{gb} B$
 1: Compute $R := R_A \cup R_B$
 2: $\mathcal{X} := \{R\}$
 3: $\mathcal{X} :=$ PreRefine($\mathcal{X}$)
 4: $(b, (C_1, \alpha, C_2)) =$ FindSplit($\mathcal{X}$)
 5: while $b = $ T do
 6:     $\mathcal{X} :=$ Refine($\mathcal{X}, (C_1, \alpha, C_2)$)
 7:     $(b, (C_1, \alpha, C_2)) =$ FindSplit($\mathcal{X}$)
 8: if $[A]_{\mathcal{X}} = [B]_{\mathcal{X}}$ then
 9:     return T
10: else
11:     return F
```

The algorithm starts with computing the set R of all configurations reachable from A or B. It then iteratively constructs set $R/\simeq_{gb}$, i.e., the set of equivalence classes of R under $\simeq_{gb}$. The iteration procedure starts with the coarsest partition $\mathcal{X} = \{R\}$, and then keeps refining the current partition by analyzing one-step difference until $\mathcal{X}$ satisfies the definition of ground branching bisimulation. It will surely terminate as the initial partition is finite, and after each iteration we will obtain a strictly finer partition. Moreover, when it terminates, the final partition is just $R/\simeq_{gb}$. Hence to decide whether A and B are bisimilar, we only need to check whether A and B belong to the same equivalence class of the final partition. Now we describe the algorithm in detail.

4.1.1. Preprocess the partition

After computing the reachable configurations of A and B, the procedure **PreRefine**$(\mathcal{X})$ is called to make a preprocessing of the partition. It simply refines the partition $\mathcal{X}$ according to $qv(P)$ and $\mathrm{tr}_{qv(P)}(\rho)$ for each configuration $\langle P, \rho\rangle$ in the partition. After this procedure, all configurations in the same subset of the refined partition will have the same free quantum variables and equal partial trace.

4.1.2. Find the splitter

According to the definition of ground branching bisimilarity, the splitter should be defined based on ℓ-transitions and p-transitions. Let us take ℓ-transition for example. Given an equivalence $\mathcal{E}$ on Con and two equivalence classes $C, C' \in \mathcal{E}/Con$. Now suppose there are two configurations $A_1, A_2 \in C$ such that the ℓ-transition $A_1 \rightsquigarrow_{\mathcal{E}}\xrightarrow{\ell} C'$ of A_1 cannot be bisimulated by A_2. In this case, the equivalence class C can be refined further using a splitter defined by the ℓ-transition. According to the definition of ℓ-transition, $A_1 \rightsquigarrow_{\mathcal{E}}\xrightarrow{\ell} C'$ is associated with a regular ϵ-tree $t^{A_1}_{\mathcal{E}}$ satisfying that $L \xrightarrow{\ell} L' \in C'$ for every leaf L of $t^{A_1}_{\mathcal{E}}$. As L and A_1 belong to the same equivalence class C, the difference between A_1 and A_2 about the existence of the ℓ-transition can be exactly characterized by the difference between L and A_2 regarding the existence of the *one-step* ℓ action. Thus when we use the ℓ-transition $A_1 \rightsquigarrow_{\mathcal{E}}\xrightarrow{\ell} C'$ to define a splitter, it can be interpreted as a one-step state-changing transition between *equivalence classes* C and C'. Consequently, the pair (ℓ, C') can be used as a splitter for C.

The following definition of $TranPair$ then captures the above observation about splitters. Given a partition $\mathcal{X}$ and an equivalence class $C \in \mathcal{X}$, for any configuration $A \in C$, we define

$$
\begin{aligned}
&TranPair(A)\\
&= \{(\ell, C') \mid A \xrightarrow{\ell} A' \in C' \wedge (\ell \neq \tau \vee C' \neq C)\} \ \cup\\
&\quad \{(\widehat{p}\tau, C') \mid A \xrightarrow{\coprod_{i\in I} p_i\tau} \textstyle\coprod_{i\in I} A_i \wedge (\exists i \in I.\ A_i \in C' \neq C)\},
\end{aligned}
$$

where the symbol $\widehat{p}\tau$ is used to indicate any probabilistic action $p\tau$ while do not specify the concrete probability value p. We further define the set of all possible splitters for C as follows:

$$TranPair(C) = \bigcup_{A\in C} TranPair(A).$$

FindSplit$(\mathcal{X})$ Find a splitter of the current partition.

```
Input: A partition $\mathcal{X}$
Output: A splitter of $\mathcal{X}$ if there is one
 1: for all $C \in \mathcal{X}$ do
 2:     Compute the set $TranPair(C)$ for $C$
 3:     for all $(\alpha, C') \in TranPair(C)$ do
 4:         if $\alpha = \widehat{p}\tau$ then
 5:             $\mathcal{X}_{Split} \leftarrow$ SplitP($C, \widehat{p}\tau, C'$)
 6:             if $|\mathcal{X}_{Split}| > 1$ then
 7:                 return (T, ($C, \widehat{p}\tau, C'$))
 8:         else if $\alpha = \ell$ then
 9:             $\mathcal{X}_{Split} \leftarrow$ SplitL($C, \ell, C'$)
10:             if $|\mathcal{X}_{Split}| > 1$ then
11:                 return (T, ($C, \ell, C'$))
12: return (F, ($\emptyset, \tau, \emptyset$))
```

Now for each $(\alpha, C') \in TranPair(C)$, according to the type of α, **FindSplit$(\mathcal{X})$** will invoke **SplitP** and **SplitL** respectively.

- **SplitP**$(C_1, \widehat{p}\tau, C_2)$: It splits C_1 by using the splitter $(\widehat{p}\tau, C_2)$, which is associated with a p-transition. Firstly, for each configuration $A \in$

SplitP($\mathcal{C}_1, \widehat{p}\tau, \mathcal{C}_2$).

```
Input: A triple $(\mathcal{C}_1, \widehat{p}\tau, \mathcal{C}_2)$
Output: A partition of $\mathcal{C}_1$ according to the splitter $(\widehat{p}\tau, \mathcal{C}_2)$
1: Construct the set $Init = \{A \in \mathcal{C}_1 \mid A \xrightarrow{\coprod_{i\in I} p_i\tau} \coprod_{i\in I} A_i \wedge (\exists i \in I. A_i \in \mathcal{C}_2)\}$
2: Compute the partition $\mathcal{X}_{Init} = Init/{=_p}$
3: $\mathcal{X}_{Split} \leftarrow$ SplitDeltaP$(\mathcal{C}_1, \mathcal{X}_{Init})$
4: return $\mathcal{X}_{Split}$
```

SplitDeltaP($\mathcal{C}_1, \mathcal{X}_{Init}$).

```
Input: A pair $(\mathcal{C}_1, \mathcal{X}_{Init})$
Output: A partition of $\mathcal{C}_1$
1: $K \leftarrow |\mathcal{X}_{Init}|$, $\{\mathcal{B}_1, \cdots, \mathcal{B}_K\} \leftarrow \mathcal{X}_{Init}$
2: for all $i \in [K]$ do
3:     $toCon \leftarrow \mathbf{T}$, $\mathcal{X}_{Dec} \leftarrow \mathcal{B}_i$, $\mathcal{X}_{Und} \leftarrow \mathcal{C}_1 \setminus \bigcup_{j\in[K]} \mathcal{B}_j$
4:     while $toCon = \mathbf{T}$ do
5:         $toCon \leftarrow \mathbf{F}$
6:         for all $B \in \mathcal{X}_{Und}$ do
7:             if there exists $B'$ such that $B \xrightarrow{\tau} B' \in \mathcal{X}_{Dec}$, or there exists $\{B_j\}_{j\in J}$
    such that $B \xrightarrow{p_j\tau} B_j \in \mathcal{X}_{Dec}$ holds for all $j \in J$ then
8:                 $Array[B, i] \leftarrow 1$, $toCon \leftarrow \mathbf{T}$
9:                 $\mathcal{X}_{Und} \leftarrow \mathcal{X}_{Und} \setminus \{B\}$, $\mathcal{X}_{Dec} \leftarrow \mathcal{X}_{Dec} \cup \{B\}$
10: for all $i \in [K]$ do
11:     $\mathcal{B}'_i \leftarrow \mathcal{B}_i$
12: for all $B \in \mathcal{C}_1 \setminus \bigcup_{i\in[K]} \mathcal{B}_i$ do
13:     $Num \leftarrow 0$, $m \leftarrow 0$
14:     for all $j \in [K]$ do
15:         if $Array[B, j] = 1$ then
16:             $Num \leftarrow Num + 1$, $m \leftarrow j$
17:     if $Num = 1$ then
18:         $\mathcal{B}'_m \leftarrow \mathcal{B}'_m \cup \{B\}$
19: $\mathcal{X}_{Rem} \leftarrow \mathcal{C}_1 \setminus \bigcup_{i\in[K]} \mathcal{B}'_i$
20: if $\mathcal{X}_{Rem} = \emptyset$ then
21:     return $\{\mathcal{B}'_1, \cdots, \mathcal{B}'_K\}$
22: else
23:     return $\{\mathcal{B}'_1, \cdots, \mathcal{B}'_K\} \cup \mathcal{X}_{Rem}$
```

$\mathcal{C}_1$, if it can perform a state-changing probabilistic transition, then we added it to the set $Init$. Then we construct the partition $\mathcal{X}_{Init}$ for the set $Init$ according to the equivalence relation $=_p$, which is defined as $A =_p A'$ if and only if

$$\mathsf{P}_{\mathcal{E}_\mathcal{X}}(A \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{C}_2) = \mathsf{P}_{\mathcal{E}_\mathcal{X}}(A' \xrightarrow{\coprod_{j\in[k']} p'_j\tau} \mathcal{C}_2).$$

In other words, configurations in $Init$ are partitioned according to their respective normalized probability of leaving $\mathcal{C}_1$ to $\mathcal{C}_2$. Next the procedure **SplitDeltaP** is invoked to split $\mathcal{C}_1$ according to $\mathcal{X}_{Init}$. Now suppose $\mathcal{X}_{Init} = \{\mathcal{B}_1, \cdots, \mathcal{B}_K\}$ (where $K = |\mathcal{X}_{Init}|$) and $\mathcal{X}_{Und} = \mathcal{C}_1 \setminus \bigcup_{i\in[K]} \mathcal{B}_i$. More specifically, for all $i \in [K]$, **SplitDeltaP** will add all configurations $B \in \mathcal{X}_{Und}$ which satisfy the following conditions (denoted by Δ^i_P) into $\mathcal{B}_i$, and form a new equivalence class $\mathcal{B}'_i$:
– There exists an ϵ-graph $G^B_\mathcal{X}$ with $sink(G^B_\mathcal{X}) \subseteq \mathcal{B}_i$;
– For any other $\mathcal{B}_j \in \mathcal{X}_{Init}$ with $j \neq i$, there does not exist any ϵ-graph $G^B_\mathcal{X}$ with $sink(G^B_\mathcal{X}) \subseteq \mathcal{B}_j$.
The remaining configurations are collected into a set called $\mathcal{X}_{Rem}$. Now $\{\mathcal{B}'_1, \cdots, \mathcal{B}'_K\} \cup \mathcal{X}_{Rem}$ forms a refinement of $\mathcal{C}_1$.
We further explain how to verify the property Δ^i_P for every $i \in [K]$. Here we define a two-dimensional boolean array $Array[B, i]$, with the first entry $B \in \mathcal{C}_1 \setminus \bigcup_{i\in[K]} \mathcal{B}_i$ and the second entry $i \in [K]$. We will ensure that $Array[B, i]$ is set to 1 if and only if there exists an ϵ-graph $G^B_\mathcal{X}$ with $sink(G^B_\mathcal{X}) \subseteq \mathcal{B}_i$. This will then imply that B satisfies property Δ^i_P iff $Array[B, i] = 1$ and $Array[B, j] = 0$ for all $j \neq i$. To check the existence of such ϵ-graphs, we can proceed as follows for each $i \in [K]$:

1. Initialize $\mathcal{X}_{Dec}$ to $\mathcal{B}_i$ and $\mathcal{X}_{Und}$ to $\mathcal{C}_1 \setminus \bigcup_{j\in[K]} \mathcal{B}_j$.
2. For each process $B \in \mathcal{X}_{Und}$, we consider the immediate transitions of B. If there exists B' such that $B \xrightarrow{\tau} B' \in \mathcal{X}_{Dec}$ (i.e., this nondeterministic internal transition arrives $\mathcal{X}_{Dec}$), or there exists $\{B_j\}_{j\in J}$ such that $B \xrightarrow{p_j\tau} B_j \in \mathcal{X}_{Dec}$ holds for all $j \in J$ (i.e., all branches of this probabilistic transition arrive $\mathcal{X}_{Dec}$), then we will set $Array[B, i] = 1$, delete it from $\mathcal{X}_{Und}$, and add it into $\mathcal{X}_{Dec}$.
3. Repeat step 2 until the set $\mathcal{X}_{Dec}$ does not change, the resulting set $\mathcal{X}_{Dec}$ will contain exactly these configurations B satisfying that there exists an ϵ-graph $G^B_\mathcal{X}$ with $sink(G^B_\mathcal{X}) \subseteq \mathcal{B}_i$.

SplitL($\mathcal{C}_1, \ell, \mathcal{C}_2$).

```
Input: A triple $(\mathcal{C}_1, \ell, \mathcal{C}_2)$
Output: A partition of $\mathcal{C}_1$ according to the splitter $(\ell, \mathcal{C}_2)$
1: Construct the set $\mathcal{B}_{Init} = \{B \in \mathcal{C}_1 \mid B \xrightarrow{\ell} B' \in \mathcal{C}_2\}$
2: $\mathcal{X}_{Split} \leftarrow$ SplitDeltaL$(\mathcal{C}_1, \mathcal{B}_{Init})$
3: return $\mathcal{X}_{Split}$
```

- **SplitL**($\mathcal{C}_1, \ell, \mathcal{C}_2$): It splits $\mathcal{C}_1$ by using the splitter $(\ell, \mathcal{C}_2)$, which is associated with an ℓ-transition. Now for each configuration $A \in \mathcal{C}_1$, if it can perform an ℓ action and evolve into a configuration $A' \in \mathcal{C}_2$, we add it into the set $\mathcal{B}_{Init}$. Then the procedure **SplitDeltaL** will add all configurations $A \in \mathcal{C}_1 \setminus \mathcal{B}_{Init}$ which satisfies the following condition (denoted by Δ_L) into $\mathcal{B}_{Init}$, and form a new equivalence class $\mathcal{B}_T$:
  – There exists an ϵ-graph $G^B_\mathcal{X}$ such that $sink(G^B_\mathcal{X}) \subseteq \mathcal{B}_{Init}$.
  The remaining configurations will be put into the set $\mathcal{B}_F$. Now $\mathcal{B}_T \cup \mathcal{B}_F$ forms a refinement of $\mathcal{C}_1$.

SplitDeltaL($\mathcal{C}_1, \mathcal{B}_{Init}$).

```
Input: A pair $(\mathcal{C}_1, \mathcal{B}_{Init})$
Output: A partition of $\mathcal{C}_1$
1: $toCon \leftarrow \mathbf{T}$, $\mathcal{B}_{Und} \leftarrow \mathcal{C}_1 \setminus \mathcal{B}_{Init}$, $\mathcal{B}_{Dec} \leftarrow \mathcal{B}_{Init}$
2: while $toCon = \mathbf{T}$ do
3:     $toCon \leftarrow \mathbf{F}$
4:     for all $B \in \mathcal{B}_{Und}$ do
5:         if there exists $B'$ such that $B \xrightarrow{\tau} B' \in \mathcal{B}_{Dec}$, or there exists $\{B_j\}_{j\in J}$
    such that $B \xrightarrow{p_j\tau} B_j \in \mathcal{B}_{Dec}$ holds for all $j \in J$ then
6:             $Array[B] \leftarrow 1$, $toCon \leftarrow \mathbf{T}$
7:             $\mathcal{B}_{Und} \leftarrow \mathcal{B}_{Und} \setminus \{B\}$, $\mathcal{B}_{Dec} \leftarrow \mathcal{B}_{Dec} \cup \{B\}$
8: $\mathcal{B}_T \leftarrow \mathcal{B}_{Init}$
9: for all $B \in \mathcal{C}_1 \setminus \mathcal{B}_{Init}$ do
10:     if $Array[B] = 1$ then
11:         $\mathcal{B}_T \leftarrow \mathcal{B}_T \cup \{B\}$
12: $\mathcal{B}_F \leftarrow \mathcal{C}_1 \setminus \mathcal{B}_T$
13: if $\mathcal{B}_F = \emptyset$ then
14:     return $\{\mathcal{B}_T\}$
15: else
16:     return $\{\mathcal{B}_T, \mathcal{B}_F\}$
```

4.1.3. Carry out the refinement

Once a splitter $(\alpha, \mathcal{C}_2)$ for some $\mathcal{C}_1$ has been identified, the procedure **Refine**($\mathcal{X}, (\mathcal{C}_1, \alpha, \mathcal{C}_2)$) can refine the partition $\mathcal{X}$ by using this splitter. According to the type of α, this procedure will invoke **SplitP** or **SplitL** correspondingly to obtain the refined partition.

4.2. Correctness and complexity

As we have discussed earlier, **GroundBranBisim** will terminate in finite steps. We then show its correctness as the following proposition.

Proposition 4.2 *(Correctness). Given two configurations A and B, the algorithm* ***GroundBranBisim***(A, B) *returns true if and only if* $A \simeq_{gb} B$.

Refine $(\mathcal{X}, (C_1, \alpha, C_2))$ Refine $\mathcal{X}$ according to the splitter (α, C_2) for C_1.

```
Input: A partition $\mathcal{X}$ and a splitter $(\alpha, C_2)$ for $C_1$
Output: A refined partition $\mathcal{X}_{Refine}$ for $\mathcal{X}$
1: if $\alpha = \widehat{p}\tau$ then
2:     $\mathcal{X}_{Split} \leftarrow$ SplitP$(C_1, \varphi\tau, C_2)$
3: else if $\alpha = \ell$ then
4:     $\mathcal{X}_{Split} \leftarrow$ SplitL$(C_1, \ell, C_2)$
5: $\mathcal{X}_{Refine} \leftarrow \mathcal{X} \setminus \{C_1\} \cup \mathcal{X}_{Split}$
6: return $\mathcal{X}_{Refine}$
```

Proof. The algorithm first computes the set R that contains all configurations reachable from A and B. Then it forms the coarsest partition $\mathcal{X} = \{R\}$ and refines it. We will show that when $\mathcal{X}$ cannot be refined further, $\mathcal{X} = R/\simeq_{gb}$. It then implies that $[A]_{\mathcal{X}} = [B]_{\mathcal{X}}$ iff $A \simeq_{gb} B$.

Suppose there exists a splitter (α, C_2) for some equivalence class C_1, the algorithm will start a new run of **Refine**. Let k denote the number of executions of **Refine**, and $\mathcal{X}_k$ be the current partition $\mathcal{X}$ after the execution of **Refine**$_k$. It is easy to see that each $\mathcal{X}_{k+1}$ is strictly finer than $\mathcal{X}_k$. Let n denote the total number of executions of **Refine**, then $n \leq |R|$.

We still need to show that each $\mathcal{X}_k$ is no finer than $R/\simeq_{gb}$, in other words $\simeq_{gb} \subseteq \mathcal{E}_{\mathcal{X}_k}$ holds for all $k \in [n]$, where $\mathcal{E}_{\mathcal{X}_k}$ is the induced equivalence of $\mathcal{X}_k$. This is proved by induction on k. The base case is obvious. Now suppose $\simeq_{gb} \subseteq \mathcal{E}_{\mathcal{X}_k}$, we will show that $\simeq_{gb} \subseteq \mathcal{E}_{\mathcal{X}_{k+1}}$. Consider the execution of **Refine**$_{k+1}$. If the splitter for some C_1 is of the form $(\widehat{p}\tau, C_2)$, we refine the set C_1 into $\{\mathcal{B}_1, \cdots, \mathcal{B}_K\} \cup \mathcal{X}_{Rem}$. Now suppose $A \in \mathcal{B}_i$ for some $i \in [K]$ and $B \notin \mathcal{B}_i$, then $(A, B) \notin \mathcal{E}_{\mathcal{X}_{k+1}}$. We need to show that $(A, B) \notin \simeq_{gb}$. We know that A satisfies property Δ^i_P while B does not. Hence there exists a p-transition $A \twoheadrightarrow_{\mathcal{E}_{\mathcal{X}_k}} \xrightarrow{p_i} C_2$ of A which cannot be bisimulated by B. Thus by induction hypothesis $\simeq_{gb} \subseteq \mathcal{E}_{\mathcal{X}_k}$, we can obtain that $(A, B) \notin \simeq_{gb}$. □

Before concluding this section, we analyze the time complexity of our equivalence checking algorithm and compare it with the one in [22] for ground weak bisimilarity.

Theorem 4.3. *Let the number of configurations reachable from A and B be n. The time complexity of the algorithm **GroundBranBisim**(A, B) is $\mathcal{O}(n^7)$.*

Proof. Since $K = |\mathcal{X}_{Init}| \leq |C_1| \leq n$, the **for** loop at stage 2 of procedure **SplitDeltaP** can run no more than n times. Since $|\mathcal{X}_{Und}| \leq |C_1| \leq n$, the **while** loop at stage 4 can run no more n times. In each iteration of the while loop, all possible one-step transitions for configurations in $\mathcal{X}_{Und}$ need to be examined and can be done in $\mathcal{O}(n^2)$ time. The resulting complexity of **SplitDeltaP** is therefore $\mathcal{O}(n \cdot n \cdot n^2) = \mathcal{O}(n^4)$, which implies that the complexity of procedure **SplitP** is also $\mathcal{O}(n^4)$. Similarly, we can obtain that the complexity of **SplitL** is $\mathcal{O}(n^3)$. Since there are n configurations in all and each configuration has at most n different transitions, the inner **for** loop at stage 3 in procedure **FindSplit** can run no more than n^2 times. Since **SplitP** has complexity $\mathcal{O}(n^4)$ and **SplitL** $\mathcal{O}(n^3)$, it follows that the overall complexity of **FindSplit** is $\mathcal{O}(n^6)$. Every time **FindSplit** is executed and returns T, we will obtain a strictly finer partition. Then the **while** loop at stage 5 of **GroundBranBisim** can run at most n times. It is not hard to see that the complexity of **PreRefine** and **Refine** are $\mathcal{O}(n^2)$ and $\mathcal{O}(n^4)$, respectively. As we already know that **FindSplit** works in $\mathcal{O}(n^6)$ steps, summing the total shows that **GroundBranBisim** executes $\mathcal{O}(n^2 + n \cdot (n^6 + n^4)) = \mathcal{O}(n^7)$ stages. □

Remark 4.4. In [22], Qin et al. give a polynomial-time equivalence checking algorithm for the ground weak bisimilarity proposed in [4]. Here we give a brief analysis of the lower bound of the algorithm in [22]. Let the number of configurations reachable from A and B be n. As noted in [22], the total number of state pairs examined in the algorithm is $\Theta(n^4)$. Now consider any such pair (A, B). For each transition of A, the algorithm uses the technique in [24] to check if there exists a weak matching transition for B. In [22], the authors reduce the weak matching transition checking problem to a linear programming problem. The number of variables and constraints in the LP problem are both $\Theta(n^2)$. Since the best algorithm for solving the LP problem with size N has time complexity $\Omega(N^2)$, the time complexity to solving the reduced LP problem would be $\Omega(n^4)$. Thus the overall time complexity of the verification algorithm in [22] is $\Omega(n^8)$.

5. Conclusions and future work

In this paper, we propose a new branching bisimilarity for qCCS, which can be seen as a conservative generalization of van Glabbeek's branching bisimilarity for classical CCS. Compared to the previous work, our branching bisimilarity is the first congruence relation that preserves branching structures of the quantum processes. We also propose a polynomial-time checking algorithm for the ground branching bisimilarity and show that it is computationally more efficient than the one in [22] for ground weak bisimilarity. In the future, we would like to develop efficient tools for automatic branching bisimilarity checking, especially for applications like quantum communication protocols verification.

CRediT authorship contribution statement

Hao Wu: Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Qizhe Yang:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Huan Long:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

We are grateful to Prof. Yuxi Fu and Prof. Yuxin Deng for the valuable discussion. We thank BASICS members for their interest. We also thank the editor and the reviewers for their valuable comments and suggestions. The support from the National Natural Science Foundation of China (62072299) is acknowledged.

References

[1] T. Basten, Branching bisimilarity is an equivalence indeed!, Inf. Process. Lett. 58 (3) (1996) 141–147, https://doi.org/10.1016/0020-0190(96)00034-8.

[2] V. Castiglioni, S. Tini, Raiders of the lost equivalence: probabilistic branching bisimilarity, Inf. Process. Lett. 159–160 (2020) 105947, https://doi.org/10.1016/j.ipl.2020.105947.

[3] T.A.S. Davidson, Formal Verification Techniques Using Quantum Process Calculus, Ph.D. thesis, University of Warwick, 2012.

[4] Y. Deng, Y. Feng, Open bisimulation for quantum processes, in: J.C.M. Baeten, T. Ball, F.S. de Boer (Eds.), Theoretical Computer Science, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 119–133.

[5] Y. Deng, Y. Feng, Open bisimulation for quantum processes, (full version), https://doi.org/10.48550/arXiv.1201.0416, 2012.

[6] Y. Feng, Y. Deng, M. Ying, Symbolic bisimulation for quantum processes, ACM Trans. Comput. Log. 15 (2) (2014) 14, https://doi.org/10.1145/2579818.

[7] Y. Feng, R. Duan, Z. Ji, M. Ying, Probabilistic bisimulations for quantum processes, Inf. Comput. 205 (11) (2007) 1608–1639, https://doi.org/10.1016/j.ic.2007.08.001.

[8] Y. Feng, R. Duan, M. Ying, Bisimulation for quantum processes, in: Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 523–534.

[9] R.P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. 21 (6) (1982) 467–488, https://doi.org/10.1007/BF02650179.

[10] Y. Fu, Theory of interaction, Theor. Comput. Sci. 611 (2016) 1–49, https://doi.org/10.1016/j.tcs.2015.07.043.

[11] Y. Fu, Model independent approach to probabilistic models, Theor. Comput. Sci. 869 (2021) 181–194, https://doi.org/10.1016/j.tcs.2021.04.001.

[12] S.J. Gay, R. Nagarajan, Communicating quantum processes, in: Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'05, Association for Computing Machinery, New York, NY, USA, 2005, pp. 145–157.

[13] S. Graf, J. Sifakis, Readiness semantics for regular processes with silent actions, in: T. Ottmann (Ed.), Automata, Languages and Programming, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1987, pp. 115–125.

[14] M. Hennessy, A proof system for communicating processes with value-passing, Form. Asp. Comput. 3 (4) (1991) 346–366, https://doi.org/10.1007/BF01642508.

[15] M. Hennessy, A. Ingolfsdottir, A theory of communicating processes with value passing, Inf. Comput. 107 (2) (1993) 202–236, https://doi.org/10.1006/inco.1993.1067.

[16] P. Jorrand, M. Lalire, Toward a quantum process algebra, in: Proceedings of the 1st Conference on Computing Frontiers, CF'04, Association for Computing Machinery, New York, NY, USA, 2004, pp. 111–119.

[17] M. Lalire, Relations among quantum processes: bisimilarity and congruence, Math. Struct. Comput. Sci. 16 (3) (2006) 407–428, https://doi.org/10.1017/S096012950600524X.

[18] R. Milner, Communication and Concurrency, Prentice-Hall, Inc., 1989.

[19] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, I, Inf. Comput. 100 (1) (1992) 1–40, https://doi.org/10.1016/0890-5401(92)90008-4.

[20] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, II, Inf. Comput. 100 (1) (1992) 41–77, https://doi.org/10.1016/0890-5401(92)90009-5.

[21] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, New York, 2010, 10th anniversary ed edn.

[22] X. Qin, Y. Deng, W. Du, Verifying quantum communication protocols with ground bisimulation, in: A. Biere, D. Parker (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2020, pp. 21–38.

[23] R. Segala, Modeling and Verification of Randomized Distributed Real-Time Systems, Thesis, Massachusetts Institute of Technology, 1995.

[24] A. Turrini, H. Hermanns, Polynomial time decision algorithms for probabilistic automata, Inf. Comput. 244 (2015) 134–171, https://doi.org/10.1016/j.ic.2015.07.004.

[25] R.J. van Glabbeek, W.P. Weijland, Branching time and abstraction in bisimulation semantics, J. ACM 43 (3) (1996) 555–600, https://doi.org/10.1145/233551.233556.

[26] W.K. Wootters, W.H. Zurek, A single quantum cannot be cloned, Nature 299 (5886) (1982) 802–803.

[27] H. Wu, H. Long, Probabilistic weak bisimulation and axiomatization for probabilistic models, Inf. Process. Lett. 182 (2023) 106399, https://doi.org/10.1016/j.ipl.2023.106399.

[28] M. Ying, Y. Feng, R. Duan, An algebra of quantum processes, ACM Trans. Comput. Log. 10 (3) (2009) 36.

[29] W. Zhang, H. Long, X. Xu, Uniform random process model revisited, in: A.W. Lin (Ed.), Programming Languages and Systems, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 388–404.