硕士学位论文

基于 TDD 的量子模型检测中的可达性分析

作者姓名：XX

指导教师：

学位类别：工学硕士

学科专业：计算机科学与技术

培养单位：

2024 年 6 月

Reachability Analysis in Quantum Model Checking Based on TDD

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Engineering

in Computer Science and Technology

By

XX

June, 2024

XX 大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。承诺除文中已经注明引用的内容外，本论文不包含任何其他个人或集体享有著作权的研究成果，未在以往任何学位申请中全部或部分提交。对本论文所涉及的研究工作做出贡献的其他个人或集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日　　期：

XX 大学
学位论文授权使用声明

本人完全了解并同意遵守 XX 大学有关收集、保存和使用学位论文的规定，即 XX 大学有权按照学术研究公开原则和保护知识产权的原则，保留并向国家指定或 XX 指定机构送交学位论文的电子版和印刷版文件，且电子版与印刷版内容应完全相同，允许该论文被检索、查阅和借阅，公布本学位论文的全部或部分内容，可以采用扫描、影印、缩印等复制手段以及其他法律许可的方式保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：　　导师签名：

日　　期：　　日　　期：

摘 要

随着近年来量子计算机在规模和可靠性方面的快速发展，针对量子系统进行自动化验证已成为亟待解决的重要问题。传统的模型检测技术在面对量子系统时往往遇到资源消耗过多的瓶颈。为此，本文提出了一种基于张量决策图 (TDD) 的量子模型检测新方法，旨在降低资源需求，扩展量子模型检测的适用范围。

本文首先，对量子计算、模型检测及其相关数学基础进行了概述，为读者了解研究内容做好铺垫。然后介绍了 TDD 数据结构，它本质上是一种带权重的决策树，用于紧凑高效地表示张量网络。接着阐述了如何将量子线路转化为 TDD 表示，并介绍了 TDD 的规范化、化简等操作。

在此基础上，提出了基于 TDD 的量子模型检测算法流程。首先将量子系统建模为量子迁移系统，然后给出了计算 TDD 表示的子空间的基、子空间并运算以及系统一步迁移的具体算法。为加速计算，设计了多种优化策略，包括线路拆分、基于 TDD 的划分，以及近似 TDD 表示等。

通过在多种量子算法实例上的实验对比，结果表明这些优化方法能够有效降低模型检测中一步迁移过程的时间和空间资源消耗，从而提高算法的可扩展性。其中，基于线路拆分的 contraction 技术优化效果最为显著，能显著提升处理线路层数较深的量子算法的能力。

最后，总结了该研究工作对量子模型检测的贡献，并讨论了未来的工作方向，如结合更复杂量子系统建模、改进 TDD 的表示能力等。本文的研究为量子计算机软硬件可靠性验证提供了有价值的新工具支持。

关键词：模型检测，量子计算，张量决策图

Abstract

With the rapid development of quantum computers in terms of scale and reliability in recent years, automated verification for quantum systems has become an important issue that needs urgent resolution. Traditional model checking techniques often encounter the bottleneck of excessive resource consumption when facing quantum systems. To this end, this paper proposes a new quantum model checking method based on Tensor Decision Diagrams (TDD), aimed at reducing resource requirements and expanding the applicability of quantum model checking.

First, this paper provides an overview of quantum computing, model checking, and their related mathematical foundations, laying the groundwork for readers to understand the research content. Then, it introduces the TDD data structure, which is essentially a weighted decision tree used for compact and efficient representation of tensor networks. Following this, it explains how to convert quantum circuits into TDD representations and introduces operations such as normalization and simplification of TDDs.

Based on this foundation, a quantum model checking algorithm process based on TDD is proposed. It models quantum systems as quantum transition systems, and then presents specific algorithms for computing the basis of the subspaces represented by TDD, subspace union operations, and one-step transitions of the system. To accelerate computation, various optimization strategies are designed, including circuit splitting, partitioning based on TDD, and approximate TDD representation.

Experimental comparisons on various quantum algorithm instances show that these optimization methods can effectively reduce the time and space resource consumption in the model checking one-step transition process, thereby improving the scalability of the algorithm. Among them, the contraction technique optimization based on circuit splitting shows the most significant effect, significantly enhancing the ability to handle quantum algorithms with deeper circuit layers.

Finally, the paper summarizes the contributions of this research work to quantum model checking and discusses future work directions, such as integrating more complex quantum system modeling and improving the representational capability of TDD. The research in this paper provides valuable new tool support for the reliability verification of quantum computer hardware and software.

Key Words: Model Checking, Quantum Computation, Tensor Decision Diagrams

目 录

图目录

表目录

算法目录

缩写列表

BDD	Binary Decision Diagrams
BV	Bernstein-Vazirani algorithm
CQP	Communicating Quantum Processes
EQPL	Exogenous Quantum Propositional Logic
QA	Quantum Automata
QCTL	Quantum Computation Tree Logic
QDA	Quantum Design Automation
QFT	Quantum Fourier Transform
QLTL	Quantum Linear-time temporal Logic
QRW	Quantum Random Walk
QMC	Quantum Model Checking
QMDD	Quantum multiple-valued Decision Diagram
QuIDD	Quantum Information Decision Diagram
RUS	Repeat Until Success
TDD	Tensor Decision Diagrams

第 1 章　绪论

近年来，量子计算因其在特定问题上相比经典计算提供的指数级加速能力而成为科技领域的焦点。随着量子计算技术的迅猛发展，确保量子系统和算法的正确性与可靠性成为了迫切需要解决的问题。此外，物理量子计算机中量子比特数量的快速增长，也对验证方法提出了更为自动化的需求。

模型检测（Model Checking）作为一种自动化的形式化方法，专门用于验证有限状态系统是否满足特定性质。这种方法最初由 E. M. Clarke 和 E. A. Emerson 提出[3–5]，并已在经典计算机的软件和硬件设计中得到广泛应用。例如，嵌入式系统设计中常用 UML (Unified Modeling Language) 活动图来确保硬件设计符合预定规范[6]。在量子计算领域，量子模型检测同样展现出其验证量子系统行为的潜力，特别是在可达性分析方面。然而随着量子系统中量子比特的增加，量子模型检测的资源消耗会呈现指数级增长。本文工作将重点探讨采用张量决策图 (Tensor Decision Diagrams，或 TDD) 作为数据结构，从而缓解量子模型检测的计算资源消耗。

1.1　研究背景与研究问题

1.1.1　研究背景

“量子” 概念最早由 Max Planck 在 1901 年提出，开启了量子物理学的先河[7]。20 世纪 20 年代，量子理论的发展为波粒二象性提供了理论基础，奠定了现代物理学的基石[8]。经历两次世界大战期间，不仅促进了计算理论的飞速发展，如图灵机的提出[9]，也加速了量子理论在核物理学等领域的应用[10]。

随着物理学家将量子力学模型应用于计算问题并将经典比特替换为量子比特，量子力学和计算机科学领域开始融合。1980 年，Paul Benioff 基于量子理论提出了量子图灵机的概念，为量子计算奠定了理论基础[11]。随着经典计算机在模拟量子动力学过程中遇到的计算限制，Yuri Manin 和 Richard Feynman 分别提出，利用量子现象的计算模型可能更为高效[12,13]。此外，Charles Bennett 和 Gilles Brassard 在 1984 年提出量子密钥分发，进一步展示了量子理论在加强信息安全性方面的潜力[14]。

进入 90 年代，量子计算的发展迎来了新的突破。Peter Shor 的算法证明了量子计算机在破解传统加密方面的潜力[15] 。该算法是量子算法发展史上重要的里程碑之一，向人们证明了量子计算对实际问题的加速能力。之后 Grover 的算法为解决非结构化搜索问题提供了量子加速的可能[16] 。同时，Seth Lloyd 证明量子计算机可以模拟量子系统，而无需经典模拟中存在的指数开销[17]，验证了费曼 1982 年的猜想[18]。

多年来，实验学家利用各种可能的技术路线构建小型量子计算机。1998 年，

两个量子比特的量子计算机证明了该技术的可行性[19]。2007 年，加拿大 DWave 公司展示了一台具有 16 个量子比特位处理器原型的绝热量子计算机 (Quantum Annealing) “Orion”。2011 年，DWave 官宣发布了全球第一款商用型量子计算机 “D-Wave One”，其构建了 128 个量子比特，声称用来解决最优化问题。2019 年，Google AI 和 NASA 宣布通过在有 54 量子比特的超导量子计算机上，实现了量子霸权 (Quantum Supermacy)，执行了任何经典计算机都无法完成的计算[20]。尽管，这一说法的有效性需要更多研究[21]。但这样的说法极大增强了人们对量子计算机的热情，促进了各国政府与私人机构对该领域的投资。

近几年，物理量子计算机已经开始尝试规模化拓展及容错计算。以 2023 年为例，接下来列举一些比较重要的硬件发展。在国际方面：

(1) IBM 公司宣布成功开发出了名为 “Condor” 的超导量子处理器，该处理器拥有 1121 个量子比特的运算能力[22]。

(2) IonQ 公司在钡离子阱平台上，实现了 29 个算法量子比特的重大成就，展现了离子阱技术在量子计算中的应用潜力[23]。

(3) Atom Computing 公司推出了第二代中性原子量子计算平台，具备 1180 个量子比特[24]。

2023 年国内方面：

(1) 中国科学技术大学的团队成功构建了 “九章三号” 光量子计算机，该计算机成功实现了 255 个光子的量子计算能力[25]。

(2) 北京量子信息科学研究院、中科院物理研究所和清华大学共同研发推出了 “夸父” 超导量子云平台，其后端最高接入了 136 比特超导量子芯片[26]。

随着量子计算领域的快速发展，对量子系统的可靠性提出了更多需要，对应的验证需求也在不断增长。经典的形式化验证技术，如模型检测 (Model Checking) 和定理证明 (Theorem Proving)，虽然成熟但面临新挑战。模型检测技术通过逻辑公式检测状态迁移系统的行为，虽然自动化程度高，但存在状态空间爆炸问题。定理证明通过数学逻辑来验证系统模型，尽管在处理复杂问题时具有优势，但需要人的干预，自动化程度低。

1.1.2 研究的问题

随着量子计算硬件的发展，量子系统将越来越复杂。自动化程度低，需要大量人力参与的定理证明方法将难以应对。因此本次研究专注于更加自动化的量子模型检测。而在量子模型检测领域，存在一个关键挑战，即状态空间爆炸问题。量子计算的状态空间 $\mathcal{H}$ 维度 $dim(\mathcal{H}) = 2^n$，其中 n 为比特数量。随着比特数量的增加，状态空间的维数会指数级地增长，这极大地增加了应用模型检测来验证量子系统属性的复杂度。

为应对这一挑战，能否通过引入更高效的数据结构，从而以较少的资源表示量子状态和量子线路，并计算量子模型检测的最终结果。同时，在新的数据结构下能否有更好的优化方案，进一步降低量子模型检测过程中的计算资源的消耗。

这些是本次研究中的核心问题。

1.2　研究现状与相关工作

量子模型检测的早期工作集中在验证量子通信协议。随着量子计算的不断发展，逐渐出现了对量子程序的验证工具。如今量子模型检测又集中在了线路验证方面的工作。在整个量子模型检测的发展过程中，我国国内的研究者 Mingsheng Ying 等人深入参与并深刻影响了该领域的研究。

量子模型检测的早期工作旨在验证量子通信协议。由于量子叠加态的特殊性质，量子测量带有概率性。因此 David Parker 等人采用概率模型检测工具 PRISM[27] 来验证量子协议的正确性，如超密编码、量子传输和量子纠错等。之后在引入外源量子命题逻辑 (Exogenous Quantum Propositional Logic，或 EQPL)[28] 的基础上，Pedro Baltazar 等人研究了量子计算树逻辑 (Quantum Computation Tree Logic，或 QCTL) 模型检测问题，并验证了量子密钥分发 BB84 协议[14] 的正确性[29,30]。类似地，在在引入外源量子命题逻辑后，Paulo Mateus 等人则研究了量子线性时序逻辑 (Quantum Linear-time temporal Logic，或 QLTL) 的模型检测问题[31]。Timothy Davidson 等人将量子通信协议在量子通信进程 (Communicating Quantum Processes，或 CQP)[32] 中建模，并检测了由 QCTL 指定的相关属性[33,34]。Simon J. Gay 等人还开发了用于量子通信协议的模型检测工具[35,36]，但其应用限于必须建模为量子线路的协议，才能在 Stabilizer code[37] 中形式表示。Ebrahim Ardeshir-Larijani 等人则将此技术扩展到 Stabilizer code 之外，用于检测量子协议的等效性[38,39]。

此外，模型检测技术也被应用于一般量子系统的检测，包括物理系统和量子程序。Mingsheng Ying 等人采用量子自动机 (Quantum Automata，或 QA) 作为系统模型[40]，在 Birkhoff-von Neumann 量子逻辑[41] 的基础上提出了一种检测某些线性时间属性的方法[42]。Yangjia Li 等人在此基础上研究了量子自动机的可达性问题[43]。Shenggang Ying 等人还研究了量子马尔科夫链 (Quantum Markov Chains)[44] 和量子马尔科夫决策过程的模型检测问题，并提出了一种分解马尔科夫链的新量子图理论[45,46]。Yuan Feng 等人引入超算子值马尔科夫链概念，作为量子程序和通信协议的高级模型，并开发了检测一系列属性的算法[47–49]。此外，Yuan Feng 等人还开发了一个基于概率模型检测工具 Iscas Mc[50] 的量子程序和协议的模型检测工具 QPMA[51]。

随着量子计算硬件规模的快速增长，量子线路的验证成为一个重要问题。开始的研究主要集中在二元决策图 (Binary Decision Diagrams，或 BDD) 在量子计算下的推广算法，如量子信息决策图 (Quantum Information Decision Diagram，或 QuIDD)[52]，量子多值决策图 (Quantum Multiple-valued Decision Diagram，或 QMDD)[53] 等，从而对组合式量子线路进行等效性检测。最近的研究结构为张量决策图 (Tensor Decision Diagram，或 TDD)[54]。

显然，随着越来越复杂的物理可实现化的硬件出现，将会出现更加复杂的，

更加针对于的，新的验证问题。比如量子存储[55]，量子反馈网络[56]，重复至成功 (Repeated Until Success，或 RUS) 的量子线路[1]。量子模型检测的高度自动化为量子线路的验证提供了更多便利。

过去量子模型检测研究追求的是普遍性，即只针对检测量子系统的一般可达性和时间逻辑属性。未来量子模型检测的最重要目标是寻找一类更简单易于检测的属性。这是因为当前量子模型检测的执行效率非常低，且仅适用于非常小规模和深度较小的量子线路。因此，需要确定一类更简单易于检测的属性，以便当前的量子模型检测工具可以高效地进行检测[57]。这需要更多在时序逻辑上的研究工作。但同时更实用的模型检测工具也能在这一研究过程中起到一定帮助。

1.3 研究目的和重要性

本次研究的主要目的是实现资源消耗更少的量子模型检测工具。本次研究的主要挑战是计算资源的指数级增长。由于量子线路运算空间随着量子比特的线性增加而指数级膨胀，传统的计算方法并不能很好应对。而目前对量子线路表达效率较好的是基于张量网络 (Tensor Network) 的张量决策图 (Tensor Decision Diagrams)。因此本文主要工作是使用 TDD 作为数据结构，构建量子模型检测，并通过具体实验验证方案的效率提升。

如今量子计算存在大量的验证需求，因此非常需要可靠的验证工具。比如在量子计算中的自动设计领域 (Quantum Design Automation，或 QDA) 中有大量验证问题。具体来说是对同一算法的不同仿真 (synthesis) 方案是否等价。比如一类 RUS(Repeat Until Success) 线路等价性问题。图1-1展示了两个 RUS 量子线路。当线路中测量门结果为 00 时，第三个比特的输出态均为 $(I+2iZ)/\sqrt{5}|\psi\rangle$。这样的等价性检验任务，可以转换为模型检测中的可达空间是否一致。能够更快地验

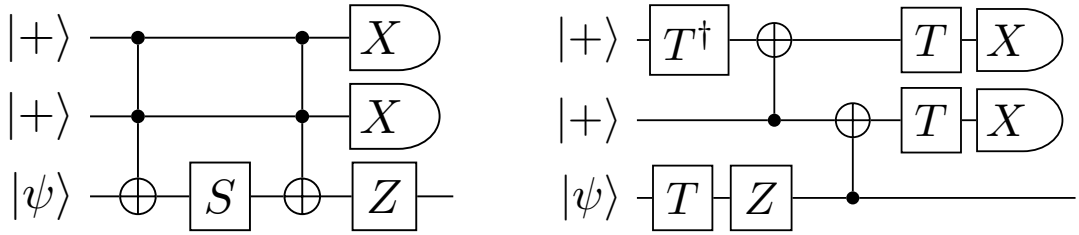


图 1-1 两个等价的 RUS 线路[1]

证不同的 synthesis 算法的等价性，在量子计算的实际研发过程中有着非常重要的实际作用。

而通过借助新的数据结构 TDD，可以更方便的表示量子状态以及量子线路，并计算最终的结果。由于基于 BDD 的模型检查算法中使用的许多优化技术可以推广到收缩量子线路张量网络上[58]。因此 TDD 特别适用于实现可达性分析和模型检查算法。这些为应用 TDD 解决量子模型检测问题提供了可能的方案。

1.4　论文结构概述

主要工作内容

本文工作中，首先调研了 TDD 在量子线路表示的优势，然后设计了量子模型检测中所需要的基础算法，最后进行了数值实验，分析量子模型检测中应用 TDD 表示的可行性。具体的工作内容如下：

(1) 在量子模型检测中，子空间的表示以及子空间的并是比较重要的内容。因此本文工作中，首先设计了基于 TDD 的对子空间进行基分解的算法，然后设计了基于 TDD 的子空间的并算法。

(2) 在 TDD 表示子空间的基础上，提出了基于 TDD 的量子迁移系统的一步迁移算法。为量子模型检测提供了算法基础。同时也设计了基于线路分割以及 TDD 结构的优化方案。

(3) 针对量子模型检测，设计实现了需要的 TDD 软件。在此基础上设计了数值实验，验证了 TDD 在一步迁移算法中的可行性，以及优化方案的执行效率提升。从而验证了 TDD 在量子模型检测中的实用性。

论文结构

本章为绪论，主要介绍了研究的背景、目的和意义。本文的剩余章节的安排如下：

第二章为**背景介绍**，该章节主要介绍了相关背景内容，为后续研究奠定理论基础。该章节具体包括量子计算，模型检测和 TDD 介绍。其中量子计算相关概念的介绍，从量子力学的数学基础，即线性代数出发，然后介绍了量子力学的四条基本假设，最后介绍了量子线路。模型检测相关概念的介绍，从经典模型检测出发，然后介绍了时序逻辑的基本概念，最后简单介绍了量子模型检测中用到的量子逻辑。TDD 相关概念的介绍，从张量网络表示量子线路出发。基于张量网络，介绍了 TDD 的定义。最后将 TDD 与类似表示进行对比，说明了本次研究选择 TDD 的原因。

第三章为 **在量子模型检测中应用 TDD 表示**，该章节主要介绍了本文的研究方法。该章节详细阐述了基于 TDD 的量子模型检测方法，包括基于 TDD 表示的子空间算法、基于 TDD 的量子一步迁移算法和针对量子模型检测的优化技术，最后还介绍了相对应的软件系统设计。在该章节中首先回顾了量子迁移系统，然后设计了几种比较重要的量子模型检测中的算法。比如对 TDD 表示的子空间进行基分解的算法，以及基于 TDD 表示的子空间的并的算法。在 TDD 表示的子空间算法基础上，设计了量子一步迁移算法。之后，讨论了针对量子模型检测的问题，对 TDD 表示的一些优化，其中包括 addition、contraction 两种线路拆分方法，以及基于窗函数和用子空间近似两种基于 TDD 结构的优化方法。该章的最后，简单介绍了 TDD 的软件系统的模块内容。

第四章为 **实验设计与评估**，该章节设计并评估了多种模型检测的优化方法，重点分析了线路拆分和 TDD 结构优化对降低资源消耗的效果。该章节的主要内

容是基于对第三章中的多种优化方法，以一步迁移算法为例，进行的数值实验。首先以同样的量子线路 Grover_15 为例，讨论了参数选取范围对线路拆分方法的影响，并给出了比较好的参数选取建议。然后选取了 Grover, QFT, BV, GHZ, QRW 五种不同的量子线路算法，根据参数对方案选择的影响，选择了合适的 addition 和 contraction 算法参数进行了实验，从而比较两种方法的适用性。最后以 Grover_40 和 QFT_100 为例，讨论了基于 TDD 结构的两种方法的适用性。

第五章为 **总结与展望**，该章节主要是对全文内容进行总结，并对本次研究内容的进一步发展进行了展望。

第 2 章　背景介绍

本章中简要介绍本文工作中的一些重要的基础知识。本章将分为三节。第一节介绍量子计算相关背景，包括量子力学和量子线路。第二节介绍模型检测相关背景，包括迁移系统，时序逻辑和量子模型检测。第三节介绍 TDD 相关背景，包含 TDD 的定义和与类似方法的比较。

2.1　量子计算简介

本节将分两个部分介绍量子计算相关背景，第一部分介绍量子力学，第二部分介绍量子线路模型，具体内容可以参考[59]。

量子计算机 (quantum computer) 是一种利用量子比特特性进行计算的设备。而量子力学是描述量子比特特性的理论。量子比特的特殊性质在于其可以处于叠加态，这与经典比特的二进制状态不同。量子比特的状态空间可以用希尔伯特空间 (Hilbert space)$\mathcal{H}$ 表示[59]，即定义了内积运算 (inner product) 的复向量空间。因此量子比特状态可以用 $\mathcal{H}$ 中的向量表示，量子操作用 $\mathcal{H}$ 上的线性算子 (linear operator) 表示。

而量子线路 (quantum circuit) 是一种描述量子计算的模型[59]。在量子线路中，通过量子比特的初始化、应用量子门、测量以及其他可能的操作的序列来构建和执行量子计算任务。量子线路通常从左向右阅读，每个量子门的作用是将输入的量子比特状态转变为输出状态，该过程可以认为是量子门的酉矩阵与输入的量子状态的乘积。

在量子计算机上可以执行各种算法和计算任务，如量子搜索[16]、量子因子分解[15] 和量子模拟[12] 等。量子计算的潜力在于其能够在某些特定问题上比经典计算机更高效地进行计算，尤其在处理大规模数据和解决复杂问题方面具有潜在优势。

2.1.1　量子力学

在量子计算中，主要使用矩阵力学作为理论基础。本小节主要介绍矩阵力学。矩阵力学以线性代数为基础，根据实验物理中总结的假设，建立了完整描述量子系统的理论体系。

线性代数

线性代数的基本研究对象是向量空间 (vector space)，也称作线性空间 (Linear Spaces)。这是一种抽象的数学结构，用于描述向量的集合。向量空间在向量加法 (Vector Addition) 和标量乘法 (scalar multiplication) 封闭[60]。对于一个向量空间，向量的类型满足特定条件。向量可以是几何向量、函数、多项式或任何满足向量

空间公理的对象。同时还需要定义标量域 (Scalars)$\mathbb{F}$，在大多数情况下，标量是实数 (Real Numbers)$\mathbb{R}$ 或复数 (Complex Numbers)$\mathbb{C}$，用于通过标量乘法运算改变向量的大小。最后向量空间的还需要遵循一系列公理。这些公理规定了向量加法和标量乘法的性质，如加法的交换律和结合律、加法和标量乘法的分配律、加法单位元的存在性 (即零向量，Zero Vector)，以及乘法单位元 (即标量 1) 的存在性。这些公理确保了向量空间内运算的一致性和预测性。

此外子空间 (Subspace) 也是是向量空间中的一个重要概念。子空间指的是一个向量空间的一个子集，同时它自身也是一个向量空间。因此子空间遵循向量空间的向量加法和标量乘法运算封闭的性质。线性算子 (Linear Operator)，也称为线性映射 (Linear Mapping) 或线性变换 (Linear Transformation)，是在向量空间中的一种特殊函数，它在两个向量空间之间映射向量，保持向量加法和标量乘法的结构。线性算子定义如定义2.1所示。

定义 2.1. [60] 设 V 和 W 是两个向量空间，算子 $T : V \to W$ 是线性算子，当且仅当 T 满足以下两个条件：

(1) 加法性 (Additivity)：对于所有 $u, v \in V$，有 $T(u+v) = T(u) + T(v)$。

(2) 齐次性 (Homogeneity)：对于所有标量 $a \in \mathbb{F}$ 和所有 $v \in V$，有 $T(av) = aT(v)$。

当 T 是 $V \to V$ 的线性算子，则称 T 是定义在向量空间 V 上的线性算子。其中恒等算子 (identity operator) 是一类特殊的线性算子，其定义如定义2.2所示。

定义 2.2. [60] 设 V 是一个向量空间，T 是向量空间 V 上的线性算子。T 是恒等算子，当且仅当 T 满足以下条件：

$$T(v) = v, \quad \forall v \in V \tag{2-1}$$

此外有的向量空间还有内积 (inner product) 和张量积 (tensor product) 操作。而内积定义如定义2.3所示。

定义 2.3. [60] 设 V 是一个向量空间，V 上有一个映射 $(\cdot,\cdot) : V \times V \to \mathbb{F}$。该映射是内积，当且仅当对于任意两个向量 u 和 v，(u, v) 满足以下性质：

(1) 共轭对称性 (Conjugate Symmetry)：$(u, v) = \overline{(v, u)}$，其中 $\overline{(v, u)}$ 是 (v, u) 的复共轭，当标量域为实数时，表现为对称性，即 $(u, v) = (v, u)$。

(2) 线性 (Linearity) 或齐次性：$(\alpha u_1 + \beta u_2, v) = \alpha(u_1, v) + \beta(u_2, v)$，其中 $\alpha, \beta \in \mathbb{F}$。

(3) 正定性 (Positive Definiteness)：$(v, v) \geq 0$ 且仅当 v 为加法单位元时等号成立。

可以看到内积是一种特殊形式的线性算子。当两个向量内积为 0 时，称这两个向量正交。而张量积则提供了一种方法来构造新的向量空间。其定义如定义2.4所示。

定义 2.4. [60] 设 V 和 W 是两个向量空间，它们的张量积 $V \otimes W$ 是一个新的向量空间，其必须满足以下性质：

(1) 对于每一对元素 $v \in V$ 和 $w \in W$，存在一个元素 $v \otimes w \in V \otimes W$，称为它们的张量积。

(2) 张量积满足双线性性（Bilinearity）：对于所有 $v, v' \in V$，$w, w' \in W$ 以及所有标量 $a, b \in \mathbb{F}$，有

$$(av + bv') \otimes w = a(v \otimes w) + b(v' \otimes w), \tag{2-2}$$

$$v \otimes (aw + bw') = a(v \otimes w) + b(v \otimes w'). \tag{2-3}$$

(3) 张量积是唯一的，意味着对于任何与 $V \otimes W$ 具有相同性质的向量空间，都存在一个自然同构映射 (Natural Isomorphism) 到 $V \otimes W$。

矩阵力学的数学基础建立在希尔伯特空间 $\mathcal{H}$，即一个可以进行内积运算的复向量空间。希尔伯特空间中的向量由复数 n 元组 $(c_1, c_2, \cdots, c_n)$ 构成，可以记作 $\mathbb{C}^n$，其中 $\mathbb{C}$ 表示复数。因此复向量也可以有以下列矩阵形式：

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \tag{2-4}$$

$\mathbb{C}^n$ 中的标量为复数。因此可以定义 $\mathbb{C}^n$ 矩阵形式的标量乘法运算为：

$$c' \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} c' \cdot c_1 \\ c' \cdot c_2 \\ \vdots \\ c' \cdot c_n \end{bmatrix} \tag{2-5}$$

其中等式右边的乘法为复数乘法，因此复数 1 为乘法单位元。类似地，$\mathbb{C}^n$ 矩阵形式的向量加法可以定义为：

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} + \begin{bmatrix} c'_1 \\ c'_2 \\ \vdots \\ c'_n \end{bmatrix} = \begin{bmatrix} c_1 + c'_1 \\ c_2 + c'_2 \\ \vdots \\ c_n + c'_n \end{bmatrix} \tag{2-6}$$

其中等式右边的乘法为复数加法，因此零向量的矩阵形式如下：

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2-7}$$

通常将该向量记作 $\mathbf{0}$。

类似地，可以定义 $\mathbb{C}^n$ 矩阵形式的内积和张量积运算为：

$$\left(\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \begin{bmatrix} c_1' \\ c_2' \\ \vdots \\ c_n' \end{bmatrix} \right) = \sum_{i=1}^{n} c_i^* c_i' \tag{2-8}$$

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \otimes \begin{bmatrix} c_1' \\ c_2' \\ \vdots \\ c_n' \end{bmatrix} = \begin{bmatrix} c_1 c_1' \\ c_1 c_2' \\ \vdots \\ c_1 c_n' \\ c_2 c_1' \\ \vdots \\ c_n c_n' \end{bmatrix} \tag{2-9}$$

在量子计算中，通常将向量 ψ 记作狄拉克右矢符号 $|\psi\rangle$。对向量 $|\psi\rangle$ 中的复数全部取共轭，可以得到 $|\psi\rangle$ 的共轭记作 $|\psi\rangle^*$。向量 $|\psi\rangle$ 的共轭转置可以记作 $|\psi\rangle^\dagger$，也称对偶向量，可以用狄拉克左矢符号 $\langle\psi|$ 表示。因此向量之间的内积也可以用以下方法表示：

$$(|\psi\rangle, |\phi\rangle) = \langle\psi|\phi\rangle, \quad \forall|\psi\rangle, |\phi\rangle \in \mathcal{H} \tag{2-10}$$

希尔伯特空间 $\mathcal{H}$ 中的线性算子集合，通常记作 $\mathcal{L}(\mathcal{H})$。对于算子 A，其伴随算子 (adjoint operator)$A^\dagger$ 的定义如定义2.5所示。在伴随算子基础上可以如定义2.6所示，定义 Hermite 算子 (Hermite operator)。

定义 2.5. [59] 设算子 $A \in \mathcal{L}(\mathcal{H})$，$A^\dagger$ 是 A 的伴随算子满足，那么 $A^\dagger$ 满足：

$$\langle\psi|A^\dagger|\phi\rangle = \langle\phi|A|\psi\rangle^*, \quad \forall|\psi\rangle, |\phi\rangle \in \mathcal{H} \tag{2-11}$$

定义 2.6. [59] 设算子 $A \in \mathcal{L}(\mathcal{H})$。算子 A 是 Hermite 算子，当且仅当 A 满足：

$$A = A^\dagger \tag{2-12}$$

而在 Hermite 算子的基础上，可以继续定义几类重要的算子，如定义2.7中的投影算子 (project oroperator)，定义2.8中的正规算子 (normal oroperator)，以及定义2.9中的酉算子 (unitary oroperator)。

定义 2.7. [59] 设算子 $A \in \mathcal{L}(\mathcal{H})$，且 A 是 Hermite 算子。那么 A 是投影算子，当且仅当 A 满足：

$$A^2 = A \tag{2-13}$$

定义 2.8. [59] 设算子 $A \in \mathcal{L}(\mathcal{H})$，且 A 是 Hermite 算子。那么 A 是正规算子，当且仅当 A 满足：

$$A^\dagger A = AA^\dagger \tag{2-14}$$

定义 2.9. [59] 设算子 $A \in \mathcal{L}(\mathcal{H})$，且 A 是 Hermite 算子。那么 A 是酉算子，当且仅当 A 满足：

$$A^\dagger A = I \tag{2-15}$$

其中 I 表示希尔伯特空间中的恒等算子。

其中酉算子保持向量之间的内积，即：

$$\begin{aligned}(U|\psi\rangle, U|\phi\rangle) &= \langle\psi|U^\dagger U|\phi\rangle \\ &= \langle\psi|I|\phi\rangle = \langle\psi|\phi\rangle = (|\psi\rangle, |\phi\rangle), \forall|\psi\rangle, \quad |\phi\rangle \in \mathcal{H}\end{aligned} \tag{2-16}$$

量子力学基本假设

在量子力学中，有四条基本假设，其数学框架是在不断实验中总结，假设，验证而得到的。下面分别简单介绍一下这四条假设。

假设 2.1. [59] 每个孤立的物理系统都可以通过一个内积复向量空间（即希尔伯特空间）来表征，此空间被视为系统的状态空间 *(state space)*。该系统的全部性质由位于状态空间内的一个单位向量，即状态向量 *(state vector)*，完整定义。

量子力学的第一条假设，提出了量子系统的描述方式。最简单的量子系统是单个量子比特 (qubit) 构成的二维空间。取 $|0\rangle$ 和 $|1\rangle$ 为该状态空间的一组标准正交基，则该状态空间中的所有状态向量 $|\psi\rangle$ 都可以用以下方式表示：

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2-17}$$

其中 $\alpha, \beta \in \mathbb{C}$，并满足：

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2-18}$$

等式 (2-18) 确保了 $|\psi\rangle$ 的模长为 1，即保证了 $|\psi\rangle$ 是单位向量：

$$\langle\psi|\psi\rangle = 1 \tag{2-19}$$

等式 (2-19) 称为状态向量的归一化条件 (normalization condition)。而 α, β 也被称为对应基状态向量的振幅 (amplitude)。

例 2.1. 式子 (2-20) 中的叠加态，处在 $|0\rangle$ 状态的振幅为 $\frac{1}{\sqrt{2}}$，处在 $|1\rangle$ 状态的振幅为 $-\frac{1}{\sqrt{2}}$。

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2-20}$$

假设 2.2. [59] *一个封闭量子系统 (closed quantum system) 的演化 (evolution) 由酉变化 (unitary operation) 体现。具体来讲，系统在 t_1 时刻的状态 $|\psi\rangle$ 与在 t_2 时刻的状态 $|\psi'\rangle$ 之间，通过一个只与 t_1 和 t_2 两个时间点相关的酉操作符 U 连接:*

$$|\psi'\rangle = U|\psi\rangle \tag{2-21}$$

酉变化可以用酉算子进行表示。对于酉算子 U，始终满足

$$UU^\dagger = I \tag{2-22}$$

这是因为这些演化都是可逆的。当令单个量子比特的状态空间的正交基 $|0\rangle$ 和 $|1\rangle$ 的矩阵形式为：

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2-23}$$

可以得到常见单比特酉变化的矩阵形式，比如 Pauli 算子：

$$I = \begin{bmatrix} 1,0 \\ 0,1 \end{bmatrix} \quad X = \begin{bmatrix} 0,1 \\ 1,0 \end{bmatrix} \quad Y = \begin{bmatrix} 0,-i \\ i,0 \end{bmatrix} \quad Z = \begin{bmatrix} 1,0 \\ 0,-1 \end{bmatrix} \tag{2-24}$$

比如 Hardmard 算子和 T 门 (也可以称为 $\pi/8$ 门)：

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1,1 \\ 1,-1 \end{bmatrix} \tag{2-25}$$

$$T = \begin{bmatrix} 1, & 0 \\ 0, & \exp(i\pi/4) \end{bmatrix} \tag{2-26}$$

假设 2.3. [59] 量子测量 *(quantum measurement)* 可以通过一系列测量算子 *(measurement operator)* M_m 刻画。这些算符作用于被测量系统的状态空间上。其中下标 m 对应实验中可能发生的测量结果。

由于测量结果的概率和为 1，因此测量算子 M_m 满足完备性，即：

$$\sum_m M_m^\dagger M_m = I \tag{2-27}$$

假设测量前量子系统的状态为 $|\psi\rangle$，那么测量后得到结果 m 的概率为

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{2-28}$$

测量后，系统的状态变为

$$|\psi'\rangle = \frac{M_m|\psi\rangle}{\sqrt{p(m)}} = \frac{M_m|\psi\rangle}{\langle\psi| M_m^\dagger M_m |\psi\rangle} \tag{2-29}$$

假设 2.4. [59] 复合系统 *(composite system)* 的状态空间是其各个子空间的状态空间的张量积。如果复合系统从 1 到 n 的子系统，并且第 i 个系统的状态为 $|\psi_i\rangle$。则复合系统的状态为 $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$。

假设2.4中的复合系统，假设了子系统之间的相互作用，并没有改变复合系统状态空间的结构。此时的复合系统处于乘积态 (product state)。例如两个处于 $|0\rangle$ 状态的单量子组成的双量子比特的复合系统，可以有以下方式的表示：

$$|0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \tag{2-30}$$

类似地，可以得到 $|01\rangle, |10\rangle, |11\rangle$ 的矩阵形式。从而可以得到双量子复合系统的操作。式子 (2-31) 为双量子复合系统中常见的受控非门操作的矩阵形式。

$$CX = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix} \tag{2-31}$$

此外还存在复合系统的量子状态不能分解成各个子系统量子状态的张量积的情况。此时复合系统处于纠缠状态 (entangled state)，复合系统不能表示为子系统的张量积形式。例如该双量子比特的复合系统的状态，就不能表示为任何单比

特状态向量的张量积：

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2-32}$$

2.1.2 量子线路

在经典计算机理论中，通常使用图灵机作为抽象模型。在讨论图灵机时，会设想一个容量无限的计算机。然而在实际中，计算机的容量是有限的。因此在实际讨论时，一般会使用线路模型，作为图灵机的替代模型。该模型在计算力方面与图灵机等价，但在许多应用场景中更为实用和符合实际。线路模型是由导线 (wire) 和逻辑门 (gate) 构成，它们分别负责信息的传递和基本计算操作。

量子计算领域也是类似情况。一般使用量子线路 (quantum curcuit) 作为量子计算的抽象模型。量子线路可以描述各种量子计算的过程。而在量子线路中，逻辑门为量子操作，导线则传递量子比特信息。量子操作主要有量子门 (quantum gate) 和量子测量。而和经典线路模型不同，量子线路一般会固定阅读方向。量子信息固定从左到右传递。因此量子线路模型也可以看作经过一系列量子门，将线路左边的输入量子状态转换为线路右边的输出量子状态的模型。

根据量子门作用的量子比特数，量子门可以被分为单量子比特门 (single qubite gate) 和多量子比特门 (multiple qubit gate)。量子门对输入量子状态的作用，可以认为是量子门的酉矩阵与输入的量子状态的乘积。

常见的单量子比特门有 Pauli 门和 Hardmard 门，矩阵形式如式子 (2-24) 和式子 (2-25) 所示。而多量子比特门中比较常见的是双量子比特门。以受控门 (controlled gate) 为例，其输入量子比特分别为控制量子比特 (control qubit) 和目标量子比特 (target qubit)。假设 U 是作用于目标量子比特的任意单比特量子操作，则称该门为受控 U 门。受控 U 门根据控制量子比特的值，决定是否将量子操作 U 作用于目标量子比特。若控制量子比特的值为 $|0\rangle$，则目标量子比特值 $|\psi\rangle$ 不变，若控制量子比特的值为 $|1\rangle$，则目标量子比特值变为 $U|\psi\rangle$。受控门中最常见的是受控非门 (controlled not gate)，其矩阵形式为式子 (2-31)。可以从矩阵形式看到当控制量子比特为 $|0\rangle$ 则不做变换；若为 $|1\rangle$ 则对目标量子比特进行比特翻转，将输入的 $|0\rangle$ 变换为 $|1\rangle$，$|1\rangle$ 变换为 $|0\rangle$。

在量子线路中，量子测量是量子计算的一个关键环节。它涉及到对量子比特的状态进行观测，从而获取信息。与传统的二进制比特不同，量子比特可以处于叠加态，同时代表 0 和 1。当对一个量子比特进行测量时，它会从其叠加态“坍缩”到一个确定的状态，即 0 或 1。这个过程是随机的，但概率可以通过量子态的波函数来预测。概率的计算过程遵循假设2.3。在量子电路中，测量通常是在计算过程的最后进行，用于读取和解释量子线路的结果。

例 2.2. 图2-1 所示的量子线路展示了一个具体的量子线路示例，可以用于制备 EPR 态。其中有单比特门 H，以及双比特门 CX。假设该量子线路的初始状态

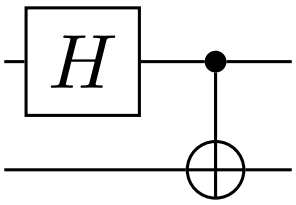


图 2-1 制备 EPR 态的量子线路图

为 $|\psi\rangle = |0\rangle\,|0\rangle$，则输出状态为 $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$。该状态是 EPR 态的一种，是量子计算中一类重要的纠缠态。对该状态的具体计算过程如式子 (2-33) 所示。

$$\begin{aligned}
|\psi_{out}\rangle &= CX \cdot H \otimes I \cdot |\psi_{in}\rangle \\
&= CX \cdot H \otimes I \cdot |00\rangle \\
&= CX \cdot H \otimes I \cdot |0\rangle \otimes |0\rangle \\
&= CX \cdot H\,|0\rangle \otimes I\,|0\rangle \\
&= CX \cdot \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle \\
&= \frac{|0\rangle|0\rangle + |1\rangle \otimes X|0\rangle}{\sqrt{2}} \\
&= \frac{|00\rangle + |11\rangle}{\sqrt{2}}
\end{aligned} \tag{2-33}$$

2.2　模型检测简介

本节将简要介绍了模型检测中的迁移系统、时序逻辑的验证以及量子模型检测。迁移系统是模型检测的基础，定义了系统状态、行为及状态转移关系。时序逻辑用于指定待验证的属性，包括状态命题和路径命题。而本次研究中，选择使用 Birkhoff-von Neumann 量子逻辑来描述量子系统的性质，命题的表示和逻辑结构。这些都为本文工作的研究提供了重要的理论基础。

2.2.1　迁移系统

迁移系统广泛应用于模型检测中待检测系统的建模，具体定义如定义2.10所示。

定义 2.10. [61] 迁移系统为一个四元组，具体包含：

$$\mathcal{M} = (S, S_0, \Sigma, R) \tag{2-34}$$

其中 S 为系统状态集合；S_0 为系统初态集合，因此满足 $S_0 \subseteq S$；$\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ 是转移关系符号的集合；$R \subseteq S \times \Sigma \times S$ 则代表转移关系。

此外迁移系统还需要有 AP 描述系统原子命题。L 是标记函数，将状态映射为状态满足的原子命题集合。需要验证的属性 φ 将表述为逻辑公式。对于某个状态集 $S' \subseteq S$，通过转移关系 R 得到的系统状态可以表示为

$$R(S') := \{t \in S \mid (s, \sigma, t) \in R \text{ 并且 } s \in S', \sigma \in \Sigma\} \tag{2-35}$$

系统的有限路径片段 π 是一个有限状态序列 $s_0, s_1 \ldots s_n$，其中 $s_0 \in S_0$。s_i 满足 $s_{i-1} \xrightarrow{\sigma_i} s_i, \sigma_i \in \Sigma$，对于所有 $0 < i \leq n$，其中 $n \geq 0$。无限路径片段 π 是一个无限状态序列 $s_0, s_1 \ldots$，使得对于所有 $i > 0$，$s_{i-1} \xrightarrow{\sigma_i} s_i, \sigma_i \in \Sigma$。在路径中 $\pi[i] = s_i, \pi[i) = s_i \ldots$。所有以 S_0 中元素为开始的路径，构成了路径集合 $Path\left(S_0\right)$。

量子模型检测的迁移系统类似。区别在于状态空间用 $\mathcal{H}$，转移关系为量子操作。量子迁移系统的具体定义如定义2.11所示。

定义 2.11. [62] 量子迁移系统为一个四元组，具体包含：

$$\mathcal{M} = \{\mathcal{H}, S, \Sigma, \mathcal{T}\} \tag{2-36}$$

其中 $\mathcal{H}$ 为希尔伯特空间，表示量子状态空间；S 为系统初态集合，因此满足 $S \subseteq \mathcal{H}$；$\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ 表示符号的集合；$\mathcal{T}$ 则代表量子系统的状态转移关系。

例 2.3. 图2-2 所示的迁移系统展示了一个简化版的可调节台灯系统。在该例子中，系统状态 $S = \{\text{off, low, high}\}$，分别代表熄灭，低亮度，高亮度状态。系统初态是 $S_0 = \{\text{off}\}$。系统行为 $\Sigma = \{\text{p1, p2}\}$，分别表示电源开关和亮度调节开关。转移关系图中已经展示。在台灯系统中，考虑到台灯的亮度，可以定义以下原子命题：

- light：表示台灯处于照明状态。
- bright：表示台灯正在高亮度照明。

因此 $L(\text{off}) = \{\varnothing\}$，$L(\text{low}) = \{\text{light}\}$，$L(\text{high}) = \{\text{light, bright}\}$。该系统的一条路径为 $\pi = \text{off, low, off}, \cdots$。该路径是路径集合 $Path(\{\text{off}\})$ 的元素。

2.2.2 时序逻辑中的可达性问题

本小节将简单介绍时序逻辑中的可达性问题，具体内容可以参考[63]。经典模型检测使用时序逻辑指定待验证的属性 φ。时序逻辑命题的运算符有以下两类。

定义 2.12. [63] 给定一个原子命题集合 AP，则状态命题公式 (State formulas)：

$$\Phi ::= a | \exists\varphi | \forall\varphi | \neg\Phi | \Phi_1 \wedge \Phi_2 \tag{2-37}$$

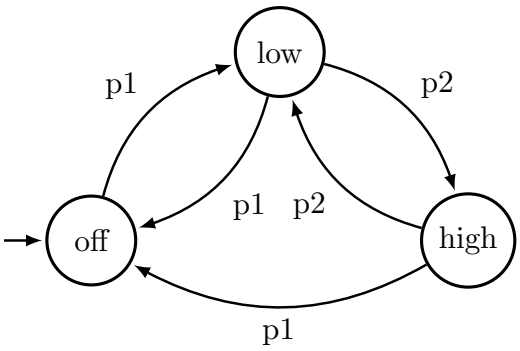


图 2-2 简化版的可调节台灯迁移系统

其中 $a \in AP$，φ 是路径命题公式。

路径命题公式 (Path formulas)：

$$\varphi := O\Phi \mid \Phi_1 U \Phi_2 \tag{2-38}$$

其中 Φ, Φ_1, Φ_2 是状态命题公式。

给定模型的一个状态为 s，路径为 π，则具体满足条件分别如下：

- $s \vDash a, iff\ (s) \vDash a$
- $s \vDash \exists\varphi, iff\ \pi \vDash \varphi$ 对一些 $\pi \in Path\,(s)$
- $s \vDash \forall\varphi, iff\ \pi \vDash \varphi$ 对所有 $\pi \in Path(s)$
- $s \vDash \neg\Phi, iff\ s \nvDash \Phi$
- $s \vDash \varphi \wedge \psi, iff\ s \vDash \varphi\ and\ s \vDash \psi$
- $\pi \vDash O\Phi, iff\ \pi\,[1) \vDash \Phi$
- $\pi \vDash \Phi_1 U \Phi_2, iff\ \exists j \geq 0.\pi\,[j) \vDash \Phi_2$ 同时对所有 $0 \leq i < j$ 有 $\pi\,[i] \vDash \Phi_1$

图2-3 展示了两种路径命题公式的直观示意图。

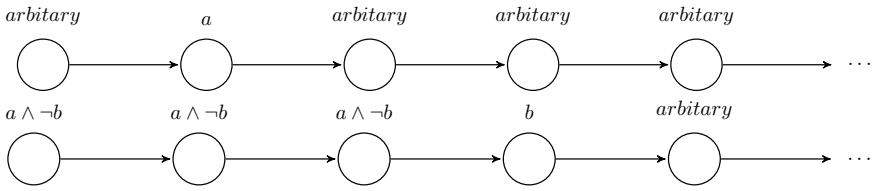


图 2-3 路径命题公式 $\pi \vDash Oa$ 与 $\pi \vDash aUb$ 的图示

在模型检测中，有三类比较重要的可达性问题，分别是可达性、持续可达性以及重复可达性。过程中主要涉及以下路径命题公式：◇ 表示最终 (eventually)，□ 表示总是 (always)，◇□ 表示总是最终 (always eventually)，□◇ 表示最终总是 (eventually always)。其中 ◇ 和 □ 具体定义为：

- $\Diamond\varphi \overset{\text{def}}{=} \text{True}U\varphi$
- $\Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$

图2-4展示了这两种路径命题公式的直观示意图。

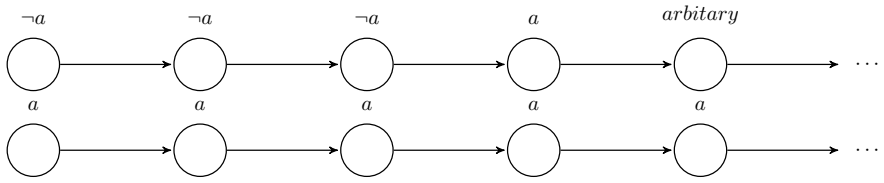


图 2-4 路径命题公式 $\pi \vDash \Diamond a$ 与 $\pi \vDash \Box a$ 的图示

具体的可满足条件为：

- $\pi \vDash \Diamond\varphi, iff \exists j \geq 0.\pi[j) \vDash \varphi$
- $\pi \vDash \Box\varphi, iff \forall j \geq 0.\pi[j) \vDash \varphi$
- $\pi \vDash \Diamond\Box\varphi, iff \exists i \geq 0.\forall j \geq i, \pi[j) \vDash \varphi$
- $\pi \vDash \Box\Diamond\varphi, iff \forall i \geq 0.\exists j \geq i, \pi[j) \vDash \varphi$

例 2.4. 在例子2.3中介绍了一个简化版的可调节台灯的迁移系统模型。状态 off 下一步一定是 low ，同时存在 off 到达状态 high 的路径，因此满足下列状态公式：

$$\text{off} \vDash \neg \text{ light} \tag{2-39}$$

$$\text{off} \vDash \neg \text{ light} \wedge \neg \text{ bright} \tag{2-40}$$

$$\text{off} \vDash \exists O\text{bright} \tag{2-41}$$

$$\text{off} \vDash \forall O\text{light} \tag{2-42}$$

由于该系统的初态是 off 。因此该系统的路径满足下列路径命题公式：

$$\pi \vDash O \text{ light} \tag{2-43}$$

$$\pi \vDash \Diamond \text{ light} \tag{2-44}$$

2.2.3 量子模型检测使用的量子逻辑

本小节将简单介绍在量子模型中使用的量子逻辑，具体内容可以参考[62]。目前量子的模型检测，主要使用 Birkhoff-von Neumann Quantum Logic 来描述量子系统的性质。Birkhoff-von Neumann 量子逻辑是一种非经典逻辑，用于描述量子力学中事件的逻辑结构。它由 Birkhoff 和 von Neumann 在 1936 年首次提出。在量子逻辑中，命题的集合不再形成布尔代数，而是形成一个投影算子的正交完备格，这与传统的逻辑系统不同。

量子命题

在 Birkhoff-von Neumann 量子逻辑中，量子系统的状态空间可以由希尔伯特空间 (Hilbert space) 来描述，每个量子命题对应希尔伯特空间的一个闭子空间。对于系统的状态 $|\psi\rangle$，如果它属于某个特定的闭子空间 $\mathcal{X}$，可以说这个命题是真的。

例 2.5. [62] 考虑以下量子逻辑命题：

- 命题 $\mathcal{X}$：在时间 t 时，量子粒子的位置 x 坐标在区间 $[a,b]$ 内。
- 命题 $\mathcal{Y}$：在时间 t 时，量子粒子的动量 y 坐标在区间 $[a,b]$ 内。

这些命题 $\mathcal{X}$ 和 $\mathcal{Y}$ 可以通过粒子的状态希尔伯特空间的特定子空间来表示。

上述将子空间视为原子命题的想法也可以用量子测量来解释。假设系统的一个基本性质由希尔伯特空间 $\mathcal{H}$ 的一个闭子空间 $\mathcal{X}$ 描述。在量子力学中，为了检查这个性质是否满足，需要对系统当前态 $|\psi\rangle$ 执行一个二元 (是或否) 测量 $\{P_{\mathcal{X}}, P_{\mathcal{X}}^{\perp}\}$，其中 $P_{\mathcal{X}}$ 和 $P_{\mathcal{X}}^{\perp}$ 分别是到 $\mathcal{X}$ 和它的正交补 $\mathcal{X}^{\perp}$ 的投影。测量结果通常是非确定性，其对应概率分别为：

- $\mathcal{X}$ 在 $|\psi\rangle$ 中满足的概率是 $\langle\psi|P_{\mathcal{X}}|\psi\rangle$。
- 不满足的概率是 $\langle\psi|P_{\mathcal{X}}^{\perp}|\psi\rangle = 1 - \langle\psi|P_{\mathcal{X}}|\psi\rangle$。

同时可以通过设置一个阈值 $\lambda \in [0,1]$ 来定义关于满足的概率的量化关系：

定义 2.13. [62] 如果 $\langle\psi|P_{\mathcal{X}}|\psi\rangle \rhd \lambda$，则称 $|\psi\rangle$ 是 $(\lambda, \rhd)$ 满足子空间 $\mathcal{X}$ 描述的命题。其中其中 $\rhd \in \{<, \leq, >, \geq\}$。

在本文工作的研究中只考虑定性满足，即阈值 λ 为 0 或 1 时的 $(\lambda, \rhd)$ 满足。显然，对于任何纯态 $|\psi\rangle$ 和子空间 $\mathcal{X}$，都有：

- 如果 $|\psi\rangle \in \mathcal{X}$，那么 $\mathcal{X}$ 在 $|\psi\rangle$ 中 $(1, \geq)$-满足；
- 如果 $|\psi\rangle \in \mathcal{X}^{\perp}$，那么 $\mathcal{X}$ 在 $|\psi\rangle$ 中 $(0, \leq)$-满足。

量子逻辑中的连接词

在数学上，这种集合的命题逻辑结构可以使用格理论 (lattice theory) 来描述，其中格中的元素对应于量子事件，格的操作则对应于逻辑运算。在确定了原子命题后，需要引入连接词，这些连接词可以用来构建更复杂的命题，以描述量子系统的复杂属性。在语义上，这些可以被视为在希尔伯特空间 $\mathcal{H}$ 的一个子空间 $S(\mathcal{H})$ 中的代数操作[62]。具体如下：

- 子空间之间的包含关系 $\subseteq$ 在 $S(\mathcal{H})$ 中是一个偏序关系，它可以理解为量子逻辑的蕴含 (元逻辑)。
- 一个子空间 $\mathcal{X}$ 的正交补 $\mathcal{X}^{\perp}$ 在量子逻辑中用作否定的解释。
- $S(\mathcal{H})$ 对交集是封闭的，即对于 $S(\mathcal{H})$ 中的任何元素族 $\{\mathcal{X}_i\}$，都有 $\bigcap_i \mathcal{X}_i \in S(\mathcal{H})$。在量子逻辑中用于表示合取。
- 对于一组子空间 $\{\mathcal{X}_i\}$，这些子空间的并定义为 $\bigvee_i \mathcal{X}_i = \operatorname{span}\left(\bigcup_i \mathcal{X}_i\right)$.。在量子逻辑中，析取被解释为子空间的并。

量子逻辑中的析取被解释为子空间的并运算。$(S(\mathcal{H}), \cap, \vee, \perp)$ 构成一个正交模格 (orthomodular lattice)，$\subseteq$ 是其序关系，这是 Birkhoff–von Neumann 量子逻辑的代数模型。

在实际应用中，通常只选择 $S(\mathcal{H})$ 的一个子集 $\boldsymbol{AP}$ 作为原子命题的集合。$\boldsymbol{AP}$ 的元素可以被看作是真正关注的命题，而其他命题可能是无关的。为了算法的目

的，通常假设 AP 是 $S(\mathcal{H})$ 的一个可列或者甚至是有限的子集，而不是作为原子命题的集合使用 $S(\mathcal{H})$ 本身，因为 $S(\mathcal{H})$ 是无穷无尽的。

量子逻辑中的满足

在量子逻辑中，“满足” (satisfaction) 意味着量子系统的状态可以在逻辑表达式指定的条件下被视为真。对于任何原子命题 $\mathcal{X} \in AP$ 和状态 $|\psi\rangle \in \mathcal{H}$，如果 $|\psi\rangle \in \mathcal{X}$，那么说状态 $|\psi\rangle$ 满足 $\mathcal{X}$。用 $L(|\psi\rangle)$ 表示在状态 $|\psi\rangle$ 中被满足的原子命题集合:

$$L(|\psi\rangle) = \{\mathcal{X} \in AP : |\psi\rangle \in \mathcal{X}\} \tag{2-45}$$

有时，需要在一个状态 $|\psi\rangle$ 和一个不在 AP 中的命题 $\mathcal{X}$ 之间建立更一般的满足关系。例如，在某些应用中，可能对一个命题 $\mathcal{X}$ 是否被一个状态 $|\psi\rangle$ 满足感兴趣，但由于内存的考虑，量子模型检测中一般只会选择了一个非常有限的不包括 $\mathcal{X}$ 的原子命题集合。

给定一个原子命题集合 AP,对于一个命题 $\mathcal{X}, \mathcal{X} \in S(\mathcal{H})$。如果 $\bigcap_{\mathcal{Y}\in L(|\psi\rangle)} \mathcal{Y} \subseteq \mathcal{X}$，那么状态 $|\psi\rangle$ 满足 $\mathcal{X}$，记作 $|\psi\rangle \vDash_{AP} \mathcal{X}$ 或简单地 $|\psi\rangle \vDash \mathcal{X}$。直观地说，$\bigcap_{\mathcal{Y}\in L(|\psi\rangle)} \mathcal{Y}$ 是可以用原子命题来定义并描述状态 $|\psi\rangle$ 的最弱陈述。因此，包含关系意味着在状态 $|\psi\rangle$ 中成立的原子命题集合共同蕴含命题 $\mathcal{X}$。而如果 $\mathcal{X} \in AP$，那么 $|\psi\rangle \vDash \mathcal{X}$ 当且仅当 $|\psi\rangle \in \mathcal{X}$。

例 2.6. 假设 $\mathcal{H}$ 是一个 n 维的希尔伯特空间，其标准正交基为 $\{|0\rangle, |1\rangle, \cdots, |n-1\rangle\}$ $(n \geq 4)$。定义例子2.2中的 EPR 态为待验证的状态，即 $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |3\rangle)$。讨论此时 $|\psi\rangle$ 的满足情况。

首先讨论将 AP 定义为与基态 $|0\rangle$ 正交的子空间时，即 $AP = \{\mathcal{Y} \in S(\mathcal{H}) : |0\rangle \perp \mathcal{Y}\}$。由于 $(|0\rangle, |\psi\rangle) = \frac{1}{\sqrt{2}}(\langle 0|0\rangle + \langle 0|3\rangle) = \frac{1}{\sqrt{2}} \neq 0$，因此 $L(|\psi\rangle) = \varnothing$。对于空集中的所有元素取交得到全空间，所以有:

$$\bigcap_{\mathcal{Y}\in L(|\psi\rangle)} \mathcal{Y} = \mathcal{H}. \tag{2-46}$$

也就是说，对于任何 $\mathcal{X} \in S(\mathcal{H})$，$|\psi\rangle \vDash \mathcal{X}$ 成立当且仅当 $\mathcal{X} = \mathcal{H}$。

而如果 AP 定义为包含基态 $|1\rangle$ 的子空间，即 $AP = \{\mathcal{X} \in S(\mathcal{H}) : |1\rangle \in \mathcal{X}\}$，那么可以得到:

$$\bigcap_{\mathcal{Y}\in L(|\psi\rangle)} \mathcal{Y} = \text{span}\{|\psi\rangle, |1\rangle\} = \text{span}\{|0\rangle, |1\rangle, |3\rangle\} \tag{2-47}$$

此时 $|\psi\rangle \vDash \mathcal{X}$ 成立当且仅当 $|\psi\rangle, |1\rangle \in \mathcal{X}$。

2.3　TDD 简介

TDD 是一种决策图样式的数据结构，用于使用张量网络表示量子线路。本节将从用张量网络表示量子线路开始，简单介绍张量网络如何转换到 TDD，最后简单介绍 TDD 与类似表示的对比。有关 TDD 的具体内容，可以参考[54]。

2.3.1　TDD 定义

张量是与一组与索引序列 $I = \{x_1, \ldots, x_n\}$ 相关联的多维线性映射。在量子计算中，可以假设只从 $\{0,1\}$ 中取值。因此，张量定义为定义2.14所示。

定义 2.14. [64] 当张量只从从 $\{0,1\}$ 中取值时，对于定义在索引序列 I 上的张量 ϕ，其可以表示为一个字典

$$\phi : \{0,1\}^I \to \mathbb{C} \tag{2-48}$$

其中 $\mathbb{C}$ 为复数域。

因此一个向量表示为 $[\alpha_0, \alpha_1]$ 的量子比特 x 可以描述为秩为 1 的张量 ϕ_x，其中 $\phi_x(0) = \alpha_0, \phi_x(1) = \alpha_1$。具有输入比特 x 和输出比特 y 的单比特门可以表示为秩为 2 的张量 ϕ_{xy}。类似的 n 比特量子门可以表示成一个秩为 $2n$ 的张量。在张量表示中，一般不区分输入和输出索引。

张量之间最重要的运算之一是收缩 (contraction)。收缩是通过对共享索引求和获得新的张量的操作。具体来说，设张量 $\gamma_{\vec{x},\vec{z}}$ 和 $\xi_{\vec{y},\vec{z}}$ 的共享索引集是 $\vec{z}$。设收缩后的新张量为 $\phi_{\vec{x},\vec{y}}$，则其表达式如式子 (2-49) 所示。

$$\phi_{\vec{x},\vec{y}}(\vec{a},\vec{b}) = \sum_{\vec{c}\in\{0,1\}^{\vec{z}}} \gamma_{\vec{x},\vec{z}}(\vec{a},\vec{c}) \cdot \xi_{\vec{y},\vec{z}}(\vec{b},\vec{c}). \tag{2-49}$$

另一个重要的张量操作是切片 (slicing)，它对应于布尔函数的余因子运算。设 ϕ 是索引序列为 I 上的张量。任取一个索引 $x \in I$，对 ϕ 进行关于 $x = c$ 与 $c \in \{0,1\}$ 的切片也是一个张量 $\phi|_{x=c}$，在索引集 $I' = I/\{x\}$ 上定义如式子 (2-50) 所示。

$$\phi|_{x=c}(\vec{a}) := \phi(c,\vec{a}) \tag{2-50}$$

对于任意 $\vec{a} \in \{0,1\}^n$。称 $\phi|_{x=0}$ 和 $\phi|_{x=1}$ 分别为 ϕ 关于 x 的负切片 (negative slicing) 和正切片 (positive slicing)。如果 $\phi|_{x=0} \neq \phi|_{x=1}$，称索引 $x \in I$ 对于 ϕ 是本质的 (essential)。

张量网络是一个无向图 $G = (V, E)$。其中顶点集 V 中的每个顶点 v 表示一个张量。边集 E 中每条边 e 代表与相邻两个张量相关联的公共索引。通过以任意顺序收缩连接的张量，可以得到一个秩为 m 的张量，其中 m 是 G 中开放边数。这个独立于收缩顺序的张量也称为该张量网络的张量表示[64]。张量网络提供了

一种新的量子线路表示方法[65]。而当给定量子线路中所有量子门的输入和输出状态的索引值，收缩掉共享的索引，就可以得到量子线路的张量表示。

TDD 是一种具有决策图和张量网络特征的数据结构[54]。它可用于表示张量，并进一步表示量子线路。与 BDD(Boolean Decision Diagrams) 类似，TDD 是一种建立在索引顺序 $I = \{x_1, \ldots, x_n\}$ 上的决策树模型。具体定义定义2.15所示。

定义 2.15. [54] TDD 是一个建立在索引顺序 $I = \{x_1, \ldots, x_n\}$ 上的有根节点，带权重的有向无环图，其中包括：

$$\mathcal{F} = (V, E, index, value, low, high, w) \tag{2-51}$$

其中:

(1) V 是一个有限节点集，被划分为非终端节点 V_N 和终端节点 V_T。用 $r_{\mathcal{F}}$ 表示 $\mathcal{F}$ 的唯一根节点。

(2) $E = \left\{(v, low(v)), (v, high(v)) : v \in V_N\right\}$ 是树中所有边集合，其中 $(v, low(v))$ 和 $(v, high(v))$ 分别称为 v 的低边和高边，分别指向该节点索引的负切片和正切片。根结点 $r_{\mathcal{F}}$ 具有唯一的入射边 $e_{\mathcal{F}}$，该入射边没有源结点。

(3) $index : V_r \to I$ 将每个非终端节点分配给 I 中的索引。

(4) $value : V_T \to \mathbb{C}$ 将每个终端节点赋予一个复数值。

(5) low 和 $high$ 都是 $V_N \to V$ 中的映射，它们分别为每个非终端节点指定其低边和高边后继。

(6) $w : E \to \mathbb{C}$ 将每条边赋予一个复数权重。特别地，$w\left(e_r\right)$ 称为 $\mathcal{F}$ 的权重，并记作 $w_{\mathcal{F}}$。

对于 TDD 中的一个节点 v，如果 v 是终端节点，则 $\phi(v) := value(v)$ 是一个秩为 0 的张量，即常数，也可以称为一个平凡 TDD。如果 v 是非终端节点，则根据定义，该节点表示的张量如式子 (2-52) 所示。

$$\phi(v) := w_0 \cdot \overline{x_v} \cdot \phi(\text{low}(v)) + w_1 \cdot x_v \cdot \phi(high(v)) \tag{2-52}$$

其中 $x_v = index(v), \overline{x_v} = 1 - index(v)$。而整个 TDD 也可以表示为：

$$\phi(\mathcal{F}) = w_{\mathcal{F}} \cdot \phi\left(r_{\mathcal{F}}\right) \tag{2-53}$$

根据定义得到的 TDD 在结构上显然比较冗余，可以进行进一步化简。具体过程是先进行规范化 (normalized)，然后进行化简 (reduced)[54]。其中规范化的目的是使得 TDD 的终端节点只包含 0 和 1，同时将终端节点的张量值沿路径逐步上移。具体规范步骤如下：

(1) 如果 v 是一个非零值 $value(v) \neq 1$ 的终端节点，则将其值设置为 1，并将每个入边的权重 w 更改为 $value(v) \cdot w$。

(2) 假设 v 是一个非终端节点，且 $\phi(v) \neq 0$。首先规范化 $\phi(low(v))$ 和 $\phi(high(v))$。完成 $\phi(low(v))$ 和 $\phi(high(v))$ 的规范化后。如果 $\phi(low(v)) \neq 0$，并且此时 $\phi(high(v)) = 0$ 或 $|w_0| \geq |w_1|$，则将 w 设置为 w_0。否则，将 w 设置为 w_1。然后更新 $w_0 := w_0/w, w_1 := w_1/w$。此时完成 v 的规范化。

在完成规范化后，此时的 TDD 终端节点只包含 0 和 1。可以进一步简化 TDD，使得 TDD 只包含一个终端节点 1，同时尽量使用重复出现的张量。具体简化步骤如下：

(1) 合并所有值为 1 的终端节点。删除所有终端 0 节点，并将它们的入边重定向到唯一的终端节点，并将它们的权重重置为 0。

(2) 将所有权重为 0 的边重定向到终端节点。如果根节点的入边权重为 0，则终端节点成为新的根，该 TDD 为空。删除所有从根节点不可达的节点，以及涉及它们的所有边。

(3) 如果一个节点 v 的低边和高边后继相同，并且其低边和高边具有相同的权重 w，则删除该节点。如果入边权重 $w = 0$，则将其传入边重定向到终端节点。否则，将其传入边重定向到其后继节点。

(4) 合并两个具有相同索引、相同 0 和 1 后继以及对应边上相同权重的节点。

为了阅读简洁，下文中无特殊说明的 TDD 均指化简后的 TDD，即 reduced tensor decision diagrams。

2.3.2 TDD 与类似表示

TDD 是一种相对较新的数据结构，用于表示和操作张量网络。与之类似的表示方法有张量网络 TN 和量子多值决策图 QMDD。本节的最后将讨论定义2.14中的张量网络 TN，定义2.15中的张量决策图 TDD，以及 QMDD[66] 三种表示的效率。

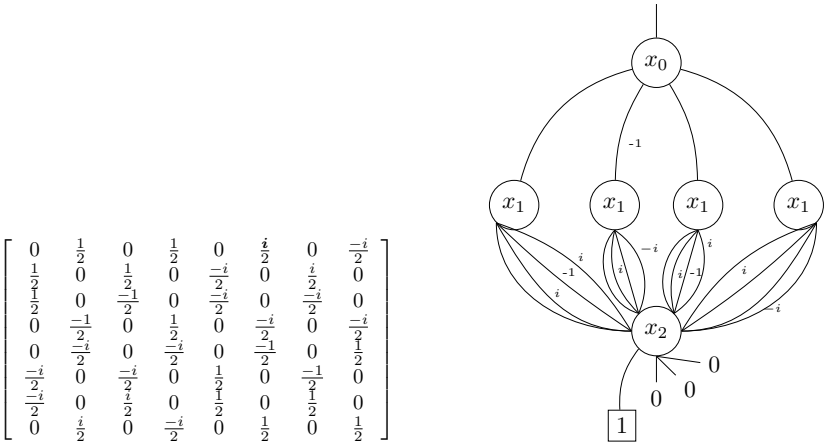
张量网络 TN 提供了量子线路更紧凑的表现形式。基于张量网络的收缩，如式子 (2-49) 所示。而随着收缩过程，直接用张量网络的张量会逐渐变大，最终整个电路会被表示为秩为 $2n$ 的张量，空间复杂度相当于矩阵表示。而 TDD 基于决策数的结构可以在收缩过程中始终保持较好的表示。因此 TDD 表示量子线路，一般会比 TN 节省更多资源。

QMDD(Quantum Multiple-Valued Decision Diagrams)，即量子多值决策图提供了一种紧凑而系统的方法来描述量子过程。目前 QMDD 已有效地用于量子线路的合成[2] 和和验证任务[67,68]。图2-5展示了一个矩阵到最终 QMDD 的一个过程。

可以看到 QMDD 和 TDD 的相似点有：

(1) 二者都存在唯一的终端顶点，值为 1。该终端顶点没有出边。

(2) 二者都有唯一一个顶点作为起始顶点，它有一个单独的入边，该入边本身没有源顶点。



(a) 一个 clifford+T 线路的矩阵形式　　(b) 图2-5a中矩阵的 QMDD 表示

图 2-5 从矩阵到 QMDD 的示例[2]

(3) 二者中的每条边 (包括指向起始顶点的边) 都有一个关联的复数值权重。

(4) 二者中的索引都按一定顺序排列，并且都满足以下两条规则：

i) 每个索引在从起始顶点到终端顶点的每条路径上最多出现一次。

ii) 来自索引排序低的非终端顶点的边指向索引排序高的非终端顶点或终端顶点。

(5) 二者都尽量复用节点，即没有两个标记索引相同的非终端顶点有相同的出边集 (即连接节点和边权重都一致)。

QMDD 和 TDD 主要不同的在于 QMDD 结构中，每个非终端节点连接有四个后继节点。而在 TDD 结构中，每一个非终端节点分别连接着两个后继节点。因此从原理上讲，如果采用相同的排序规则，量子线路的 TDD 表达方式的节点数大约是其 QMDD 表达方式节点数的两倍。但由于它们都拥有相同数目的加权边并记录了相同数量的权值（即复数）。此时 TDD 表达方式所需的内存空间与 QMDD 表达方式所需的内存空间持平。

虽然 TDD 的表示形式和 QMDD 紧凑程度相近。但是 TDD 非常灵活，可以轻松与张量网络中的一些优化技术结合以进一步提高其表现。

综上，TDD 的效率一般比 TN 的表示效率高，与 QMDD 类似。但 TDD 的高度灵活性，使得 TDD 的发展可能比 QMDD 更好。特别地，[54] 中进行了三者的数值对比实验，验证了 TDD 的表示优势。而在最新关于 TDD 的工作[69] 中实现了的 TDD 工具，在 redundancy-rich 量子线路的实验基准上相比 Google 的张量网络库快 37 倍，相比 QMDD 快 25 倍。这表明 TDD 在表示量子电路方面有着显著优势。这也是本次研究中选择 TDD 作为表示线路的最重要原因。

2.4 本章小结

本章主要介绍了本文工作的研究的相关背景知识，包括量子计算，模型检测和 TDD 的基础理论。

第一部分介绍了量子计算的相关知识，包括量子力学基本原理、量子线路模型等。量子力学建立在希尔伯特空间、算符等数学基础之上，描述了量子系统的状态、演化和测量。量子线路则是一种描述量子计算的模型，通过对量子比特进行初始化、施加量子门操作和测量等来执行量子计算任务。

第二部分介绍了模型检测的基础知识，包括迁移系统、时序逻辑和量子模型检测。迁移系统用于对待检测系统建模，定义了系统状态、行为及状态转移关系。时序逻辑用于指定待验证的属性，讨论了状态命题和路径命题等。量子模型检测则利用 Birkhoff-von Neumann 量子逻辑描述量子系统的性质，阐述了量子逻辑中命题的表示、逻辑结构及满足关系等。

第三部分从张量网络出发，介绍了 TDD 的相关定义。TDD 是基于量子张量网络的表示方法。因此这部分从张量网络表示量子线路的方法开始。然后，对 TDD 数据结构进行了简单定义。随后，本章详细介绍了如何将张量网络转换为 TDD 表示，包括 TDD 的规范化和化简策略。为了全面理解 TDD 的实际效果，将其与 QMDD 和张量网络的表示方法进行了简单讨论，分析了 TDD 的优势。

本章的背景知识为后续开展基于 TDD 的量子模型检测研究奠定了理论基础。

第 3 章　在量子模型检测中应用 TDD 表示

张量决策图 (TDD) 是一种基于张量网络，结合了二元决策图的表示优势的数据结构。TDD 针对于量子计算，可以有很好的空间优势。同时 TDD 非常灵活，可以很方便的结合过去在二元决策图的模型检测算法的优化技术。本章将主要介绍在本文工作中使用的方法，因此将主要介绍如何应用 TDD 进行模型检测并作出针对性改进，最后介绍本文工作中的软件实现。

3.1　将量子系统建模为量子迁移系统

在2.2.1节中介绍了迁移系统，其中定义2.11给出了量子迁移系统的简单定义。这里简单回顾一下。对于一个希尔伯特空间 $\mathcal{H}$ 。基于 $\mathcal{H}$ 的量子转移系统 $\mathcal{M}$ 可以表述为四元组 $(\mathcal{H}, S, \Sigma, \mathcal{T})$，这里 S 作为 $\mathcal{H}$ 的一个封闭子空间，被定义为初始空间。$\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ 是一系列符号集合，$\mathcal{T} = \{\mathcal{T}_\sigma, \sigma \in \Sigma\}$ 则代表对 $\mathcal{H}$ 执行的一组量子操作。

例 3.1. 一个单量子比特的系统可能遭受两种潜在的噪声影响：比特位翻转和相位翻转。如果不能准确判断会发生哪一种噪声，这样的系统可以被表达为一个量子转移系统 $\mathcal{M} = \left(\mathcal{H}_2, S, \{1,2\}, \{\mathcal{T}_1, \mathcal{T}_2\}\right)$，这里 S 代表 $\mathcal{H}$ 中的一个子空间，$\mathcal{T}_1 = \{\sqrt{p}I, \sqrt{1-p}X\}$ 和 $\mathcal{T}_2 = \{\sqrt{p}I, \sqrt{1-p}Z\}$。$I, X, Z$ 分别代表恒等矩阵、Pauli X 矩阵和 Pauli Z 矩阵。

例 3.2. 量子线路也可以表达为一个量子迁移系统。图3-1展示了实现两量子比特 Grover 迭代的线路[16]，这是 Grover 算法的一个基本过程，用于找到布尔函数 $f(x) = 1$ 的解。该类算法需要验证的属性是能否始终进入布尔函数解所在的子空间。对于该线路，第一个 CCX 门表示搜索用的 oracle，实现了 $O|x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle$，其中 $f(x) = x_1 \wedge x_2$。因此，该线路 oracle 的解是 11 。其他门实现了一个补 $2|\psi\rangle\langle\psi| - I$，其中 $|\psi\rangle = \frac{1}{\sqrt{2^3}} \sum_{i=0}^{2^3-1} |i\rangle$。给定输入状态 $|{+}{+}{-}\rangle = \frac{1}{2}\sum_{i=0}^{3}|i\rangle|-\rangle$，线路首先将状态变为 $\frac{1}{2}\sum_{i=0}^{2}|i\rangle|-\rangle - \frac{1}{2}|11\rangle|-\rangle$，然后变为 $|11\rangle|-\rangle$。对于状态空间 $S = span\{|{+}{+}{-}\rangle, |11{-}\rangle\}$。当目前系统状态 $|\varphi\rangle \in S$，下一步系统状态总会在 S 中。

因此该系统可以表示为量子迁移系统 $(\mathcal{H}_8, S, \{1\}, \mathcal{T})$ 进行建模，其中 $\mathcal{T}_1 = (2|\psi\rangle\langle\psi| - I)O$，需要检查的属性是 $\mathcal{T}_1(S) = S$。

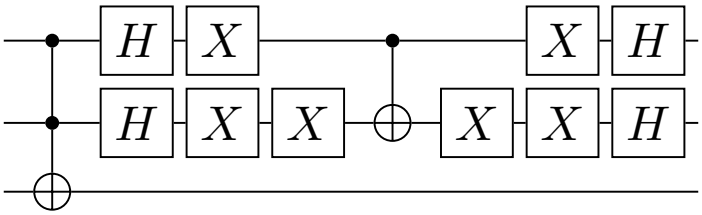


图 3-1 Grover_3 的量子线路图

3.2 基于 TDD 表示的子空间算法

3.2.1 子空间的分解

在2.2.3节中介绍了量子逻辑中的原子命题用希尔伯特空间中的子空间表示。因此在量子模型检测中，需要用到子空间的表示。而一种比较方便的表示子空间的办法是将子空间表示为一组基态。给定子空间投影算子的矩阵形式，如果有一个非零的列向量，那么在该向量的方向上就可以找到一个基向量。然后，通过正交化过程，可以递归地找到原始子空间的一组正交基向量。

但是使用矩阵表示进行所有列的遍历会遇到很高的复杂度。而通过使用子空间投影算子的 TDD 表示，就可以可以容易地找到第一个非零列。具体方法是寻找该 TDD 表示的最左侧非零路径。通过这种图的方法避免了在计算过程中需要显式表示相应的向量，也减少了复杂度 。综上所述，算法1给出了一个计算子空间的一组正交基的方法。在此算法基础上，可以给出根据当前系统状态计算下一步系统状态的算法2.

算法 1 给出投影算子 P 的一组正交基

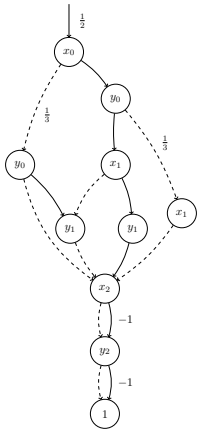
输入： 子空间 S 的 TDD 形式投影算子 $P = P_S$
输出： P 的一组正交基分解 B

$B \leftarrow \{\}$
如果 P 是空的 **那么 返回** B
否则
    $|i\rangle \leftarrow$ TDD 表示 P 中最左侧非零路径所表示的列号 i
    $|u_i\rangle \leftarrow$ 由在 TDD 表示 P 中具有列号 $|i\rangle$ 的路径表示的状态
    $|v_i\rangle \leftarrow \frac{|u_i\rangle}{\||u_i\rangle\|}$
    $P \leftarrow P - |v_i\rangle\langle v_i|$
    $B' \leftarrow$ 递归调用本算法，得到新的 P 的计算基分解
    $B \leftarrow B \cup \{|v_i\rangle\} \cup B'$
结束 如果
返回 B

例 3.3. 例子3.2将 Grover 算法的线路建模为量子迁移系统。其中用到了子空间 $S = span\{|++-\rangle, |11-\rangle\}$。相应投影算子 P 的矩阵和 TDD 表示如图 3-2 所示。根据该 TDD 表示求解原子空间的基的过程如下：

$$P = \frac{1}{6}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 \end{bmatrix}$$

(a) 投影算子 P 的矩阵形式



(b) 投影算子 P 的 TDD 形式

图 3-2 子空间 $S = span\{|++-\rangle, |11-\rangle\}$ 投影算子的矩阵和 TDD 表示

(1) 应用图算法,得到最左侧路径对应于 $(x_0, x_1, x_2, y_0, y_1, y_2) = (0,0,0,0,0,0)$, 这意味着投影算子的第一列非零。

(2) 遍历所有 $(x_0, x_1, x_2) = (0,0,0)$ 的路径,得到向量 $|v_1\rangle = \frac{1}{6}[1,-1,1,-1,1,-1,0,0]$。

(3) $|v_1\rangle$ 标准化为 $|v_1\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)|-\rangle$。

(4) 设 $P' = P - |v_1\rangle\langle v_1|$。那么 P' 等于 $|11-\rangle\langle 11-|$。

(5) 对 P' 重复上述的过程，获得另一组基 $|v_2\rangle = |11\rangle|-\rangle$，此时 TDD 为空。因此 $|v_1\rangle, |v_2\rangle$ 是子空间 S 的一个基。

3.2.2 子空间的并

在2.2.3节中介绍了量子逻辑中的连接词。其中子空间之间的包含理解为量子逻辑的蕴含；子空间的正交补理解为量子逻辑的否定；子空间的交理解为量子逻辑的合取；子空间的并理解为量子逻辑的析取。其中最复杂的是子空间的并。本小节主要讨论在 TDD 表示下子空间的并。

设 $S = S_1 \vee S_2$ 为两个子空间 S_1, S_2 的并集。假设 $B_1 = \{|\psi_{11}\rangle, \cdots, |\psi_{1k}\rangle\}$ 和 $B_2 = \{|\psi_{21}\rangle, \cdots, |\psi_{2l}\rangle\}$ 分别是 S_1 和 S_2 的一组正交基。子空间 S 的一组正交基 B 可以通过格拉姆-施密特正交化方法 (Gram–Schmidt process) 计算而来，具体方法如下。

(1) 令 $B = B_1$，并定义 $P = \sum_{j=1}^{k} |\psi_{1j}\rangle\langle\psi_{1j}|$。

(2) 遍历 B_2 中的基向量。假设当前向量为 $|\psi_{2j}\rangle$。计算 $|u_j\rangle = |\psi_{2j}\rangle - P|\psi_{2j}\rangle$ 。

(3) 如果 $|u_j\rangle$ 为 0，则考虑下一个向量；否则，此时它与 P 正交。先对 $|u_j\rangle$ 标准化为 $|v_j\rangle = \frac{|u_j\rangle}{||u_j\rangle|}$，将 $|v_j\rangle$ 添加到 B 中。同时，还将 P 更新为 $P + |v_j\rangle\langle v_j|$。

(4) 重复上述过程直到遍历完 B_2 中的所有元素。此时 B 将成为 S 的一个基，P 将成为到 S 的投影算子。

这个过程展示了如何从一组生成向量开始，通过计算和正交化过程，构建出一个复合空间的正交归一化基。在量子计算和量子信息中，这种方法特别有用，

因为它允许准确地描述和操控量子态的子空间。

例 3.4. 例子3.2中对 Grover_3 线路建模，例子3.3对涉及的子空间进行了分解。考虑分解的逆过程，设 P_1, P_2 为两个一维子空间 S_1, S_2 的投影算子，分别由 $B_1 = \{|++-\rangle\}$ 和 $B_2 = \{|11-\rangle\}$ 生成。显然，$P_1 = |++-\rangle\langle++-|$。将 B_1 完善为 $S = S_1 \vee S_2$ 的一个基。计算得到 $|u\rangle = |11-\rangle - P_1|11-\rangle = |11-\rangle - \frac{1}{4}|++-\rangle = [-\frac{1}{4}, \frac{1}{4}, -\frac{1}{4}, \frac{1}{4}, -\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, -\frac{3}{4}]^T$，可以被标准化为 $|v\rangle = -\frac{1}{2\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle - 3|11\rangle)|-\rangle$。那么 $B = \{|++-\rangle, |v\rangle\}$ 是一个正交归一化基，且 $P = P_1 + |v\rangle\langle v|$ 是相应的投影算子。

3.3 计算一步迁移

结合子空间的分解算法，可以给出对于量子系统的一步映射算法，具体如算法2所示。

算法 2 基于迁移系统的一步映射算法

输入： 一个量子迁移系统 $(\mathcal{H}, S, \Sigma, \mathcal{T})$, 其中转移关系 $\mathcal{T} = \{\mathcal{T}_\sigma \mid \sigma \in \Sigma\}$，$\mathcal{T}_\sigma = \{E_{\sigma,j_\sigma}\}$

输出： 系统下一步状态 $\mathcal{T}(S)$ 的投影算子 P

1: $P \leftarrow 0$
2: $B \leftarrow$ 调用算法1，得到 S 的计算基分解
3: $K \leftarrow \cup_{\sigma,j_\sigma}\{E_{\sigma,j_\sigma}\}$
4: **遍历** $|\psi\rangle$ **在** B, E **在** K **执行**
5: $\quad |\phi\rangle \leftarrow$ 收缩 $|\psi\rangle$ 和 E 所有相同索引
6: $\quad P = P \vee \text{span}\{|\phi\rangle\}$
7: **结束 遍历**
8: **返回** P

在此基础上，可以计算该量子迁移系统 $(\mathcal{H}, S, \Sigma, \mathcal{T})$ 的可达空间。具体方法如下：

(1) 初始化可达空间为 $R = S$，并计算可达空间的投影算子 P_R。

(2) 调用算法2，计算系统 $(\mathcal{H}, S, \Sigma, \mathcal{T})$ 下一步状态 $S' = \mathcal{T}(S)$。

(3) 调用算法1，分解子空间得到 S' 的一组基 $B = \{|\psi_1\rangle, \cdots, |\psi_n\rangle\}$。同时初始化一个空的状态空间 S''。

(4) 遍历 B 中的基，假设当前向量为 $|\psi_i\rangle$，计算得到 $|u_i\rangle = P|\psi_i\rangle$，并标准化为 $|v_j\rangle = \frac{|u_j\rangle}{||u_j\rangle|}$。

(5) 当 $|v_j\rangle$ 为 0 时，考虑下一个向量；否则，此时 $|v_j\rangle$ 与 P 正交，将该向量张开的空间并到子空间 S'' 中，即 $S'' = S'' \vee span\{|v_j\rangle\}$。同时更新可达空间 $R = R \vee span\{|v_j\rangle\}$，即将 P 更新为 $P + |v_j\rangle\langle v_j|$。

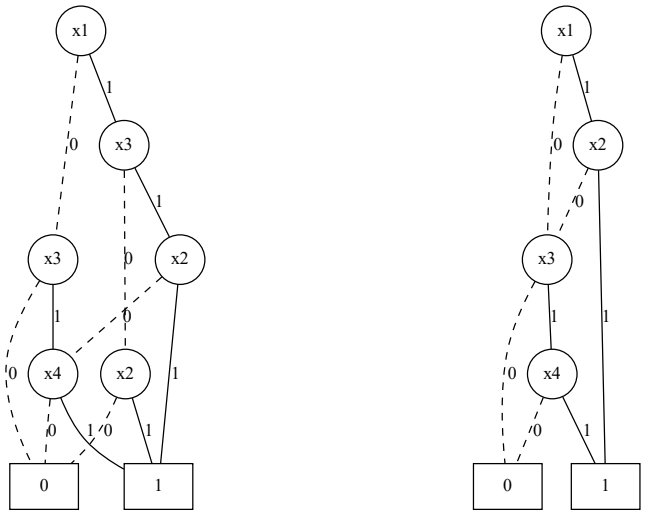
(6) 遍历结束后，如果 S'' 为空，说明该量子系统的可达空间已经收敛，即此时的 R 就是系统的可达空间。否则用 S'' 作为量子系统的初态子空间，计算迁移系统 $(\mathcal{H}, S'', \Sigma, \mathcal{T})$ 的下一步状态，并重复上述3到5的过程。

3.4　针对模型检测的改进

本文工作的主要目的是借助 TDD 数据结构，构建能快速计算量子模型检测中可达问题的方案。本文工作的主要挑战在于尽可能减少程序的运行时间以及空间资源。为此，需要采用一系列方法来开发更有效的算法，以优化 TDD 操作和收缩张量网络。其中包括开发新技术来分割线路和优化 TDD 结构。下面简单介绍一下具体改进方法。

索引顺序的调整

在 BDD 中，索引的顺序很重要。因为索引顺序会直接影响 BDD 的大小。一个良好的变量顺序可以使得 BDD 比一个糟糕的变量顺序小得多。图3-3的了两张图都表示了布尔函数 $f(x1,x2,x3,x4) = x1x2 + x3x4$。但图3-3b的结构更简单，图中边和节点的数目更少。其中图3-3a的索引顺序为 {x1,x3,x2,x4}，图3-3b的索引顺序为 {x1,x2,x3,x4}。找到一个好的索引顺序是一个 NP 问题。在工程实现中，目前只能通过小规模线路上寻求规律，然后在更大规模线路中应用较优顺序。



(a) 索引顺序为 {x1,x3,x2,x4}　　**(b) 索引顺序为 {x1,x2,x3,x4}**

图 3-3 布尔函数 $f(x1,x2,x3,x4) = x1x2 + x3x4$ 在不同索引顺序下的 BDD

addition 的线路拆分方案

关于常用的量子线路划分方法，第一种被称为 addition[70]。将量子线路视为张量网络，首先将一个量子线路 C 转换成无向图 G。G 中的每个节点表示量子线路的一个索引，并且如果它们是相同门的输入或输出索引，则在 G 中连接两个节点。并且当满足以下两个条件之一时输入和输出索引不变：

- 是对角线量子门的输入和输出索引；
- 是受控门的控制比特位的输入和输出索引。

在得到无向图后，就可以通过对连通度高索引进行张量切片，从而对量子线路进行划分。

例 3.5. 图3-4展示了图3-1中 Grover_3 线路图的索引链接图。该图描述了量子线路的连通性，通过选择图中连通度最大的索引可以对线路进行分割。因此选择图中连通度较大的 $x_1^1, x_1^3 x_2^1$ 可以对线路进行较好的划分。

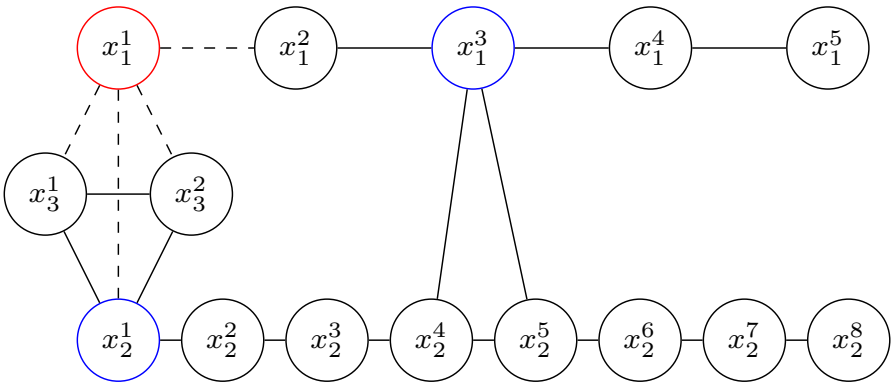


图 3-4 Grover_3 的索引连接图

contraction 的线路拆分方案

另一种常用的线路划分方法成为 contraction。该方法是2.3.2小节中的 TDD part I 的延申。在这一方法中，将量子线路划分为若干个较小的部分，其收缩等于原始线路。应用两个预设整数参数 $k1$ 和 $k2$，将线路划分为若干小线路。其中每个小线路涉及最多 $k1$ 个量子比特，并且与至多跨越不同部件的 $k2$ 个多比特门相连。

例 3.6. 图3-5展示了对 Bit flip 线路进行 k1=3,k2=2 的拆分结果。

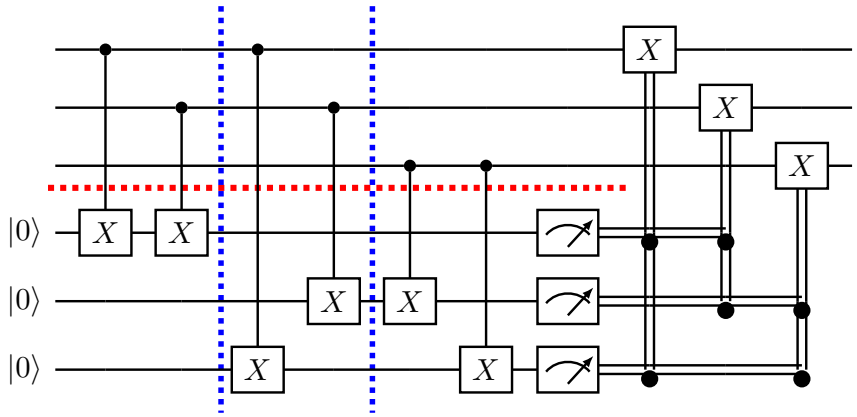
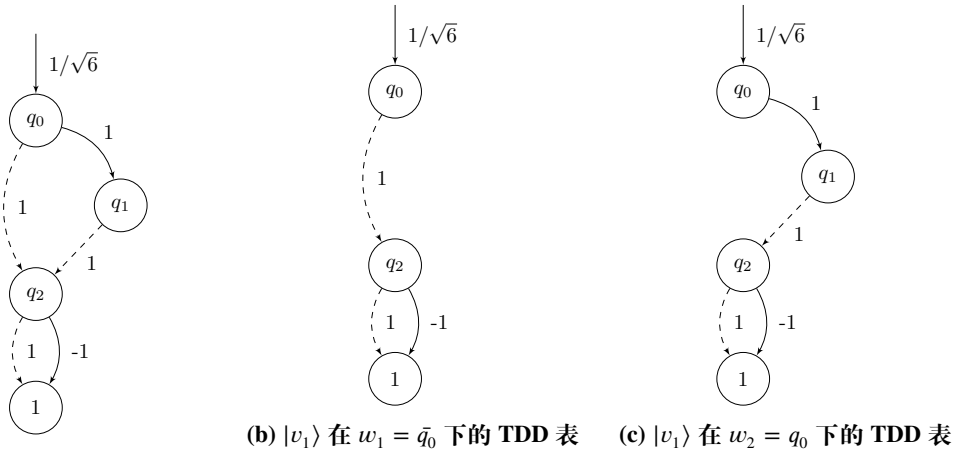


图 3-5 对 Bit flip 线路进行 contraction 的拆分

基于窗函数对 TDD 分割

按照[71]中关于经典模型检测方法的讨论，可以设计基于 TDD 的划分法。即利用布尔函数将一个大张量细分成若干个小张量的优化方法。设 φ 是一个含有 n



(a) $|v_1\rangle$ 的 TDD 表示　**(b) $|v_1\rangle$ 在 $w_1 = \bar{q}_0$ 下的 TDD 表示**　**(c) $|v_1\rangle$ 在 $w_2 = q_0$ 下的 TDD 表示**

图 3-6 $|v_1\rangle = \frac{\sqrt{2}}{\sqrt{3}}|0\rangle|+\rangle|-\rangle + \frac{1}{\sqrt{3}}|1\rangle|0\rangle|-\rangle$ 的 TDD 表示与窗口函数分解

个索引的张量。从 $\{0,1\}^n$ 到 $\{0,1\}$ 的集合中选取一系列窗口布尔函数 (window boolean function)$\{w_1,\cdots,w_k\}$，这些函数对于同一输入，始终满足以下条件：

- $w_1 + \cdots + w_k = 1$
- 对任意 $i \neq j$，$w_i \cdot w_j = 0$

因此可以看到对于一个索引 a，有且只有一个窗口函数 $w_i(a) = 1$，其他窗口函数都为 0。对一个张量 φ，令 $\varphi_i = \varphi \cdot w_i$。此时如果 $w_i(a) = 1$，则 $\varphi(a) = \varphi_i(a)$；而当 $w_i(a) = 0$，则 $\varphi_i(a) = 0$，此时存在另一个窗口函数使得 $w_j(a) = 1$，$\varphi(a) = \varphi_j(a)$。进而，可以得出 $\varphi = \varphi_1 + \cdots + \varphi_k$。基于这样的张量划分方法，也可以对 TDD 进行划分。

例 3.7. 以例子3.3中的量子态 $|v_1\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)|-\rangle = \frac{\sqrt{2}}{\sqrt{3}}|0\rangle|+\rangle|-\rangle + \frac{1}{\sqrt{3}}|1\rangle|0\rangle|-\rangle$ 作为示例。将 q_0 量子比特看作是一个布尔变量，设 $w_1 = \bar{q}_0$ 和 $w_2 = q_0$。窗口函数 $w_1 + w_2 = 1$，并且互斥。$|v_1\rangle \cdot w_1$ 与 $|v_1\rangle \cdot w_2$ 分别代表（非标准化的）状态 $\frac{\sqrt{2}}{\sqrt{3}}|0\rangle|+\rangle|-\rangle$ 和 $\frac{1}{\sqrt{3}}|1\rangle|0\rangle|-\rangle$。$|v_1\rangle$ 的 TDD 表示及其窗口函数分割的 TDD 可在图3-6 中查看。

用子空间近似 TDD 表示 $|\psi\rangle$

通过上述方法的优化，最终得到的 TDD 可能依然偏大。在很多实际应用中，对子空间进行一个合理的过度估计已足够，这与经典案例中的做法相似[72–74]。针对特定量子态 $|\psi\rangle$，能够通过包含它的适当子空间来近似 $|\psi\rangle$。例如，假设已通过张量加法或窗口函数分割将量子态分为 $|\psi\rangle = |\psi_1\rangle + |\psi_2\rangle$。于是，可以通过 $span\{|\psi_1\rangle, |\psi_2\rangle\}$ 子空间近似地表示 $|\psi\rangle$。

进一步地，也可以对量子态 $|\psi\rangle$ 进行张量积估计。即找到一系列张量积态 $|\psi_1\rangle,\cdots,|\psi_k\rangle$，使得 $|\psi\rangle$ 位于它们张成的空间之内。这样，便能够计算出一个更大系统的映射。但这种方法的成本在于过估计子空间的维度可能会过大。给定

量子态 $|\psi\rangle$，若 $|\psi\rangle = \sum_{j=0}^{2^{k-1}-1} |j\rangle |\phi\rangle |\gamma_j\rangle$，认为它的第 k 个量子比特是可分割的。对于每个态 $|\psi\rangle$，首先探查是否存在可分割的量子比特。若存在，提取出相应的态并移除该量子比特；否则，将采用那些振幅非零的计算基态进行估计。

例 3.8. 再次以例子3.3中的量子态 $|v_1\rangle = \frac{1}{\sqrt{3}}(|00\rangle+|01\rangle+|10\rangle)|-\rangle$ 为例。图 3-6a给出了该量子态的 TDD 表示。可以发现其第三个量子比特 q_2 是可分割的，即所有的 q_0，q_1 都会经过同样的 q_2 结点。q_2 对应的量子态为 $|-\rangle$。在移除这个量子比特后，得到 $|\psi\rangle = \frac{1}{\sqrt{3}}(|00\rangle+|01\rangle+|10\rangle)$，该态属于子空间 $span\{|00\rangle, |01\rangle, |10\rangle\}$。从而为态 $|v_1\rangle$ 找到了一种张量积的近似表示 $\{|00-\rangle, |01-\rangle, |10-\rangle\}$。

3.5 软件系统实现

为了实现软件的高效运行，模块化设计至关重要。每个模块在软件系统中扮演着关键角色，并且具有特定的功能和目的。图3-7包含了本次研究中必要的软件模块，展示了各个软件模块的调用关系。具体模块的作用如下：

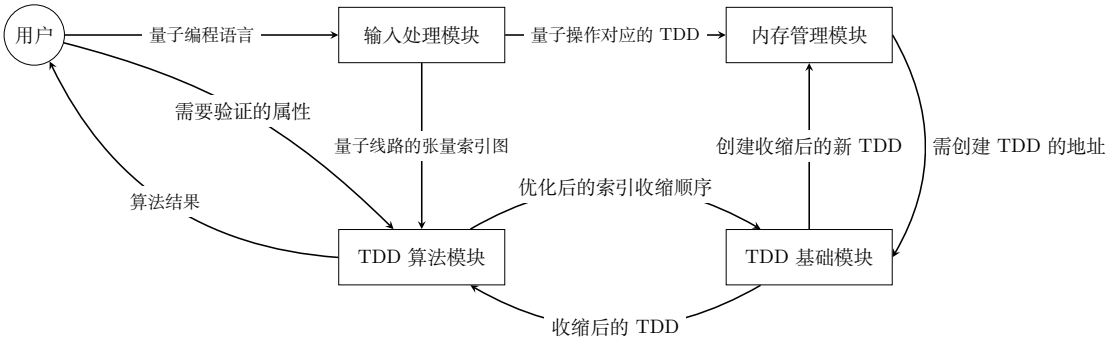


图 3-7 软件模块之间的调用关系

- **输入处理模块**：该模块的主要职责是处理输入数据，例如接收用 OpenQASM 格式编写的量子算法代码。其核心功能是将根据代码对应的量子操作，创建相应的 TDD，并生成量子线路对应的张量索引图。鉴于当前存在多种量子编程语言，此模块的模块化处理能够显著提升系统的灵活性和兼容性。
- **TDD 算法模块**：此模块为 TDD 提供更复杂的算法支持。例如，它能够调整节点收缩的顺序，以优化系统运行效率。该模块还可以导出 TDD 的树状结构图。树状结构图的导出功能则有助于用户更好地理解和分析 TDD 的结构。此外，它还能执行其他高级功能，如检验 TDD 是否存在于特定子空间中，对 TDD 进行分割。因此该模块的输入是量子线路的张量索引图，以及计算过程和计算结果的 TDD。该模块的输出多种多样，可以是线路对应的 TDD，也可能是某条属性是否满足。该模块是软件系统的核心，也是可拓展性最好的模块。
- **TDD 基础模块**：该模块主要执行 TDD 之间的操作，例如 TDD 之间的收缩或加法。该模块的输入一般是张量收缩的顺序，输出是满足某些条件或者全部张量收缩后的 TDD。

- **内存管理模块**：本模块负责管理 TDD 节点的存储和维护。每当需要创建新的 TDD 节点时，它会运用哈希算法与现有节点进行对比，以避免重复创建相同节点。因此该模块的输入主要是某个需要的 TDD，输出则会返回该 TDD 的存储地址。该模块不仅减少了内存占用，还提高了软件系统的运行效率。

例 3.9. 以例子3.2中的 Grover_3 量子线路的量子模型迁移系统为例。其对应的 OpenQASM 格式的量子编程代码如代码段 1 所示。

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[3];
ccx q[0],q[1],q[2];
h q[0];
h q[1];
x q[0];
x q[1];
h q[1];
cx q[0],q[1];
h q[1];
x q[0];
x q[1];
h q[0];
h q[1];
```

代码段 1 Grover_3 量子线路的 qasm 代码

输入处理模块首先遍历量子线路，得到如图3-4的量子索引图。然后根据每个量子门对应的矩阵形式和分配到的索引创建 TDD。之后输入处理模块将索引图输出给 TDD 算法模块。

TDD 算法模块根据张量索引图和实际验证的属性，得到需要收缩的索引和对应的索引顺序。在本例中，模型待验证的属性是 $\mathcal{T}_1(S) = S$。其中子空间 S 的投影算子 TDD 结构如图3-2b所示。$\mathcal{T}_1$ 为整个量子线路，因此需要收缩整个量子线路。按照图3-4中的索引图对量子线路的收缩不进行优化，直接收缩后，得到 TDD 结构如图3-8。将图3-8和图3-2b的两个 TDD 收缩后，得到的新的 TDD 就是子空间 $\mathcal{T}_1(S)$ 投影算子的 TDD 表示。验证新的 TDD 表示是否在子空间 S 就可以验证属性。结果表明该量子迁移系统满足属性。

在本例的验证过程中，存在大量的优化空间。例如转移关系的 TDD 生成，即量子线路的收缩，可以通过量子线路拆分技术优化。子空间的 TDD 表示也可以通过子空间近似表示进行优化。如例3.8中对 S 的空间表示进行了近似。而量子空间与转移的 TDD 都可以应用基于 TDD 结构的窗函数进行优化。如例3.3中对 S 的 TDD 结构进行了窗函数的优化。TDD 算法模型中的优化方案是提升量子模型检测效率的关键。

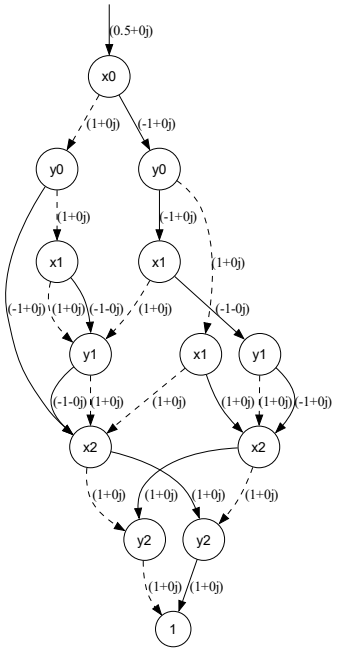


图 3-8 图3-1中 Grover_3 量子线路的 TDD 形式

3.6 本章小结

本章主要介绍了在本文工作中采用的基于张量决策图 (TDD) 的量子模型检测方法。首先简单回顾了如何将量子系统建模为量子迁移系统。然后给出了计算子空间基、子空间并运算、以及系统一步迁移的算法。为加速模型检测，本章还介绍了一些针对性优化方法，包括索引顺序调整、addition 线路拆分、contraction 线路拆分、基于 TDD 的划分，以及近似 TDD 表示等技术。最后，概述了软件系统的模块化设计，包括输入处理、内存管理、TDD 基础运算和 TDD 算法等模块。

通过上述方法，本文以 TDD 为数据结构，构建了一种能高效计算量子模型检测中可达性分析问题的解决方案。TDD 结合了 BDD 的紧凑表示和张量网络的计算优势，为解决量子计算中的模型检测问题提供了新的可能性。软件系统的模块化设计也为高效实现这一方案奠定了基础。总的来说，本章介绍的技术为量子模型检测提供了一种有前景的新方法。

第 4 章　实验设计与评估

在上一章节中，详细介绍了 TDD 表示，如何在量子模型检测中应用 TDD，并在最后简单介绍整体软件设计。本文工作的目的是应用 TDD 的表示能力，降低量子模型检测的资源消耗，为大规模的量子系统验证提供更实用的工具。其中一步迁移算法，即量子迁移系统的一步映射是模型检测中，反复用的算法。可以说利用更少的资源进行一步迁移，就可以降低计算量子系统的可达空间时的资源消耗。因此本章节的实验主要围绕一步迁移算法，讨论各种优化方案的实际效果。

4.1　实验设计

本小节主要介绍具体的实验设计。在实验测试基准方面，本次研究选择了五种不同类型的量子算法。在实验的量化方面，本次研究用程序运行时间量化优化方案的时间资源，用运行过程中的最大节点数量化优化方案的空间资源。

实验基准设计

本次研究中，选取了五类不同的量子算法作为基准实验。分别是 Grover，QFT，BV，GHZ，QRW 算法。

Grover 算法，是一种量子搜索算法[16]。它能够在未排序的数据库中查找特定元素，其搜索效率远超经典算法。Grover 算法展示了量子计算在搜索问题上的指数级别的加速能力，是一类非常重要的量子算法。Grover 算法的量子电路包括初始化、Oracle 和反射（也称为 Grover 演算）几个关键部分。初始化过程将所有量子位置于叠加状态，以确保搜索空间的均匀覆盖。Oracle 是根据搜索目标设计的，它能够标记出目标态。最后，反射操作增强了目标态的概率幅度，而减少了非目标态的概率幅度。重复 Oracle 和反射操作可以显著提高找到目标项的概率。本次实验中选取了搜索 $11\cdots1$ 的 Oracle。图3-1是本次实验中 Grover_3 的量子线路图。其余的 Grover 算法的量子线路图设计类似。

QFT 算法 (Quantum Frourier transform)，即量子傅里叶变换是量子算法中的核心组成部分，特别是在量子计算中的素数分解和周期性问题求解中占有重要地位[59]。QFT 能够高效地将量子态从时域转换到频域，是许多量子算法，包括著名的 Shor 算法的基础。QFT 的电路设计利用了量子位之间的相位关系来执行变换。核心组件是一系列的受控旋转门，它们在不同的量子位上以不同的旋转角度施加，从而实现复数矩阵的乘法。QFT 电路的高效性在于其使用了远少于经典傅里叶变换所需的运算步骤。图4-1是本次实验中 QFT_3 的量子线路图。其余的 QFT 算法的量子线路图设计类似。

BV 算法，即 Bernstein-Vazirani 算法展示了量子计算在解决特定的线性代数

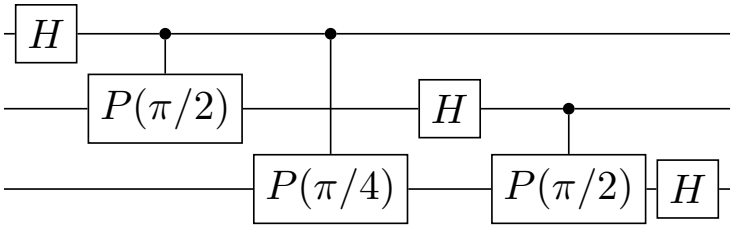


图 4-1 QFT_3 的量子线路图

问题上的优势[75]。通过一次量子测量即可确定一个隐藏的二进制字符串，而经典算法需要多次查询。该算法证明了量子计算在处理信息编码和解码问题上的潜力。BV 算法的量子电路相对简单，核心是一次量子查询。电路首先将所有量子位初始化到叠加态，然后应用一个特殊的 Oracle，该 Oracle 能够对输入的二进制字符串进行编码。通过对量子位的一次测量，就可以直接获得这个隐藏的字符串，展示了量子并行处理的能力。本次实验中，为了实验中的 Oracle 实现方便，隐藏字符串均选择了 $1\cdots1$。图4-2是本次实验中 BV_3 的量子线路图。其余的 BV 算法的量子线路图设计类似。

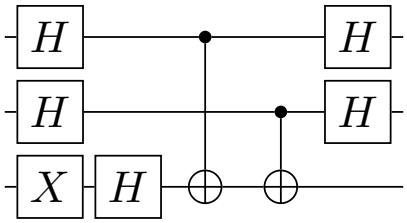


图 4-2 BV_3 的量子线路图

GHZ 态，即 Greenberger-Horne-Zeilinger 态是一种特殊的量子纠缠态，涉及三个或更多的量子位[76]。它是量子信息科学的一个重要组成部分，尤其在量子通信和量子计算中的应用。GHZ 态的生成和操控是量子计算能力的一个重要指标。生成 GHZ 态的量子电路从一个量子位的叠加态开始，该量子位通过一系列的受控非门与其他量子位相连。这个过程在量子位之间产生强烈的纠缠，形成了 GHZ 态。这种状态对于测试量子纠缠和量子通信协议极为重要。图4-3是本次实验中 GHZ_3 的量子线路图。其余的 GHZ 算法的量子线路图设计类似。

QRW，即量子随机游走是在量子态空间中的随机游走，与经典随机游走在概率分布上有本质的不同[77]。QRW 在量子算法设计、量子搜索以及量子密码学中有广泛的应用，是理解量子计算概念和开发新量子算法的一个有力工具。QRW 的量子电路设计依赖于叠加和纠缠的特性，以及量子干涉效应。电路通过交替应用移动操作和硬币投掷操作（用量子逻辑门实现）在量子位上模拟随机游走的过程。这种设计可以在量子计算中模拟复杂的概率过程，为开发新算法提供了一种

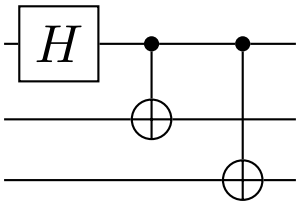


图 4-3 GHZ_3 的量子线路图

工具。本次实验中，选取了在 0 到 $2^n - 1$ 循环链上的量子随机游走。图4-4是本次实验中 QRW_3 的量子线路图。其余的 QRW 算法的量子线路图设计类似。

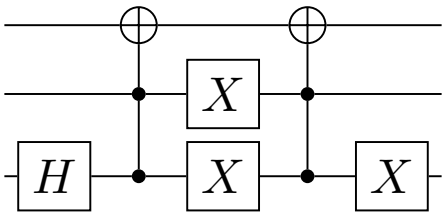


图 4-4 QRW_3 的量子线路图

比较指标与实验内容

在本次实验过程中，时间资源可以用计算过程中的耗时度量。空间资源方面，由于不同编译语言，不同计算平台之间存在差异。所以用计算过程中 TDD 节点个数进行度量空间资源。同时由于最大节点数可以表现实验中内存资源占用最多的时刻。因此实验中通过比较最大节点数讨论空间资源占用。

具体实验方面，本次研究主要是为了验证基于 TDD 的量子模型检测的实用性与优化方案的效率提升。在3.4节中具体讨论了五种进一步的优化方案。分别是调整 TDD 索引顺序，addition 和 contraction 两种通过线路拆分进行优化的方案，以及 TDD 拆分和用子空间近似表示两种对 TDD 优化方法。其中调整 TDD 索引顺序是在实验前进行的，因此这里不再展示。基于这些优化方案，本次实验将首先进行数值实验，对线路拆分的参数选择进行分析。然后进行对比实验，比较没有使用优化方案的 Basic 方法，addition 电路拆分，contraction 电路拆分三种方法的资源占用。最后再进行对比实验，验证两种对 TDD 结构的拆分，是否能减少计算资源的消耗。

最后本章节中的实验是均在 Intel Xeon-Gold-5215 CPU，512GB RAM 的硬件平台进行的。

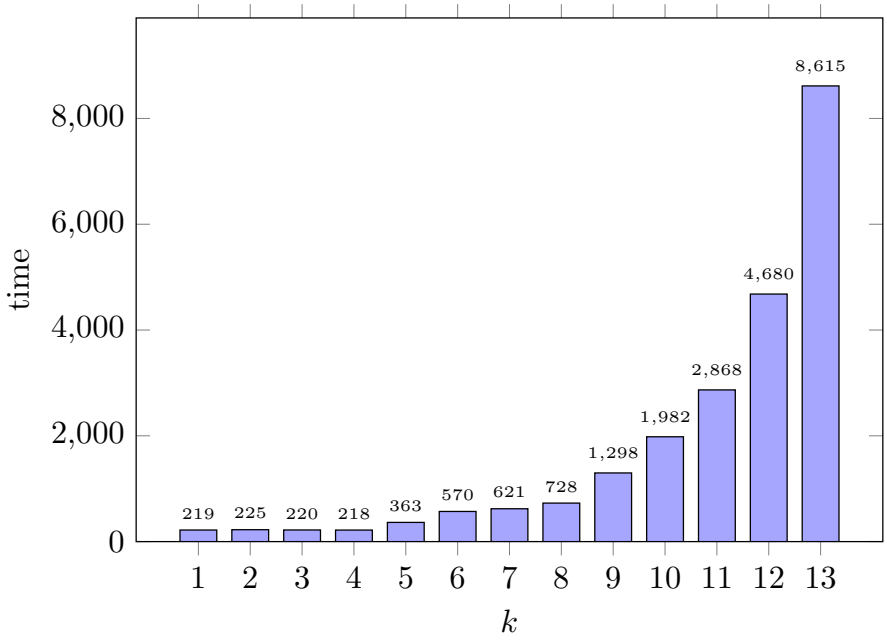


图 4-5 不同参数 k 对 Grover_15 线路的 additon 划分方案的时间影响

4.2 线路划分技术的参数选择

本节中都以 Grover_15 线路为例，研究不同参数线路划分方案性能的影响。首先讨论参数 k 对 addition 线路划分方案性能的影响。使用参数 k 从 1 到 13 的 addition 线路划分方案的一步迁移。表4-1为具体数据。根据表4-1绘制折线图4-5。从折线图4-5中，可以看出，当参数 k 小于五时，总时间不会发生显著变化。但当参数 k 增大时，时间将会指数级增长。这是因为随着 k 的增长，将线路划分成 2^k 部分。这说明，通常不应该将线路划分成太多部分，只有在存在明显中心时才进行划分。

因此最好不将 addition 线路划分方案的参数设置的过大。

表 4-1 对 grover_15 应用不同的 addition 参数的时间对比

k	1	2	3	4	5	6	7	8	9	10	11	12	13
时间	219	225	220	218	363	570	621	728	1298	1982	2868	4680	8615

以 Grover_15 线路为例，继续研究参数 k_1 和 k_2 对 contraction 线路划分方案性能的影响。表4-2展示了对 Grover_15 应用不同的 contraction 参数的时间比较，单位为秒。其中表4-2中蓝色表示用时为用时超过 10 秒的参数，紫色表示用时小于等于 2 秒。深紫色表示用时超过 100 秒，即特别长的参数对。深紫色表示用时小于等于 1.5 秒，表示用时特别少的参数对。

从表4-2中的颜色可以看出，时间比较长的蓝色部分主要集中在图下半部分。这说明只要不把参数 k_1 设置得太大，contraction 线路划分方案方法就会很有效。这意味着对于参数有很宽的选择范围。

同时从表格中还可以看到，当将 contraction 线路划分方案的参数 k_1 设置在

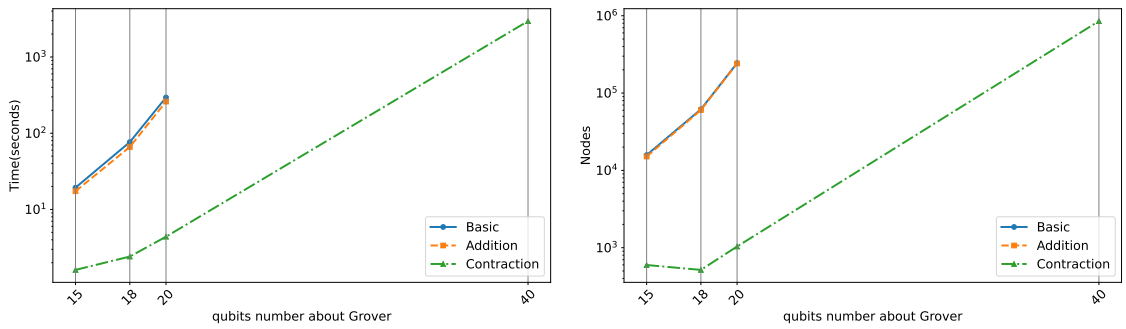
2 到 9 之间，k_2 设置在 2 到 6 之间时，时间基本都是比较短的紫色。这意味着通常最好将 contraction 拆分方案的参数设置为适中的值。

表 4-2 对 grover_15 应用不同的 contration 参数的时间对比

k1 \ k2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2.8	2.2	2.1	2.0	1.9	2.0	2.1	2.0	2.1	2.0	2.0	2.1	2.2	2.1	2.1
2	2.6	2.0	2.0	1.8	2.0	2.0	2.0	2.0	2.1	2.0	2.3	2.0	2.3	2.3	2.4
3	2.2	1.9	1.8	1.6	2.0	1.9	2.1	2.1	2.5	2.3	2.7	2.3	3.1	2.8	3.3
4	2.3	1.8	2.0	1.7	2.0	2.1	2.2	2.1	2.6	2.3	2.8	2.7	3.3	3.0	3.3
5	2.2	1.7	1.9	1.6	1.9	2.0	2.3	1.9	2.5	2.3	2.8	2.7	3.4	3.0	3.6
6	2.1	1.5	1.8	1.7	2.2	1.9	2.5	2.2	2.9	2.8	3.1	2.9	3.7	3.7	4.2
7	2.1	1.5	1.9	1.6	2.2	1.9	2.5	2.2	2.8	3.0	3.6	3.3	4.2	5.7	5.0
8	2.0	1.7	1.8	1.7	2.1	2.0	2.4	2.2	2.8	2.8	3.7	3.4	4.3	4.8	5.2
9	2.1	1.5	2.0	1.4	2.2	2.0	2.5	2.0	3.3	2.9	3.7	3.5	4.9	4.7	5.8
10	2.3	1.9	2.3	1.6	2.6	2.7	3.1	2.2	4.0	3.6	4.6	3.9	5.6	5.2	7.5
11	3.2	3.2	3.5	3.1	4.7	4.2	5.6	4.2	6.8	7.2	7.6	6.3	9.0	8.1	11
12	5.6	6.0	7.2	6.0	8.3	9.0	8.9	7.8	11	11	12	11	12	15	16
13	11	12	14	12	15	18	18	15	18	20	18	32	32	30	25
14	20	21	24	32	31	44	77	50	86	109	68	133	70	119	142
15	28	30	31	53	69	111	85	81	102	153	114	130	166	162	235

4.3　线路划分技术

为了方便比较，本节将首先利用数据绘制图像，然后分析具体趋势。其中蓝色圆点折线始终代表没有线路拆分技术的 Basic 方法，橙色方块折线始终代表 addition 拆分技术方法，绿色三角块折线始终代表 contraction 线路拆分技术方法。图中时间单位均为秒。参数选择上，根据4.2节的结论，将 addition 优化方法中的参数为 $k = 1$，addition 优化方法中的参数为 $k1 = k2 = 4$。



(a) 对 Grover 算法应用不同线路拆分技术的时间对比

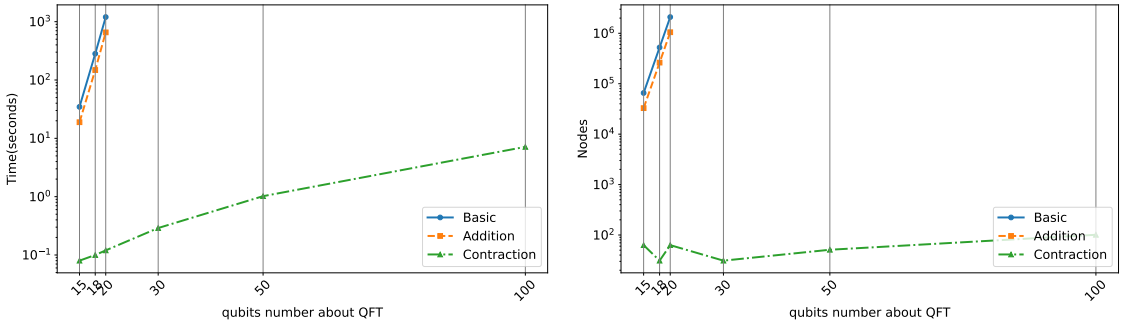
(b) 对 Grover 算法应用不同线路拆分技术的最大节点对比

图 4-6 对 Grover 算法运行一步迁移算法时不同线路拆分技术的资源对比

图4-6表示了对 Grover 搜索算法运行不同一步迁移算法的资源对比。可以看到不论是基础算法 (basic) 和 addition 方案在超过 20 量子比特时都无法在规定时间内完成。而 contraction 方案，在 40 个比特以后才不能在超时前完成计算。

图4-7表示了对 quantum Fourier transform（QFT）算法运行不同一步迁移算法的资源对比。可以看到不论是基础算法 (basic) 和 addition 方案在超过 20 量子

比特后无法进行计算。而 contraction 方法在 100 比特前始终没有超时，特别是最大节点数相比其他两种方法显著地小。

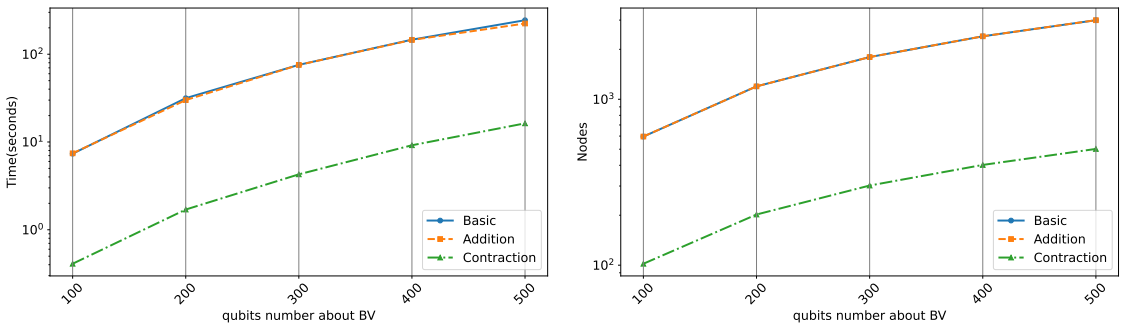


(a) 对 QFT 算法应用不同线路拆分技术的时间对比

(b) 对 QFT 算法应用不同线路拆分技术的最大节点对比

图 4-7 对 QFT 算法运行一步迁移算法时不同线路拆分技术的资源对比

图4-8表示了对 Bernstein–Vazirani（BV）算法运行不同一步迁移算法的资源对比。尽管三种方法都能在 5 分钟内计算 500 量子比特的 Bernstein-Vazirani 算法的图像。但在该方法中绿色的 contraction 方法依然比其他两种方法有优势。



(a) 对 BV 算法应用不同线路拆分技术的时间对比

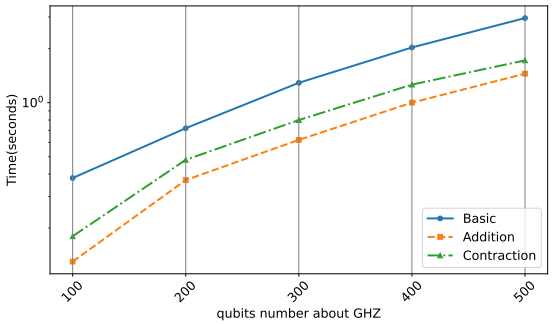
(b) 对 BV 算法应用不同线路拆分技术的最大节点对比

图 4-8 对 BV 算法运行一步迁移算法时不同线路拆分技术的资源对比

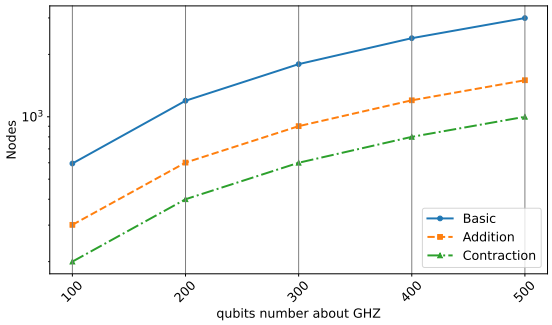
图4-9表示了对 Greenberger–Horne–Zeilinger （GHZ）状态制备线路运行不同一步迁移算法的资源对比。三种方法也都能在 4 秒钟内完成 500 量子比特的 GHZ 算法的图像计算。但是时间方面 addition 方法比较好，但和 contraction 差距之间不明显。最大节点数方面，contraction 则同样比 addition 好，但差距不明显。

图4-10表示了对在 2^n 环上的 quantum random walk （QRW）算法运行不同一步迁移算法的资源对比。可以看到基础算法 (basic) 和 addition 方案再次在超过 20 量子比特后不能计算。而 contraction 方法在 100 比特前始终没有超时，空间资源方面的最大节点数在 20 比特之后趋于稳定。

总的来说，三种不同的线路方案中，addition 线路划分方案法优于基础算法，而 contraction 线路划分方案则是最佳选择，其性能大大超过其他两种方法。比

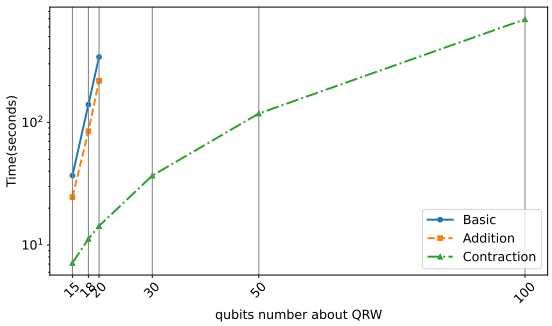


(a) 对 GHZ 算法应用不同线路拆分技术的时间对比

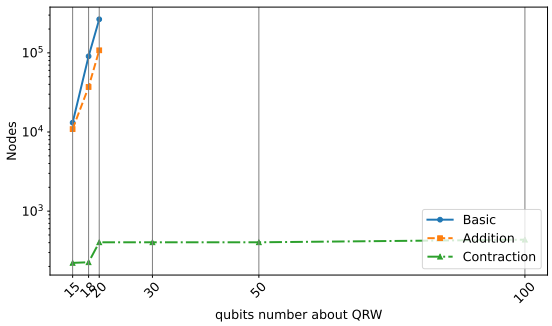


(b) 对 GHZ 算法应用不同线路拆分技术的最大节点对比

图 4-9 对 GHZ 算法运行一步迁移算法时不同线路拆分技术的资源对比



(a) 对 QRW 算法应用不同线路拆分技术的时间对比



(b) 对 QRW 算法应用不同线路拆分技术的最大节点对比

图 4-10 对 QRW 算法运行一步迁移算法时不同线路拆分技术的资源对比

如，对于 QRW_20，它仅需要 14 秒，而 addition 线路划分方案法和 contraction 线路划分方案分别只需要 218 秒和 341 秒。此外，它能够处理远超 20 量子比特的 Grover、QFT 和 QRW 线路。更重要的是，对于 QFT、BV、GHZ 和 QRW，contraction 线路划分方案的 TDD 最大节点数呈线性增长。

对于 contraction 线路划分方案，可以看到，无论在时间还是空间消耗方面，它都比基础算法具有指数级的效率提升。可能的原因是这种算法避免了对整个功能的计算。

而尽管 contraction 线路划分方案总是优于 addition 线路划分方案，但保留 addition 线路划分方案是必要的。首先，注意到 addition 线路划分方案对不同线路的性能表现各不相同，这意味着 addition 线路划分方案具有其独特效果。比如 GHZ 线路中，addition 方案的时间资源比较小。特别是对于中心化线路，即在无向图中某些索引的度远大于其他的线路。因此有些线路天然需要使用 addition 线路划分方案进行分割。另一方面，addition 线路划分方案可以与 contraction 线路划分方案结合使用。同时对于 addition 线路划分方案，由于线路被划分成许多更简单的部分，因此可以使用更小的空间消耗计算每个部分的图像。也就是当空间资源有限时，可以增加参数 k，将线路切割成更多部分。然后可以逐部分一步迁移，再将它们相加。还可以使用二级缓存来帮助完成这一方案的最终计算。

表4-3给出了三种方法的资源消耗的详细数据。time 表示计算 TDD 收缩的总时间，单位为秒，最大节点数表示计算过程中 TDD 的节点最大个数。basic 表示没有使用优化技术，addition 表示使用3.4节中的 addition 优化技术，contraction 表示使用3.4节中的 contraction 优化技术。“-”表示超过一小时的运行上限。

表 4-3 对不同测试实验应用一步迁移算法

Benchmark	basic		addition 优化方案		contraction 优化方案	
	时间	最大节点数	时间	最大节点数	时间	最大节点数
Grover_15	19.33	15785	17.35	15099	1.61	597
Grover_18	76.47	61694	66.02	60332	2.41	516
Grover_20	294.65	243946	259.87	241240	4.39	1036
Grover_40	-		-		2953.57	851973
QFT_15	34.64	65536	18.88	32770	0.08	63
QFT_18	282.12	524288	148.13	262146	0.10	31
QFT_20	1199.21	2097152	655.19	1048578	0.12	63
QFT_30	-		-		0.29	31
QFT_50	-		-		1.02	51
QFT_100	-		-		7.14	101
BV_100	7.36	596	7.43	596	0.41	102
BV_200	31.57	1196	30.03	1196	1.70	202
BV_300	75.66	1796	75.56	1796	4.28	302
BV_400	146.47	2396	145.40	2396	9.18	402
BV_500	244.15	2996	223.90	2996	16.31	502
GHZ_100	0.38	595	0.13	301	0.18	200
GHZ_200	0.72	1195	0.37	601	0.48	400
GHZ_300	1.29	1795	0.62	901	0.80	600
GHZ_400	2.03	2395	1.00	1201	1.26	800
GHZ_500	2.96	2995	1.45	1501	1.72	1000
QRW_15	36.86	13122	24.59	10882	7.16	222
QRW_18	139.76	90538	84.69	37064	11.23	226
QRW_20	341.05	265614	218.29	107714	14.31	404
QRW_30	-		-		36.82	404
QRW_50	-		-		118.08	404
QRW_100	-		-		692.08	436

4.4　对 TDD 结构的优化

最后为了验证基于 TDD 优化方法，即对 TDD 的分割和近似方法在降低系统的一步迁移过程中内存占用以及缓解内存溢出问题上的效果。实验中，选取了 Grover_40 和 QFT_100 作为测试案例，分别采用了 TDD 的分割和子空间近似的方法进行系统的一步迁移，并记录了 TDD 的处理时间及其最大节点数量。实验成果汇编于表4-4。表中，变量 k 用于指示采用的分割次数。当 $k = 1$ 时，将原 TDD 一分为二，形成两个较为均衡的部分，从而显著减少了节点数量。$k = 2$ 时，进一步将一个分支细分为两个小部分。而到了 $k = 3$，在另一个分支上继续分割，节点数因此再次大幅度降低。

对于 Grover_40 线路，初始使用子空间 $S = span\{|0\cdots0\rangle\}$ 以简化过程。从表4-4可见，在不进行一步迁移算法优化的情况下，最大的 TDD 节点数达到了 589,865。通过在达到最大 TDD 之前实施基于 TDD 的分割，可以将最大节点数减至 393,423。若对两个分支均分割 TDD，则最大节点数可进一步降至 245,814，不足原数的一半。此外，这一过程几乎不会增加时间消耗。

但有时，计算末尾才会出现最大的 TDD。这意味着，即便在整个计算过程中已将 TDD 分割为多个小片段，最大的 TDD 仍难以避免，除非采取近似方法。这里以初始子空间为 $S = span\{|+\cdots+--\cdots-\rangle\}$（最后 20 个量子比特全为 $|-\rangle$）的 QFT_100 为例，计算结束时出现了含有 524,369 节点的最大 TDD。因此，虽

表 4-4 TDD 拆分与近似的优化方案

线路	优化方法		$k=0$	$k=1$	$k=2$	$k=3$
Grover_40	TDD 分割	时间	1,510.42	1,519.24	1,459.02	1,495.20
		最大节点个数	589,865	393,423	393,239	245,814
QFT_100	子空间近似	时间	121.28	118.78	116.69	128.31
		最大节点个数	524,369	262,226	262,226	131,155

然会在计算过程中通过 TDD 分割获得多个小 TDD，但在最后不会将它们合并。相反，用这些 TDD 张成的子空间来近似原始 TDD。进而，通过将原来的一维子空间近似为二维子空间，可以将最大 TDD 的节点数降至 262,226；将原空间近似为四维时，节点数可降至 131,155。在整个近似过程中，时间消耗同样不会显著增加。值得注意的是，实际验证时无需将这些小 TDD 同时存储于内存中，而可以逐一进行计算与验证。

4.5 本章小结

本章首先设计了实验基准测试集，选取了比较指标，然后对几种优化一步迁移算法的方法，进行了实验对比。对比的化方法包括线路拆分 (addition 和 contraction) 和对 TDD 结构的优化 (拆分和子空间近似)。

实验结果表明，在线路拆分方法中，参数的选择很关键。对 addition 方法，参数 k 不宜过大，通常取 2-4 比较合适；对 contraction 方法，参数 k1、k2 适中值范围较宽，通常取 2-9 和 2-6 较为合适。

而 contraction 线路拆分方法在降低时间和空间资源消耗方面效果最佳，能够显著提高算法效率，尤其对线路层数较深的量子算法很有优势。addition 线路拆分方法次之，但是对某些特殊线路也有一定优化效果。

另一方面，针对 TDD 结构进行拆分和子空间近似也能有效降低内存占用，避免内存溢出，且时间开销较小。尤其是当出现大的最终 TDD 时，子空间近似方法很有帮助。

总的来说，上述优化方法能够有效降低一步迁移算法过程中的时间和空间资源消耗，提高量子模型检测的可扩展性，为大规模量子系统验证提供实用的工具。

第 5 章　总结与展望

5.1　论文总结

近几年的量子计算机的发展方向是规模化拓展以及尝试容错计算。这些软硬件的发展，对量子系统的可靠性提出了更高的要求。而量子模型检测是验证量子系统可靠性的一种自动化方法。其面临的主要困难是随着量子系统中量子比特增加，资源需求指数增长。

本次研究的目的是设计更高效的量子模型检测工具。而在 TDD 的数据结构基础上，本次研究设计了有效识别给定子空间基的算法，并针对子空间及量子线路，提出了多种优化策略。基于此，本次研究设计了对应的软件工具。

最后以量子迁移系统中一步迁移算法为例，本次研究设计了数值实验。实验结果验证了本次研究中软件工具的可行性，以及优化方案的效率提升。特别地，数值实验还验证了采用基于 contraction 的线路分割算法能大幅提升量子迁移系统中一步迁移算法的效率。

5.2　工作展望

在未来，伴随量子计算的不断发展，关于验证量子系统，比如量子线路和量子系统的工作会越来越重要。本文研究过程中使用 TDD 表示对量子模型检测进行一定程度的探索，但仍然有以下工作需要完善。

(1) 在本次工作中，主要涉及量子迁移系统中的可达空间。在后续工作中，计划利用本文的方法关注那些只可以通过时态逻辑表述的属性。

(2) 在本次工作中，主要涉及了可以用量子线路描述的描述系统。在后续工作中，还需要更深入探讨更为复杂系统的量子系统，特别是那些无法仅以量子线路形式表达的系统。

(3) 在本次工作中，展示了对 TDD 优化而提高效率的实例。后续工作中应进一步提高 TDD 的表示能力，如利用一些特殊的张量等价关系。同时本次研究中主要使用的是基于 python 开发的 TDD。在后续工作中，应当使用更高效率的程序语言，如 C 语言重新设计 TDD，从而提高 TDD 的运算能力。

参考文献

[1] Bocharov A, Roetteler M, Svore K M. Efficient synthesis of universal repeat-until-success quantum circuits [J/OL]. Physical Review Letters, 2015, 114(8). http://dx.doi.org/10.1103/physrevlett.114.080502.

[2] Niemann P, Wille R, Drechsler R. Advanced exact synthesis of clifford+ t circuits [J]. Quantum Information Processing, 2020, 19: 1-23.

[3] Emerson E A, Clarke E M. Characterizing correctness properties of parallel programs using fixpoints [M/OL]. Springer Berlin Heidelberg, 1980: 169–181. http://dx.doi.org/10.1007/3-540-10003-2_69.

[4] Clarke E M, Emerson E A. Design and synthesis of synchronization skeletons using branching time temporal logic [C/OL]//Workshop on logic of programs. Springer, 1981: 52-71. http://dx.doi.org/10.1007/bfb0025774.

[5] Clarke E M, Emerson E A, Sistla A P. Automatic verification of finite-state concurrent systems using temporal logic specifications [J/OL]. ACM Transactions on Programming Languages and Systems, 1986, 8(2): 244–263. http://dx.doi.org/10.1145/5397.5399.

[6] Grobelna I, Grobelny M, Adamski M. Model checking of uml activity diagrams using a rule-based logical model [M/OL]. Springer International Publishing, 2015: 153–163. http://dx.doi.org/10.1007/978-3-319-26725-8_12.

[7] Planck M. On the law of distribution of energy in the normal spectrum [J]. Annalen der physik, 1901, 4(553): 1.

[8] Bhatta V S. Plurality of wave–particle duality [J]. Current Science, 2020, 118(9): 1365-1374.

[9] Hodges A. Alan turing: The enigma: The book that inspired the film" the imitation game" [M]. Princeton University Press, 2014.

[10] Mårtensson-Pendrill A M. The manhattan project—a part of physics history [J]. Physics education, 2006, 41(6): 493.

[11] Benioff P. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines [J]. Journal of statistical physics, 1980, 22: 563-591.

[12] Feynman R P. Simulating physics with computers [M/OL]. CRC Press, 2002: 133–153. http://dx.doi.org/10.1201/9780429500459-11.

[13] Manin I I. Vychislimoe i nevychislimoe [J]. 1980.

[14] Bennett C H, Brassard G. Quantum cryptography: Public key distribution and coin tossing [J]. Theoretical computer science, 2014, 560: 7-11.

[15] Shor P. Algorithms for quantum computation: discrete logarithms and factoring [C/OL]// SFCS-94: Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press, 1994. http://dx.doi.org/10.1109/sfcs.1994.365700.

[16] Grover L K. A fast quantum mechanical algorithm for database search [C/OL]//STOC ' 96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC ' 96. ACM Press, 1996. http://dx.doi.org/10.1145/237814.237866.

[17] Lloyd S. Universal quantum simulators [J]. Science, 1996, 273(5278): 1073-1078.

[18] Cao Y, Romero J, Olson J P, et al. Quantum chemistry in the age of quantum computing [J]. Chemical reviews, 2019, 119(19): 10856-10915.

[19] Chuang I L, Gershenfeld N, Kubinec M. Experimental implementation of fast quantum searching [J]. Physical review letters, 1998, 80(15): 3408.

[20] Arute F, Arya K, Babbush R, et al. Quantum supremacy using a programmable superconducting processor [J]. Nature, 2019, 574(7779): 505-510.

[21] Pednault E, Gunnels J, Maslov D, et al. On quantum supremacy [J]. IBM Research Blog, 2019, 21.

[22] Gambetta J. The hardware and software for the era of quantum utility is here [EB/OL]. 2023. https://www.ibm.com/quantum/blog/quantum-roadmap-2033.

[23] Chen J S, Nielsen E, Ebert M, et al. Benchmarking a trapped-ion quantum computer with 29 algorithmic qubits [Z]. 2023.

[24] Atom Computing. Quantum startup atom computing first to exceed 1,000 qubits [EB/OL]. 2023. https://atom-computing.com/quantum-startup-atom-computing-first-to-exceed-1000-qubits/.

[25] Deng Y H, Gu Y C, Liu H L, et al. Gaussian boson sampling with pseudo-photon-number-resolving detectors and quantum computational advantage [J]. Physical review letters, 2023, 131(15): 150601.

[26] Beijing Academy of Quantum Information Sciences and Tsinghua University. China unveils new quantum computing cloud platform [Z]. 2023.

[27] Kwiatkowska M, Norman G, Parker D. Probabilistic symbolic model checking with prism: A hybrid approach [J]. International journal on software tools for technology transfer, 6: 128-142.

[28] Mateus P, Sernadas A. Weakly complete axiomatization of exogenous quantum propositional logic [J]. Information and Computation, 204(5): 771-794.

[29] Baltazar P, Chadha R, Mateus P. Quantum computation tree logic—model checking and complete calculus [J]. International Journal of Quantum Information, 2008, 6(02): 219-236.

[30] Baltazar P, Chadha R, Mateus P, et al. Towards model-checking quantum security protocols [C]//2007 First International Conference on Quantum, Nano, and Micro Technologies (ICQNM'07). IEEE, 2007: 14-14.

[31] Mateus P, Ramos J, Sernadas A, et al. Temporal logics for reasoning about quantum systems [J]. Semantic techniques in quantum computation, 2009: 389-413.

[32] Gay S J, Nagarajan R. Communicating quantum processes [C]//Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming languages. 2005: 145-157.

[33] Davidson T A. Formal verification techniques using quantum process calculus [D]. University of Warwick, 2012.

[34] Davidson T A, Gay S J, Mlnarik H, et al. Model checking for communicating quantum processes. [J]. Int. J. Unconv. Comput., 2012, 8(1): 73-98.

[35] Gay S J, Nagarajan R, Papanikolaou N. Qmc: A model checker for quantum systems [M/OL]. Springer Berlin Heidelberg, 2008: 543–547. http://dx.doi.org/10.1007/978-3-540-70545-1_51.

[36] Gay S J, Nagarajan R, Papanikolaou N. Specification and verification of quantum protocols [J]. Semantic Techniques in Quantum Computation, 2010: 414-472.

[37] Gottesman D. Stabilizer codes and quantum error correction [M]. California Institute of Technology.

[38] Ardeshir-Larijani E, Gay S J, Nagarajan R. Equivalence checking of quantum protocols [C]// Tools and Algorithms for the Construction and Analysis of Systems: 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice

of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings 19. Springer, 2013: 478-492.

[39] Ardeshir-Larijani E, Gay S J, Nagarajan R. Verification of concurrent quantum protocols by equivalence checking [C]//Tools and Algorithms for the Construction and Analysis of Systems: 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings 20. Springer, 2014: 500-514.

[40] Kondacs A, Watrous J. On the power of quantum finite state automata [C]//Proceedings 38th annual symposium on foundations of computer science. IEEE, 1997: 66-75.

[41] Birkhoff G, Von Neumann J. The logic of quantum mechanics [J]. Selected Papers on Algebra and Topology, 1987, 37(4): 67.

[42] Ying M, Li Y, Yu N, et al. Model-checking linear-time properties of quantum systems [J]. ACM Transactions on Computational Logic (TOCL), 2014, 15(3): 1-31.

[43] Li Y, Ying M. (un) decidable problems about reachability of quantum systems [C]//CONCUR 2014–Concurrency Theory: 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings 25. Springer, 2014: 482-496.

[44] Ying M, Yu N, Feng Y, et al. Verification of quantum programs [J]. Science of Computer Programming, 78(9): 1679-1700.

[45] Ying S, Feng Y, Yu N, et al. Reachability probabilities of quantum markov chains [C]// CONCUR 2013–Concurrency Theory: 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings 24. Springer, 2013: 334-348.

[46] Ying S, Ying M. Reachability analysis of quantum markov decision processes [J]. Information and Computation, 263: 31-51.

[47] Feng Y, Yu N, Ying M. Model checking quantum markov chains [J]. Journal of Computer and System Sciences, 2013, 79(7): 1181-1198.

[48] Feng Y, Yu N, Ying M. Reachability analysis of recursive quantum markov chains [C]// Mathematical Foundations of Computer Science 2013: 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings 38. Springer, 2013: 385-396.

[49] Feng Y, Hahn E M, Turrini A, et al. Model checking omega-regular properties for quantum markov chains [C]//28th International Conference on Concurrency Theory (CONCUR 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[50] Hahn E M, Li Y, Schewe S, et al. iscas m c: a web-based probabilistic model checker [C]// FM 2014: Formal Methods: 19th International Symposium, Singapore, May 12-16, 2014. Proceedings 19. Springer, 2014: 312-317.

[51] Feng Y, Hahn E M, Turrini A, et al. Qpmc: A model checker for quantum programs and protocols [C]//FM 2015: Formal Methods: 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings 20. Springer, 2015: 265-272.

[52] Viamontes G F, Markov I L, Hayes J P. Improving gate-level simulation of quantum circuits [J/OL]. Quantum Information Processing, 2003, 2(5): 347–380. http://dx.doi.org/10.1023/b:qinp.0000022725.70000.4a.

[53] Seiter J, Soeken M, Wille R, et al. Property checking of quantum circuits using quantum multiple-valued decision diagrams [M/OL]. Springer Berlin Heidelberg, 2013: 183–196. http://dx.doi.org/10.1007/978-3-642-36315-3_15.

[54] Hong X, Zhou X, Li S, et al. A tensor network based decision diagram for representation of

quantum circuits [J/OL]. ACM Transactions on Design Automation of Electronic Systems, 2022, 27(6): 1–30. http://dx.doi.org/10.1145/3514355.

[55] Kerckhoff J, Nurdin H I, Pavlichin D S, et al. Designing quantum memories with embedded control: Photonic circuits for autonomous quantum error correction [J/OL]. Physical Review Letters, 2010, 105(4). http://dx.doi.org/10.1103/physrevlett.105.040502.

[56] Gough J, James M R. Quantum feedback networks: Hamiltonian formulation [J/OL]. Communications in Mathematical Physics, 2008, 287(3): 1109–1132. http://dx.doi.org/10.1007/s00220-008-0698-8.

[57] Ying M. Model checking for verification of quantum circuits [C]//Formal Methods: 24th International Symposium, FM 2021, Virtual Event, November 20–26, 2021, Proceedings 24. Springer, 2021: 23-39.

[58] Chaki S, Gurfinkel A. Bdd-based symbolic model checking [M/OL]. Springer International Publishing, 2018: 219–245. http://dx.doi.org/10.1007/978-3-319-10575-8_8.

[59] Nielsen M A, Chuang I L. Quantum computation and quantum information [M]. Cambridge university press, 2010.

[60] Greub W H. Linear algebra: volume 23 [M]. Springer Science & Business Media, 2012.

[61] Baier C, Katoen J P. Principles of model checking [M]. MIT press, 2008.

[62] Ying M, Feng Y. Model checking quantum systems: Principles and algorithms [M]. Cambridge University Press, 2021.

[63] Goranko V. Elements in philosophy and logic: Temporal logics [M/OL]. Cambridge University Press, 2023. DOI: 10.1017/9781009170093.

[64] Biamonte J. Lectures on quantum tensor networks [J]. arXiv preprint arXiv:1912.10049, 2019.

[65] Pednault E, Gunnels J A, Nannicini G, et al. Breaking the 49-qubit barrier in the simulation of quantum circuits [J]. arXiv preprint arXiv:1710.05867, 2017, 15.

[66] Miller D, Thornton M. Qmdd: A decision diagram structure for reversible and quantum circuits [C/OL]//36th International Symposium on Multiple-Valued Logic (ISMVL'06). 2006: 30-30. DOI: 10.1109/ISMVL.2006.35.

[67] Burgholzer L, Raymond R, Wille R. Verifying results of the ibm qiskit quantum circuit compilation flow [C]//2020 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, 2020: 356-365.

[68] Burgholzer L, Wille R. Advanced equivalence checking for quantum circuits [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(9): 1810-1824.

[69] Zhang Q, Saligane M, Kim H S, et al. Quantum circuit simulation with fast tensor decision diagram [J]. arXiv preprint arXiv:2401.11362, 2024.

[70] Chen J, Zhang F, Huang C, et al. Classical simulation of intermediate-size quantum circuits [J]. arXiv preprint arXiv:1805.01450, 2018.

[71] Narayan A, Jain J, Fujita M, et al. Partitioned robdds-a compact, canonical and efficiently manipulable representation for boolean functions [C]//Proceedings of International Conference on Computer Aided Design. IEEE, 1996: 547-554.

[72] Cho H, Hachtel G D, Macii E, et al. Algorithms for approximate FSM traversal based on state space decomposition [J/OL]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1996. DOI: 10.1109/43.552080.

[73] Lin H, Li P. Parallel hierarchical reachability analysis for analog verification [C]//Proceedings of the 51st Annual Design Automation Conference. 2014: 1-6.

[74] Wang C, Hachtel G D, Somenzi F. The compositional far side of image computation [C]// ICCAD '03: Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design. USA: IEEE Computer Society, 2003: 334.

[75] Nagata K, Resconi G, Nakamura T, et al. A generalization of the bernstein-vazirani algorithm [J]. MOJ Ecol. Environ. Sci, 2017, 2(1): 00010.

[76] Greenberger D M, Horne M A, Zeilinger A. Going beyond bell' s theorem [M]//Bell' s theorem, quantum theory and conceptions of the universe. Springer, 1989: 69-72.

[77] Kempe J. Quantum random walks: an introductory overview [J]. Contemporary Physics, 2003, 44(4): 307-327.

附录 一　部分优化方案的 python 实现

第3.4节介绍了针对基于 TDD 的量子模型检测方法的一些优化策略。第4章节证实了量子拆分技术能够有效降低计算资源消耗，是一种非常实用的优化方案。本附录将以 Python 为例，介绍这些优化方法的具体实现。

在代码实现中，变量 Ts 一般表示一个量子迁移系统，其中包含以下变量：

- Ts.initial_states 是一个列表，包含所有量子迁移系统的初态。
- Ts.operators 是一个列表，按顺序包含所有量子线路的操作。

基于这样的量子迁移系统设计，可以设计代码段 2。此代码段核心使用张量收缩技术，以计算量子迁移系统的一次状态转移。具体来说，就是基于系统的初始状态来预测其下一状态空间。这里，simulation 函数负责执行量子操作的 TDD 与量子态的收缩操作，而 join 函数则用于将得到的新量子状态并入当前状态子空间中。因此，代码逻辑首先从系统的初始态出发，构建一个初始的状态子空间。接着，它遍历所有量子操作，对每一个初始态执行操作，通过 simulation 函数计算操作后的结果，并利用 join 函数将这些新状态并入到状态子空间中，以完成系统下一步状态空间的构建。

```
def image(Ts):
    sp= generate_initial_space(Ts.initial_states)
    for op in Ts.operators:
        for phi in Ts.initial_states:
            res=simulation(Ts.operators[op],phi)
            sp=join(sp,res)
    return sp
```

代码段 2 利用 TDD 收缩直接计算一步迁移

类似地，可以设计代码段 3 中的 addition 优化方案。其中 tn 为表示量子线路的量子张量网络。不同在于其中的 Ts 的 operators 将会在运行中生成。其他运行逻辑与代码段 2 类似，码段 2 的函数首先生成系统的初始状态子空间。然后，对系统中的每个量子操作，通过函数 addition_partition 应用分区优化 tn 的索引，从而得到 Ts 的量子操作。这一步调整了量子操作，以便提高执行效率。在应用了优化后的量子操作进行状态转移的计算过程中，函数对每个初始态进行遍历。对于每个量子操作，它初始化一个表示恒等操作的 TDD，作为累加的基础。接着，通过遍历量子操作的所有部分，使用 simulation 函数计算每部分的效果，并利用 add 函数将这些 TDD 累加到 res 中。由于应用索引划分了线路，相当于对张量网络进行了切片。因此这里需要使用 TDD 的加法，即 add 函数。这样最后的 res 就是对应量子初态在执行所有量子操作后的量子状态。最后通过 join 函数将该量子状态的结果并入到状态子空间中。其中的函数 addition_partition 如代码段 4 所示。

```
def image_add_par(tn,Ts,k=0):
    sp= generate_initial_space(Ts.initial_states)

    Ts.operators[op]=addition_partition(tn,k)

    for op in Ts.operators:
        for phi in Ts.initial_states:
            res=get_identity_tdd()
            for p in Ts.operators[op]:
                temp=simulation(p,phi)
                res=add(res,temp)
            sp=join(sp,res)
    return sp
```

代码段 3 对量子线路应用 addition 优化方案计算一步迁移

```
def addition_partition(tn,k=0):
    """partition a tensor network tn according to k nodes with biggest
    degree"""

    lin_graph.add_nodes_from(tn.index_set)

    deg=[lin_graph.degree(key) for key in lin_graph.nodes]

    big_degres=[]
    node_list=list(lin_graph.nodes)

    t = k
    while t:
        biggest_degree=max(deg)
        if biggest_degree<=0:
            break
        idx=deg.index(biggest_degree)
        big_degres.append(node_list[idx])
        deg[idx]=0
        t-=1

    dd_list=[]
    for t in range(2**k):
        b=list(bin(t)[2:])
        b.reverse()
        temp_tn=copy.copy(tn)
        temp_tn.tensors=[]
        for tensor in tn.tensors:
            temp_ts=copy.copy(tensor)
            for idx in tensor.index_set:
                if idx.key in big_degres:
                    if big_degres.index(idx.key)< len(b) and b[big_degres.
    index(idx.key)]=='1':
                        temp_ts.dd=cont(temp_ts.tdd(),Tensor(ket1,[idx]).tdd
    ())
```

```
                    else:
                        temp_ts.dd=cont(temp_ts.tdd(),Tensor(ket0,[idx]).tdd
    ())
            temp_tn.tensors.append(temp_ts)
        dd_list.append(temp_tn.cont())

    return dd_list
```

代码段 4 addition 方案中的电路划分函数

同样也可以设计对量子线路应用 contraction 的优化方案，具体如代码段 5。该函数利用两个参数 k1 和 k2 来调节收缩分区的策略，以达到更优的计算性能。函数的输入的 tn 和 Ts 的情况和 addition 方案相同。tn 也是表示量子线路的量子张量网络。Ts 的 operators 同样会在运行中生成。初始步骤与之前类似，函数从生成系统初始状态的子空间开始。不同在于对于量子系统中定义的每个量子操作，根据参数 k1 和 k2，使用 contraction_partition 函数对其进行分区。在这一步中 Ts 中的量子操作将会按块划分，因此之后的 Ts 中的 operators 是一个列表，包含所有在该块的量子操作。这一步骤的目的是重新组织量子操作的结构，使得后续的计算过程更加高效。在经过优化的量子操作上执行状态转移计算时，该函数遍历所有初始状态。对于每一个量子操作，它从当前的量子状态开始，逐步应用经过收缩优化的量子操作的各个部分。这里的 simulation 函数用于计算经过每一步操作后的结果。每次操作后得到的新状态通过 join 函数并入到状态子空间中。其中的函数 contraction_partition 如代码段 6 所示。

```
def image_cont_par(tn,Ts,k1=0,k2=0):
    sp= generate_initial_space(Ts.initial_states)

    Ts.operators=contraction_partition(tn,k1,k2)

    for op in Ts.operators:
        for phi in Ts.initial_states:
            res=phi
            for p in Ts.operators[op]:
                res=simulation(p,res)
            sp=join(sp,res)
    return sp
```

代码段 5 对量子线路应用 contraction 优化方案计算一步迁移

```
def contraction_partition(tn,k1=0,k2=0):
    dd_list=[]

    cir_par=circuit_partition(tn,k1,k2)
    for ver in cir_par:
        for hor in ver:
            tdd,node = TensorNetwork(hor).cont()
            dd_list.append(tdd)
```

```
    return dd_list

def circuit_partition(tn,k1=0,k2=0):
    """The first partition scheme;
    cx_max is the number of CNOTs allowed to be cut
    """

    num_qubit=tn.qubits_num

    if k1==0:
        blocks_num=2
        k1=np.ceil(num_qubit/2)
    else:
        blocks_num=int(np.ceil(num_qubit/k1))
    res=[[[] for _ in range(blocks_num)]]

    if k2==0:
        cx_max=num_qubit//2
    else:
        cx_max=k2

    cx_num=0
    level=0

    for tensor in tn.tensors:
        qubit_list = tensor.qubits
        mi=int(min(qubit_list)//k1)
        ma=int(max(qubit_list)//k1)
        if mi==ma:
            res[level][mi].append(tensor)
        else:
            cx_num+=1
            if cx_num<=cx_max:
                res[level][int(qubit_list[-1]//k1)].append(tensor)
            else:
                level+=1
                res.append([])
                for k in range(blocks_num):
                    res[level].append([])
                res[level][int(qubit_list[-1]//k1)].append(tensor)
                cx_num=1

    return res
```

代码段 6 contraction 方案中的电路划分函数

作者简历及攻读学位期间发表的学术论文与其他相关学术成果

作者简历:

2017 年 9 月——2021 年 6 月，在西安电子科技大学大学计算机科学与技术院获得学士学位。

2021 年 9 月——2024 年 6 月，在中国科学院软件研究所攻读硕士学位。