Classical Analysis of Quantum Computation
(古典的手法による量子計算の解析)

by

Kentaro Honda
本多健太郎

A Doctor Thesis
博士論文

Submitted to
the Graduate School of the University of Tokyo
on December 11, 2015
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Information Science and Technology
in Computer Science

Thesis Supervisor: Masami Hagiya　萩谷昌己
Professor of Computer Science

ABSTRACT

Quantum computation is based on quantum physics and different from classical computation based on classical physics. It has non-classical characteristics such as quantum superposition principle, uncertainty principle, and no-cloning theorem and has been mainly studied by original methodology. However, it does not mean we cannot investigate quantum computation using traditional methods that have been studied in non-quantum area. We believe that quite a lot of methods are useful to investigate quantum computation and it is important to find such methods. When they are found, they will accelerate the research of quantum computation using the knowledge about them. Moreover, due to their classical nature, they will naturally help us to analyse quantum computation using classical computers. In order to support the idea, we show two classical methodologies, abstract interpretation and classical public-key cryptography, enhance analysis in two topics of quantum computation, entanglement in quantum programming languages and blind quantum computation protocols.

The behaviour of entangled quantum system is counterintuitive. Therefore, it is important to know how entangled states are in quantum programs without executing them. In order to statistically analyse the problem, it was proposed to apply the abstract interpretation technique to the analysis. However, the proposed method does not store information about entangled states and ignores the fact that an operator on multiple qubits may undo entanglements, and hence the method only gives a rough approximation. We combine the method with the stabiliser formalism. We show abstract interpretation can be used to efficiently analyse the existence of entanglement in quantum programs with higher precision.

Blind quantum computation protocols give users having no powerful quantum devices a method to delegate their quantum computation to remote quantum servers without leaking any crucial information about their computation. Some protocols enable users to verify whether the quantum servers honestly do the delegated computation, but no protocols give third parties ways to analyse the computation. We propose a new blind quantum computation protocol using a classical public-key encryption scheme. Our protocol is based on an existing protocol, but third parties, who have only classical computers and check the classical message between users and quantum servers, can analyse the computation and verify whether users obtain the correct outcomes or not.

These results show classical methodology, which achieves a development outside the area of quantum computation, may be useful for investigation of quantum computation.

論文要旨

量子力学に基づく量子計算は，古典物理学に基づいた古典計算とは異なる新たなパラダイムをもたらした．重ね合わせの原理や不確定性原理，複製不可能性定理など古典計算の持たない独特の特性を持ち，様々な独自の手法の考案・発展を通じて研究が進展してきた．しかし，このことは量子計算とは無関係に発展してきた手法，特に量子計算の勃興以前から存在し，研究されている手法が量子計算において全く通用しないことを意味するものではない．少なからぬ古典的な手法は量子計算の研究において役立つものであろう．このような手法は，それに対する古典的な知見を量子計算へと活かすことができるのみならず，その古典的という制約により，古典計算機による量子計算の解析を助けると考えらえる．本論文では，この考えの例証として抽象解釈と古典公開鍵暗号という 2 つの手法が量子計算の解析に有用であることを示す．

量子もつれの存在は，我々の直感に反する効果をもたらす．したがって，量子プログラムの中で何処に量子もつれが存在しているかを知ることは重要である．これを静的に解析する手法として，抽象解釈を応用する手法が提案されていた．しかしながら，この手法ではもつれた状態についての情報を保持せず，一度もつれた状態は測定以外では戻せないなど粗い近似に止まっていた．我々はスタビライザー形式を活用することでこの手法を改善し，抽象解釈によって，古典計算機で可能な範囲にとどまりながらも，より精度の高い量子もつれの存在解析が可能であることを示す．

ブラインド量子計算は，十分な量子デバイスを持たないユーザに対して，実行する計算を完全に秘匿しながら，量子コンピュータを持つ他者に計算を委託する手法を提供する．このプロトコルの中には，委託した計算が正しく実行されているかをユーザ自身が確かめられるようにするものがあるが，ユーザ以外の第三者に解析を許すものは存在しない．これに対して，既存のプロトコルを古典公開鍵暗号を用いて改良した新たなプロトコルを提案する．提案プロトコルでは，ユーザと量子コンピュータとの間で交換される古典情報のみに基づき，ユーザが正しい計算を得られているかを古典計算機しか持たない第三者が検証できるようになることを示す．

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Masami Hagiya for his constant advice and encouragement. I would like to thank Yoshihiko Kakutani for his advice and many discussions. I was helped to study the work in Chapter 4. Takahiro Kubota first pointed out me the possibility of evil Alice. Joseph Fitzsimons pointed out a flow of my first security model. Tomoyuki Morimae helped me to refine my rough idea. I would like to appreciate their help. I would like to thank François Le Gall, Naoki Kobayashi, Hiroshi Imai, Ichiro Hasuo, and Takeshi Koshiba for their valuable comments. I also thank Hasuo laboratory for giving me many opportunities to show my work. Finally, I thank all current and past members of Hagiya laboratory. The work was supported by JSPS Grant-in-Aid for JSPS Fellows.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
  - 1.1.1 Quantum programming language . . . 2
  - 1.1.2 Blind quantum computation protocol . . . 3
- 1.2 Contributions . . . 3
- 1.3 Related work . . . 5
  - 1.3.1 Stabiliser abstract semantics . . . 5
  - 1.3.2 Publicly verifiable blind quantum computation . . . 6
- 1.4 Organisation . . . 7

2 Preliminaries **8**
- 2.1 Notation . . . 8
- 2.2 Quantum computation . . . 9
  - 2.2.1 Rudiments of quantum computation . . . 9
  - 2.2.2 Stabiliser formalism . . . 12
  - 2.2.3 Measurement-based quantum computation . . . 14
- 2.3 Abstract interpretation . . . 15
- 2.4 Encryption scheme . . . 16

3 Stabiliser Abstract Semantics **20**
- 3.1 Basis abstract semantics . . . 20
  - 3.1.1 Quantum imperative language . . . 20
  - 3.1.2 Basis abstract semantics . . . 23
- 3.2 Stabiliser abstract semantics . . . 25
  - 3.2.1 Motivation and idea . . . 25
  - 3.2.2 Stabiliser array . . . 27
  - 3.2.3 Stabiliser domain . . . 31
  - 3.2.4 Stabiliser abstract semantics . . . 34
  - 3.2.5 Comparison with the basis semantics . . . 41
- 3.3 Extended stabiliser abstract semantics . . . 43
  - 3.3.1 Motivation and idea . . . 43
  - 3.3.2 Extended stabiliser domain . . . 45
  - 3.3.3 Extended stabiliser abstract semantics . . . 53

4 Publicly Verifiable Blind Quantum Computation **59**
- 4.1 Verifiable blind quantum computation protocol . . . 59
  - 4.1.1 Delegated quantum computation protocol . . . 59
  - 4.1.2 The Fitzsimons–Kashefi protocol . . . 61
- 4.2 Public Verifiability . . . 63
- 4.3 Towards achievement of public verifiability . . . 66
- 4.4 New verifiable blind quantum computation protocol . . . 67
- 4.5 Publicly verifiable blind quantum computation protocol . . . 74

4.6 Encryption scheme 80
4.6.1 Inattentive evaluation of circuits 80
4.6.2 Adjustment of inattentive evaluation of circuits 82
4.6.3 Publicly verifiable BQC protocol exists 89

5 Conclusions **91**
5.1 Stabiliser abstract semantics 91
5.1.1 Future work 92
5.2 Publicly verifiable blind quantum computation 92
5.2.1 Future work 93

References **95**

Chapter 1

Introduction

1.1 Background

Quantum computation has attracted many researchers by its uniqueness. Though models of quantum computation were first proposed as quantum versions of classical computation models such as Turing machines [26], logical circuits [27], and cellular automata [55], quantum computation today has its own models. They include adiabatic quantum computation [34, 35], topological quantum computation [40], teleportation-based quantum computation [53], and measurement-based quantum computation [93, 94]. These models have no classical counterparts because they essentially use unique features of quantum physics such as the quantum superposition principle, the existence of anyons and quantum entanglement, and quantum measurement. The fact may indicate that quantum computation is fundamentally different from classical computation. The uniqueness of quantum computation also appears in quantum algorithms. Many algorithms show advantages of quantum computation over classical computation. Some algorithms find answers with fewer queries than the classical lower bounds [6, 28, 56]. Some algorithms give ways to solve problems in polynomial time that are believed to be intractable for any classical computer [4, 40, 100].

One may think that the uniqueness and advantages of quantum computation mean that classical computers are useless for studying quantum computation: Classical computers are, so to speak, a restricted form of quantum computers, which is slower than the non-restricted form and cannot capture features of quantum computation. So, appropriate tools to study quantum computation are not classical but quantum computers. However, we claim that it is important to investigate how we can use classical computers to analyse quantum computation. One reason to use classical computers is that there is no certainly universal quantum computer now. Therefore, we cannot use a quantum computer for research of quantum computation. Another reason is difficulty in using a quantum computer. Even if it appears, a quantum computer will be very expensive and most people will not be able to buy it. Moreover, if we can use it, it will be difficult to use a quantum computer for everyday tasks because of decoherence. Users will have to carefully choose which parts of computation they will perform on quantum computers, and they will use classical computers to perform the other parts. Therefore, it is necessary to enable classical computers to analyse quantum computation as far as possible.

In order to efficiently analyse quantum computation, we use classical traditional methods. Quantum computation has its own framework. It is described using linear algebras and has own methodology. These facts make us tend to regard classical methods useless to investigate quantum computation. Some are

indeed used, but many classical methods have not been used or studied well. However, we believe that quite a lot of methods help us to study quantum computation and that if we find useful classical methods, they will accelerate the research of quantum computation using the knowledge about them. As evidences, we show two classical methods are useful to investigate quantum computation. Specifically, we use abstract interpretation and classical public-key encryption schemes to analyse entanglement in quantum programming languages and blind quantum computation protocols. The details of these topics are as follows.

1.1.1 Quantum programming language

It is reasonable to expect that quantum programming languages will be necessary to operate quantum computers when they appear. As preparation for the future, various quantum programming languages have been proposed and studied until now [41, 104]. They include imperative languages [83, 96], functional languages [97, 99, 106], languages for measurement-based quantum computation [23, 24], and process calculi [37, 62], if we can call them programming languages. One direction of the research of quantum programming languages is to sophisticate languages. That makes it easy for programmers to write quantum programs and implement quantum information systems.

Another direction is program analysis. Sophisticated classical programming languages generally enable us to construct large and complicated systems. That is also true for quantum programming languages. For example, a quantum circuit composed of thirty trillion quantum gates was generated using Quipper [54]. It is too difficult to check their correctness by hands, so it is necessary to perform automatic verification and analysis. While verification and analysis are well-studied in classical languages, those in quantum programming languages have their own targets and difficulties. One is linearity. Quantum programming languages are required to be linear, which means that the same variable cannot be used twice at the same time. Since any transformation of a quantum state is linear, quantum data cannot be cloned, but programmers will be able to duplicate quantum data if they can freely use (quantum) variables. Therefore, quantum programming languages are required to forbid them to do so. A well-established method to achieve that is the use of linear type systems [5, 98]. Based on linear logic, these systems guarantee that duplication of any variable does not occur. Another and more interesting target of analysis is quantum entanglement. Although there are linear classical languages, entanglement analysis is unique to quantum languages because the existence of quantum entanglement is a unique feature of quantum physics. Quantum entanglement is known to be a key resource of quantum communication [12, 13] and computation [93, 94]. Unexpected lack of entanglement dooms quantum protocols. Entanglement analysis will help us to find it. Moreover, it will be useful for finding bugs [61]. Quantum algorithms often use ancilla systems. These systems are temporarily used, and they are discarded when they are no longer useful. In order not to disturb the state of the other system, the state of the ancilla is expected not to be entangled with that of the other system. The existence of entanglement suggests that the program contains bugs. Several papers studied analysis of entanglement in quantum programs [42, 43, 61, 87, 89, 92].

1.1.2 Blind quantum computation protocol

A blind quantum computation (BQC) protocol [16] is a communication protocol between two parties, Alice and Bob, to solve the following problem. Suppose Alice has a function f and an input x, and wants to learn $f(x)$. Although the evaluation needs quantum computation, she does not have a quantum computer. Then, she delegates the evaluation to Bob's quantum server. Difficulty is that she does not trust him. While she wants to use his better computational ability, she does not want to let him learn anything about x, $f(x)$, or f. She tries to keep them secret as far as possible. A similar problem was proposed in [36]. At that time, no quantum resources were considered and f was not required to be secure. In the paper, it was shown that some function such as discrete logarithms can be securely delegated in the sense of information theory. Nevertheless, it was shown in [2] that no **NP**-hard computation can be securely delegated unless the polynomial hierarchy collapses at the third level. Around twenty years later, the problem was tackled with quantum resources [7]. The first BQC protocol was proposed in [18]. The protocol achieves universality in the sense that Alice can securely delegate any quantum computation, but it requires Alice to have quantum memory.

The protocol in [16] is a universal BQC protocol. Advantages of the protocol are that she is required to have no quantum device except for a single qubit generator and that it guarantees the perfect security. Especially, Alice who does not have quantum memory can execute the protocol. The main idea of the protocol is use of the measurement-based quantum computation model. The model divides quantum computation into classical parts and quantum parts. Encrypting these parts by the one-time pad and random rotations, the protocol achieves the perfect security.

Since then, much research has been focused on the topic. Some experimentally demonstrated BQC protocols [9, 10]. Some discussed the security of BQC protocols [31, 45]. Another direction of research has been to propose more practical protocols. Although the original protocol used the brickwork states, some proposed BQC protocols using other quantum states such as the decorated RHG lattice [78] and the AKLT state [77]. Furthermore, BQC protocols were proposed in other models including the quantum circuit model [46], the DQC1 model [63], the continuous-variable model [74], and the ancilla-driven model [102]. As there is no efficient BQC protocol without interaction [108], Alice has to interact to Bob in BQC protocols. Several papers proposed more efficient protocols in terms of communication between them [46, 72, 90]. Moreover, many papers weakened the requirements on Alice [16, 32, 68, 79, 80].

The other direction has been about verifiability. In BQC protocols, Bob cannot learn anything about the computation. However, he can tamper with the output, $f(x)$. Because **BQP**, the complexity class of problems solved by quantum computers in polynomial time, is believed not to be included by **NP**, Alice may be unable to confirm the correctness of the given output. A property of a BQC protocol named verifiability guarantees that she can refuse an incorrect output with a high probability. Several papers proposed BQC protocols with the property [3, 39, 58, 75, 76].

1.2 Contributions

The contributions of Chapter 3 are as follows.

- We propose a new abstract semantics of the quantum imperative language

using stabilisers, prove that the semantics is sound, and show the semantics enables strictly finer analysis than an existing abstract semantics does by establishing a Galois connexion between our domain and the existing domain.

- We generalise stabilisers and propose another new abstract semantics using these extended stabilisers. We also prove the soundness of this semantics and show that the semantics is more concrete than the previous semantics with stabilisers.

The contributions of Chapter 4 are as follows.

- We formulate a new property of BQC protocols named public verifiability.
- We propose a publicly verifiable BQC protocol under the assumption of the existence of an appropriate encryption scheme.
- We show the existence of such an appropriate scheme and thus the existence of a publicly verifiable BQC protocol.

We briefly describe the details.

- In Chapter 3, we propose an abstract semantics of a quantum programming language, the quantum imperative language, using stabilisers. Although abstract interpretation was used to analyse entanglement [89], the existing abstract semantics uses bases of single qubit states and decides whether separable states become entangled through a unitary operator on multiple qubits. Hence, the semantics is naive and fails to exactly analyse entanglement in simple examples. Then, we generalise bases from those of single qubit states to those of multiple qubit states. We decide to use stabilisers as expressions of bases of multiple qubit states and prove that separability of states can be analysed through separability of stabilisers. The result justifies our usage of stabilisers. Using stabilisers, we propose a new abstract domain and new abstract semantics. We define a soundness relation and prove that the semantics is sound. Although it is difficult to exactly compute some part of the semantics, we show an approximation is easy to compute. Furthermore, we estimate the time complexity of computing an approximation of the semantics and conclude that it is efficient for constant-depth programs. Then, we illustrate how the semantics works better. We construct a Galois connexion between our domain and the existing domain, and prove that the existing semantics is a sound abstraction of our semantics. Next, we propose another abstract semantics. The previous semantics has no tolerance to non-Clifford gates. Abstracting non-Pauli matrices, we generalise stabilisers to extended stabilisers. These extended stabilisers can be understood as abstractions of general bases, although stabilisers are bases of stabiliser states. In spite of the existence of abstracted non-Pauli matrices, extended stabilisers tell us separability of the associated bases. Using the fact, we propose a new abstract domain and a new abstract semantics. We also prove that the semantics is sound, its approximation can be efficiently computed, and it is strictly better than the previous semantics. These results show that abstract interpretation is a valuable method for quantum computation.
- In Chapter 4, we propose a publicly verifiable BQC protocol. We illustrate how verifiability is insufficient for some practical problem. In order to solve

the problem, we discuss what kind of property is necessary. We formulate public verifiability as an answer of the question. Then, we suggest a classical public-key encryption scheme helps a verifiable BQC protocol to achieve public verifiability. We concretise the idea and propose a new protocol with an encryption scheme. We impose several conditions to the scheme and show the protocol is a verifiable BQC protocol under the conditions. Because the conditions seem to be too strong, we relax them and show a protocol is a verifiable BQC protocol under the weaker conditions. In order to achieve public verifiability, a security condition is additionally imposed. We propose a protocol and prove that it is a publicly verifiable BQC protocol under the several conditions and an assumption that the client cannot create entanglement. Finally, we construct an encryption scheme satisfying these conditions and thus show a publicly verifiable BQC protocol exists under the assumption. Although classical public-key encryption schemes usually have been described as rivals for quantum computing, our result shows that they enhance quantum computation.

1.3 Related work

1.3.1 Stabiliser abstract semantics

In Chapter 3, we propose two abstract semantics of a quantum programming language, the quantum imperative language (QIL), to analyse quantum entanglement. As mentioned in a previous section, analysis of quantum entanglement in quantum programming languages has been studied in several papers [42, 43, 61, 87, 89, 92].

- Perdrix proposed an abstract semantics of QIL to analyse quantum entanglement [89]. As discussed in Chapter 3, our abstract semantics is based on the semantics. It uses facts that a measured qubit is not entangled with the other qubits, QIL has no unitary transformation creating entanglement other than CX, and $\mathrm{CX}(|\psi\rangle \otimes |\phi\rangle)$ is not entangled when either $|\psi\rangle$ is $|0\rangle$ or $|1\rangle$, or $|\phi\rangle$ is $|+\rangle$ or $|-\rangle$. Its abstract state records information about basis where the state of each qubit belongs, and the abstract semantics uses the information to decide whether qubits are entangled after applying CX. In Chapter 3, we show that our semantics gives finer entanglement analysis than the semantics in [89] does. This is because Perdrix's semantics overlooks facts that CX undoes some entanglement and that measurement may destroy entanglements between unmeasured qubits.

- Prost and Zerrari used Hoare-like logic to analyse entanglement in a higher-order quantum programming language [92]. The analysis is based on a similar idea to [89]. The logic has an assertion $\|u$ that means u is in the standard basis, but it has no assertion about X basis. It concludes that $\mathrm{CX}(|\psi\rangle \otimes |\phi\rangle)$ is not entangled only when $|\psi\rangle$ is in the standard basis. Therefore, we can say that their analysis is coarser than Perdrix's analysis and thus ours.

- Perdrix proposed a type system on a functional language to analyse entanglement [87]. Its typing rules are based on a similar idea to the above ideas. However, it does not use any information about basis. Therefore, it always reasons that two qubits are possibly entangled pair after a unitary transformation on two qubits is applied to them. Therefore, it gives worse analysis than ours does.

- A compilation framework named ScaffCC was proposed in [61]. Entanglement analysis is performed on the framework. Although we analyse entanglement using information about an input state, ScaffCC does not use it. In other words, its analysis is state-independent. It uses the fact that the unitary operator CX is Hermitian. CX creates entanglement, but it is cancelled when CX is applied to the same qubits again. Even if two CX's are not applied successively, entanglement is cancelled when the states of the qubits do not essentially change. In [61], the condition that the state does not essentially change is formulated as the condition that the qubits are not used as target qubits of other operators. We admit an advantage of the method in some cases. A program `CX(`j`,`k`);CX(`j`,`k`)` does not entangle qubits. While our method reasons that the output state may be entangled when the input abstract state contains $(\{\, j \,\}, \blacksquare)$ and $(\{\, k \,\}, \blacksquare)$, which mean that the abstract state have no information about the states of the qubits j, k, the above method always reasons correctly. However, because the method is state-independent, it cannot reason separability depending on the input state. For example, it cannot say that CX $|00\rangle$ is separable. Moreover, the method does not cover any universal set or the Clifford group. We should emphasise that our method can be used to analyse entanglement of any quantum program.

- In Chapter 3, we use the stabiliser formalism to define abstract semantics. The stabiliser formalism was used to analyse entanglement in model checking tools [42, 43]. A big difference between them and ours is that these tools restrict operators to the Clifford operators. This is because they use pure states to trace all possible runs. Moreover, they do not record a partition about entangled qubits but compute the partition using the bipartite normal form [8] of a stabiliser when it is needed. It contrasts with our method. We decompose stabiliser arrays when we can and the principle allows us to use non-Clifford operators.

1.3.2 Publicly verifiable blind quantum computation

- Verifiability is a property that enables Alice to judge whether she obtains the correct outcome and there are several BQC protocols satisfying verifiability [3, 39, 76, 58]. Our new property, public verifiability, is similar to verifiability in the sense that both properties enable parties who have no sufficient quantum devices to judge whether Alice obtains the correct outcome. They are different in the parties who can do verification. Verifiability helps only Alice, the client, but public verifiability guarantees that anyone can correctly guess whether Alice obtains a correct outcome. Especially, public verifiability enables a third party to verify that. There is no study of BQC protocols that cares about a third party. That may be because, in BQC protocols, Alice is always assumed to be honest. As discussed in Chapter 4, public verifiability is useful to solve a conflict between possibly evil Alice and possibly evil Bob. There is no reason to trust Alice, so public verifiability is a meaningful property and our study shows a new direction of the research of BQC protocols.

- We used a classical public-key encryption scheme to achieve public verifiability. In the area of quantum computation and communication, it is not so unusual to use classical cryptography. For instance, in quantum key distribution such as BB84 [11], quantum states are used to generate a shared

secret bit string and then they will be used as a private key of the one-time pad. Many BQC protocols such as [16] use the one-time pad to achieve perfect blindness. However, what is common is to use the one-time pad, a classical *private-key* encryption scheme. It is not common to use a classical *public-key* encryption scheme. As far as we know, this is the first time that classical public-key encryption schemes collaborate with BQC protocols, and perhaps with quantum computation.

- Our public verifiability is a new property of BQC protocols. However, in the area of classical computation, a similar property has already been discussed [38, 49, 85, 86]. Parno et al. [86] defined "public verifiability" as a property of outsourced computation. To avoid confusion, we use the adjective "classical" for the "public verifiability". The outsourced computation enables a computationally weak client to delegate computation to a powerful server. Roughly speaking, classical public verifiability guarantees that anyone can verify that the server returns the correct outcome. Slightly more formally, classical public verifiability was defined as follows. First, the client sends the server a function f and an input x where the client wants to learn $f(x)$. The client also computes a verification key from f and x, and makes the key public. The server outputs $f(x)$ with a "proof". Verifiers including third parties check the proof with the verification key and verify the correctness of the outcome. Classical public verifiability is a property that computationally guarantees that the server cannot deceive the verifiers. We should note that classical public verifiability does not care about the possibility of evil Alice. In classical public verifiability, third parties are on the side of Alice. On the other hand, they behave neutrally in our public verifiability. Papamanthou et al. [85] proposed a similar model named signatures of correct computation, where verifiers can verify the outcome without a verification key issued by the client. However, the model uses a trusted party and verifiers use a key issued by the party instead. Note that our public verifiability and our publicly verifiable protocol do not need such a trusted party.

1.4 Organisation

The rest of this thesis is organised as follows. In the next chapter, we describe basic notions necessary to read the thesis. In Chapter 3, we enter entanglement analysis in quantum programming language. We introduce two abstract semantics. In Chapter 4, we move to BQC protocols. We propose a publicly verifiable BQC protocol. In the last chapter, we summarise our work. Then, we discuss possible future work.

Chapter 2

Preliminaries

2.1 Notation

In this thesis, we use the following notation.

- $\mathbb{N}$ and $\mathbb{C}$ are the sets of natural numbers and complex numbers. $\mathbb{B}$ is the set of bit values $\{\,0, 1\,\}$. $\mathbb{B}^*$ is the set of bit strings. $[<n]$ is the set of natural numbers less than n.
- $\mathcal{P}X$ is the power set of a set X. $X \Rightarrow Y$ is the function space from X to a set Y. For any $f \in X \Rightarrow Y$, $x \in X$, and $y \in Y$, $f[x \mapsto y]$ is the new function $g \in X \Rightarrow Y$ defined by $g(x) = y$ and $g(z) = f(z)$ for any $x \neq z \in X$.
- $\mathbb{Z}_n$ is the ring of integers modulo n. $\mathbb{Z}_n^*$ is the multiplicative group of integers modulo n. $\mathrm{GL}(n, F)$ is the general linear group of $n \times n$ matrices over a field F. $\langle g_0, g_1, \ldots, g_{m-1} \rangle$ is the abelian group generated by $g_0, g_1, \ldots, g_{m-1}$. It is also denoted by $\langle g_j \rangle_{j<m}$. We always assume each g_j is independent.
- $\{a_j\}_{j \in J}$ and $(a_j)_{j \in J}$ denote a family and a sequence, respectively. There is not so big a difference between them. The elements of the latter are ordered, but those of the former are not necessarily so.
- $M_{\{j,k\}}$ denotes the (j, k)th entry of a matrix M. $M_{\{j,\}}$ is the jth row of a matrix M. $s_{\{j\}}$ denotes the jth entry of s. When s is a sequence $(s_k)_k$, $s_{\{j\}}$ is s_j. When s is a tensor product $s = \bigotimes_k s_k$, $s_{\{j\}}$ is s_j.
- $\mathbf{\Pi}^X$ is the set of partitions of a set X. An order $\leq_{\mathbf{\Pi}}$ on $\mathbf{\Pi}^X$ is defined by $P \leq_{\mathbf{\Pi}} P'$ if and only if for any $Q' \in P'$, there exist $Q_0, \ldots, Q_{m-1} \in P$ such that $Q' = \bigcup_j Q_j$. $(\mathbf{\Pi}^X, \leq_{\mathbf{\Pi}})$ is known to be a lattice.
- Let G be a graph. $N_G(v)$ is the set of neighbourhoods of a vertex v.
- $x \leftarrow X$ means the event of taking x from a random variable X. When X is a set, it is the event of sampling x uniformly randomly from a set X.
- I is the identity operator, which usually acts on $\mathbb{C}^2$. $L^\dagger$ and $\mathrm{tr}L$ are the adjoint and the trace of a linear operator L.
- pr_0 is the zeroth projection map. $\mathrm{pr}_0(x, y) = x$ and $\mathrm{pr}_0(X \times Y) = X$.
- For $U_{(j)}$ and $\rho_{[j]}$, see the next section.

When we define a function f as follows, $f(x)$ is y_1 if and only if P_0 does not hold but P_1 holds. In other words, the conditions in the definition are always exclusive.

$$f(x) = \begin{cases} y_0 & (P_0 \text{ holds}) \\ y_1 & (P_1 \text{ holds}) \\ y_o & (\text{otherwise}) \end{cases} \tag{2.1}$$

2.2 Quantum computation

We briefly describe the basics of quantum computation. For the details, see a textbook such as [82].

2.2.1 Rudiments of quantum computation

Quantum state A *quantum system* is a physical entity. It is equipped with a Hilbert space $\mathcal{H}$ called the *state space*. We assume that any state space has the finite dimension and therefore it is isomorphic to $\mathbb{C}^n$ with some n. We identify a quantum system with its state space. A *pure state* of a quantum system is a ray of the state space. As a representative of a ray, we use a unit length vector. A *qubit* is a quantum system whose state space is $\mathbb{C}^2$. We take a basis of the space and write it as $|0\rangle$ and $|1\rangle$. The basis is fixed through the thesis and used as the standard basis of $\mathbb{C}^2$. Generally, a pure state of a qubit is $\alpha|0\rangle + \beta|1\rangle$ where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

Example 2.2.1. Let θ be a real number. The following vector is a state of a qubit.

$$|+_\theta\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{e^{i\theta}}{\sqrt{2}}|1\rangle \tag{2.2}$$

It is easy to see θ and $\theta + 2\pi$ determine the same state. We write $|+\rangle$ and $|-\rangle$ to denote $|+_0\rangle$ and $|+_\pi\rangle$.

The state space of a composite system is defined by the tensor product of the state spaces of the subsystems. Therefore, the state space of n qubits is $\mathbb{C}^{2^{\otimes n}} \simeq \mathbb{C}^{2^n}$. From now on, we focus on multiple qubits. We assume that the dimension of any state space is a power of two. When the pure state of each subsystem is known, the pure state of the composite system is given by the tensor product of them. For example, when pure states of quantum systems $\mathcal{H}_A$ and $\mathcal{H}_B$ are $|+\rangle$ and $|0\rangle$, the whole pure state of the system $\mathcal{H}_A \otimes \mathcal{H}_B$ is $|+\rangle \otimes |0\rangle$.

Notation 2.2.2. We write $|+\rangle_{[A]}$ to emphasise that $|+\rangle$ is a state of $\mathcal{H}_A$.

Notation 2.2.3. We use Dirac's bra-ket notation [30]. A ket notation $|\rangle$ denotes a pure state, or equivalently a linear function from $\mathbb{C}$ to the state space. A bra notation $\langle|$ gives its dual. For a ket $|\psi\rangle$, the dual $\langle\psi|$ is a linear function from the state space to $\mathbb{C}$. Given a ket $|\phi\rangle$, it outputs the inner product of $|\psi\rangle$ and $|\phi\rangle$, which is denoted by $\langle\psi|\phi\rangle$. Concatenation of a ket and a bra represents the composition of them as linear functions. The tensor product of two kets or two bras are denoted by concatenation of their labels. For example, $|+\rangle \otimes |0\rangle$ is written by $|+0\rangle$. It gives a canonical embedding of a bit string s into a quantum state $|s\rangle$.

A *density operator* on a finite dimensional Hilbert space is a positive semidefinite operator on the space whose trace is less than or equal to one. $\mathcal{D}_n$ is the set of density operators on n qubits. The *Löwner order* $\sqsubseteq$ is an order on the set: $\rho \sqsubseteq \sigma$ if and only if $\sigma - \rho$ is positive semidefinite.

Proposition 2.2.4 ([97]). *The partially ordered set $(\mathcal{D}_n, \sqsubseteq)$ is a complete partially ordered set. Furthermore, the poset is not a lattice.*

A *mixed state* of a quantum system is a density operator on the state space. A composite mixed state is also given the tensor product of the mixed states of the subsystems whenever they are known. We use a term *quantum state* to denote a pure or mixed state of a quantum system.

A mixed state can be understood as an ensemble of pure states. For any mixed state ρ, there exists an ensemble of pure states $\{|\psi_j\rangle\}_j$ such that

$$\rho = \sum_j p_j |\psi_j\rangle\langle\psi_j| \tag{2.3}$$

where p_j is the probability of taking $|\psi_j\rangle$ and they satisfy $\sum_j p_j \leq 1$. A mixed state $|\psi\rangle\langle\psi|$ is identified with the pure state $|\psi\rangle$.

Another view of a mixed state is a state of a subsystem. When a state of a composite system $\mathcal{H}_A \otimes \mathcal{H}_B$ is ρ, the state of $\mathcal{H}_A$ is virtually understood as $\mathrm{tr}_B(\rho)$, ignoring the state of $\mathcal{H}_B$. The *partial trace* tr_B is a linear function defined by $\mathrm{tr}_B(\rho \otimes \sigma) = \rho \otimes \mathrm{tr}(\sigma)$. The partial trace of a pure state is not always a pure state. Any mixed state is a partial trace of a pure state of larger system. For any mixed state ρ of $\mathcal{H}_A$, there exist a system $\mathcal{H}_B$ and a pure state $|\psi\rangle$ of $\mathcal{H}_A \otimes \mathcal{H}_B$ such that $\mathrm{tr}_B(|\psi\rangle\langle\psi|) = \rho$. The former view shows a mixed state is a generalisation of a pure state. However, the latter tells us that if we can use any number of qubits, there is no essential difference between a pure state and a mixed state.

Unitary transformation A deterministic transformation of a quantum state is a unitary operator on the state space. When a pure state $|\psi\rangle$ is transformed by a unitary operator U, the state is changed into $U|\psi\rangle$. Similarly, a mixed state ρ is changed into $U\rho U^\dagger$. When a quantum state of a system $\mathcal{H}_A$ is transformed by U, the entire state of a composite system $\mathcal{H}_A \otimes \mathcal{H}_B$ is transformed by $U \otimes \mathrm{I}$ where I is the identity operator acting on $\mathcal{H}_B$.

Notation 2.2.5. We use $U_{(A)}$ to denote the transformation $U \otimes \mathrm{I}$.

Example 2.2.6. The following operators are transformations on single qubit.

$$\begin{aligned} \mathrm{X} &= |0\rangle\langle 1| + |1\rangle\langle 0| \qquad \mathrm{Y} = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \\ \mathrm{H} &= |+\rangle\langle 0| + |-\rangle\langle 1| \qquad \mathrm{Z}(\theta) = |0\rangle\langle 0| + e^{i\theta}|1\rangle\langle 1| \end{aligned} \tag{2.4}$$

Z, S, and T are defined to be $\mathrm{Z}(\pi)$, $\mathrm{Z}(\frac{\pi}{2})$, and $\mathrm{Z}(\frac{\pi}{4})$. They act on single qubit as follows

$$\mathrm{H}|0\rangle = |+\rangle \qquad \mathrm{Z}(\theta)|+_\phi\rangle = |+_{\phi+\theta}\rangle \qquad \mathrm{X}|+_\phi\rangle = |+_{-\phi}\rangle \tag{2.5}$$

The following is transformations on two qubits.

$$\mathrm{CX} = |0\rangle\langle 0| \otimes \mathrm{I} + |1\rangle\langle 1| \otimes \mathrm{X} \qquad \mathrm{CZ} = |0\rangle\langle 0| \otimes \mathrm{I} + |1\rangle\langle 1| \otimes \mathrm{Z} \tag{2.6}$$

Any unitary operator can be written by the above unitary operators, their tensor products, and their compositions. Specifically, a set $\{\mathrm{H}, \mathrm{Z}(\theta), \mathrm{CX}\}$ is sufficient to construct any unitary operator. Such a set is said to be *universal*. $\{\mathrm{H}, \mathrm{T}, \mathrm{CX}\}$ is not universal because it is finite. However, by the Solovay-Kitaev theorem, the set suffices to efficiently construct an approximation of any unitary operator in arbitrary precision. Such a set is *approximately universal*, or simply universal. We use the set later.

A well-known feature of quantum states is that they cannot be cloned. This feature illustrates how quantum states differ from classical states.

Theorem 2.2.7 ([29, 107])**.** *Let $\mathcal{H}$ be a Hilbert space. Assume the dimension is larger than* 1*. There is no copying operator $f\colon \mathcal{H} \to \mathcal{H} \otimes \mathcal{H}$, which satisfies $f(|\psi\rangle) = |\psi\rangle \otimes |\psi\rangle$ for any $|\psi\rangle \in \mathcal{H}$. In particular, when there exists a unitary operator U on $\mathcal{H} \otimes \mathcal{H}$ and $f(|\psi\rangle) = U(|\psi\rangle \otimes |\phi\rangle)$ with some $|\phi\rangle \in \mathcal{H}$, any distinct copyable vectors are orthogonal.*

Quantum measurement Measurement extracts classical information from a quantum state. In the thesis, measurement is a basis. When a quantum system in the state $|\psi\rangle$ is measured by a basis $(\phi_j)_j$ of the quantum system, the measurement result k is obtained by the probability $|\langle\phi_k|\psi\rangle|^2$, and the state is changed into $|\phi_k\rangle$. When a quantum state of a composite system $\mathcal{H}_A \otimes \mathcal{H}_B$ is $|\psi\rangle$ and $\mathcal{H}_A$ is measured by $(\phi_j)_j$, we obtain k as the measurement result in the probability $\| \langle\phi_k|\psi\rangle \|^2$ and the post-measurement state is $\frac{1}{\|\langle\phi_k|\psi\rangle\|}(|\phi_k\rangle\langle\phi_k| \otimes \mathrm{I})|\psi\rangle$. If a pre-measurement state is ρ, the post-measurement state is $\sum_j |\phi_j\rangle\langle\phi_j|_{(A)}\rho|\phi_j\rangle\langle\phi_j|_{(A)}$ where $\mathcal{H}_A$ is the measured quantum system and $(\phi_k)_k$ is the measurement. We call measurement of a quantum system by a basis $(|\phi_k\rangle)_k$ by *measurement in a basis* $(|\phi_k\rangle)_k$. Moreover, *measurement in an angle* θ is measurement in the basis $(|+_\theta\rangle, |+_{\theta+\pi}\rangle)$.

Quantum entanglement Let $|\psi\rangle$ be a pure state of a composite system $\mathcal{H}_A \otimes \mathcal{H}_B$. The state is said to be *separable* if there exist pure states $|\phi\rangle$ and $|\varphi\rangle$ on $\mathcal{H}_A$ and $\mathcal{H}_B$ such that $|\psi\rangle = |\phi\rangle \otimes |\varphi\rangle$. In that case, these quantum systems are said to be separable. More generally, when P is a partition of the subsystems, a state of a composite system is *P-separable* if there exists $\left\{|\phi\rangle_{[Q]}\right\}_{Q\in P}$ such that

$$|\psi\rangle = \bigotimes_{Q\in P} |\phi\rangle_{[Q]}. \tag{2.7}$$

Example 2.2.8. The following pure states are both entangled.

$$|\mathrm{GHZ}\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle \qquad |\mathrm{GHZ}_4\rangle = \frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{\sqrt{2}}|1111\rangle \tag{2.8}$$

A mixed state ρ of the system is *P-separable* if there exists an ensemble of families of mixed states $\left\{\left\{\rho_{j,[Q]}\right\}_{Q\in P}\right\}_j$ such that

$$\rho = \sum_j p_j \bigotimes_{Q\in P} \rho_{j,[Q]} \tag{2.9}$$

where p_j is the probability of taking $\left\{\rho_{j,[Q]}\right\}_{Q\in P}$. A non-separable state is *entangled*. An ensemble of P-separable pure states is certainly a P-separable mixed state. However, the converse is not necessarily true.

Example 2.2.9. The following pure states are both entangled.

$$|\mathrm{Bell}_0\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \quad |\mathrm{Bell}_2\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \tag{2.10}$$

However, the uniform ensemble of them is separable.

$$\frac{1}{2}|\mathrm{Bell}_0\rangle\langle\mathrm{Bell}_0| + \frac{1}{2}|\mathrm{Bell}_2\rangle\langle\mathrm{Bell}_2| = \frac{1}{2}|00\rangle\langle 00| + \frac{1}{2}|11\rangle\langle 11| \tag{2.11}$$

Important features of quantum entanglement are that it is created by unitary transformations on multiple qubits and it is destroyed by quantum measurement. For any separable state $|\psi\rangle = |\phi\rangle \otimes |\varphi\rangle$ of $\mathcal{H}_A \otimes \mathcal{H}_B$ and any unitary transformation U on $\mathcal{H}_A$, $U_{(A)}|\psi\rangle = U|\phi\rangle \otimes |\varphi\rangle$ is definitely separable. Any local unitary transformation cannot create quantum entanglement. However, $|+0\rangle$ is transformed into $|\mathrm{Bell}_0\rangle$ by CX. The former is separable, but the latter is entangled. Contrary to unitary transformation, local quantum measurement suffices to break an entanglement. When $|\psi\rangle$ is a state of a composite system and the jth qubit is measured in the standard basis, the post-measurement state is $\frac{1}{\||0\rangle_{(j)}|\psi\rangle\|}(|0\rangle_{[j]} \otimes (\langle 0|_{(j)}|\psi\rangle))$ or $\frac{1}{\||1\rangle_{(j)}|\psi\rangle\|}(|1\rangle_{[j]} \otimes (\langle 1|_{(j)}|\psi\rangle))$. The measured qubit is always separable from the others.

Quantum entanglement plays a central role in quantum information science. It separates quantum physics from classical physics via the violation of Bell's inequality; it allows a quantum state to teleport; and it forms a computational model to quantum computation, which we will explain in a later subsection.

2.2.2 Stabiliser formalism

A quantum computer is believed to exceed a classical computer. However, a restricted quantum computer is not necessarily so. For example, if it cannot use any unitary operator other than CX, any computation from $|0\rangle$ can be executed using a classical computer because CX does not create quantum superposition. That is a trivial example, but there exists an interesting restriction, which is the stabiliser formalism.

Stabiliser state The unitary operators I, X, Y, and Z are called *Pauli matrices*. Except for I, any Pauli matrix anticommutes with another Pauli matrix. Moreover, $\mathrm{XYZ} = i\mathrm{I}$. Therefore, $\{\pm 1, \pm i\} \times \{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ is a group, which is called the *Pauli group*. It can be generalised to multiple qubits. A Pauli matrix (on n qubits) is a tensor product of n Pauli matrices. They form a group with multiplcative factors ± 1 and $\pm i$, which is the Pauli group (on n qubits). An element of the Pauli group either commutes or anticommutes with another element.

Let V be a subspace of $\mathbb{C}^{2^n}$, the state space of n qubits. A *stabiliser of* V is a subgroup of the Pauli group on n qubits that stabilises any vector in V. More explicitly, it is the set of Pauli matrices P such that for any $|\psi\rangle \in V$, $P|\psi\rangle = |\psi\rangle$. A subgroup is said to be a *stabiliser* if it is a stabiliser of a nontrivial subspace. A subgroup of the Pauli group on n qubits is a stabiliser if and only if it is abelian and it does not contain $-\mathrm{I}^{\otimes n}$. A stabiliser generated by m independent Pauli matrices stabilises a subspace of dimension 2^{n-m}. In particular, $\langle P_0, \ldots, P_{n-1}\rangle$ stabilises a unique state, which is called the *stabiliser state*. While such a stabiliser has 2^n elements, it is completely determined by n generators. A standard description of a quantum state needs 2^n complex numbers, so these generators give us a compact way to express a quantum state.

Notation 2.2.10. We usually omit to write $\otimes$ in a tensor product of Pauli matrices. $\mathrm{I} \otimes \mathrm{X} \otimes \mathrm{Y}$ is denoted by IXY. Although the notation seems to cause confusion between a tensor product and a product, it is always easy to distinguish them from the context.

Example 2.2.11. $|0\rangle$ is the stabiliser state defined by $\langle \mathrm{Z}\rangle$. $|-\rangle$ is the stabiliser state defined by $\langle -\mathrm{X}\rangle$. The stabiliser state of $\langle \mathrm{XXX}, \mathrm{ZZI}, \mathrm{IZZ}\rangle$ is nothing but $|\mathrm{GHZ}\rangle$.

Clifford transformation Suppose a Pauli matrix P stabilises a state $|\psi\rangle$, which means $P|\psi\rangle = |\psi\rangle$. Given a unitary transformation U, $U|\psi\rangle$ is stabilised by $UPU^\dagger$. Indeed, $UPU^\dagger U|\psi\rangle = UP|\psi\rangle = U|\psi\rangle$. Therefore, when the stabiliser state of a stabiliser S is $|\psi\rangle$, $U|\psi\rangle$ is the stabiliser state of $USU^\dagger$ where $S = \langle P_0, \ldots, P_{n-1}\rangle$ and $USU^\dagger = \langle UP_0U^\dagger, \ldots, UP_{n-1}U^\dagger\rangle$. Unfortunately, we cannot use every unitary transformation within the stabiliser formalism because $UPU^\dagger$ is not always a Pauli matrix even when P is so. A subgroup of the group of unitary operators on n qubits is the *Clifford group* if it is the normaliser of the Pauli group on n qubits. A *Clifford operator* is an element of the Clifford group. It is known that the Clifford group is generated by unitary operators H, S, CX. For example, $\mathrm{H}_{(0)}$ changes a stabiliser $\langle \mathrm{ZY}, \mathrm{YX}\rangle$ into $\langle \mathrm{XY}, -\mathrm{YX}\rangle$. The set $\{\mathrm{H}, \mathrm{S}, \mathrm{CX}\}$ is not a universal set, although the set $\{\mathrm{H}, \mathrm{T}, \mathrm{CX}\}$ is universal. The unitary operator T is not a Clifford operator. Indeed, $\mathrm{TXT}^\dagger = \frac{1}{\sqrt{2}}\mathrm{X} + \frac{1}{\sqrt{2}}\mathrm{Y}$, which is not a Pauli matrix.

Measurement in the stabiliser formalism The stabiliser formalism also allows us to measure a quantum system having a stabiliser state. Suppose a pre-measurement state is the stabiliser state defined by $\langle P_0, \ldots, P_{n-1}\rangle$. When the jth qubit is measured in the standard basis, either no $P_{k\{j\}}$ anticommutes with Z or there is $P_{k\{j\}}$ anticommuting with Z. In the former case, the result of the measurement is deterministic. As Z commutes with any $P_{k\{j\}}$, $\mathrm{Z}_{(j)}$ commutes with any generator of the stabiliser. It means that either $\mathrm{Z}_{(j)}$ or $-\mathrm{Z}_{(j)}$ belongs to the stabiliser. If not, $\mathrm{Z}_{(j)}$ is independent from the generators and $\langle P_0, \ldots, P_{n-1}, \mathrm{Z}_{(j)}\rangle$ stabilises a subspace of the dimension 2^{-1}. When $\mathrm{Z}_{(j)}$ or $-\mathrm{Z}_{(j)}$ belongs to the stabiliser, the measurement result is 0 or 1, respectively. This measurement does not change the stabiliser state. On the other hand, the measurement result is probabilistically determined in the case that there is $P_{k\{j\}}$ anticommuting with Z. If both $P_{k\{j\}}$ and $P_{l\{j\}}$ anticommutes with Z, we can replace the lth generator with P_kP_l, which does not change the stabiliser. Therefore, without loss of generality, we can assume there exists a unique k such that $P_{k\{j\}}$ anticommutes with Z. Then, the measurement result b is uniformly randomly chosen from $\mathbb{B}$ and the post-measurement state is stabilised by $\langle P_0, \ldots, P_{k-1}, (-1)^b\mathrm{Z}_{(j)}, P_{k+1}, \ldots, P_{n-1}\rangle$.

Gottesman–Knill theorem A stabiliser compactly expresses a quantum state. A quantum state on n qubits is completely determined by n^2 Pauli matrices and n multiplicative factors. Both Clifford transformations and measurement in the standard basis do not break the expression. Therefore, a classical computer can efficiently simulate quantum computation within the stabiliser formalism. The Gottesman–Knill theorem states the fact.

Theorem 2.2.12 ([52, 82])**.** *A classical computer can efficiently perform any quantum computation starting from a quantum state in the standard basis, applying* H*,* S*, and* CX*, and measuring qubits in the standard basis.*

Since one non-Clifford operator T is sufficient to make the Clifford group universal, the stabiliser formalism may be understood as a maximal classical subset of quantum computation.

At the end of the subsection, we emphasise that a stabiliser state may be an entangled state. Although the existence of quantum entanglement is a unique feature of quantum computation, it does not suffice to exceed classical computation.

2.2.3 Measurement-based quantum computation

There are many computational models of quantum computation such as quantum circuits and quantum Turing machine. Measurement-based quantum computation (MBQC) [93, 94] is one of them. This model has no corresponding classical computational model because it fully uses quantum entanglement and quantum measurement, which are features of quantum computation. In the model, an entangled state is first created and then destroyed by quantum measurement.

Let G be an undirected simple graph (V, E). The *graph state* $|G\rangle$ is a stabiliser state of $|V|$ qubits. Each qubit is labelled a different vertex of the graph. The state is stabilised by the stabiliser $\left\langle \mathrm{X}_{[v]} \otimes \bigotimes_{(v,u)\in E} \mathrm{Z}_{[u]} \right\rangle_{v\in V}$. The stabiliser formalism tells us how to construct the state. First, the state of any qubit is set $|+\rangle$. This is the stabiliser state of $\langle \mathrm{X} \rangle$ and therefore the whole state is stabilised by $\left\langle \mathrm{X}_{[v]} \right\rangle_{v\in V}$. Then, for any edge, a unitary operator CZ is applied to the adjacent qubits. Since $\mathrm{CZ}(\mathrm{XI})\mathrm{CZ}^\dagger = \mathrm{XZ}$ and $\mathrm{CZ}(\mathrm{ZI})\mathrm{CZ}^\dagger = \mathrm{ZI}$, it changes the stabiliser of the entire state into $\left\langle \mathrm{X}_{[v]} \otimes \bigotimes_{(v,u)\in E} \mathrm{Z}_{[u]} \right\rangle_{v\in V}$.

More generally, an *open graph state* is defined. An *open graph* consists of an undirected simple graph (V, E) with two sets of vertices I, O having the same size. These sets are called *input vertices* and *output vertices*. An open graph state is created in the same manner as a graph state except that the state of any input vertex is not set $|+\rangle$. A *flow* [17, 22] on an open graph (G, I, O) is a pair of a function $f\colon (V \setminus O) \to (V \setminus I)$ and an order $\leq$ on V that satisfies the following three conditions for any non-output vertex v:

1. v is adjacent to the vertex $f(v)$,
2. $v \leq f(v)$, and
3. for any vertex u adjacent to $f(v)$, $v \leq u$.

For any open graph, a flow is unique if it exists [25].

Measurement-based quantum computation is specified by an open graph with a flow and an angle for each non-output vertex, and it implements a unitary transformation from the input qubits to the output qubits. Here, input and output qubits mean qubits labelled input and output vertices, respectively. Each angle determines measurement of each qubit, and hence the set of them determines the unitary transformation. The computation is performed as follows. Suppose there are qubits having possibly unknown states. Using the qubits as the input qubits, the open graph is first created with auxiliary qubits whose states are all $|+\rangle$. Then, non-output qubits are measured. An order of the measurement respects the order in the flow: if $v \leq u$, v is measured earlier than u is. Each non-output qubit v is measured in the associated angle with a modification. Let ϕ be the associated angle. The actual angle ϕ' is determined by the measurement results of the previously measured qubits:

$$\phi' = (-1)^{\sum_{u\in f^{-1}(v)} b_u} \phi + \pi \sum_{v\in N_G(f(u))} b_u \tag{2.12}$$

where b_u is the measurement result of u. Note that the conditions of a flow guarantees that all qubits where ϕ' depends are previously measured, and hence ϕ' is well-defined. After measuring all non-output qubits, only the output qubits remain unmeasured. Finally, the state of each output qubit v is transformed by $\mathrm{X}^{\sum_{u\in f^{-1}(v)} b_u} \mathrm{Z}^{\sum_{v\in N_G(f(u))} b_u}$. The state of the output qubits are the output state of the computation.

Example 2.2.13 ([94])**.** Let us take the straight line graph of five vertices. Explicitly, $V = [<5]$ and $E = \{\, (n, n+1) \mid n \in [<4] \,\}$. The vertex 0 and 4 are the input and output vertices. The order is the usual order and therefore the function of the flow assigns the next vertex $n+1$ to a vertex n. Suppose each vertex n corresponds to an angle θ_n where all $\theta_n = 0$ except for $\theta_2 = \frac{\pi}{4}$. Then, the output state is $\mathrm{T}\,|\psi\rangle$ where $|\psi\rangle$ is the state of the input qubit. Generally, if $\theta_0 = 0$, the output state is $\mathrm{HZ}(\theta_3)\mathrm{HZ}(\theta_2)\mathrm{HZ}(\theta_1)\mathrm{H}\,|\psi\rangle$. Any unitary operator on single qubit can be decomposed into such operators.

The above example shows any single qubit unitary operator is implemented on a sufficiently long straight line graph. In general, not all open graphs allow us to implement any unitary operator. However, there are *universal* open graph states such as brickwork states [16], where the term universal means that any unitary operator can be efficiently implemented on the state.

2.3 Abstract interpretation

Abstract interpretation [19, 20, 21] is a method for sound approximations. It answers a question about a program without running it. In other words, it helps us to statistically analyse a property of a program. The analysis is sound but not generally complete. It can be understood through a simple example.

Example 2.3.1 ([20, 101])**.** Suppose we want to know the sign of the following.

$$-13490673 \times (5718927 + 5317859) - 8123756 \tag{2.13}$$

One way is computing the result value. However, we can deduce that it is negative by abstracting each number to its sign.

$$(-) \times ((+) + (+)) - (+) \tag{2.14}$$

The abstraction fails when we replace 5317859 with -68794123. The sign of $(+) + (-)$ is not determined. Then, we prepare $(\pm)$ for upper approximation of $(+)$ and $(-)$. With the symbol, we can perform sound analysis of signs of numbers.

Abstract interpretation describes a relationship between two domains: a *concrete* (*semantic*) *domain* and an *abstract* (*semantic*) *domain*. On each domain, a semantics of programs is defined: a *concrete semantics* and an *abstract semantics*. In the above example, $\mathbb{Z}$ and $\{\, (+), (-), (\pm) \,\}$ are a concrete and an abstract domains and semantics such as $5718927 + 5317859 = 11036786$ and $(+) + (+) = (+)$ are concrete and abstract semantics. A relationship between two domains is formulated as a soundness relation.

Definition 2.3.2. Let L be a concrete domain and M be an abstract domain. A *soundness relation* is a subset of $L \times M$. Let $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_M$ be a concrete and an abstract semantics. $[\![\cdot]\!]_M$ is a *sound abstraction* of $[\![\cdot]\!]_L$ if there exists a sound relation σ such that for any program C, $(x, y) \in \sigma$ implies $([\![C]\!]_L(x), [\![C]\!]_M(y)) \in \sigma$.

Concrete and abstract domains may form a Galois connexion. In this case, a soundness relation is naturally induced.

Definition 2.3.3. Let $(L, \leq_L)$ and $(M, \leq_M)$ be posets. A pair of functions $\alpha\colon L \to M$ and $\gamma\colon M \to L$ is a *Galois connexion* if for any $x \in L$ and $y \in L$, $\alpha(x) \leq_M y$ if and only if $x \leq_L \gamma(y)$.

Proposition 2.3.4. *α and γ are monotone.*

Proposition 2.3.5. *The definition of a Galois connexion is equivalent to the following definition: for any $x \in L$ and $y \in L$, $x \leq_L \gamma(\alpha(x))$ and $\alpha(\gamma(y)) \leq_M y$.*

Let L be a concrete domain and M be an abstract domain. Their orders show which one has more information: smaller one has more information. When $\alpha\colon L \to M$ and $\gamma\colon M \to L$ form a Galois connexion, α and γ are called an *abstraction function* and a *concretisation function*. The inequality $x \leq_L \gamma(\alpha(x))$ shows the abstraction loses some information.

Proposition 2.3.6. *Let $\alpha\colon L \to M$ and $\gamma\colon M \to L$ and assume they form a Galois connexion. Moreover, assume $[\![\cdot]\!]_M$ is monotone. Let σ be the set $\{\,(x, y) \mid \alpha(x) \leq_M y\,\}$. Assume for any program C and $x \in L$, $\alpha([\![C]\!]_L(x)) \leq_M [\![C]\!]_M(\alpha(x))$. Then, $[\![\cdot]\!]_M$ is a sound abstraction of $[\![\cdot]\!]_L$.*

2.4 Encryption scheme

An encryption scheme describes how to protect messages from non-authorised parties. In the thesis, we use public-key encryption schemes. These schemes are specified by determining how to generate keys; how to encrypt messages; and how to decrypt encrypted messages.

Definition 2.4.1. A (*public-key*) *encryption scheme* $\mathcal{E}$ is a triplet (G, E, D) consisting of

- a probabilistic algorithm G that computes a pair of an *encryption key* $e \in \mathbb{B}^*$ and a *decryption key* $d \in \mathbb{B}^*$ from 1^n where n is a natural number and it is called the *security parameter*,
- a probabilistic algorithm E that, given bit strings e and m, computes a *ciphertext* $\alpha \in \mathbb{B}^*$ *of* m, or outputs an error, and
- a deterministic algorithm D that, given bit strings e, d, and α, computes the *plaintext* $m \in \mathbb{B}^*$ *of* α, or outputs an error.

We call G, E, and D the *key-generator*, the *encryption*, and the *decryption* of the encryption scheme, respectively. We define, for any $n \in \mathbb{N}$, the *key space* $\mathbf{KS}_n$ as the support of the distribution of $G(1^n)$, the *encryption key space* $\mathbf{EKS}_n$ as $\mathrm{pr}_0\mathbf{KS}_n$, and the *decryption key space* $\mathbf{DKS}_n$ as $\mathrm{pr}_1\mathbf{KS}_n$. Each encryption key $e \in \mathbf{EKS}_n$ is assumed to have an associated set of bit strings $\mathbf{PS}_{n,e}$, which is called the *plaintext space*. The *ciphertext space* $\mathbf{CS}_{n,e,m}$ is the support of the distribution of $E(e, m)$ where $m \in \mathbf{PS}_{n,e}$. $\mathbf{CS}_{n,e}$ is the union of these spaces $\bigcup_{m \in \mathbf{PS}_{n,e}} \mathbf{CS}_{n,e,m}$. We assume the following holds for any $n \in \mathbb{N}$.

$$\Pr\left[m' = m \;\middle|\; \begin{array}{l}(e, d) \leftarrow G(1^n), m \leftarrow \mathbf{PS}_{n,e} \\ \alpha \leftarrow E(e, m), m' \leftarrow D(e, d, \alpha)\end{array}\right] = 1 \tag{2.15}$$

Therefore, without any confusion, we can call an element of any $\mathbf{PS}_{n,e}$ by a *plaintext*. We assume that all elements of any plaintext space have the same lengths. We say an encryption scheme is *classical* if the decryption is a polynomial-time algorithm, the key-generator and the encryption are probabilistic polynomial-time algorithms, and the lengths of plaintexts in $\mathbf{PS}_{n,e}$ are polynomially-bounded with respect to n.

The equation (2.15) claims that we always obtain the original plaintext when we generate a pair of keys, encrypt a plaintext by the encryption key, and decrypt the ciphertext by the decryption key. However, the equation does not tell us what happens when we encrypt a bit string not in the plaintext space; when we encrypt a plaintext by a bit string not in the encryption key space; when we decrypt a ciphertext by a decryption key not generated with the encryption key, and so on. A bit string in an appropriate set is said to be *valid*. For example, a bit string in $\mathbf{EKS}_n$ is a valid encryption key, and a bit string is an *invalid* encryption key when it does not belong to $\mathbf{EKS}_n$ but is used to encryption a plaintext.

Example 2.4.2. The trivial encryption scheme is a classical public-key encryption scheme such that the key-generator is $G(1^n) = (1^n, 1^n)$, the encryption is $E_{\mathbf{T}}(e, m) = m$, and the decryption is $D_{\mathbf{T}}(e, d, \alpha) = \alpha$.

Example 2.4.3. The ElGamal encryption scheme **ElGamal** [33] is a classical public-key encryption scheme $(G_{\mathbf{EG}}, E_{\mathbf{EG}}, D_{\mathbf{EG}})$ defined as follows. A safe prime is a prime $2p + 1$ such that p is also a prime. Given 1^n, the key-generator $G_{\mathbf{EG}}$ generates a safe prime $2p+1$ and a generator g of the subgroup G_p of order p of the cyclic group $\mathbb{Z}^*_{2p+1}$ where the bit length of p is n, chooses uniformly randomly x from $\mathbb{Z}_p$, and outputs $((p, g, g^x), x)$. The plaintext space $\mathbf{PS}_{n,(p,g,g^x)}$ is the group G_p. For each plaintext $g_m \in G_p$, $E_{\mathbf{EG}}((p, g, g^x), g_m)$ is $(g_m g^{xr}, g^r)$ where r is randomly chosen from $\mathbb{Z}_p$. The decryption $D_{\mathbf{EG}}$, given $((p, g, g^x), x, (g_c, g_r))$, outputs $g_c(g_r^x)^{-1}$. It is easy to check that the scheme satisfies (2.15): $g_m g^{xr}(g_r^x)^{-1} = g_m$.

Example 2.4.4. The Goldwasser-Micali encryption scheme **GM** [50] is a classical public-key encryption scheme $(G_{\mathbf{GM}}, E_{\mathbf{GM}}, D_{\mathbf{GM}})$ defined as follows. The plaintext space $\mathbf{PS}$ is $\mathbb{B}$, which is independent of the security parameter. Let $n = \prod_j p_j$ be the product of primes $\{p_j\}$. $x \in \mathbb{Z}^*_n$ is said to be a quadratic residue mod n if there exists $y \in \mathbb{Z}^*_n$ such that $x = y^2 \pmod n$. If no such element exists, x is a quadratic non-residue mod n. The Jacobi symbol $\left(\frac{x}{n}\right)$ is $\prod_j \left(\frac{x}{p_j}\right)$ where $\left(\frac{x}{p_j}\right)$ is 0 if $x = 0 \pmod{p_j}$, -1 if it is a quadratic non-residue mod p_j, and otherwise 1. Given 1^n, the key-generator $G_{\mathbf{EG}}$ generates two primes p, q, randomly chooses a quadratic non-residue x mod $n = pq$ such that the Jacobi symbol $\left(\frac{x}{n}\right) = 1$, and outputs $((n, x), (p, q))$. For each bit value $b \in \mathbb{B}$, $E_{\mathbf{GM}}((n, x), b)$ is $x^b r^2 \pmod n$ where r is a randomly chosen element of $\mathbb{Z}^*_n$. Given $((n, x), (p, q), \alpha)$, $D_{\mathbf{GM}}$ computes whether α is a quadratic residue mod n from the knowledge (p, q).

When a public-key encryption scheme is used, an encryption key is public, although the decryption key is kept secret. Therefore, anyone can create a ciphertext. Suppose Alice receives a ciphertext. She sends either the ciphertext or a ciphertext of her own plaintext. Even if an enemy does not have the decryption key, s/he is able to guess what she does by observing her and comparing the sent ciphertext with the received ciphertext. If she can reencrypt the plaintext of the received ciphertext, that is impossible. The rerandomisation forbids anyone to trace flows of ciphertexts.

Definition 2.4.5. An encryption scheme $\mathcal{E} = (G, E, D)$ is (*perfectly*) *rerandomisable* if it has a probabilistic algorithm R that computes a bit string from two bit strings. The algorithm R is assumed to satisfy the following.

- For any $n \in \mathbb{N}$, any encryption key $e \in \mathbf{EKS}_n$, any plaintext $m \in \mathbf{PS}_{n,e}$, and any ciphertext $\alpha \in \mathbf{CS}_{n,e,m}$, $R(e, \alpha)$ is uniformly distributed on $\mathbf{CS}_{n,e,m}$.

We call R the *rerandomisation*. When $\mathcal{E}$ is classical, we always assume R is a probabilistic polynomial-time algorithm.

Example 2.4.6. The trivial encryption scheme is trivially rerandomisable. Both the ElGamal and Goldwasser-Micali encryption schemes are rerandomisable. Given (p, g, g^x) and (g_c, g_r), one can compute $(g_c g^{xs}, g_r g^s)$ with a randomly chosen $s \in \mathbb{Z}_p$. When $(g_c, g_r) = (g_m g^{xr}, g^r)$, $(g_c g^{xs}, g_r g^s)$ is $(g_m g^{x(r+s)}, g^{r+s})$, which is a ciphertext of g_m. For **GM**, $x^b r^2$ is rerandomised to $x^b r^2 s^2 = x^b (rs)^2$ with a uniformly randomly chosen $s \in \mathbb{Z}_n^*$. This is uniformly random because $\mathbb{Z}_n^*$ is a multiplicative group.

The rerandomisation changes ciphertexts but does not change their plaintexts. Another operation on ciphertexts changes plaintexts. A homomorphic encryption scheme allows anyone to perform computation on encrypted data.

Definition 2.4.7. Let $\mathcal{E} = (G, E, D)$ be an encryption scheme and F be a set. Any element f of the set F is assumed to be equipped with its arity $l \in \mathbb{N}$ and functions $f_e \colon \mathbf{PS}_{n,e}^l \to \mathbf{PS}_{n,e}$ for any $n \in \mathbb{N}$ and $e \in \mathbf{EKS}_n$. An encryption scheme is *homomorphic with respect to* F if it has a family of probabilistic algorithms C_f each of whom computes β from $l+1$ bit strings $e, \alpha_0, \ldots, \alpha_{l-1}$ where l is the arity of f and satisfies the following for any $n \in \mathbb{N}$.

$$\Pr\left[m = m' \;\middle|\; \begin{array}{l} (e,d) \leftarrow G(1^n), m_0, \ldots, m_{l-1} \leftarrow \mathbf{PS}_{n,e} \\ \alpha_0 \leftarrow E(e, m_0), \ldots, \alpha_{l-1} \leftarrow E(e, m_{l-1}) \\ \beta \leftarrow C_f(e, \alpha_0, \ldots, \alpha_{l-1}) \\ m \leftarrow D(e, d, \beta), m' \leftarrow f_e(m_0, \ldots, m_{l-1}) \end{array} \right] = 1 \tag{2.16}$$

When $\mathcal{E}$ is classical, we always assume all C_f are probabilistic polynomial-time algorithms. Moreover, when $\mathcal{E}$ is rerandomisable, we assume that for any $n \in \mathbb{N}$, any $e \in \mathbf{EKS}_n$, and any $m_0, \ldots, m_{l-1} \in \mathbf{PS}_{n,e}$, $C_f(e, E(e, m_0), \ldots, E(e, m_{l-1}))$ is the uniform distribution on the set $\mathbf{CS}_{n,e,f_e(m_0,\ldots,m_{l-1})}$.

Example 2.4.8. The trivial encryption scheme is trivially homomorphic with respect to any set. The ElGamal encryption scheme is known to be multiplicatively homomorphic. Given an encryption key (p, g, g^x), ciphertexts $(g_m g^{xr}, g^r)$ and $(g_l g^{xs}, g^s)$, anyone can obtain a new ciphertext $(g_m g_l g^{x(r+s)}, g^{r+s})$ by multiplying the ciphertexts. The plaintext is $g_m g_l$, which is the product of two plaintexts g_m and g_l. The Goldwasser-Micali encryption scheme is homomorphic with respect to the bit addition. Multiplying two ciphertexts $x^b r^2$ and $x^c s^2$, $x^{b+c}(rs)^2$ is obtained. The plaintext of the ciphertext is a bit value $b + c$.

Finally, we give the definitions of security of encryption schemes. The first definition states that a ciphertext gives anyone no meaningful information, even if one has partial information about the plaintext and one does not need to learn the entire plaintext. On the other hand, the second states that anyone cannot guess distinguish ciphertexts of a known plaintext from ciphertexts of another known plaintext. These two definitions are known to be equal.

Definition 2.4.9. Let $\mathcal{E} = (G, E, D)$ be a classical encryption scheme such that for any $n \in \mathbb{N}$, there exists $\mathbf{PS}_n$ such that $\mathbf{PS}_{n,e} = \mathbf{PS}_n$ for any $e \in \mathbf{EKS}_n$. $\mathcal{E}$ is *semantically secure* if for any probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm B such that for any family $\{X_n\}_{n \in \mathbb{N}}$ of random variables X_n such that $\mathbf{PS}_n$ includes the support of the

distribution of X_n, and any polynomially-bounded function $f, h\colon \mathbf{PS}_n \to \mathbb{B}^*$,

$$\left| \Pr\left[v = w \;\middle|\; \begin{array}{l} m \leftarrow X_n \\ (e,d) \leftarrow G(1^n) \\ \alpha \leftarrow E(e,m) \\ a \leftarrow h(1^n, m) \\ v \leftarrow A(1^n, e, \alpha, l_n, a) \\ w \leftarrow f(1^n, m) \end{array} \right] - \Pr\left[v = w \;\middle|\; \begin{array}{l} m \leftarrow X_n \\ a \leftarrow h(1^n, m) \\ v \leftarrow B(1^n, l_n, a) \\ w \leftarrow f(1^n, m) \end{array} \right] \right| \tag{2.17}$$

is negligible with respect to n where l_n is the length of plaintexts in $\mathbf{PS}_n$.

Definition 2.4.10. Let $\mathcal{E} = (G, E, D)$ be a classical encryption scheme such that the plaintext space is independent of encryption keys, that is $\mathbf{PS}_{n,e} = \mathbf{PS}_n$ for any $e \in \mathbf{EKS}_n$. $\mathcal{E}$ has *indistinguishable encryptions* if for any probabilistic polynomial-time algorithm A, any $\{x_n\}_{n\in\mathbb{N}}, \{y_n\}_{n\in\mathbb{N}}$ such that $x_n, y_n \in \mathbf{PS}_n$, and any family of polynomially-bounded bit strings $\{z_n\}_{n\in\mathbb{N}}$,

$$\left| \Pr\left[v = 1 \;\middle|\; \begin{array}{l} (e,d) \leftarrow G(1^n) \\ c \leftarrow E(e, x_n) \\ v \leftarrow A(e, c, z_n) \end{array} \right] - \Pr\left[v = 1 \;\middle|\; \begin{array}{l} (e,d) \leftarrow G(1^n) \\ c \leftarrow E(e, y_n) \\ v \leftarrow A(e, c, z_n) \end{array} \right] \right| \tag{2.18}$$

is negligible with respect to n.

Theorem 2.4.11 ([48])**.** *Let $\mathcal{E} = (G, E, D)$ be a classical encryption scheme such that the plaintext space is independent of encryption keys. $\mathcal{E}$ is semantically secure if and only if it has indistinguishable encryptions.*

The following security is stronger than the above. In the above, the plaintexts are chosen independently of the encryption key. On the other hand, an attacker can choose plaintexts after learning the encryption key in the following security model.

Definition 2.4.12. A classical encryption scheme (G, E, D) has *indistinguishable encryptions under chosen plaintext attacks* if for any probabilistic polynomial-time Turing machine A, B and any family of polynomially-bounded bit strings $\{z_n\}_{n\in\mathbb{N}}$,

$$\begin{aligned} &\left| \Pr\left[v = 1 \;\middle|\; \begin{array}{l} (e,d) \leftarrow G(1^n), ((x,y),\sigma) \leftarrow A(e, z_n) \\ c \leftarrow E(e,x), v \leftarrow B(\sigma, c) \end{array} \right] \right. \\ &\left. - \Pr\left[v = 1 \;\middle|\; \begin{array}{l} (e,d) \leftarrow G(1^n), ((x,y),\sigma) \leftarrow A(e, z_n) \\ c \leftarrow E(e,y), v \leftarrow B(\sigma, c) \end{array} \right] \right| \end{aligned} \tag{2.19}$$

is negligible where $x, y \in \mathbf{PS}_{n,e}$. A classical encryption scheme is *IND-CPA secure* if it has indistinguishable encryptions under chosen plaintext attacks.

Proposition 2.4.13 ([48])**.** *Let $\mathcal{E} = (G, E, D)$ be a classical encryption scheme such that the plaintext space is independent of encryption keys. If $\mathcal{E}$ has indistinguishable encryptions under chosen plaintext attacks, it has indistinguishable encryptions.*

Example 2.4.14. The trivial encryption scheme is not semantically secure. Both the ElGamal and Goldwasser-Micali encryption schemes are IND-CPA secure under the decisional Diffie-Hellman (DDH) assumption and the quadratic residuosity assumption [50, 103].

Chapter 3

Stabiliser Abstract Semantics

Abstract interpretation is a novel way to analyse programs. In [89], it was introduced to quantum programming languages for entanglement analysis. An abstract semantics was defined and showed to be a sound approximation of a concrete semantics. However, it does not work well in some simple examples. In the chapter, we propose two abstract domains and two abstract semantics with the stabiliser formalism. Both semantics are proved to be sound approximations. Interestingly, the existing abstract domains and these two domains are connected via Galois connexions. We can understand the existing and two abstract semantics are in the relationship of approximations.

The chapter is organised as follows. In Section 3.1, we describe the existing abstract semantics. In the next section (Section 3.2), we propose our first abstract semantics using the stabiliser formalism. In Section 3.3, we propose the second abstract semantics by abstracting non-Pauli matrices. In each section, we show the soundness of the semantics and compare another semantics.

This chapter is based on our paper [59].

3.1 Basis abstract semantics

3.1.1 Quantum imperative language

First of all, we explain a programming language we analyse through the whole chapter. The language is called the quantum imperative language [88, 89].

Definition 3.1.1. Let $N \in \mathbb{N}$. The set $\mathbf{Q}_N$ of *variables of the quantum imperative language* is a set $\{ \mathtt{q}_0, \ldots, \mathtt{q}_{N-1} \}$. The *quantum imperative language* (*QIL*) is the set $\mathbf{QIL}_N$ of (*QIL*) *programs* (ranged over by C, C') defined by the following BNF notation.

$$\begin{aligned} C, C' ::= {} & \mathtt{skip} \mid C\,\mathtt{;}\,C' \mid \mathtt{X(}j\mathtt{)} \mid \mathtt{Y(}j\mathtt{)} \mid \mathtt{Z(}j\mathtt{)} \mid \mathtt{H(}j\mathtt{)} \mid \mathtt{S(}j\mathtt{)} \mid \mathtt{T(}j\mathtt{)} \mid \mathtt{CX(}j\mathtt{,}k\mathtt{)} \\ & \mid \mathtt{if}\ j\ \mathtt{then}\ C\ \mathtt{else}\ C'\ \mathtt{fi} \mid \mathtt{while}\ j\ \mathtt{do}\ C\ \mathtt{od} \end{aligned} \tag{3.1}$$

where $j, k \in \mathbf{Q}_N$ and $j \neq k$.

Notation 3.1.2. We usually identify $\mathbf{Q}_N$ with $[<N]$. During the chapter, we use N to denote the number of variables and we omit to write the subscripts of $\mathbf{Q}_N$ and $\mathbf{QIL}_N$.

The quantum imperative language is a quantum programming language. The intuitive meaning of the language is as follows. There are N qubits and each variable denotes a qubit: $\mathtt{q}_j$ denotes the jth qubit. We identify a variable with the associated qubit. QIL programs perform operations on the qubits. `skip` does noting. $C\,\mathtt{;}\,C'$ is the sequential composition of C and C'. C' runs after

C finish being executed. The remaining programs in the first line of (3.1) denote unitary operations. For example, `CX(`j`,`k`)` is application of CX to the jth and kth qubits. The other two programs involve quantum measurement. In `if` j `then` C `else` C' `fi`, the jth qubit is measured in the standard basis, and then either C or C' runs depending on the measurement result. When it is zero, C is executed. The program `while` j `do` C `od` repeats measurement of jth qubit in the standard basis and executions of C, until the measurement result is one. If the first measurement result is one, `while` j `do` C `od` does not execute C.

Each QIL program C has two natural numbers $\mathrm{sz}(C), \mathrm{dp}(C)$ as its size and depth. Roughly speaking, the size and depth of a QIL program are the numbers of constructors and nested `while` loops in the program, respectively. Note that we do not count `if` to calculate the depth of a program.

Definition 3.1.3. Functions $\mathrm{sz}, \mathrm{dp}\colon \mathbf{QIL} \to \mathbb{N}$ are inductively defined as follows.

$$\begin{aligned} \mathrm{sz}(\texttt{skip}) = \mathrm{sz}(U\texttt{(}j\texttt{)}) &= \mathrm{sz}(\texttt{CX(}j\texttt{,}k\texttt{)}) = 1 \\ \mathrm{sz}\left(C\texttt{;}C'\right) &= \mathrm{sz}(C) + \mathrm{sz}\left(C'\right) \\ \mathrm{sz}\left(\texttt{if } j \texttt{ then } C \texttt{ else } C' \texttt{ fi}\right) &= \mathrm{sz}(C) + \mathrm{sz}\left(C'\right) + 1 \\ \mathrm{sz}(\texttt{while } j \texttt{ do } C \texttt{ od}) &= \mathrm{sz}(C) + 1 \end{aligned} \tag{3.2}$$

$$\begin{aligned} \mathrm{dp}(\texttt{skip}) = \mathrm{dp}(U\texttt{(}j\texttt{)}) &= \mathrm{dp}(\texttt{CX(}j\texttt{,}k\texttt{)}) = 0 \\ \mathrm{dp}\left(\texttt{if } j \texttt{ then } C \texttt{ else } C' \texttt{ fi}\right) &= \mathrm{dp}\left(C\texttt{;}C'\right) = \max\left\{\mathrm{dp}(C), \mathrm{dp}\left(C'\right)\right\} \\ \mathrm{dp}(\texttt{while } j \texttt{ do } C \texttt{ od}) &= \mathrm{dp}(C) + 1 \end{aligned} \tag{3.3}$$

where $U \in \{\texttt{X}, \texttt{Y}, \texttt{Z}, \texttt{H}, \texttt{S}, \texttt{T}\}$. $\mathrm{sz}(C)$ and $\mathrm{dp}(C)$ are the *size* and *depth* of a program C.

By definition, we immediately obtain a relation between these two numbers: $\mathrm{dp}(C) < \mathrm{sz}(C)$. $\mathrm{dp}(C)$ is equal to $\mathrm{sz}(C) - 1$ when C contains neither `;` nor `if`.

The quantum imperative language has no explicit quantum measurement and initialisation of variables. However, the classical structures of the language such as `if` suffice to implement them.

$$\texttt{MEASURE}_j \equiv \texttt{if } j \texttt{ then skip else skip fi} \tag{3.4}$$

$$\texttt{INIT}_j \equiv \texttt{if } j \texttt{ then skip else X(}j\texttt{) fi} \tag{3.5}$$

$$\texttt{INIT} \equiv \texttt{INIT}_0\texttt{;INIT}_1\texttt{;}\cdots\texttt{;INIT}_{N-1} \tag{3.6}$$

The program $\texttt{MEASURE}_j$ just measures the jth qubit in the standard basis. The next program sets the state of the jth qubit $|0\rangle$. Therefore, the last program initialises the states of all qubits.

Another example is quantum teleportation [12]. It needs conditional application of unitary operators, which can be implemented using `if`.

```
teleportation ≡ CX(0,1);
                H(0);
                if 0
                  then if 1 then skip else X(2) fi
                  else if 1 then Z(2) else X(2);Z(2) fi
                fi
```
(3.7)

The intuitive meaning of QIL programs is justified by a concrete semantics of the language. The semantics is defined as a function on the set of density matrices.

$$
\begin{aligned}
[\![\texttt{skip}]\!](\rho) &= \rho \\
[\![C;C']\!](\rho) &= [\![C']\!]([\![C]\!](\rho)) \\
[\![U\,(j)]\!](\rho) &= U_{(j)}\rho U_{(j)}^{\dagger} \\
[\![\texttt{CX(}j\texttt{,}k\texttt{)}]\!](\rho) &= \mathrm{CX}_{(j,k)}\rho\mathrm{CX}_{(j,k)}^{\dagger} \\
[\![\texttt{if } j \texttt{ then } C \texttt{ else } C' \texttt{ fi}]\!](\rho) &= [\![C]\!](|0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)}) + [\![C']\!](|1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)}) \\
[\![\texttt{while } j \texttt{ do } C \texttt{ od}]\!](\rho) &= \sum_{n\in\mathbb{N}} |1\rangle\langle 1|_{(j)}\mathrm{meas}_{C,j}^{n}(\rho)|1\rangle\langle 1|_{(j)}
\end{aligned}
$$

where $j, k \in \mathbf{Q}$, $U \in \{\, \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{H}, \mathtt{S}, \mathtt{T} \,\}$ and $\mathrm{meas}_{C,j}(\rho) = [\![C]\!](|0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)})$.

Figure 3.1: Concrete semantics of quantum imperative language

Definition 3.1.4. The *concrete semantics* of QIL is the function $[\![\cdot]\!] : \mathbf{QIL} \to \mathcal{D}_N \Rightarrow \mathcal{D}_N$ defined in Figure 3.1.

The infinite sum $\sum_{n\in\mathbb{N}} |1\rangle\langle 1|_{(j)}\mathrm{meas}_{C,j}^{n}(\rho)|1\rangle\langle 1|_{(j)}$ converges because the set of density matrices $\mathcal{D}_N$ is a complete partially ordered set. It is worth noting that the trace of $[\![\texttt{while } j \texttt{ do } C \texttt{ od}]\!](\rho)$ may be less than that of ρ. It reflects the fact that `while` may cause an infinite loop. This is the reason why we use density matrices whose traces are less than one.

Notation 3.1.5. Let $\rho, \rho', \rho'', \rho''' \in \mathcal{D}_N$, $j \in \mathbf{Q}$, and $C, C' \in \mathbf{QIL}$. We write $\rho \xrightarrow{[\![C]\!]} \rho'$ when $[\![C]\!](\rho) = \rho'$. When no confusion arises, we write $\rho \xrightarrow{C} \rho'$ to denote $\rho \xrightarrow{[\![C]\!]} \rho'$. $\rho \xrightarrow{C} \rho' \xrightarrow{C'} \rho''$ denotes $[\![C]\!](\rho) = \rho'$ and $[\![C']\!](\rho') = \rho''$, so $[\![C;C']\!](\rho) = \rho''$. For $\texttt{if } j \texttt{ then } C \texttt{ else } C' \texttt{ fi}$, we use the following notation.

$$
\begin{aligned}
\rho \to |0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)} &\xrightarrow{C} \rho' \to \rho''' \\
\searrow |1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)} &\xrightarrow{C'} \rho'' \nearrow
\end{aligned}
\tag{3.8}
$$

where $\rho' = [\![C]\!](|0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)})$, $\rho'' = [\![C']\!](|1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)})$, and $\rho''' = \rho' + \rho''$. Therefore, $\rho''' = [\![\texttt{if } j \texttt{ then } C \texttt{ else } C' \texttt{ fi}]\!](\rho)$.

Example 3.1.6. Let $\rho \in \mathcal{D}_N$.

$$
\begin{aligned}
\rho \to |0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)} &\xrightarrow{\texttt{skip}} |0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)} \to \sigma \\
\searrow |1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)} &\xrightarrow{\texttt{skip}} |1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)} \nearrow
\end{aligned}
\tag{3.9}
$$

$$
\begin{aligned}
\rho \to |0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)} &\xrightarrow{\texttt{skip}} |0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)} \to \mathrm{tr}_j\rho \otimes |0\rangle\langle 0|_{[j]} \\
\searrow |1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)} &\xrightarrow{\texttt{X(}j\texttt{)}} |0\rangle\langle 1|_{(j)}\rho|1\rangle\langle 0|_{(j)} \nearrow
\end{aligned}
\tag{3.10}
$$

$$
\rho \xrightarrow{\texttt{INIT}_0} \mathrm{tr}_0\rho \otimes |0\rangle\langle 0|_{[0]} \xrightarrow{\texttt{INIT}_1} \cdots \xrightarrow{\texttt{INIT}_{N-1}} \mathrm{tr}\rho|00\cdots 0\rangle\langle 00\cdots 0| \tag{3.11}
$$

where $\sigma = |0\rangle\langle 0|_{(j)}\rho|0\rangle\langle 0|_{(j)} + |1\rangle\langle 1|_{(j)}\rho|1\rangle\langle 1|_{(j)}$. Therefore, $[\![\texttt{MEASURE}_j]\!](\rho)$ is σ, $[\![\texttt{INIT}_j]\!](\rho)$ is $\mathrm{tr}_j\rho \otimes |0\rangle\langle 0|_{(j)}$, and $[\![\texttt{INIT}]\!](\rho) = \mathrm{tr}\rho|00\cdots 0\rangle\langle 00\cdots 0|$.

Example 3.1.7. For simplicity, we assume $N = 3$ in the example. Let $\sigma \in \mathcal{D}_1$.

$$
[\![\texttt{teleportation}]\!](\sigma \otimes |\mathrm{Bell}_0\rangle\langle \mathrm{Bell}_0|) = \frac{1}{4}\mathrm{I}^{\otimes 2} \otimes \sigma \tag{3.12}
$$

3.1.2 Basis abstract semantics

Suppose we have a QIL program C. If we run the program C on an input quantum state ρ, we will obtain a quantum state $[\![C]\!](\rho)$. A question is how entanglements between these qubits change through the program C; how the program C evolves entanglements. That can be viewed differently. Given a quantum state σ, it is difficult to determine whether it is separable [57, 60]. When we have a program C and a state ρ such that $[\![C]\!](\rho) = \sigma$, what can we say about entanglements in the state? If we know how C changes entanglements and how entangled qubits in ρ are, then we can learn something about σ. This is why we want to analyse entanglement in quantum programs.

For example, in Example 3.1.7, we can see that the first and second qubits of the qubits in the input state are entangled. However, they are separable after running `teleportation`. That means the program `teleportation` destroys the entanglement between them. In a paper [89], abstract interpretation was used to analyse entanglement. Its domain is based on the following observation: Quantum entanglement results from application of a unitary operator on multiple qubits. Since QIL has no multiple qubits operator other than CX, `CX(`j`,`k`)` is the place where qubits become entangled. An important fact is that CX does not always entangle given qubits. If either the state of the jth qubit is in the standard basis or that of the kth qubit is in the X basis, these qubits are separable even after applying CX:

$$\mathrm{CX}(|d\rangle \otimes |\psi\rangle) = |d\rangle \otimes (\mathrm{Z}^d |\psi\rangle) \tag{3.13}$$

$$\mathrm{CX}(|\psi\rangle \otimes |+_{d\pi}\rangle) = (\mathrm{X}^d |\psi\rangle) \otimes |+_{d\pi}\rangle \tag{3.14}$$

where $d \in \mathbb{B}$, $|\psi\rangle$ is a pure state. Therefore, we can guess evolution of entanglements from information about basis where the state of each qubit belongs. In the domain, a quantum state is abstracted into a pair of a partition and a function. The former records how entangled the state is and the latter records bases.

Definition 3.1.8. Let $B = \{\, \mathrm{Z}, \mathrm{X}, \top, \bot \,\}$. An order $\leq_B$ on the set is defined by $\bot \leq_B \mathrm{Z} \leq_B \top$ and $\bot \leq_B \mathrm{X} \leq_B \top$. $\mathbf{A}^{\mathbf{Q}}$ is defined as $\mathbf{\Pi}^{\mathbf{Q}} \times (\mathbf{Q} \Rightarrow B)$. An order $\leq_{\mathbf{A}}$ on the domain is the product order. That is, for any $(P, b), (P', b') \in \mathbf{A}^{\mathbf{Q}}$, $(P, b) \leq_{\mathbf{A}} (P', b')$ if and only if $P \leq_{\mathbf{\Pi}} P'$ and $b(j) \leq_B b'(j)$ for any $j \in \mathbf{Q}$.

Since the domain uses the information about bases, we call $\mathbf{A}^{\mathbf{Q}}$ the *basis domain*. The order of the domain tells us which one has more information.

Proposition 3.1.9 ([89])**.** $\mathbf{A}^{\mathbf{Q}}$ *is a complete lattice.*

Note that $\mathcal{D}_N$ is not a lattice. The domain $\mathbf{A}^{\mathbf{Q}}$ gives an approximation of entanglement in a state: Let (P, b) be an abstract state and ρ be a concrete state. If there is no $Q \in P$ such that $\{\, j, k \,\} \subset Q$, the jth and kth qubits of the quantum variables in ρ are separable. However, even if $\{\, j, k \,\} \subset Q \in P$ for some Q, it does not mean these qubits are entangled. Which abstract state is a sound approximation of a concrete state is formalised as the following soundness relation.

Definition 3.1.10. Let $\rho \in \mathcal{D}_N$ and $(P, b) \in \mathbf{A}^{\mathbf{Q}}$. We write $(P, b) \vDash_{\mathbf{A}} \rho$ if ρ is P-separable and the following holds for any $j \in \mathbf{Q}$.

- If $b(j) = \mathrm{Z}$, $\langle 0|_{(j)} \rho |1\rangle_{(j)} = 0$.
- If $b(j) = \mathrm{X}$, $\langle +|_{(j)} \rho |-\rangle_{(j)} = 0$.

$$
\begin{aligned}
[\![\texttt{skip}]\!]_{\mathbf{A}}(P,b) &= (P,b)\\
[\![C;C']\!]_{\mathbf{A}}(P,b) &= [\![C']\!]_{\mathbf{A}}([\![C]\!]_{\mathbf{A}}(P,b))\\
[\![\sigma(j)]\!]_{\mathbf{A}}(P,b) &= (P,b)\\
[\![\texttt{H}(j)]\!]_{\mathbf{A}}(P,b) &= \begin{cases} (P,b[j\mapsto \mathrm{X}]) & (b(j)=\mathrm{Z})\\ (P,b[j\mapsto \mathrm{Z}]) & (b(j)=\mathrm{X})\\ (P,b) & (\text{otherwise}) \end{cases}\\
[\![U(j)]\!]_{\mathbf{A}}(P,b) &= \begin{cases} (P,b[j\mapsto \top]) & (b(j)=\mathrm{X})\\ (P,b[j\mapsto \mathrm{Z}]) & (b(j)=\bot)\\ (P,b) & (\text{otherwise}) \end{cases}\\
[\![\texttt{CX}(j,k)]\!]_{\mathbf{A}}(P,b) &= \begin{cases} (P,b[j\mapsto \mathrm{Z},k\mapsto \mathrm{X}]) & (b(j)=b(k)=\bot)\\ (P,b[j\mapsto \mathrm{Z}]) & (b(j)=\bot \text{ and } b(k)>_{\mathbf{A}}\bot)\\ (P,b[k\mapsto \mathrm{X}]) & (b(j)>_{\mathbf{A}}\bot \text{ and } b(k)=\bot)\\ (P,b) & (b(j)=\mathrm{Z} \text{ or } b(k)=\mathrm{X})\\ (P\vee P_{j,k}, & \\ \quad b[j\mapsto\top,k\mapsto\top]) & (\text{otherwise}) \end{cases}
\end{aligned}
$$

$$
\left[\!\!\left[\begin{array}{l} \texttt{if } j\\ \quad \texttt{then } C\\ \quad \texttt{else } C'\\ \texttt{fi} \end{array} \right]\!\!\right]_{\mathbf{A}} (P,b) = [\![C]\!]_{\mathbf{A}}(\mathrm{meas}_{\mathbf{A},j}(P,b)) \vee [\![C']\!]_{\mathbf{A}}(\mathrm{meas}_{\mathbf{A},j}(P,b))
$$

$$
[\![\texttt{while } j \texttt{ do } C \texttt{ od}]\!]_{\mathbf{A}}(P,b) = \bigvee_{n\in\mathbb{N}} (\mathrm{meas}_{\mathbf{A},j}\circ([\![C]\!]_{\mathbf{A}}\circ\mathrm{meas}_{\mathbf{A},j})^n)(P,b)
$$

where $j,k\in\mathbf{Q}$, $\sigma\in\{\,\mathtt{X},\mathtt{Y},\mathtt{Z}\,\}$, $U\in\{\,\mathtt{S},\mathtt{T}\,\}$, $P_{j,k}=\{\{\,l\,\}\mid l\neq j,k\,\}\cup\{\{\,j,k\,\}\}$, and $\mathrm{meas}_{\mathbf{A},j}(P,b)=(\{\{\,j\,\}\}\cup\{\,Q\setminus\{\,j\,\}\mid Q\in P\,\},b[j\mapsto\mathrm{Z}])$.

Figure 3.2: Basis abstract semantics of quantum imperative language

- If $b(j)=\bot$, $\langle 0|_{(j)}\rho|1\rangle_{(j)}=\langle +|_{(j)}\rho|-\rangle_{(j)}=0$.

The first condition states that ρ can be decomposed into $|0\rangle\langle 0|_{(j)}\otimes\sigma_0+|1\rangle\langle 1|_{(j)}\otimes\sigma_1$. Similarly, the second condition means that ρ is $|+\rangle\langle +|_{(j)}\otimes\sigma_0+|-\rangle\langle -|_{(j)}\otimes\sigma_1$ with some σ_0 and σ_1. Therefore, the third condition means that $\rho=\frac{1}{2}\mathrm{I}_{(j)}\otimes\sigma$ with some σ.

On the abstraction of states, an abstract semantics was given to QIL programs. We call the semantics the *basis abstract semantics*.

Definition 3.1.11. The *basis abstract semantics* of QIL programs is a function $[\![\cdot]\!]_{\mathbf{A}}\colon \mathbf{QIL}\to\mathbf{A}^{\mathbf{Q}}\to\mathbf{A}^{\mathbf{Q}}$ defined in Figure 3.2.

Although the Pauli operators do not change the basis, the operator H exchanges a state in the standard basis with a state in the X basis. Both T and S commute with Z, so if a state belongs to the standard basis, the applied state still belongs to the basis. The semantics of CX follows from the observation described before. $\mathrm{meas}_{\mathbf{A},j}$ is measurement of the jth qubit. After measuring the jth qubit, the state of the qubit definitely belongs to the standard basis and the qubit is separable from the other qubits.

Notation 3.1.12. We sometimes write $(a_j)_{j<N}$ to denote a function b such that $b(j)=a_j$.

Example 3.1.13. Let $(P, b) \in \mathbf{A}^{\mathbf{Q}}$.

$$[\![\texttt{MEASURE}_j]\!](P,b) = (\{\{\, j \,\}\} \cup \{\, Q \setminus \{\, j \,\} \mid Q \in P \,\}, b[j \mapsto \mathrm{Z}]) \tag{3.15}$$

$$[\![\texttt{INIT}_j]\!](P,b) = (\{\{\, j \,\}\} \cup \{\, Q \setminus \{\, j \,\} \mid Q \in P \,\}, b[j \mapsto \mathrm{Z}]) \tag{3.16}$$

$$[\![\texttt{INIT}]\!](\rho) = (\{\, \{\, j \,\} \mid j \in \mathbf{Q} \,\}, \mathrm{Z}) \tag{3.17}$$

where the last Z is the constant function to Z. Assume $(P, b) \vDash_{\mathbf{A}} \rho$. In particular, ρ is P-separable. Then, $\mathrm{tr}_j\rho \otimes |0\rangle\langle 0|_{[j]}$ is P'-separable where $P' = \{\{\, j \,\}\} \cup \{\, Q \setminus \{\, j \,\} \mid Q \in P \,\}$. $\langle 0|_{(j)}\mathrm{tr}_j\rho \otimes |0\rangle\langle 0|_{[j]}|1\rangle_{(j)} = \mathrm{tr}_j\rho \otimes \langle 0|0\rangle\,\langle 0|1\rangle = 0$. Therefore, $(\{\{\, j \,\}\} \cup \{\, Q \setminus \{\, j \,\} \mid Q \in P \,\}, b[j \mapsto \mathrm{Z}]) \vDash_{\mathbf{A}} \mathrm{tr}_j\rho \otimes |0\rangle\langle 0|_{[j]}$.

Example 3.1.14. Assume $N = 3$.

$$\begin{aligned}
&[\![\texttt{teleportation}]\!]_{\mathbf{A}}(\{\{\, 0 \,\}, \{\, 1, 2 \,\}\}, (\top, \top, \top)) \\
&= [\![\texttt{if 1 then skip else X(2) fi}]\!]_{\mathbf{A}}(\{\{\, 0 \,\}, \{\, 1, 2 \,\}\}, (\mathrm{Z}, \top, \top)) \\
&\vee [\![\texttt{if 1 then Z(2) else X(2);Z(2) fi}]\!]_{\mathbf{A}}(\{\{\, 0 \,\}, \{\, 1, 2 \,\}\}, (\mathrm{Z}, \top, \top)) \\
&= (\{\{\, 0 \,\}, \{\, 1 \,\}, \{\, 2 \,\}\}, (\mathrm{Z}, \mathrm{Z}, \top))
\end{aligned} \tag{3.18}$$

It is easy to see that $(\{\{\, 0 \,\}, \{\, 1, 2 \,\}\}, (\top, \top, \top)) \vDash_{\mathbf{A}} \sigma \otimes |\mathrm{Bell}_0\rangle\langle\mathrm{Bell}_0|$. Since $\langle 0|\mathrm{I}|1\rangle = 0$, $(\{\{\, 0 \,\}, \{\, 1 \,\}, \{\, 2 \,\}\}, (\mathrm{Z}, \mathrm{Z}, \top)) \vDash_{\mathbf{A}} \frac{1}{4}\mathrm{I}^{\otimes 2} \otimes \sigma$.

These examples suggest that the basis abstract semantics correctly works. Indeed, the semantics is sound. If a sound approximation of an input state is given, the semantics gives a sound approximation of an output state. Moreover, the semantics respects the order of the basis domain.

Lemma 3.1.15 ([89])**.** *Let $C \in$ **QIL** and $(P, b), (P', b') \in \mathbf{A}^{\mathbf{Q}}$ such that $(P, b) \leq_{\mathbf{A}} (P', b')$. $[\![C]\!]_{\mathbf{A}}((P, b)) \leq_{\mathbf{A}} [\![C]\!]_{\mathbf{A}}((P', b'))$.*

Theorem 3.1.16 ([89])**.** *For any $C \in$ **QIL**, any $(P, b) \in \mathbf{A}^{\mathbf{Q}}$, and any state ρ, $(P, b) \vDash_{\mathbf{A}} \rho$ implies $[\![C]\!]_{\mathbf{A}}(P, b) \vDash_{\mathbf{A}} [\![C]\!](\rho)$.*

3.2 Stabiliser abstract semantics

3.2.1 Motivation and idea

As explained in the previous section, the basis abstract semantics $[\![\cdot]\!]_{\mathbf{A}}$ gives a good approximation of how quantum entanglement evolves through a QIL program. However, it can be found in very simple examples that the semantics fails to precisely trace the evolution.

Example 3.2.1. Let us define the following programs.

$$\texttt{GHZ} \equiv \texttt{INIT;H(0);CX(0,1);CX(1,2)} \tag{3.19}$$

$$\texttt{INVERTGHZ} \equiv \texttt{GHZ;CX(1,2);CX(0,1);H(0)} \tag{3.20}$$

$$\texttt{BREAKGHZ} \equiv \texttt{GHZ;MEASURE}_0 \tag{3.21}$$

In the concrete semantics, the output states are as follows. For the sake of simplicity, we assume $N = 3$.

$$\begin{aligned}
&\rho \xrightarrow{\texttt{INIT}} \mathrm{tr}\rho\,|000\rangle\langle 000| \xrightarrow{\texttt{H(0)}} \mathrm{tr}\rho\,|{+}00\rangle\langle{+}00| \\
&\xrightarrow{\texttt{CX(0,1)}} \mathrm{tr}\rho\,|\mathrm{Bell}_0\rangle\langle\mathrm{Bell}_0| \otimes |0\rangle\langle 0| \xrightarrow{\texttt{CX(1,2)}} \mathrm{tr}\rho\,|\mathrm{GHZ}\rangle\langle\mathrm{GHZ}|
\end{aligned} \tag{3.22}$$

$$\begin{aligned}
&\rho \xrightarrow{\texttt{GHZ}} \mathrm{tr}\rho\,|\mathrm{GHZ}\rangle\langle\mathrm{GHZ}| \xrightarrow{\texttt{CX(1,2)}} \mathrm{tr}\rho\,|\mathrm{Bell}_0\rangle\langle\mathrm{Bell}_0| \otimes |0\rangle\langle 0| \\
&\xrightarrow{\texttt{CX(0,1)}} \mathrm{tr}\rho\,|{+}00\rangle\langle{+}00| \xrightarrow{\texttt{H(0)}} \mathrm{tr}\rho\,|000\rangle\langle 000|
\end{aligned} \tag{3.23}$$

$$\rho \xrightarrow{\texttt{GHZ}} \mathrm{tr}\rho\,|\mathrm{GHZ}\rangle\langle\mathrm{GHZ}| \xrightarrow{\texttt{MEASURE}_{(0)}} \frac{1}{2}\mathrm{tr}\rho(|000\rangle\langle 000| + |111\rangle\langle 111|). \tag{3.24}$$

Therefore, we can find that all qubits in $[\![\texttt{GHZ}]\!](\rho)$ are entangled but those in $[\![\texttt{INVERTGHZ}]\!](\rho)$ or $[\![\texttt{BREAKGHZ}]\!](\rho)$ are all separable. The transitions of the basis abstract semantics are as follows.

$$\begin{aligned}(P,b) &\xrightarrow{\texttt{INIT}} (\{\{0\},\{1\},\{2\}\},(\mathrm{Z},\mathrm{Z},\mathrm{Z}))\\ &\xrightarrow{\texttt{H(0)}} (\{\{0\},\{1\},\{2\}\},(\mathrm{X},\mathrm{Z},\mathrm{Z}))\\ &\xrightarrow{\texttt{CX(0,1)}} (\{\{0,1\},\{2\}\},(\top,\top,\mathrm{Z}))\\ &\xrightarrow{\texttt{CX(1,2)}} (\{\{0,1,2\}\},(\top,\top,\top))\end{aligned} \tag{3.25}$$

$$\begin{aligned}(P,b) &\xrightarrow{\texttt{GHZ}} (\{\{0,1,2\}\},(\top,\top,\top)) \xrightarrow{\texttt{CX(0,2)}} (\{\{0,1,2\}\},(\top,\top,\top))\\ &\xrightarrow{\texttt{CX(0,1)}} (\{\{0,1,2\}\},(\top,\top,\top)) \xrightarrow{\texttt{H(0)}} (\{\{0,1,2\}\},(\top,\top,\top))\end{aligned} \tag{3.26}$$

$$(P,b) \xrightarrow{\texttt{GHZ}} (\{\{0,1,2\}\},(\top,\top,\top)) \xrightarrow{\texttt{MEASURE}_{(0)}} (\{\{0\},\{1,2\}\},(\mathrm{Z},\top,\top)) \tag{3.27}$$

The basis abstract semantics succeeds in reasoning entanglement in $[\![\texttt{GHZ}]\!](\rho)$. However, it claims that all qubits in $[\![\texttt{INVERTGHZ}]\!](\rho)$ are entangled and that the first and second qubits of the three qubits in $[\![\texttt{BREAKGHZ}]\!](\rho)$ are entangled.

Of course, the purpose of the abstract semantics is to give an approximation. Therefore, these examples are not counterexamples of the soundness of the semantics. However, needless to say, more precise approximations are better as long as they are efficiently computable. Why does the basis semantics fail in the above examples? Tracing transitions, we can find that the causes of the failures are $(\{\{0,1,2\}\},(\top,\top,\top)) \xrightarrow{\texttt{CX(0,2)}} (\{\{0,1,2\}\},(\top,\top,\top))$ in `INVERTGHZ` and, in `BREAKGHZ`, $(\{\{0,1,2\}\},(\top,\top,\top)) \xrightarrow{\texttt{MEASURE}_{(0)}} (\{\{0\},\{1,2\}\},(\mathrm{Z},\top,\top))$. In the basis abstract semantics, the unitary operator on multiple qubits, `CX(`j`,`k`)`, just entangles quantum variables and only measurement destroys entanglements. However, the unitary operator also undoes an entanglement:

$$\mathrm{CX}\left(\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)\right) = \frac{1}{\sqrt{2}}(|00\rangle+|10\rangle) = |{+}0\rangle \tag{3.28}$$

Moreover, in the basis semantics, measurement destroys only entanglements with the measured qubit. In the concrete semantics, measurement may make non-measured qubits separable as shown in (3.24).

How can we obtain a better semantics? The above observations do not mean that the operator on multiple qubits always undoes an entanglement or measurement destroys any entanglements between qubits. Indeed,

$$\mathrm{CX}\left(\frac{1}{\sqrt{2}}(|0{-}\rangle+|1{+}\rangle)\right) = \frac{1}{\sqrt{2}}(|0{-}\rangle+|1{+}\rangle) \tag{3.29}$$

$$\begin{aligned}&|0\rangle\langle 0|_{(0)}\rho|0\rangle\langle 0|_{(0)} + |1\rangle\langle 1|_{(0)}\rho|1\rangle\langle 1|_{(0)}\\ &= \frac{1}{2}|0\rangle\langle 0|\otimes|\mathrm{Bell}_0\rangle\langle\mathrm{Bell}_0| + \frac{1}{2}|1\rangle\langle 1|\otimes|\mathrm{Bell}_2\rangle\langle\mathrm{Bell}_2|\end{aligned} \tag{3.30}$$

where $\rho = \mathrm{H}_{(0)}|\mathrm{GHZ}\rangle\langle\mathrm{GHZ}|\mathrm{H}^\dagger_{(0)}$. It means that we have to record some information about the entangled state and decide, for example, whether CX undoes an entanglement or not. The basis abstract semantics does not do that. After a variable is entangled with other variables, a function in a basis abstract state gives no meaningful information about the variable. How can we record information about entangled qubits? In order to make our semantics useful, it should be easy to compute. These restrictions lead us to the stabiliser formalism.

3.2.2 Stabiliser array

We follow an idea of the basis domain: bases are good abstractions of states. As explained in the previous chapter, a stabiliser uniquely determines its stabiliser states. Here, we regard a stabiliser as an expression of a basis.

Definition 3.2.2. Let $S = \langle M_0, \ldots, M_{n-1} \rangle$ be a stabiliser on n qubits. The S *basis* is the set of simultaneous eigenstates of S, which is the set of all $|\psi\rangle$ such that for any $M \in S$, $M|\psi\rangle = \pm|\psi\rangle$.

Example 3.2.3. The basis defined by the stabiliser $\langle \mathrm{Z} \rangle$ is the standard basis. The $\langle \mathrm{XX}, \mathrm{ZZ} \rangle$ basis is nothing but the Bell basis, which contains $|\mathrm{Bell}_0\rangle$ and $|\mathrm{Bell}_2\rangle$.

The set of stabilisers is isomorphic to the set of stabiliser states. Hence, a stabiliser basis does not uniquely determine a stabiliser. However, it is uniquely determined up to signs.

Proposition 3.2.4. *Let $S = \langle M_0, \ldots, M_{n-1} \rangle$ and S' be stabilisers on n qubits. The S basis is the S' basis if and only if $S' = \langle (-1)^{s_0} M_0, \ldots, (-1)^{s_{n-1}} M_{n-1} \rangle$ with some $s_0, \ldots, s_{n-1} \in \mathbb{B}$.*

Proof. Assume $S' = \langle (-1)^{s_0} M_0, \ldots, (-1)^{s_{n-1}} M_{n-1} \rangle$. $|\psi\rangle$ is an eigenstate of M_j if and only if $|\psi\rangle$ is an eigenstate of $-M_j$. Conversely, assume that the S basis is the S' basis. Take $M_j \in S$. It commutes with any element of S'. Therefore, $M_j \in S'$ or $-M_j \in S'$. □

The proposition tells us that the signs of Pauli matrices in a stabiliser are no longer meaningful in our interpretation. In the chapter, we omit to care about the signs. Furthermore, we ignore any global phase. For example, we say that Y is the product of X and Z, write $(\mathrm{X} \otimes \mathrm{X})(\mathrm{Z} \otimes \mathrm{Z}) = \mathrm{Y} \otimes \mathrm{Y}$, and so on. However, we should emphasise that we exactly keep the meaning of the term "commute". X does not commute with Z, although we write $\mathrm{XZ} = \mathrm{Y} = \mathrm{ZX}$. Any element of a stabiliser commutes with another element.

Now, let us explain an expression of stabilisers. In the stabiliser formalism, a generator determines a stabiliser. We employ a variant of tableaus [1, 8, 82] consisting of Pauli matrices as an expression of generators. Note that we do not use destabiliser generators [1].

Definition 3.2.5. Let n be a natural number. A *syntactic stabiliser array on n qubits* is an $n \times n$ matrix S of Pauli matrices such that $\left\langle \bigotimes_j S_{\{0,j\}}, \ldots, \bigotimes_j S_{\{n-1,j\}} \right\rangle$ is a stabiliser on n qubits. The *syntactic stabiliser space on n qubits* $\mathbf{SSS}_n$ is the set of syntactic stabiliser arrays on n qubits. We usually identify a row $S_{\{j,\}}$ of a syntactic stabiliser array with a Pauli matrix $\bigotimes_k S_{\{j,k\}}$. *Multiplication of the jth row and the kth row* is an operation that replaces the jth row with the product of the jth row and the kth row. Note that the product has no sign.

Roughly speaking, a syntactic stabiliser array is an ordered generator. It is an expression of a stabiliser array.

Definition 3.2.6. Let S, T be syntactic stabiliser arrays. We write $S \simeq T$ if any row of S is a product of rows of T, and vice versa. The *stabiliser space on n qubits* $\mathbf{SS}_n$ is the quotient set $\mathbf{SSS}_n/\simeq$. A *stabiliser array* is an element of the space.

A stabiliser array represents a sign-less stabiliser. It is a mathematical object. What we operate is a syntactic stabiliser array. Each algorithm is performed on a syntactic one.

Notation 3.2.7. A stabiliser array is usually denoted by its representative.

Example 3.2.8. The following matrices are stabiliser arrays.

$$\begin{bmatrix} \mathrm{Z} \end{bmatrix}, \begin{bmatrix} \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} \end{bmatrix} = \begin{bmatrix} \mathrm{Y} & \mathrm{Y} \\ \mathrm{Z} & \mathrm{Z} \end{bmatrix} = \begin{bmatrix} \mathrm{Z} & \mathrm{Z} \\ \mathrm{Y} & \mathrm{Y} \end{bmatrix}, \begin{bmatrix} \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{bmatrix} \tag{3.31}$$

None of the following matrices is a stabiliser array.

$$\begin{bmatrix} \mathrm{X} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{Z} \end{bmatrix}, \begin{bmatrix} \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{Y} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{bmatrix}, \begin{bmatrix} \mathrm{I} & \mathrm{I} & \mathrm{I} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{bmatrix} \tag{3.32}$$

Notation 3.2.9. We also use the generator notation. For example, $\begin{bmatrix} \mathrm{X} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{X} \end{bmatrix}$ is written as $\langle \mathrm{XZ}, \mathrm{ZX} \rangle$. Note that it is a stabiliser array. When the above matrix is a syntactic stabiliser array, we use $(\mathrm{XZ}, \mathrm{ZX})$ to denote it because it has the order.

Definition 3.2.10. Let S be a stabiliser on n qubits and S' be the associated stabiliser array. The S' *basis* is the S basis.

Proposition 3.2.11. *Any stabiliser basis is uniquely determined by its stabiliser array.*

Proof. By Proposition 3.2.4. □

In order to use stabiliser arrays to analyse entanglement, we have to confirm that stabiliser arrays are equipped with enough operations. Can we check whether given two stabiliser arrays are the same or not? The equality check of syntactic stabiliser arrays is obvious, and that is still easy for stabiliser arrays: Check whether each row of an array commutes with all rows of the other array.

How about separability? As explained before, we want to know whether a state is separable or not after applying CX to the state. How can we check whether a stabiliser basis state is separable or not? If it is separable, how can we decompose the state? Although they are not obvious, we can do that via stabiliser arrays and a variant of the row-reduced echelon form (RREF) algorithm [8]. First of all, we define separability on stabiliser arrays.

Definition 3.2.12. Let S and T be syntactic stabiliser arrays on n and m qubits. The *tensor product* $S \otimes T$ of S and T is the matrix defined by

$$(S \otimes T)_{\{j,k\}} = \begin{cases} S_{\{j,k\}} & (j,k < n) \\ T_{\{j-n,k-n\}} & (j,k \geq n) \\ \mathrm{I} & (\text{otherwise}) \end{cases}. \tag{3.33}$$

More generally, if $P = \{ Q_0, Q_1 \}$ is a partition of $[<(n+m)]$ such that $|Q_0| = n$,

$$(S \otimes_P T)_{\{j,k\}} = \begin{cases} S_{\{j,l\}} & (j < n \text{ and } k = k_l \in Q_0) \\ T_{\{j-n,l\}} & (j \geq n \text{ and } k = k_l \in Q_1) \\ \mathrm{I} & (\text{otherwise}) \end{cases}. \tag{3.34}$$

We usually omit to write the subscript P.

Proposition 3.2.13. *The tensor product of syntactic stabiliser arrays is a syntactic stabiliser array. The tensor product of stabiliser arrays is well-defined.* □

Definition 3.2.14. Let S be a syntactic stabiliser array on n qubits and P be a partition of $[<n]$. S is *P-separable* if there exists $\left\{S_{[Q]}\right\}_{Q\in P}$ such that each $S_{[Q]}$ is a syntactic stabiliser array on $|Q|$ qubits and $S \simeq \bigotimes_{Q\in P} S_{[Q]}$. $\mathbf{NSS}_n$ is defined to be the set of stabiliser arrays on n qubits that are not P-separable unless P is a singleton. **NSS** is the union of them.

The following proposition justifies that we use stabiliser arrays to analyse separability of stabiliser basis states.

Proposition 3.2.15. *Let S be a stabiliser array on n qubits and P be a partition of $[<n]$. S is P-separable if and only if all S basis states are P-separable.*

Proof. It is enough to prove the proposition when $P = \{ Q_0, Q_1 \}$. Assume $S = S_{[Q_0]} \otimes S_{[Q_1]}$. Let $\{|\psi_j\rangle\}_j$, $\{|\sigma_j\rangle\}_j$, and $\{|\phi_j\rangle\}_j$ be the $S, S_{[Q_0]}$, and $S_{[Q_1]}$ bases. Take $|\psi_j\rangle$. $|\psi_j\rangle = \sum_{k,l} \alpha_{k,l} |\sigma_k\rangle \otimes |\phi_l\rangle$ with some $\{\alpha_{k,l}\}_{k,l}$. For any $M \in S_{[Q_0]}$, the eigenvalue of $M \otimes \mathrm{I}$ corresponding to $|\psi_j\rangle$ is either 1 or -1. There exists a unique $|\sigma_k\rangle$ that has the same eigenvalues. Similarly, we can obtain $|\phi_l\rangle$, and $|\psi_j\rangle = \alpha_{k,l} |\sigma_k\rangle \otimes |\phi_l\rangle$.

Assume all S basis states are P-separable. Let $\{|\psi_j\rangle\}_j$ be the S basis. By the assumption, each $|\psi_j\rangle$ can be written as $|\sigma_j\rangle \otimes |\phi_j\rangle$. Take a row $S_{\{k,\}} = M_k \otimes N_k$. If $(M_k \otimes N_k) |\psi_j\rangle = \pm |\psi_j\rangle$, $|\sigma_j\rangle = (\mathrm{I} \otimes \langle\phi_j|) |\psi_j\rangle = \pm(\mathrm{I} \otimes \langle\phi_j|)(M_k \otimes N_k)(|\sigma_j\rangle \otimes |\phi_j\rangle) = \pm \langle\phi_j|N_k|\phi_j\rangle M_k |\sigma_j\rangle$. Therefore, $|\sigma_j\rangle$ is an eigenstate of M_k. It is true for any j and k. Since $\{|\sigma_j\rangle\}_j$ is a basis, all $\{M_k\}_k$ commute. □

Note that the above proof shows that if the $S \otimes T$ basis is the tensor product of two bases, these two bases are the S basis and the T basis. Therefore, we try to decompose stabiliser arrays. The decomposition is essentially unique. The following proposition and corollary shows there exists the finest decomposition.

Proposition 3.2.16. *Let S be a syntactic stabiliser array and P, P' be a partition of $[<n]$. If S is both P-separable and P'-separable, S is $P \wedge P'$-separable.*

Proof. Take $\left\{S_{P,[Q]}\right\}_{Q\in P}$ and $\left\{S_{P',[Q]}\right\}_{Q\in P'}$ such that $S \simeq \bigotimes_{Q\in P} S_{P,[Q]}$ and $S \simeq \bigotimes_{Q\in P'} S_{P',[Q]}$. Take $A \in P \wedge P'$. Let $Q \in P$ and $Q' \in P'$ be such that $A = Q \cap Q'$. Let $T = S_{P,[Q]}$ and $T' = S_{P',[Q']}$. Take a row $T_{\{i,\}}$ such that $T_{\{i,j\}} \neq \mathrm{I}$ for some $j \in A$. Define M by $M_{\{j\}}$ is $T_{\{i,j\}}$ if $j \in A$ and otherwise I. Since $T_{\{i,\}}$ commutes with all rows of T', M also commutes with all of them. It means that M commutes with all rows of T and there exists T'' such that $T \simeq T''$ and $T''_{\{k,\}} = M$ for some k. Repeating the operation, we can finally obtain R such that $T \simeq R$ and for any row $R_{\{i,\}}$, $R_{\{i,j\}} \neq \mathrm{I}$ with some $j \in A$ if and only if $R_{\{i,j\}} = \mathrm{I}$ for all $j \notin A$. The number $N(A)$ of rows $R_{\{i,\}}$ such that $R_{\{i,j\}} \neq \mathrm{I}$ with some $j \in A$ is at most $|A|$ because these rows are independent. That is true for any $A' \in P \wedge P'$ such that $A' \subset Q$. The sum of these numbers should be equal to the number of all rows, $|Q|$. $\sum_{A\subset Q} N(A) \leq \sum_{A\subset Q} |A| = |Q|$, so $N(A) = |A|$ for any A. Therefore, the rows form a syntactic stabiliser array $S_{P\wedge P',[A]}$. By construction, $S_{P,[Q]} \simeq \bigotimes_{Q\supset A\in P\wedge P'} S_{P\wedge P',[A]}$. □

Corollary 3.2.17. *For any syntactic stabiliser array, there exists the finest partition P such that the array is P-separable.* □

Finally, we show an algorithm to decompose a syntactic stabiliser array.

Proposition 3.2.18. *Let S be a syntactic stabiliser array on n qubits and P be a partition of $[<n]$. Let S' be an output of Algorithm 1. Then, $S \simeq S'$. Furthermore, S is P-separable if and only if there exists $\left\{S'_{[Q]}\right\}_{Q\in P}$ such that $S' = \bigotimes_{Q\in P} S'_{[Q]}$.*

Algorithm 1 Decomposition of a syntactic stabiliser array S on n qubits

1: $r = 0$.
2: **for** $c = 0$ to $n-1$ **do**
3: **if** There exists r_0 such that $r \le r_0$ and $S_{r_0,c} \neq \mathrm{I}$ **then**
4: Take the smallest r_0.
5: Swap the rth row and the r_0th row.
6: **if** There exists r_1 such that $r < r_1$ and $S_{r,c} \neq S_{r_1,c} \neq \mathrm{I}$ **then**
7: Take the smallest r_1.
8: Swap the $r+1$th row and the r_1th row.
9: For any r'th row such that $S_{r',c}$ is either $S_{r,c}$ or $S_{r,c}S_{r+1,c}$, multiply the row and the rth row.
10: For any r'th row such that $S_{r',c}$ is $S_{r+1,c}$, multiply the row and the $r+1$th row.
11: $r = r+2$.
12: **else**
13: For any r'th row such that $r' > r$ and $S_{r',c} = S_{r,c}$, multiply the row and the rth row.
14: **if** There exists r_2 such that $r > r_2$ and the Pauli matrix $S_{r_2,c}$ is neither I nor $S_{r,c}$ **then**
15: Take the smallest r_2.
16: For any r'th row such that $r' < r$ and $S_{r',c}$ is either $S_{r,c}$ or $S_{r,c}S_{r_2,c}$, multiply the row and the rth row.
17: **else**
18: For any r'th row such that $r' < r$ and $S_{r',c}$ is $S_{r,c}$, multiply the row and the rth row.
19: **end if**
20: $r = r+1$.
21: **end if**
22: **end if**
23: **end for**

Proof. We first observe an output of the algorithm has a row echelon-like form such as

$$T = \begin{bmatrix} \boxed{T_{0,0}} & \mathrm{I} & \mathrm{I} & T_{0,3} & T_{0,4} & \cdots & T_{0,n-2} & T_{0,n-1} \\ \mathrm{I} & \boxed{T_{1,1}} & \mathrm{I} & T_{1,3} & T_{1,4} & \cdots & T_{1,n-2} & T_{1,n-1} \\ \mathrm{I} & \boxed{T_{2,1}} & \mathrm{I} & T_{2,3} & T_{2,4} & \cdots & T_{2,n-2} & T_{2,n-1} \\ \mathrm{I} & \mathrm{I} & \boxed{T_{3,2}} & T_{3,3} & T_{3,4} & \cdots & T_{3,n-2} & T_{3,n-1} \\ \mathrm{I} & \mathrm{I} & \boxed{T_{4,2}} & T_{4,3} & T_{4,4} & \cdots & T_{4,n-2} & T_{4,n-1} \\ \mathrm{I} & \mathrm{I} & \mathrm{I} & \boxed{T_{5,3}} & T_{5,4} & \cdots & T_{5,n-2} & T_{5,n-1} \\ \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \boxed{T_{6,4}} & \cdots & T_{6,n-2} & T_{6,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \cdots & T_{n-1,n-2} & T_{n-1,n-1} \end{bmatrix}. \tag{3.35}$$

A pair such as $T_{1,1}$ and $T_{2,1}$ does not commute with each other and any entry above the upper line such as $T_{1,3}$ anticommutes with an entry within the

lines, $T_{5,3}$. We note the upper entries such as $T_{1,3}$ are either I or some element that is fixed for each column. Let M be a product of rows. That is, $M = T^{s_0}_{\{0,\}} T^{s_1}_{\{1,\}} \cdots T^{s_{n-1}}_{\{n-1,\}}$ with some $(s_j)_j$. Checking each entry, we can inductively determine $(s_j)_j$. For example, if $M_{\{0\}} \neq \mathrm{I}$, $s_0 = 1$.

Now, we prove the proposition. Since the algorithm performs no operation except exchanges of rows and multiplication, $S \simeq S'$. If $S' = \bigotimes_{Q \in P} S'_{[Q]}$, S is P-separable by definition.

Suppose S is P-separable. That is, there exists $\left\{S_{[Q]}\right\}_{Q \in P}$ such that $S \simeq \bigotimes_{Q \in P} S_{[Q]}$. We claim that each row $S'_{\{i,\}}$ has a unique $Q \in P$ such that $S'_{\{i,j\}} \neq \mathrm{I}$ for some $j \in Q$. If that holds, the number of such rows are at most $|Q|$ for any Q. Since the sum of such numbers should be n, the number is indeed $|Q|$ and the statement holds. Take $S'_{\{i,\}}$ and $Q \in P$ such that $S'_{\{i,j\}} \neq \mathrm{I}$ for some $j \in Q$. If no Q exists, $S'_{\{i,j\}} = \mathrm{I}$ for any j, but it contradicts the independence unless $n = 1$. Define T as follows: $T_{\{j\}}$ is $S'_{\{i,j\}}$ if $j \in Q$ and otherwise I. Since T commutes with any row of $S_{[Q]}$, T is a product of rows of S'. By the first observation, $T = S'_{i,}$. □

3.2.3 Stabiliser domain

Now, it is time to extend the basis domain. Instead of an assignment to individual variables, we assign a stabiliser array to each block of a partition. It tells us a basis where the state of a block belongs unless the state is not a stabiliser state. Naturally, we have to care about non-stabiliser states. Indeed, QIL is universal and has a non-Clifford operator `T`. It means even if we start to run a QIL program from a stabiliser state, the state may grow to a non-stabiliser state, which we cannot express by any stabiliser array. We prepare a symbol ■ to denote such a non-stabiliser basis state. Precisely speaking, ■ does not mean that the state belongs to a non-stabiliser basis. It just claims that we do not have any information about the basis. We assume that ■ is an absorbing element with respect to unitary transformation and tensor products. That is, $U \blacksquare U^\dagger = \blacksquare$ for any unitary U and for any stabiliser arrays S, $\blacksquare \otimes S = \blacksquare$ and $S \otimes \blacksquare = \blacksquare$. Therefore, we say ■ commutes with anything.

Definition 3.2.19. Let $k \geq 1$. Define $\mathbf{SS}^*_k$ as $\mathbf{SS} \cup \{\blacksquare\}$ if $k > 1$ and as $\mathbf{SS} \cup \{\mathrm{I}, \blacksquare\}$ if $k = 1$. We define $\mathbf{NSS}^*_k$ in the same manner. $\mathbf{SS}^*$ and $\mathbf{NSS}^*$ are the unions of $\mathbf{SS}^*_k$ and $\mathbf{NSS}^*_k$, respectively. Let $A \subset \mathbf{Q}$. $\alpha \subset \mathcal{P}A \times \mathbf{SS}^*$ is a *stabiliser preassignment on* A if $\mathrm{pr}_0(\alpha)$ is a partition of A and for any $(Q, S) \in \alpha$, $S \in \mathbf{SS}^*_{|Q|}$. A stabiliser preassignment on A is said to be a *stabiliser assignment on* A if it is a subset of $\mathcal{P}A \times \mathbf{NSS}^*$. A subset of a stabiliser preassignment on A is a *stabiliser subpreassignment* and a subset of a stabiliser assignment on A is a *stabiliser subassignment*. The *stabiliser domain* $\mathbf{S^Q}$ is the set of stabiliser assignments on $\mathbf{Q}$. We omit to refer to A if it is clear from the context, especially, when $A = \mathbf{Q}$. We usually call a stabiliser assignment by an assignment.

Proposition 3.2.20. *A subpreassignment is a preassignment. A subassignment is an assignment.* □

Notation 3.2.21. Let α be a preassignment on A. We frequently identify a preassignment α with a function from A to $\mathcal{P}A \times \mathbf{SS}^*$. Specifically, $\alpha(i) = (Q, S)$ such that $i \in Q$ and $(Q, S) \in \alpha$. Furthermore, we define $\alpha_0 \colon A \to \mathcal{P}A$ and $\alpha_1 \colon A \to \mathbf{SS}^*$ by $\alpha_0(i) = Q$ and $\alpha_1(i) = S$ where $(Q, S) = \alpha(i)$. We sometimes use α_0 and α_1 to denote $\{\, Q \mid (Q, S) \in \alpha \,\}$ and $\{\, S \mid (Q, S) \in \alpha \,\}$, respectively. However, what α_0 means is always clear from the context.

We use the update notation. $\alpha[i \mapsto (Q', S')]$ is $(\alpha \setminus \{ \alpha(i) \}) \cup \{ (Q', S') \}$. When we use the notation, we always assume $(\alpha \setminus \{ \alpha(i) \}) \cup \{ (Q', S') \}$ is a preassignment. That is, Q' is equal to $\alpha_0(i)$ and S' belongs to $\mathbf{NSS}^*_{|Q'|}$. Since Q' is obvious, we often omit it. $\alpha[i \mapsto S']$ is $(\alpha \setminus \{ \alpha(i) \}) \cup \{ (\alpha_0(i), S') \}$. We repeatedly emphasise S' is assumed to belong to $\mathbf{NSS}^*_{|\alpha_0(i)|}$. The notation can be extended. $\alpha[i \mapsto (Q_0, S_0), \ldots, (Q_{n-1}, S_{n-1})]$ is $(\alpha \setminus \{ \alpha(i) \}) \cup \{ (Q_0, S_0), \ldots, (Q_{n-1}, S_{n-1}) \}$. $\alpha[i_0, \ldots, i_{m-1} \mapsto (Q', S')]$ is defined as $(\alpha \setminus \{ \alpha(i_0), \ldots, \alpha(i_{m-1}) \}) \cup \{ (Q', S') \}$. $\alpha[i_0 \mapsto (Q_0, S_0), \ldots, i_{l-1} \mapsto (Q_{l-1}, S_{l-1})]$ is the union of $\alpha \setminus \{ \alpha(i_0), \ldots, \alpha(i_{l-1}) \}$ and $\{ (Q_0, S_0), \ldots, (Q_{l-1}, S_{l-1}) \}$. We also omit to write blocks Q. A preassignment $\alpha[i_0, \ldots, i_{m-1} \mapsto S']$ is $\alpha[i_0, \ldots, i_{m-1} \mapsto (\bigcup_m \alpha_0(i_m), S')]$. $\alpha[i_0 \mapsto S_0, \ldots, i_{l-1} \mapsto S_{l-1}]$ is $\alpha[i_0 \mapsto (\alpha_0(i_0), S_0), \ldots, i_{l-1} \mapsto (\alpha_0(i_{l-1}), S_{l-1})]$.

We sometimes use a direct sum of matrices to denote a stabiliser preassignment. It has a header row to denote the associated block with each stabiliser array. For example, $\{ (\{ 0, 2 \}, \langle \mathrm{ZZ}, \mathrm{XX} \rangle), (\{ 1 \}, \langle \mathrm{Y} \rangle) \}$ is written as

$$\left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \mathrm{Z} & \mathrm{Z} & \\ \mathrm{X} & \mathrm{X} & \\ \hline & & \mathrm{Y} \end{array}\right]. \tag{3.36}$$

The stabiliser domain has an order. Intuitively speaking, α is smaller than β means α has more information. If they exist, the top has no information and the bottom is well-informed.

Definition 3.2.22. Let $S, T \in \mathbf{SS}^*$. A relation $\leq_{\mathbf{SS}^*}$ is defined by $S \leq_{\mathbf{SS}^*} T$ if and only if either $S = \mathrm{I}$, $S = T$, or $T = \blacksquare$. $\leq_{\mathbf{NSS}^*}$ is the restriction of $\leq_{\mathbf{SS}^*}$ on $\mathbf{NSS}^*$. Let $A \subset \mathbf{Q}$. Let α and β be stabiliser preassignments on A. A relation $\leq_{\mathbf{S}}$ is defined by $\alpha \leq_{\mathbf{S}} \beta$ if and only if $\alpha_0 \leq_{\mathbf{\Pi}} \beta_0$ and for any $i \in A$, $\bigodot_{j \in \beta_0(i)} \alpha(j) \leq_{\mathbf{SS}^*} \beta_1(i)$ where $\bigodot_{j \in \beta_0(i)} \alpha(j)$ is defined by the following.

$$\bigodot_{j \in Q} (Q_j, S_j) = \begin{cases} S_j & (\text{all } Q_j \text{ are the same}) \\ \mathrm{I} & (\text{all } S_j \text{ are I}) \\ \blacksquare & (\text{otherwise}) \end{cases} \tag{3.37}$$

We also use $\leq_{\mathbf{S}}$ to denote the restriction of the relation on $\mathbf{S}^{\mathbf{Q}}$.

Note that $Q_j = Q_k$ implies $S_j = S_k$. Therefore, $\bigodot$ is well-defined.

Proposition 3.2.23. *$\leq_{\boldsymbol{SS}^*}$, $\leq_{\boldsymbol{NSS}^*}$, and $\leq_{\mathbf{S}}$ are orders.*

Proof. We only prove transitivity of $\leq_{\mathbf{S}}$. The others are obvious. Let α, β, and γ be stabiliser preassigment on A such that $\alpha \leq_{\mathbf{S}} \beta \leq_{\mathbf{S}} \gamma$. By transitivity of $\leq_{\mathbf{\Pi}}$, $\alpha_0 \leq_{\mathbf{\Pi}} \gamma_0$. Take $i \in A$. Let $S = \bigodot_{j \in \gamma_0(i)} \alpha(j)$. If $S \neq \mathrm{I}, \blacksquare$, $\alpha_0(i)$ is equal to $\gamma_0(i)$ and thus to $\beta_0(i)$. By transitivity of $\leq_{\mathbf{SS}^*}$, $\alpha_1(i) \leq_{\mathbf{SS}^*} \gamma_1(i)$. Assume $S = \blacksquare$. If there exists $j \in \gamma_0(i)$ such that $\alpha_1(j) = \blacksquare$, then $\beta_1(j) = \gamma_1(j) = \blacksquare$. Otherwise, there exist $j, k \in \gamma_0(i)$ such that $\alpha_0(j) \neq \alpha_0(k)$ and $\alpha_1(j) \neq \mathrm{I}$. If $k \in \beta_0(j)$, $\beta_1(j) = \gamma_1(j) = \blacksquare$. If $k \notin \beta_0(j)$, $\gamma_1(i) = \blacksquare$. □

Proposition 3.2.24. *Let α and β be preassignments. $\alpha \leq_{\mathbf{S}} \beta$ if and only if for any $(R, T) \in \beta$, there exists a subpreassignment α' such that $\alpha' \leq_{\mathbf{S}} \{ (R, T) \}$.* □

The order gives a lattice structure to the stabiliser domain $\mathbf{S}^{\mathbf{Q}}$, which is needed to use the domain for giving an abstract semantics.

Proposition 3.2.25. *$(\mathbf{S}^{\mathbf{Q}}, \leq_{\mathbf{S}})$ is a complete lattice.*

Proof. We first note a finite lattice is automatically complete and $\mathbf{NSS}^*$ is trivially a lattice. The top and the bottom are $\{(\mathbf{Q}, \blacksquare)\}$ and $\{(\{i\}, \mathrm{I}) \mid i \in \mathbf{Q}\}$, respectively. Let α, β be assignments. Since the set of partitions is a lattice, we can obtain the join and meet of α_0 and β_0. Let J and M be the join and the meet. Take $Q \in J$. Define S as the join of $\bigodot_{j \in Q} \alpha(j)$ and $\bigodot_{j \in Q} \beta(j)$. By construction, the set of (Q, S) is the join of α and β. Let γ be $\emptyset$. Take $Q \in M$. Let $(A, T) \in \alpha$ and $(B, R) \in \beta$ be such that $Q \subset A \cap B$. If $(A, T) = (B, R)$, add (A, T) to γ. If $Q = A$ and $R = \blacksquare$, add (A, T) to γ. If $Q = B$ and $T = \blacksquare$, add (B, R) to γ. If $T = R = \blacksquare$, add $(Q, \blacksquare)$ to γ. If none holds, add $(\{i\}, \mathrm{I})$ to γ for any $i \in Q$. Repeating the process for each $Q \in M$, γ finally becomes the meet of α and β. □

We gave an intuitive explanation of the stabiliser domain: how an abstract state approximates a concrete state. Now, we give a formal definition. We follow the definition of the basis domain but impose a more strict restriction.

Definition 3.2.26. Let $A \subset \mathbf{Q}$ and α be a preassignment on A. α is *sound* for $\{\rho_{Q,S}\}_{(Q,S)\in\alpha}$ if it satisfies the following: $\rho_{Q,S}$ is a quantum state on Q; $\rho_{Q,S}$ is $\frac{1}{2}\mathrm{I}$ when $S = \mathrm{I}$ and if $S \in \mathbf{SS}$, $\rho_{Q,S} = |\psi\rangle\langle\psi|$ where $|\psi\rangle$ is an S basis state.

α is said to be a *sound approximation of* a quantum state ρ, denoted by $\alpha \vDash_{\mathbf{S}} \rho$, if there exist $\left\{(p_j, \{\rho_{j,Q,S}\}_{(Q,S)\in\alpha})\right\}_j$ such that $0 \le p_j \le 1$, $\sum_j p_j \le 1$, α is sound for each $\{\rho_{j,Q,S}\}_{(Q,S)\in\alpha}$, and

$$\rho = \sum_j p_j \bigotimes_{(Q,S)\in\alpha} \rho_{j,Q,S}. \tag{3.38}$$

We say such $\left\{(p_j, \{\rho_{j,Q,S}\}_{(Q,S)\in\alpha})\right\}_j$ is a *sound decomposition of* ρ *for* α.

Example 3.2.27.

$$\left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \mathrm{Z} & \mathrm{Z} & \\ \mathrm{X} & \mathrm{X} & \\ \hline & & \mathrm{Z} \end{array}\right] \vDash_{\mathbf{S}} \frac{1}{2}|000\rangle\langle 000| + \frac{1}{2}|101\rangle\langle 101| + \frac{1}{2}|000\rangle\langle 101| + \frac{1}{2}|101\rangle\langle 000| \tag{3.39}$$

By definition, the following propositions obviously hold.

Proposition 3.2.28. *Let α be a preassignment. α is sound for $\{\rho_{Q,S}\}_{(Q,S)\in\alpha}$ if and only if for any $(Q, S) \in \alpha$, (Q, S) is sound for $\rho_{Q,S}$.* □

Proposition 3.2.29. *Let α be a preassignment and ρ, σ be quantum states such that the sum is a quantum state. If α is a sound approximation of both ρ and σ, it is also a sound approximation of $\rho + \sigma$.* □

An intuition about the order of the domain is justified by the following proposition. Indeed, the order $\le_{\mathbf{S}}$ was defined so that the proposition holds.

Proposition 3.2.30. *Let ρ be a state and α and β be preassignments. Assume $\alpha \le_{\mathbf{S}} \beta$. If α is a sound approximation of ρ, so is β.*

Proof. Let $\left\{(p_j, \{\rho_{j,Q,S}\})\right\}$ be a sound decomposition of ρ for α. Take $(R, T) \in \beta$. Since $\alpha \le_{\mathbf{S}} \beta$, there exist $(Q_0, S_0), \ldots, (Q_{n-1}, S_{n-1}) \in \alpha$ such that $\bigcup_k Q_k = R$ and $\bigodot_k (Q_k, S_k) \le_{\mathbf{S}} T$. If T is either I or $\blacksquare$, (R, T) is sound for $\bigotimes_k \rho_{j,Q_k,S_k}$. If $T \in \mathbf{SS}$, either $n = 1$ or all S_ks are I. In the latter case, $\frac{1}{2^n}\mathrm{I}^{\otimes n}$ can be decomposed to a sum of the T basis states. □

Proposition 3.2.31. *Let α and β be assignments. Assume that for any state ρ, if α is a sound approximation of ρ, so is β. Then, $\alpha \leq_{\mathbf{S}} \beta$.*

Proof. For each n, fix a non-stabiliser entangled state $|\psi_n\rangle$ of n qubits. Moreover, for each stabiliser array S, fix a stabiliser state $|\psi_S\rangle$. Define ρ as follows

$$\rho = \bigotimes_{Q \in \alpha_0} \sigma_{[Q]} \tag{3.40}$$

where

$$\sigma_{[Q]} = \begin{cases} \frac{1}{2}\mathrm{I} & ((Q, \mathrm{I}) \in \alpha) \\ |\psi_S\rangle\langle\psi_S| & ((Q, S) \in \alpha) \\ |\psi_n\rangle\langle\psi_n| & (\text{otherwise}) \end{cases} . \tag{3.41}$$

α is trivially a sound approximation of ρ. By assumption, β is a sound approximation of ρ. ρ is not P-separable unless $\alpha_0 \leq_{\mathbf{\Pi}} P$. Hence, $\alpha_0 \leq_{\mathbf{\Pi}} \beta_0$. Moreover, since α is an assignment, any $\sigma_{[Q]}$ is entangled. Therefore, $\alpha \leq_{\mathbf{S}} \beta$. □

3.2.4 Stabiliser abstract semantics

In the previous subsection, we investigated the stabiliser domain. Now, we have no obstacle to an abstract semantics.

Definition 3.2.32. The *stabiliser abstract semantics* of QIL programs is a function $[\![\cdot]\!]_{\mathbf{S}} \colon \mathbf{QIL} \to \mathbf{S^Q} \to \mathbf{S^Q}$ defined in Figure 3.3. $\mathrm{up}_{\mathbf{S}}$ is a function that decomposes stabiliser arrays so that an output is an assignment. $\mathrm{up}_{\mathbf{S}}(Q, \blacksquare) = \{\, (Q, \blacksquare) \,\}$ and $\mathrm{up}_{\mathbf{S}}(Q, S) = \{\, (Q_0, S_0), \ldots, (Q_{m-1}, S_{m-1}) \,\}$ such that $\{Q_i\}_i$ is a partition of Q, $S_i \in \mathbf{NSS}_{|Q_i|}$, and $\bigotimes_i S_i = S$. $\mathrm{meas}_{\mathbf{S},i}$ performs measurement and is defined as follows.

$$\mathrm{meas}_{\mathbf{S},i}(\alpha) = \begin{cases} \alpha[i \mapsto \langle \mathrm{Z} \rangle] & (|\alpha_0(i)| = 1) \\ \alpha[i \mapsto (\{\, i \,\}, \langle \mathrm{Z} \rangle), (\alpha_0 \setminus \{\, i \,\}, \blacksquare)] & (\alpha_1(i) = \blacksquare) \\ \alpha[i \mapsto \mathrm{up}_{\mathbf{S}}(\alpha_0(i), \mathrm{meas}_{\mathrm{sf},i}(\alpha_1(i)))] & (\text{otherwise}) \end{cases} \tag{3.42}$$

where $\mathrm{meas}_{\mathrm{sf},i}$ is the measurement of the ith qubit in the stabiliser formalism.

Proposition 3.2.33. $[\![\cdot]\!]_{\mathbf{S}}$ *is well-defined.*

Proof. We only prove that $S = \mathrm{CX}_{(i,j)}(\alpha_1(i) \otimes \alpha_1(j))\mathrm{CX}^{\dagger}_{(i,j)}$ belongs to $\mathbf{NSS}^*$. Assume S is separable: $S = T \otimes U$. If T contains neither i nor j, any row of T is unchanged through $\mathrm{CX}_{(i,j)}$ and therefore T is an evidence of either or both of $\alpha_1(i)$ and $\alpha_1(j)$ are separable. Assume T and U contains i and j, respectively. By the definition of $[\![\cdot]\!]_{\mathbf{S}}$, the ith column of $\alpha_0(i)$ contains X or Y. Therefore, S has a row whose ith and jth columns are (X, X) or (Y, X). Because $\alpha_1(j) \neq \langle \mathrm{X} \rangle$, S has no row whose ith and jth columns are (I, X). It contradicts $S = T \otimes U$. □

The stabiliser semantics is not so different from the basis semantics. A different point is that the stabiliser semantics of `CX(i,j)` has an extra condition: $\alpha_0(i) = \alpha_0(j)$. This condition means that i and j are entangled before being applied $\mathrm{CX}_{(i,j)}$. For this case, while the basis semantics cannot do anything, the stabiliser semantics is able to undo the entanglement via $\mathrm{up}_{\mathbf{S}}$. It is worth noting that we use not a preassignment but an assignment. This is because the behaviour of $[\![\texttt{T(}i\texttt{)}]\!]_{\mathbf{S}}$. It discards the stabiliser array on the associated block. Therefore, if a block is meaninglessly large, $[\![\texttt{T(}i\texttt{)}]\!]_{\mathbf{S}}$ removes too much information. Using $\mathrm{up}_{\mathbf{S}}$, we reduce the damage from a non-Clifford operator.

Now, we show the first main theorem of the section: the stabiliser semantics is sound.

$$
\begin{aligned}
[\![\texttt{skip}]\!]_{\mathbf{S}}(\alpha) &= \alpha \\
[\![C\,;C']\!]_{\mathbf{S}}(\alpha) &= [\![C']\!]_{\mathbf{S}}([\![C]\!]_{\mathbf{S}}(\alpha)) \\
[\![U\,\texttt{(}i\texttt{)}]\!]_{\mathbf{S}}(\alpha) &= \alpha[i \mapsto U_{(i)}\alpha_1(i)U_{(i)}^\dagger] \\
[\![\texttt{T(}i\texttt{)}]\!]_{\mathbf{S}}(\alpha) &= \begin{cases} \alpha & (\alpha_1(i) \text{ and } \mathrm{Z}_{(i)} \text{ commute}) \\ \alpha[i \mapsto \blacksquare] & (\text{otherwise}) \end{cases} \\
[\![\texttt{CX(}i\texttt{,}j\texttt{)}]\!]_{\mathbf{S}}(\alpha) &= \begin{cases} \alpha[i \mapsto \mathrm{up}_{\mathbf{S}}(\alpha_0(i), \mathrm{CX}_{(i,j)}\alpha_1(i)\mathrm{CX}_{(i,j)}^\dagger)] & (\alpha_0(i) = \alpha_0(j)) \\ \alpha & (\alpha_1(i) = \mathrm{I} \text{ and } \alpha_1(j) = \mathrm{I}) \\ \alpha[i \mapsto \langle \mathrm{Z} \rangle] & (\alpha_1(i) = \mathrm{I}) \\ \alpha[j \mapsto \langle \mathrm{X} \rangle] & (\alpha_1(j) = \mathrm{I}) \\ \alpha & (\alpha_1(i) = \langle \mathrm{Z} \rangle \text{ or } \alpha_1(j) = \langle \mathrm{X} \rangle) \\ \alpha[i, j \mapsto \mathrm{CX}_{(i,j)}(\alpha_1(i) \otimes \alpha_1(j))\mathrm{CX}_{(i,j)}^\dagger] & (\text{otherwise}) \end{cases}
\end{aligned}
$$

$$
\left[\!\!\left[\begin{array}{l} \texttt{if } i \\ \quad \texttt{then } C \\ \quad \texttt{else } C' \\ \texttt{fi} \end{array} \right]\!\!\right]_{\mathbf{S}} (\alpha) = [\![C]\!]_{\mathbf{S}}(\mathrm{meas}_{\mathbf{S},i}(\alpha)) \vee [\![C']\!]_{\mathbf{S}}(\mathrm{meas}_{\mathbf{S},i}(\alpha))
$$

$$
\left[\!\!\left[\begin{array}{l} \texttt{while } i \\ \texttt{do} \\ \quad C \\ \texttt{od} \end{array} \right]\!\!\right]_{\mathbf{S}} (\alpha) = \bigvee_{n \in \mathbb{N}} \mathrm{meas}_{\mathbf{S},i}(([\![C]\!]_{\mathbf{S}} \circ \mathrm{meas}_{\mathbf{S},i})^n(\alpha))
$$

where $U \in \{ \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{H}, \mathtt{S} \}$.

Figure 3.3: Stabiliser abstract semantics of quantum imperative language

Lemma 3.2.34. *Let ρ be a concrete state and α be an assignment such that $\alpha \vDash_{\mathbf{S}} \rho$. For any $i \in \mathbf{Q}$, $d \in \mathbb{B}$, $\mathrm{meas}_{\mathbf{S},i}(\alpha)$ is a sound approximation of $|d\rangle\langle d|_{(i)}\rho|d\rangle\langle d|_{(i)}$.*

Proof. Let $\{(p_j, \{\rho_{j,Q,S}\})\}$ be a sound decomposition of ρ for α. Let (Q, S) be $\alpha(i)$. If $|Q| = 1$, $|d\rangle\langle d|_{(i)}\rho_{j,Q,S}|d\rangle\langle d|_{(i)} = |d\rangle\langle d|$ and therefore $(Q, \langle \mathrm{Z} \rangle)$ is sound for it. If $S = \blacksquare$, (Q, S) is changed into $\{ (\{ i \}, \langle \mathrm{Z} \rangle), (Q \setminus \{ i \}, \blacksquare) \}$ and the latter element is sound for $\langle d|_{(i)}\rho_{j,Q,S}|d\rangle_{(i)}$. Assume S is a stabiliser array. $\rho_{j,Q,S} = |\psi\rangle\langle\psi|$ with some S basis state $|\psi\rangle$. The stabiliser formalism ensures $\mathrm{meas}_{\mathrm{sf},i}(S)$ stabilises $|0\rangle\langle 0|_{(i)}|\psi\rangle$. □

Theorem 3.2.35. *For any concrete state $\rho \in \mathcal{D}_N$, any assignment $\alpha \in \mathbf{S}^{\mathbf{Q}}$, and any QIL program $C \in \mathbf{QIL}$, $\alpha \vDash_{\mathbf{S}} \rho$ implies $[\![C]\!]_{\mathbf{S}}(\alpha) \vDash_{\mathbf{S}} [\![C]\!](\rho)$.*

Proof. We prove the statement by the induction on the structure of C. By Proposition 3.2.28, we ignore the unchanged blocks. For `skip` and $C\,;C'$, the statement trivially holds. Take α and ρ such that $\alpha \vDash_{\mathbf{S}} \rho$. Let $\{(p_k, \{\rho_{k,Q,S}\})\}$ be a sound decomposition of ρ for α.

- (U) Take $U \in \{ \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{H}, \mathtt{S} \}$. Let (Q, S) be $\alpha(i)$. $(Q, U_{(i)}SU_{(i)}^\dagger)$ is sound for $U_{(i)}\rho_{k,Q,S}U_{(i)}^\dagger$. If $U_{(i)}SU_{(i)}^\dagger$ is I, so is S and hence $\rho_{k,Q,S} = U_{(i)}\rho_{k,Q,S}U_{(i)}^\dagger = \frac{1}{2}\mathrm{I}$. Moreover, if $|\psi\rangle$ is an S basis state, $U|\psi\rangle$ is a $U_{(i)}SU_{(i)}^\dagger$ basis state.
- (`T`) By the same argument as the above, $[\![\texttt{T(}i\texttt{)}]\!]_{\mathbf{S}}(\alpha) \vDash_{\mathbf{S}} [\![\texttt{T(}i\texttt{)}]\!](\rho)$. Note that $\blacksquare$ is sound for anything.

(`CX`) We will show $[\![\texttt{CX}(i,j)]\!]_{\mathbf{S}}(\alpha) \vDash_{\mathbf{S}} [\![\texttt{CX}(i,j)]\!](\rho)$ by case analysis.

- Assume $\alpha_0(i) = \alpha_0(j)$. Let (Q,S) be $\alpha(i)$. If $S = \blacksquare$, (Q,S) is trivially sound for $\mathrm{CX}_{(i,j)}\rho_{k,Q,S}\mathrm{CX}^\dagger_{(i,j)}$. Assume $S \neq \blacksquare$. $\mathrm{CX}_{(i,j)}S\mathrm{CX}^\dagger_{(i,j)}$ is sound for $\mathrm{CX}_{(i,j)}\rho_{k,Q,S}\mathrm{CX}^\dagger_{(i,j)}$. Since $|Q| > 1$, $S \in \mathbf{NSS}$, and $\rho_{k,Q,S}$ is an S basis state. $\mathrm{CX}_{(i,j)}\rho_{k,Q,S}\mathrm{CX}^\dagger_{(i,j)}$ is a $\mathrm{CX}_{(i,j)}S\mathrm{CX}^\dagger_{(i,j)}$ basis state and it can be decomposed into appropriate stabiliser basis states by Proposition 3.2.15.
- Assume $\alpha_0(i) \neq \alpha_0(j)$. Let (Q,S) and (R,T) be $\alpha(i)$ and $\alpha(j)$, respectively.
  * Assume $S = T = \mathrm{I}$. $\rho_{k,Q,S} = \rho_{k,R,T} = \frac{1}{2}\mathrm{I}$. $\mathrm{CX}(\frac{1}{4}\mathrm{I}^{\otimes 2})\mathrm{CX}^\dagger = \frac{1}{4}\mathrm{I}^{\otimes 2}$.
  * Assume $S = \mathrm{I}$ but $T \neq \mathrm{I}$. Then, $\rho_{k,Q,S} = \frac{1}{2}\mathrm{I}$ and
  $$\begin{aligned}&\mathrm{CX}_{(i,j)}\left(\frac{1}{2}\mathrm{I}\otimes\rho_{k,R,T}\right)\mathrm{CX}^\dagger_{(i,j)}\\ &= \frac{1}{2}|0\rangle\langle 0|_{[i]}\otimes\rho_{k,R,T} + \frac{1}{2}|1\rangle\langle 1|_{[i]}\otimes \mathrm{X}_{(j)}\rho_{k,R,T}\mathrm{X}^\dagger_{(j)}.\end{aligned} \tag{3.43}$$
  Since any stabiliser basis is invariant under the transformation by $\mathrm{X}_{(j)}$, (R,T) is sound for both $\rho_{k,R,T}$ and $\mathrm{X}_{(j)}\rho_{k,R,T}\mathrm{X}^\dagger_{(j)}$.
  * Assume $S = \langle \mathrm{Z}\rangle$ and $T \neq \mathrm{I}$. We can write $\rho_{k,Q,S} = |d\rangle\langle d|$ where $d \in \mathbb{B}$.
  $$\mathrm{CX}_{(i,j)}(|d\rangle\langle d|\otimes\rho_{k,R,T})\mathrm{CX}^\dagger_{(i,j)} = |d\rangle\langle d|_{[i]}\otimes {\mathrm{X}_{(j)}}^d\rho_{k,R,T}{\mathrm{X}_{(j)}}^{d\dagger}. \tag{3.44}$$
  It is the same as the above.
  * Finally, assume $S \neq \mathrm{I}, \langle \mathrm{Z}\rangle$ and $T \neq \mathrm{I}, \langle \mathrm{X}\rangle$. $(Q\cup R, S\otimes T)$ is trivially sound for $\mathrm{CX}_{(i,j)}\rho_{k,Q,S}\otimes\rho_{k,R,T}\mathrm{CX}^\dagger_{(i,j)}$.

(`if`) Let C, C' be QIL programs. By Lemma 3.2.34 and the induction hypothesis, $[\![C]\!]_{\mathbf{S}}(\mathrm{meas}_{\mathbf{S},i}(\alpha))$ and $[\![C']\!]_{\mathbf{S}}(\mathrm{meas}_{\mathbf{S},i}(\alpha))$ are sound approximations for $[\![C]\!](|0\rangle\langle 0|_{(i)}\rho|0\rangle\langle 0|_{(i)})$ and $[\![C']\!](|1\rangle\langle 1|_{(i)}\rho|1\rangle\langle 1|_{(i)})$, respectively. Then, $[\![\texttt{if}\ i\ \texttt{then}\ C\ \texttt{else}\ C'\ \texttt{fi}]\!]_{\mathbf{S}} \vDash_{\mathbf{S}} [\![\texttt{if}\ i\ \texttt{then}\ C\ \texttt{else}\ C'\ \texttt{fi}]\!]$ is a consequence of Propositions 3.2.29 and 3.2.30.

(`while`) Let $i \in \mathbf{Q}$ and $C \in \mathbf{QIL}$. By Lemma 3.2.34 and the induction hypothesis, for any $n \in \mathbb{N}$, $\mathrm{meas}_{\mathbf{S},i}(([\![C]\!]_{\mathbf{S}} \circ \mathrm{meas}_{\mathbf{S},i})^n(\alpha))$ is a sound approximation of $|1\rangle\langle 1|_{(i)}\mathrm{meas}^n_{C,i}(\rho)|1\rangle\langle 1|_{(i)}$. Therefore, for any $n \in \mathbb{N}$, $[\![\texttt{while}\ i\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{S}}(\alpha)$ is a sound approximation of $|1\rangle\langle 1|_{(i)}\mathrm{meas}^n_{C,i}(\rho)|1\rangle\langle 1|_{(i)}$. By Proposition 3.2.29, for any $N \in \mathbb{N}$, this stabiliser abstract state is also a sound approximation of $\sum_{n\le N}|1\rangle\langle 1|_{(i)}\mathrm{meas}^n_{C,i}(\rho)|1\rangle\langle 1|_{(i)}$. Therefore, the sum can be written as $\sum_j p_{j,N}\bigotimes\rho_{j,Q,S}\otimes\sigma_{j,N}$. Here, $(Q,S) \in [\![\texttt{while}\ i\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{S}}(\alpha)$ is sound for $\rho_{j,Q,S}$, $\bigotimes\rho_{j,Q,S}$ and $\bigotimes\rho_{k,Q,S}$ are assumed to be orthogonal unless $j = k$, and $\sigma_{j,N}$ is $\{\, Q \mid (Q,\blacksquare) \in \beta \,\}$-separable. Since the state converges and projections are continuous, $p_{j,N}$ and $\sigma_{j,N}$ converges when $N \to \infty$. The set of separable states is closed, so the limit of $\sigma_{j,N}$ is separable. Therefore, $[\![\texttt{while}\ i\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{S}}(\alpha)$ is a sound approximation of $[\![\texttt{while}\ i\ \texttt{do}\ C\ \texttt{od}]\!](\rho)$.

□

The soundness theorem states that we can use the stabiliser semantics to analyse entanglement in QIL programs. The theorem guarantees nothing but

that the semantics never gives wrong analysis, which the basis semantics is also guaranteed by Theorem 3.1.16. We show that the stabiliser semantics has an advantage over the basis semantics by computing the stabiliser semantics of our motivating examples.

Example 3.2.36. $[\![\texttt{GHZ}]\!]_{\mathbf{S}}(\alpha)$ is $\{\,(\{\,0,1,2\,\},\langle \mathrm{XXX},\mathrm{ZZI},\mathrm{IZZ}\rangle)\,\}$. Both abstract states $[\![\texttt{INVERTGHZ}]\!]_{\mathbf{S}}(\alpha), [\![\texttt{BREAKGHZ}]\!]_{\mathbf{S}}(\alpha)$ are $\{\,(\{\,0\,\},\langle \mathrm{Z}\rangle),(\{\,1\,\},\langle \mathrm{Z}\rangle),(\{\,2\,\},\langle \mathrm{Z}\rangle)\,\}$. Indeed,

$$\alpha \xrightarrow{\texttt{INIT}} \left[\begin{array}{c|c|c} 0 & 1 & 2 \\ \hline \mathrm{Z} & & \\ \hline & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{H(0)}} \left[\begin{array}{c|c|c} 0 & 1 & 2 \\ \hline \mathrm{X} & & \\ \hline & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right]$$
$$\xrightarrow{\texttt{CX(1,2)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(1,2)}} \left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \tag{3.45}$$
$$\xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{c|c|c} 0 & 1 & 2 \\ \hline \mathrm{X} & & \\ \hline & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{H(0)}} \left[\begin{array}{c|c|c} 0 & 1 & 2 \\ \hline \mathrm{Z} & & \\ \hline & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right]$$

$$\alpha \xrightarrow{\texttt{GHZ}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{MEASURE}_0} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{Z} & \mathrm{I} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{I} & \mathrm{Z} \end{array}\right] \to \left[\begin{array}{c|c|c} 0 & 1 & 2 \\ \hline \mathrm{Z} & & \\ \hline & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \tag{3.46}$$

In the last arrow, we explicitly write how $\mathrm{up}_{\mathbf{S}}$ works.

The semantics shows all variables are separable after running `INVERTGHZ` or `BREAKGHZ`, which we learned from the concrete semantics. The traces in the examples show how the stabiliser semantics nicely works. Then, a question arises. The above example tells us that the stabiliser semantics is better than the basis semantics in some cases. Is that true for any case? We will make a comparison in the next subsection.

Before proceeding the next subsection, we must confess a drawback of the stabiliser semantics: It is not monotone, although that the basis abstract semantics is monotone as stated in Lemma 3.1.15. Worse than that, we have a specific program that inverts the order of some specific inputs.

Proposition 3.2.37. $[\![\cdot]\!]_{\mathbf{S}}$ *is not monotone. Furthermore, there exist assignments* α, β *and a QIL program* C *such that* $\alpha <_{\mathbf{S}} \beta$ *and* $[\![C]\!]_{\mathbf{S}}(\alpha) >_{\mathbf{S}} [\![C]\!]_{\mathbf{S}}(\beta)$.

Proof. Let $\alpha = \{\,(\{\,0\,\},\mathrm{I}),(\{\,1\,\},\langle \mathrm{Z}\rangle)\,\}$, $\beta = \{\,(\{\,0\,\},\langle \mathrm{X}\rangle),(\{\,1\,\},\langle \mathrm{Z}\rangle)\,\}$, and $C =$

`CX(0,1);S(0);H(0);CX(0,1);T(1)`. Then,

$$\begin{aligned}
&\left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{I} & \\ \hline & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{Z} & \\ \hline & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{S(0)}} \left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{Z} & \\ \hline & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{H(0)}} \left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{X} & \\ \hline & \mathrm{Z} \end{array}\right] \\
&\xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{cc} 0 & 1 \\ \hline \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{T(1)}} \left[\begin{array}{cc} 0 & 1 \\ \hline \blacksquare & \blacksquare \end{array}\right]
\end{aligned} \tag{3.47}$$

$$\begin{aligned}
&\left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{X} & \\ \hline & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{cc} 0 & 1 \\ \hline \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{S(0)}} \left[\begin{array}{cc} 0 & 1 \\ \hline \mathrm{Y} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{H(0)}} \left[\begin{array}{cc} 0 & 1 \\ \hline \mathrm{Y} & \mathrm{X} \\ \mathrm{X} & \mathrm{Z} \end{array}\right] \\
&\xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{Y} & \\ \hline & \mathrm{Y} \end{array}\right] \xrightarrow{\texttt{T(1)}} \left[\begin{array}{c|c} 0 & 1 \\ \hline \mathrm{Y} & \\ \hline & \blacksquare \end{array}\right].
\end{aligned} \tag{3.48}$$

□

This is very troublesome. Until now, we have not referred to `while` i `do` C `od`. The stabiliser semantics of the program needs to compute the least upper bound of finite approximations. As the stabiliser domain $\mathbf{S}^{\mathbf{Q}}$ is finite, the computation necessarily terminates. However, it may take superpolynomial time because $\mathbf{S}^{\mathbf{Q}}$ has the superpolynomial number of elements. If the semantics is monotone, we could compute an upper approximation, the least fixed point of $\boldsymbol{\lambda}\beta.(\mathrm{meas}_{\mathbf{S},i}(\llbracket C \rrbracket_{\mathbf{S}}(\beta)) \vee \mathrm{meas}_{\mathbf{S},i}(\alpha))$. Starting from the bottom, the function ascends the stabiliser domain. The height of the domain grows linearly with respect to N, so we could compute it in polynomial time.

Let us investigate the program in the above proposition more precisely. Comparing (3.47) to (3.48), we can find the first $\xrightarrow{\texttt{CX(0,1)}}$ breaks monotonicity. Before passing the operator, the former says the state of the zeroth qubit is $\frac{1}{2}\mathrm{I}$, but the latter says it is a $\langle \mathrm{X} \rangle$ stabiliser state. Then, the former remains separable but the latter becomes entangled. This situation causes the trouble. Indeed, we can find that no other situation destroys the monotonicity.

Lemma 3.2.38. *Let α be an assignment and C be a QIL program. For any $i \in \mathbf{Q}$, if $(\llbracket C \rrbracket_{\mathbf{S}}(\alpha))_1(i) = \mathrm{I}$, then $\alpha_1(i) = \mathrm{I}$.*

Proof. By the induction on the structure of C. Any program does not produce I. Note that $USU^\dagger = \mathrm{I}$ if and only if $S = \mathrm{I}$. □

Corollary 3.2.39. *Let $I(\alpha)$ be $|\{\, Q \mid (Q, \mathrm{I}) \in \alpha \,\}|$. For any $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $C \in$ **QIL**, $I(\alpha) \geq I(\llbracket C \rrbracket_{\mathbf{S}}(\alpha))$.* □

Lemma 3.2.40. $\mathrm{up}_{\mathbf{S}}$ *is monotone.*

Proof. Let α and $\{\, (R, T) \,\}$ be preassignments such that $\alpha \leq_{\mathbf{S}} \{\, (R, T) \,\}$. We claim $\bigcup_{(Q,S)\in\alpha} \mathrm{up}_{\mathbf{S}}(Q, S) \leq_{\mathbf{S}} \mathrm{up}_{\mathbf{S}}(R, T)$. If $T = \blacksquare$, then $\mathrm{up}_{\mathbf{S}}(R, T) = (R, \blacksquare)$. If $\bigodot \mathrm{up}_{\mathbf{S}}(Q_i, S_i)$ is I, then $\mathrm{up}_{\mathbf{S}}(Q_i, S_i)$ is $\{\, (Q_i, \mathrm{I}) \,\}$. Otherwise, $\alpha = \{\, (R, T) \,\}$. □

Lemma 3.2.41. $\mathrm{meas}_{\mathbf{S},i}$ *is monotone.*

Proof. Let α, β be assignments such that $\alpha \leq_{\mathbf{S}} \beta$. We first note that both $\mathrm{meas}_{\mathbf{S},i}(\alpha)$ and $\mathrm{meas}_{\mathbf{S},i}(\beta)$ definitely contain $(\{\, i \,\}, \langle \mathrm{Z} \rangle)$. If $\beta_1(i) = \blacksquare$, then $(\mathrm{meas}_{\mathbf{S},i}(\beta))_1(j) = \blacksquare$ for any $j \in \beta_0(i) \setminus \{\, i \,\}$. Moreover, when $\bigodot_{j\in\beta_0(i)} \alpha(j) = \mathrm{I}$, $\bigodot_{j\in\beta_0(i)\setminus\{\, i \,\}} \alpha(j)$ remains I. □

Proposition 3.2.42. *Let α, β be assignments such that $\alpha \leq_{\mathbf{S}} \beta$. Let C be a QIL program. Assume for any $i \in \mathbf{Q}$, $\alpha_1(i) = \mathrm{I}$ implies $\beta_1(i) = \blacksquare$. Then, $[\![C]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} [\![C]\!]_{\mathbf{S}}(\beta)$.*

Proof. Note that $[\![C]\!]_{\mathbf{S}}(\alpha)$ and $[\![C]\!]_{\mathbf{S}}(\beta)$ satisfy the assumption because $\blacksquare$ is preserved unless $\mathrm{meas}_{\mathbf{S},i}$ is applied. When it is applied, both $\alpha_1(i)$ and $\beta_1(i)$ change to $\langle \mathrm{Z} \rangle$.

We prove the statement by the induction on the structure of C. By Lemma 3.2.38, it is easy to see `skip` and C`;`C' satisfies the statement. Let α, β be assignments such that $\alpha \leq_{\mathbf{S}} \beta$.

- Let $U \in \{ \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{H}, \mathtt{S} \}$. $[\![U\,\mathtt{(}i\mathtt{)}]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} [\![U\,\mathtt{(}i\mathtt{)}]\!]_{\mathbf{S}}(\beta)$ from the fact that U preserves I and $\blacksquare$. We can apply a similar argument to `T(`i`)`.
- By monotonicity of $\mathrm{meas}_{\mathbf{S},i}$ and the induction hypothesis, the statement holds for `if` i `then` C `else` C' `fi` and `while` i `do` C `od`.
- Finally, we examine `CX(`i`,`j`)`. If $\alpha_0(i) = \alpha_0(j)$, so is β. By monotonicity of $\mathrm{up}_{\mathbf{S}}$, $[\![\mathtt{CX(}i\mathtt{,}j\mathtt{)}]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} [\![\mathtt{CX(}i\mathtt{,}j\mathtt{)}]\!]_{\mathbf{S}}(\beta)$. Assume $\alpha_0(i) \neq \alpha_0(j)$. If neither $\alpha_1(i)$ nor $\alpha_1(j)$ is I, then either $(\alpha_1(i), \alpha_1(j)) = (\beta_1(i), \beta_1(j))$ or at least one of $\{ \beta_1(i), \beta_1(j) \}$ is $\blacksquare$. If both of $\alpha_1(i)$ and $\alpha_1(j)$ is I, then $\bigodot_{k \in \beta_0(i)} \alpha(i)$ is I or $\blacksquare$. The proposition trivially holds. Finally, assume $\alpha_1(i) = \mathrm{I}$ and $\alpha_1(j) \neq \mathrm{I}$. By the assumption, $\beta_1(i) = \blacksquare$. If $\beta_1(j) \neq \langle \mathrm{X} \rangle$, $([\![\mathtt{CX(}i\mathtt{,}j\mathtt{)}]\!]_{\mathbf{S}}(\beta))_1(j) = \blacksquare$. Otherwise, $\alpha_1(j) = \langle \mathrm{X} \rangle$.

□

In the above proof, we used the assumption "for any $i \in \mathbf{Q}$, $\alpha_1(i) = \mathrm{I}$ implies $\beta_1(i) = \blacksquare$" only when $C =$ `CX(`i`,`j`)` and exactly one of $\alpha_1(i)$ and $\alpha_1(j)$ is I. Indeed, (3.47) is such a situation. Therefore, we can say the stabiliser semantics is monotone when it does not happen.

Corollary 3.2.43. *Let α, β be assignments such that $\alpha \leq_{\mathbf{S}} \beta$. Let C be a QIL program. Assume $I(\alpha) = I([\![C]\!]_{\mathbf{S}}(\alpha))$. Then, $[\![C]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} [\![C]\!]_{\mathbf{S}}(\beta)$.* □

By the structural induction, we can find that $I(\alpha) = I([\![C]\!]_{\mathbf{S}}(\alpha))$ if and only if for any i such that $\alpha_1(i) = \mathrm{I}$, neither i is measured nor C contains `CX(`i`,`j`)` with j such that $\alpha_1(j) \neq \mathrm{I}$. Therefore, if we run the program again, the number of I is preserved.

Proposition 3.2.44. *Let α be an assignment and C be a QIL program. Assume that if C contains* `CX(`i`,`j`)`*, either $\alpha_1(i) = \alpha_1(j) = \mathrm{I}$ or $\alpha_1(i) \neq \mathrm{I}$ and $\alpha_1(j) \neq \mathrm{I}$. Moreover, assume that if C contains* `if` i `then` C' `else` C'' `fi` *or* `while` i `do` C' `od`*, $\alpha_1(i) \neq \mathrm{I}$. Then, $I(\alpha) = I([\![C]\!]_{\mathbf{S}}(\alpha))$.* □

Corollary 3.2.45. *Let α, β be assignments such that $\alpha_1(i) = \mathrm{I}$ if and only if $\beta_1(i) = \mathrm{I}$ for any $i \in \mathbf{Q}$. Let C be a QIL program. Assume $I([\![C]\!]_{\mathbf{S}}(\alpha))$ is $I(\alpha)$. Then, $I([\![C]\!]_{\mathbf{S}}(\beta))$ is $I(\beta)$.* □

Corollary 3.2.46. *Let α be an assignment and C be a QIL program. If $I([\![C]\!]_{\mathbf{S}}(\alpha))$ is $I(\alpha)$, so is $I([\![C]\!]_{\mathbf{S}}([\![C]\!]_{\mathbf{S}}(\alpha)))$.* □

How to approximate $[\![$`while` j `do` C `od`$]\!]_{\mathbf{S}}(\alpha)$ is summarised in Algorithm 2. By Corollary 3.2.39, $(I((\mathrm{meas}_{\mathbf{S},i} \circ [\![C]\!]_{\mathbf{S}})^n(\mathrm{meas}_{\mathbf{S},i}(\alpha))))_{n \in \mathbb{N}}$ is a decreasing sequence and it reaches the fixed point for some n. Then, $(x_m)_{m \in \mathbb{N}}$ is an increasing sequence where $x_0 = (\mathrm{meas}_{\mathbf{S},i} \circ [\![C]\!]_{\mathbf{S}})^n(\mathrm{meas}_{\mathbf{S},i}(\alpha))$ and $x_{m+1} = (\mathrm{meas}_{\mathbf{S},i} \circ [\![C]\!]_{\mathbf{S}})(x_m) \vee x_m$. Note that x_m and x_{m+1} has the same I's.

Algorithm 2 Approximation of $[\![\texttt{while}\ j\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{S}}(\alpha)$

$\beta = \mathrm{meas}_{\mathbf{S},j}(\alpha)$.
$\epsilon = \beta$.
repeat
    $\alpha = \beta$.
    $\epsilon = \alpha \vee \epsilon$.
    $\beta = \mathrm{meas}_{\mathbf{S},j}([\![C]\!]_{\mathbf{S}}(\alpha))$.
until α and β have the same number of I's
$\beta = \alpha \vee \beta$.
while $\alpha \neq \beta$ **do**
    $\alpha = \beta$.
    $\beta = \mathrm{meas}_{\mathbf{S},j}([\![C]\!]_{\mathbf{S}}(\alpha)) \vee \alpha$.
end while
return $\epsilon \vee \alpha$.

The definition of the stabiliser semantics in Figure 3.3 can be understood as an algorithm to compute the semantics. Except for $[\![\texttt{while}\ j\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{S}}$, the semantics can be evaluated in polynomial time with respect to N provided that its components can be efficiently evaluated. For example, if $[\![C]\!]_{\mathbf{S}}$ and $[\![C']\!]_{\mathbf{S}}$ can be efficiently evaluated, we can compute $[\![\texttt{if}\ j\ \texttt{then}\ C\ \texttt{else}\ C'\ \texttt{fi}]\!]_{subSTAB}(\alpha)$ in polynomial time because both $\mathrm{meas}_{\mathbf{S},j}$ and $\vee$ can be efficiently evaluated. Since we have an approximation algorithm for $[\![\texttt{while}\ j\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{S}}$, we have an approximation algorithm for the stabiliser semantics. The algorithm can compute one step of the semantics in polynomial time in the above sense. Moreover, it works in polynomial time for constant-depth programs. We estimate its time complexity. Here, we assume that we can obtain and rewrite any element of any stabiliser in any assignment in unit time.

Theorem 3.2.47. *Let α be an assignment, C be a QIL program, s be its size, and d be its depth. $[\![C]\!]_{\mathbf{S}}(\alpha)$ is approximately computed in $O(3^d N^{d+3} s)$ time.*

Proof. Let $t(C)$ be an upper bound of the time complexity of computing an approximation of $[\![C]\!]_{\mathbf{S}}(\alpha)$ for any α. We use $t(s,d)$ to denote an upper bound of $t(C)$ for any QIL program C whose size is at most s and depth is at most d. Algorithm 1 shows that we can compute $\mathrm{up}_{\mathbf{S}}(Q,S)$ in $O(N^3)$ time and thus we can compute $\mathrm{meas}_{\mathbf{S},j}(\alpha)$ in $O(N^3)$ time. Therefore, if we ignore $\texttt{while}\ j\ \texttt{do}\ C\ \texttt{od}$, $t(s,d) = \max_{0\leq l<s}\{O(N^3) + t(j,d) + t(s-1-j,d)\}$. Let us investigate Algorithm 2. In the first loop in Algorithm 2, $\mathrm{meas}_{\mathbf{S},j}([\![C]\!]_{\mathbf{S}}(\alpha))$ is computed at most N times because the number of I's in α is never greater than N. In the second loop, $\mathrm{meas}_{\mathbf{S},j}([\![C]\!]_{\mathbf{S}}(\alpha))$ is computed at most $2N$ times. This is because α forms an increasing sequence, all I's are preserved in the sequence, and each block in α corresponds to either a stabiliser array or ■. Therefore, $t(d,s) \geq 3Nt(d-1,s-1)$ and thus $t(d,s) = \max\{3Nt(d-1,s-1), \max_{0\leq l<s}\{O(N^3) + t(j,d) + t(s-1-j,d)\}\}$. □

The size s of a program is independent of the number of variables N of QIL. However, we can estimate s is not too small with respect to N. Take a program C and assume $2s$ is less than N where $s = \mathrm{sz}(C)$. Since any size-one program contains at most two variables, $2s$ is the maximum number of variables occurring in C. The assumption means that C does not touch all variables and hence it can be regarded as a program of QIL having fewer variables such as $\mathbf{QIL}_{N-1}$. Therefore, it is reasonable to assume that $N \leq 2s$ holds.

Corollary 3.2.48. *Let α be an assignment, C be a QIL program, s be its size, and d be its depth. Assume $N \leq 2s$. $[\![C]\!]_{\mathbf{S}}(\alpha)$ is approximately computed in $O(3^d s^{d+4})$ time. When d is constant, an approximation of $[\![C]\!]_{\mathbf{S}}(\alpha)$ is computed in polynomial time with respect to s.* □

Corollary 3.2.49. *Let α be an assignment, C be a constant-depth QIL program whose size is polynomial with respect to N. Then, an approximation of $[\![C]\!]_{\mathbf{S}}(\alpha)$ is computed in polynomial time with respect to N.* □

3.2.5 Comparison with the basis semantics

We saw the stabiliser semantics is not monotone (Proposition 3.2.37), although the basis semantics is (Lemma 3.1.15). Nevertheless, the stabiliser semantics is better than the basis semantics in some sense. In the subsection, we compare the stabiliser semantics with the basis semantics. We show that their domains are connected by a Galois connexion and prove that the basis semantics is a sound approximation of the stabiliser semantics.

First of all, we define translations $\mathrm{conc}_{\mathbf{SA}}, \mathrm{abst}_{\mathbf{SA}}$ between the basis domain and the stabiliser domain.

Definition 3.2.50. In the definition, we identify $\bot$, Z, and X with I, $\langle \mathrm{Z} \rangle$, and $\langle \mathrm{X} \rangle$, respectively. Functions $\mathrm{conc}_{\mathbf{SA}} \colon \mathbf{A}^{\mathbf{Q}} \to \mathbf{S}^{\mathbf{Q}}$ and $\mathrm{abst}_{\mathbf{SA}} \colon \mathbf{S}^{\mathbf{Q}} \to \mathbf{A}^{\mathbf{Q}}$ are defined as follows.

$$\mathrm{conc}_{\mathbf{SA}}((P,b)) = \bigcup_{Q \in P} \left(\{ (Q \setminus s(Q,b), \blacksquare) \} \cup \bigcup_{i \in s(Q,b)} \{ (i, b(i)) \} \right) \tag{3.49}$$

$$\mathrm{abst}_{\mathbf{SA}}(\alpha) = (\alpha_0, c) \tag{3.50}$$

where

$$s(Q,b) = \{ i \mid i \in Q, b(i) \neq \top \} \tag{3.51}$$

$$c(j) = \begin{cases} \alpha_1(j) & (\alpha_1(j) = \mathrm{I}, \langle \mathrm{Z} \rangle, \langle \mathrm{X} \rangle) \\ \top & (\text{otherwise}) \end{cases} . \tag{3.52}$$

Proposition 3.2.51. *Both* $\mathrm{conc}_{\mathbf{SA}}$ *and* $\mathrm{abst}_{\mathbf{SA}}$ *are monotone.* □

Proposition 3.2.52. $\mathrm{conc}_{\mathbf{SA}}$ *and* $\mathrm{abst}_{\mathbf{SA}}$ *form a Galois connexion.*

Proof. Take $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $(P,b) \in \mathbf{A}^{\mathbf{Q}}$. Assume $\mathrm{abst}_{\mathbf{SA}}(\alpha) \leq_{\mathbf{A}} (P,b)$. Then, $\alpha_0 \leq_{\mathbf{\Pi}} P$ and for any $j \in \mathbf{Q}$, at least one of the following is true: $\alpha_1(j) = b(j)$, $\alpha_1(j) = \mathrm{I}$, or $b(j) = \top$. Let $\beta = \mathrm{conc}_{\mathbf{SA}}((P,b))$. Take $j \in \mathbf{Q}$ such that $\beta(j) \neq \blacksquare$. By definition, $b(j) \neq \top$ and thus $\blacksquare \neq \alpha_1(j) = b(j)$ or $\alpha_1(j) = \mathrm{I}$. In both cases, the size of $\alpha_0(j)$ is one. Therefore, $\alpha \leq_{\mathbf{S}} \beta$. The converse is easy. □

The translation $\mathrm{conc}_{\mathbf{SA}}$ may divide a partition in a basis abstract state to several parts. This is because a basis domain permits its state to waste the information about the basis. For example, $(\{\{0,1\}\}, (\mathrm{Z},\mathrm{Z}))$ is a basis abstract state. The definition of $\vDash_{\mathbf{A}}$ says it is a sound approximation of separable concrete states such as $\frac{1}{2}|00\rangle\langle 00| + \frac{1}{2}|11\rangle\langle 11|$. Indeed, the division does not cause any trouble, or rather preservation causes a trouble: If we preserve the partition, the above state is translated into $\{ (\{0,1\}, \blacksquare) \}$ because we require any stabiliser array in a stabiliser abstract state not to be separable. The translation loses the information. For example, $\mathrm{meas}_{\mathbf{A},0}$ changes the above state into $(\{\{0\},\{1\}\}, (\mathrm{Z},\mathrm{Z}))$, but we

cannot deduce from $\{ (\{ 0, 1 \}, \blacksquare) \}$ that the first qubit is the $\langle \mathrm{Z} \rangle$ basis state after measurement.

Then, we define a soundness relation, which is induced by the Galois connexion.

Definition 3.2.53. Let $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $(P, b) \in \mathbf{A}^{\mathbf{Q}}$. We write $\alpha \lhd_{\mathbf{SA}} (P, b)$ if $\alpha \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}((P, b))$

The relation has another characterisation. A sound approximation of a stabiliser array approximating a quantum state is a sound approximation of the quantum state.

Proposition 3.2.54. *Let $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $(P, b) \in \mathbf{A}^{\mathbf{Q}}$. $\alpha \lhd_{\mathbf{SA}} (P, b)$ if and only if for any $\rho \in \mathcal{D}_N$, $\alpha \vDash_{\mathbf{S}} \rho$ implies $(P, b) \vDash_{\mathbf{A}} \rho$.*

Lemma 3.2.55. *For any $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and any $\rho \in \mathcal{D}_N$ such that $\alpha \vDash_{\mathbf{S}} \rho$, $\mathrm{abst}_{\mathbf{SA}}(\alpha) \vDash_{\mathbf{A}} \rho$.* □

Lemma 3.2.56. *Let $\rho \in \mathcal{D}_N$ and $(P, b) \in \mathbf{A}^{\mathbf{Q}}$ such that $(P, b) \vDash_{\mathbf{A}} \rho$. Then, $\mathrm{conc}_{\mathbf{SA}}((P, b)) \vDash_{\mathbf{S}} \rho$.*

Proof. Assume $(P, b) \vDash_{\mathbf{A}} \rho$. First, we show that if $b(i) \neq \top$, then ρ is $\{ Q \setminus \{ i \} \mid Q \in P \} \cup \{ i \}$-separable. Without loss of generality, we can assume $b(i) = \mathrm{Z}$. Since ρ is P-separable, $\rho = \sum_j p_j \bigotimes_{Q \in P} \rho_{j,Q}$. Take $R \in P$ such that $i \in R$. Let $R' = R \setminus \{ i \}$. $\rho_{j,R} = \sum_{k,l \in \mathbb{B}} \alpha_{j,k,l} \sigma_{j,k,l,R'} \otimes |k\rangle\langle l|_{[i]}$ with some $\alpha_{j,k,l}$, $\sigma_{j,R'}$. $0 = \langle 0|_{(i)} \rho |1\rangle_{(i)} = \sum_j \alpha_{j,0,1} p_j \bigotimes_{Q \neq R} \rho_{j,Q} \otimes \sigma_{j,k,l,R'}$. Therefore,

$$\begin{aligned} \rho &= \sum_{k,l} \sum_j \alpha_{j,k,l} p_j \bigotimes_{Q \neq R} \rho_{j,Q} \otimes \sigma_{j,k,l,R'} \otimes |k\rangle\langle l|_{[i]} \\ &= \sum_k \sum_j \alpha_{j,k} p_j \bigotimes_{Q \neq R} \rho_{j,Q} \otimes \sigma_{j,k,R'} \otimes |k\rangle\langle k|_{[i]} \end{aligned} \tag{3.53}$$

Because $\rho_{j,R}$ is semi-definite, $\alpha_{j,k}$ is non-negative. The above equation also shows that $\rho_{j,R}$ can be decomposed to some state and the $\langle \mathrm{Z} \rangle$ basis state. □

Proof of Proposition 3.2.54. Assume $\alpha \lhd_{\mathbf{SA}} (P, b)$. Take $\rho \in \mathcal{D}_N$ and assume $\alpha \vDash_{\mathbf{S}} \rho$. By Proposition 3.2.30, $\mathrm{conc}_{\mathbf{SA}}((P, b)) \vDash_{\mathbf{S}} \rho$. Then, by Lemma 3.2.55, $(P, b) \geq_{\mathbf{A}} \mathrm{abst}_{\mathbf{SA}}(\mathrm{conc}_{\mathbf{SA}}((P, b))) \vDash_{\mathbf{A}} \rho$.

Conversely, assume that for any $\rho \in \mathcal{D}_N$, $\alpha \vDash_{\mathbf{S}} \rho$ implies $(P, b) \vDash_{\mathbf{A}} \rho$. By Lemma 3.2.55, for any $\rho \in \mathcal{D}_N$, $\alpha \vDash_{\mathbf{S}} \rho$ implies $\mathrm{conc}_{\mathbf{SA}}((P, b)) \vDash_{\mathbf{S}} \rho$. By Proposition 3.2.31, $\alpha \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}((P, b))$. □

Finally, we show the second main theorem of the section. The stabiliser semantics is a concretisation of the basis semantics.

Theorem 3.2.57. *For any $C \in \mathbf{QIL}$, any $\alpha \in \mathbf{S}^{\mathbf{Q}}$, and any $(P, b) \in \mathbf{A}^{\mathbf{Q}}$, if $\alpha \lhd_{\mathbf{SA}} (P, b)$, then $[\![C]\!]_{\mathbf{S}}(\alpha) \lhd_{\mathbf{SA}} [\![C]\!]_{\mathbf{A}}((P, b))$.*

Proof. By the induction on the structure of C. Take $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $(P, b) \in \mathbf{A}^{\mathbf{Q}}$ such that $\alpha \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}((P, b))$. Let $\beta = \mathrm{conc}_{\mathbf{SA}}((P, b))$.

- $[\![\mathtt{skip}]\!]_{\mathbf{S}}(\alpha) = \alpha \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}((P, b)) = \mathrm{conc}_{\mathbf{SA}}([\![\mathtt{skip}]\!]_{\mathbf{S}}((P, b)))$.
- By the induction hypothesis, $[\![C]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![C]\!]_{\mathbf{A}}((P, b)))$. By the induction hypothesis again, $[\![C']\!]_{\mathbf{S}}([\![C]\!]_{\mathbf{S}}(\alpha)) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![C']\!]_{\mathbf{A}}([\![C]\!]_{\mathbf{A}}((P, b))))$. Therefore, $[\![C\,;C']\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![C\,;C']\!]_{\mathbf{A}}((P, b)))$.

- Take $i \in \mathbf{Q}$. Assume $\{i\} \notin P$. Any single qubit unitary operator U does not change a partition, so $[\![U(i)]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![U(i)]\!]_{\mathbf{A}}((P,b)))$. Assume $\alpha_0(i) = \{i\}$ and $\{i\} \in P$. If $\alpha_1(i) = \mathrm{I}$, $[\![U(i)]\!]_{\mathbf{S}}(\alpha) = \mathrm{I}$. If not, $(\mathrm{conc}_{\mathbf{SA}}((P,b)))_1(i)$ is $\alpha_1(i)$ or $\blacksquare$.

- If $\{i,j\} \subset Q \in P$ for some Q, P is invariant under $[\![\mathtt{CX}(i,j)]\!]_{\mathbf{A}}$ and $(\mathrm{conc}_{\mathbf{SA}}((P,b)))_1(i) = \blacksquare$. Assume not. Since conditions are almost equal, $[\![\mathtt{CX}(i,j)]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![\mathtt{CX}(i,j)]\!]_{\mathbf{A}}((P,b)))$. The exceptions are $b(i) = b(j) = \bot$ and the else condition, but it makes $\leq_{\mathbf{S}}$ easier to hold.

- First, we show $[\![\mathrm{meas}_{\mathbf{S},i}]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![\mathrm{meas}_{\mathbf{A},i}]\!]_{\mathbf{A}}((P,b)))$. Let $(Q,S) = \alpha(i)$ and $(R,T) = \beta(i)$. There exist $(Q_0,S_0),\ldots,(Q_{m-1},S_{m-1})$ such that $Q \cup \bigcup_k Q_k = R$ and $(Q,S) \odot \bigodot_k (Q_k,S_k) \leq_{\mathbf{S}} T$. $(Q \setminus \{i\}) \cup \bigcup_k Q_k = R \setminus \{i\}$. $T \neq \blacksquare$ only if $R = \{i\}$. Hence, the claim holds. By the property of $\vee$ and the induction hypothesis, $[\![\mathtt{if}\ i\ \mathtt{then}\ C\ \mathtt{else}\ C'\ \mathtt{fi}]\!]_{\mathbf{S}}(\alpha) \leq_{\mathbf{S}} \mathrm{conc}_{\mathbf{SA}}([\![\mathtt{if}\ i\ \mathtt{then}\ C\ \mathtt{else}\ C'\ \mathtt{fi}]\!]_{\mathbf{A}}((P,b)))$.

- Similarly, $\mathtt{while}\ i\ \mathtt{do}\ C\ \mathtt{od}$ satisfies the statement.

□

From INVERTGHZ and BREAKGHZ, we conclude the stabiliser semantics is strictly better than the basis semantics.

3.3 Extended stabiliser abstract semantics

3.3.1 Motivation and idea

In the previous section, we proposed the stabiliser semantics, which is a strictly better semantics than the basis semantics. Can we improve the semantics? The stabiliser semantics gives us exact entanglement analysis provided that programs are within the framework of the stabiliser formalism. In order to obtain a better semantics, we have to investigate the black hole $\blacksquare$.

Recall the symbol $\blacksquare$ states that the abstract state does not have any information about the state of the associated block. We change a stabiliser array into the symbol when the non-Clifford operator T is applied to a non-standard basis state. Let us see the following example.

$$\mathtt{TMEASURE} \equiv \mathtt{GHZ;T(0);MEASURE}_1 \tag{3.54}$$

The stabiliser semantics tells us that the first qubit is separable from the others but they may be entangled.

$$\alpha \xrightarrow{\mathtt{GHZ}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\mathtt{T(0)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \blacksquare & & \end{array}\right] \xrightarrow{\mathtt{MEASURE}_1} \left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \blacksquare & & \\ & & \mathrm{Z} \end{array}\right] \tag{3.55}$$

This is not an exact analysis. Indeed, a concrete state is changed as follows.

$$\begin{aligned} \rho &\xrightarrow{\mathtt{GHZ}} |\mathrm{GHZ}\rangle\langle\mathrm{GHZ}| \\ &\xrightarrow{\mathtt{T(0)}} \frac{1}{2}|000\rangle\langle 000| + \frac{e^{i\frac{\pi}{4}}}{2}|111\rangle\langle 000| + \frac{e^{-i\frac{\pi}{4}}}{2}|000\rangle\langle 111| + \frac{1}{2}|111\rangle\langle 111| \\ &\xrightarrow{\mathtt{MEASURE}_1} \frac{1}{2}|000\rangle\langle 000| + \frac{1}{2}|111\rangle\langle 111| \end{aligned} \tag{3.56}$$

The last state shows all variables are separable. Furthermore, we can find it is a stabiliser basis state. Although the non-Clifford operator changes a state into a non-stabiliser basis state, measurement completely removes the effect of the operator. The example suggests us that the stabiliser semantics is too timid and that we may obtain some useful information if we refrain from using the symbol ■. Let us try to do that. Recall that if a state $|\psi\rangle$ is stabilised by a stabiliser S, $USU^\dagger$ "stabilises" $U|\psi\rangle$, even when $USU^\dagger$ is not a stabiliser.

$$\begin{aligned}\alpha \xrightarrow{\texttt{GHZ}} \left[\begin{array}{ccc|c} 0 & 1 & 2 & \\ \hline \mathrm{X} & \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} & \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} & \end{array}\right] \xrightarrow{\texttt{T(0)}} \left[\begin{array}{ccc|c} 0 & 1 & 2 & \\ \hline \mathrm{TXT}^\dagger & \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} & \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} & \end{array}\right] \\ \xrightarrow{\texttt{MEASURE}_1} \left[\begin{array}{ccc|c} 0 & 1 & 2 & \\ \hline \mathrm{I} & \mathrm{Z} & \mathrm{I} & \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} & \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} & \end{array}\right] \to \left[\begin{array}{c|c|c|c} 0 & 1 & 2 & \\ \hline \mathrm{Z} & & & \\ \hline & \mathrm{Z} & & \\ \hline & & \mathrm{Z} & \end{array}\right].\end{aligned} \tag{3.57}$$

We also try to do that in another example.

$$\texttt{TINIT} \equiv \texttt{INIT;H(0);CX(0,1)} \tag{3.58}$$

$$\texttt{TUNDO} \equiv \texttt{TINIT;T(0);CX(1,2);H(1);H(2);S(1);CX(1,2)} \tag{3.59}$$

While

$$\alpha \xrightarrow{\texttt{TINIT}} \left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{T(0)}} \left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \blacksquare & & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(0,1)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \blacksquare & & \end{array}\right] \tag{3.60}$$

and the last assignment is unchanged until the end of `TUNDO`, we can see that the zeroth qubit changes its entanglement partner from the first qubit to the second qubit as follows.

$$\begin{aligned}&\left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{T(0)}} \left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \mathrm{TXT}^\dagger & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(1,2)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{TXT}^\dagger & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right] \\ &\xrightarrow{\texttt{H(1)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{TXT}^\dagger & \mathrm{Z} & \mathrm{X} \\ \mathrm{Z} & \mathrm{X} & \mathrm{I} \\ \mathrm{I} & \mathrm{X} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{H(2)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{TXT}^\dagger & \mathrm{Z} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{X} & \mathrm{I} \\ \mathrm{I} & \mathrm{X} & \mathrm{X} \end{array}\right] \xrightarrow{\texttt{S(1)}} \\ &\left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{TXT}^\dagger & \mathrm{Z} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{Y} & \mathrm{I} \\ \mathrm{I} & \mathrm{Y} & \mathrm{X} \end{array}\right] \xrightarrow{\texttt{CX(1,2)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{TXT}^\dagger & \mathrm{I} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{Y} & \mathrm{X} \\ \mathrm{I} & \mathrm{Y} & \mathrm{I} \end{array}\right] \to \left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \mathrm{TXT}^\dagger & \mathrm{Z} & \\ \mathrm{Z} & \mathrm{X} & \\ \hline & & \mathrm{Y} \end{array}\right]\end{aligned} \tag{3.61}$$

These examples indicate that it is not impossible to tame non-stabiliser basis states. Even if the non-Clifford operator is applied, its effect may be localised and may be finally removed as shown in (3.57). We should point out that both programs do not touch the non-Pauli matrix $\mathrm{TXT}^\dagger$. After passing `T(0)`, all programs are performed on the first and second qubits. All transitions do not depend on the non-Pauli matrix. Indeed, if we add `CX(1,0)` at the end of `TUNDO`,

we would obtain the following.

$$\left[\begin{array}{ccc|c} 0 & & 2 & 1 \\ \hline \multicolumn{3}{c|}{\mathrm{CX}_{(1,0)}(\mathrm{TXT}^\dagger \otimes \mathrm{Z})\mathrm{CX}^\dagger_{(1,0)}} & \\ \mathrm{Y} & & \mathrm{Y} & \\ \hline & & & \mathrm{Y} \end{array}\right] \tag{3.62}$$

The effect of the non-Clifford operator spreads, although the spread never happens when we stay away from it.

Those examples show that even after being applied `T(`i`)` and some elements are turned to non-Pauli matrices, the other elements still have useful information. We use those elements to improve an abstract semantics. Naturally, we cannot record non-Pauli matrices, so we abstract them. We introduce a new symbol ♡ to denote non-Pauli matrices. Using the symbol, the matrices at the end of (3.61) and at (3.62) are written as follows.

$$\left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \heartsuit & \mathrm{Z} & \\ \mathrm{Z} & \mathrm{X} & \\ \hline & & \mathrm{Y} \end{array}\right] \qquad \left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \heartsuit & \heartsuit & \\ \mathrm{Y} & \mathrm{Y} & \\ \hline & & \mathrm{Y} \end{array}\right] \tag{3.63}$$

Here, ♡♡ denotes a non-Pauli matrix. The number of ♡ signifies the matrix operates two qubits. It does not mean the matrix can be decomposed into the tensor product of two matrices.

Precisely speaking, the symbol ♡ does not mean the matrix is not a Pauli matrix as the symbol ■ does not mean a non-stabiliser. It may denote a Pauli matrix: while $\mathrm{TTXT}^\dagger\mathrm{T}^\dagger = \mathrm{SXS}^\dagger = \mathrm{Y}$, we must deduce $\mathrm{T}\heartsuit\mathrm{T}^\dagger = \heartsuit$ because we have no information what ♡ denotes. The symbol ♡ states that we cannot guarantee the matrix is a Pauli matrix. In the next subsection, we will discuss what ♡ means.

3.3.2 Extended stabiliser domain

In the previous subsection, we introduced the symbol ♡. Before defining a new domain, we investigate the symbol. A question is what ♡ denotes. What is an instance of ♡? It is not an abstraction of any matrix. In the previous subsection, we wrote that the symbol ♡ is used when some unitary operator is applied. It is natural to define ♡ as an abstraction of any matrix that we can obtain as the result of application of unitary matrices to a Pauli matrix, e.g., $U\mathrm{X}U^\dagger$. Now, we want to axiomatise the meaning by examining such matrices. Let P be a Pauli matrix. First, we can find $UPU^\dagger$ is a Hermitian unitary matrix: $(UPU^\dagger)^\dagger = (U^\dagger)^\dagger PU^\dagger = UPU^\dagger$. Moreover, the matrix $UPU^\dagger$ is traceless, i.e., the trace is zero, unless $P = \mathrm{I}^{\otimes n}$. Since the trace is cyclic, $\mathrm{tr}\left(UPU^\dagger\right) = \mathrm{tr}\left(PU^\dagger U\right) = \mathrm{tr}P = 0$ provided $P \neq \mathrm{I}^{\otimes n}$. Therefore, an instance of ♡ should be either a traceless Hermitian unitary matrix or $\mathrm{I}^{\otimes n}$. Next, we move our attention from a row to an entire stabiliser array. Let P and Q be rows of a stabiliser array. As they are independent, PQ is a Pauli matrix and hence $\mathrm{tr}\,(PQ) = 0$. The relation is preserved under unitary transformation. $\mathrm{tr}\left(UPU^\dagger UQU^\dagger\right) = \mathrm{tr}\left(UPQU^\dagger\right) = \mathrm{tr}\left(PQU^\dagger U\right) = \mathrm{tr}\,(PQ) = 0$. In other words, the rows $UPU^\dagger$ are orthonormal with respect to the Hilbert-Schmidt inner product. The above traceless condition is a special case when $Q = \mathrm{I}^{\otimes n}$.

Now, we have all we need. In the stabiliser semantics, we used the relation between some bases and stabiliser arrays: a stabiliser array uniquely determines

a basis. We employed a stabiliser array S as an expression of a basis. Recall that $USU^\dagger$ "stabilises" $U|\psi\rangle$ when U stabilises $|\psi\rangle$. In other words, $USU^\dagger$ uniquely determines a basis. That is also true for orthonormal abelian groups of Hermitian unitary matrices. Therefore, we can use such a group as an expression of a basis.

Proposition 3.3.1. *Let $\langle S_0, \ldots, S_{n-1}\rangle$ be an orthonormal abelian group of Hermitian unitary matrices on n qubits. The states stabilised by $\langle (-1)^{s_0}S_0, \ldots, (-1)^{s_{n-1}}S_{n-1}\rangle$ forms a unique orthonormal basis.*

Lemma 3.3.2. *Let $S = \langle S_0, \ldots, S_{m-1}\rangle$ be an abelian group of Hermitian unitary matrices on n qubits. Assume that S is orthonormal with respect to Hilbert-Schmidt inner product. For any $0 \leq l < m-1$, $\mathrm{tr}\left(S_0 \cdots S_{m-1-l} P_{S_{m-l}}^{\sigma_{m-l}} \cdots P_{S_{m-1}}^{\sigma_{m-1}}\right) = 0$ where $\sigma_j \in \{+,-\}$ and $P_T^{\pm} = \frac{1}{2}(\mathrm{I} \pm T)$.*

Proof. We prove it by the induction on m.

- By assumption, $\mathrm{tr}\,(S_0) = 0$.
- First, $\mathrm{tr}\,(S_0 \cdots S_{m-1}) = 0$ by assumption. Assume for any S such that $|S| = m$, $\mathrm{tr}\left(S_0 \cdots S_{m-l-1} P_{S_{m-l}}^{\sigma_{m-l}} \cdots P_{S_{m-1}}^{\sigma_{m-1}}\right) = 0$ for any l. Take S whose size is $m+1$ and l. Assume that the statement holds for S and l.

$$\begin{aligned}
&\mathrm{tr}\left(S_0 \cdots S_{m-l-1} P_{S_{m-l}}^{\sigma_{m-l}} P_{S_{m-l+1}}^{\sigma_{m-l+1}} \cdots P_{S_m}^{\sigma_m}\right) \\
&= \frac{1}{2}\mathrm{tr}\left(S_0 \cdots S_{m-l-1}(\mathrm{I}\sigma_{m-l}S_{m-l}) P_{S_{m-l+1}}^{\sigma_{m-l+1}} \cdots P_{S_m}^{\sigma_m}\right) \\
&= \frac{1}{2}\mathrm{tr}\left(S_0 \cdots S_{m-1-l} P_{S_{m-l+1}}^{\sigma_{m-l+1}} \cdots P_{S_m}^{\sigma_m}\right) \\
&\qquad \sigma_{m-l}\frac{1}{2}\mathrm{tr}\left(S_0 \cdots S_{m-l-1} S_{m-l} P_{S_{m-l+1}}^{\sigma_{m-l+1}} \cdots P_{S_m}^{\sigma_m}\right) \\
&= 0\sigma_{m-l}0 = 0.
\end{aligned} \tag{3.64}$$

□

Corollary 3.3.3. *Let $S = \langle S_0, \ldots, S_{m-1}\rangle$ be an orthonormal abelian group of Hermitian unitary matrices on n qubits. For any $\sigma_j \in \{+,-\}$, $\mathrm{tr}\left(P_0^+ P_1^{\sigma_1} \cdots P_{m-1}^{\sigma_{m-1}}\right) = \mathrm{tr}\left(P_0^- P_1^{\sigma_1} \cdots P_{m-1}^{\sigma_{m-1}}\right)$.*

Proof. $0 = \mathrm{tr}\left(S_0 P_1^{\sigma_1} \cdots P_{m-1}^{\sigma_{m-1}}\right) = \mathrm{tr}\left((P_0^+ - P_0^-) P_1^{\sigma_1} \cdots P_{m-1}^{\sigma_{m-1}}\right)$. □

Lemma 3.3.4. *Let $S = \langle S_0, \ldots, S_{m-1}\rangle$ be an orthonormal abelian group of Hermitian unitary matrices on n qubits. For any k such that $0 \leq k < m$, $\mathrm{tr}\left(P_0^+ \cdots P_{m-1}^+\right) = \mathrm{tr}\left(P_0^- \cdots P_k^- P_{k+1}^+ \cdots P_{m-1}^+\right)$.*

Proof.

- By the above corollary, $\mathrm{tr}\left(P_0^+ \cdots P_{m-1}^+\right) = \mathrm{tr}\left(P_0^- P_1^+ \cdots P_k^-\right)$.
- Assume the statement holds for k.

$$\begin{aligned}
&\mathrm{tr}\left(P_0^+ \cdots P_{m-1}^+\right) = \mathrm{tr}\left(P_0^- \cdots P_k^- P_{k+1}^+ \cdots P_{m-1}^+\right) \\
&= \mathrm{tr}\left(P_{k+1}^+ \cdots P_{m-1}^+ P_0^- \cdots P_k^-\right) = \mathrm{tr}\left(P_{k+1}^- \cdots P_{m-1}^+ P_0^- \cdots P_k^-\right) \\
&= \mathrm{tr}\left(P_0^- \cdots P_{k+1}^- P_{k+2}^+ \cdots P_{m-1}^+\right).
\end{aligned} \tag{3.65}$$

□

Corollary 3.3.5. *Let $S = \langle S_0, \ldots, S_{m-1}\rangle$ be an orthonormal abelian group of Hermitian unitary matrices on n qubits. A vector space stabilised by S has the dimension 2^{n-m}.* □

Proof of Proposition 3.3.1. Let $\{|\psi_j\rangle\}_j$. Given $\{s_i\}_i$, $\langle (-1)^{s_0} S_0, \ldots, (-1)^{s_{n-1}} S_{n-1}\rangle$ stabilises a unique state $|\psi_j\rangle$. For any $i \neq j$, there exists S_k such that the eigenvalues corresponding to $|\psi_i\rangle$ and $|\psi_j\rangle$ are different. Then, $\langle\psi_i|\psi_j\rangle = \langle\psi_i|S_k|\psi_j\rangle = -\langle\psi_i|\psi_j\rangle$. □

Definition 3.3.6. Let $S = \langle S_0, \ldots, S_{n-1}\rangle$ be an orthonormal abelian group of Hermitian unitary matrices on n qubits. The S *basis* is the basis defined in Proposition 3.3.1.

Example 3.3.7. $\left\{ \frac{1}{\sqrt{2}}(|00\rangle \pm e^{i\frac{\pi}{4}} |11\rangle), \frac{1}{\sqrt{2}}(|01\rangle \pm e^{i\frac{\pi}{4}} |10\rangle) \right\}$ is the $\langle \mathrm{TXT}^\dagger \otimes \mathrm{X}, \mathrm{ZZ}\rangle$ basis.

Proposition 3.3.8. *Any orthonormal basis is the S basis with some S.*

Proof. Let $\{|\psi_i\rangle\}_i$ be an orthonormal basis. Define $U = \sum_i |\psi_i\rangle\langle i|$ where $\{|i\rangle\}_i$ is the standard basis. U is a unitary matrix and satisfies $|\psi_i\rangle = U|i\rangle$. Let S be the stabiliser $\langle \mathrm{ZI}\cdots\mathrm{I}, \mathrm{IZI}\cdots\mathrm{I}, \ldots, \mathrm{I}\cdots\mathrm{IZ}\rangle$. S stabilises the standard basis, so $USU^\dagger$ stabilises $\{|\psi_i\rangle\}_i$. □

We showed that an orthonormal abelian group of Hermitian unitary matrices uniquely determines an orthonormal basis. Although a stabiliser basis is stabilised by a unique stabiliser array, a basis can be stabilised by multiple abelian groups.

Example 3.3.9. Let $U = \mathrm{I}^{\otimes 3} - |011\rangle\langle 011| - |100\rangle\langle 100| + |011\rangle\langle 100| + |100\rangle\langle 011|$. U is a unitary matrix and the standard basis is invariant under the matrix. Therefore, both $\langle \mathrm{ZII}, \mathrm{IZI}, \mathrm{IIZ}\rangle$ and $\langle U(\mathrm{ZII})U^\dagger, U(\mathrm{IZI})U^\dagger, U(\mathrm{IIZ})U^\dagger\rangle$ stabilises the standard basis. However, these two groups are not the same.

We concluded from the observation of unitary transformation that a matrix having ♡ should be an abstraction of some orthonormal abelian group. Next, let us examine transformation via classical branches, `if` i `then` C `else` C' `fi`. Both the basis semantics and the stabiliser semantics are defined through join operators. The concrete semantics is defined by addition. That is, the summation is abstracted to the join. This is justified by the properties of abstract states: If α is a sound approximation of both ρ and σ, it is also a sound approximation of $\rho+\sigma$; If α is a sound approximation of ρ and β is larger than α, β is also a sound approximation of ρ. In the stabiliser semantics, Propositions 3.2.29 and 3.2.30 state the facts. How about a semantics with ♡? Suppose we have $\begin{bmatrix} \mathrm{Z} & ♡ \\ ♡ & \mathrm{Z} \end{bmatrix}$. As discussed before, this is an abstraction of matrices such as $\begin{bmatrix} \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} \end{bmatrix}$ and $\begin{bmatrix} \mathrm{Z} & \mathrm{X} \\ \mathrm{X} & \mathrm{Z} \end{bmatrix}$. The former and the latter are, for example, approximations of $\frac{1}{2}|00\rangle\langle 00|$ and $\frac{1}{4}|00\rangle\langle 00| + \frac{1}{4}|00\rangle\langle 11| + \frac{1}{4}|11\rangle\langle 00| + \frac{1}{4}|11\rangle\langle 11|$. Both are approximated by $\begin{bmatrix} \mathrm{Z} & ♡ \\ ♡ & \mathrm{Z} \end{bmatrix}$. However, the sum is not so. $\frac{3}{4}|00\rangle\langle 00| + \frac{1}{4}|00\rangle\langle 11| + \frac{1}{4}|11\rangle\langle 00| + \frac{1}{4}|11\rangle\langle 11|$ is not stabilised by any Hermitian unitary matrix of form $\begin{bmatrix} \mathrm{Z} & ♡ \\ ♡ & \mathrm{Z} \end{bmatrix}$.

Obviously, the trouble results from the existence of ♡. Both properties are satisfied, if no ♡ exists. The properties are used only in the semantics of `if` i `then` C `else` C' `fi` and `while` i `do` C `od`, that is when a concrete state is defined by the summation. One solution is to change a matrix to the symbol ■ when the join is needed. However, rows with no ♡ are too meaningful to discard. Any state approximated by an instance of $\begin{bmatrix} \mathrm{Z} & ♡ \\ \mathrm{X} & \mathrm{Z} \end{bmatrix}$ is necessarily stabilised by

XZ. We cannot use Z♡ to abstract the summation but can use XZ. For the reason, we remove only rows having ♡. We allow a stabiliser array to be a non-square matrix, e.g., $\begin{bmatrix} \mathrm{X} & \mathrm{Z} \end{bmatrix}$. Now, we have the definition of generalisation of stabiliser arrays.

Definition 3.3.10. Let n be a natural number. An *extended Pauli matrix on n qubits* is an n-product of elements of $\{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}, \heartsuit\}$. Let P be an extended Pauli matrix and $S = \{\, i \mid P_{\{i\}} = \heartsuit \,\}$. S is said to be the *hole of P*. Let W be a Hermitian unitary matrix on the hole S. A Hermitian unitary matrix R is an *instance of P with W* if

$$R = W \otimes \bigotimes_{i \notin S} P_{\{i\}}. \tag{3.66}$$

A Hermitian unitary matrix R is an *instance of P* if there exists a Hermitian unitary matrix W such that R is an instance of P with W. We call this W the *caulk of R for P*. We assume that P is an instance of P with I when P does not contain ♡.

Definition 3.3.11. The set $\{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}, \heartsuit\}$ is a monoid with zero, where the zero object is ♡ and the other products are inherited from the group of Pauli matrices. We say ♡ *commutes* with any element Q of the set. Let P and Q be extended Pauli matrices on n qubits. The *product of P and Q* is the elementwise product: its ith element is $P_{\{i\}}Q_{\{i\}}$.

Proposition 3.3.12. *Let P and Q be extended Pauli matrices on n qubits. Let R and S be instances of P and Q, respectively. Then, RS is an instance of PQ.*

Proof. Let W and X be the caulks of R and S for P and Q, respectively. Let A and B be the holes of P and Q. Note that the hole of PQ is $A \cup B$ because ♡ is an absorbing element.

$$\begin{aligned} RS &= (W \otimes \bigotimes_{i \notin A} P_{\{i\}})(X \otimes \bigotimes_{i \notin B} Q_{\{i\}}) \\ &= ((W \otimes \bigotimes_{i \in B \setminus A} P_{\{i\}}) \otimes \bigotimes_{i \notin A \cup B} P_{\{i\}})((X \otimes \bigotimes_{i \in A \setminus B} Q_{\{i\}}) \otimes \bigotimes_{i \notin A \cup B} Q_{\{i\}}) \\ &= ((W \otimes \bigotimes_{i \in B \setminus A} P_{\{i\}})(X \otimes \bigotimes_{i \in A \setminus B} Q_{\{i\}})) \otimes \bigotimes_{i \notin A \cup B} PQ_{\{i\}}. \end{aligned} \tag{3.67}$$

□

Definition 3.3.13. Let n and k be natural numbers such that $k \leq n$. A *k-syntactic extended stabiliser array on n qubits* is a $k \times n$ matrix whose rows are extended Pauli matrices such that there are instances $(R_i)_{i<k}$ of them that form an independent generator of an orthonormal abelian group. If $k = n$, an array is said to be *full*. The 0-syntactic extended stabiliser array is denoted by ■. We call $(R_i)_{i<k}$ an *instance of* the syntactic extended stabiliser array. The group generated by an instance is also called by an instance. The rows having ♡ are said to be *rough* rows and the other rows are *flat* rows. The *k-syntactic extended stabiliser space on n qubits* $\mathbf{SES}_{k,n}$ is the set of k-syntactic extended stabiliser arrays on n qubits. The *k-syntactic reduced stabiliser space on n qubits* $\mathbf{SRS}_{k,n}$ is the set of k-syntactic extended stabiliser arrays on n qubits where no ♡ occurs. $\mathbf{SES}_n$ and $\mathbf{SRS}_n$ are the unions of $\mathbf{SES}_{k,n}$ and $\mathbf{SRS}_{k,n}$, respectively.

Definition 3.3.14. Let n and k be natural numbers such that $k \leq n$. Let $M \in \mathrm{GL}(k, \mathbb{B})$ and $E \in \mathbf{SES}_{k,n}$. A $k \times n$ matrix $M \bullet E$ is defined by

$$(M \bullet E)_{\{i,\}} = \prod_j E_{\{j,\}}{}^{M_{\{i,j\}}} \tag{3.68}$$

-

Proposition 3.3.15. *Let $M \in \mathrm{GL}(k, \mathbb{B})$ and $E \in \boldsymbol{SES}_{k,n}$. $M \bullet E$ is a k-syntactic extended stabiliser array.*

Proof. Let $(R_i)_i$ be an instance of E. An instance of $(M \bullet E)_{\{i,\}}$ is given by $\prod_j R_j{}^{M_{\{i,j\}}}$. For $M \bullet E \in \mathbf{SES}_{k,n}$, it is enough to show they are independent. Assume $\prod_j R_j{}^{M_{\{i,j\}}}$ is a product of $\prod_j R_j{}^{M_{\{k,j\}}}$ and $\prod_j R_j{}^{M_{\{l,j\}}}$. The product $(\prod_j R_j{}^{M_{\{k,j\}}})(\prod_j R_j{}^{M_{\{l,j\}}})$ is $\prod_j R_j{}^{M_{\{k,j\}}+M_{\{l,j\}}}$. Since $\langle R_j \rangle_j$ are independent, it means $M_{\{k,\}} + M_{\{l,\}} = M_{\{i,\}}$ but it contradicts M is invertible. □

Definition 3.3.16. Let E and F be k-syntactic extended stabiliser arrays on n qubits. We write $E \simeq F$ if there exists $M \in \mathrm{GL}(k, \mathbb{B})$ such that $F = M \bullet E$ and $E = M^{-1} \bullet F$.

Proposition 3.3.17. *$\simeq$ is an equivalence relation.*

Proof. We prove $\simeq$ is transitive. Let E, F, G be k-syntactic extended stabiliser arrays on n qubits. Assume $E \simeq F$ and $F \simeq G$. Let M and N be the associated matrices.

$$G_{\{i,j\}} = \prod_k F_{\{k,j\}}{}^{N_{\{i,k\}}} = \prod_k \prod_l E_{\{l,j\}}{}^{N_{\{i,k\}} M_{\{k,l\}}} \tag{3.69}$$

Let $P = \prod_l E_{\{l,j\}}{}^{\sum_k N_{\{i,k\}} M_{\{k,l\}}}$. If neither P nor $G_{\{i,j\}}$ is $\heartsuit$, they are equal. If $G_{\{i,j\}}$ is not $\heartsuit$, there is no k, l, j such that $E_{\{l,j\}} = \heartsuit$ and $N_{\{i,k\}} = M_{\{k,l\}} = 1$. Hence, $P \neq \heartsuit$. Conversely, assume $P \neq \heartsuit$.

$$P = \prod_l \prod_m F_{\{m,j\}}{}^{\sum_k N_{\{i,k\}} M_{\{k,l\}} M^{-1}{}_{\{l,m\}}} \tag{3.70}$$

By the same argument, $\prod_m F_{\{m,j\}}{}^{\sum_l \sum_k N_{\{i,k\}} M_{\{k,l\}} M^{-1}{}_{\{l,m\}}} \neq \heartsuit$. It is nothing but $G_{\{i,j\}}$. □

Definition 3.3.18. The *k-extended stabiliser space on n qubits* $\mathbf{ES}_{k,n}$ is the quotient set $\mathbf{SES}_{k,n}/\simeq$ and the *extended stabiliser space on n qubits* $\mathbf{ES}_n$ is their union. An *extended stabiliser array* is an element of the space. $\mathbf{RS}_{k,n}$ and $\mathbf{RS}_n$ are defined similarly.

Proposition 3.3.19. *If $E \simeq F$, an instance of E is of F.*

Proof. By the proof of Proposition 3.3.15. □

Proposition 3.3.20. *If $E \simeq F$, the numbers of $\heartsuit$ in E and F are the same.*

Proof. Let M be the associated matrix. Since $E_{\{i,\}} = \prod_j F_{\{j,\}}{}^{M_{\{i,j\}}}$ and $\heartsuit$ is an absorbing element, the number of $\heartsuit$ does not decrease. If so, it contradicts the fact that M is invertible. □

Corollary 3.3.21. *$\boldsymbol{SRS}_{k,n}$ is closed under $\simeq$.* □

Notation 3.3.22. We use the same notation as for stabiliser arrays.

Example 3.3.23. The following matrices are extended stabiliser arrays.

$$\begin{bmatrix} \heartsuit & \heartsuit \end{bmatrix} \quad \begin{bmatrix} \mathrm{Z} & \heartsuit \\ \heartsuit & \mathrm{Z} \end{bmatrix} \quad \begin{bmatrix} \mathrm{Z} & \mathrm{X} & \mathrm{Y} \\ \mathrm{Y} & \mathrm{I} & \mathrm{X} \end{bmatrix} \quad \begin{bmatrix} \heartsuit & \heartsuit & \mathrm{Z} \\ \heartsuit & \heartsuit & \heartsuit \\ \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{bmatrix} \tag{3.71}$$

The rightmost array has instances such as

$$\begin{bmatrix} \mathrm{I} & \mathrm{I} & \mathrm{Z} \\ \mathrm{I} & \mathrm{Y} & \mathrm{I} \\ \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{bmatrix} \begin{bmatrix} \mathrm{Z} & \mathrm{X} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{Y} & \mathrm{Y} \\ \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{bmatrix} \begin{bmatrix} \mathrm{I} & \mathrm{I} & \mathrm{Z} \\ \frac{1}{2}(\mathrm{ZZI} + \mathrm{YXI} + \mathrm{ZZZ} - \mathrm{YXZ}) & & \\ \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{bmatrix}. \tag{3.72}$$

The following matrix is not an extended stabiliser array.

$$\begin{bmatrix} \heartsuit & \mathrm{Z} & \mathrm{Y} \\ \mathrm{I} & \mathrm{X} & \mathrm{I} \\ \heartsuit & \heartsuit & \mathrm{X} \end{bmatrix} \tag{3.73}$$

Whatever instance is chosen, the top and the middle rows do not commute.

An instance of an extended stabiliser array decide what state the array approximates. An array may not be full and hence its instance may not determine a basis. Hence, we use projections to define sound approximations as the basis domain does.

Definition 3.3.24. Let $A \subset \mathbf{Q}$, $B \subset A$, ρ be a quantum state on A, and E be an extended stabiliser array on $|B|$ qubits. We say an instance $\langle R_i \rangle_i$ of E is *sound for ρ on B* if $(P^+_{R_i})_{[B]} \rho (P^-_{R_i})_{[B]} = 0$ for any R_i.

Proposition 3.3.25. *Let ρ be a quantum state. Let W and V be commuting Hermitian unitary matrices. If $P^+_W \rho P^-_W = P^+_V \rho P^-_V = 0$, $P^+_{WV} \rho P^-_{WV} = 0$.* □

Corollary 3.3.26. *The soundness is well-defined.* □

Next, we define separability. The tensor product of syntactic extended stabiliser arrays is defined in the same manner as that of syntactic stabiliser arrays. However, we impose an extra condition on separability of syntactic extended stabiliser arrays.

Definition 3.3.27. Let E be a syntactic extended stabiliser array on n qubits and P be a partition of $[<n]$. S is *P-separable* if there exists $\left\{S_{[Q]}\right\}_{Q \in P}$ such that $S \simeq \bigotimes_{Q \in P} S_{[Q]}$ where each $S_{[Q]}$ is a syntactic extended stabiliser array on $|Q|$ qubits and at most one $S_{[R]}$ is not full. $\mathbf{NES}_n$ is the set of extended stabiliser arrays on n qubits that are not P-separable unless P is a singleton. The union is $\mathbf{NES}$. $\mathbf{NRS}_n$ and $\mathbf{NRS}$ are defined similarly.

Separability imposes the condition of being full on any component except for one component. It leads a rather strange consequence. $E \otimes F$ is not necessarily separable. However, the condition is important. If it is not satisfied, we cannot decompose an approximated state.

Example 3.3.28. $\begin{bmatrix} \mathrm{Z} & \mathrm{Z} \end{bmatrix} \otimes \begin{bmatrix} \mathrm{Z} & \mathrm{Z} \end{bmatrix} = \begin{bmatrix} \mathrm{Z} & \mathrm{Z} & \mathrm{I} & \mathrm{I} \\ \mathrm{I} & \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{bmatrix}$ is not separable. Indeed, the extended stabiliser array is sound for an entangled state $|\mathrm{GHZ}_4\rangle\langle\mathrm{GHZ}_4|$.

Proposition 3.3.29. *Let $A \subset \mathbf{Q}$, P be a partition of A, Q be a block of P, ρ be a quantum state on A, and E be an extended stabiliser array on $|Q|$ qubits. Assume ρ is P-separable and E has a sound instance for ρ on Q. Moreover, assume there exists a partition O of Q such that E is O-separable, that is $E = \bigotimes_{X \in O} E_{[X]}$. Then, ρ is $(P \setminus \{ Q \}) \cup O$-separable and each $E_{[X]}$ has a sound instance for ρ.*

Proof. Without loss of generality, we can assume $X = \{ B, C \}$, $E = F \otimes G$, and F is full. Let $\langle R_i \rangle_i$ be a sound instance of E. As $E = F \otimes G$, there exist instances $\langle V_i \rangle_i$ and $\langle W_i \rangle_i$ of F and G such that $(V_i \otimes \mathrm{I})_i$ and $(\mathrm{I} \otimes W_i)_i$ generate $\langle R_i \rangle_i$. These instances are sound. Let $\rho = \sum_k p_k \bigotimes_{D \in P} \rho_{k,D}$. Let $\{|\psi_i\rangle\}_i$ be the $\langle V_i \rangle_i$ basis. We can write $\rho_{k,Q}$ as $\sum_{i,j} |\psi_i\rangle\langle\psi_j| \otimes \sigma_{k,i,j,C}$.

$$\begin{aligned} \rho &= \sum_k \bigotimes_{D \in P \setminus \{ Q \}} \rho_{k,D} \otimes \sum_{i,j} |\psi_i\rangle\langle\psi_j| \otimes \sigma_{k,i,j,C} \\ &= \sum_{k,i} \bigotimes_{D \in P \setminus \{ Q \}} \rho_{k,D} \otimes |\psi_i\rangle\langle\psi_i| \otimes \sigma_{k,i,i,C}. \end{aligned} \tag{3.74}$$

The second equation is a consequence of $P^+_{V_i} \rho P^-_{V_i} = 0$ for any i. $\sigma_{k,i,i,C}$ is a state because $\langle \phi | \sigma_{k,i,i,C} | \phi \rangle = \langle \psi_i \phi | \rho_{k,Q} | \psi_i \phi \rangle \geq 0$. □

Notation 3.3.30. We use $\bigoplus_{Q \in P} S_{[Q]}$ to denote $\bigotimes_{Q \in P} S_{[Q]}$ such that at most one $S_{[Q]}$ is not full.

In spite of the existence of $\heartsuit$, we can also use Algorithm 1 to decompose a syntactic extended stabiliser array with a little modification.

Proposition 3.3.31. *Let E be a syntactic extended stabiliser array on n qubits and P be a partition of $[<n]$. Let S' be an output of Algorithm 3. Then, $S \simeq S'$. Furthermore, S is P-separable if and only if there exist $\left\{ S'_{[Q]} \right\}_{Q \in P}$ such that $S' = \bigoplus_{Q \in P} S'_{[Q]}$.*

Algorithm 3 Decomposition of a syntactic extended stabiliser array S on n qubits

1: Swap rows so that all rough and flat rows are on upper and lower sides, respectively. Let t be the threshold.
2: $r = t$.
3: Start Algorithm 1 from Step 2.

Proof. A proof is almost the same as that of Proposition 3.2.18. After running the algorithm, we obtain an output, which has a similar form to (3.35). The difference is that the upper rows are rough rows. We note that no flat row is a product of rows containing a rough rows because $\heartsuit$ is an absorbing element. Assume S is P-separable. By the proof of Proposition 3.2.18, any flat row is within some block of P. Take $Q \in P$ and a rough row ${S'}_{\{i,\}}$ such that ${S'}_{\{i,j\}} = \heartsuit$ for some $j \in Q$. We claim that whenever ${S'}_{\{i,k\}} \neq \mathrm{I}$, $k \in Q$. Let k be the smallest one such that ${S'}_{\{i,k\}} \neq \mathrm{I}$ and $k \notin Q$. First, assume ${S'}_{\{i,k\}} = \heartsuit$. Since $\simeq$ does not change the number of $\heartsuit$, k should belong to Q. Then, ${S'}_{\{i,k\}}$ is a Pauli matrix. If there is no flat row ${S'}_{\{j,\}}$ such that ${S'}_{\{j,\}} \neq \mathrm{I}$, we cannot remove ${S'}_{\{i,k\}}$ from the row. It contradicts that S' is P-separable. There are two possibilities: there exists a flat row ${S'}_{\{j,\}}$ such that ${S'}_{\{j,k\}} \neq \mathrm{I}$ and for any $l < k$, ${S'}_{\{j,l\}} = \mathrm{I}$; or any flat row such that ${S'}_{\{j,k\}} \neq \mathrm{I}$ has some column l such that ${S'}_{\{j,l\}} \neq \mathrm{I}$. In the former case, since ${S'}_{\{i,k\}} \neq \mathrm{I}$, such row is unique. By the form of the output,

${S'}_{\{i,k\}}$ anticommutes with ${S'}_{\{j,k\}}$. Therefore, S' is not P-separable. The latter case contradicts that k is the smallest one. □

Corollary 3.3.32. *Any syntactic extended stabiliser array has the finest partition P such that it is P-separable.* □

We investigated extended stabiliser arrays. Using the arrays, we define a new domain called the extended stabiliser domain. The construction is essentially the same as the stabiliser domain. The only difference is that extended stabiliser arrays are used instead of stabiliser arrays.

Definition 3.3.33. $\mathbf{ES}^*_k$ is defined to be $\mathbf{ES}_k$ if $k > 1$ and $\mathbf{ES}_1 \cup \{ \mathrm{I} \}$ if $k = 1$. $\mathbf{ES}^*$ is the union of them. $\mathbf{RS}^*_k$, $\mathbf{NES}^*_k$, $\mathbf{NRS}^*_k$, $\mathbf{RS}^*$, $\mathbf{NES}^*$, and $\mathbf{NRS}^*$ are defined in the same manner. Let $A \subset \mathbf{Q}$. $\gamma \subset \mathcal{P}A \times \mathbf{ES}^*$ (resp. $\gamma \subset \mathcal{P}A \times \mathbf{RS}^*$) is an *extended* (resp. *reduced*) *preassignment on A* if $\mathrm{pr}_0(\gamma)$ is a partition of A and for any $(Q, E) \in \gamma$, $E \in \mathbf{ES}^*_{|Q|}$ (resp. $E \in \mathbf{RS}^*_{|Q|}$). An extended (resp. reduced) preassignment on A is said to be an *extended* (resp. *reduced*) *assignment on A* if it is a subset of $\mathcal{P}A \times \mathbf{NES}^*$ (resp. of $\mathcal{P}A \times \mathbf{NRS}^*$). The *extended* (resp. *reduced*) *stabiliser domain* $\mathbf{E}^{\mathbf{Q}}$ (resp. $\mathbf{R}^{\mathbf{Q}}$) is the set of extended (resp. reduced) stabiliser assignments on $\mathbf{Q}$.

Definition 3.3.34. $\mathrm{up}_{\mathbf{E}}$ is defined by $\mathrm{up}_{\mathbf{E}}(Q, E) = \{ (Q_0, E_0), \ldots, (Q_{m-1}, E_{m-1}) \}$ such that $\{Q_i\}_i$ is a partition of Q, $E_i \in \mathbf{NSS}_{|Q_i|}$, and $\bigoplus_i E_i = E$. The normalisation $\mathrm{nml}\colon \mathbf{E}^{\mathbf{Q}} \to \mathbf{R}^{\mathbf{Q}}$ is a function defined as follows.

$$\mathrm{nml}(\{(Q_i, E_i)\}_i) = \bigcup_i \mathrm{up}_{\mathbf{E}}(Q_i, E_{i,nl}) \tag{3.75}$$

where $E_{i,nl}$ is the reduced stabiliser array obtained by removing all rough rows of E_i.

As written before, a "join operator" on the extended stabiliser domain is defined through the normalisation function and the join on the reduced stabiliser domain. First, let us define an order on the extended and reduced stabiliser domains.

Definition 3.3.35. Let $E, F \in \mathbf{ES}^*$. $E \leq_{\mathbf{ES}^*} F$ if and only if either $E = \mathrm{I}$, $E = F$, or $F = \blacksquare$. $\leq_{\mathbf{RS}^*}$, $\leq_{\mathbf{NES}^*}$, and $\leq_{\mathbf{NRS}^*}$ are the restrictions of $\leq_{\mathbf{RS}^*}$ on the associated spaces. Let $A \subset \mathbf{Q}$. Let γ and δ be extended preassignments on A. $\gamma \leq_{\mathbf{E}} \delta$ if and only if $\gamma_0 \leq_{\mathbf{\Pi}} \delta_0$ and for any $i \in A$, $\bigodot_{j \in \delta_0(i)} \gamma(j) \leq_{\mathbf{ES}^*} \delta_1(i)$. $\bigodot_{j \in \delta_0(i)} \gamma(j)$ is defined as follows.

$$\bigodot_{j \in Q}(Q_j, E_j) = \begin{cases} E_j & (\text{all } Q_j \text{ are the same}) \\ \mathrm{I} & (\text{all } E_j \text{ are I}) \\ \blacksquare & (\text{otherwise}) \end{cases} \tag{3.76}$$

Proposition 3.3.36. *$\mathbf{E}^{\mathbf{Q}}$ and $\mathbf{R}^{\mathbf{Q}}$ are lattices.* □

Definition 3.3.37. Let γ and δ be extended preassignments on A. The *approximate join of* them is $\mathrm{nml}(\gamma) \vee \mathrm{nml}(\delta)$, which is denoted by $\gamma \uplus \delta$.

In order to obtain the approximate join, we need to compute whether given two reduced stabiliser arrays are equal or not. We can do that by checking independence of generators composed of all rows of two arrays [8].

Definition 3.3.38. Let ρ be a quantum state and γ be an extended preassignment. γ is a *sound approximation of* ρ, denoted by $\gamma \vDash_{\mathbf{E}} \rho$, if it is $\mathrm{pr}_0(\gamma)$-separable and for any (Q, E), the following holds:

- if $E = \mathrm{I}$, there exists some σ such that $\rho = \frac{1}{2}\mathrm{I}_{[Q]} \otimes \sigma$ and
- if $E \in \mathbf{ES}$, a sound instance exists.

We list useful propositions about this soundness relation.

Proposition 3.3.39. *Let ρ be a quantum state and γ be an extended preassignment such that $\gamma \vDash_{\mathbf{E}} \rho$. Then, $\gamma[i \mapsto \mathrm{up}_{\mathbf{E}}(\gamma_0(i), \gamma_1(i))] \vDash_{\mathbf{E}} \rho$.*

Proof. By Proposition 3.3.29. □

Proposition 3.3.40. *For any quantum state ρ and extended preassignment γ, if $\gamma \vDash_{\mathbf{E}} \rho$, then $\mathrm{nml}(\gamma) \vDash_{\mathbf{E}} \rho$.* □

Proposition 3.3.41. *For any quantum state ρ and extended preassignment γ, δ such that $\gamma \leq_{\mathbf{E}} \delta$, if $\gamma \vDash_{\mathbf{E}} \rho$, then $\delta \vDash_{\mathbf{E}} \rho$.*

Proof. Take $(R, F) \in \delta$. There exist $(Q_0, E_0), \ldots, (Q_{l-1}, E_{l-1}) \in \gamma$ such that $\{Q_i\}_i$ is a partition of R and $\bigodot_j (Q_j, E_j) \leq_{\mathbf{ES}^*} F$.

- Assume $F = \mathrm{I}$. Then, $l = 1$ and $(Q_0, E_0) = (R, F)$.
- Assume $\bigodot_j (Q_j, E_j) = \mathrm{I}$. Then, there exists σ such that $\rho = \frac{1}{2^{|R|}}\mathrm{I}^{\otimes |R|} \otimes \sigma$. $P_V^+ \rho P_V^- = 0$ is satisfied whenever V acts on R.
- Otherwise, $l = 1$ and $(Q_0, E_0) = (R, F)$.

□

Although an extended assignment does not necessarily approximate the sum of approximated states, a reduced assignment does. Hence, the approximate join of extended assignments is an approximation of the summation of quantum states.

Proposition 3.3.42. *For any quantum state ρ, σ and reduced preassignment γ, if $\gamma \vDash_{\mathbf{E}} \rho$ and $\gamma \vDash_{\mathbf{E}} \sigma$, then $\gamma \vDash_{\mathbf{E}} \rho + \sigma$.* □

Corollary 3.3.43. *For any quantum state ρ, σ and extended preassignment γ, δ, if $\gamma \vDash_{\mathbf{E}} \rho$ and $\delta \vDash_{\mathbf{E}} \sigma$, then $\gamma \uplus \delta \vDash_{\mathbf{E}} \rho + \sigma$.* □

3.3.3 Extended stabiliser abstract semantics

We proposed the extended stabiliser domain and the approximate join operator on it. The operator gives a sound approximation of the summation of states. We use the domain and operator to define the extended stabiliser semantics. Before showing the definition, we define auxiliary functions: pnml, $\mathrm{meas}_{\mathbf{E},i}$, and $\mathrm{add}_{\heartsuit}$.

Definition 3.3.44. Let $\gamma \in \mathbf{E}^{\mathbf{Q}}$, $i \in \mathbf{Q}$, $(Q, E) = \gamma(i)$, and $A = \{\mathrm{X}, \mathrm{Y}\}$. pnml is defined in a similar way to nml. The difference is that pnml removes only rows having $\heartsuit$ in the ith column.

$$\mathrm{pnml}(i, (Q, E)) = \mathrm{up}_{\mathbf{E}}(Q, E_{nl,i}) \tag{3.77}$$

where $E_{nl,i}$ is the extended stabiliser array obtained by removing all rows whose ith columns are $\heartsuit$. $\mathrm{meas}_{\mathbf{E},i}(\gamma)$ is δ defined below.

$$\delta = \begin{cases} \gamma[i \mapsto \langle \mathrm{Z} \rangle] & (|\gamma_0(i)| = 1) \\ \gamma[i \mapsto \mathrm{up}_{\mathbf{S}}(\gamma_0(i), \mathrm{meas}_{\mathrm{sf},i}(\gamma_1(i)))] & (\gamma_1(i) \in \mathbf{SS}) \\ \gamma[i \mapsto \\ \quad (\{\, i \,\}, \langle \mathrm{Z} \rangle), \mathrm{up}_{\mathbf{E}}(\gamma_0(i) \setminus \{\, i \,\}, F)] & ((\gamma_1(i))_{\{j,i\}} \neq \heartsuit \text{ for any } j \\ & \quad \text{and a unique } k \text{ exists} \\ & \quad \text{s.t. } (\gamma_1(i))_{\{k,i\}} \in A) \\ \gamma[i \mapsto (\{\, i \,\}, \langle \mathrm{Z} \rangle), (\gamma_0 \setminus \{\, i \,\}, \blacksquare)] & (\text{otherwise}) \end{cases} \tag{3.78}$$

where F is obtained by removing the kth row and the ith column from $\gamma_1(i)$. Finally, $\mathrm{add}_{\heartsuit}$ is defined as follows.

$$\mathrm{add}_{\heartsuit}(i, E) = \begin{cases} E & (\text{no } j \text{ s.t. } E_{\{j,i\}} \in A) \\ F & (\text{unique } j \text{ exists s.t. } E_{\{j,\}} \text{ is flat and } E_{\{j,i\}} \in A) \\ G & (\text{otherwise}) \end{cases} \tag{3.79}$$

where F is obtained by changing $E_{\{j,i\}}$ to $\heartsuit$, and G is obtained by the following process. Take a syntactic extended stabiliser array of E that satisfies the following: For any j such that $E_{\{j,i\}} \in A$, no k exists such that $E_{\{k,i\}} \in A$ and the hole of $E_{\{k,\}}$ includes the hole of $E_{\{j,\}}$. G is obtained by changing all $E_{\{j,i\}}$ in A to $\heartsuit$.

Proposition 3.3.45. pnml, $\mathrm{meas}_{\mathbf{E},i}$, *and* $\mathrm{add}_{\heartsuit}$ *are well-defined.*

Proof. Let $E \simeq E'$ and M be the associated matrix. $\mathrm{pnml}(i, (Q, E))$ is well-defined because the numbers of $\heartsuit$ in E and E' are the same. The conditions in $\mathrm{meas}_{\mathbf{E},i}$ are exclusive. We care about only the third case. Assume E and E' has their unique row k and k' satisfying the condition. Since ${E'}_{\{j,i\}}$ anti-commutes with $E_{\{k,i\}}$, $M_{\{j,k\}} = 0$ for any $j \neq k'$. If it is 1, ${E'}_{\{j,i\}} \in A$. It contradicts the uniqueness. Therefore, the same M makes the outputs of $\mathrm{meas}_{\mathbf{E},i}$ equal. Finally, we show well-definedness of $\mathrm{add}_{\heartsuit}$. Note that the conditions are exclusive. The second and third conditions are almost the same as the third condition of $\mathrm{meas}_{\mathbf{E},i}$. If ${E'}_{\{j,i\}} \notin A \cup \{\, \heartsuit \,\}$, $M_{\{j,k\}} = 0$. When ${E'}_{\{j,i\}} = \heartsuit$, the change of $E_{\{k,i\}}$ can be ignored. □

$\mathrm{meas}_{\mathbf{E},i}$ means the measurement of the ith qubit. The difference between $\mathrm{meas}_{\mathbf{E},i}$ and $\mathrm{meas}_{\mathbf{S},i}$ is the third condition. The condition shows the case where $\heartsuit$ does not affect the measurement. Since $\gamma_1(i)$ is not a stabiliser array, it has a rough row. However, $\heartsuit$ is not in the ith column. This is the case in one of our motivating examples. pnml and $\mathrm{add}_{\heartsuit}$ are removals and insertion of $\heartsuit$. The conditions of $\mathrm{add}_{\heartsuit}$ is a bit complicated because we want to insert $\heartsuit$ as few as possible.

Definition 3.3.46. The *extended stabiliser abstract semantics* of QIL programs is a function $[\![\cdot]\!]_{\mathbf{E}} \colon \mathbf{QIL} \to \mathbf{E}^{\mathbf{Q}} \to \mathbf{E}^{\mathbf{Q}}$ defined in Figure 3.4. Here, $U \heartsuit U^\dagger = \heartsuit$, $\mathrm{CX}(\heartsuit \otimes P)\mathrm{CX}^\dagger = \heartsuit\heartsuit$, and $\mathrm{CX}(P \otimes \heartsuit)\mathrm{CX}^\dagger = \heartsuit\heartsuit$ for any U and P.

The semantics seems to be almost the same as the stabiliser semantics. One of the differences is the use of pnml. The reason is that a Hermitian unitary matrix does not necessarily commute or anticommute with X. For example, $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{e^{i\frac{\pi}{4}}}{\sqrt{2}}|1\rangle$ is a $\langle \mathrm{TXT}^\dagger \rangle$ basis state. However, $\mathrm{X}|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{e^{-i\frac{\pi}{4}}}{\sqrt{2}}|1\rangle$ is no longer a $\langle \mathrm{TXT}^\dagger \rangle$ basis state. Indeed, $\langle\psi|\mathrm{X}|\psi\rangle = \frac{1+i}{2}$. Therefore, we cannot specify which basis the concrete state belongs. WE did not use pnml in the stabiliser semantics because any Pauli matrix satisfies the above condition. X changes a stabiliser state within the same basis.

Next, we proceed to the soundness theorem.

$$
\begin{aligned}
[\![\texttt{skip}]\!]_{\mathbf{E}}(\gamma) &= \gamma \\
[\![C\,;C']\!]_{\mathbf{E}}(\gamma) &= [\![C']\!]_{\mathbf{E}}([\![C]\!]_{\mathbf{E}}(\gamma)) \\
[\![U\,\texttt{(}i\texttt{)}]\!]_{\mathbf{E}}(\gamma) &= \gamma[i \mapsto U_{(i)}\gamma_1(i)U_{(i)}^{\dagger}] \\
[\![\texttt{T(}i\texttt{)}]\!]_{\mathbf{E}}(\gamma) &= \begin{cases} \gamma & (\gamma_1(i) \text{ and } \mathrm{Z}_{(i)} \text{ commute}) \\ \gamma[i \mapsto \mathrm{add}_{\heartsuit}(i, \gamma_1(i))] & (\text{otherwise}) \end{cases} \\
[\![\texttt{CX(}i\texttt{,}j\texttt{)}]\!]_{\mathbf{E}}(\gamma) &= \begin{cases} \gamma[i \mapsto \mathrm{up}_{\mathbf{E}}(\gamma_0(i), \mathrm{CX}_{(i,j)}\gamma_1(i)\mathrm{CX}_{(i,j)}^{\dagger})] & (\gamma_0(i) = \gamma_0(j)) \\ \gamma & (\gamma_1(i) = \mathrm{I} \text{ and } \gamma_1(j) = \mathrm{I}) \\ \gamma[i \mapsto \langle \mathrm{Z} \rangle][j \mapsto \mathrm{pnml}(j, \gamma(j))] & (\gamma_1(i) = \mathrm{I}) \\ \gamma[j \mapsto \langle \mathrm{X} \rangle][i \mapsto \mathrm{pnml}(i, \gamma(i))] & (\gamma_1(j) = \mathrm{I}) \\ \gamma[j \mapsto \mathrm{pnml}(j, \gamma(j))] & (\gamma_1(i) = \langle \mathrm{Z} \rangle) \\ \gamma[i \mapsto \mathrm{pnml}(i, \gamma(i))] & (\gamma_1(j) = \langle \mathrm{X} \rangle) \\ \gamma[i, j \mapsto \mathrm{CX}_{(i,j)}(\gamma_1(i) \otimes \gamma_1(j))\mathrm{CX}_{(i,j)}^{\dagger}] & (\text{otherwise}) \end{cases}
\end{aligned}
$$

$$
\left[\!\!\left[\begin{array}{l} \texttt{if } i \\ \quad \texttt{then } C \\ \quad \texttt{else } C' \\ \texttt{fi} \end{array} \right]\!\!\right]_{\mathbf{E}} (\gamma) = [\![C]\!]_{\mathbf{E}}(\mathrm{meas}_{\mathbf{E},i}(\gamma)) \uplus [\![C']\!]_{\mathbf{E}}(\mathrm{meas}_{\mathbf{E},i}(\gamma))
$$

$$
\left[\!\!\left[\begin{array}{l} \texttt{while } i \\ \texttt{do} \\ \quad C \\ \texttt{od} \end{array} \right]\!\!\right]_{\mathbf{E}} (\gamma) = \biguplus_{n \in \mathbb{N}} \mathrm{meas}_{\mathbf{E},i}(([\![C]\!]_{\mathbf{E}} \circ \mathrm{meas}_{\mathbf{E},i})^n(\gamma))
$$

where $U \in \{ \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{H}, \mathtt{S} \}$.

Figure 3.4: Extended stabiliser abstract semantics of quantum imperative language

Lemma 3.3.47. *Let ρ be a quantum state and γ be an assignment such that $\gamma \vDash_{\mathbf{E}} \rho$. For any $i \in \mathbf{Q}$, $d \in \mathbb{B}$, $\mathrm{meas}_{\mathbf{E},i}(\gamma)$ is a sound approximation of $|d\rangle\langle d|_{(i)}\rho|d\rangle\langle d|_{(i)}$.*

Proof. Let (Q, E) be $\gamma(i)$.

- Assume $|Q| = 1$. Whatever E is, $|0\rangle\langle 0|_{(i)}|0\rangle\langle 0|_{(i)}\rho|0\rangle\langle 0|_{(i)}|1\rangle\langle 1|_{(i)} = 0$.
- Assume $|A| \neq 1$ and E is a stabiliser. Thanks to the stabiliser formalism, $\mathrm{meas}_{\mathbf{E},i}(\gamma) \vDash_{\mathbf{E}} |d\rangle\langle d|_{(i)}\rho|d\rangle\langle d|_{(i)}$.
- Assume $|A| \neq 1$ and exactly one row of E has X or Y in the ith column. Let r be the row and $\langle R_i \rangle_i$ be a sound instance of E. By assumption, if $q \neq r$, any R_q can be written as $\mathrm{Z}_{[i]} \otimes V_q$. We claim V_q is a sound instance of $\langle 0|_{(i)}\rho|0\rangle_{(i)}$.

$$
0 = |0\rangle\langle 0|_{(i)} P_{R_q}^{+} \rho P_{R_q}^{-} |0\rangle\langle 0|_{(i)} = |0\rangle\langle 0|_{[i]} \otimes (P_{V_q}^{+} \langle 0|_{(i)}\rho|0\rangle_{(i)} P_{V_q}^{-}) \tag{3.80}
$$

- In the last case, the statement trivially holds.

□

Theorem 3.3.48. *For any concrete state $\rho \in \mathcal{D}_N$, any extended assignment $\gamma \in \mathbf{E}^{\mathbf{Q}}$, and any QIL program $C \in \mathbf{QIL}$, $\gamma \vDash_{\mathbf{E}} \rho$ implies $[\![C]\!]_{\mathbf{E}}(\gamma) \vDash_{\mathbf{E}} [\![C]\!](\rho)$.*

Proof. We prove the theorem by the structural induction on C. For `skip` and $C;C'$, the statement trivially holds. Take γ and ρ such that $\gamma \vDash_{\mathbf{E}} \rho$.

(U) Take $i \in \mathbf{Q}$ and $U \in \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \mathrm{H}, \mathrm{S}\}$. Let $(Q, E) = \gamma(i)$. If $U_{(i)} E U_{(i)}^\dagger = \mathrm{I}$, then $E = \mathrm{I}$ and $\rho = \frac{1}{2}\mathrm{I} \otimes \sigma$ with some σ. ρ is unchanged through $U\texttt{(}i\texttt{)}$. Assume $U_{(i)} E U_{(i)}^\dagger \neq \mathrm{I}$. Then, $E \neq \mathrm{I}$. Let $\langle R_i \rangle_i$ be a sound instance of E. $\left\langle U_{(i)} R_i U_{(i)}^\dagger \right\rangle_i$ is a sound instance of $U_{(i)} E U_{(i)}^\dagger$. Indeed,

$$P^+_{U_{(i)} R_i U_{(i)}^\dagger} U_{(i)} \rho U_{(i)}^\dagger P^-_{U_{(i)} R_i U_{(i)}^\dagger} = U_{(i)} P^+_{R_i} U_{(i)}^\dagger U_{(i)} \rho U_{(i)}^\dagger U_{(i)} P^-_{R_i} U_{(i)}^\dagger = 0. \quad (3.81)$$

(`T`) Take $i \in \mathbf{Q}$. When E commutes with $\mathrm{Z}_{(i)}$, the above argument can be applied. Assume not. Let $\langle R_j \rangle_j$ be a sound instance of E. Take k such that $E_{\{k,i\}}$ anticommutes with Z. Define $\langle S_j \rangle_j$ as $S_j = R_j$ for any $j \neq k$ and $S_k = \mathrm{T}_{(i)} R_k \mathrm{T}_{(i)}^\dagger$. $\langle S_j \rangle_j$ is a sound instance of $\mathrm{add}_\heartsuit(i, E)$. More precisely, let W_k be the caulk of R_k for $E_{\{k,\}}$. Define $X_k = W_k \otimes \mathrm{T} E_{\{k,i\}} \mathrm{T}^\dagger$. X_k is the caulk of S_k for $\mathrm{add}_\heartsuit(i, E)$.

(`CX`) Take $i, j \in \mathbf{Q}$. Let $(Q, E) = \gamma(i)$ and $(R, F) = \gamma(j)$.

- Assume $Q = R$. By Proposition 3.3.29 and the same argument as we did for $U\texttt{(}i\texttt{)}$.
- Assume $Q \neq R$.
  - If $E = F = \mathrm{I}$, $\mathrm{CX} \frac{1}{4}\mathrm{I}^{\otimes 2} \mathrm{CX}^\dagger = \frac{1}{4}\mathrm{I}^{\otimes 2}$.
  - Assume $E = \mathrm{I}$ and $F \neq \mathrm{I}$. Take σ such that $\rho = \frac{1}{2}\mathrm{I}_{[i]} \otimes \sigma$. $\mathrm{CX}_{(i,j)} (\frac{1}{2}\mathrm{I}_{[i]} \otimes \sigma) \mathrm{CX}_{(i,j)}^\dagger = \frac{1}{2}|0\rangle\langle 0|_{[i]} \otimes \sigma + \frac{1}{2}|1\rangle\langle 1|_{[i]} \otimes \mathrm{X}_{(j)} \sigma \mathrm{X}_{(j)}^\dagger$. Hence, $|0\rangle\langle 0|_i \mathrm{CX}_{(i,j)} (\frac{1}{2}\mathrm{I}_{[i]} \otimes \sigma) \mathrm{CX}_{(i,j)}^\dagger |1\rangle\langle 1|_i = 0$. Let $\langle R_k \rangle_k$ be a sound instance of F. Removing all rows whose hole contains i, we obtain $\langle S_l \rangle_l$. It is an instance of $F_{nl,j}$, which is defined in the definition of pnml. We claim it is a sound instance. Because any two Pauli matrices commute or anti-commute,

$$P^+_{S_l} \mathrm{X}_{(j)} \sigma \mathrm{X}_{(j)}^\dagger P^-_{S_l} = \mathrm{X}_{(j)} P^+_{S_l} \sigma P^-_{S_l} \mathrm{X}_{(j)}^\dagger = 0. \quad (3.82)$$

  - The other cases are the same as the above.

(`if`) By the induction hypothesis, Corollary 3.3.43 and Proposition 3.3.47, the abstract state $[\![\texttt{if } i \texttt{ then } C \texttt{ else } C' \texttt{ fi}]\!]_{\mathbf{E}}(\gamma)$ is a sound approximation of the concrete state $[\![\texttt{if } i \texttt{ then } C \texttt{ else } C' \texttt{ fi}]\!](\rho)$ whenever $\gamma \vDash_{\mathbf{E}} \rho$.

(`while`) The same argument as above and the soundness theorem for the stabiliser semantics can be applied.

□

Although we essentially showed it before, the extended stabiliser semantics precisely analyses entanglement in the motivating example as follows.

Example 3.3.49.

$$\alpha \xrightarrow{\texttt{GHZ}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{T(0)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \heartsuit & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{MEASURE}_1} \left[\begin{array}{c|c|c} 0 & 1 & 2 \\ \hline \mathrm{Z} & & \\ \hline & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right]. \quad (3.83)$$

$$\left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \mathrm{X} & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{T(0)}} \left[\begin{array}{cc|c} 0 & 1 & 2 \\ \hline \heartsuit & \mathrm{X} & \\ \mathrm{Z} & \mathrm{Z} & \\ \hline & & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{CX(1,2)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \heartsuit & \mathrm{X} & \mathrm{X} \\ \mathrm{Z} & \mathrm{Z} & \mathrm{I} \\ \mathrm{I} & \mathrm{Z} & \mathrm{Z} \end{array}\right]$$
$$\xrightarrow{\texttt{H(1)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \heartsuit & \mathrm{Z} & \mathrm{X} \\ \mathrm{Z} & \mathrm{X} & \mathrm{I} \\ \mathrm{I} & \mathrm{X} & \mathrm{Z} \end{array}\right] \xrightarrow{\texttt{H(2)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \heartsuit & \mathrm{Z} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{X} & \mathrm{I} \\ \mathrm{I} & \mathrm{X} & \mathrm{X} \end{array}\right] \tag{3.84}$$
$$\xrightarrow{\texttt{S(1)}} \left[\begin{array}{ccc} 0 & 1 & 2 \\ \hline \heartsuit & \mathrm{Z} & \mathrm{Z} \\ \mathrm{Z} & \mathrm{Y} & \mathrm{I} \\ \mathrm{I} & \mathrm{Y} & \mathrm{X} \end{array}\right] \xrightarrow{\texttt{CX(1,2)}} \left[\begin{array}{cc|c} 0 & 2 & 1 \\ \hline \heartsuit & \mathrm{Z} & \\ \mathrm{Z} & \mathrm{X} & \\ \hline & & \mathrm{Y} \end{array}\right]$$

The extended stabiliser semantics is the same as the stabiliser semantics provided that the state is a stabiliser array and the non-Clifford gate is not used. Therefore, this semantics is also not monotone. However, as the order structure of the extended stabiliser domain is the same as of the stabiliser domain, the same argument shows the extended stabiliser semantics is monotone under some assumption. In the same manner, we can compute an upper approximation of $[\![\texttt{while}\ i\ \texttt{do}\ C\ \texttt{od}]\!]_{\mathbf{E}}$.

Lemma 3.3.50. *All* $\mathrm{up}_{\mathbf{E}}$, $\mathrm{meas}_{\mathbf{E},i}$, nml, pnml, *and* $\mathrm{add}_{\heartsuit}$ *are monotone.* □

Proposition 3.3.51. *Let* γ, δ *be extended assignments such that* $\gamma \leq_{\mathbf{E}} \delta$ *and* C *be a QIL program. Assume that for any* $i \in \mathbf{Q}$, *if* $\gamma_1(i) = \mathrm{I}$, *then* $\delta_1(i) = \blacksquare$. *Then,* $[\![C]\!]_{\mathbf{E}}(\gamma) \leq_{\mathbf{E}} [\![C]\!]_{\mathbf{E}}(\delta)$. □

Although the extended stabiliser semantics uses nml, pnml, and $\mathrm{add}_{\heartsuit}$, which do not occur in the stabiliser semantics, they do not affect the order of the time complexity of an approximation algorithm for the extended stabiliser semantics. Therefore, the same argument as for the stabiliser semantics shows its time complexity.

Theorem 3.3.52. *Let* γ *be an extended assignment,* C *be a QIL program,* s *be its size, and* d *be its depth.* $[\![C]\!]_{\mathbf{E}}(\gamma)$ *is approximately computed in* $O(3^d N^{d+3} s)$ *time.* □

Corollary 3.3.53. *Let* γ *be an extended assignment,* C *be a QIL program,* s *be its size, and* d *be its depth. Assume that* $N \leq 2s$. $[\![C]\!]_{\mathbf{E}}(\gamma)$ *is approximately computed in* $O(3^d s^{d+4})$ *time. When* d *is constant, an approximation of* $[\![C]\!]_{\mathbf{E}}(\gamma)$ *is computed in polynomial time with respect to* s. □

Corollary 3.3.54. *Let* γ *be an extended assignment,* C *be a constant-depth QIL program of polynomial size with respect to* N. *An approximation of* $[\![C]\!]_{\mathbf{E}}(\gamma)$ *is computed in polynomial time with respect to* N.

Finally, we point out the extended stabiliser semantics is a sound concretisation of the stabiliser semantics.

Definition 3.3.55. A function $\mathrm{conc}_{\mathbf{ES}} \colon \mathbf{S}^{\mathbf{Q}} \to \mathbf{E}^{\mathbf{Q}}$ is an embedding function. $\mathrm{abst}_{\mathbf{ES}} \colon \mathbf{E}^{\mathbf{Q}} \to \mathbf{S}^{\mathbf{Q}}$ is defined as follows.

$$\mathrm{abst}_{\mathbf{ES}}(\gamma)(j) = \begin{cases} \gamma(j) & (\gamma(j) \in \mathbf{SS}^*) \\ (\gamma_0(j), \blacksquare) & (\text{otherwise}) \end{cases}. \tag{3.85}$$

The difference between these semantics is existence of $\heartsuit$ and non-full matrices. They are expressed as $\blacksquare$ in a stabiliser assignment. Note that $\mathrm{abst}_{\mathbf{ES}}(\mathrm{conc}_{\mathbf{ES}}(\alpha))$ is α itself.

Proposition 3.3.56. $\mathrm{conc}_{\mathbf{ES}}$ *is monotone.* $\mathrm{abst}_{\mathbf{ES}}$ *is well-defined and monotone.* □

Proposition 3.3.57. $\mathrm{conc}_{\mathbf{ES}}$ *and* $\mathrm{abst}_{\mathbf{ES}}$ *form a Galois connexion*

Proof. Take $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $\gamma \in \mathbf{E}^{\mathbf{Q}}$. $\mathrm{abst}_{\mathbf{ES}}(\mathrm{conc}_{\mathbf{ES}}(\alpha)) = \alpha \leq_{\mathbf{S}} \alpha$. $\gamma \leq_{\mathbf{E}} \mathrm{abst}_{\mathbf{ES}}(\gamma) = \mathrm{conc}_{\mathbf{ES}}(\mathrm{abst}_{\mathbf{ES}}(\gamma))$. □

Although we have a Galois connexion, we cannot use the soundness relation induced by the connexion. This is due to the lack of monotonicity. The induced relation σ is upper-closed. Explicitly, $(\gamma, \alpha) \in \sigma$ and $\alpha \leq_{\mathbf{S}} \beta$ implies $(\gamma, \beta) \in \sigma$. Choosing α as γ, $(\alpha, \beta) \in \sigma$. However, we learned that some $(\alpha, \beta) \in \sigma$ and $C \in \mathbf{QIL}$ satisfy $[\![C]\!]_{\mathbf{S}}(\alpha) \not\leq_{\mathbf{S}} [\![C]\!]_{\mathbf{S}}(\beta)$. Hence, $([\![C]\!]_{\mathbf{S}}(\alpha), [\![C]\!]_{\mathbf{S}}(\beta)) \notin \sigma$. The soundness is broken. Then, we employ a strict and inelegant relation.

Definition 3.3.58. Let $\alpha \in \mathbf{S}^{\mathbf{Q}}$ and $\gamma \in \mathbf{E}^{\mathbf{Q}}$. We write $\gamma \lhd_{\mathbf{ES}} \alpha$ if the following holds: $\gamma \leq_{\mathbf{E}} \alpha$; for any $i \in \mathbf{Q}$, $\alpha_1(i) = \mathrm{I}$ if and only if $\gamma_1(i) = \mathrm{I}$; and for any $i \in \mathbf{Q}$ such that $\alpha_1(i) \neq \blacksquare$, $\alpha_1(i) = \gamma_1(i)$.

The relation states that a sound approximation must have the same information as to where I is. Therefore, the relation is not closed under $\leq_{\mathbf{S}}$. As shown before, monotonicity is broken due to difference in the number of I.

By definition, $\mathrm{abst}_{\mathbf{ES}}(\gamma)$ is the smallest one among α such that $\gamma \lhd_{\mathbf{ES}} \alpha$.

Proposition 3.3.59. *For any* $\gamma \in \mathbf{E}^{\mathbf{Q}}$, $\gamma \lhd_{\mathbf{ES}} \mathrm{abst}_{\mathbf{ES}}(\gamma)$. □

Proposition 3.3.60. *Let* $\gamma \in \mathbf{E}^{\mathbf{Q}}$ *and* $\alpha \in \mathbf{S}^{\mathbf{Q}}$. *If* $\gamma \lhd_{\mathbf{ES}} \alpha$, $\mathrm{abst}_{\mathbf{ES}}(\gamma) \leq_{\mathbf{S}} \alpha$. □

Under the soundness relation, we can say the stabiliser semantics is a sound approximation of the extended stabiliser semantics.

Theorem 3.3.61. *Let* C *be a QIL program,* γ *be an extended stabilise assignment, and* α *be a stabiliser assignment. Assume* $\gamma \lhd_{\mathbf{ES}} \alpha$. *Then,* $[\![C]\!]_{\mathbf{E}}(\gamma) \lhd_{\mathbf{ES}} [\![C]\!]_{\mathbf{S}}(\alpha)$.

Proof. By the induction on the structure of C. □

An stabiliser assignment α satisfies $\alpha \lhd_{\mathbf{ES}} \alpha$. Therefore, we can understand that the soundness relation claims that the extended stabiliser semantics is more concrete than the stabiliser semantics.

Chapter 4

Publicly Verifiable Blind Quantum Computation

Quantum physics brings computer science for quantum algorithms and quantum key distribution. The former is believed to outperform classical algorithms. A well-known example is the prime factorisation. It is difficult to compute in a classical computer, but it can be solved in the polynomial time when a quantum computer is used. Quantum key distribution achieves unconditional security. Its security relies only on the correctness of quantum physics and does not need any assumption about computational resources. Blind quantum computation (BQC) protocols are the place where these two notions meet: secure quantum computation. Amazingly, these protocols achieve the perfect security. They do not leak any information other than the size of computation. Although they are wonderful protocols, their security is too strong. These protocols reject any analysis of the computation. That causes a practical problem. In this chapter, we tackle the problem with the aid of classical encryption schemes.

The chapter is organised as follows. In Section 4.1, we review an existing verifiable BQC protocol and notions of correctness, universality, blindness, and verifiability. Next, in Section 4.2, we describe what is a problem. We show how a classical computer cannot analyse the computation in the protocol and why a naive solution does not work. We propose a new notion, public verifiability, and give its definition. Then, in Section 4.3, we show our idea and a summary of our new protocol. In Sections 4.4, 4.5, we instantiate the protocol. In the former section, we ignore public verifiability and show our protocol is a verifiable BQC protocol. In the latter section, we prove our protocol is publicly verifiable. In both sections, we discuss conditions of encryption schemes that are required to make our protocol have such properties. In the last section, we discuss encryption schemes. We construct an encryption scheme satisfying the conditions discussed in the preceding section.

4.1 Verifiable blind quantum computation protocol

4.1.1 Delegated quantum computation protocol

Suppose there is a party, Alice, who does not have a quantum computer but has limited quantum devices. She wants to perform some quantum computation but cannot do it herself, because the computation exceeds her quantum ability. Now, suppose that there is another party, Bob, who has a quantum computer and that the two parties are connected via a quantum channel and a classical channel. We assume both channels are noise-less, that is these channels carry data without deforming them. A *delegated quantum computation* (DQC) protocol

is a communication protocol between such two parties. A DQC protocol has the set of quantum computations, and the protocol allows Alice to delegate the computation to Bob as far as the computation belongs to the associated set.

Definition 4.1.1. A DQC protocol is said to be *universal* if it allows Alice to delegate any quantum computation to Bob.

We mainly focus on the quantum computations with classical inputs and outputs. Alice has a bit string s and a description of a unitary matrix U, and wants to obtain a measurement result of $U|s\rangle$. Here, the measurement is assumed to be performed in the X basis. We do not care about how Alice obtains the string and the description. We assume that Alice is given them with some auxiliary data as an input. We refer to them as an *input of a DQC protocol* provided that U is an allowed operation in the DQC protocol. Because $U|s\rangle = U\bigotimes_i X^{s_i}|0\rangle = U'|0\rangle$, the input string can be embedded into a unitary matrix. For the sake of simplicity, we assume the input string is always 0. Executing a DQC protocol, Alice finally obtains an *outcome of a DQC protocol*. As written above, the outcome is a bit string. When $U|0\rangle$ is not in the X basis, the outcome of the DQC protocol is not deterministic but probabilistic. We say an outcome is *correct* if it is a possible measurement result of $U|0\rangle$. In order to boost the computational ability of Alice, any outcome of the DQC protocol should be correct. We assume that Alice can accept or reject an outcome in a DQC protocol. She rejects an outcome if it is seemed to be incorrect. How to decide whether she accepts or rejects an outcome is defined in a DQC protocol.

Definition 4.1.2. A DQC protocol is *correct* if for any input, an outcome is correct and Alice accepts it provided that both parties respect the protocol.

We say a party is *honest* if the party respects the protocol. If not, a party is said to be *malicious* or *evil*. A malicious party does anything and may deviate from the protocol. Therefore, Alice will obtain an incorrect outcome if she runs a correct DQC protocol with evil Bob.

A blind quantum computation protocol is a correct DQC protocol with an extra property called blindness. Blindness allows Alice to hide any information about an input and an outcome but upper bounds of the sizes of them. The security given by blindness is unconditional, more precisely, perfect one. That is preserved even if Bob is malicious. Therefore, even if Bob has unlimited computational power and does anything, he has no chance to obtain any information.

Definition 4.1.3. Let L be a function. A DQC protocol is *blind while leaking at most* L if for any input X, whatever Bob does,

1. $L(X)$ determines the distribution of all classical information that Bob obtains and
2. $L(X)$ and the above distribution determine the state of the quantum system that Bob obtains

provided that Alice is honest. We often omit to write "leaking at most L".

Definition 4.1.4. A *blind quantum computation* (BQC) *protocol* is a correct and blind DQC protocol.

The definition of blindness cares about the case of malicious Bob. It guarantees that malicious Bob cannot learn anything except $L(X)$ about an input

X. However, Bob is still able to disturb the computation in a BQC protocol. Because Alice has no quantum computer, she may be unaware that an outcome is incorrect. Verifiability is a property of a DQC protocol that prevents such bad situations. Recall that Alice can accept or reject an outcome. The property allows honest Alice to reject an incorrect outcome of a DQC protocol with a bounded error probability.

Definition 4.1.5. Let ϵ is a real number such that $0 \leq \epsilon < 1$. A DQC protocol is said to be *ϵ-verifiable* if whatever Bob does, the probability of honest Alice accepting an incorrect outcome is at most ϵ.

4.1.2 The Fitzsimons–Kashefi protocol

The Fitzsimons–Kashefi (FK) protocol [39] is a verifiable BQC protocol proposed by Fitzsimons and Kashefi. The FK protocol is an improvement on the Broadbent–Fitzsimons–Kashefi (BFK) protocol [16], which is a BQC protocol. The (generic) FK protocol is shown in Protocol 4. Letting $t = 0$, we obtain the (generic) BFK protocol. First, we explain why the BFK protocol ($t = 0$) is a BQC protocol. The protocol uses a variant of MBQC, where Alice determines the measurement angles and Bob measures the qubits with the angles. Instead of $|+\rangle$, Bob uses rotated states $|+_\theta\rangle$ to construct a graph state. Because $\mathrm{Z}(\theta)$ and CZ commutes,

$$\left(\prod_{(i,j)\in E} \mathrm{CZ}_{(i,j)}\right)\left(\bigotimes_{i\in V} |+_{\theta_i}\rangle\right) = \left(\prod_{(i,j)\in E} \mathrm{CZ}_{(i,j)}\right)\left(\bigotimes_{i\in V} \mathrm{Z}(\theta_i)\,|+\rangle\right)$$
$$= \left(\bigotimes_{i\in V} \mathrm{Z}(\theta_i)\right)\left(\prod_{(i,j)\in E} \mathrm{CZ}_{(i,j)}\right)\left(\bigotimes_{i\in V} |+\rangle\right) = \left(\bigotimes_{i\in V} \mathrm{Z}(\theta_i)\right)|G\rangle\,. \tag{4.3}$$

Hence, Bob has a rotated graph state. Alice adds θ_i to the measurement angle in order to correct the rotation. Let δ'_i be such that $\delta_i = \delta'_i + \theta_i + \pi r_i$.

$$\bigotimes_{i\in V} \langle +_{\delta'_i+\theta_i+\pi r_i}| \bigotimes_{i\in V} \mathrm{Z}(\theta_i)\,|G\rangle = \bigotimes_{i\in V} \langle +_{\delta'_i+\pi r_i}|\,|G\rangle \tag{4.4}$$

Addition of π to δ'_i inverts the measurement result, so "measurement of the ith qubit of all qubits in $\bigotimes_{i\in V} \mathrm{Z}(\theta_i)\,|G\rangle$ with δ_i and result is b_i" is nothing but "measurement of the ith qubit of all qubits in $|G\rangle$ with δ'_i and result is $b_i + r_i$". It is nothing but a distributed MBQC. Hence, the protocol correctly works. As θ_i is uniformly randomly chosen, the angle δ_i is one-time padded. Similarly, the measurement result b_i is one-time padded by a random bit r_i. Therefore, the protocol is blind. Now, let $t > 0$. A qubit in T is called a *trap qubit*, and a qubit whose state is $|0\rangle$ or $|1\rangle$ is called a *dummy qubit*. Since $\mathrm{CZ}(|d\rangle\otimes|\psi\rangle) = |d\rangle\otimes Z^d\,|\psi\rangle$, any dummy qubit is not entangled with other qubits after being applied the operator CZ.

$$\prod_{(i,j)\in E} \mathrm{CZ}_{(i,j)} \bigotimes_{i\in V} |q_i\rangle = \bigotimes_{i\in N_G(T)} |d_i\rangle \otimes \prod_{\substack{(i,j)\in E \\ i,j\notin N_G(T)}} \mathrm{CZ}_{(i,j)} \bigotimes_{i\notin N_G(T)} \prod_{j\in N_G(i)\cap N_G(T)} \mathrm{Z}^{d_j}\,|q_i\rangle$$
$$= \bigotimes_{i\in N_G(T)} |d_i\rangle \otimes \bigotimes_{t\in T} |+_{\theta_t}\rangle \otimes |G'\rangle\,. \tag{4.5}$$

Therefore, any trap qubit t is isolated, and its measurement result b_t is definitely r_t because $\delta_t = \theta_t + \pi r_t$. Even if Bob deviates from the protocol and returns

Protocol 4 The Fitzsimons–Kashefi protocol [39]

Input

- Natural numbers n, t.
- A graph $G = (V, E)$ where vertices are labelled from 0 to $|V| - 1$.
- A description of a unitary operator U on n qubits.

Alice sends Bob G.
Alice randomly selects t vertices T.
Let $G' = (V', E')$ be the graph obtained by removing T and $N_G(T)$ from G, and $f_{G'}$ be its flow.
Alice computes the measurement angles $\{\phi_i\}_{i \in V'}$ to compute U where each ϕ_i belongs to $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$. Set $\phi_i = 0$ for any $i \in T \cup N_G(T)$.
for all $i \in V$ **do**
    if $i \in N_G(T)$ **then**
        Alice randomly chooses d_i from $\mathbb{B}$.
    end if
    Alice randomly chooses θ_i from $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$.
    Alice randomly chooses r_i from $\mathbb{B}$.
end for
for all $i \in V$ **do**
    Alice sends Bob a single qubit whose state is $|q_i\rangle$ where

$$|q_i\rangle = \begin{cases} |d_i\rangle & (i \in N_G(T)) \\ \prod_{j \in N_G(i) \cap N_G(T)} \mathrm{Z}^{d_j} |+_{\theta_i}\rangle & (i \notin N_G(T)) \end{cases} \tag{4.1}$$

end for
Bob creates the state $\prod_{(i,j) \in E} \mathrm{CZ}_{(i,j)} \bigotimes_{i \in V} |q_i\rangle$.
for $i = 0$ to $|V| - 1$ **do**
    Alice sends Bob δ_i where

$$\delta_i = (-1)^{\sum_{j \in X_i} b_j + r_j} \phi_i + \theta_i + \pi r_i + \pi \sum_{j \in Z_i} (b_j + r_j) \pmod{2\pi} \tag{4.2}$$

and $X_i = f_{G'}^{-1}(i)$ and $Z_i = \{ j \mid i \in N_G(f_{G'}(j)) \}$. Here, $f_{G'}^{-1}(i) = \emptyset$ if $i \notin G'$.
    Bob measures the ith qubit with δ_i.
    Bob sends Alice the measurement result b_i.
end for
Alice accepts the outcome if $b_t = r_t$ for all $t \in T$.

incorrect measurement results, he will change the measurement result of a trap qubit with some probability, because he does not know where the trap qubits are. This is why the FK protocol is verifiable. Finally, we note that the FK protocol requires Alice only to generate single qubits whose states are in the set $\left\{ \left| +_{\frac{k\pi}{4}} \right\rangle, |d\rangle \mid k \in [<8], d \in \mathbb{B} \right\}$.

Theorem 4.1.6 ([39])**.** *The FK protocol (Protocol 4) is a verifiable BQC protocol. Specifically, the following holds.*

- *For any unitary matrix U, the distribution of an outcome on an input including U coincides with the distribution of the measurement results of $U|0\rangle$ provided U is computable in G'.*
- *It is blind while leaking at most G.*

- *It is* $\left(1 - \frac{t}{|V|}\right)$*-verifiable.*

Fitzsimons and Kashefi also introduced the dotted-complete graph to make the protocol universal. Let N be a natural number. The dotted-complete graph $\tilde{K}_N$ is the complete graph of N vertices added a vertex in the middle of each edge. It has $N + \frac{N(N-1)}{2} = \frac{N(N+1)}{2}$ vertices and $2\frac{N(N-1)}{2} = N(N-1)$ edges. A removal of an added vertex with the connected edges removes the associated edge in the original complete graph. On the other hand, removing an added vertex and connecting the connected vertices keep the associated edge. Through these operations, we can obtain any graph having at most N vertices. In construction of a graph state, the former operation and latter can be realised by setting the added vertex a dummy qubit and $|+_{\pm\frac{\pi}{2}}\rangle$, respectively. First, Alice prepares $\tilde{K}_{3N}$ and randomly divides it into three $\tilde{K}_N$. Bob does not know how to divide the graph. One $\tilde{K}_N$ is used for the computation and the others are traps. All added vertices of one $\tilde{K}_N$ and all original vertices of one $\tilde{K}_N$ are the trap qubits. They run Protocol 4 with the construction. Universality is a consequence of the property of the dotted-complete graph. Furthermore, encoding the computation in a fault-tolerant manner where at most d Pauli errors will be detected or recovered, the protocol can achieve $(1 - \frac{t}{|V|})^d$-verifiability. However, the specific probability and how to encode the computation are not important for the later discussion. Therefore, we use ϵ_{FK} as a number such that the FK protocol (Protocol 4) is ϵ_{FK}-verifiable.

4.2 Public Verifiability

The FK protocol has several good properties. In particular, it is verifiable and allows Alice to verify whether she receives a correct outcome. However, the protocol does not allow a third party to do that. The reader may wonder why a third party need do that. We explain the reason by the following story.

There is a judge. One day, the judge is allocated to hear a case between Alice and Bob. Alice says

> I wanted to perform quantum computation. But, I didn't have much money and I couldn't buy a quantum computer. It's too expensive for me. So, I decided to delegate the computation to Bob's quantum server using the FK protocol. Of course, he charged me the server fee, but it is much lower than prices of quantum computers, so I could pay. After finishing running the FK protocol with the server, I obtained an outcome. But, I found that the measurement result b_t of some trap qubit t didn't coincide with the expected result r_t. So, Bob was malicious. (Recall that we assumed the channels are noise-less.) Unfortunately, I had computed a **BQP**-complete problem, so I can't directly check whether the outcome is correct. So, I had to discard the outcome. It's a breach of contract. He should pay me back.

On the other hand, Bob says

> I did obey the FK protocol. I properly applied CZ to the received qubits. I measured all qubits with the received measurement angles $\{\delta_i\}_{i\in V}$ and sent the measurement results $\{b_i\}_{i\in V}$. But, she surprisingly claimed that she had been given an incorrect outcome. The liar isn't me but her. The complaint should be dismissed.

The judge first finds either Alice or Bob lies. We do not care about the situation where both parties simultaneously lie. It is acceptable that an evil party suffers. We assume either Alice or Bob is honest. The judge has to decide whether Alice should pay or not. S/he thinks it is reasonable to order Bob to pay back if Alice is received an incorrect outcome. Hence, the question is whether Alice obtains a correct outcome or not. We emphasise that who wants to know the answer is neither Alice nor Bob but a third party. Verifiability is a property giving the answer to Alice and nobody else. The property does not help the judge. Now, the judge starts to collect information for a decision. Fortunately, Alice and Bob used a public classical channel and the judge can all messages sent via the classical channel. In fact, it is not due to luck. The use of the public channel never hurt the security since the FK protocol is blind: Bob cannot learn anything, not to mention any third party. However, due to the no-cloning theorem, the judge cannot obtain any quantum message. In summary, the judge has to decide with all classical messages. We say a protocol is publicly verifiable if it allows anyone to do that.

It is easy to see the FK protocol is not publicly verifiable. All classical messages in the FK protocol are G, $\{\delta_i\}_{i\in V}$, and $\{b_i\}_{i\in V}$. However, as explained above, each measurement angle δ_i and measurement result b_i are one-time padded using a random angle θ_i and a random bit r_i privately owned by Alice, so $\{\delta_i\}_{i\in V}$ and $\{b_i\}_{i\in V}$ are nothing but random sequences. Obviously, the shape of the graph G does not help the judge. Thus, s/he cannot obtain any meaningful information. The perfect security of the FK protocol makes all public information meaningless and hinders the judge from deciding.

Furthermore, due to the perfect security, it is impossible for the judge to decide even if s/he exercises her/his authority. Suppose the judge orders Alice and Bob to submit their private information. Then, the judge obtains the locations of trap qubits T', the computational angles $\{\phi'_i\}_{i\in V}$, the directions of dummy qubits $\{d'_i\}_{i\in N_G(T')}$, the random angles $\{\theta'_i\}_{i\in V}$, the random bits $\{r'_i\}_{i\in V}$, and, if the judge has a quantum device, the post-measurement quantum states. Here, we use prime symbols because there is no evidence that they honestly submit their private information. When the judge finds all measurement results of trap qubits $\{b_i\}_{i\in T'}$ are the same as their expected results $\{r'_i\}_{i\in T'}$, there is no longer any problem: Alice lies. Similarly, if the judge finds that the state of the ith qubit is not $|+_{\delta_i+\pi b_i}\rangle$, the judge can conclude that Bob lies. However, the judge cannot decide, when s/he finds that all quantum states are correct but there exists a trap qubit t' whose measurement result $b_{t'}$ does not coincide with the associated random bit $r'_{t'}$. The fact that the quantum states are $\bigotimes_{i\in V}|+_{\delta_i+\pi b_i}\rangle$ and $b_{t'} \neq r'_{t'}$ for some $t' \in T'$ means either that Alice is honest and $r'_{t'} = r_{t'} \neq b_{t'}$ or that Alice is evil and $r'_{t'} \neq r_{t'} = b_{t'}$. Unfortunately, there is no way to decide which one is true. Evil Alice can pretend to be a pitiful victim by submitting all information honestly except that $r'_t = r_t + 1$ and $\theta'_t = \theta_t + \pi$ for some trap qubit t. Note that $\theta'_t + \pi r'_t = \theta_t + \pi + \pi r_t + \pi = \theta_t + \pi r_t = \delta_t$. Honest Alice will submit the same information when she executes the FK protocol with the information and evil but unlucky Bob just flips the measurement result of the trap qubit t. On the other hand, it also easy for evil Bob to submit $\bigotimes_{i\in V}|+_{\delta_i+\pi b_i}\rangle$. Therefore, the judge cannot decide unless the evil party is unwise.

Let us move back to public verifiability. The property requires some additional information to be public so that anyone can decide whether Alice obtains a correct outcome. However, once it is disclosed, evil Bob will be able to improve his strategy. Hence, the information should be hidden until it becomes impossible to send an incorrect outcome. On the other hand, it should be protected

so that evil Alice cannot forge it. We can find such conflicts in the setting of commitment schemes. The no-go theorem for unconditionally hiding and unconditionally binding commitment schemes suggests that there is no *unconditional* public verifiability [69, 73]. Therefore, we focus on *computational* public verifiability. Fortunately, a DQC protocol naturally allows us to relax a restriction for Alice, because it assumes that Alice has a limited computational ability. Otherwise, she need not delegate her computation. Here, we assume that Alice is a probabilistic polynomial-time Turing machine. Now, we reach the definition of public verifiability.

Definition 4.2.1. Let ξ be a function such that $0 \leq \xi(X) < 1$ for any input X. A DQC protocol is *ξ-publicly verifiable* if there exists a polynomial-time algorithm P that computes "accept" or "reject" from all classical messages in the protocol and satisfies the following two conditions.

1. For any strategy of Bob and any input X, the probability that P outputs "accept" but Alice does not obtain a correct outcome is less than $\xi(X)$ when she follows the protocol.

2. For any probabilistic polynomial-time algorithm A executing the DQC protocol with honest Bob, there exists a probabilistic polynomial-time algorithm A' such that for any family of polynomially-bounded input $\{X_n\}_{n\in\mathbb{N}}$ and any polynomially-bounded functions h, f,

$$\left|\Pr\left[v = w \;\middle|\; \begin{array}{l} a \leftarrow h(1^n, X_n), (v, m) \leftarrow A(1^n, X_n, a) \\ j \leftarrow P(m), j = \text{reject}, w \leftarrow f(1^n, X_n) \end{array}\right]\right.$$
$$\left. - \Pr\left[v = w \;\middle|\; a \leftarrow h(1^n, X_n), v \leftarrow A'(1^n, X_n, a), w \leftarrow f(1^n, X_n)\right]\right| \quad (4.6)$$

   is negligible where m is all classical messages with Bob.

We call P a *public verifier*.

The first condition states that when a third party accepts, Bob cannot deceive Alice without a bounded probability. The second states that Alice cannot gain non-negligible information when a third party rejects. The second condition is borrowed from semantic security and prohibits evil Alice from obtaining any non-negligible partial information. This is because evil Alice can hide what is the outcome. A DQC protocol usually assumes that the last messages are the outcome. However, in a BQC protocol, Alice can secretly add meaningless computation to the end of computation so that she gains the actual outcome without receiving the seeming outcome.

By definition, we can immediately prove that public verifiability implies verifiability. The implication is strict. There exists a DQC protocol that is verifiable but not publicly verifiable.

Corollary 4.2.2. *A ξ-publicly verifiable DQC protocol is ξ-verifiable.* □

Proposition 4.2.3. *The FK protocol is not publicly verifiable unless* ***BQP*** $=$ ***BPP***.

Proof. Recall that all classical messages are G, $\{\delta_i\}_{i\in V}$, and $\{b_i\}_{i\in V}$. Assume there exists a public verifier P. If there exists G such that P always returns "accept", evil Bob flips all measurement results and the first condition is broken. Therefore, for any G, there exists $\{\delta_i\}_{i\in V}$ and $\{b_i\}_{i\in V}$ such that P returns "reject". Let A be the algorithm that follows the FK protocol. Because the messages are uniformly random, there exists non-zero probability that P rejects. By

the assumption, there exists a probabilistic polynomial-time algorithm A' that simulates A communicating with honest Bob. Let L be a language in **BQP** and $\{U_n\}_{n\in\mathbb{N}}$ be unitary matrices for the language. Let h be a constant function 0 and f be a characteristic function. Then, for any $x_n \in \{0,1\}^n$, embedding it with U_n into an input, A can correctly compute whether $x_n \in L$ except the probability $\frac{1}{3}$. Therefore, A' can compute it except the probability $\frac{1}{3} - \nu n$ where ν is a negligible function. Therefore, **BPP** includes **BQP**. □

Conversely, if **BQP** is **BPP**, Alice can compute any quantum computation without executing the FK protocol. Therefore, the FK protocol is 0-publicly verifiable with the public verifier that always returns "reject".

Now, we close the story about the judge. S/he learns the above proposition and finds that the judge cannot correctly guess. The judge tosses a coin to decide. The result is a tail, so the judge orders Bob to pay back. Regrettably, the truth is Alice lies. Lucky Alice succeeds in deceiving the judge and pitiable Bob works for nothing.

At the end of this section, we point out that it is easy to resolve the problem when there exists a trustworthy neutral party. Instead of Alice, the party decides the locations of traps and the random bits of them, i.e., the expected measurement results. Then, the party secretly sends Alice them. After executing the FK protocol, the party checks whether the measurement results of the traps are the expected ones. Therefore, if the judge had consulted before execution, the judge could correctly decide. However, it is much too difficult to confirm the judge is really trustful. From the view of another person, there is no reason to assume the judge is neutral. If evil Bob bribes the judge, s/he will leak the locations to him. If the judge is on Alice's side, the judge will lie about the expected results. That is the reason why we defined public verifiability so that anyone could verify executions. In other words, we do not trust any specific party but do believe that the majority in the world is honest.

4.3 Towards achievement of public verifiability

In the previous section, we defined public verifiability and saw that the FK protocol does not have the property. However, the FK protocol has several wonderful properties as shown in the first section. Therefore, we decide to modify the FK protocol so that it has public verifiability without weakening the protocol. In the section, we show an idea to achieve public verifiability.

Let us recall the definition of public verifiability. The second condition requires Alice to be unable to obtain any partial information when a public verifier rejects. Because the measurement results contain partial information, it means that the results should be hidden from Alice as long as there is a possibility of a public verifier rejecting. On the other hand, it must be impossible for evil Bob to change these measurement results. Therefore, we use a commitment scheme to hide the information from Alice as follows.

1: First, Alice and Bob execute the FK protocol. However, instead of sending each measurement result, he commits it to her.
2: Then, the outcome is accepted or rejected. The execution ends if rejected.
3: Finally, Bob reveals the committed measurement results.

However, the above simple protocol is immediately faced with two problems and hence it cannot work.

Let us recall MBQC and the FK protocol. In MBQC, each measurement angle depends on the previous measurement results. Indeed, Alice is required to

compute Equation (4.2) in the FK protocol. If she has no measurement results, she cannot decide the measurement angle and cannot run the FK protocol. In fact, she should not compute the measurement angle. Because she has all $\{r_i\}_{i\in V}$, $\{\phi_i\}_{i\in V}$, $\{\theta_i\}_{i\in V}$, $\{X_i\}_{i\in V}$, and $\{Z_i\}_{i\in V}$, Equation (4.2) can be understood as $\delta_i = g_i(\{b_j\}_{j<i})$ with some function g_i known by Alice. Computing the inverse of g_i inductively, she can restore measurement results b_j. The other problem is how Alice accepts or rejects an outcome. The FK protocol uses the fact she has the measurement results of the trap qubits in the verification. If she does not know $\{b_t\}_{t\in T}$, she cannot check whether $b_t = r_t$.

In order to resolve the two problems, we make further modifications. First, we change the party who verify the outcome from Alice to Bob. After executing the FK protocol, Alice discloses the traps, i.e., T and $\{r_t\}_{t\in T}$. Using them, Bob checks whether $b_t = r_t$ for any trap qubit t. Note that he certainly knows the measurement results $\{b_i\}_{i\in V}$. Then, he decides whether he accepts or rejects the outcome and hence whether he reveals the measurement results or not. Side effects of the modification are that we can force Alice to disclose the true traps and anyone can learn about the traps. If Alice discloses false traps, she will be unable to obtain the outcome. For the first problem, we require the scheme to be homomorphic and make Alice commit the measurement angles. In order to measure qubits, Bob has to learn the measurement angles and thus has to be able to reveal the measurement angles. Therefore, we decide to use a public-key encryption scheme and implement a commitment scheme using it.

In summary, our protocol is, roughly speaking, as follows.

1: First, Bob generates a pair of keys and announces an encryption key.
2: Alice and Bob execute the FK protocol, encrypting all messages.
3: Next, Alice discloses the traps.
4: Then, Bob accepts or rejects the outcome. The execution ends if rejected.
5: Finally, Bob reveals the decryption key.

4.4 New verifiable blind quantum computation protocol

In the previous section, we showed the outline of our protocol. The next step is an instantiation of the protocol. A question arises: Which encryption scheme can we use? In this and the next sections, we discuss sufficient conditions for our protocol. In the section, we focus on how to preserve the properties of the FK protocol. For a while, we forget about public verifiability. Conditions for the property will be discussed in the next section. Here, we give sufficient conditions for verifiable BQC protocols, show instantiations of our protocol with the conditions, and prove they are indeed verifiable BQC protocols.

The differences of our protocol and the FK protocol are the encryptions of all classical messages and the disclosure of the traps. An encryption scheme has to be chosen so that the differences do not create any hole. We can find that the encryptions of measurement angles involve the risk of leaks of private information. Alice computes a ciphertext of a measurement angle from the received ciphertexts of the previous measurement results. Therefore, the random part in the ciphertext may depend on the random parts in the received ciphertexts. Evil Bob will learn which measurement results a measurement angle depends. Therefore, we assume that an encryption scheme is rerandomisable, which make Bob unable to extract anything but the plaintext. The condition is not enough. Evil Bob will send invalid messages so that the messages of Alice contains extra information. We illustrate how that works with the following example.

Example 4.4.1. Suppose Alice and Bob use Goldwasser-Micali encryption scheme. An encryption key is (n, x) and Alice has a bit b_0 and a set $I \subset \{0, 1, 2\}$. The bit b_0 and the set I is her secret. Alice computes a ciphertext α_0 of b_0 and receives two encrypted bits α_1, α_2 from Bob. Then, she sends $\beta = r^2 \prod_{i \in I} \alpha_i$ with randomly chosen $r \in \mathbb{Z}_n^*$. It seems that she successfully sends the bit summation of bits in I and no other information is leaked. Now, suppose Bob is evil. He sends $n = p_0p_1p_2p_3p_4p_5$ and $x \in \mathbb{Z}_n^*$ such that $\left(\frac{x}{p_0}\right) = \left(\frac{x}{p_1}\right) = -1$ and $\left(\frac{x}{p_i}\right) = 1$ for other i. Moreover, α_i is taken such that $\left(\frac{\alpha_i}{p_{2i}}\right) = \left(\frac{\alpha_i}{p_{2i+1}}\right) = -1$ and $\left(\frac{\alpha_i}{p_j}\right) = 1$ for other j. Note that $\left(\frac{x}{n}\right) = \left(\frac{\alpha_1}{n}\right) = \left(\frac{\alpha_2}{n}\right) = 1$ and they are seemed to Alice to be valid ciphertexts unless she knows n is not the product of two primes. Then, $\left(\frac{r^2}{p_i}\right) = 1$ for any i and $\left(\frac{\alpha_j}{p_{2k}}\right) = 1$ unless $j = k$. Therefore, $\left(\frac{\beta}{p_{2i}}\right) = -1$ if and only if $i \in I$ for $i \in \{1, 2\}$. Moreover, $\left(\frac{\beta}{p_0}\right) = -1$ implies $0 \in I$ and $b_0 = 1$. It means evil Bob succeeds in obtaining partial information about I and b_0, which are the private information of Alice.

Such a trick will be revealed when Bob discloses the decryption key. However, the trick succeeds in damaging blindness, which states Bob cannot learn anything whatever he does. Notice that the bit summation is needed to compute measurement angles. Hence, we require that Alice can verify whether Bob's messages are valid.

Definition 4.4.2. We say a classical public-key encryption scheme (G, E, D) is *verifiable* if there exist polynomial-time algorithms V_E, V_D, and V_C such that for any $n \in \mathbb{N}$, any $e', d', \alpha' \in \mathbb{B}^*$, and any $e \in \mathbf{EKS}_n$,

$$V_E(n, e') = \begin{cases} 1 & (e' \in \mathbf{EKS}_n) \\ 0 & (e' \notin \mathbf{EKS}_n) \end{cases} \tag{4.7}$$

$$V_D(n, e, d') = \begin{cases} 1 & ((e, d') \in \mathbf{KS}_n) \\ 0 & ((e, d') \notin \mathbf{KS}_n) \end{cases} \tag{4.8}$$

$$V_C(e, \alpha') = \begin{cases} 1 & (\alpha' \in \mathbf{CS}_{n,e}) \\ 0 & (\alpha' \notin \mathbf{CS}_{n,e}) \end{cases} \tag{4.9}$$

If an encryption scheme is verifiable, Bob will not send invalid messages. Even if he does, the invalid messages are certainly detected by Alice. Still, he is able to send a valid ciphertext of a plaintext that exceeds the range of measurement bits, i.e. $\mathbb{B}$. Recall that any angle is a multiple of $\frac{\pi}{4}$. Therefore, an angle can be encoded into a number in $[<8]$. For the reason, we assume the plaintext space of an encryption scheme is $[<8]$.

In fact, the above conditions are sufficient. No additional conditions are needed except for being homomorphic. The following functions are needed to compute the measurement angles. Hence, we assume that the encryption scheme is homomorphic with respect to these functions.

Definition 4.4.3. We define the family of functions $\mathbf{F}_{\mathrm{B}} = \left\{\gamma_{l,X,Z}^{p,q}\right\}_{l \in \mathbb{N}, X, Z \in [<l], p, q \in P}$ where each $\gamma_{l,X,Z}^{p,q}$ is a function from $[<8]^{l+m}$ to $[<8]$ such that

$$\gamma_{l,X,Z}^{p,q}(\{b_j\}_{0 \le j < l}) = p + (-1)^{\sum_{j \in X} b_j} q + 4 \sum_{j \in Z} b_j \pmod 8. \tag{4.10}$$

We obtain the functions from Equation (4.2) by classifying terms into those known by Alice and the others, and by dividing them by $\frac{\pi}{4}$.

Now, it is time to show an instantiation of our protocol. We use a classical public-key encryption scheme such that the plaintext space **PS** is [<8], and it is verifiable, rerandomisable, and $\mathbf{F}_{\mathrm{B}}$-homomorphic. With the encryption scheme, our protocol can be concretised as shown in Protocol 5 where n' is the preshared security parameter. Each time Alice or Bob is received a message, the party checks whether the received message is valid, and aborts if the party finds it is invalid. Because a third party can access these messages and compute V_E, V_D, and V_C, the party will be able to learn which party sends an invalid message when the protocol aborts for the reason. That will make a third party able to get a better guess than one that public verifiability gives. But, we do not enter into the details now.

We prove Protocol 5 is correct, blind, and verifiable. The proof is a reduction into the FK protocol. Recall that the differences between our protocol and the FK protocol are that messages are encrypted and that the traps are disclosed. The latter does not depend on a choice of an encryption scheme. For later discussions, our proof goes through a modified FK protocol, where Alice and Bob executes the FK protocol but Alice finally discloses the traps. The detail is shown in Protocol 6. This protocol can be understood as a variant of Protocol 5 with the trivial encryption scheme. Note that the trivial encryption scheme is trivially verifiable, rerandomisable, and $\mathbf{F}_{\mathrm{B}}$-homomorphic. First, we prove Protocol 6 is correct, blind, and verifiable to show the disclosure of the traps does not benefit evil Bob and hence never damage the property of the FK protocol.

Lemma 4.4.4. *Assume both Alice and Bob respect Protocol 6. The protocol never aborts. The distribution of an outcome on an input including a unitary operator U coincides with the distribution of the measurement results of $U|0\rangle$ provided U is computable in G'. Alice always accepts the outcome.*

Proof. Assume both Alice and Bob respect the protocol. Alice has the same information between the FK protocol and Protocol 6. Therefore, due to the correctness of the FK protocol, Alice obtains a correct outcome and accepts it if the protocol is not aborted. We show Bob does not abort the protocol. Because Alice is honest, he obtains the true T and $\{r_t\}_{t\in T}$. Hence, Bob aborts if and only if Alice rejects the outcome. Therefore, Bob never aborts. □

Lemma 4.4.5. *Protocol 6 is blind while leaking at most G and t.*

Proof. In Protocol 6, all classical information obtained by Bob is G, $\{\delta_i\}_{i\in V}$, $\{b_i\}_{i\in V}$, T, and $\{r_t\}_{t\in T}$. Therefore, he learns about G and $t = |T|$. T is uniformly randomly chosen from the vertices of G, and each r_t for $t \in T$ is also uniformly randomly chosen from $\mathbb{B}$. The blindness of the FK protocol tells us that $\{\delta_i\}_{i\in V}$ and $\{b_i\}_{i\in V}$ are determined by some distribution for fixed G. (In fact, it is the uniform distribution.) Hence, the distribution of all classical information is determined by G and t.

Fix the classical information. A state of any qubit not in T or $N_G(T)$ is the same as of the qubit in the FK protocol. Let a state $|q_i\rangle$ be such that $i \in T$. Then, $|q_i\rangle = |+_{\theta_i}\rangle = |+_{\delta_i - \pi r_i}\rangle$ and thus $|q_i\rangle$ is fixed. Let $i \in N_G(T)$. $|q_i\rangle = |d_i\rangle$ with randomly chosen d_i. d_i is independently chosen. Therefore, the state of i is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{1}{2}\mathrm{I}$. □

Lemma 4.4.6. *Protocol 6 is ϵ_{FK}-verifiable.*

Proof. Assume not. Thus, there exists a strategy of Bob such that the probability that honest Alice accepts an incorrect outcome exceeds ϵ_{FK}. Suppose Bob

Protocol 5 Verifiable blind quantum computation protocol with encryption

Input

- Natural numbers n, t.
- A graph $G = (V, E)$ where vertices are labelled from 0 to $|V| - 1$.
- A description of a unitary operator U on n qubits.

Alice sends Bob G.
Alice randomly selects t vertices T.
Let $G' = (V', E')$ be the graph obtained by removing T and $N_G(T)$ from G, and $f_{G'}$ be its flow.
Alice computes the angles $\{\phi_i\}_{i \in V'}$ to compute U where each ϕ_i belongs to $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$. Set $\phi_i = 0$ for any $i \in T \cup N_G(T)$. Let $p_i = \frac{4}{\pi}\phi_i$.
for all $i \in V$ **do**
    if $i \in N_G(T)$ **then**
        Alice randomly chooses d_i from $\mathbb{B}$.
    end if
    Alice randomly chooses k_i from $[<8]$. Let $\theta_i = \frac{\pi}{4}k_i$.
    Alice randomly chooses r_i from $\mathbb{B}$.
end for
for all $i \in V$ **do**
    Alice sends Bob a single qubit whose state is $|q_i\rangle$ defined by Equation (4.1).
end for
Bob creates the state $\prod_{(i,j)\in E} \mathrm{CZ}_{(i,j)} \bigotimes_{i \in V} |q_i\rangle$.
Bob generates keys $(e, d) \leftarrow G(1^{n'})$ and sends Alice the encryption key e.
Alice aborts if $V_E(n', e) = 0$.
for $i = 0$ to $|V| - 1$ **do**
    Alice computes α_i and sends Bob it where

$$\alpha_i \leftarrow C^{o,p_i}_{\gamma_{i,X_i,Z_i}}(e, \{\beta_j\}_{j<i}) \tag{4.11}$$

and

$$o = (-1)^{\sum_{j\in X_i} r_j} p_i + k_i + 4r_i + 4\sum_{j \in Z_i} r_j \pmod 8. \tag{4.12}$$

    Bob aborts if $V_C(n', e, \alpha_i) = 0$.
    Bob computes $l_i \leftarrow D(e, d, \alpha_i)$. Let $\delta_i = \frac{\pi}{4}l_i$.
    Bob measures the ith qubit with δ_i and obtains b_i.
    Bob computes $\beta_i \leftarrow E(e, b_i)$ and sends Alice β_i.
    Alice aborts if $V_C(n', e, \beta_i) = 0$.
end for
Alice sends T and $\{r_t\}_{t\in T}$ to Bob.
Bob aborts if $b_t \neq r_t$ for some $t \in T$.
Bob sends Alice the decryption key d.
Alice aborts if $V_D(n', e, d) = 0$.
Alice computes $b_i \leftarrow D(e, d, \beta_i)$ for all $i \in V$.
Alice accepts the outcome if $b_t = r_t$ for all $t \in T$.

Protocol 6 The FK protocol with disclosure of trap

Input

- Natural numbers n, t.
- A graph $G = (V, E)$ where vertices are labelled from 0 to $|V| - 1$.
- A description of a unitary operator U on n qubits.

1: Alice sends Bob G.
2: Alice randomly selects t vertices T.
3: Let $G' = (V', E')$ be the graph obtained by removing T and $N_G(T)$ from G, and $f_{G'}$ be its flow.
4: Alice computes the measurement angles $\{\phi_i\}_{i \in V'}$ to compute U where each ϕ_i belongs to $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$. Set $\phi_i = 0$ for any $i \in T \cup N_G(T)$.
5: **for all** $i \in V$ **do**
6: **if** $i \in N_G(T)$ **then**
7: Alice randomly chooses d_i from $\mathbb{B}$.
8: **end if**
9: Alice randomly chooses θ_i from $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$.
10: Alice randomly chooses r_i from $\mathbb{B}$.
11: **end for**
12: **for all** $i \in V$ **do**
13: Alice sends Bob a single qubit whose state is $|q_i\rangle$ defined by Equation (4.1).
14: **end for**
15: Bob creates the state $\prod_{(i,j) \in E} \mathrm{CZ}_{(i,j)} \bigotimes_{i \in V} |q_i\rangle$.
16: **for** $i = 0$ to $|V| - 1$ **do**
17: Alice sends Bob δ_i defined by Equation (4.2).
18: Bob measures the ith qubit with δ_i.
19: Bob sends Alice the measurement result b_i.
20: **end for**
21: Alice sends T and $\{r_t\}_{t \in T}$ to Bob.
22: Bob aborts if $b_t \neq r_t$ for some $t \in T$.
23: Alice accepts the outcome if $b_t = r_t$ for all $t \in T$.

succeeds in making Alice accept an incorrect outcome with the strategy. Because Alice is honest, it means $b_t = r_t$ for any trap qubit t. Moreover, that Alice accepts means that Bob does not abort. Notice that Bob cannot do anything after receiving T and $\{r_t\}_{t \in T}$ except aborting. He already sends all b_i so he cannot change any of them. Therefore, Bob can use the same strategy in the FK protocol and achieves the same probability. It contradicts the definition of ϵ_{FK}. □

Precisely speaking, Protocol 6 is weaker than the FK protocol because it leaks the number of traps t. It allows Bob to guess a more precise upper bound of the size of the computation of Alice. If t is determined by G, Protocol 6 will be the same as the FK protocol. Indeed, for the dotted-complete graph G, t is fixed. Therefore, We do not care about the new leak.

Now, we prove that Protocol 5 is a verifiable BQC protocol. Recall we use a verifiable, rerandomisable, and $\mathbf{F}_{\mathrm{B}}$-homomorphic classical public-key encryption scheme such that the message space **PS** is $[<8]$.

Lemma 4.4.7. *Assume both Alice and Bob respect Protocol 5. The protocol never aborts. The distribution of an outcome on an input including a unitary operator U is the same as the distribution of the measurement results of $U|0\rangle$ provided U is computable in G'. Alice always accepts the outcome.*

Proof. Assume both Alice and Bob respect the protocol. All steps until Bob creates the graph state are the same in Protocol 5 and Protocol 6.

$$\begin{aligned}
&\frac{\pi}{4}D(e,d,\alpha_i) = \frac{\pi}{4}D(e,d,C_{\gamma^{o,p_i}_{i,X_i,Z_i}}(e,\{\beta_j\}_{j<i})) \\
&= \frac{\pi}{4}D(e,d,C_{\gamma^{o,p_i}_{i,X_i,Z_i}}(e,\{E(e,b_j)\}_{j<i})) = \frac{\pi}{4}\gamma^{o,p_i}_{i,X_i,Z_i}(\{b_j\}_{j<i}) \\
&= \frac{\pi}{4}\left((-1)^{\sum_{j\in X_i} r_j}p_i + k_i + 4r_i + 4\sum_{j\in Z_i} r_j + (-1)^{\sum_{j\in X_i} b_j}p_i + 4\sum_{j\in Z_i} b_j \pmod 8\right) \\
&= (-1)^{\sum_{j\in X_i} r_j}\phi_i + \theta_i + \pi r_i + \pi\sum_{j\in Z_i} r_j + (-1)^{\sum_{j\in X_i} b_j}\phi_i + \pi\sum_{j\in Z_i} b_j \pmod{2\pi} \\
&= \delta_i \qquad (4.13)
\end{aligned}$$

Therefore, Bob correctly obtains δ_i defined by Equation (4.2). Similarly, Alice obtains $D(e,d,E(e,b_i)) = b_i$ and therefore she obtains a correct outcome unless that the protocol aborts. Since Alice and Bob are honest, an encryption key, a decryption key, and all ciphertexts are valid. By Equations 4.7, 4.8, and 4.9, they do not abort. Because honest Alice and Bob never abort in Protocol 6, they never do in the protocol. □

Lemma 4.4.8. *Protocol 5 is blind while leaking at most G and t.*

Proof. We will show the classical information obtained by Bob in Protocol 5 is essentially the same as that in Protocol 6. Because the quantum state are the same, that implies the protocol is blind.

All Bob obtains is G, $\{\alpha_i\}_{i\in V}$, $\{b_i\}_{i\in V}$, T, and $\{r_t\}_{t\in T}$. The difference between both protocols is that $\{\alpha_i\}_{i\in V}$ and $\{\delta_i\}_{i\in V}$. Take α_i. Bob has the ciphertext, so Alice does not find $V_E(n,e) = 0$ or $V_C(n,e,\beta_j)$ for any $j < i$, because of equations 4.7 and 4.9. β_i is a ciphertext of a plaintext in [<8].

$$\begin{aligned}
&\gamma^{p,q}_{l,X,Z}(\{b_j + 2k_j\}_{0\le j<l}) = p + (-1)^{\sum_{j\in X}(b_j+2k_j)}q + 4\sum_{j\in Z}(b_j + 2k_j) \pmod 8 \\
&= p + (-1)^{\sum_{j\in X} b_j}q + 4\sum_{j\in Z} b_j \pmod 8 = \gamma^{p,q}_{l,X,Z}(\{b_j\}_{0\le j<l}). \qquad (4.14)
\end{aligned}$$

Therefore, we can regard each β_i as a ciphertext of 0 or 1. α_i is a ciphertext of δ_i. Since an encryption scheme is rerandomisable, α_i is uniformly distributed on the set of ciphertexts of δ_i. □

Lemma 4.4.9. *Protocol 5 is ϵ_{FK}-verifiable.*

Proof. Assume Alice respects the protocol and that Protocol 5 is not ϵ_{FK}-verifiable. Therefore, there exists an input and a strategy of Bob such that the probability of Alice accepting an incorrect outcome is larger than ϵ_{FK}. Sending invalid messages will be detected and will decrease the succeeding probability. Therefore, we ignore such strategies.

Then, Alice and Bob executes the Protocol 6. Bob simulates the above strategy. When a message is come from Alice, he encrypts it, rerandomise it, and executes the strategy. After finishing the execution, he decrypts the result and the residue of the plaintext modulo 2 to Alice.

Due to the proof of blindness and correctness, if Alice executes Protocol 6 honestly, he can honestly executes Protocol 5 with his simulated strategy. In particular, when he decides to accept an outcome, Alice also accepts the outcome.

Therefore, a strategy of Bob in Protocol 5 can be interpreted as one in Protocol 6. It means that he succeeds in making Alice accept an incorrect outcome with a larger probability than ϵ_{FK}. It contradicts the definition of ϵ_{FK}. □

Theorem 4.4.10. *Protocol 5 is a verifiable BQC protocol.* □

We proved an encryption scheme preserves the property of the FK protocol if the scheme satisfies several conditions. However, we do not know whether such encryption scheme exists other than the trivial encryption scheme. The conditions are sufficient but unnecessarily strong. In the rest of the section, we weaken the conditions.

Let us recall that we required an encryption scheme to be verifiable. The purpose was that the ciphertext Alice sends does not leak any information other than the desired measurement angle. As long as the purpose is achieved, we can weaken the definition of verifiability. In the definition of verifiability, we required the existence of polynomial-time algorithms that never output wrong answers. Because of correctness, we cannot admit any false negative error. That is, Alice has to judge a valid encryption key, a valid ciphertext, and a valid decryption key to be valid. However, we can admit false positive errors for verification of an encryption key and a ciphertext provided that no information is leaked. On the other hand, we cannot admit false positive errors for a decryption key. If so, there is a possibility that Alice overlooks an invalid decryption key, and that increases a probability of Alice accepting an incorrect outcome. Similarly, we have to care about the case such that there exists two decryption keys for an encryption key, and there exists a bit string such that the decryption outputs different strings when different keys are used. That allows evil Bob to choose which key he discloses and to secretly change the plaintext of the sent messages. Note that for any valid ciphertext, that never happens.

Proposition 4.4.11. *Let (G, E, D) be a public-key encryption scheme. For any $n \in \mathbb{N}$,*

$$\Pr\left[m = m' \;\middle|\; \begin{array}{l} (e,d),(e',d') \leftarrow G(1^n), e = e', \alpha \leftarrow \boldsymbol{CS}_{n,e} \\ m \leftarrow D(e,d,\alpha), m' \leftarrow D(e,d',\alpha) \end{array}\right] = 1 \tag{4.15}$$

Proof. Take $(e,d),(e',d') \in \mathbf{KS}_n$ and assume $e = e'$. Because $\alpha \in \mathbf{CS}_{n,e}$, there exists $m \in P$ such that $\alpha \in \mathbf{CS}_{n,e,m}$. The definition of encryption schemes,

$$D(e,d,\alpha) = D(e,d,E(e,m)) = m = D(e,d',E(e,m)) = D(e,d',\alpha). \tag{4.16}$$

□

However, we now admit false positive errors for a ciphertext. That does not satisfy the premise of the above proposition.

Definition 4.4.12. Let $\mathcal{E} = (G, E, D)$ be an F-homomorphic classical public-key encryption scheme. We say $\mathcal{E}$ is *covered* if there exist probabilistic polynomial-time algorithms V_{CE} and V_{CC} and polynomial-time algorithms V_{CED} and V_{CCK} that outputs either 0 or 1 and satisfies the following.

- For any $n \in \mathbb{N}$, $\Pr[v = 1 \mid (e,d) \leftarrow G(1^n), v \leftarrow V_{CE}(n,e)] = 1$.
- For any $n \in \mathbb{N}$, $\Pr[v = 1 \mid (e,d) \leftarrow G(1^n), \alpha \leftarrow \mathbf{CS}_{n,e}, v \leftarrow V_{CC}(e,\alpha)] = 1$.
- For any $n \in \mathbb{N}$ and $e', d' \in \mathbb{B}^*$,

$$V_{CED}(n,e',d') = \begin{cases} 1 & ((e',d') \in \mathbf{KS}_n) \\ 0 & ((e',d') \notin \mathbf{KS}_n) \end{cases}. \tag{4.17}$$

- For any $n \in \mathbb{N}$, $(e,d) \in \mathbf{KS}_n$, and $\alpha' \in \mathbb{B}^*$,

$$V_{CCK}(e,d,\alpha') = \begin{cases} 1 & (\alpha' \in \mathbf{CS}_{n,e}) \\ 0 & (\alpha' \notin \mathbf{CS}_{n,e}) \end{cases} . \tag{4.18}$$

- For any $n, l \in \mathbb{N}$ and any $e' \notin \mathbf{EKS}_n$ such that $V_{WE}(n,e) = 1$, there exists a distribution on the bit strings that satisfies the following.
  - For any $f \in F$ whose arity is l and for any bit string $\alpha_0, \ldots, \alpha_{l-1} \in \mathbb{B}^*$, $C_f(e', \alpha_0, \ldots, \alpha_{l-1})$ obeys the distribution.
- For any $n, l \in \mathbb{N}$, any $e \in \mathbf{EKS}_n$, and any $\alpha \notin E(e,P)$ such that $V_{WC}(e,\alpha) = 1$, there exists a distribution on the bit strings that satisfies the following.
  - For any $f \in F$ whose arity is l and for any $\alpha_0, \ldots, \alpha_{l-1} \in \mathbb{B}^*$ such that $\alpha_i = \alpha$ for some i, $C_f(e, \alpha_0, \ldots, \alpha_{l-1})$ obeys the distribution.

If evil Bob sends an invalid encryption key, he will receive a random bit string that obeys a predetermined distribution. Similarly, when Bob sends an invalid encryption key, he will also receive a random bit string. He cannot obtain new information, sending an invalid message. The reader may wonder if evil Bob can learn whether an invalid message is used to compute the measurement angle. That is true but not a problem. This is because he cannot learn which function Alice uses. Bob knows each measurement angle is computed from his messages. In particular, it can be computed from all messages. Hence, if Alice always computes the angles with all received messages, he cannot learn anything new.

With these conditions, we can prove that a variant of Protocol 5 is a verifiable BQC protocol by the same arguments. For completeness, we show the detail in Protocol 7 and show a proof.

Theorem 4.4.13. *Let $\mathcal{E}$ be a covered, rerandomisable, and $\mathbf{F}_{\mathrm{B}}$-homomorphic classical public-key encryption scheme such that the plaintext space is $[<8]$. Protocol 7 is a verifiable BQC protocol.*

Proof. Correctness immediately follows from the correctness of Protocol 5 and the fact we do not admit any false negative errors (Definition 4.4.12). For blindness and verifiability, assume Alice is honest. All classical information obtained by Bob is G, $\{\alpha_i\}_{i\in V}$, $\{b_i\}_{i\in V}$, T, and $\{r_t\}_{t\in T}$. If he sends an invalid encryption key or invalid ciphertexts, α_i obeys a fixed distribution. It is predetermined, so it is independent of an input. Therefore, the protocol is blind.

Evil Bob can send invalid messages. However, using V_{CED} and V_{CCK}, Alice finally finds they are invalid. If she finds, she aborts. She never accepts. Therefore, Bob cannot use invalid messages to make Alice accept an incorrect outcome. Therefore, the success probability is not larger than one in Protocol 5. □

Finally, we point out that we can change the plaintext space from $[<8]$ to $\mathbb{B}$ without affecting the properties. It just needs to change the definition of $\mathbf{F}_{\mathrm{B}}$.

4.5 Publicly verifiable blind quantum computation protocol

In the previous section, we left the security of an encryption scheme. Indeed, as pointed out before, the trivial encryption scheme satisfies all conditions proposed in the previous section. It is time to discuss about the security. Here, we discuss about it and give an instantiation of our protocol, which is publicly verifiable BQC protocol.

Protocol 7 Verifiable blind quantum computation protocol with encryption satisfying weaker conditions

Input

- Natural numbers n, t.
- A graph $G = (V, E)$ where vertices are labelled from 0 to $|V| - 1$.
- A description of a unitary operator U on n qubits.

Alice sends Bob G.
Alice randomly selects t vertices T.
Let $G' = (V', E')$ be the graph obtained by removing T and $N_G(T)$ from G, and $f_{G'}$ be its flow.
Alice computes the angles $\{\phi_i\}_{i \in V'}$ to compute U where each ϕ_i belongs to $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$. Set $\phi_i = 0$ for any $i \in T \cup N_G(T)$. Let $p_i = \frac{4}{\pi}\phi_i$.
for all $i \in V$ **do**
    if $i \in N_G(T)$ **then**
        Alice randomly chooses d_i from $\mathbb{B}$.
    end if
    Alice randomly chooses k_i from $[<8]$. Let $\theta_i = \frac{\pi}{4}k_i$.
    Alice randomly chooses r_i from $\mathbb{B}$.
end for
for all $i \in V$ **do**
    Alice sends Bob a single qubit whose state is $|q_i\rangle$ defined by Equation (4.1).
end for
Bob creates the state $\prod_{(i,j) \in E} \mathrm{CZ}_{(i,j)} \bigotimes_{i \in V} |q_i\rangle$.
Bob generates keys $(e, d) \leftarrow G(1^{n'})$ and sends Alice the encryption key e.
Alice aborts if $V_{CE}(n', e) = 0$.
for $i = 0$ to $|V| - 1$ **do**
    Alice computes α_i defined by Equation (4.11) and sends Bob it.
    Bob aborts if $V_{CCK}(e, d, \alpha_i) = 0$.
    Bob computes $l_i \leftarrow D(e, d, \alpha_i)$. Let $\delta_i = \frac{\pi}{4}l_i$.
    Bob measures the ith qubit with δ_i and obtains b_i.
    Bob computes $\beta_i \leftarrow E(e, b_i)$ and sends Alice β_i.
    Alice aborts if $V_{CC}(e, \beta_i) = 0$.
end for
Alice sends T and $\{r_t\}_{t \in T}$ to Bob.
Bob aborts if $b_t \neq r_t$ for some $t \in T$.
Bob sends Alice the decryption key d.
Alice aborts if $V_{CED}(n', e, d) = 0$.
Alice aborts if $V_{CCK}(e, d, \beta_i) = 0$ for some $i \in V$.
Alice computes $b_i \leftarrow D(e, d, \beta_i)$ for all $i \in V$.
Alice accepts the outcome if $b_t = r_t$ for all $t \in T$.

What should an encryption scheme satisfy to be secure against Alice? In the definition of public verifiability, we assume that Alice is a polynomial-time Turing machine. The plaintexts of the ciphertexts obtained by Alice are the measurement results. The distribution of them is determined by the measurement angles sent by Alice. She sends them after receiving an encryption key. Therefore, she can adjust the distribution for the key. It seems that it is enough that we use an IND-CPA secure encryption scheme. The security guarantees that a plaintext is concealed from Alice, even if she determines a distribution of plaintexts using the knowledge of an encryption key. In fact, it is not sufficient. In our setting, Alice can do more. That comes from the fact that the plaintexts, i.e. the measurement results are outputs of quantum operations. These operations are chosen by Alice. Therefore, Alice can choose distribution of plaintexts using Bob's quantum power. Precisely speaking, Alice cannot freely choose the plaintexts due to the probabilistic nature of quantum measurement, and each measurement result is randomly chosen. However, the whole measurement results may correlate with the outputs of quantum operations. The IND-CPA security assumes that the plaintext is chosen by a polynomial-time algorithm, so our situation is out of its scope. Some IND-CPA secure encryption schemes allow quantum computers to compute their decryption keys from their encryption keys. For example, although the ElGamal encryption scheme is known to be IND-CPA secure, a quantum computer can obtain its decryption key because it can efficiently compute discrete logarithms. For these schemes, plaintexts may depend on or be itself their decryption keys. Such attacks are known to be key-dependent message (KDM) attacks [14, 71]. Roughly speaking, an encryption scheme is *KDM secure with respect to a set of functions* F if any attacker cannot distinguish ciphertexts of $f(d)$ from ciphertexts of some fixed plaintext where f is any function in F and d is the decryption key. We can conclude that the encryption scheme should be KDM secure with respect to all quantum computations.

A problem is that it is too difficult to find an encryption scheme that is KDM secure and satisfies the conditions shown in the previous section. Stronger security tends to need an encryption scheme having a more complex structure. On the other hand, such a structure makes it difficult for a scheme to be homomorphic and to be tolerant to invalid attacks, i.e., to be covered. Hence, we employ the opposite way. We impose a weaker requirement on the security of the encryption scheme but do a stronger requirement than being homomorphic. The idea is as follows. The reason why we have to care about KDM attacks is that Alice can decide measurement angles after receiving an encryption key. Therefore, if an encryption key is chosen after Bob receives a measurement angle, there is no possibility of such attacks. Any encryption key is definitely independent of a measurement result, a plaintext. However, that means each measurement result is encrypted by a different encryption key because each measurement result has to be encrypted before receiving the measurement angle of the next qubit. Alice has to compute a ciphertext of a measurement angle from measurement results encrypted by different keys. Now, being homomorphic is not sufficient.

Definition 4.5.1. Let $\mathcal{E} = (G, E, D)$ be a classical encryption scheme whose plaintext spaces are independent of the security parameter. Let F be a set of functions on the plaintext space. Let L_F be the set of the arities of functions in F, that is $L_F = \{\, l \mid f\colon \mathbf{PS}^l \to \mathbf{PS} \in F \,\}$. $F_l = \{\, f \in F \mid f\colon \mathbf{PS}^l \to \mathbf{PS} \,\}$. We say $\mathcal{E}$ is *unitedly homomorphic with respect to* F, or *unitedly* F*-homomorphic*, if there exist probabilistic polynomial-time algorithms $\{E_{H,f}\}_{f\in F}$ and polynomial-time algorithms $\{D_{H,l}\}_{l\in L_F}$ such that

- $E_{H,f\in F_l}$ computes a bit string from $2l$ bit strings,
- $D_{H,l}$ computes a bit string from $2l+1$ bit strings, and
- for any $f \in F_l$ and any $n \in \mathbb{N}$,

$$\Pr\left[m = m' \;\middle|\; \begin{array}{l} (e_0,d_0),\ldots,(e_{l-1},d_{l-1}) \leftarrow G(1^n) \\ m_0,\ldots,m_{l-1} \leftarrow \mathbf{PS} \\ \alpha_0 \leftarrow E(e_0,m_0) \\ \vdots \\ \alpha_{l-1} \leftarrow E(e_{l-1},m_{l-1}) \\ \beta \leftarrow E_{H,f}(\{e_i\}_{i<l},\{\alpha_i\}_{i<l}) \\ m \leftarrow D_{H,l}(\{e_i\}_{i<l},\{d_i\}_{i<l},\beta) \\ m' \leftarrow f(\{m_i\}_{i<l}) \end{array} \right] = 1 \tag{4.19}$$

- for any $f \in F_l$, any $n \in \mathbb{N}$, any $(e_0,d_0),\ldots,(e_{l-1},d_{l-1}) \in \mathbf{KS}$, any $m_0,\ldots,m_{l-1} \in \mathbf{PS}$, and any $\alpha_0 \in \mathbf{CS}_{n,e_0,m_0},\ldots,\alpha_{l-1} \in \mathbf{CS}_{n,e_{l-1},m_{l-1}}$, $E_{H,f}(\{e_i\}_{i<l},\{\alpha_i\}_{i<l})$ is the uniform distribution on the set $C_{m,\{e_i\}_{i<l}}$ where m is $f(\{m_i\}_{i<l})$,

$$C_{m,\{e_i\}_{i<l}} = \bigcup_{\substack{g\in F_l \\ \text{s.t. } m=g(\{m_i\}_{i<l})}} S_g, \tag{4.20}$$

  and S_g is the support of the distribution of $E_{H,g}(\{e_i\}_{i<l},\{E(e_i,m_i)\}_{i<l})$.

Definition 4.5.2. Let $\mathcal{E} = (G,E,D)$ be a unitedly F-homomorphic classical public-key encryption scheme. We say $\mathcal{E}$ is *unitedly covered* if there exist probabilistic polynomial-time algorithms V_{CE} and V_{CC} and polynomial-time algorithms V_{CED}, V_{CCK}, and $\{V_{CCH,l}\}_{l\in L_F}$ that outputs either 0 or 1 and satisfies the first four requirements in Definition 4.4.12 and the following.

- For any $l \in L_F$, any $n \in \mathbb{N}$ and any $e' \notin \mathbf{EKS}_n$ such that $V_{WE}(n,e) = 1$, there exists a distribution on the bit strings that satisfies the following.
  - For any $f \in F_l$ and for any $e_0,\ldots,e_{l-1},\alpha_0,\ldots,\alpha_{l-1} \in \mathbb{B}^*$ such that $e_i = e'$ for some i, $E_{H,f}(\{e_i\}_{i<l},\{\alpha_i\}_{i<l})$ obeys the distribution.
- For any $l \in L_F$, any $n \in \mathbb{N}$, any $e,e_0,\ldots,e_{l-1} \in \mathbf{EKS}_n$, and any $\alpha \notin \mathbf{CS}_{n,e}$ such that $V_{WC}(e,\alpha) = 1$, there exists a distribution on the bit strings that satisfies the following.
  - For any $f \in F_l$ and for any $\alpha_0,\ldots,\alpha_{l-1} \in \mathbb{B}^*$ such that $\alpha_i = \alpha$ and $e_i = e$ for some i, $E_{H,f}(\{e_i\}_{i<l},\{\alpha_i\}_{i<l})$ obeys the distribution.
- For any $n \in \mathbb{N}$, any $l \in L_f$, any $(e_0,d_0),(e_{l-1},d_{l-1}) \in \mathbf{KS}_n$, and $\beta' \in \mathbb{B}^*$,

$$V_{CCH,l}(\{e_i\}_{i<l},\{d_i\}_{i<l},\beta') = \begin{cases} 1 & (\beta' \in \bigcup_{m\in\mathbf{PS}} C_{m,\{e_i\}_{i<l}}) \\ 0 & (\beta' \notin \bigcup_{m\in\mathbf{PS}} C_{m,\{e_i\}_{i<l}}) \end{cases}. \tag{4.21}$$

  where $C_{m,\{e_i\}_{i<l}}$ is defined by Equation (4.20).

Under the conditions, our protocol is still a verifiable BQC protocol. Moreover, the protocol is publicly verifiable with some assumptions. Instead of C_f, we use $E_{H,f}$ so Protocol is slightly changed. We give a final instantiation of our protocol in Protocol 8. We assume that an encryption scheme is unitedly $\mathbf{F}_{\mathrm{B}}$-homomorphic, rerandomisable, and unitedly covered, the plaintext space is $[<8]$, and it has indistinguishable encryptions. We need not IND-CPA security because a plaintext is independent of an encryption key.

Protocol 8 Publicly Verifiable BQC protocol

Input

- Natural numbers n, t.
- A graph $G = (V, E)$ where vertices are labelled from 0 to $|V| - 1$.
- A description of a unitary operator U on n qubits.

Alice sends Bob G.
Alice randomly selects t vertices T.
Let $G' = (V', E')$ be the graph obtained by removing T and $N_G(T)$ from G, and $f_{G'}$ be its flow.
Alice computes the angles $\{\phi_i\}_{i \in V'}$ to compute U where each ϕ_i belongs to $\left\{ \frac{k\pi}{4} \mid k \in [<8] \right\}$. Set $\phi_i = 0$ for any $i \in T \cup N_G(T)$. Let $p_i = \frac{4}{\pi}\phi_i$.
for all $i \in V$ **do**
    if $i \in N_G(T)$ **then**
        Alice randomly chooses d_i from $\mathbb{B}$.
    end if
    Alice randomly chooses k_i from $[<8]$. Let $\theta_i = \frac{\pi}{4}k_i$.
    Alice randomly chooses r_i from $\mathbb{B}$.
end for
for all $i \in V$ **do**
    Alice sends Bob a single qubit whose state is $|q_i\rangle$ defined by Equation (4.1).
end for
Bob creates the state $\prod_{(i,j) \in E} \mathrm{CZ}_{(i,j)} \bigotimes_{i \in V} |q_i\rangle$.
for $i = 0$ to $|V| - 1$ **do**
    if $i = 0$ **then**
        Alice computes δ_0 and sends it to Bob where δ_0 is defined by Equation (4.2).
    else
        Alice computes $\alpha_i \leftarrow E_{H,f}(\{e_j\}_{j<i}, \{\beta_j\}_{j<i})$ and sends it where the function f is $\gamma^{o,p_i}_{i,X_i,Z_i}$ and o is defined by Equation (4.12).
        Bob aborts if $V_{CCH,i}(\{e_j\}_{j<i}, \{d_j\}_{j<i}, \alpha_i) = 0$.
        Bob computes $l_i \leftarrow D_{H,i}(e, d, \alpha_i)$. Let $\delta_i = \frac{\pi}{4}l_i$.
    end if
    Bob measures the ith qubit with δ_i and obtains b_i.
    Bob generates keys $(e_i, d_i) \leftarrow G(1^{n'})$.
    Bob computes $\beta_i \leftarrow E(e_i, b_i)$ and sends Alice e_i and β_i.
    Alice aborts if $V_{CE}(n', e_i) = 0$ or $V_{CC}(e_i, \beta_i) = 0$.
end for
Alice sends T and $\{r_t\}_{t \in T}$ to Bob.
Bob aborts if $b_t \neq r_t$ for some $t \in T$.
Bob sends Alice $\{d_i\}_{i \in V}$.
Alice aborts if $V_{CED}(n', e_i, d_i) = 0$ for some i.
Alice aborts if $V_{CCK}(e_i, d_i, \beta_i) = 0$ for some i.
Alice computes $b_i \leftarrow D(e_i, d_i, \beta_i)$ for all $i \in V$.
Alice accepts the outcome if $b_t = r_t$ for all $t \in T$.

Public verifier

- A public verifier accepts if Bob sends all decryption keys, $V_{CED}(n, e_i, d_i) = 1$ and $V_{CCK}(e_i, d_i, \beta_i) = 1$ for any i, and $b_t = r_t$ for all $t \in T$.

Theorem 4.5.3. *Let $\mathcal{E}$ be a unitedly covered, rerandomisable, and unitedly $\mathbf{F}_{\mathrm{B}}$-homomorphic classical public-key encryption scheme such that the plaintext space is $[<8]$. Protocol 8 is a verifiable BQC protocol.*

Proof. The protocol does not abort when all parties are honest because we do not admit any false negative errors and Protocol 7 is correct. Alice obtains a correct outcome and accepts it, due to the correctness of Protocol 7 and Equation (4.19). Note that a public verifier accepts because the verifier does the same as Alice does and honest Alice accepts an outcome. Therefore, the protocol is correct. A proof of blindness and verifiability is the same as the proof for Protocol 7 mutatis mutandis. □

Now, we show our protocol is publicly verifiable. We impose two additional assumptions. One is the number of qubits is polynomial. The other is that Alice cannot create any entangled state. The former is natural. If the number of qubits is superpolynomial, Alice cannot send these qubits. The latter is imposed to prevent Alice from directly reading the measurement results. If Alice can create a pair of qubits in an entangled state, she will send the half to Bob, and establish a secret quantum channel to Bob's quantum system. Our protocol relies on an assumption that Alice has no information about measurement results except for the ciphertexts. It is difficult to estimate an upper bound of leakage information via a secret quantum channel. Finally, note that even if Alice cannot create any entangled state, she can execute our protocol.

Theorem 4.5.4. *Let $\mathcal{E}$ be a unitedly covered, rerandomisable, and unitedly $\mathbf{F}_{\mathrm{B}}$-homomorphic classical public-key encryption scheme such that the plaintext space is $[<8]$, and it has indistinguishable encryptions. Moreover, we assume that the numbers of vertices of graphs are polynomially bounded with respect to the security parameter n', and that Alice cannot create any entangled state. Protocol 8 is ϵ_{FK}-publicly verifiable.*

Proof. The first condition of public verifiability holds with $\xi = \epsilon_{\mathrm{FK}}$ due to the fact that a public verifier accepts if and only if honest Alice accepts an outcome.

We will prove that the protocol satisfies the second condition. Assume that Bob is honest. First, we note that Alice does not have any decryption key when a public verifier rejects. When the verifier rejects, Bob does not disclose the decryption keys, $V_{CED}(n, e_i, d_i) = 1$ or $V_{CCK}(e_i, d_i, \beta_i) = 1$ for some $i \in V$, or $b_t \neq r_t$ for some $t \in T$. If Bob sends the decryption keys, he confirms $b_t = r_t$ for any $t \in T$. Since he is honest, $V_{CED}(n, e_i, d_i)$ and $V_{CCK}(e_i, d_i, \beta_i)$ are always 1. Therefore, Bob does not send the decryption keys. He aborts the protocol somewhere. Next, we note that a strategy of Alice is definitely classical. Alice is assumed to be unable to create any entangled states. Therefore, all qubits sent by her is single qubits, and she does not have any secret quantum channel to Bob. Finally, we note that Alice gains nothing by sending invalid ciphertexts. Even if she does, Bob immediately finds that it is invalid because of the definition of $V_{CCH,i}$, and he aborts. Moreover, since a public verifier always rejects when Bob aborts, that cannot make a public verifier accept. Therefore, we assume that Alice never send invalid ciphertexts in the protocol.

Fix a strategy of Alice. Let A be a probabilistic polynomial-time algorithm running Protocol 8 with honest Bob that implements the strategy. We will show a probabilistic polynomial-time algorithm B simulates A. Let A^i be the same algorithm as A except that Bob always sends a ciphertext of zero after he sends a ciphertext of the ith measurement result. $A = A^{t(n')}$ where $t(n')$ is the number

of qubits with polynomially-bounded function t. Let $X_{n'}$ be an input of the protocol, and h, f be polynomially-bounded functions. Define $p^{(i)}$ as follows.

$$p^{(i)} = \Pr\left[v = w \;\middle|\; \begin{array}{l} a \leftarrow h(1^{n'}, X_{n'}), (v, m) \leftarrow A^i(1^{n'}, X_{n'}, a) \\ j \leftarrow P(m), j = \text{reject}, w \leftarrow f(1^{n'}, X_{n'}) \end{array}\right] \tag{4.22}$$

Because an encryption scheme has indistinguishable encryptions, $\left|p^{(i+1)} - p^{(i)}\right|$ is negligible for any i. Therefore, $\left|p^{(t(n'))} - p^{(0)}\right|$ is still negligible. In $A^{(0)}$, Alice receives ciphertexts of zeros from Bob. These ciphertexts can be computed herself. Therefore, simulating Bob's behaviour in A^0, we obtain a probabilistic polynomial-time algorithm B. By definition, it achieves the same probability $p^{(0)}$. □

4.6 Encryption scheme

We showed there exists a publicly verifiable BQC protocol provided that there exists an encryption scheme that satisfies several conditions. In the section, we show there exists such a good scheme.

4.6.1 Inattentive evaluation of circuits

From the view of Bob, communication with Alice can be understood as secret computation: He sends an encrypted bit. Then, she applies a secret function to it, and sends back the output. Equivalently, it can be understood as a cooperative evaluation of a known function with private inputs. There is a known function. Alice and Bob has their own private inputs. They compute the output of the function without letting each party learn the input of the other party. These tasks are known as private computation, secure circuit evaluation, and so on. Inattentive evaluation of circuits [95] is a method to tackle the task. Here, we review the method.

Suppose the following task. Bob has private bits $x_0, \ldots, x_{n-1}$. Alice has a private circuit C. The circuit is implemented by OR gates and NOT gates. Each input is assumed to have the same depth, that is the number of applied OR gates. That is, an input is, for example, a form of $\neg(P_0 \vee P_1) \vee (P_2 \vee P_3)$, not $\neg P_0 \vee (P_1 \vee P_2)$. The latter can be rewritten as $(\neg P_0 \vee \neg P_0) \vee (P_1 \vee P_2)$. Now, Alice and Bob want to obtain $C(x_0, \ldots, x_{n-1})$. The encoding is performed inductively. Sets $\left\{\mathbf{Enc}^k\right\}_{k\in\mathbb{N}}$, $\left\{\mathbf{Add}^{k+1}\right\}_{k\in\mathbb{N}}$, $\left\{\mathbf{Enc}_b^k\right\}_{b\in\mathbb{B},k\in\mathbb{N}}$, and $\left\{\mathbf{Add}_b^{k+1}\right\}_{b\in\mathbb{B},k\in\mathbb{N}}$ are defined as follows.

$$\mathbf{Enc}^k = \mathbf{Enc}_0^k \cup \mathbf{Enc}_1^k \tag{4.23}$$

$$\mathbf{Add}^k = \mathbf{Add}_0^k \cup \mathbf{Add}_1^k \tag{4.24}$$

$$\mathbf{Enc}_b^0 = \{\, (b, 1-b) \,\} \tag{4.25}$$

$$\mathbf{Add}_b^{k+1} = \{\, (a_0, a_1) \mid a_i \in \mathbf{Enc}_{b_i}^k, b_0 + b_1 = b \pmod 2 \,\} \tag{4.26}$$

$$\mathbf{Enc}_b^{k+1} = \{\, (a_0, a_1, a_2, a_3) \mid a_i \in \mathbf{Add}_{b_i}^{k+1}, \text{ exactly three of } \{b_i\}_{i<4} \text{ is } b \,\} \tag{4.27}$$

The subscript b of each set represents that an element of the set encodes the bit b. For example, $(((0,1),(1,0)),((1,0),(1,0)),((0,1),(1,0)),((1,0),(0,1)))$ encodes a bit 1. Using these sets, the evaluation of circuits was implemented as shown in Algorithm 9. An element of $\mathbf{Enc}^k$ is said to be in the kth layer. The procedures $\textsc{Or}^k$ and $\textsc{Not}^k$ are evaluations of OR and NOT gates in the kth layer, respectively. Although $\textsc{Not}^k$ does not change the layer, $\textsc{Or}^k$ lifts it, that is the

Algorithm 9 Operations in inattentive evaluation of circuits [95]

procedure $\textsc{Or}^k(x \in \mathbf{Enc}^k,\ y \in \mathbf{Enc}^k)$
  Output $((x, 0_k), (y, 0_k), (x, y), (1_k, 0_k))$ where $b_k \in \mathbf{Enc}_b^k$.
end procedure

procedure $\textsc{Not}^0((a, b) \in \mathbf{Enc}^0)$
  Output (b, a).
end procedure
procedure $\textsc{Not}^{k+1}(((a_0, b_0), (a_1, b_1), (a_2, b_2), (a_3, b_3)) \in \mathbf{Enc}^{k+1})$
  Output $((\textsc{Not}^k(a_0), b_0), (\textsc{Not}^k(a_1), b_1), (\textsc{Not}^k(a_2), b_2), (\textsc{Not}^k(a_3), b_3))$.
end procedure

procedure $\textsc{Decode}^0((a, b) \in \mathbf{Enc}^0)$
  Output a.
end procedure
procedure $\textsc{Decode}^{k+1}((a_0, a_1, a_2, a_3) \in \mathbf{Enc}^{k+1})$
  Output b such that exactly three of $\left\{\textsc{SubDecode}^{k+1}(a_i)\right\}_{i<4}$ is b.
end procedure

procedure $\textsc{SubDecode}^{k+1}((a, b) \in \mathbf{Add}^{k+1})$
  Output $\textsc{Decode}^k(a) + \textsc{Decode}^k(b) \pmod 2$.
end procedure

procedure $\textsc{Randomise}^0(b \in \mathbf{Enc}^0)$
  Output b.
end procedure
procedure $\textsc{Randomise}^{k+1}((a_0, a_1, a_2, a_3) \in \mathbf{Enc}^{k+1})$
  for $i = 0$ to 3 **do**
    $c_i \leftarrow \textsc{SubRandomise}^{k+1}(a_i)$
  end for
  $\sigma \leftarrow S_4$ where S_4 is the permutation group of order 4.
  Output $(c_{\sigma(0)}, c_{\sigma(1)}, c_{\sigma(2)}, c_{\sigma(3)})$.
end procedure

procedure $\textsc{SubRandomise}^{k+1}((a, b) \in \mathbf{Add}^{k+1})$
  $c \leftarrow \textsc{Randomise}^k(a), d \leftarrow \textsc{Randomise}^k(b)$.
  $r \leftarrow \mathbb{B}$.
  if $r = 0$ **then**
    Output (c, d).
  else
    Output $(\textsc{Not}^k(c), \textsc{Not}^k(d))$.
  end if
end procedure

domain and codomain of $\textsc{Or}^k$ are $\mathbf{Enc}^k \times \mathbf{Enc}^k$ and $\mathbf{Enc}^{k+1}$. $\textsc{SubDecode}^k$ and $\textsc{Decode}^k$ shows how to decode elements of $\mathbf{Add}^k$ and $\mathbf{Enc}^k$, respectively. Similarly, $\textsc{SubDecode}^k$ and $\textsc{Randomise}^k$ shows how to rerandomise elements. Then, receiving inputs as pairs $((x_0, 1-x_0), \ldots, (x_{n-1}, 1-x_{n-1}))$, Alice can obtain a tuple encoding $C(x_0, \ldots, x_{n-1})$. Randomise hides all information other than $C(x_0, \ldots, x_{n-1})$ from Bob. The method does not touch $x_0, \ldots, x_{n-1}$, so it works even if each x_i is encrypted. Let $\mathcal{E}$ be an IND-CPA secure, rerandomisable classical encryption scheme. The following protocol allows Alice and Bob to perform the task described above.

1: Bob generates a key $(e, d) \leftarrow G(1^n)$.
2: Bob encrypts his inputs $\alpha_i \leftarrow E(e, x_i)$ and its negation $\beta_i \leftarrow E(e, 1-x_i)$
3: Bob sends $(\alpha_0, \beta_0), \ldots, (\alpha_{n-1}, \beta_{n-1})$ Alice.
4: Alice applies Or and Not to the received inputs according to her circuit C.
5: Alice applies Randomise, and finally rerandomise each element of the output tuple.
6: Alice sends the tuple Bob.
7: Bob applies Decode and finally the decryption to the received output.

Theorem 4.6.1 ([95])**.** *Assume both parties are honest. The output of the above protocol is $C(x_0, \ldots, x_{n-1})$. Alice cannot distinguish the two possible inputs without a negligible probability in the sense of IND-CPA security. Bob learns nothing other than the output $C(x_0, \ldots, x_{n-1})$.*

The method gives perfect security to Alice. However, it has two flaws. One is that it needs much longer messages. An element of $\mathbf{Enc}^{k+1}$ consists of eight elements of $\mathbf{Enc}^k$. Therefore, the former is eight times as long as the latter. The method cannot evaluate circuits whose depth grows faster than logarithm, i.e., circuits that does not belong to $\mathbf{NC}^1$. In our protocol, Alice needs to evaluate a function with the received ciphertexts, the size of which grows linearly. The other problem is that it is not tolerant to invalid messages. The method requires Bob to send a pair of ciphertexts of different bits, i.e., $(0,1)$ or $(1,0)$. Evil Bob may send a pair of ciphertext of the same bits, i.e., $(0,0)$ or $(1,1)$. If an encryption scheme is IND-CPA secure, Alice cannot distinguish invalid pairs from valid pairs without negligible probability. Hence, even if the encryption scheme is verifiable, evil Bob can use invalid messages without being detected. In the original paper [95], the authors use non-interactive zero-knowledge proofs to detect such attacks. However, we cannot use this mode. Non-interactive zero-knowledge proofs require Alice and Bob to share a common random string, which is produced by a trustworthy party. We cannot assume the existence of such a party. If so, public verifiability is no longer meaningful, as pointed out at the end of Section 4.2.

4.6.2 Adjustment of inattentive evaluation of circuits

In order to use inattentive evaluation of circuits, we have to adjust the method to our protocol. First of all, we change the plaintext space of an encryption scheme from $[<8]$ to $\mathbb{B}$. As noted before, that causes nothing dangerous. A measurement angle belong to $[<8]$ but it can be expressed as three bits. The measurement result is already an element of $\mathbb{B}$ and, as shown in the proof of Lemma 4.4.8, any other element of $[<8]$ works as its parity. Therefore, the change does not influence even behaviours of an evil party. Alice computes measurement angles bitwisely.

Table 4.1: Patterns of $\gamma^{p,q}_{l,X,Z}$

ζ_0	ζ_1	ζ_2	ζ_3	Formula	a_{00}	a_{01}	a_{10}	a_{11}
0	0	0	0	0	0	0	0	0
b_X	b_Z	1	0	$\neg(b_X \vee b_Z)$	0	0	0	1
b_X	b_Z	b_X	0	$(b_X \vee b_Z) \oplus b_X$	0	0	1	0
$\neg b_X$	0	0	0	$\neg b_X$	0	0	1	1
b_X	b_Z	0	b_Z	$(b_X \vee b_Z) \oplus b_Z$	0	1	0	0
0	$\neg b_Z$	0	0	$\neg b_Z$	0	1	0	1
b_X	0	0	b_Z	$b_X \oplus b_Z$	0	1	1	0
$\neg b_X$	$\neg b_Z$	0	0	$\neg b_X \vee \neg b_Z$	0	1	1	1
$\neg b_X$	$\neg b_Z$	1	0	$\neg(\neg b_X \vee \neg b_Z)$	1	0	0	0
$\neg b_X$	0	0	b_Z	$\neg b_X \oplus b_Z$	1	0	0	1
0	b_Z	0	0	b_Z	1	0	1	0
b_X	b_Z	0	$\neg b_Z$	$(b_X \vee b_Z) \oplus \neg b_Z$	1	0	1	1
b_X	0	0	0	b_X	1	1	0	0
b_X	b_Z	$\neg b_X$	0	$(b_X \vee b_Z) \oplus \neg b_X$	1	1	0	1
b_X	b_Z	1	0	$\neg(b_X \vee b_Z)$	1	1	1	0
1	0	0	0	1	1	1	1	1

First, we focus on the first problem. For a while, we assume that Bob is honest in the sense that he does not send a pair of the same bits. Let us see Equation (4.10), which Alice needs to compute. In the equation, the inputs appear in the forms of bit summations and nothing else. Hence, if we prepare an encoding of a bit summation that has a variable length, the depth no longer grows linearly. Indeed, each bit of Equation (4.10) can be computed by evaluation of a constant-depth formula.

Proposition 4.6.2. *Let $\gamma^{p,q}_{l,X,Z}$ be a function in $\mathbf{F}_{\mathrm{B}}$. Although the output of the function is three bits, Alice wants to compute the jth bit of it for $0 \leq j < 2$. Then, there exists four elements $\zeta_0, \zeta_1, \zeta_2, \zeta_3$ of a set $\{ 0, 1, \mathbf{b}_X, \mathbf{b}_Z, \neg\mathbf{b}_X, \neg\mathbf{b}_Z \}$ such that for any $\{b_i\}_{i<l}$, the jth bit of $\gamma^{p,q}_{l,X,Z}(\{b_i\}_{i<l})$ is 1 if and only if $(\zeta_0 \vee \zeta_1) \oplus (\zeta_2 \vee \zeta_3)[\bigoplus_{i\in X} b_i, \bigoplus_{i\in Z} b_i / \mathbf{b}_X, \mathbf{b}_Z]$ is 1. Here, $P[A/B]$ means the substitution of A for B in P.*

Proof. By the definition of $\gamma^{p,q}_{l,X,Z}$, it can be written as $g(\oplus_{i\in X} b_i, \oplus_{i\in Z} b_i)$ for some function g. The output is 0 or 1. Hence, there exists four bits $a_{00}, a_{01}, a_{10}, a_{11}$ such that $g(b_X, b_Z) = a_{b_X b_Z}$. All patterns can be expressed by $(\zeta_0 \vee \zeta_1) \oplus (\zeta_2 \vee \zeta_3)$ with $\zeta_i \in \{ 0, 1, b_X, b_Z, \neg b_X, \neg b_Z \}$. See Table 4.1. The column "Formula" lists simplified $(\zeta_0 \vee \zeta_1) \oplus (\zeta_2 \vee \zeta_3)$. A boring check shows that in each row of the table, for any $\mathbf{b}_X, \mathbf{b}_Z \in \mathbb{B}$, $(\zeta_0 \vee \zeta_1) \oplus (\zeta_2 \vee \zeta_3) = 1$ if and only if $g(\mathbf{b}_X, \mathbf{b}_Z) = a_{\mathbf{b}_X \mathbf{b}_Z} = 1$. □

For an encoding of bit summation, we follow an idea of $\mathbf{Add}^{k+1}$. A pair in the set encodes the bit summation of its two elements. Instead of a pair, we use a sequence. $\bigoplus_{i\in X} b_i$ can be encoded to a sequence $(a_i)_{i<l}$ where a_i is b_i if $i \in X$ and otherwise 0. With the encoding, Alice can compute a measurement angle.

For the completeness, we show the details.

$$\mathbf{As}_b^0 = \mathbf{Enc}_b^0 \tag{4.28}$$

$$\mathbf{As}_b^{1,l} = \bigcup_{\bigoplus b_i = b} \prod_{i<l} \mathbf{As}_{b_i}^0 \tag{4.29}$$

$$\mathbf{As}_b^{2,l} = \left\{ ((a_i, c_i))_{i<4} \;\middle|\; \begin{array}{l} a_i \in \mathbf{As}_{b_i}^{1,l} c_i \in \mathbf{As}_{d_i}^{1,l} \\ \text{exactly three of } \{b_i \oplus d_i\}_{i<4} \text{ is } b \end{array} \right\} \tag{4.30}$$

$$\mathbf{As}_b^{3,l} = \bigcup_{b_0 \oplus b_1 = b} \mathbf{As}_{b_0}^{2,l} \times \mathbf{As}_{b_1}^{2,l} \tag{4.31}$$

where $l \in \mathbb{N}$. An element of $\mathbf{As}_b^k$ is said to be in the kth layer. The second layer is used to compute OR gates. In the first and third layers, bit summations are computed. Although it performs trivial work, we explicitly write the zeroth layer, because the layer will be replaced later. For each layer, $\textsc{eEncode}^k$ produces an element of $\mathbf{As}_b^k$ defined below. Suppose Alice has measurement results $\{(b_i, c_i)\}_{i<l}$ where $b_i \in \mathbb{B}$ and $c_i = 1 - b_i$. Moreover, she has $\zeta_0, \ldots, \zeta_3$ such that $P = (\zeta_0 \vee \zeta_1) \oplus (\zeta_2 \vee \zeta_3)$ expresses a bit of a measurement angle. Then, Alice computes $M = \textsc{eEncode}^3(P, \{(b_i, c_i)\}_{i<l})$, and sends an output of $\textsc{eRandomise}^3(M)$ to Bob. Bob computes $b = \textsc{eDecode}^3(M)$, which is a bit of the measurement angle. The operations $\textsc{eEncode}^3$, $\textsc{eRandomise}^3$, and $\textsc{eDecode}^3$ are shown in Algorithms 10 and 11. The lengths of elements of $\mathbf{As}_b^0$, $\mathbf{As}_b^{1,l}$, $\mathbf{As}_b^{2,l}$, and $\mathbf{As}_b^{3,l}$ are 2, $2l$, $16l$, and $32l$, respectively. Therefore, the length of message grows linearly and Alice can use the method to compute measurement angles. The method, of course, works correctly, and hides the formula from Bob. The following propositions trivially hold.

Proposition 4.6.3. *For any $b \in \mathbb{B}$ and $\zeta \in \{0, 1, \mathbf{b}\}$, $\textsc{eEncode}^0(\zeta, (b, 1-b))$ belongs to $\boldsymbol{As}_{\zeta[b/\mathbf{b}]}^0$.* □

Proposition 4.6.4. *Let $k \in \{1, 2, 3\}$, $l \in \mathbb{N}$, and X, Z be subsets of $[<l]$. Let P be a formula whose form is appropriate for the kth layer. Let $\{b_i\}_{i<l}$ be a bit sequence. Define $b = P[\bigoplus_{i \in X} b_i, \bigoplus_{i \in Z} b_i / \mathbf{b}_X, \mathbf{b}_Z]$. Then, $\boldsymbol{As}_b^{k,l}$ contains $\textsc{eEncode}^k(P, \{(b_i, 1-b_i)\}_{i<l})$.* □

Proposition 4.6.5. *Let $k \in [<4]$, $b \in \mathbb{B}$, and $l \in \mathbb{N}$. $\textsc{eNot}^k$ is an isomorphism between $\boldsymbol{As}_b^{k,l}$ and $\boldsymbol{As}_{1-b}^{k,l}$. Here, $\boldsymbol{As}_b^{0,l}$ means $\boldsymbol{As}_b^0$.* □

Proposition 4.6.6. *Let $k \in [<4]$, $b \in \mathbb{B}$, and $l \in \mathbb{N}$. For any $M \in \boldsymbol{As}_b^{k,l}$, $\textsc{eDecode}^k(M) = b$.* □

Proposition 4.6.7. *Let $k \in [<4]$, $b \in \mathbb{B}$, and $l \in \mathbb{N}$. There exists a distribution on $\boldsymbol{As}_b^{k,l}$ such that for any $M \in \boldsymbol{As}_b^{k,l}$, $\textsc{eRandomise}^k(M)$ is the distribution.*

Proof. We only show when $k = 1$. For the other layer, the statement holds due to Theorem 4.6.1.

Let $\{b_i\}_{i<l}$ be such that $M \in \prod_{i<l} \mathbf{As}_{b_i}^0$. For any $\{c_i\}_{i<l}$ such that $\bigoplus c_i = b$, $\{b_i \oplus c_i\}_{i<l}$ is a bit string that contains even number 1's. Conversely, such a sequence $\{d_i\}_{i<l}$ creates a bit string $\{b_i \oplus d_i\}_{i<l}$ that satisfies $\bigoplus b_i \oplus d_i = b$. It defines an isomorphism. $\textsc{eRandomise}^1$ uniformly randomly chooses an $l-1$-length bit string The set of $l-1$-length strings is isomorphic to the set of l-length bit string having even number 1's. Therefore, $\textsc{eRandomise}^1$ uniformly chooses $\{c_i\}_{i<l}$ such that $\bigoplus c_i = b$. Because the statement holds when $k = 0$, the statement also holds when $k = 1$. □

Algorithm 10 Inattentive evaluation of measurement angles

procedure $\textsc{eEncode}^0(\zeta \in \{\, 0, 1, \mathbf{b} \,\}, (b, c))$
  if $\zeta = 0$ **then**
    Output $(0, 1)$.
  else if $\zeta = 1$ **then**
    Output $(1, 0)$.
  else
    Output (b, c).
  end if
end procedure

procedure $\textsc{eEncode}^1(\zeta \in \{\, 0, 1, \mathbf{b}_X, \mathbf{b}_Z, \neg\mathbf{b}_X, \neg\mathbf{b}_Z \,\}, \{(b_i, c_i)\}_{i<l})$
  if ζ is 0 or 1 **then**
    for all $i < l$ **do**
      $x_i = \textsc{eEncode}^0(0, (b_i, c_i))$.
    end for
  else
    Let Y be either X or Z such that ζ is $\mathbf{b}_Y$ or $\neg\mathbf{b}_Y$.
    for all $i < l$ **do**
      if $i \in Y$ **then**
        $x_i = \textsc{eEncode}^0(\mathbf{b}, (b_i, c_i))$.
      else
        $x_i = \textsc{eEncode}^0(0, (b_i, c_i))$.
      end if
    end for
  end if
  if ζ is either 1 or of the form $\neg\eta$ **then**
    $x_0 = \textsc{eNot}^0(x_0)$.
  end if
  Output $(x_i)_{i<l}$.
end procedure

procedure $\textsc{eEncode}^2(\eta \vee \zeta, \{(b_i, c_i)\}_{i<l})$
  $x = \textsc{eEncode}^1(\eta, \{(b_i, c_i)\}_{i<l})$.
  $y = \textsc{eEncode}^1(\zeta, \{(b_i, c_i)\}_{i<l})$.
  $z = \textsc{eEncode}^1(0, \{(b_i, c_i)\}_{i<l})$.
  $w = \textsc{eEncode}^1(1, \{(b_i, c_i)\}_{i<l})$.
  Output $((x, z), (y, z), (x, y), (w, z))$.
end procedure

procedure $\textsc{eEncode}^3(\eta \oplus \zeta, \{(b_i, c_i)\}_{i<l})$
  $x = \textsc{eEncode}^2(\eta, \{(b_i, c_i)\}_{i<l})$.
  $y = \textsc{eEncode}^2(\zeta, \{(b_i, c_i)\}_{i<l})$.
  Output (x, y).
end procedure

Algorithm 11 Inattentive evaluation of measurement angles (Cont.)

EDECODE^2 is the same as DECODE except it uses EDECODE^1 instead of SUBDECODE.
EDECODE^3 is the same as SUBDECODE except it uses EDECODE^2 instead of DECODE.
ERANDOMISE^2 is the same as RANDOMISE except it uses ERANDOMISE^1 instead of SUBRANDOMISE.
ERANDOMISE^3 is the same as SUBRANDOMISE except it uses ERANDOMISE^2 and ENOT^2 instead of RANDOMISE and NOT.
ENOT^0 is the same as NOT^0.
ENOT^2 is the same as NOT^{k+1} except it uses ENOT^1 instead of NOT^k.

procedure EDECODE$^0((b, c))$
  Output b.
end procedure

procedure EDECODE$^1((x_i)_{i<l})$
  Output $\bigoplus_{i<l}$ EDECODE$^0(x_i)$.
end procedure

procedure ERANDOMISE$^0((b, c))$
  Output (b, c).
end procedure

procedure ERANDOMISE$^1((x_i)_{i<l})$
  $c = 0$.
  for all $i < l - 1$ **do**
    $y_i =$ERANDOMISE$^0(x_i)$.
    $r \leftarrow \mathbb{B}$.
    if $r = 1$ **then**
      $y_i =$ENOT$^0(y_i)$.
      $c = c + 1$.
    end if
  end for
  $y_{l-1} =$ERANDOMISE$^0(x_{l-1})$.
  if c is odd **then**
    $y_{l-1} =$ENOT$^0(y_{l-1})$.
  end if
  Output $(y_i)_{i<l}$.
end procedure

procedure ENOT$^1((x_i)_{i<l})$
  $x_0 =$ ENOT$^0(x_0)$
  Output $(x_i)_{i<l}$.
end procedure

procedure ENOT$^3((x, y))$
  Output (ENOT$^2(x)$, y)
end procedure

We note that eEncode and eRandomise make an encryption scheme to be unitedly $\mathbf{F}_{\mathrm{B}}$-homomorphic. In eEncode^1, the ith element (b_i, c_i) of an input $\{(b_i, c_i)\}_{i<l}$ is used to compute the ith element x_i of the output. eRandomise^1 does not change the order. Suppose the input is ciphertexts and the ith element is encrypted by the ith encryption key. Even after eEncode and eRandomise, the ith element of the output is still a ciphertext by the ith encryption key. Therefore, Bob can decrypt the output.

Let us move to the other problem. We care about evil Bob. What happens when Bob sends $(0,0)$ instead of $(0,1)$? Suppose Alice evaluates a formula $(\zeta_0 \vee \zeta_1) \oplus (\zeta_2 \vee \zeta_3)$ with an input containing $(0,0)$ and sends the output to Bob. Then, he checks whether the output contains $(0,0)$ and learns whether each ζ_i is constant or not. Of course, he cannot do that when Alice removes the second elements of all pairs in the zeroth layer before sending the output. The modification does not obstruct eDecode. However, even if so, he can guess that. He just counts the number of zeros. Since $(0,0)$ is invariant under eNot, if an output has many zeros, it probably depends on the inputs.

In order to prevent such attacks, we modify the zeroth layer. Roughly speaking, an idea is to encode a constant using the given input so that even when he sends invalid messages, Bob cannot distinguish a constant from a value depending on the inputs.

$$\mathbf{ZAs}_0^{-1,b} = \left\{ \{(a_i, c_i)\}_{i<4} \;\middle|\; \begin{array}{l} \{(a_i, c_i)\}_{i<4} \text{ has two } (b,b), \text{ one} \\ (1-b, 1-b) \text{ and one } (0,1) \text{ or } (1,0). \end{array} \right\} \tag{4.32}$$

$$\mathbf{ZAs}_1^{-1,b} = \left\{ \{(a_i, c_i)\}_{i<4} \;\middle|\; \exists j \text{ s.t. } a_j = c_j = b, \forall k \neq j, a_k \oplus c_k = 1 \right\} \tag{4.33}$$

$$\mathbf{ZAs}_b^{0,b'} = \mathbf{ZAs}_b^{-1,b'} \times \mathbf{ZAs}_{1-b}^{-1,b'} \tag{4.34}$$

b shows an input from Bob. We can immediately define $\mathbf{ZAs}_b^{k,\{b_i\}_{i<l}}$ in the same manner as $\mathbf{As}$. How to encode a constant are shown in Algorithm 12. We prepare algorithms such as zEncode^0. We use them instead of algorithms such as eEncode^0 and define algorithms such as zEncode^3.

First, we show the method correctly works when Bob is honest.

Proposition 4.6.8. *For any $b \in \mathbb{B}$ and $\zeta \in \{0, 1, \mathbf{b}\}$, zEncode$^0(\zeta, (b, 1-b))$ belongs to $\boldsymbol{ZAs}^{0,b}_{\zeta[b/\mathbf{b}]}$.*

Proof. When $b = 0$, $x = \{(0,0),(0,0),(1,1),(0,1)\}$, $y = \{(0,0),(0,1),(0,1),(0,1)\}$, $z = \{(0,0),(0,0),(1,1),(0,1)\}$, and $w = \{(0,0),(0,1),(0,1),(1,0)\}$. When $b = 1$, $x = \{(1,1),(1,1),(0,0),(1,0)\}$, $y = \{(1,1),(1,0),(1,0),(1,0)\}$, $z = \{(1,1),(0,1),(1,0),(1,0)\}$, and $w = \{(1,1),(1,1),(0,0),(1,0)\}$. Here, we regard them as sets. □

Proposition 4.6.9. *Let $b, b' \in \mathbb{B}$. zNot^0 is an isomorphism between $\boldsymbol{ZAs}_b^{0,b'}$ and $\boldsymbol{As}_{1-b}^{0,b'}$.* □

Proposition 4.6.10. *Let $b, b' \in \mathbb{B}$. For any $M \in \boldsymbol{ZAs}_b^{0,b'}$, zDecode$^k(M) = b$.* □

Proposition 4.6.11. *Let $b, b' \in \mathbb{B}$. There exists a distribution on $\boldsymbol{ZAs}_b^{0,b'}$ such that for any $M \in \boldsymbol{ZAs}_b^{0,b'}$, zRandomise$^0(M)$ is the distribution.*

Proof. Let $S = S_4 \times \mathbb{B}$. We first note $\mathbf{ZAs}_b^{-1,b}$ is closed under a permutation defined by an element of S. Take $b' \in \mathbb{B}$ and $\{(a_{i0}, a_{i1})\}_{i<4} \in \mathbf{ZAs}_0^{-1,b'}$. Let

Algorithm 12 Modification of the zeroth layer

procedure zEncode$^0(\zeta \in \{0, 1, \mathbf{b}\}, (b, c))$
  $x = ((b, 0), (b, 1), (b, b), (c, c))$.
  $y = ((b, 0), (b, 1), (b, c), (b, c))$.
  $z = ((b, 0), (c, 1), (b, b), (b, c))$.
  $w = ((c, 0), (b, 1), (b, b), (b, c))$.
  if $\zeta = 0$ **then**
    Output (x, y).
  else if $\zeta = 1$ **then**
    Output (y, x).
  else
    Output (z, w).
  end if
end procedure

procedure zDecode$^0((\{(a_i, b_i)\}_{i<4}, c))$
  Output b such that exactly three of $\{a_i \oplus b_i\}_{i<4}$ is b.
end procedure

procedure zRandomise$^0((\{(a_{i0}, a_{i1})\}_{i<4}, \{(b_{i0}, b_{i1})\}_{i<4}))$
  $\sigma \leftarrow S_4$.
  $\pi \leftarrow S_4$.
  for all $i < 4$ **do**
    $r \leftarrow \mathbb{B}$.
    $(c_{i0}, c_{i1}) = (a_{\sigma(i)r}, a_{\sigma(i)(1-r)})$.
    $r' \leftarrow \mathbb{B}$.
    $(d_{i0}, d_{i1}) = (b_{\pi(i)r'}, b_{\pi(i)(1-r')})$.
  end for
  Output $(\{(c_{i0}, c_{i1})\}_{i<4}, \{(d_{i0}, d_{i1})\}_{i<4})$.
end procedure

procedure zNot$^0((x, y))$
  Output (y, x).
end procedure

j_a be such that $a_{j_a0} = a_{j_a1} = 1 - b'$, k_a be such that $a_{k_a0} \neq c_{k_a0}$, and r_a be $a_{k_a r_a} = 0$. Let $\{(c_{i0}, c_{i1})\}_{i<4} \in \mathbf{ZAs}_0^{-1,b'}$. Similarly, we can take j_c, k_c, r_c. The values $j_a, j_c, k_a, k_c, r_a, r_c$ defines how to permute $\{(a_{i0}, a_{i1})\}_{i<4} \in \mathbf{ZAs}_0^{-1,b'}$ to obtain $\{(c_{i0}, c_{i1})\}_{i<4} \in \mathbf{ZAs}_0^{-1,b'}$. The permutation can be extended to elements of S, and the number of these elements are the same for any j_c, k_c, r_c. Therefore, zRandomise^0 uniformly distributes an element of $\mathbf{ZAs}_0^{-1,b'}$. The same can be held in $\mathbf{ZAs}_1^{-1,b'}$. Hence, the statement holds. □

For the same reason as noted before, zEncode and zRandomise make an encryption scheme to be unitedly $\mathbf{F}_\mathrm{B}$-homomorphic.

Theorem 4.6.12. *Let $\mathcal{E}$ be a rerandomisable and verifiable classical public-key encryption scheme such that the plaintext space is $\mathbb{B}$. Then, $\mathcal{E}$ is unitedly $\mathbf{F}_\mathrm{B}$-homomorphic with* zEncode^3 *and* zRandomise^3. □.

Now, suppose Bob sends (b, b). Then, $x = y = z = w$ in zEncode^0. Therefore, the output of zEncode^0 is independent of ζ and no information is leaked.

Furthermore, via zRandomise^1, that influences the other elements.

$$\mathbf{IZAs}^{0,b} = \{ (x,x) \mid x = ((b,0),(b,1),(b,b),(b,b)) \} \tag{4.35}$$

$$\mathbf{IZAs}^{1,\{b_i\}_{i<l},j_0,\ldots,j_{m-1}} = \prod_{i\neq j_0,\ldots,j_{m-1}} \mathbf{ZAs}^{0,b_i} \times \prod_{i=j_0,\ldots,j_{m-1}} \mathbf{IZAs}^{0,b_i} \tag{4.36}$$

where j_k are distinct elements of $[<l]$ and $0 < m < l$.

Proposition 4.6.13. *Let $\{b_i\}_{i<l}$ be a bit string, m be such that $0 < m < l$, and $j_0,\ldots,j_{m-1} \in [<l]$. There is a distribution on $\boldsymbol{IZAs}^{1,\{b_i\}_{i<l},j_0,\ldots,j_{m-1}}$ such that for any $M \in \boldsymbol{IZAs}^{1,\{b_i\}_{i<l},j_0,\ldots,j_{m-1}}$, zRandomise$^1(M)$ obeys the distribution.*

Proof. For a bit string $(a_i)_{i\neq j_0,\ldots,j_{m-1}}$, we write $X_{(a_i)_{i\neq j_0,\ldots,j_{m-1}}}$ to denote a set $\prod_{i\neq j_0,\ldots,j_{m-1}} \mathbf{ZAs}^{0,b_i}_{a_i} \times \prod_{i=j_0,\ldots,j_{m-1}} \mathbf{IZAs}^{0,b_i}$. Let $a = (a_i)_{i\neq j_0,\ldots,j_{m-1}}$ be a bit string such that $M \in X_a$. For any bit string $c = (c_i)_{i\neq j_0,\ldots,j_{m-1}}$, M can be converted to an element of X_c by applying zNot^0 to elements such that $a_i \oplus c_i = 1$. Because $\mathbf{IZAs}^{0,b_i}$ is invariant under zNot^0, $c = (c_i)_{i\neq j_0,\ldots,j_{m-1}}$ such that an output of zRandomise^1 belongs to X_c is determined by an $l-m$ bit string. zRandomise^1 uniformly randomly chooses an $l-1$-length bit string, which is also a uniformly random $l-m$-length bit string. zRandomise^0 obeys some distribution determined by each set, so the statement also holds. □

As a consequence, zEncode and zRandomise make an encryption schemes to be unitedly covered.

Theorem 4.6.14. *Let $\mathcal{E}$ be a rerandomisable and verifiable classical public-key encryption scheme such that the plaintext space is $\mathbb{B}$. Then, $\mathcal{E}$ is unitedly $\mathbf{F}_{\mathrm{B}}$-homomorphic and unitedly covered with zEncode^3 and zRandomise^3.* □

4.6.3 Publicly verifiable BQC protocol exists

Due to Theorem 4.6.14, all we have to do is to find a rerandomisable and verifiable classical public-key encryption scheme such that the plaintext space is $\mathbb{B}$. Fortunately, we can easily find such an encryption scheme. We use the ElGamal encryption scheme.

Definition 4.6.15. Define $\mathcal{E}_{\mathbf{BEG}} = (G_{\mathbf{BEG}}, E_{\mathbf{BEG}}, D_{\mathbf{BEG}}, R_{\mathbf{BEG}})$ as follows. $G_{\mathbf{BEG}} = G_{\mathbf{EG}}$. Let (p,g,g^x) be an encryption key. The plaintext space $\mathbf{PS}$ is $\mathbb{B}$. Let $b \in \mathbb{B}$. $E_{\mathbf{BEG}}((p,g,g^x),b)$ is $(g^m g^{xr}, g^r)$ where r is random chosen from $\mathbb{Z}_p$ and m is 0 if $b = 0$ and otherwise randomly chosen from $\mathbb{Z}_p^*$. $D_{\mathbf{BEG}}((p,g,g^x),x,(g_m,g_r))$ is 0 if $g_m(g_r^x)^{-1} = 1$ and otherwise 1. $R_{\mathbf{BEG}}((p,g,g^x),x,(g_m,g_r))$ is $(g_m^y g^{xz}, g_r^y g^z)$ where y and z are uniformly randomly chosen from $\mathbb{Z}_p^*$ and $\mathbb{Z}_p$, respectively.

Proposition 4.6.16. *$\mathcal{E}_{\mathbf{BEG}}$ is a rerandomisable and verifiable classical encryption scheme.*

Proof. $\mathcal{E}_{\mathbf{BEG}}$ is a classical encryption scheme because $\mathcal{E}_{\mathbf{EG}}$ is. Let $e = (p,g,g^x)$ be an encryption key. The ciphertext spaces are $\mathbf{CS}_{n,e,0} = \{ (g^{xr}, g^r) \mid r \in \mathbb{Z}_p \}$ and $\mathbf{CS}_{n,e,1} = \{ (g^m g^{xr}, g^r) \mid m \in \mathbb{Z}_p^*, r \in \mathbb{Z}_p \}$.

Let $(g^{xr}, g^r) \in \mathbf{CS}_{n,e,0}$. An output of $R_{\mathbf{BEG}}(e,x,(g^{xr},g^r))$ is $(g^{xry}g^{xz}, g^{ry}g^z) = (g^{x(ry+z)}, g^{ry+z})$ with randomly chosen y and z. For any $r,s \in \mathbb{Z}_p$ and any $y \in \mathbb{Z}_p^*$, there exists a unique $z \in \mathbb{Z}_p$ such that $s = ry + z$. Therefore, an output is uniformly distributed on $\mathbf{CS}_{n,e,0}$. Let $(g^m g^{xr}, g^r) \in \mathbf{CS}_{n,e,1}$. An output

of $R_{\mathbf{BEG}}(e, x, (g^m g^{xr}, g^r))$ is $(g^{my} g^{xry} g^{xz}, g^{ry} g^z) = (g^{my} g^{x(ry+z)}, g^{ry+z})$ with randomly chosen y and z. Because $\mathbb{Z}_p^*$ is an abelian group, my is a uniformly random chosen element of $\mathbb{Z}_p^*$ if y is uniformly chosen. Therefore, an output is uniformly distributed on $\mathbf{CS}_{n,e,1}$.

Finally, we show $\mathcal{E}_{\mathbf{BEG}}$ is verifiable. Let $2p+1$ be a safe prime and G_p be the subgroup of order p of the cyclic group $\mathbb{Z}_{2p+1}^*$. We note that $x \in \mathbb{Z}_{2p+1}^*$ belongs to G_p if and only if x^p is 1 or not. The latter can be checked in polynomial-time. An encryption key (p, g, g^x) is required to be satisfy the conditions that p is a safe prime, g, g^x belongs to G_p. Conversely, if a trio satisfies the above conditions, it is a valid encryption key. The decryption key is unique. Since G_p is cyclic, $g^x = g^y$ and $x, y \in \mathbb{Z}_p$ means $x = y$. The whole ciphertext space $\mathbf{CS}_{n,e,0} \cup \mathbf{CS}_{n,e,1}$ is $\{ (g^{xr}, g^r) \mid r \in \mathbb{Z}_p \} \cup \{ (g^m g^{xr}, g^r) \mid m \in \mathbb{Z}_p^*, r \in \mathbb{Z}_p \} = G_p \times G_p$. □

Proposition 4.6.17. *$\mathcal{E}_{\mathbf{BEG}}$ is IND-CPA secure under the DDH assumption.*

Proof. If a ciphertext of 0 can be distinguished from a ciphertext of 1, a ciphertext of 0 can be distinguished from a ciphertext of non-zero plaintext in the ElGamal encryption scheme. □

Using $\mathcal{E}_{\mathbf{BEG}}$, zEncode, and zRandomise, Protocol 8 is a publicly verifiable BQC protocol.

Corollary 4.6.18. *There exists an ϵ_{FK}-verifiable BQC protocol that is ϵ_{FK}-publicly verifiable if the numbers of vertices of graphs are polynomially bounded with respect to the security parameter n', and that Alice cannot create any entangled states, and the DDH assumption holds.*

Proof. The statement follows from Theorems 4.5.3, 4.5.4, and 4.6.14 and Propositions 4.6.16 and 4.6.17. □

Chapter 5

Conclusions

In Chapters 3, 4, we showed that two classical methods are useful to analyse quantum computation. They illustrate that there are classical methods that have not been used well to investigate quantum computation but is indeed useful. We believe that such direction of research will accelerate and broaden the research of quantum computation. The summary and future work of each chapter are as follows.

5.1 Stabiliser abstract semantics

In Chapter 3, we proposed two new abstract semantics. They were based on an existing abstract semantics that we call the basis abstract semantics. The basis abstract semantics was introduced as application of abstract interpretation to quantum computation [89]. It is based on observations that unitary operators acting on multiple qubits entangle qubits unless their states belong to some bases and that quantum measurement destroys all entanglement with the measured qubit. However, the semantics ignores the facts that these unitary operators can undo entanglements and that quantum measurement may destroy entanglement between unmeasured qubits.

In order to reflect the facts, we decided to use stabilisers to abstract entangled states. Based on the idea, we proposed an abstract semantics named the stabiliser abstract semantics in Section 3.2. First, we discussed how to decompose stabiliser arrays (Algorithm 1). We proposed the stabiliser domain and the stabiliser abstract semantics in Subsections 3.2.3, 3.2.4. Then, we proved the soundness of the semantics in Theorem 3.2.35. Therefore, the semantics never gives wrong approximations of separability of concrete states. Although the semantics lacks monotonicity (Proposition 3.2.37) and it may take superpolynomial time to compute `while` loop, we showed that an upper approximation of it converges by repeating computation polynomial times. Moreover, we showed that an approximation of the semantics of any constant-depth program can be computable in polynomial time with respect to the program size. At the end of the section (Subsection 3.2.5), we showed that there exists a Galois connexion between the stabiliser domain and the basis domain, and that the basis abstract semantics is a sound approximation of the stabiliser abstract semantics.

In the next section, we improved the stabiliser abstract semantics. The semantics analyses nothing about non-stabiliser states, more precisely, about states of qubits that go through non-stabiliser states. We stopped crushing a stabiliser into a symbol ■ and introduced a symbol ♡. We proved that orthonormal abelian groups of Hermitian unitary matrices are appropriate to instances of arrays containing ♡ (Proposition 3.3.1) and defined extended stabiliser arrays (Defini-

tion 3.3.1). Although the definition of extended stabiliser arrays is more complex than that of stabiliser arrays, we can efficiently decompose stabiliser arrays (Algorithm 3). With these arrays, we proposed our second abstract semantics named the extended stabiliser abstract semantics in Subsection 3.3.3. We proved that the semantics is sound in Theorem 3.3.48. Although it inherits various features of the stabiliser abstract semantics, the extended stabiliser abstract semantics is strictly better than the stabiliser abstract semantics (Theorem 3.3.48). It is worth noting that an upper approximation of the semantics of any constant-depth program can be efficiently computed in the extended stabiliser abstract semantics. Therefore, we can use the semantics and classical computers to analyse quantum entanglement in quantum programs.

5.1.1 Future work

Before moving the next section, we list possible directions of further work.

- In the thesis, we proposed two domains. They have essentially the same structures. The reason is that both the sets of stabiliser arrays and extended stabiliser arrays are flat lattices. Can we give a more complex order to each set? In particular, the set of extended stabiliser arrays may have another order.

- We used the normalisation operator nml when we take the approximate join of extended stabiliser arrays. This is because an extended stabiliser array γ has many instances and $\gamma \vDash_{\mathbf{E}} \rho$ and $\gamma \vDash_{\mathbf{E}} \sigma$ do not necessarily imply $\gamma \vDash_{\mathbf{E}} \rho + \sigma$. Therefore, if we know ρ and σ have a common instance, we do not have to normalise the extended stabiliser array. By recording whether a block is touched, we will achieve it.

- We gave algorithms to compute the stabiliser abstract semantics and the extended stabiliser abstract semantics. Precisely speaking, they include upper approximations. However, we have no implementation of these algorithms. They are needed to be implemented.

5.2 Publicly verifiable blind quantum computation

We formulated a new property of BQC protocols, which is named public verifiability (Definition 4.2.1) in Section 4.2. Although several existing BQC protocols achieve a property called verifiability, the property does not help third parties analyse quantum computation on BQC protocols. Hence, they cannot analyse computation in existing BQC protocols, even if they can use quantum computers. We illustrated how the impossibility causes practical problems in the section. Our new property is necessary for classical computers to analyse BQC computation. After that, we first proposed new verifiable BQC protocols (Protocols 5, 7) in Section 4.4. These protocols were proved to be correct, blind, and verifiable provided that appropriate encryption schemes exist in Theorems 4.4.10, 4.4.13. However, we did not show whether such an encryption scheme really exists. Then, we proposed a publicly verifiable BQC protocol (Protocol 8) in Section 4.5. In order to find an appropriate encryption schemes, we relaxed the security condition on encryption schemes. No security condition was imposed on schemes except for the condition that they have indistinguishable encryptions, but the schemes were required to be homomorphic with different encryption keys. We proved in Theorem 4.5.4 that the above protocol is publicly verifiable provided

that such an encryption scheme exists and Alice cannot create any entangled state. In Section 4.6, we proved that such an appropriate encryption scheme exists (Theorem 4.6.14 and Proposition 4.6.17). We employed inattentive evaluation of circuits and the ElGamal encryption scheme. The former allows us to use different keys with the perfect security and the latter does us to verify the validity of encryption keys. As a consequence, we proved that a publicly verifiable BQC protocol exists under the restriction on quantum devices of Alice (Corollary 4.6.18).

Although our protocol is based on the FK protocol, our technique is purely classical. Therefore, it can be adapted for other protocols having the same style as the FK protocol has.

5.2.1 Future work

We list several questions here.

- In Section 4.2, we inferred that no *unconditionally* publicly verifiable BQC protocol exists due to the no-go theorem for unconditionally secure bit commitment scheme. We did not give any formal proof, so it is needed to be proved. Moreover, precisely speaking, there are unconditionally secure bit commitment schemes [64, 65, 66, 67, 70]. The security is guaranteed by special relativity. These schemes assume that both parties have own agents in a distant place. They communicate with these agents to commit a bit value. The committed bit value cannot change due to the impossibility of superluminal signalling. In order to apply these schemes to BQC protocols for public verifiability, we have to assume that both Alice and Bob has own trustful agents. This is a weaker assumption than they trust the same party but it is stronger than our setting, where Alice and Bob do not trust any other party. If we accept the assumption, can we achieve *unconditionally* publicly verifiable BQC protocol?

- Our publicly verifiable BQC protocol is publicly verifiable under an assumption that Alice cannot create any entangled state. If she has an ability to create entangled state of any number of qubits and has a quantum measurement device, she will be able to perform MBQC herself. On the other hand, if she creates only constant-size entanged states, she will be unable to obtain sufficient information to guess measurement results. Therefore, the assumption seems not to be essential. Can we remove the assumption?

- Our protocol computes classical outputs from classical inputs. Many BQC protocols including the FK protocol work on quantum input and quantum output. Can we improve our protocol so that it takes quantum input and computes quantum output? It is obvious that our protocol works on quantum input. A challenge is to compute quantum output. In verifiable BQC protocols, quantum output contain traps. Alice receives quantum outputs and checks its correctness by measuring the trap qubits. The measurement is privately performed, so no classical information about traps in quantum output is revealed. For quantum output publicly verifiable BQC protocols, we have to bind quantum output to some classical information. Naturally, we have to change the definition of public verifiability when we allow protocols to work with quantum input and output.

- We have focused on single-server and generation-only BQC protocols. There are other styles of BQC protocols. One is double-server protocols, where

Alice delegates computation to two servers, Bob1 and Bob2 [16, 80]. Although Alice having no quantum device cannot perform a BQC protocol with Bob [81], she can blindly delegate quantum computation to two servers. The other style is measurement-only protocols [58, 76, 79]. In the protocol, Bob generates an entangled quantum state and sends qubits to Alice one by one. Alice performs MBQC by measuring each qubit with her desired angle. The protocol requires Alice to have a quantum measurement device instead of a single qubit generator. Can we give public verifiability to these styles of BQC protocols? The double-server protocol works unless the two servers communicate with each other. The assumption conflicts a premise of public verifiability that communication between Alice and a server is public. However, if Alice uses three servers and is able to access quantum channels, the assumption can be removed [68]. The measuring-Alice protocol and public verifiability have serious conflicts. One is that there are no classical messages. Bob sends only qubits and Alice sends nothing. Anyone cannot observe anything in their communication. Moreover, a big advantage of the protocol is that the security is guaranteed by no-signalling principle, which is known to be strictly weaker than the axiom of quantum physics [91]. Giving public verifiability, the advantage will be lost.

- We proved the existence of appropriate encryption schemes using inattentive evaluation of circuits. Is there another appropriate encryption scheme? Can we reduce the length of messages between Alice and Bob? Moreover, we imposed strong conditions on encryption schemes. We refused any possibility that the security of the FK protocol is damaged. Although we weakened the conditions, they are still strong. Can we relax the conditions more? For example, a protocol having imperfect blindness was proposed in [32]. For such a protocol, we may be able to relax the definition of united cover. Then, we may be able to use encryption schemes such as fully homomorphic encryption schemes [44, 84, 105] and techniques such as zero-knowledge proofs [15, 51].

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70:052328, Nov 2004.

[2] Martin Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, STOC '87, pages 195–203, New York, NY, USA, 1987. ACM.

[3] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. In *Innovations in Computer Science*, pages 453–469, 2010.

[4] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the Jones polynomial. *Algorithmica*, 55(3):395–421, 2009.

[5] Thorsten Altenkirch and Jonathan Grattage. A functional quantum programming language. In *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 249–258, June 2005.

[6] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

[7] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 04(05):883–898, 2006.

[8] Koenraad M R Audenaert and Martin B Plenio. Entanglement on mixed stabilizer states: normal forms and reduction procedures. *New Journal of Physics*, 7(1):170, 2005.

[9] Stefanie Barz, Joseph F. Fitzsimons, Elham Kashefi, and Philip Walther. Experimental verification of quantum computation. *Nature Physics*, 9(11):727–731, Nov 2013.

[10] Stefanie Barz, Elham Kashefi, Anne Broadbent, Joseph F. Fitzsimons, Anton Zeilinger, and Philip Walther. Demonstration of blind quantum computing. *Science*, 335(6066):303–308, 2012.

[11] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *IEEE International Conference on Computers, Systems and Signal Processing*, pages 175–179, 1984.

[12] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70(13):1895–1899, March 1993.

[13] Charles H. Bennett and Stephen J. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69:2881–2884, Nov 1992.

[14] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer Berlin Heidelberg, 2008.

[15] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156 – 189, 1988.

[16] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium on*, pages 517–526, Oct 2009.

[17] Daniel E. Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9(8):250, 2007.

[18] Andrew M. Childs. Secure assisted quantum computation. *Quantum Information and Computation*, 5(6):456–466, September 2005.

[19] Patrick Cousot and Radhia Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106–130. Dunod, Paris, France, 1976.

[20] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, pages 238–252, New York, NY, USA, 1977. ACM.

[21] Patrick Cousot and Radhia Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.

[22] Vincent Danos and Elham Kashefi. Determinism in the one-way model. *Physical Review A*, 74:052310, Nov 2006.

[23] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *Journal of the ACM*, 54(2), April 2007.

[24] Vincent Danos, Elham Kashefi, Prakash Panangaden, and Simon Perdrix. Extended measurement calculus. In *Semantic Techniques in Quantum Computation*, pages 235–310. Cambridge University Press, 2009.

[25] Niel de Beaudrap. Finding flows in the one-way measurement model. *Physical Review A*, 77:022328, Feb 2008.

[26] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 400(1818):97–117, 1985.

[27] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 425(1868):73–90, 1989.

[28] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 439(1907):553–558, 1992.

[29] D. Dieks. Communication by EPR devices. *Physics Letters A*, 92(6):271 – 272, 1982.

[30] Paul A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35:416–418, 7 1939.

[31] Vedran Dunjko, JosephF. Fitzsimons, Christopher Portmann, and Renato Renner. Composable security of delegated quantum computation. In *Advances in Cryptology ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 406–425. Springer Berlin Heidelberg, 2014.

[32] Vedran Dunjko, Elham Kashefi, and Anthony Leverrier. Blind quantum computing with weak coherent pulses. *Physical Review Letters*, 108:200502, May 2012.

[33] Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469–472, Jul 1985.

[34] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475, 2001.

[35] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106, 2000.

[36] Joan Feigenbaum. Encrypting problem instances. In *Advances in Cryptology — CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 477–488. Springer Berlin Heidelberg, 1986.

[37] Yuan Feng, Runyao Duan, Zhengfeng Ji, and Mingsheng Ying. Probabilistic bisimulations for quantum processes. *Information and Computation*, 205(11):1608 – 1639, 2007.

[38] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 501–512, New York, NY, USA, 2012. ACM.

[39] Joseph F. Fitzsimons and Elham Kashefi. Unconditionally verifiable blind computation. arXiv:1203.5217, 2012.

[40] Michael H. Freedman, Alexei Kitaev, Michael J. Larsen, and Zhenghan Wang. Topological quantum computation. *Bulletin of the American Mathematical Society*, 40(1):31–38, 2003.

[41] Simon J. Gay. Quantum programming languages: survey and bibliography. *Mathematical Structures in Computer Science*, 16:581–600, 8 2006.

[42] Simon J. Gay, Rajagopal Nagarajan, and Nikolaos Papanikolaou. QMC: A model checker for quantum systems. In *Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*, pages 543–547. Springer Berlin Heidelberg, 2008.

[43] Simon J. Gay, Rajagopal Nagarajan, and Nikolaos Papanikolaou. Specification and verification of quantum protocols. In *Semantic Techniques in Quantum Computation*, pages 414–472. Cambridge University Press, 2009.

[44] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[45] Alexandru Gheorghiu, Elham Kashefi, and Petros Wallden. Robustness and device independence of verifiable blind quantum computing. *New Journal of Physics*, 17(8):083040, 2015.

[46] Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G. Rudolph. Efficient universal blind quantum computation. *Physical Review Letters*, 111:230501, Dec 2013.

[47] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[48] Oded Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[49] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC '13, pages 555–564, New York, NY, USA, 2013. ACM.

[50] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[51] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM.

[52] Daniel Gottesman. The Heisenberg representation of quantum computers. *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, 1999.

[53] Daniel Gottesman and Isaac L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, Nov 1999.

[54] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: A scalable quantum programming language. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, pages 333–342, New York, NY, USA, 2013. ACM.

[55] Gerhard Grössing and Anton Zeilinger. Quantum cellular automata. *Complex Systems*, 2(2):197–208, 1988.

[56] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[57] Leonid Gurvits. Classical deterministic complexity of Edmonds' problem and quantum entanglement. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC '03, pages 10–19, New York, NY, USA, 2003. ACM.

[58] Masahito Hayashi and Tomoyuki Morimae. Verifiable measurement-only blind quantum computing with stabilizer testing. *Physical Review Letters*, 115:220502, Nov 2015.

[59] Kentaro Honda. Analysis of quantum entanglement in quantum programs using stabilizer formalism. In Proceedings of the 12th International Workshop on *Quantum Physics and Logic*, volume 195 of *Electronic Proceedings in Theoretical Computer Science*, pages 262–272. Open Publishing Association, 2015.

[60] Lawrence M. Ioannou. Computational complexity of the quantum separability problem. *Quantum Information and Computation*, 7(4):335–370, May 2007.

[61] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. ScaffCC: Scalable compilation and analysis of quantum programs. *Parallel Computing*, 45:2 – 17, 2015.

[62] Philippe Jorrand and Marie Lalire. Toward a quantum process algebra. In *Proceedings of the 1st Conference on Computing Frontiers*, CF '04, pages 111–119, New York, NY, USA, 2004. ACM.

[63] Theodoros Kapourniotis, Elham Kashefi, and Animesh Datta. Blindness and verification of quantum computation with one pure qubit. In *9th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2014)*, volume 27 of *Leibniz International Proceedings in Informatics*, pages 176–204, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[64] Adrian Kent. Unconditionally secure bit commitment. *Physical Review Letters*, 83:1447–1450, Aug 1999.

[65] Adrian Kent. Secure classical bit commitment using fixed capacity communication channels. *Journal of Cryptology*, 18(4):313–335, 2005.

[66] Adrian Kent. Unconditionally secure bit commitment with flying qudits. *New Journal of Physics*, 13(11):113015, 2011.

[67] Adrian Kent. Unconditionally secure bit commitment by transmitting measurement outcomes. *Physical Review Letters*, 109:130501, Sep 2012.

[68] Qin Li, Wai Hong Chan, Chunhui Wu, and Zhonghua Wen. Triple-server blind quantum computation using entanglement swapping. *Physical Review A*, 89:040302, Apr 2014.

[69] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Physical Review Letters*, 78:3410–3413, Apr 1997.

[70] Tommaso Lunghi, Jedrzej Kaniewski, Felix Bussières, Raphael Houlmann, Marco Tomamichel, Adrian Kent, Nicolas Gisin, Stephanie Wehner, and Hugo Zbinden. Experimental bit commitment based on quantum communication and special relativity. *Physical Review Letters*, 111:180504, Nov 2013.

[71] Tal Malkin, Isamu Teranishi, and Moti Yung. Key dependent message security: Recent results and applications. In *Proceedings of the First ACM Conference on Data and Application Security and Privacy*, CODASPY '11, pages 3–12, New York, NY, USA, 2011. ACM.

[72] Atul Mantri, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Optimal blind quantum computation. *Physical Review Letters*, 111:230502, Dec 2013.

[73] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Physical Review Letters*, 78:3414–3417, Apr 1997.

[74] Tomoyuki Morimae. Continuous-variable blind quantum computation. *Physical Review Letters*, 109:230502, Dec 2012.

[75] Tomoyuki Morimae. Quantum computation: Honesty test. *Nature Physics*, 9(11):693–694, Nov 2013.

[76] Tomoyuki Morimae. Verification for measurement-only blind quantum computing. *Physical Review A*, 89:060302, Jun 2014.

[77] Tomoyuki Morimae, Vedran Dunjko, and Elham Kashefi. Ground state blind quantum computation on AKLT state. *Quantum Information and Computation*, 15:0200–0234, 2015.

[78] Tomoyuki Morimae and Keisuke Fujii. Blind topological measurement-based quantum computation. *Nature Communications*, 3:1036, Sep 2012.

[79] Tomoyuki Morimae and Keisuke Fujii. Blind quantum computation protocol in which Alice only makes measurements. *Physical Review A*, 87:050301, May 2013.

[80] Tomoyuki Morimae and Keisuke Fujii. Secure entanglement distillation for double-server blind quantum computation. *Physical Review Letters*, 111:020502, Jul 2013.

[81] Tomoyuki Morimae and Takeshi Koshiba. Impossibility of secure cloud quantum computing for classical client. arXiv:1407.1636, 2014.

[82] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Oct 2000.

[83] Bernhard Ömer. Quantum programming in QCL. Master's thesis, Technical University of Vienna, 2000.

[84] Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In *Advances in Cryptology CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553. Springer Berlin Heidelberg, 2014.

[85] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography*, volume 7785, pages 222–242. Springer Berlin Heidelberg, 2013.

[86] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*, volume 7194, pages 422–439. Springer Berlin Heidelberg, 2012.

[87] Simon Perdrix. Quantum patterns and types for entanglement and separability. *Electronic Notes in Theoretical Computer Science*, 170(0):125–138, 2007.

[88] Simon Perdrix. A hierarchy of quantum semantics. *Electronic Notes in Theoretical Computer Science*, 192(3):71–83, 2008.

[89] Simon Perdrix. Quantum entanglement analysis based on abstract interpretation. In *Static Analysis*, volume 5079 of *Lecture Notes in Computer Science*, pages 270–282. Springer Berlin Heidelberg, 2008.

[90] Carlos A. Pérez-Delgado and Joseph F. Fitzsimons. Iterated gate teleportation and blind quantum computation. *Physical Review Letters*, 114:220502, Jun 2015.

[91] Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994.

[92] Frédéric Prost and Chaouki Zerrari. Reasoning about entanglement and separability in quantum higher-order functions. In *Unconventional Computation*, volume 5715 of *Lecture Notes in Computer Science*, pages 219–235. Springer Berlin Heidelberg, 2009.

[93] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86:5188–5191, May 2001.

[94] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68:022312, Aug 2003.

[95] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 554–566, 1999.

[96] Jeff W. Sanders and Paolo Zuliani. Quantum programming. In *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*, pages 80–99. Springer Berlin Heidelberg, 2000.

[97] Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14:527–586, 8 2004.

[98] Peter Selinger and Benoît Valiron. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science*, 16:527–552, 6 2006.

[99] Peter Selinger and Benoît Valiron. Quantum lambda calculus. In *Semantic Techniques in Quantum Computation*, pages 135–172. Cambridge University Press, 2009.

[100] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[101] Michel Sintzoff. Calculating properties of programs by valuations on specific models. *SIGPLAN Notice*, 7(1):203–207, January 1972.

[102] Takahiro Sueki, Takeshi Koshiba, and Tomoyuki Morimae. Ancilla-driven universal blind quantum computation. *Physical Review A*, 87:060301, Jun 2013.

[103] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer Berlin Heidelberg, 1998.

[104] Benoît Valiron. Quantum computation: From a programmer's perspective. *New Generation Computing*, 31(1):1–26, 2013.

[105] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer Berlin Heidelberg, 2010.

[106] André van Tonder. A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33(5):1109–1135, 2004.

[107] William K. Wootters and Wojciech Hubert Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, Oct 1982.

[108] Li Yu, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Physical Review A*, 90:050303, Nov 2014.