UNIVERSITÉ DE LORRAINE | BIBLIOTHÈQUES UNIVERSITAIRES

AVERTISSEMENT
DROIT D'AUTEUR

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document. Toute contrefaçon, plagiat ou reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

UNIVERSITÉ DE LORRAINE
34 COURS LÉOPOLD — CS 25233
54052 NANCY CEDEX
TÉL : 03 54 50 54 00
CONTACT@UNIV-LORRAINE.FR
WWW.UNIV-LORRAINE.FR

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Combining classical and quantum control: A logical, syntactic and semantic perspective

THÈSE

présentée et soutenue publiquement le 17 décembre 2025

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Kinnari V Dave

Composition du jury

Présidents :	Véronique Cortier Directrice de Recherche, CNRS.
Rapporteurs :	Chris Heunen Professor, University of Edinburgh.
	Peter Selinger Professor, Dalhousie University.
Membres :	Cluadia Faggian Chargée de Recherche, CNRS.
	Romain Péchoux (Thesis director) Professeur, Université de Lorraine.
	Vladimir Zamdzhiev (Co-supervisor) Chercheur, ISFP, INRIA.

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgements

These three years as a PhD student have taught me many things that go beyond the excellent and extensive knowledge I gained in quantum computing.

A PhD without the support of supervisors would be an arduous and daunting task, and so I would like to first thank my supervisors, Romain Péchoux and Vladimir Zamdzhiev, for making this journey both interesting and insightful. I am also grateful to my jury members — Véronique Cortier, Claudia Faggian, Chris Heunen, and Peter Selinger — for taking the time to read this manuscript, provide valuable suggestions, and participate in my thesis defense. I would like to thank my collaborators, Louis Lemonnier and Alejandro Díaz-Caro, for their contributions to the research we produced during this time. Working with Alejandro in particular was both rewarding and enjoyable, and I am deeply thankful for his mentorship and guidance.

A PhD is as much an emotional journey as an intellectual one, and I would like to thank my close friends — Chandranan Dhar, Bharath Roy Chaudhury, Sayanika Roy Chaudhury, and Jérémie Koenig — for being my support system away from home. I am especially grateful to my sister, Vedija Dave, and my brother-in-law, Sneh Parikh, for helping me through the intense moments that are inevitable in the life of a PhD student. My parents, Vijay Dave and Sujata Dave, have provided relentless support and encouragement, which has been vital in moments of doubt and exhaustion. Finally, I would like to thank my fiancé, Shivam Trivedi, whom I met at a turning point in my PhD, for his patience, unwavering support, and belief in me as I navigate the end of this remarkable experience.

To my belated grandparents, Ratilal & Anusuya Dave.
To my parents, Vijay & Sujata Dave.

Contents

1
Introduction (en français) **1**

1.1 Les points de vue de Heisenberg et de Schrödinger sur la mécanique quantique . . 1
1.2 Langages de programmation quantiques - Aperçu général 3
1.3 Contrôle classique et quantique 5
1.4 Contribution et organisation de la thèse 6

2
Introduction

2.1 Heisenberg vs. Schrödinger views of Quantum mechanics 9
2.2 Quantum Programming Languages - An overview 11
2.3 Classical and Quantum Control 12
2.4 Contribution and organization of the thesis 14

3
Mathematical Background

3.1 Hilbert spaces and Quantum mechanics 15
3.2 von Neumann Algebras 24
3.3 C^*-algebras 31
3.4 The Simply Typed Lambda Calculus 38
3.5 Category Theory 40
3.6 Linear Logic 52

4
Quantum Programming Languages

4.0.1 Towards a quantum programming language 60
4.0.2 Quantum λ-calculi 63
4.0.3 Linear algebraic calculi & Proof languages 65
4.0.4 Symmetric pattern-matching and quantum control 67

4.0.5 A calculus for quantum control 68
4.0.6 Other languages 75

5
Combining quantum and classical control

5.1 Quantum Control Fragment 81
5.2 The main calculus 81
5.2.1 Syntax 81
5.3 Operational Semantics 83
5.4 Denotational Semantics 94
5.4.1 Soundness and Adequacy 101
5.5 Example: Quantum Walk Search 103
5.6 Related Work 104
5.7 Conclusion 105

6
A linear algebraic λ-calculus combining quantum and classical control

6.1 Introduction 107
6.2 Syntax and Derivation rules 108
6.3 Operational Semantics 112
6.3.1 Correctness 115
6.4 Denotational Semantics 128
6.5 Encoding vectors and matrices 137
6.6 Related Work 139
6.7 Conclusion 140

7
Conclusion

Bibliography **143**

List of Figures

1.1 Exemples de programmes. 6

2.1 Example programs. 13

3.1 Dirac notation. 19
3.2 Syntax of the simply typed lambda calculus. 38
3.3 Typing judgements for the simply typed lambda calculus. 39
3.4 Interpretation of typing judgements for the simply typed λ-calculus. 48
3.5 Proof rules (Classical Logic). 54
3.6 Proof rules (Classical Linear Logic). 56
3.7 Proof rules (Intuitionistic Logic). 56
3.8 Proof rules (Intuitionistic Linear Logic). 57

4.1 A simple quantum control flow. 60
4.2 QPL syntax (excerpt). 62
4.3 Typing judgements (excerpt). 62
4.4 Terms of the quantum lambda calculus. 64
4.5 Reduction rules (an excerpt). 64
4.6 Syntax. 67
4.7 Orthogonality of values. 68
4.8 The term grammar. 69
4.9 Rules for the orthogonality relation. 71
4.10 Orthonormal basis predicates. 71
4.11 Formation rules for terms and unitaries (excerpt). 72
4.12 Interpretation of pure quantum terms and unitaries. 74
4.13 Basic equational rules. 75
4.14 Congruence equational rules. 75
4.15 Vector space and linear applications equational rules. 76
4.16 Computational equational rules of the pure subsystem. 77

5.1 A program and the corresponding circuit. 79
5.2 Syntax of the classically controlled system. 82
5.3 Typing rules for terms. 83
5.4 Graphical representation of a well-formed quantum configuration. 85
5.5 Unitary permutations for reduction rules. 87
5.6 Reduction rules. 87
5.7 Categorical relations. 94
5.8 Interpretation of types. 98

5.9 Interpretation of typing judgements. 99

6.1 Syntax. 109
6.2 Derivation rules. 111
6.3 Reduction relations $\leftrightarrows$ and $\rightarrow$. 114
6.4 Interpretation of judgements. 130

1

Introduction (en français)

Cette thèse se concentre sur les aspects théoriques et logiques de la conception des langages de programmation pour l'informatique quantique. L'informatique quantique est un domaine prometteur qui repousse les limites de ce qui est physiquement calculable dans un délai raisonnable en exploitant les principes qui sous-tendent la mécanique quantique. Par conséquent, la question de la conception de langages de programmation adaptés au matériel quantique est devenue de plus en plus pertinente. Cependant, l'informatique quantique est régie par des principes fondamentalement différents de ceux de l'informatique classique. Elle fonctionne sous les contraintes de principes tels que le *non-clonage*, l'*intrication*, la *réversibilité* et la *superposition*. Les systèmes de types et les modèles de calcul traditionnels, tels que le lambda calcul, doivent donc être adaptés pour prendre en charge les données quantiques et leurs opérations de manière cohérente et compositionnelle. Cette thèse explore la conception des langages de programmation à travers le prisme de la théorie des catégories, de la théorie des types, des modèles dénotationnels, de la logique et des lambda calculs algébriques linéaires afin de permettre de raisonner sur les programmes quantiques d'ordre supérieur tout en respectant les lois de la mécanique quantique.

De nombreux travaux ont été menés dans ce domaine de recherche depuis le début des années 2000. On peut alors se demander quel nouvel aspect un langage supplémentaire apporte à une littérature déjà très riche. Cette thèse offre deux nouvelles perspectives dans ce domaine, l'une en adoptant une approche strictement *sémantique* et l'autre en adoptant une approche *logique* de la conception des langages. Nous insistons sur le fait que la division traditionnelle des langages de programmation quantique et des langages de preuve en deux catégories basées sur le flot de contrôle est à la fois restrictive et inutile ; ce que les contributions de cette thèse cherchent à illustrer. Par ailleurs, nous tentons de relier la sémantique des langages que nous discutons *directement* aux structures mathématiques qui constituent un formalisme naturel pour la mécanique quantique, tissant ainsi des liens étroits entre la physique et la conception des langages. Au lieu de fabriquer artificiellement une syntaxe de base correspondant à l'information quantique et de définir séparément leur comportement sémantique particulier, nous tentons de concevoir un cadre qui prend naturellement en charge ces constructions, éliminant ainsi la nécessité de définir des notions supplémentaires.

1.1 Les points de vue de Heisenberg et de Schrödinger sur la mécanique quantique

La mécanique quantique est un cadre fondamental qui régit le comportement des systèmes physiques à l'échelle des atomes et des particules subatomiques. Contrairement au cas clas-

sique, où l'état d'un système est décrit par des variables déterministes telles que la position, les états des systèmes quantiques sont décrits par des vecteurs d'état (fonctions d'onde) dans un espace de Hilbert. Les observables sont modélisés à l'aide d'opérateurs hermitiens, tandis que la mesure est irréversible et produit un résultat probabiliste. La mécanique quantique est régie par les principes fondamentaux que sont la *superposition*, l'*intrication* et l'*évolution unitaire*. La formulation *à l'état pur* de la mécanique quantique représente un état purement quantique par un vecteur d'état $|\psi\rangle$ dans un espace de Hilbert. Un tel état fournit la description complète de l'état quantique du système considéré. Cependant, dans la réalité, une connaissance complète n'est pas toujours possible. Il est souvent utile de modéliser les interactions entre des systèmes purement quantiques et des systèmes classiques, ou d'intégrer le fait que l'état complet d'un système quantique n'est pas connu. Dans de tels circonstances, la formulation de la mécanique quantique en termes d'état mixte s'avère plus utile. Dans ce contexte, l'état d'un système potentiellement hybride ou *ouvert* (c'est-à-dire en interaction avec l'environnement) est décrit à l'aide d'un type particulier de matrices souvent désignées par des lettres grecques telles que ρ, et appelées *matrices de densité*.

La formulation à l'aide d'états mixtes de la mécanique quantique est utilisée par deux visions physiques largement acceptées de cette théorie : la vision de Schrödinger et celle de Heisenberg. Ces deux visions offrent une perspective complémentaire sur la manière dont les systèmes physiques évoluent dans le temps selon les lois de la mécanique quantique. Dans la conception de Schrödinger, les états quantiques évoluent avec le temps et les observables sont représentés par des opérateurs statiques, indépendants du temps, qui modifient un vecteur d'état dépendant du temps. En revanche, la conception de Heisenberg fixe le vecteur d'état et représente les observables comme des opérateurs évoluant dans le temps qui agissent sur ce vecteur d'état.

La formulation mathématique de ces deux visions, ainsi qu'une dualité formelle entre elles, sont présentées à l'aide d'espaces appropriés d'opérateurs sur des espaces de Hilbert. Pour un espace de Hilbert H donné, l'espace $B(H) := \{x : H \to H \mid x \text{ est un opérateur borné}\}$, possède également la structure d'un espace vectoriel lorsqu'il est équipé d'opérations canoniques d'addition et de multiplication (composition) d'opérateurs. De plus, cet espace satisfait certaines propriétés *topologiques* supplémentaires, ce qui en fait une *algèbre de von Neumann*. Un espace d'opérateurs tout aussi important sur un espace de Hilbert H est $T(H)$, l'ensemble des opérateurs *de classe trace* sur H. $T(H)$ n'est pas nécessairement une algèbre de von Neumann en général, mais c'est un *espace d'opérateurs*. $T(H)$ peut être compris de manière informelle comme l'ensemble des opérateurs sur H qui admettent la notion de trace. $B(H)$ et $T(H)$ sont tous deux des exemples importants d'espaces d'opérateurs. Ces espaces constituent les pierres angulaires de la formulation mathématique de la dualité Schrödinger-Heisenberg.

Une opération quantique (souvent appelée canal quantique), d'un espace de Hilbert H_1 à un espace de Hilbert H_2, est alors représentée comme une application *Complètement Positive et Préservant la Trace* (CPTP) $\phi : T(H_1) \to T(H_2)$ dans la vision de Schrödinger. En revanche, ce même canal quantique est représenté comme une application *Normale, Complètement Positive et Unitaire* (NCPU) $\phi^* : B(H_2) \to B(H_1)$dans la vision de Heisenberg, dont la direction est opposée à celle de ϕ et qui est déterminée comme l'adjoint *de l'espace de Banach* de ϕ. La formulation mathématique de la dualité Schrödinger-Heisenberg aboutit à un résultat qui stipule qu'il existe une correspondance bijective entre les applications CPTP $\phi : T(H_1) \to T(H_2)$ et les applications NCPU $\phi^* : B(H_2) \to B(H_1)$.

La plupart des langages de programmation adoptent souvent implicitement la vision de Schrödinger pour formuler leur sémantique, en se concentrant sur l'évolution des états quantiques (registres quantiques) à travers une séquence de portes. L'adoption de la vision de Heisenberg ou, plus généralement, l'utilisation de la formulation mathématique basée sur l'algèbre de von

Neumann (comme nous le faisons dans cette thèse) présente cependant un avantage évident. On ne sait pas très bien comment, dans le contexte d'opérateurs préservant la trace, on pourrait modéliser les informations classiques avec les informations quantiques. C'est un problème qui se résout immédiatement lorsqu'on travaille avec les algèbres de von Neumann. En effet, les algèbres de von Neumann *commutatives* (c'est-à-dire les algèbres de von Neumann dans lesquelles l'opération de multiplication est commutative) peuvent être utilisées pour représenter des informations purement classiques. Un langage complet intégrant ces deux points de vue à l'aide d'espaces d'opérateurs reste un problème ouvert qui se situe au delà du cadre de cette thèse.

1.2 Langages de programmation quantiques - Aperçu général

La conception et le développement des langages de programmation quantiques ont évolué parallèlement aux avancées théoriques et expérimentales dans le domaine de l'informatique quantique. Ce domaine est né de la rencontre entre l'informatique et la physique, sous l'impulsion des travaux fondamentaux de pionniers tels que Feynman, Deutsch et bien d'autres. La machine de Turing quantique [B82, D85] - équivalent quantique de la machine de Turing classique, imaginée par Deutsch en 1985, a introduit la notion d'*universalité quantique*, tandis que les algorithmes développés par Shor [S94] et Grover [G96] ont démontré sa puissance en montrant une accélération exponentielle par rapport aux algorithmes classiques pour quelques problèmes de calcul. Ces percées ont mis en évidence la nécessité de disposer de modèles de calcul et de langages capables d'exprimer et de raisonner sur les processus quantiques - des processus fondamentalement différents de leurs équivalents classiques et régis par des principes tels que la *superposition*, l'*intrication*, l'*évolution unitaire*, le *non-clonage* et la *mesure*.

Certaines des toutes premières tentatives d'élaboration d'ordinateurs quantiques se sont concentrées sur des *modèles de circuits de bas niveau*, similaires à l' approche centrée sur le matériel adoptée par la programmation classique en langage assembleur. Le langage QCL (Quantum Computation Language) [O03] introduit par Bernhard Ömer en 2000, l'un des premiers langages de programmation quantique majeurs, suivait une approche impérative très proche du C, avec prise en charge de la manipulation explicite des registres quantiques et des structures de contrôle. QCL modélisait le contrôle classique sur les données quantiques, mais ne disposait pas d'une sémantique formelle ou d'un système de types permettant de garantir des propriétés quantiques tels que le non-clonage. Quipper [GLRSV13], un langage plus récent (Green et. al, 2013), a adopté une approche fonctionnelle et a étendu la programmation au niveau des circuits en intégrant des fonctionnalités quantiques dans Haskell, permettant ainsi la génération de circuits évolutifs. Cependant, il fonctionne encore largement dans un environnement hôte classique avec peu de garanties formelles quant à l'exactitude quantique.

Les limites d'une approche focalisée sur des circuits de bas niveau étant devenues évidentes, en particulier pour la programmation quantique d'ordre supérieur, compositionnelle et correcte par construction, la conception des langages de programmation quantiques a pris un tournant plus fonctionnel et plus logique. Le lambda calcul quantique, introduit indépendamment par van Tonder [V04](2004) et Selinger & Valiron [SV06](2006), a étendu le lambda calcul classique aux données et opérations quantiques. Ces modèles imposent notamment des contraintes de linéarité afin de garantir la non-duplication des données quantiques. De plus, les opérations quantiques sont modélisées comme des transformations unitaires et des résultats de mesure probabilistes. Le lambda calcul quantique de Selinger & Valiron a intégré un système de types linéaires inspiré de la logique linéaire [G87](Girard, 1987). Ce dernier garantit que les variables quantiques ne sont utilisées qu'une seule fois et empêche ainsi efficacement la duplication des états quantiques.

Ces modèles ont ouvert la voie au développement de langages de programmation riches sur le plan sémantique et sûrs sur le plan des types, adaptés à l'informatique quantique. De plus, la correspondance *Curry-Howard-Lambek* [H80, L80] a inspiré la conception d'une *sémantique dénotationnelle* pour ces modèles, en utilisant la théorie développée par la mécanique quantique catégorielle [AC04], abordée pour la première fois par Abramsky & Coecke en 2004. Leurs travaux ont introduit les *dagger compact closed categories* , dans lesquelles les programmes peuvent être interprétés comme des morphismes dans une catégorie monoidale symétrique dotée d'une structure quantique suffisante. Ce cadre a également servi un objectif plus profond. Grâce à la correspondance *Curry-Howard-Lambek*, ces derniers ont formalisé les principes d'intrication, d'unitarité et de composition, afin de les intégrer dans la programmation quantique. Ainsi, ce cadre a servi de fondement mathématique à la programmation quantique et a soutenu le raisonnement équationnel et les calculs graphiques tels que le calcul ZX [CD08, CD08a, B14].

Une autre ligne de recherche explorée en parallèle est celle des calculs lambda algébriques linéaires tels que *Lineal* [AD17](Arrighi & Dowek, 2008), qui augmentent le lambda calcul classique avec une structure d'espace vectoriel sur les termes, leur permettant d'être superposés, mis à l'échelle et combinés linéairement. Ces modèles représentent le calcul quantique comme des chemins parallèles à explorer, à travers une combinaison linéaire de chemins de calcul, suivant de près la formulation de l'espace de Hilbert et des opérateurs linéaires de la mécanique quantique. Cela a notamment entraîné de nouveaux défis centrés sur la conception des langages de programmation, ceux liés à la confluence et à la normalisation des chemins de réduction, car la sémantique doit préserver la structure linéaire des termes et respecter l'interférence.

Une approche plus logique, qui intégre également la structure de l'espace vectoriel sur les termes, a été adoptée par Díaz-Caro et al. dans [DCDR19, DCM22, DCM20, DGMV19]. Certains de ces travaux ont intégré une nouvelle modalité dans un cadre logique pour y parvenir, tandis que d'autres s'inspiraient du cadre dénotationnel établi dans des catégories appropriées de semi-modules et d'anneaux.

Les approches récentes prennent une tournure plus pratique, comme le montrent des travaux tels que *QWire* [PRZ17], *Quipper* [GLRSV13], *Proto-Quipper* [FKRS20] et *Silq* [BBMV20]. Silq intègre le principe du calcul automatique pour gérer les qubits auxiliaires et garantir la réversibilité. Cela préserve automatiquement la structure quantique de la sémantique tout en réduisant la charge de travail du programmeur. Les nouveaux frameworks tels que Cirq [OTCH20], Qiskit [WVN19] et PennyLane [BISG18] se concentrent sur le calcul hybride classique-quantique et les algorithmes variationnels ciblant les dispositifs quantiques à échelle intermédiaire à court terme (NISQ).

Tous ces travaux ont constitué des tentatives crédibles et fructueuses pour maintenir une adhésion stricte aux principes fondamentaux de la mécanique quantique :

- **La superposition** nécessite la prise en charge de la représentation et de la manipulation d'une combinaison linéaire de termes ;
- **L'intrication** nécessite de s'éloigner des notions bien acceptées d'indépendance des variables et de localité ;
- **Le non-clonage** rend nécessaire l'utilisation de systèmes de types linéaires ou affines ;
- **La mesure**, opération fondamentalement non déterministe et irréversible, doit être soigneusement modélisée dans la sémantique.

La conception de langages de programmation pour l'informatique quantique est un domaine riche, à la croisée de l'informatique théorique, de la logique mathématique et de la physique

quantique. À mesure que le matériel quantique continue d'évoluer, le besoin de langages de programmation adaptables et expressifs, offrant également des garanties formelles, un suivi strict de l'utilisation des ressources et une sémantique compositionnelle, demeure. Une conception idéale unifierait ces fondements théoriques avec des chaînes d'outils pratiques capables de prendre en charge le développement de logiciels quantiques évolutifs.

1.3 Contrôle classique et quantique

Il existe deux grands paradigmes dans la conception des langages de programmation quantiques basés sur le flot de contrôle des programmes. L'un est le contrôle dit « classique » : ici, le flot de données est conditionné par des données classiques pouvant résulter d'une mesure quantique. Les systèmes de types pour les langages de programmation quantiques basés sur le contrôle classique sont suffisamment expressifs pour modéliser des effets tels que la préparation d'états, l'évolution unitaire des données quantiques et les effets probabilistes induits par les mesures. Les lambda calculs qui sont basés sur ce paradigme utilisent naturellement des formalismes pour les structures quantiques basés sur les états mixtes (matrices de densité et superopérateurs, par exemple). En revanche, dans l'approche du contrôle quantique, l'accent est mis sur le calcul à état pur et l'expression des principes quantiques par la superposition de termes, les transformations unitaires et la réversibilité. L'approche du contrôle classique modélise un ensemble fixe de portes unitaires de base comme des constantes, avec une sémantique codée en dur. Les transformations unitaires plus complexes sont ensuite modélisées comme des circuits composés de ces constantes de base. L'approche du contrôle quantique fournit toutefois un support syntaxique et sémantique plus fondamental pour modéliser directement des transformations unitaires arbitraires, plutôt que de les décomposer en un programme compositionnel composé de diverses constantes. Cela rend la syntaxe du programme plus concise et libère le programmeur du fardeau de la décomposition. Cependant, l'approche du contrôle quantique a un coût, dans la mesure où elle ne fournit généralement pas de support pour les calculs efficaces, en particulier les programmes hybrides avec mesure.

La différence entre ces deux paradigmes peut être mieux comprise à l'aide d'exemples de programmes. Considérons les deux exemples simples de pseudo-code donnés dans la Figure 1.1. L'exemple de la Figure 1.1a mesure un qubit et, en fonction de la valeur binaire obtenue (données classiques), applique la porte Hadamard à un autre qubit ou le renvoie tel quel. Par conséquent, le flot de contrôle dans cet exemple est conditionné par des données classiques, qui sont obtenues à la suite d'une opération quantique efficace. Il s'agit d'un exemple particulièrement simple de l'idée selon laquelle un programme traite les circuits quantiques comme des données à manipuler par des programmes dictés par un flot de contrôle classique. L'approche complémentaire du flot de contrôle quantique, en revanche, soutient que les circuits quantiques peuvent eux-mêmes être considérés comme des programmes. Il s'agit de l'analogue quantique des langages de programmation de haut niveau purement classiques. Dans un tel contexte, tout élément d'un circuit quantique doit être pris en charge en tant que syntaxe. Par conséquent, toutes les opérations quantiques autorisées doivent être traitées comme des constructions syntaxiques. L'exemple de la Figure 1.1b applique la *version contrôlée* (notée CU) d'une porte unitaire fixe arbitraire à deux qubits. Ici, le flot de contrôle est basé sur des données quantiques, puisque la porte CU applique la porte unitaire U, ou l'opération d'identité ou une combinaison linéaire des deux sur le deuxième qubit, en fonction de l'état du premier. Par conséquent, un langage basé sur le contrôle quantique doit prendre en charge syntaxiquement la construction de toute porte unitaire arbitraire U, ainsi que des opérations telles que sa version contrôlée. Les deux exemples évoqués

ci-dessus traitent du flot de contrôle élémentaire (quantique et classique) basé sur des instructions conditionnelles terminales. Ceux-ci peuvent être généralisés davantage pour fixer également des points et des boucles, où l'exécution du corps de la structure pertinente dépend désormais des données quantiques (respectivement classiques) pour prendre en charge des versions avancées du contrôle quantique (respectivement classique). Il convient de noter que les langages basés sur le contrôle quantique s'appuient essentiellement sur le *principe de la mesure différée pour* garantir qu'aucune puissance expressive n'est perdue lors de l'adoption de ce paradigme. Le principe de la mesure différée stipule que toute mesure dans un circuit quantique peut être différée jusqu'à la fin du calcul, sans affecter le résultat final.

$$
\begin{aligned}
&\textit{let } x = |1\rangle \ \textit{ in}\\
&\textit{let } y = {}^1\!/\!\sqrt{2} \cdot |0\rangle + {}^1\!/\!\sqrt{2} \cdot |1\rangle \ \textit{ in}\\
&\textit{case } \textit{meas}(x) \ \textit{ of}\\
&\quad \mid 0 \mapsto \textit{Had } x\\
&\quad \mid 1 \mapsto x
\end{aligned}
$$

(a) Un programme avec un flux de contrôle classique.

Let U denote a 2×2 unitary gate:

$$
\begin{aligned}
&\textit{let } x = |0\rangle \otimes |0\rangle \ \textit{ in}\\
&CU(x)
\end{aligned}
$$

(b) Un programme avec flux de contrôle quantique.

Figure 1.1: Exemples de programmes.

Ainsi, les deux approches ont leurs avantages et leurs inconvénients, mais ne parviennent pas à fournir une solution complète pour concevoir un langage de programmation suffisamment expressif et pratique pour écrire des programmes hybrides quantiques et classiques. Cette thèse tente de convaincre les lecteurs que cette division est artificielle et que les langages modernes auraient tout intérêt à s'éloigner de l'adhésion à l'un de ces groupes. À l'instar du développement des langages de programmation classiques, nous souhaitons faire progresser l'état de l'art vers la conception d'un langage de plus *haut niveau*, qui s'éloigne de la description habituelle *bas niveau* des programmes en termes de circuits.

1.4 Contribution et organisation de la thèse

Cette thèse produit principalement deux langages différents : l'un est un langage théorique qui peut être considéré comme une extension du lambda calcul quantique, et l'autre est un langage de preuve pour un fragment de logique linéaire. Ces deux travaux ont été publiés respectivement lors des conférences FoSSaCS '25 et APLAS '25. Ils peuvent tous deux être considérés comme un ajout à la littérature sur les langages de programmation pour l'informatique quantique, tout en s'éloignant de la vision complémentaire du contrôle quantique et classique décrite ci-dessus. Le premier travail est une combinaison de certains travaux décrits dans les chapitres 4 et 5. La partie décrite dans le chapitre 5 est l'œuvre de l'auteur. Ensemble, ils décrivent un langage qui combine le contrôle quantique et classique en *intégrant* un fragment purement quantique dans un cadre plus large qui prend en charge le calcul classique ainsi que toutes les autres opérations efficaces susceptibles d'accompagner un calcul quantique. En outre, cet ouvrage décrit également une sémantique dénotationnelle basée sur les algèbres de von Neumann pour le langage. Il aborde la dualité Schrödinger-Heisenberg évoquée plus haut.

Le travail présenté au chapitre 6 décrit un langage de preuve linéaire pour le fragment intu-

itionniste multiplicatif additif de la logique linéaire, et suggère qu'il s'agit d'une logique idéale pour modéliser le calcul quantique à états mixtes. En effet, la syntaxe ne comporte aucune construction particulière pour les opérations quantiques, mais établit à la fois une sémantique opérationnelle et une sémantique dénotationnelle qui intègrent naturellement ces opérations. Elle peut également être considérée comme un lambda calcul algébrique linéaire. Le modèle dénotationnel est basé sur les *C*-algèbres*, une généralisation des algèbres de von Neumann. Il convient de noter que la sémantique opérationnelle devient également particulièrement simple, car nous supprimons la nécessité d'une configuration spéciale pour décrire les états quantiques dans les modèles du lambda calcul connus précédemment, à savoir les *configurations quantiques*.

Le reste de cette thèse est organisé comme suit :

- Le chapitre 3 décrit toutes les bases mathématiques nécessaires à la compréhension des chapitres suivants, en mettant l'accent sur les espaces de Hilbert, les principes fondamentaux du calcul quantique, le lambda calcul simplement typé, la logique linéaire, la théorie des catégories, les algèbres de von Neumann et les algèbres C*.
- Le chapitre 4 donne une brève description de la littérature sur les langages de programmation quantiques existants, avec certaines sections décrivant les détails pertinents des travaux les plus proches de ceux présentés dans cette thèse.
- Le chapitre 5 décrit l'une des contributions originales de cette thèse et décrit en détail une extension du lambda calcul quantique intégrant à la fois le contrôle quantique et classique.
- Le chapitre 6 décrit l'autre contribution originale de cette thèse, qui est un langage de preuve pour un fragment de logique linéaire servant de modèle pour le calcul quantique à états mixtes.

2

Introduction

The focus of this thesis is on the theoretical and logical aspects of programming language design for quantum computing. Quantum computing is emerging as a promising field expanding the limits of what is physically computable in a reasonable time span, by exploiting the principles underlying quantum mechanics. Therefore, the question of designing suitable programming languages tailored to quantum hardware has become increasingly relevant. However, quantum computation is governed by fundamentally different principles than classical computing. It operates under the constraints of principles such as *no-cloning*, *entanglement*, *reversibility* and *superposition*. Traditional type systems and models of computation such as the lambda calculus, must therefore be adapted to accommodate quantum data and its operations in a sound and compositional way. This thesis explores the design of programming languages through the lens of category theory, type theory, denotational models, logic and linear algebraic lambda calculi to contribute to the foundational reasoning of higher-order quantum programs while remaining faithful to the underlying laws of quantum mechanics.

There has been a lot of work in this line of research since the early 2000s. One might then ask, what new aspect does yet another language add to an already existing vast literature? This thesis offers two new insights into the field, one by taking a strictly *semantic* approach and another by taking a *logical* approach to language design. We insist that the traditional existing divisions of quantum programming languages and proof languages into two categories based on control flow, is both restrictive and unnecessary. The contribution of this thesis is a testament to this statement. Even more, we attempt to connect the semantics of the languages we discuss *directly* to mathematical structures which are a natural choice of framework for quantum mechanics, thus tightly knitting the physics with language design. Instead of artificially manufacturing *ground* syntax corresponding to quantum information and separately defining their special semantic behavior, we attempt to design a framework which naturally provides support for such constructs, thus removing the need for defining additional notions.

2.1 Heisenberg vs. Schrödinger views of Quantum mechanics

Quantum mechanics is a foundational framework governing the behavior of physical systems at the scale of atoms and subatomic particles. Unlike in the classical setting where system state is described by deterministic variables like position, quantum system states are described by state vectors (wave functions) in a Hilbert space. Observables are modeled using Hermitian operators, while measurement is irreversible resulting in a probabilistic outcome. Quantum mechanics is governed by the core principles of *superposition, entanglement* and *unitary evolution*. The *pure-*

state formulation of quantum mechanics, represents a purely quantum state by a single state vector $|\psi\rangle$ in a Hilbert space. Such a state provides the complete description of the system's quantum state. However, in reality, complete knowledge is not always possible. It is often useful to model interactions of purely quantum systems with classical ones, or incorporate the fact that the entire state of a quantum system is not known. In such instances, the *mixed-state* formulation of quantum mechanics turns out to be more useful. In this setting, the state of a possibly hybrid or *open* (i.e., interacting with the environment) is described using a special kind of matrices often denoted by Greek letters like ρ, and referred to as *density matrices*.

The mixed-state formulation of quantum mechanics is employed by two widely accepted physical views of this theory - the Schrödinger view and the Heisenberg view. These two pictures offer a complementary view on how physical systems evolve with time under the laws of quantum mechanics. In the Schrödinger picture, quantum states evolve with time and observables are represented by static, time-independent operators, modifying a time-dependent state vector. In contrast to this, the Heisenberg picture, fixes the state vector and represents observables as time-evolving operators acting on this state vector.

The mathematical formulation of these two views, and a formal duality between them is laid out using suitable spaces of *operators* on Hilbert spaces. For a given Hilbert space H, the space $B(H) := \{x : H \to H \mid x \text{ is a bounded operator}\}$, also has the structure of a vector space when equipped with canonical operations of addition and multiplication (composition) of operators. Moreover, this space satisfies certain additional *topological* properties, making it a *von Neumann algebra*. A similarly important space of operators on a Hilbert space H, is $T(H)$, the set of *trace-class* operators on H. $T(H)$ need not be a von Neumann algebra in general, but is an *operator space*. $T(H)$ can informally understood to be the set of operators on H that admit the notion of a trace. Both $B(H)$ and $T(H)$ are important instances of operator spaces. These spaces serve as foundational cornerstones in the mathematical formulation of the Schrödinger-Heisenberg duality.

A quantum operation (often referred to as a quantum channel), from a Hilbert space H_1 to a Hilbert space H_2 is then represented as a *completely positive trace preserving* (CPTP) map $\phi : T(H_1) \to T(H_2)$ in the Schrödinger picture. In contrast, this same quantum channel is represented as a *normal completely positive unital* (NCPU) map $\phi^* : B(H_2) \to B(H_1)$, whose direction is opposite to that of ϕ and is determined as the *Banach space* adjoint of ϕ. The mathematical formulation of the Schrödinger-Heisenberg duality amounts to a result which states that there is a bijective correspondence between CPTP maps $\phi : T(H_1) \to T(H_2)$ and NCPU maps $\phi^* : B(H_2) \to B(H_1)$.

Most programming languages often implicitly adopt the Schrödinger view for formulating their semantics, focusing on evolution of quantum states (quantum registers) through a sequence of gates. Adopting the Heisenberg view, or more generally using the von Neumann algebra based mathematical formulation (as we do in this thesis) has one clear advantage though. It is not clear how in the trace-class operator setting, one would model classical information along with quantum information. This is an issue that is immediately resolved when working with von Neumann algebras. This is because *commutative* von Neumann algebras (i.e., von Neumann algebras in which the multiplication operation is commutative) can be used to represent purely classical information. A more complete language which incorporates both these views using operator spaces is left for future work.

2.2 Quantum Programming Languages - An overview

The design and conception of quantum programming languages has evolved in tandem with the theoretical and experimental advances in quantum computing. The field itself is the child of a marriage between computer science and physics spearheaded by the foundational work of stalwarts such as Feynman, Deutsch and many others. The quantum Turing machine [B82, D85] - quantum counterpart of the classical Turing machine, ideated by Deutsch in 1985 introduced the notion of *quantum universality*, while algorithms developed by Shor [S94] and Grover [G96] demonstrated its power by showing exponential speedup over classical algorithms for a few computing problems. These breakthroughs highlighted the need for models of computation and languages that could express as well as reason about quantum processes - processes fundamentally different from their classical counterparts and governed by principles such as *superposition*, *entanglement*, *unitary evolution*, *no-cloning* and *measurement*.

Some of the very first attempts at programming quantum computers focused on *low-level circuit models*, similar to the hardware centric approach that classical assembly programming takes. *QCL* (Quantum Computation Language) [O03] introduced by Bernhard Ömer in 2000, one of the first major quantum programming languages followed an imperative approach closely resembling C, with support for explicit manipulation of quantum registers and control structures. QCL modeled classical control over quantum data but did not have a formal semantics or type system that could enforce principles such as the no-cloning theorem. Quipper [GLRSV13], a more recent work (Green et. al, 2013), took a functional approach and expanded circuit level programming by embedding quantum functionalities in Haskell, enabling scalable circuit generation. However, it still operates largely within a classical host environment with few formal guarantees about quantum correctness.

As limitations of such a low-level circuit based approach became apparent, in particular for higher-order, compositional and correct by construction quantum programming, the design of quantum programming languages took a more functional and logic-based turn. *The quantum lambda calculus*, introduced independently by van Tonder [V04](2004) and Selinger & Valiron [SV06](2006), extended the classical lambda calculus with quantum data and operations. Notably, these models impose linearity constraints to ensure non-duplicability of quantum data. Further, quantum operations are modeled as unitary transformations and probabilistic measurement outcomes. The quantum lambda calculus by Selinger & Valiron incorporated a linear type system inspired by Linear Logic [G87](Girard, 1987). This ensures that quantum variables are used exactly once, and effectively prevents duplication of quantum states.

These models paved the way for development of semantically rich and type-safe programming languages tailored to quantum computing. Further, the *Curry-Howard-Lambek* [H80, L80] correspondence inspired the design of *denotational semantics* for these models, using the theory laid out by categorical quantum mechanics [AC04], first discussed by Abramsky & Coecke in 2004. Their work introduced *dagger compact closed categories*, where programs can be interpreted as morphisms in a symmetric monoidal category with enough quantum structure. This framework also served a deeper purpose. Via the *Curry-Howard-Lambek* correspondence, they formalized the principles of entanglement, unitarity and composition, to be incorporated in quantum programming. Thus, it served as a mathematical foundation for quantum programming, and supported equational reasoning and graphical calculi such as the ZX-calculus [CD08, CD08a, B14].

Yet another line of development which was explored parallelly was that of linear-algebraic lambda calculi such as *Lineal* [AD17](Arrighi & Dowek, 2008), which augmented the classical lambda calculus with a vector space structure on terms, allowing them to be superposed, scaled and combined linearly. These models represented quantum computation as parallel paths to be

explored, through a linear combination of computation paths, closely following the Hilbert space and linear operator formulation of quantum mechanics. Notably, this brought in new challenges centered around programming language design, those related to confluence, and normalization of reduction paths, as the semantics must preserve the linear structure of terms and respect interference.

A more logical approach that also incorporated the vector space structure on terms was taken by Díaz-Caro et al. in [DCDR19, DCM22, DCM20, DGMV19]. Some of these works incorporated a new modality in a logical setting to achieve this, whereas others were inspired by the denotational framework laid down in suitable categories of semi-modules and rings.

Recent approaches take a more practical turn, as is evident in works like *QWire* [PRZ17], *Quipper* [GLRSV13], *Proto-Quipper* [FKRS20] and *Silq* [BBMV20]. Silq incorporates the principle of automatic uncomputation, to manage ancilla qubits and enforce reversibility. This automatically preserves the quantum structure of the semantics while reducing programmer burden. Emerging frameworks such as Cirq [OTCH20], Qiskit [WVN19] and PennyLane [BISG18] focus on hybrid classical-quantum computation and variational algorithms targeting near-term intermediate-scale quantum (NISQ) devices.

All these works have made credible and successful attempts to maintain strict adherence to the core principles of quantum mechanics:

- **Superposition** requires support for representation and manipulation of a linear combination of terms;
- **Entanglement** requires stepping away from well-accepted notions of variable independence and locality;
- **No-cloning** makes the use of linear or affine type systems necessary;
- **Measurement** being a fundamentally non-deterministic and irreversible operation, must be carefully modeled in the semantics.

Programming language design for quantum computing is a rich field at the intersection of theoretical computer science, mathematical logic and quantum physics. As quantum hardware continues to evolve, the need for adaptable and expressive programming languages that also come with formal guarantees, strict tracking of resource usage and a compositional semantics remains. An ideal design would unify these theoretical foundations with practical toolchains capable of supporting scalable quantum software development.

2.3 Classical and Quantum Control

There are two broad paradigms in the design of quantum programming languages based on the control flow in programs. One is “classical” control - here control flow of programs is conditioned on classical data potentially resulting from quantum measurement. Type systems for quantum programming languages based on classical control, are expressive enough to model effects such as state preparation, unitary evolution of quantum data and measurement induced probabilistic effects. Lambda calculi which are based on this paradigm naturally utilize frameworks for quantum structures which take the *mixed-state* approach (density matrices and superoperators for instance). In contrast, in the quantum control approach, the emphasis is on pure-state computation and expression of quantum principles through superposition of terms, unitary transformations and reversibility. The classical control approach models a fixed set of basic unitary gates as

constants, with hard-coded semantics. More involved unitary transformations are then modeled as circuits composed of these basic constants. The quantum control approach however, provides more fundamental syntactic and semantic support to model arbitrary unitary transformations directly, as opposed to breaking them down as a compositional program consisting of various constants. This makes the program syntax more concise and takes away the burden of decomposition from the programmer. However, the quantum control approach comes at a cost, in that, it does not generally provide support for effectful computations, particularly, hybrid programs with measurement.

The difference in these two paradigms can be best understood through example programs. Consider the two simple pseudo code examples given in Figure 2.1. The example in Figure 2.1a, measures a qubit and based on the resulting bit value (classical data), either applies the Hadamard gate to another qubit or returns it as is. Therefore, the control flow in this example is conditioned on classical data, which is obtained as a result of an effectful quantum operation. This is a particularly simple example of the idea that a program treats quantum circuits as data to be manipulated through classical control flow. The complementary approach of quantum control flow on the other hand argues, that quantum circuits can themselves be viewed as programs. This is the quantum analog of purely classical higher-level programming languages. In such a setting, any element of a quantum circuit must be supported as syntax. Therefore, all permissible quantum operations must be treated as syntactic constructs. The example in Figure 2.1b, applies the *controlled version* (denoted by CU) of an arbitrary fixed unitary gate to two qubits. Here, the control flow is based on quantum data, since the gate CU, applies the unitary gate U, or the identity operation or a linear combination of the two on the second qubit, based on the state of the first one. Therefore, a language based on quantum control must syntactically support the construction of any arbitrary unitary gate U, as well as operations such as its controlled version. The two examples discussed above, discuss elementary control flow (quantum and classical) based on terminating conditional statements. These can be generalized further to fix points and loops as well, where the execution of the body of the relevant structure now depends on quantum (respectively, classical) data to support advanced versions of quantum (respectively, classical) control. Notably, languages based on quantum control, crucially rely on the principle of *deferred measurement*, to ensure that no expressive power is lost in adopting this paradigm. The principle of deferref measurement states that any measurement in a quantum circuit can be deferred until the end of the computation, without affecting the final outcome.

$$
\begin{aligned}
&\textit{let } x = |1\rangle \ \textit{ in} \\
&\textit{let } y = {}^1\!/\!\sqrt{2} \cdot |0\rangle + {}^1\!/\!\sqrt{2} \cdot |1\rangle \ \textit{ in} \\
&\textit{case } \textit{meas}(x) \ \textit{of} \\
&\quad \mid 0 \mapsto \textit{Had } x \\
&\quad \mid 1 \mapsto x
\end{aligned}
$$

(a) A program with classical control flow.

Let U denote a 2×2 unitary gate:

$$
\begin{aligned}
&\textit{let } x = |0\rangle \otimes |0\rangle \ \textit{ in} \\
&CU(x)
\end{aligned}
$$

(b) A program with quantum control flow.

Figure 2.1: Example programs.

Thus, both approaches have their own advantages and disadvantages, but fail to provide a complete solution to design a programming language expressive and convenient enough to write hybrid quantum and classical programs. This thesis is an attempt to convince the readers that

this division is artificial, and modern languages would do well in moving away from adhering to one of these groups. Mirroring the development of classical programming languages, we wish to advance the state of the art towards designing a more *high-level* language, thus moving away from *low-level* circuit description of programs.

2.4 Contribution and organization of the thesis

This thesis primarily produces two different languages- one a theoretical framework which can be viewed as an extension of the quantum lambda calculus and the other a proof language for a fragment of Linear logic. Both of these works have been published at FoSSaCS '25 and APLAS '25 respectively. They can both be understood as addition to the programming language literature for quantum computing, while moving away from the complementary view of quantum and classical control described above. The first work, is a combination of some work described in Chapter 4 and Chapter 5. The part described in Chapter 5 is the work of the author. Together, they describe a language which combines quantum and classical control by *embedding* a purely quantum fragment into a larger framework which supports classical computation as well as all other effectful operations that can be expected to accompany a quantum computation. In addition to this, the work also describes a denotational semantics based on von Neumann algebras for the language. It touches upon the Schrödinger-Heisenberg duality discussed above.

The work in Chapter 6 describes a linear proof language for the Intuitionistic Multiplicative Additive fragment of Linear Logic, and suggests that this is an ideal logic to model mixed-state quantum computation. Indeed, the syntax has no particularly special construct for any quantum operations, but lays down both an operational and denotational semantics which naturally incorporate these operations. It can also be viewed as a linear algebraic lambda calculi. The denotational model is based on *C*-algebras*, a generalization of von Neumann algebras. Notably, the operational semantics also becomes particularly simple as we do away with the need for a special configuration to describe quantum states in previously known lambda calculi models - namely *quantum configurations*.

The rest of this thesis is organized as follows:

- Chapter 3 describes all the mathematical background necessary to understand the chapters that follow, with an emphasis on Hilbert spaces, fundamentals of quantum computation, the simply typed lambda calculus, linear logic, category theory, von Neumann algebras and C*-algebras.
- Chapter 4 gives a brief description of literature on existing quantum programming languages, with some sections describing relevant details of works closest to those presented in this thesis.
- Chapter 5 describes one of the original contributions of this thesis, and fully describes an extension of the quantum lambda calculus incorporating both quantum and classical control.
- Chapter 6 describes the other original contribution of this thesis, which is a proof language for a fragment of Linear logic serving as a model for mixed-state quantum computation.

3

Mathematical Background

In this chapter, I give a brief overview of the vast mathematical literature that was needed to produce the contributions of this thesis. The aim is to provide all the technical material needed to be able to understand the chapters that follow while remaining as concise as possible.

3.1 Hilbert spaces and Quantum mechanics

In this section we will give a brief overview of the basics of Quantum Computing, and the overarching reliance of its mathematical formulation on *Hilbert spaces*. Towards this we first begin with setting up some terminology from quantum computing and highlighting how it differs from its classical counterpart. Here we assume basic knowledge of the complex numbers, real numbers and their manipulation. After that we cover some important basic material on Hilbert spaces, while assuming knowledge of linear algebraic terminology such as vector spaces, basis, linear maps etc. For a detailed exposition on the basics of linear algebra we refer the reader to [HK71]. For a detailed description of Hilbert spaces and their ubiquitousness in the mathematics of quantum mechanics we refer the reader to [C10]. For a comprehensive overview of quantum information we refer the reader to [NC10].

Classical data is encoded as *bits*, which take a value in the binary set $\{0, 1\}$. Quantum data on the other hand is encoded as *qubits*, which take a value in the spectrum of the unit sphere of the complex plane. The values $e_0 := \langle 1, 0\rangle$, $e_1 := \langle 0, 1\rangle$ which represent the *computational basis* for $\mathbb{C}^2$ "span" the unit sphere, and represent quantum states denoted by $|0\rangle$ and $|1\rangle$ respectively in Dirac notation. A qubit can be in a quantum state $\psi := \alpha e_0 + \beta e_1$, with $\alpha, \beta \in \mathbb{C}$ such that $\|\alpha\|^2 + \|\beta\|^2 = 1$. Such a state is referred to as a *pure* quantum state, and represented as $|\psi\rangle := \alpha\,|0\rangle + \beta\,|1\rangle$ using Dirac notation, leading us to the first postulate of quantum mechanics:

Postulate 1 ([NC10]). *"Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space."*

This fundamental difference between quantum and classical information, results in a larger state space for quantum computing, which is exploited while designing quantum algorithms. von Neumann's mathematical formulation of quantum mechanics is based on Hilbert spaces. We now,

provide the necessary definitions to give a formal description of Hilbert spaces, before continuing with the postulates of quantum mechanics.

Definition 1 (Complex inner product space). *A complex inner product space is a vector space V over the field of complex numbers $\mathbb{C}$ equipped with a functional $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{C}$ called the inner-product, satisfying the following conditions:*

- $\langle x, y \rangle = \overline{\langle y, x \rangle}$ *for all* $x, y \in V$;
- $\langle x, \alpha y + \beta z \rangle = \alpha \langle x, y \rangle + \beta \langle x, z \rangle$ *for all* $x, y \in V$ *and* $\alpha, \beta \in \mathbb{C}$;
- $\langle x, x \rangle \geq 0$ *for all* $x \in V$;
- $\langle x, x \rangle = 0 \Leftrightarrow x = 0$.

Example 1. The one-dimensional vector space $\mathbb{C}$ with the inner product given by $\langle x, y \rangle := \overline{x} y$ for all $x, y \in \mathbb{C}$ is an inner-product space.

Definition 2 (Complex normed vector space). *A complex normed vector space is a vector space V over the field of complex numbers $\mathbb{C}$ equipped with a functional $\| \cdot \| : V \to \mathbb{C}$ called the norm, satisfying the following conditions:*

- $\|x\| \geq 0$ *for all* $x \in V$;
- $\|x\| = 0 \Leftrightarrow x = 0$;
- $\|\alpha x\| = |\alpha| \|x\|$ *for all* $\alpha \in \mathbb{C}$, $x \in V$;
- $\|x + y\| \leq \|x\| + \|y\|$.

Example 2. The one-dimensional vector space $\mathbb{C}$ with the norm given by $\|x\| = \sqrt{\overline{x} x}$ for all $x \in \mathbb{C}$ is a complex normed vector space.

Remark 1. Every inner product space V is also a normed vector space, with the norm given by $\|x\| := \sqrt{\langle x, x \rangle}$ for all $x \in V$.

Definition 3 (Cauchy sequence). *A sequence of vectors $\{x_n\}_{n \geq 1}$ in a normed vector space is said to be Cauchy if for every $\epsilon > 0$, there exists a number M such that $\|x_n - x_m\| \leq \epsilon$ for all $n, m \geq M$.*

Definition 4 (Dense set). *A subset $D \subseteq V$ of a normed space V is said to be dense in V, if for every $x \in V$ there is a sequence $\{x_n\}_{n \geq 1}$, with $x_i \in D$, such that $\{x_n\}_{n \geq 1}$ converges to x, i.e., for every $\epsilon > 0$, there exists N so that $\|x_n - x\| < \epsilon$ for $n > N$.*

Definition 5 (Banach space (Complete normed space)). *A normed vector space V is said to be complete if every Cauchy sequence $\{x_n\}_{n \geq 1}$ converges, i.e., there is an element $x \in V$, such that $\{x_n\}_{n \geq 1}$ converges to x. Such a vector space is known as a Banach space.*

Intuitively, a complete vector space can be understood to be a vector space without holes. In general, any finite dimensional complex vector space can be completed, i.e., the "holes" can be filled. Furthermore, the vector space $\mathbb{C}^n$ is a complete normed space for every $n > 0$.

Definition 6 (Hilbert Space). *A Hilbert space H is a complete inner product space.*

Example 3. Following the explanations above, $\mathbb{C}^n$, for every $n \geq 0$ is a Hilbert space, with the inner product given by $\langle x, y \rangle := \Sigma_{i=1}^{n} \overline{x_i} y_i$.

Example 4. For any set X, consider the following set:

$$\ell^2(X) := \{\phi : X \to \mathbb{C} \mid \Sigma_x \|\phi(x)\|^2 < \infty\}$$

This is known as the ℓ^2 construction over a set, and the resulting set of *square-summable* sequences of cardinality X in $\mathbb{C}$ is also a vector space. In fact, it is a very important example of a Hilbert space, with the inner product given by $\langle \phi_1, \phi_2 \rangle := \Sigma_x \overline{\phi_1(x)} \phi_2(x)$. The ℓ^2 construction will show up again as another interesting structure in §3.5. It also plays a significant role, elucidated in the work described in Chapter 5.

The notion of a *basis* of a vector space, can be extended to Hilbert spaces as well. However, the extra inner product structure and its completeness warrant more properties as is evident in the next definition.

Definition 7 (Orthonormal basis). *Given a Hilbert space H, an orthonormal basis for H is a family of elements $B := \{e_i\}_{i \in I}$ in H satisfying the following conditions:*

- $\langle e_i, e_j \rangle = 0$ *for all* $i, j \in I$ *with* $i \neq j$;
- $\|e_i\| = 1$ *for all* $i \in I$;
- $Span(B) := \{\Sigma_{i=i}^{n} \alpha_i e_i \mid \alpha_i \in \mathbb{C}, e_i \in B, n \in \mathbb{N}\}$ *is dense in* H.

B is known as an orthonormal basis of H.

Remark 2. The standard idea of a basis being a spanning set can be applied to Hilbert spaces. In such a case, the basis so obtained is known as a *Hamel basis*. However, Hilbert spaces are a specific instance of a more general class of vector spaces called *Banach spaces*. In general, proving existence of a Hamel basis for a Banach space requires the use of Axiom of Choice. However, that of an orthonormal basis is always guaranteed. Adopting this notion, generally when we refer to a basis for a Hilbert space, we mean an orthonormal basis.

Definition 8 (Separable Hilbert space). *A Hilbert space H is said to be separable if it has a countable basis.*

Example 5. The space $\mathbb{C}^n$ is a trivial example of a separable Hilbert space, since it has a finite basis.

Example 6. The space $\ell^2(\mathbb{N})$ is a separable Hilbert space. The set $\{\mathbb{1}_n \mid n \in \mathbb{N}\}$ ($\mathbb{1}_n(m) := 1$ if $n = m$ and $\mathbb{1}_n(m) := 0$ if $n \neq m$) forms a countable basis.

Recall that a qubit takes a value of the form $\alpha |0\rangle + \beta |1\rangle$, for $\alpha, \beta \in \mathbb{C}$ such that $\|\alpha\|^2 + \|\beta\|^2 = 1$. Now we have a formal notion of what it means for $\{e_0, e_1\}$ to be a spanning set for $\mathbb{C}^2$. The set forms an orthonormal basis for the Hilbert space $\mathbb{C}^2$, which is the state space of a qubit. Note that in this case the Schauder and Hamel basis coincide, and we simply refer to the spanning set as a basis.

Formally, all possible operations that can be performed on quantum states are *linear* in nature, bringing us to our next definition.

Definition 9 (Linear operator). *A linear operator between Hilbert spaces H_1, H_2 is a map $f : H_1 \to H_2$ satisfying the condition,*

$$f(\alpha\psi + \beta\phi) = \alpha f(\psi) + \beta f(\phi)$$

for all $\alpha, \beta \in \mathbb{C}$, $\psi, \phi \in H_1$.

Example 7. A simple example of a linear operator on a hilbert space H is the map $f : \mathbb{C} \to \mathbb{C}$ given by $f(\psi) := \alpha\psi$ for a fixed $\alpha \in \mathbb{C}$.

As is with vector spaces, linear operators on finite dimensional Hilbert spaces are often represented using matrices. The linear operator described in Example 7 is represented with the 1×1 constant matrix (α). In the realm of quantum computing, linear transformations acting on quantum states are often referred to as *quantum gates*. This phrase derives inspiration from circuit diagrams, which are pictoral representations of quantum algorithms, and similar in nature to circuits that represent classical computing. Hence, any quantum computation expressed as a circuit of quantum gates, can also be wrtten as a linear transformation (consequently, as a matrix) on Hilbert spaces using appropriate combinations. A class of linear operators on Hilbert spaces, that plays a significant role in the mathematics of quantum mechanics is those of *bounded* linear operators. Their significance becomes clear in §3.2, and their role in the work of this thesis is further emphasized later in Chapter 5, Chapter 6.

Definition 10 (Bounded linear operator). *Given two Hilbert spaces H_1, H_2 a linear operator $f : H_1 \to H_2$ is said to be bounded if there exists $r \in \mathbb{R}$ such that $\|f(\psi)\| \leq |r|\|\psi\|$ for all $\psi \in H_1$.*

A very important operation, on quantum gates (equivalently on linear operators between Hilbert spaces), is that of taking the *adjoint*.

Definition 11 (Adjoint). *Given a bounded linear operator $f : H_1 \to H_2$, its adjoint, denoted by $f^\dagger : H_2 \to H_1$ is the unique linear operator satisfying the condition,*

$$\langle f^\dagger(\psi), \phi\rangle = \langle \psi, f(\phi)\rangle$$

for all $\phi \in H_1, \psi \in H_2$. An operator $f : H \to H$ is said to be self-adjoint if $f^\dagger = f$.

Example 8. The adjoint of the scalar multiplication map given in Example 7 is given by $f^\dagger(\psi) := \overline{\alpha}\psi$.

Before we proceed further with the theory, we introduce the Dirac notation in more detail here. It is important to take some space to explain it, since it is used extensively throughout this thesis, and more generally in quantum computing literature. Dirac introduces the *bra-ket* notation, to denote quantum states and their transformations. We summarize the notation used for the relevant mathematical notions in Figure 3.1.

Another class of linear operators on Hilbert spaces, that plays a significant role in the mathematics of quantum mechanics is those of *bounded* linear operators. Their significance becomes clear in §3.2, and their role in the work of this thesis is further emphasized later in Chapter 5, Chapter 6.

Note that the operators in Examples 7,8 are both clearly bounded. In fact, it can be easily shown that all linear operators on finite dimensional Hilbert spaces are bounded. Since all permissible transformations on quantum data are necessarily reversible, the class of linear operators that can act as quantum transformations is a strict sub class of linear operators. Such reversible linear operators are known as *unitary* operators on Hilbert spaces.

Definition 12 (Unitary). *A linear operator $U : H_1 \to H_2$ is said to be unitary if U^{-1} exists and $U^{-1} = U^\dagger$.*

This brings us to the next postulate of quantum mechanics.

Linear map	**Matrix representation**	**Dirac notation**
$\psi : \mathbb{C} \to H$	$\begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix}$	$\|\psi\rangle$
$\psi^\dagger : H \to \mathbb{C}$	$\begin{pmatrix} \overline{c_1} & \dots & \overline{c_m} \end{pmatrix}$	$\langle\psi\|$
$\psi^\dagger \circ \phi$	$\begin{pmatrix} \overline{c_1} \dots \overline{c_m} \end{pmatrix} \begin{pmatrix} c'_1 \\ \vdots \\ c'_m \end{pmatrix}$	$\langle\psi \mid \phi\rangle$
$\psi \circ \psi^\dagger$	$\begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} \begin{pmatrix} \overline{c_1} & \dots & \overline{c_m} \end{pmatrix}$	$P_\psi := \|\psi\rangle \langle\psi\|$
$f \circ \psi$	$\begin{pmatrix} m_{11} & \dots & m_{1m} \\ \vdots & \vdots & \vdots \\ m_{n1} & \dots & m_{nm} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix}$	$f \|\psi\rangle$
$\phi^\dagger \circ f$	$\begin{pmatrix} \overline{c_1} & \dots & \overline{c_m} \end{pmatrix} \begin{pmatrix} m_{11} & \dots & m_{1m} \\ \vdots & \vdots & \vdots \\ m_{n1} & \dots & m_{nm} \end{pmatrix}$	$\langle\phi\| f$
$\phi^\dagger \circ f \circ \psi$	$\sum_{i,j} \overline{c_i} m_{ij} c'_j \quad \in \mathbb{C}$	$\langle\phi \mid f \mid \psi\rangle$

Figure 3.1: Dirac notation.

Postulate 2 ([NC10]). *"The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2 ,*

$$|\psi'\rangle = U |\psi\rangle \,."$$

Example 9. The operator defined in Example 7 is unitary iff $\|\alpha\| = 1$.

Example 10. There are various one qubit unitary gates which play a very important role in describing circuits for quantum algorithms. The *Pauli* gates and the *Hadamard* gate defined as linear transformations in the form of matrices below are amongst some of the most significant 1-qubit quantum gates:

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

An equally important set of gates is those of the *Phase* gates. These are single qubit gates that change the relative phase of the basis states, without altering the resulting probabilities when measured in the computaiton basis. A gate resulting in a phase change of ϕ is represented by the following matrix:

$$P(\phi) := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

Another important class of operators on Hilbert spaces is that of *isometries*. Intuitively, isometries preserve distance between vectors of a Hilbert space.

Definition 13 (Isometry). *Given two Hilbert spaces H_1, H_2, a linear map $f : H_1 \to H_2$ is said to be an isometry iff*

$$\langle h_1, h_2 \rangle = \langle f(h_1), f(h_2) \rangle \qquad \forall h_1, h_2 \in H_1$$

The next important distinction between classical data and quantum data is based on what kind of transformations are permissible for both. Classical data enjoys unrestricted access and manipulation, i.e., it is available at any point to all observers, it can be freely transformed and copied. Quantum data on the other hand cannot be copied, as stated by the *no-cloning* theorem ([C10] Chapter 3), permits only reversible transformations, and can only be accessed through *measurement*, leading us to the next postulate of quantum mechanics.

Postulate 3 ([NC10]). *"Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by,*

$$p(m) = \langle \psi \mid M_m^\dagger M_m \mid \psi \rangle,$$

and the state of the system after the measurement is

$$\frac{M_m |\psi\rangle}{\sqrt{\langle \psi \mid M_m^\dagger M_m \mid \psi \rangle}}."$$

Measurement is an operation that acts on quantum data and results in classical data. It is an irreversible transformation that "destroys" quantum data. Measuring a qubit in a state $\alpha |0\rangle + \beta |1\rangle$ results in the state $|0\rangle$ with probability $\|\alpha\|^2$ and the state $|1\rangle$ with probability $\|\beta\|^2$. A bijective correspondence between the two element set $\{0, 1\}$ and the set $\{|0\rangle, |1\rangle\}$ intuitively gives the idea that measurement results in a probability distribution over classical states. Note that

measurement results in loss of information in the sense that post measurement it is impossible to infer a qubit's initial quantum state $|\psi\rangle$.

Formally, measurement is often described by a collection of linear transformation acting on Hilbert spaces, represented generally using *projector* matrices.

Example 11 (Measurement). Measurement of a single qubit in the computational basis is represented by the set of self-adjoint operators $\{P_0, P_1\}$, given by the following matrices:

$$P_0 := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad P_1 := \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Note that $P_i \circ P_j = 0$ when $i \neq j$, and $P_i^2 = P_i$ for $i \in \{0, 1\}$. Indeed, each projector acts on a quantum state to reveal the value of its coordinate with respect to the corresponding basis element, i.e. $P_0(\alpha |0\rangle + \beta |1\rangle) = \alpha |0\rangle$, $P_1(\alpha |0\rangle + \beta |1\rangle) = \beta |1\rangle$. Thus any quantum state ψ undergoes a transformation under measurement to the state $|0\rangle$ with probability $\langle |\psi\rangle, P_0(|\psi\rangle)\rangle$ $(= \|\alpha\|^2)$ and to the state $|1\rangle$ with probability $\langle |\psi\rangle, P_1(|\psi\rangle)\rangle$ $(= \|\beta\|^2)$. These projectors form a resolution of the identity operator (i.e. $P_0 + P_1 = I$), are orthogonal to each other (i.e. $P_i \circ P_j = 0$ for $i \neq j$) and they are *idempotent* (i.e. $P_i^2 = P_i$). Note that these properties hold for measurement with respect to any basis.

From Example 11 and Postulate 3 it is clear that measurement is specified by a family of projectors with respect to the computational basis. This can be generalized further, to any orthonormal basis of the state space $\mathbb{C}^2$. In particular, given an orthonormal basis $\{b_1, b_2\}$ of $\mathbb{C}^2$, projectors with respect to this basis represent measurement of a qubit in this basis. Moreover, these projectors can be applied by composing suitable unitary transformations (corresponding to rotations on the unitary sphere) with the projectors in Example 11. Specifically, if a transformation $U : \mathbb{C}^2 \to \mathbb{C}^2$ represents a change of basis from the computational basis to $\{b_1, b_2\}$, the set of projectors representing measurement in the new basis is simply $\{U \circ P_0 \circ U^{-1}, U \circ P_1 \circ U^{-1}\}$. Hence, in physical circuits, generally measurement is represented with respect to the computational basis, and any other measurement is a circuit formed by composing suitable unitary gates.

So far, the theory dealing with single qubit computations has been introduced. However, any significant quantum algorithm involves multi-qubit computations. In order to represent them mathematically, we need ways to construct larger Hilbert spaces from smaller ones. We give two important constructions here.

Definition 14 (Direct sum). *Given two hilbert spaces H_1, H_2, their direct sum $H_1 \oplus H_2$ comprised of the direct sum of the underlying vector spaces, i.e,*

$$H_1 \oplus H_2 := \{(h_1, h_2) \mid h_1 \in H_1, h_2 \in H_2\}$$

so that $\alpha(h_1, h_2) + \beta(h_1', h_2') = (\alpha h_1 + \beta h_1', \alpha h_2 + \beta h_2')$, is a Hilbert space with the inner product defined as $\langle (h_1, h_2), (h_1', h_2') \rangle := \langle h_1, h_1' \rangle + \langle h_2, h_2' \rangle$.

Remark 3. The direct sum structure can be further generalized to finitely many hilbert spaces. Moreover, the direct sum of two Hilbert spaces comes equipped with the injections $i_1 : H_1 \to H_1 \oplus H_2$, $i_2 : H_2 \to H_1 \oplus H_2$ defined in the obvious way $i_1(h_1) := (h_1, 0)$ and $i_2(h_2) := (0, h_2)$. Consequently, we have that any Hilbest space H is isomorphic to its direct sum with the one element Hilbert space $\{0\}$.

The direct sum of Hilbert spaces is a good candidate to describe a pair of quantum states, except that it violates the next principle of quantum mechanics that we describe. Quantum

mechanics is governed by the principle of *entanglement*, which states that in any compound quantum system, one of its components cannot be described independently of the other. The direct sum structure on the other hand can be completely specified by the individual spaces that comprise it. Moreover, in this construction each coordinate of the composite can be independently described. This leads us to the next construction, which is that of the tensor product of Hilbert spaces. Before that, we recall the definition of tensor product of vector spaces.

Definition 15 (Tensor product (Vector spaces)). *Given two vector spaces V_1, V_2, their tensor product is a vector space $V_1 \otimes V_2$ along with a bilinear map $_ \otimes _ : V_1 \times V_2 \to V_1 \otimes V_2$ which satisfies the universal property that for any bilinear map $h : V_1 \times V_2 \to W$, there is a unique linear map $h' : V_1 \otimes V_2 \to W$ such that $h = h' \circ (_ \otimes _)$.*

Definition 16 (Tensor product (Hilbert spaces)). *Given two Hilbert spaces H_1, H_2, the vector space $H_1 \otimes H_2$ can be equipped with the inner product $\langle h_1 \otimes h_2, h_1' \otimes h_2' \rangle = \langle h_1, h_1' \rangle \langle h_2, h_2' \rangle$. The tensor product of H_1, H_2 is the completion of the vector space $H_1 \otimes H_2$ under the norm induced by this inner product.*

Remark 4. Given a n dimensional Hilbert space H_1 and a m dimensional Hilbert space H_2, the dimension of $H_1 \oplus H_2$ is $n + m$, whereas that of $H_1 \otimes H_2$ is nm.

With all the definitions in place, we now describe the final postulate of quantum mechanics.

Postulate 4 ([NC10]). *"The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$."*

We extend the Dirac notation to states of compound systems, by $|ij\rangle := |i\rangle \otimes |j\rangle$. The smallest example of a compound quantum system would be that comprising of two qubits. Formally, the principle of entanglement states that given the state of a two qubit system, it cannot always be written as the tensor product of two single qubit states. A very well-known example of entanglement in quantum mechanics are the *Bell-states*. The states $|00\rangle + |11\rangle$ and $|01\rangle - |10\rangle$ are known as the Bell-states. Physically, entanglement implies that the two quantum systems are correlated and hence one cannot be described independently of another. Testing this out on the Bell-state and the EPR-state requires performing measurements in several bases, and prove that the outcome violates *Bell's Inequality*.

The use of the tensor product setup to model compund quantum systems, and the requirement that any transformation which is not measurement be unitary leads us to the no-cloning theorem as a formal consequence. Consider two quantum systems in states $\psi \otimes \phi_1 \in H \otimes H$, for some Hilbert space H. We would want to construct a unitary operator $U : H \otimes H \to H \otimes H$, so that it uniformly copies the state of the first component of the system onto the second component. Hence, for any two quantum states $\psi_1, \psi_2 \in H$, we would want $U(\psi_1 \otimes \phi_1) = \psi_1 \otimes \psi_1$ and $U(\psi_2 \otimes \phi_1) = \psi_2 \otimes \psi_2$. Now, using the properties of unitarity of U, and manipulating the inner product $\langle U(\psi_1 \otimes \phi_1), U(\psi_2 \otimes \phi_1) \rangle$ it can be shown that we are forced to have ψ_1, ψ_2 are either orthogonal states or equal. Despite this seemingly restrictive nature of the unitarity condition, there are many multi qubit gates which are central to quantum algorithms. We discuss some of them in the next example.

Example 12. In this example we discuss two very well-known multi qubit quantum gates, the *CNOT* and the *Hadamard* gate. In the case of 2 qubits, both the gates are represented by 4×4 matrices acting on the Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2$. The CNOT gate flips the state of the second qubit conditioned on the first, and the Hadamard is obtained by the general construction of the $2^n \times 2^n$ Hadamard matrix from the 2×2 Hadamard matrix given in Example 10. Their matrix representations in the computational basis are given below:

$$H_{4\times 4} := \begin{pmatrix} H & H \\ H & H \end{pmatrix} \qquad CNOT := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Larger unitary transformation can also be constructed as a composite of smaller ones. For any unitary transformation U acting on an n qubit system, we can construct the controlled version of this gate acting on $n+1$ qubits, so that the new qubit acts as the control qubit. The controlled version of a unitary gate U is denoted by CU and is represented by the following matrix:

$$\begin{pmatrix} I_{2\times 2} & 0_{2\times 2^n} \\ 0_{2^n \times 2} & U \end{pmatrix}$$

Additionally, as is expected, tensor products of unitaries result in a unitary transformation on the resulting tensor product space, i.e, for $U_1 : H_1 \to H_2$ and $U_2 : H_1' \to H_2'$, we have $U_1 \otimes U_2 : H_1 \otimes H_1' \to H_2 \otimes H_2'$. Before we discuss measurement of multi-qubit systems, we introduce another important class of linear operators on Hilbert spaces.

Definition 17 (Positive operator). *A bounded linear operator $f : H \to H$ is said to be positive, if there exists another bounded operator $g : H \to H$ such that $f = g^\dagger \circ g$.*

Definition 18 (Square root (Positive operator)). *Given a Hilbert space H and a positive operator $T : H \to H$, its square root is the unique positive operator $T' : H \to H$ such that, $T'^2 = T$. It is denoted by $\sqrt{T}$.*

Note that all positive operators are self-adjoint. Measurement of multi-qubit quantum systems is a canonical generalization of that of the one qubit system described in Example 11. Measurement with respect to a basis of a multi-qubit system whose state space is a Hilbert space H is simply a set $\{E_i\}_{i\in I}$ of self-adjoint positive semi-definite operators (i.e. $\langle \psi \mid E_i \mid \psi \rangle \geq 0$ for all vectors ψ), for all i. This is known as *Positive Operator Valued Measurement* (POVM), and this approach is central to our work described in Chapter 6. Moreover, this view of measurement adapts well to the next view of quantum mechanics, which happens to be crucial to the work developed in this thesis.

The quantum states we introduced in this section are called *pure states*, and as we have seen Hilbert spaces lie at the heart of the pure state formulation. This view associates vectors from Hilbert spaces, or *rays* to quantum states. However, it may often happen that there is incomplete information on the state of the entire quantum system, or it is a subsystem of a larger quantum system or there is interaction between classical and quantum data. In order to consider all these possibilities, the notion of *mixed states* was introduced. Mixed states can be intuitvely understood to be a probability distribution over pure quantum states. Unlike pure states, they are represented using a *density operator*, ρ ($= \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right|$). Note that different distributions p_i over different pure states ψ_i can give rise to the same density operator ρ.

In order to introduce density operators formally, we first need to generalize the notion of the *trace* of a matrix to operators. Any operator acting on a finite-dimensional Hilbert space,

can be represented via a matrix. Thus it is natural to want to define the notion of the trace of an operator, which can be extended to arbitrary dimensions, and is independent of the choice of a basis for the corresponding Hilbert space (hence independent of the choice of a matrix representation). In order to do so, an operator needs to be *trace class*.

Definition 19 (Trace class (Operator)). *Given a separable Hilbert space H, an orthonormal basis $\{e_i\}_{i\in\mathbb{N}}$ of H and a bounded linear operator $T : H \to H$, T is known as a trace class operator when $Tr(|\ T\ |) < \infty$, where for any operator $A : H \to H$, $Tr(\cdot)$ is defined as the formal sum,*

$$Tr(A) = \Sigma_{i\in\mathbb{N}}\langle A\ e_i, e_i\rangle,$$

and $|T| := \sqrt{T^\dagger \circ T}$.

Definition 20 (Density operator). *A density operator $\rho : H \to H$ is a linear operator which is positive and for which $Tr(\rho) = 1$.*

Density operators over Hilbert spaces are an important subclass of trace-class operators which are used extensively in Schrödinger's view of quantum mechanics. This view of quantum mechanics focuses on states and their unitary time evolution satisfying Schrödinger's differential equation. The Heisenberg view on the other hand is considered to be its "dual", is based on the time evolution of operators acting on quantum systems satisfying Heisenberg's differential equation. The mathematical formulation of Heisenberg's approach relies on von Neumann algebras - the subject of the next section. We adopt the von Neumann view of quantum mechanics to lay down the semantics of programming languages in this thesis.

3.2 von Neumann Algebras

The Hilbert space formulation of quantum mechanics is very useful when modelling pure state quantum computation. However, as we saw in the previous section, mixed states are more general, and hence capture all possible states of a physical system, including those with incomplete information. This is the approach adopted by von Neumann's view of quantum mechanics, which are based on von Neumann algebras. So von Neumann algebras are directly useful for a mixed state approach to quantum mechanics. Moreover, in reality it is far more interesting to model *hybrid* physical systems, i.e, those with both classical and quantum components. Once again, von Neumann algebras are a suitable fit for formalizing such systems, since *commutative* von Neumann algebras can be used to represent purely classical information. Finally, note that while measurement is represented as a set of projectors on a Hilbert space, it fails to be linear when viewed as a map. This limitation is easily overcome in the von Neumann algebra setting, as measurement can indeed be modelled as a linear map between von Neumann algebras.

Before we formally introduce the notion of a von Neumann algebra, we will discuss the structure that motivates this definition. In the previous section we covered some preliminaries on Hilbert spaces and operators on them. In particular bounded operators (Definition 10) on Hilbert spaces are central to von Neumann algebras. In fact, as we shall soon see, for any Hilbert space H, the space $B(H)$ of bounded operators on H and certain kinds of subsets characterize von Neumann algebras.

Definition 21 ($*$-subalgebra). *For a Hilbert space H and a set $M \subseteq B(H)$, M is a $*$-subalgebra of $B(H)$, if it is closed under addition, scalar multiplication, adjoints (Definition 11) and composition.*

Remark 5. Note that a $*$-subalgebra X is said to be *unital* if $1 \in X$, i.e., the identity operator is an element of X.

Now we define the *commutant* of a subset $X \subseteq B(H)$, which we use to define von Neumann algebras.

Definition 22 (Commutant). *Given a Hilbert space H and a set $X \subseteq B(H)$, the commutant X' of X, is defined as:*

$$X' := \{y \in B(H) \mid xy = yx \ \forall x \in X\},$$

i.e., the set of operators y which commute with operators in X w.r.t composition.

Definition 23 (Double commutant). *Given a Hilbert space H and a set $X \subseteq B(H)$, the double commutant X'' of X, is given by $X'' := (X')'$. Note that $X \subseteq X''$.*

Definition 24 (von Neumann algebra [T12] I). *A von Neumann algebra on a Hilbert space H is a $*$-subalgebra M of $B(H)$ such that $M'' = M$.*

Remark 6. It is clear from Definition 24 that a von Neumann algebra is always a unital $*$-subalgebra. Moreover, for any Hilbert space H, a von Neumann algebra $M \subseteq \mathcal{B}(H)$ always contains the zero operator on H.

Example 13. $\mathbb{C}$ and 0 are trivial examples of von Neumann algebras. Furthermore, for any Hilbert space H, $B(H)$ is a von Neumann algebra. For any $n \geq 0$, the matrix algebra $M_n(\mathbb{C})$ of complex $n \times n$ matrices forms a von Neumann algebra.

Now we introduce some topological concepts and other preliminaries necessary to state equivalent definitions of a von Neumann algebra. For a detailed description of topological spaces we refer the reader to [M89]. Different topologies on certain kinds of complex vector spaces also give rise to the structure of a von Neumann algebra and a C*-algebra respectively. We first introduce the operator norm and the operator norm topology, which gives rise to the structure of a C*-algebra.

Definition 25 (Operator norm). *Given normed vector spaces V, W and a bounded linear operator $T : V \to W$, the operator norm of T is defined as,*

$$\|T\| := sup_{\|x\|_V \leq 1} \|Tx\|_W.$$

Definition 26 (Operator norm topology). *Given a Hilbert space H, let $(x_i)_I \subset B(H)$ be a sequence of bounded operators, and let $x \in B(H)$. (x_i) is said to converge to x in the operator norm if,*

$$lim_i \|x_i - x\| = 0.$$

The topology induced by this convergence is called the operator norm topology.

In order to introduce the other topologies relevant to us, we first recall the concepts of a *directed set*, a *net* and that of a limit of a net.

Definition 27 (Directed Set). *A directed set $(D, \preceq)$ is a set D equipped with a reflexive, transitive relation $\preceq$ such that every finite subset has an upper bound.*

Example 14. A simple example of a directed set is the set $\mathbb{N}$ of natural numbers. Given two directed sets $(D_1, \preceq_1)$ and $(D_2, \preceq_2)$, their cartesian product $D_1 \times D_2$ can be made into a directed set using the relation $(d_1, d_2) \preceq (d_1', d_2')$ iff $d_1 \preceq_1 d_1'$ and $d_2 \preceq_2 d_2'$.

Definition 28 (Net). *Given a set X, a net in X is a directed set I called the index set along with a function $\nu : I \to X$. In such a case, I is said to index the net $(\nu_i)_I$.*

Example 15. For any set X, a sequence $\{x_n\}_{n\in\mathbb{N}}$ of elements of X, is a net indexed by $\mathbb{N}$.

Nets in topological spaces can have *limits*, which can be intuitively understood as a generalization of limit points of sequences. A net $(\nu_i)_I$ is said to be *eventually* or *residually* in a set X, if there is an $i \in I$, such that for every $j \geq i$, $\nu_j \in X$. If X is a topological space ([M89]), a point $x \in X$ is said to be a limit of a net $(\nu_i)_I \in X$, if for every open neighbourhood $x \in U \subseteq X$, ν is eventually in U. With all the necessary notions in place, we now introduce the notions of strong operator topology and weak operator topology.

Definition 29 (Strong operator topology). *Given a Hilbert space H, let $\{x_i\}_I \subset B(H)$ be a net of bounded operators, and let $x \in B(H)$. $\{x_i\}$ is said to converge strongly to x if,*

$$lim_i \|(x_i - x)h\| = 0 \qquad \forall h \in H.$$

*The topology induced by this convergence is called the strong operator topology (**SOT**).*

Intuitively, strong convergence is akin to "point-wise convergence" when the Hilbert space H is viewed as a metric space under the norm induced by its inner product. In contrast, convergence in the operator norm, which induces the structure of a C*-algebra on a complex vector space (discussed in the next section), corresponds to "uniform convergence". Note that, uniform convergence is stronger than point-wise convergence. Indeed, every von Neumann algebra is also a C*-algebra as we shall see later. The intuition behind the two notions of convergence becomes clear with an example.

Example 16. Consider the Hilbert space $H := L^2([0,1])$ of square-integrable functions on $[0,1]$, with the inner product given by $\langle f, g\rangle := \int_0^1 f(x)\overline{g(x)}dx$. Now consider, a sequence of linear operators on H, defined as,

$$T_n(f)(x) := \begin{cases} f(x) & \text{if } x \in [0, 1 - \frac{1}{n}] \\ 0 & \text{otherwise} \end{cases}$$

This sequence of bounded linear operators on H converges strongly to the operator $T : H \to H$ defined as,

$$T(f)(x) := \begin{cases} f(x) & x \in [0,1) \\ 0 & x = 1 \end{cases}$$

However, the convergence is not uniform. Now consider the Hilbert space $H := \ell^2(\mathbb{N})$ (Example 4), of square-summable sequences of complex numbers. Consider the sequence of linear operators on H defined as $T_n((x_1, x_2, x_3, \ldots)) := (\frac{x_1}{n}, \frac{x_2}{n}, \ldots)$. This sequence converges uniformly (i.e. in the operator norm) to the constant linear operator defined as $T(x) = 0$.

Definition 30 (Weak operator topology). *Given a Hilbert space H, let $\{x_i\}_I \subset B(H)$ be a net of bounded operators, and let $x \in B(H)$. $\{x_i\}$ is said to converge weakly to x if,*

$$lim_i \langle (x_i - x)h, h'\rangle = 0 \qquad \forall h, h' \in H.$$

*The topology induced by this convergence is called the weak operator topology (**WOT**).*

In order to see that this is indeed a weaker notion of convergence, consider the following example.

Example 17. Consider the Hilbert space $H := \ell^2(\mathbb{Z})$, and the unitary operator $U : H \to H$ defined as $U((\ldots, x_n, x_{n+1}, \ldots)) = (\ldots, x_{n+1}, x_{n+2}, \ldots)$, i.e., it shifts each coordinate by one to the right. Consider the sequence of operators $T_n := U^n$. The sequence $(T_n)_{n\in\mathbb{N}}$ converges weakly to the zero operator, but not strongly. To see this, consider any two elements $h_1, h_2 \in H$. For a $\epsilon > 0$, there is a $N \in \mathbb{N}$, such that $\sum_{n\geq N} \mid h_1(n) \mid^2 < \epsilon^2$ and $\sum_{n<-N} \mid h_2(n) \mid^2 < \epsilon^2$. Then for any $m \geq 2N$, we have,

$$\begin{aligned} \mid \langle T_m(h_1), h_2 \rangle \mid &\leq \sum_{n\in\mathbb{Z}} \mid h_1(n+m) \mid\mid h_2(n) \mid \\ &= \sum_{n<-N} \mid h_1(n+m) \mid\mid h_2(n) \mid + \sum_{n\geq m-N} \mid h_1(n+m) \mid\mid h_2(n) \mid \\ &\leq \epsilon(\|h_1\| + \|h_2\|) \end{aligned}$$

Hence, $(T_n)_{n\in\mathbb{N}}$ converges weakly to 0. However, since U is unitary, $\|T_n(h)\| = \|U^n(h)\| = \|h\|$ for any $h \in H$. Hence, $(T_n)_{n\in\mathbb{N}}$ does not converge strongly to 0.

Now we give an alternative analytic characterization of a von Neumann algebra, as opposed to Definition 24 which is algebraic. These seemingly disparate ways of describing von Neumann algebras, are connected through the *Bicommutant theorem*.

Definition 31 (von Neumann algebra II). *Given a Hilbert space H, a unital $*$-subalgebra $M \subseteq B(H)$ is said to be a von Neumann algebra if it is closed in the weak operator topology.*

Before we introduce the bicommutant theorem, we recall the definition of the closure of a set.

Definition 32 (Closure). *Given a topological space X and a subset $S \subseteq X$, $x \in X$ is said to be a limit point of S, if there exists a net $(x_i)_{i\in I} \subseteq S$, such that x is a limit of $(x_i)_{i\in I}$. The closure of S in X, denoted by $\overline{S}^X$ is S along with all its limit points in X.*

Theorem 1 (Bicommutant theorem). *Let $M \subseteq \mathcal{B}(H)$ be a unital *-subalgebra. Then,*

$$\overline{M}^{WOT} = \overline{M}^{SOT} = M''.$$

Remark 7. A von Neumann algebra M is said to be **commutative**, if $xy = yx$ for all $x, y \in M$.

Example 18. Given a von Neumann algebra M, its **center**, $\mathcal{Z}(M) := M \cap M'$, is a commutative von Neumann algebra.

As for any mathematical object, von Neumann algebras interact through special morphisms between them. Now we introduce some of these maps, before going on to describe other ways of constructing new von Neumann algebras from given ones.

Definition 33 (Involutive map). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$ is said to be involutive if $f(a^*) = f(a)^*$ for all $a \in M_1$.*

Definition 34 (Multiplicative map). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$ is said to be multiplicative if $f(ab) = f(a)f(b)$ for all $a, b \in M_1$.*

Definition 35 ($*$-morphism). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$ is said to be a $*$-morphism if it is both involutive and multiplicative.*

Example 19. Consider the finite dimensional Hilbert space $\mathbb{C}^n$, for some n. In this case the space of bounded operators on H is isomorphic to $\mathcal{M}_n(\mathbb{C})$, the space of $n \times n$ matrix algebras over $\mathbb{C}$. Consider the map $\Phi : \mathcal{M}_n(\mathbb{C}) \to \mathcal{M}_n(\mathbb{C})$, defined as $\Phi(x) := u^*xu$ for a fixed unitary $u \in \mathcal{M}_n(\mathbb{C})$. Clearly Φ is a $*$-morphism.

Definition 36 (Unital map). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$ is said to be unital if $f(1_{M_1}) = 1_{M_2}$.*

We now define a few special kinds of elements of von Neumann algebras before we go on to introduce various notions of maps between von Neumann algebras.

Definition 37 (Self adjoint element). *Given a von Neumann algebra M, an element $a \in M$ is said to be self-adjoint if $a^* = a$.*

Definition 38 (Positive element). *Given a Hilbert space H and a von Neumann algebra $M \subseteq \mathcal{B}(H)$, an operator $x \in M$ is said to be positive if $\langle h, xh\rangle \geq 0$ for all $h \in H$.*

Consider the partially ordered set $M_{sa} := \{x \in M \mid x = x^*\}$ of self-adjoint elements of a von Neumann algebra M, with the Löwner order, i.e., $x \leq y$ iff $y - x$ is a positive operator. Then an operator $x \in M$ is positive iff $x \geq 0_M$. Note that every positive element is in the set M_{sa}, since a positive element can always be decomposed as yy^* for some operator $y \in M$. This justifies the use of the terminology for positive elements.

Example 20. An important poset in any von Neumann algebra M is the set $[0,1]_M := \{x \in M \mid 0_M \leq x \leq 1_M\}$. This is often referred to as the *unit interval poset.*

The unit interval poset defined in Example 20, enjoys a stronger property than that of being a poset. It is a *directed complete partial order*, whose definition we formally recall below.

Definition 39 (Directed complete partial order). *A directed complete partial order $(P, \leq)$ is a poset in which every directed subset (Definition 27) $D \subseteq P$ has a join, i.e. sup $D \in P$ exists.*

The notion of positivity generalizes to maps between von Neumann algebras, which requires that a linear map between von Neumann algebras sends positive elements to positive elements.

Definition 40 (Positive map). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$ is said to be positive if it maps positive elements to positive elements.*

However, in general it turns out as we shall soon see, certain kinds of positive maps need not always exist. Hence, we define a slightly stronger variant, called *complete positivity* which is crucial to the construction of tensor products of von Neumann algebras. For any von Neumann algebra M, we denote the matrix algebra of $n \times n$ matrices with entries from M by $\mathcal{M}_n(M)$. Note that any linear map $f : M_1 \to M_2$ between two von Neumann algebras, can be canonically extended to a map $\mathcal{M}_n(f) : \mathcal{M}_n(M_1) \to \mathcal{M}_n(M_2)$ in the obvious way, i.e. $\mathcal{M}_n(f)((a_{ij})_{1\leq i,j\leq n}) := (f(a_{ij}))_{1\leq i,j\leq n}$.

Definition 41 (Completely positive I). *Given von Neumann algebras M_1, M_2, a positive linear map $f : M_1 \to M_2$, is said to be completely positive if, for all $n \in \mathbb{N}$, the map $\mathcal{M}_n(f) : \mathcal{M}_n(M_1) \to \mathcal{M}_n(M_2)$ is positive.*

Definition 42 (Completely positive II). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$, is said to be completely positive if, $\sum_{i,j} y_i^* f(x_i^* x_j) y_j$ is positive for all $x_1, x_2 \ldots x_n \in M_1$, and $y_1, y_2 \ldots y_n \in M_2$.*

Remark 8. Note that any unital $*$-morphism is also a completely positive unital map.

Example 21. Consider the transpose map acting on $\mathcal{M}_n(\mathbb{C})$. This map is positive for all $n \geq 1$. Moreover, it is also completely positive for $n = 1$. However, for $n \geq 2$, the transpose fails to be completely positive.

Note that for any von Neumann algebra M, $1_M \in M_{sa}$. This gives rise to the next notion of maps between von Neumann algebras.

Definition 43 (Sub-unital map). *Given von Neumann algebras M_1, M_2, a linear map $f : M_1 \to M_2$ is said to be sub-unital if $f(1_{M_1}) \leq 1_{M_2}$.*

As we have seen so far, von Neumann algebras enjoy an algebraic as well as an analytic structure. The maps we defined so far, interact well with the algebraic nature of the construction. Since, von Neumann algebras also happen to be topological spaces, a natural question that arises is that of continuity of maps. In order to talk about "continuous" maps on von Neumann algebras, we first recall the definition of *Scott continuity*.

Definition 44 (Scott continuous map). *Given two posets P, Q, a map $f : P \to Q$ is said to be Scott-continuous if it is monotonic and it preserves all directed suprema, i.e., for every directed subset $D \subseteq P$, its image $f(D)$ has a join in Q, and this join is the image of the supremum of D in P under f (i.e. $f(\bigvee D) = \bigvee f(D)$).*

Example 22. A trivial example of a Scott continuous map is the identity function on any dcpo. For a non-trivial example, consider the poset $P := [0, \infty] \cup \perp$, where $\perp < x$, for all $x \in [0, \infty]$. Then the function $f : P \to P$ defined as,

$$f(x) = \begin{cases} 0 & x = \perp \\ x + 1 & x \in [0, \infty] \end{cases}$$

is Scott continuous. This is clear since f is monotonous, and clearly preserves suprema.

Definition 45 (Normal map I). *Given two von Neumann algebras M_1, M_2, a positive subunital linear map $f : M_1 \to M_2$, is said to be normal if its restriction $f\big|_{[0,1]} : [0,1]_{M_1} \to [0,1]_{M_2}$ is Scott continuous.*

Example 23. Trace preserving maps on $B(H)$ play a very significant role in the Schrödinger picture of quantum mechanics. In this example we give an instance of a trace preserving map which is also a normal map. For a separable n-dimensional Hilbert space H, define the map $\Phi : B(H) \to B(H)$, as $\Phi(T) = Tr(T)I/n$. This map sends any operator to a scalar multiple of the identity operator on H. It is clearly a linear unital positive operator. To see that it is normal note that Tr is a normal map, i.e. $Tr(\bigvee T_i) = \bigvee Tr(T_i)$ for an increasing net of operators (T_i).

Like the center, there are other ways of constructing new von Neumann algebras from given ones. They corrrespond to the direct sum and the tensor product constructions defined for Hilbert spaces. In order to formally introduce them, first we introduce the notion of an *isometric embedding*. Recall from §3.1, that an isometry (Definition 13), for Hilbert spaces is a inner-product preserving linear operator. Extending this notion to von Neumann algebras, an isometric embedding defined on von Neumann algebras is an injective $*$-morphism. Given n Hilbert spaces $H_1, H_2 \ldots, H_n$, for each $1 \leq i \leq n$, we define an isometric embedding $\pi_i : B(H_i) \to B(H_1 \oplus \cdots \oplus H_n)$ by

$$\pi_i(x)(h_1, h_2, \ldots, h_n) = (0, \ldots, x(h_i), 0, \ldots 0) \qquad (h_1, \ldots, h_n) \in H_1 \oplus \cdots \oplus H_n$$

The maps π_i can also be seen as square matrices with 1 in the (i, i)-th entry and zeroes everywhere else. Note that this definition extends for any countable set $\{H_i\}_I$. We use these maps to define the *direct-sum* of von Neumann algebras.

Definition 46 (Direct sum (von Neumann algebras)). *Given von Neumann algebras $M_i \subseteq B(H_i)$ for $1 \leq i \leq n$, the direct sum of $M_1, M_2, \ldots, M_n$ is*

$$M_1 \oplus \cdots \oplus M_n := span\{\pi_i(x) \mid 1 \leq i \leq n, x \in M_i\}$$

It is denoted by $\bigoplus_{i=1}^n M_i$.

Note that we have $\mathcal{Z}(\bigoplus_{i=1}^n M_i) = \bigoplus_{i=1}^n \mathcal{Z}(M_i)$.

Remark 9. The above definition of direct sum of von Neumann algebras extends from finitely many von Neumann algebras to a countable set of von Neumann algebras. In the latter case, we define the set $span\{\pi_i(x) \mid 1 \leq i \leq n, x \in M_i\}$, to be the closure of the set of finite linear combinations of $\pi_i(x)$ in the weak operator topology. This is precisely set $\{(h_i)_I \in \bigoplus_I M_i \mid sup_I \|h_i\| < \infty\}$.

For Hilbert spaces $H_1, \ldots, H_n$, $\bigoplus_{i=1}^n B(H_i)$ is an obvious example of the direct sum of von Neumann algebras.

Example 24. Consider a von Neumann algebra M such that there are projections $\pi_i \in \mathcal{Z}(M)$, with $\sum_i \pi_i = 1_M$. Then $\pi_i(M) \subseteq M$ is a von Neumann algebra. Furthermore, in this case we have that there is a $*$-isomorphism $\Phi : M \to \bigoplus_i \pi_i(M)$.

Another important way of constructing von Neumann algebras is through the tensor product. In order to do so, for Hilbert spaces $H_1, \ldots H_n$, we first define an action of the algebraic tensor product $B(H_1) \otimes \cdots \otimes B(H_n)$ on $H_1 \otimes \cdots \otimes H_n$, as,

$$x_1 \otimes \cdots \otimes x_n(h_1 \otimes \cdots \otimes h_n) := x_1(h_1) \otimes \cdots \otimes x_n(h_n) \qquad \forall h_i \in H_i$$

This gives us a canonical inclusion $B(H_1) \otimes \cdots \otimes B(H_n) \subseteq B(H_1 \otimes \cdots \otimes H_n)$, thus giving us the following definition.

Definition 47 (Tensor product (von Neumann algebras)). *Given von Neumann algebras $M_i \subseteq B(H_i)$, for Hilbert spaces H_i for $1 \leq i \leq n$. The tensor product of $M_1, \ldots, M_n$ is given as,*

$$M_1 \overline{\otimes} \cdots \overline{\otimes} M_n := (M_1 \otimes \cdots \otimes M_n)'' \subseteq B(H_1 \otimes \cdots \otimes H_n).$$

It is often denoted by $\overline{\bigotimes}_{i=1}^n M_i$.

Remark 10. The above formulation of the tensor product of von Neumann algebras is consistent with its algebraic formulation. An analytic definition would be the closure of the set $span\{x_1 \otimes \cdots \otimes x_n \mid x_i \in M_i\}$ in the weak operator topology. Their equivalence is a straightforward consequence of the bicommutant theorem.

Remark 11. The tensor product construction gives another motivation for the notion of complete positivity (Definition 41, 42). To see this note that given two normal, positive linear maps $f : M_1 \to M_2, g : M_1' \to M_2'$, there need not always exist a normal, positive linear map $f \otimes g$. However, if we strengthen the condition to complete positivity, then such a map is always guaranteed to exist, i.e. for normal, completely positive maps f, g, there is a normal completely positive map $f \otimes g$.

Example 25. Consider the Hilbert spaces $\mathbb{C}^2, \mathbb{C}^3$. In this case, the set of bounded operators coincides with $M_1 := \mathcal{M}_2(\mathbb{C}), M_2 := \mathcal{M}_3(\mathbb{C})$ respectively. We have the following $*$-isomorphisms:

$$M_1 \overline{\otimes} M_2 \cong \mathcal{M}_2(\mathbb{C}) \otimes \mathcal{M}_3(\mathbb{C}) \cong \mathcal{M}_6(\mathbb{C})$$

Two important instances of von Neumann algebras, that are central to this work are $B(H)$ for a separable Hilbert space H and matrix algebras. A n-dimensional matrix algebra over a set X is the space of $n \times n$ matrices with entries from X. Since, any finite dumensional Hilbert space is isomorphic to $\mathbb{C}^n$, for some n, $B(H)$ in these cases coincides with $\mathcal{M}_n(\mathbb{C})$. Even though $\mathcal{M}_n(\mathbb{C})$ is a basic example of a von Neumann algebra, it is great for building intuition about them. The *trace functional* defined on matrix algebras, is central to the notion of a class of operators in $B(H)$, known as the *trace class operators*. These are the basis of the Schrödinger view of quantum theory, where quantum states are elements of a space of trace class operators, and operations on them are represented by *completely positive trace preserving* (cptp) maps. There is a bijective correspondence between the set of cptp maps between trace class operators, and *normal completely positive unital* (ncpu) maps between bounded operators. This correspondence is at the heart of the Schrödinger- Heisenberg duality of quantum mechanics, which we treat formally in the next section. We employ the Heisenberg view of quantum mechianics for the works described in this thesis, which dictates that quantum observables are described by elements in the space of bounded operators, and operations on them are described by ncpu maps.

In fact, direct sum of matrix algebras characterize the smaller class of finite dimensional C*-algebras, which are the subject of the next section.

3.3 C^*-algebras

As we saw in the previous section, changing the topology on spaces gave rise to different algebraic structures on vector spaces. We saw that a von Neumann algebra is a $*-$unital subalgebra of $B(H)$ closed with respect to the weak operator topology. Similarly, C*-algebras arise from the operator norm topology on $B(H)$. The formal definition of a C*-algebra is given below.

Definition 48 (C*-algebra). *A (unital) C*-algebra is a complex vector space A, endowed with*

- *a binary operation, called multiplication, which is associative and linear in both the coordinates,*
- *an element, 1 which is the multiplicative unit, i.e. $1 \cdot a = a \cdot 1 = a$ for all $a \in A$, referred to as the unit,*
- *a unary operation $(\cdot)^*$, called involution such that $(a^*)^* = a$, $(ab)^* = b^*a^*$, $(\lambda a)^* = \bar{\lambda}a^*$, and $(a+b)^* = a^* + b^*$ for all $\lambda \in \mathbb{C}$, $a, b \in A$,*
- *a complete norm $\|\cdot\|$ such that $\|ab\| \leq \|a\|\|b\|$ for all $a, b \in A$, and*

$$\|a^*a\| = \|a\|^2$$

holds; often referred to as the C-identity.*

Remark 12. A C*-algebra is said to be commutative if the multiplication operation is commutative.

Since every von Neumann algebra is clearly a C*-algebra, the examples we saw in the previous section are all important instances of C*-algebras as well.

Example 26. The C*-algebra structure on $B(H)$, for a Hilbert space H, can be given by defining the multiplication to be operator composition, the involution to be taking adjoints, and the unit to be the identity operator. The norm used is the operator norm (Definition 25).

Example 27. For any $n \in \mathbb{N}$, $\mathcal{M}_n(\mathbb{C})$, the matrix algebra of $n \times n$ matrices with complex entries is a C*-algebra. Multiplication is given by the usual matrix multiplication, the involution is defined by taking the transpose of the matrix, with complex conjugate entries (usually referred to as the *adjoint* of the matrix), and the unit is the $n \times n$ identity matrix. The norm in this case, inspired from the operator norm is given by,

$$\|M\| := sup\{\|Mx\| \mid x \in \mathbb{C}^n, \|x\| \leq 1\}.$$

Here $\|\cdot\|$ represents the standard Euclidean norm defined on $\mathbb{C}^n$.

A natural question that would arise, is what is the exact relation between von Neumann algebras and C*-algebras. Are all C*-algebras von Neumann algebras? It doesn't seem to be the case, but what is an example? It turns out the example is a very important construction that is veyr important to the matter covered in this thesis. We defer the example a but, and provide it once we formally introduce direct sums of C*-algebras.

When can we guarantee that a C*-algebra is also a von Neumann algebra? The conditions for this to hold are a result due to Sakai. In order to formally state this result, we first must introduce the notion of a W^*-algebra and some terminology related to *Banach spaces*. This will also allow us to formally introduce the Schrödinger-Heisenberg duality of quantum mechanics. Towards this, first recall the notion of the *dual* of a vector space, and its canonical extension to Banach spaces.

Definition 49 (Dual (Vector space)). *Given a vector space V over a field $\mathbb{F}$, the set $V^* := Hom(V, \mathbb{F})$ of linear functionals on V can be given a vector space structure, by defining each operation pointwise. V^* is known as the dual vector space of V.*

Example 28. For the vector space $\mathbb{C}$, its dual is isomorphic to itself, i.e. $\mathbb{C}^* \cong \mathbb{C}$.

Remark 13. For any finite dimensional vector space V, we have that $dim(V) = dim(V^*)$, and hence they are isomorphic. However, in the case of infinite dimensions, V^* is strictly larger than V.

This notion of the dual of a vector space generalizes to Banach spaces as well. However, they are not the same, with a key difference reflecting the additional topological structure of a Banach space. Hence, the dual of a vector space is often referred to as an algebraic dual, whereas that of a Banach space is referred to as a topological dual.

Definition 50 (Dual (Banach space)). *Given a Banach space X over a field $\mathbb{F}$, the set $X^* := \{f : X \to \mathbb{F} \mid f$ is linear and continuous.$\}$ of linear continuos functionals on X can be given a Banach space structure, by defining each operation pointwise and equipping it with the sup norm (i.e. $\|f\| = sup_{\|x\| \leq 1} \mid f(x) \mid$). X^* is known as the dual Banach space of X.*

Example 29. A very important example of a Banach space and its topological dual is given by the space $X := \ell^1(S)$, the set of absolutely summable sequences of complex numbers indexed by a set S, with $X^* := \ell^\infty(S)$, the set of bounded sequences of complex numbers indexed by the set S. The algebraic dual of X turns out to be much larger than $\ell^\infty(S)$, its topological dual.

Conversely, given a normed vector space X, it is useful to ask whether it is the dual of some Banach space B. If this turns out to be the case, we refere to B as the predual of X, this is denoted by $B = X_*$. We now define a W*-algebra.

Definition 51 (W*-algebra). *A W^*-algebra is a C*-algebra that is also the dual of some Banach space B (Definition 5), which is known as its predual, i.e $M = B^*$ and $B = M_*$.*

Example 30. For any Hilbert space H, the von Neumann algebra $B(H)$ is a W*-algebra, with its predual given by $T(H)$ - the space of all trace-class (Definition 19) operators on H. $T(H)$ is a vector space with operations defined pointwise, and it becomes a Banach space under the trace norm. We have that $T(H)^* \cong B(H)$, where $\cong$ is an isometric isomorphism. Any continuous linear functional on $T(H)$ arises as a trace functional of an operator obtained by composition with an element of $B(H)$, i.e., for any $\Phi : T(H) \to \mathbb{C}$, we have $\Phi(\rho) := Tr(x \cdot \rho)$ for a fixed $x \in B(H)$ and for all $\rho \in T(H)$. A formal treatment of this isomorphism is given in Section 1,[S05]. This result is at the heart of the Schrödinger-Heisenberg duality of quantum mechanics.

Furthermore, this relationship between a Banach space and a W*-algebra also gives rise to a topology on the W*-algebra, which is known as the *weak* topology.*

Definition 52 (Weak*-topology). *Given a Banach space B, let $\{x_i\}_{i\in I} \subset B^*$ be a net of bounded linear functionals on B and $x \in B^*$. $\{x_i\}_{i\in I}$ is said to converge to x weak*-ly, if $x_i(b) \to x(b)$ for all $b \in B$. The topology induced on B^* from this convergence is known as the Weak* topology.*

We now formally state Sakai's result, which links W*-algebras to von Neumann algebras.

Theorem 2 (Sakai's Theorem [S212]). *A C*-algebra A, is isometrically $*$-isomorphic to a von Neumann algebra iff A is a W*-algebra.*

Example 31. Given a net of operators $\{x_i\}_{i\in I} \in B(H)$, and an operator $x \in B(H)$, $x_i \to x$ weak*-ly iff $Tr(\rho x_i) \to Tr(\rho x)$ for all $\rho \in T(H)$. Clearly, this convergence does not coincide with that in the strong, weak or operator norm topology.

Remark 14. Note that as a consequence of Theorem 2, every von Neumann algebra is a W*-algebra.

When viewing von Neumann algebras as W*-algebras, normality of maps translates to continuity with respect to the weak*-topology.

Definition 53 (Normal map II). *Given two W*-algebras M_1, M_2, a bounded linear map $f : M_1 \to M_2$ is said to be normal if it is continuous with respect to the weak*-topology.*

Remark 15. The duality that we saw between $T(H)$ and $B(H)$ in Example 30 is the core of the mathematical equivalence of the Schrödinger-Heisenberg dual view of quantum mechanics. Normal maps on $B(H)$ correspond to maps on $T(H)$ via the Banach space duality laid out in Example 30, i.e., for a normal map $\Phi : B(H) \to B(H)$, and operators $\rho \in T(H), x \in B(H)$, we have $\langle \Phi_*(\rho), x\rangle = \langle \rho, \Phi(x)\rangle$. Φ is a *normal completely positive unital* (ncpu) map iff Φ_* (the map corresponding to Φ under the isomorphism in Example 30) is a *completely positive trace preserving* (cptp) map. Thus ncpu maps are in bijective correspondence with cptp maps. The Schrödinger view of quantum mechanics uses $T(H)$ as the state space, i.e, quantum states are viewed as density operators, and operations on them are modelled as *completely positive trace preserving* (cptp) maps. In the Heisenberg view, quantum observables are represented using self-adjoint operators with ncpu maps representing quantum channels.

As is the case with von Neumann algebras, certain elements of C*-algebras satisfy special properties. The properties resemble those presented in §3.2, and hence we present all of them in a single definition for brevity.

Definition 54. *Let A be a C*-algebra.*

- *An element $a \in A$ is self-adjoint if $a = a^*$.*

- *An element $a \in A$ is normal if $a^*a = aa^*$.*
- *An element $p \in A$ is a projection if $p^2 = p = p^*$; i.e. p is self-adjoint and idempotent.*
- *An element $u \in A$ is unitary if $u^*u = 1 = uu^*$.*
- *An element $a \in A$ is invertible, if there is a $b \in A$ such that $ab = ba = 1$; the set of invertible elements of A is denoted by A^{-1}.*
- *An element $a \in A$ is positive if there is an element $b \in A$ such that $a = b^*b$.*

Additionally, for any element $a \in A$ of a C*-algebra A, there is a set of complex numbers associated to it called the *spectrum* of a, often denoted by $\sigma(a)$. This set can be intuitively understood to be the generalization of eigenvalues of a matrix. Formally, it is defined in the next definition.

Definition 55 (Spectrum). *Given a C*-algebra A, and an element $a \in A$, the spectrum $\sigma(a)$ of a, is defined as,*

$$\sigma(a) := \{\lambda \in \mathbb{C} \mid (a - \lambda \cdot 1) \notin A^{-1}\}.$$

The spectral radius, denoted by $r(a)$ is defined as $r(a) := sup_{\lambda \in \sigma(a)} \mid \lambda \mid$.

Remark 16. The spectral radius is well-defined since it is always non-empty and bounded for any element a of a C*-algebra A.

Example 32. For a matrix algebra, the spectrum $\sigma(M)$ of any matrix M is just the set of its eigenvalues with their multiplicities.

Example 33. For the C*-algebra $C[0,1]$, discussed in Example 40, the spectrum $\sigma(f)$ of a function f coincides with its range, i.e. $\sigma(f) = range(f)$.

Remark 17. Some of the elements defined in Definition 54 can also be characterized in terms of what their spectrum looks like. For instance, the spectrum of any self-adjoint element is always a subset of $\mathbb{R}$. Similarly, the norm of any normal element coincides with its spectral radius. We will soon see a similar characterization of positive elements.

Similar to the case of von Neumann algebras, positive elements play a very important role in the theory of C*-algebras. They are a generalization of positive operators on a Hilbert space. The next lemma presents frequently used alternative characterizations of a positive element.

Lemma 1. *Let A be a C*-algebra and $a \in A$. The following are equivalent:*

- *a is positive;*
- *$\exists t \in \mathbb{R}$, s.t., $\|a - t \cdot 1\| \leq t$;*
- *a is self-adjoint and $\sigma(a) \subseteq [0, \infty)$;*
- *$a = b^2$ for a positive, self-adjoint $b \in A$.*

Example 34. For a given $n \in \mathbb{N}$, the set of positive elements of $\mathcal{M}_n(\mathbb{C})$ are matrices M, whose eigenvalues are real and for which $M = M^*$.

Example 35. For the C*-algebra $C[0,1]$, an element $f \in C[0,1]$ is positive iff $f(x) \geq 0$ for all $x \in [0,1]$.

C*-algebras interact through special maps, most of which are inspired from those defined for von Neumann algebras. We define all the similar notions for C*-algebras, and again combine them into one definition for brevity.

Definition 56. *Given A, B C*-algebras, a linear map $f : A \to B$ is called,*

- *multiplicative, if $f(ab) = f(a)f(b)$;*
- *involutive, if $f(a^*) = f(a)^*$;*
- *unital, if $f(1_A) = 1_B$;*
- *sub-unital if $1_b - f(1_A)$ is a positive element of B;*
- *positive, if $f(a)$ is positive for every positive $a \in A$;*
- *completely positive, if $\sum_{i,j} b_i^* f(a_i^* a_j) b_j$ is positive for all $a_1, a_2, \ldots a_n \in A$, and $b_1, b_2, \ldots, b_n \in B$;*
- *$*$-homomorphism if it is multiplicative and involutive;*
- *$*$-isomorphism if it is a bijective $*$-homomorphism.*

Note that the notion of normal maps is generally defined only for von Neumann algebras, and not for C*-algebras. The important maps for C*-algebras, are completely positive maps.

Example 36. For the matrix algebras $\mathcal{M}_n(\mathbb{C})$ and $\mathcal{M}_{n+1}(\mathbb{C})$, the inclusion map $i : \mathcal{M}_n(\mathbb{C}) \to \mathcal{M}_{n+1}(\mathbb{C})$, defined as,

$$i(A) := \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix}$$

is a $*$-homomorphism.

Example 37. For the C*-algebra $C[0,1]$, the linear operator $\Phi : C[0,1] \to C[0,1]$ defined as $\Phi(f)(x) := f(x^2)$, is clearly a $*$-homomorphism.

In general, a multiplicative linear map that is unital and bounded, is also involutive and hence a $*$-homomorphism. An important example of a multiplicative linear map which is not involutive is given in the next example.

Example 38. Consider the matrix algebra $\mathcal{M}_n(\mathbb{C})$. The linear operator $\Phi : \mathcal{M}_n(\mathbb{C}) \to \mathcal{M}_n(\mathbb{C})$ defined by $\Phi(a) := a^T$, is involutive and linear but not multiplicative.

New C*-algebras can be constructed from given ones, through the direct sum and the tensor product construction like von Neumann algebras. While, the underlying set structure in both the cases, bears resemblance to the algebraic construction we saw in §3.1 and in §3.2, it is the choice of the norm which crucially decides whether the resulting construction is a C*-algebra. This will become clear as we introduce both these notions for C*-algebras. We first define the direct sum.

Definition 57 (Direct sum (C*-algebras)). *Given C*-algebras $M_1, \ldots, M_n$, their direct sum $M_1 \oplus \cdots \oplus M_n$ is defined as,*

$$M_1 \oplus \cdots \oplus M_n := \{(a_1, \ldots, a_n) \mid \ a_i \in M_i\}.$$

The C-algebra structure is given as follows:*

- *Multiplication is defined pointwise, i.e., $(a_i) \cdot (b_i) = (a_i \cdot b_i)$.*
- *Involution is defined pointwise, i.e., $(a_i)^* = (a_i^*)$.*
- *Multiplicative unit is given by the tuple (1_i), where 1_i is the unit of M_i.*
- *The norm is given by the sup norm, i.e. $\|(a_i)\| := sup_i \|a_i\|_{M_i}$.*

It is easy to check that the norm defined above is complete, and satisfies the C*-identity. To highlight the importance of the choice of a norm, note that the same algebraic set, can be endowed with another norm, given by $\|(x_i)\| := \sum_i \|x_i\|_{M_i}$. However, this norm fails to satisfy the C*-identity. In fact the sup norm is the only norm which makes the above construction into a C*-algebra. However, for the case of the tensor product, such a norm is not unique as we shall soon see. Note that the above definition can be extended to the direct sum of arbitrarily many C*-algebras.

Example 39. Given countably many C*-algebras $\{A_n\}$, the algebra $\bigoplus_{n=1}^{\infty} A_n := \{(a_n) \mid a_n \in A_n,\ lim_{n\to\infty}\|a_n\| = 0,\ sup_n\|a_n\| < \infty\}$ is a C*-algebra, with the norm given by the sup norm. Another C8-algebra, that can be constructed from such a countable collection is the algebra $\prod_{n=1}^{\infty} A_n := \{(a_n) \mid a_n \in A_n,\ sup_n\|a_n\| < \infty\}$.

Example 40. Following the example above, when we take A_n to be the matrix algebra $\mathcal{M}_n(\mathbb{C})$, the C*-algebra $\mathcal{M} := \bigoplus_{n=1}^{\infty} \mathcal{M}_n(\mathbb{C})$, coinciding with the construction in Example 39 is a C*-algebra, but *not* a von Neumann algebra since it is not closed in the weak operator topology when viewed as a *-subalgebra of $B(H)$ for $H := \bigoplus_{n=1}^{\infty} \mathbb{C}^n$. It turns out that the closure of this C*-algebra under **WOT** is the von Neumann algebra $\mathcal{A} := \prod_{n=1}^{\infty} \mathcal{M}_n(\mathbb{C})$, which is is a strictly larger set than $\bigoplus_{n=1}^{\infty} \mathcal{M}_n(\mathbb{C})$. To see that $\mathcal{M}$ is not closed in **WOT**, take any $(M_n) \in \mathcal{A}$. Construct a sequence $\{x_k\} \in \mathcal{M}$ as follows:

$$x_k := (M_1, M_2, \ldots, M_k, 0, 0 \ldots 0)$$

Clearly, $\{x_k\}_{k\geq 1} \in \mathcal{M}$. Moreover, $x_k \to (M_n) \notin \mathcal{M}$.

It is natural to build intuition for a new mathematical structure, by understanding how it relates to well-known ones. This rule applies particularly, well to C*-algebras, through their "resemblance" to matrix algebras and $B(H)$, the space of bounded operators on a Hilbert space H. We now focus on making this proximity precise. The pervasiveness of matrix algebras as instances of C*-algebras cannot be highlighted enough. In fact, in finite dimensions, these are the only non-trivial C*-algebras upto direct sums as stated by the next theorem.

Theorem 3. *Given a finite-dimensional C*-algebra A, there exist positive integers k and $n_1, n_2, \ldots, n_k$ such that*

$$A \cong \bigoplus_{i=1}^{k} \mathcal{M}_{n_i}(\mathbb{C}).$$

The isomorphism in this case is a unital $*$-homomorphism, and k is unique. Moreover, $n_1, n_2 \ldots, n_k$ are also unique upto permutation.

The above theorem greatly simplifies finite-dimensional C*-algebras for us. In general, any arbitrary C*-algebra can be studied through its interaction with the space of bounded operators on a Hilbert space, through *representations*.

Definition 58 (Representation of a C*-algebra). *Given a C*-algebra A and a Hilbert space H, a representation of A on H is a pair (π, H), where $\pi : A \to B(H)$ is a $*$-homomorphism.*

The idea of "sameness" of two representations (π_1, H_1) and (π_2, H_2) of a C*-algebra A, is captured through the notion of unitary equivalence.

Definition 59 (Unitary equivalence). *Given a C*-algebra A, two representations (π_1, H_1) and (π_2, H_2) are said to be unitarily equivalent, if there is a unitary $u : H_1 \to H_2$ such that $\pi_2(a) = u\pi_2(a)u^*$ for all $a \in A$. This equivalence is denoted by $(\pi_1, H_1) \sim_u (\pi_2, H_2)$ or $\pi_1 \sim_u \pi_2$.*

Example 41. Given a matrix algebra $\mathcal{M}_n(\mathbb{C})$, consider the $*$-homomorphism $\pi : \mathcal{M}_n(\mathbb{C}) \to B(\mathbb{C}^n)$ defined as $\pi(a)(x) := a \cdot x$. It is a representation of the C*-algebra $\mathcal{M}_n(\mathbb{C})$ on the Hilbert space $\mathbb{C}^n$.

Example 42. The multiplication operator $\pi : C[0,1] \to B(L^2([0,1]))$ defined in Example 40, as $\pi(f)(x) := f \cdot x$ is a representation of the C*-algebra $C[0,1]$ on the Hilbert space $L^2([0,1])$.

Example 43. Consider the matrix algebra $\mathcal{M}_n(\mathbb{C})$, and the representation defined in Example 41. For any unitary matrix $U : \mathbb{C}^n \to \mathbb{C}^n$, the $*$-homomorphism defined as $\pi_2(a) := U\pi_1(a)U^*$, is unitarily equivalent to π_1 by construction.

Definition 60. *Given a C*-algebra A its representation (π, H) is said to be faithful if the $*$-homomorphism $\pi : A \to B(H)$ is injective and norm-preserving.*

The next theorem, known as the *Gelfand-Naimark* theorem ensures that faithful representations of C*-algebras always exist. This is a key fact in constructing the tensor product of C*-algebras.

Theorem 4 (Gelfand Naimark). *Every C*-algebra A is $*$-isomorphic to a closed (wrt. operator norm topology) $*$-subalgebra of $B(H)$ for some Hilbert space H,*

$$A \cong \pi(A) \subseteq B(H).$$

Here, $\pi : A \to B(H)$ is a faithful representation of A on H.

We now introduce the final construction of this section. The tensor product of C*-algebras differs from that of von Neumann algebras, in the sense that the construction is not unique. Precisely, this means given two C*-algebras, there are multiple ways to obtain a C*-algebra that resembles the algebraic structure of a tensor product space. In order to describe all these ways, we first describe the underlying $*$-algebra which is common to all constructions, and then define each separately with their norms.

Definition 61 (Tensor product ($*$-algebras)). *Given two $*$-algebras (unital) A, B their tensor product $A \odot B$ defined as,*

$$A \odot B := span\{a \otimes b \mid a \in A,\ b \in B\}.$$

$A \odot B$ is given a $*$-algebra structure as follows:

- *Multiplication is defined pointwise, i.e., $(a_1 \otimes b_1) \cdot (a_2 \otimes b_2) = (a_1 \cdot a_2) \otimes (b_1 \cdot b_2)$.*
- *The multiplicative unit is given by $1_A \otimes 1_B$.*
- *Involution is defined pointwise, i.e., $(a \otimes b)^* = a^* \otimes b^*$.*

The $*$-algebra $A \odot B$ defined above, can now be given a structure of a C*-algebra, by defining a norm which satisfies the C*- identity, and then completing it. We now define two of the most common ways of defining a norm to obtain a tensor product of C*-algebras.

Definition 62 (Minimal (spatial) tensor product). *Given two faithful representations $\pi_A : A \to B(H)$ and $\pi_B : B \to B(K)$, define a representation $\pi : A \odot B \to B(H \otimes K)$ of $A \odot B$ on $H \otimes K$ as,*

$$\pi(a \otimes b) := \pi_A(a) \otimes \pi_B(b)$$

and extend linearly. The completion under the norm given by $\|x\|_{min} := \|\pi(x)\|_{B(H\otimes K)}$ makes $A \odot B$ into a C-algebra. This C*-algebra is often denoted by $A \otimes_{min} B$.*

Remark 18. The Gelfand-Naimark theorem guarantees the existence of faithful representations, hence the construction in Definition 62 is always valid. To see that it is well-defined, note that the norm induced is independent of the choice of these faithful representations (faithful representations are isometric).

Definition 63 (Maximal tensor product). *Given two C*-algebras A, B, the $*$-algebra $A \odot B$ can be given the structure of a C*-algebra, by completing it under the norm,*

$$\|x\|_{max} := sup\{\|\pi(x)\| \mid \pi : A \odot B \to B(H) \text{ is a representation}\}.$$

This C-algebra is often denoted by $A \otimes_{max} B$.*

Remark 19. Once again the construction in Definition 63 is well-defined, because the set is non-empty due to Gelfand-Naimark theorem, and bounded. Further note that the two norms in Definition 62 and Definition 63 coincide for finite-dimensional C*-algebras.

Example 44. Consider the matrix algebras, $M_1 := \mathcal{M}_2(\mathbb{C})$ and $M_2 := \mathcal{M}_3(\mathbb{C})$. The matrix algebra $M_1 \otimes M_2$ is a finite-dimensional C*-algebra isomorphic to $\mathcal{M}_6(\mathbb{C})$.

3.4 The Simply Typed Lambda Calculus

Now we will give a brief introduction to the lambda calculus and its semantics. We do this, to motivate the next definition that we introduce, and to highlight its importance in theoretical computer science. The lambda calculus is a system for expressing computations, which is based on the mathematical notion of functions and their application. There are various versions of the lambda calculus, both *typed* and *untyped*. The orginal version is untyped, and expressive enough to support all computable functions. However, since it is untyped it allows formation of expressions which do not make any sense computationally and those which never terminate. To circumvent this, *types* were introduced. In the *simply typed lambda calculus*, there is support for some basic types along with *function abstractions*. Types prevent formation of computationally invalid functions. The simply typed lambda calculus also provides stronger guarantees such as *termination*. There are other variations of the lambda calculus which introduce notions such as *dependent types* and *polymorphism*. Here, we stick to the simply typed lambda calculus, with later chapters introducing modifications to support the work of this thesis.

The entire syntax of the simply typed lambda calculus is usually defined in a BNF (Backus-Naur Form) style grammar, as given below.

(Variables)	x, y, z		
(Types)	T	::=	$\mathrm{I} \mid \mathrm{T} \Rightarrow \mathrm{T}$
(Terms)	M, N	::=	$x \mid * \mid \lambda x.M \mid M\ N$

Figure 3.2: Syntax of the simply typed lambda calculus.

A type in this syntax can either be the *base* type I, or a function type $\mathrm{T} \Rightarrow \mathrm{T}$. Note that we can add more base types to the syntax, to represent types such as `Bool` or `Int` which appear in standard programming languages. Other constructs, that allow forming composite types such as *product types* can also be added. Variables are ranged over by small Latin letters x, y, z and can be intuitively understood to play a similar role as they play in mathematics. A variable is a valid term in the syntax. They can also appear in terms as *free* variables or *bound* by a λ-abstraction. A term is said to be *closed* if it contains no free variables. The term $*$ is the term corresponding to

the base type I. The term $\lambda x.M$ is known as a *lambda abstraction*, and corresponds to a function type. It can be understood as a mathematical function, which acts on the variable x through its body M. The variable x is a bound variable in this term. The term $M\ N$ represents function application, and can be understood as the application of a function term M to a term N. The correspondence between types and terms is made precise through the *typing judgements* which govern the validity of terms in the syntax. A typing judgement takes the form $\Gamma \vdash M : \mathrm{T}$, and is read as the term M having type T, under the *typing context* Γ. A typing context is a set of pairs of variables and their types, i.e., a set of the form $\{x : \mathrm{T}_1, y : \mathrm{T}_2, \ldots\}$. Typing judgements are usually presented in the style of inference rules. All the valid typing judgements for the calculus defined by the syntax given in Figure 3.2 are given below. The semantics for this language is

$$\frac{}{\Gamma, x : \mathrm{T} \vdash x : \mathrm{T}} \qquad \frac{}{\Gamma \vdash * : \mathrm{I}}$$

$$\frac{\Gamma, x : \mathrm{T}_1 \vdash M : \mathrm{T}_2}{\Gamma \vdash \lambda x.M : \mathrm{T}_1 \Rightarrow \mathrm{T}_2} \qquad \frac{\Gamma \vdash M : \mathrm{T}_1 \Rightarrow \mathrm{T}_2 \qquad \Gamma \vdash N : \mathrm{T}_1}{\Gamma \vdash M\ N : \mathrm{T}_2}$$

Figure 3.3: Typing judgements for the simply typed lambda calculus.

usually defined using three approaches, i.e., *operational*, *denotational*, and *axiomatic*. We only discuss the first two approaches here. The operational approach dictates how valid expressions from the syntax behave computationally, in a step by step manner. This is defined through a reduction relation on terms. Generally, such a relation is denoted using the symbol $\rightarrow$, and the expression $M \rightarrow N$ can be read as the term M reduces to N under the relation defined by $\rightarrow$. The most important rule for reduction relations defined on λ-calculi is the β-reduction rule. This rule defines how a function application reduces. Based on how β-reduction is defined, there are two most commonly used strategies that the reduction can take which depend on the evaluation order of terms, one is the *call-by-value* approach and the other is the *call-by-name* approach. In order to describe the two strategies, we first define the notions of *substitution* of a variable with a term and a *value*. Given two valid terms (i.e. well-typed) M, N, we write $M[N/x]$ to mean the term obtained after replacing all free occurences of x in M with N. A value V is a special term, which can no longer reduce further under $\rightarrow$, i.e. there is no other term M, such that the pair (V, M) is an element of $\rightarrow$. In the call-by-value approach, the β-reduction rule is given as follows:

$$(\lambda x.M)\ V \rightarrow M[V/x].$$

This approach insists that the argument to a lambda abstraction must be evaluated to its simplest possible form (i.e., a value), before it can be substituted in any function application term. In constrast to this approach, the call-by-name approach defines the β-reduction rule as:

$$(\lambda x.M)\ N \rightarrow M[N/x].$$

In this setting, *any* term can be substituted as an argument in function application, *before* it is evaluated under the reduction relation. To ensure that the operational semantics defined using any reduction strategy is coherent, generally a *type-safety* result ensures that a reduction step preserves types of the terms involved. Moreover, to ensure that a reduction path ends in terms taking certain forms (referred to as *values* or *normal forms*), a *progress* result ensures that any reduction path issued from an arbitrary *closed term* always terminates in a value.

The denotational approach defines the semantics of any programming language by interpreting its syntax as *mathematical objects*. The most commonly used mathematical structure for this, is that of a *category*. The necessary details about categories and their use in programming language design is discussed in §3.5.

3.5 Category Theory

Category theory is like an internal language of mathematics. It adopts a macroscopic view of the subject, where any mathematical object is studied through its interation with other objects, as opposed to a microscopic view where any object is characterized through what it comprises of. It also allows one to look for the common thread that binds seemingly disparate structures together through *universal properties*. For an exhaustive introduction to category theory we refer the reader to [L14].

In addition to this, the connection of category theory to logic has made it an appealing area of study to computer scientists, and has been foundational to the celebrated *Curry-Howard-Lambek* corrrespondence. In particular, category theory is pervasive in the realm of semantics of programming languages, where it has been extensively used to define *denotational semantics*. This approach gives an interpretation of a programming language in a mathematical structure. More precisely, a denotation of a program is viewed as a *function* from the syntax of the language to a chosen category. Such a semantics is deemed useful if it satisfies certain coherence properties. For an introduction to the use of category theory in the field of computer science we refer the reader to [P91].

Simultaneously, category theory has also generated considerable interest in the quantum mechanics community. This interest stems from the underlying principle in category theory of understanding a structure through its interaction with others. The usefulness of taking this approach to study quantum mechanics is highlighted in [HV19].

This thesis can be viewed as the outcome of a culmination of these three applications of category theory. We use it to study *von Neumann* algebras, and then exploit the useful mathematical structure of these objects to define denotational semantics of quantum programming languages. Denotational semantics will appear in §5.4, Chapter 5, and §6.4, Chapter 6. Additionally, we will give some simple examples along the way.

Definition 64 (Category). *A category $\boldsymbol{C}$ consists of:*

- *a collection $Ob(\boldsymbol{C})$ of objects;*
- *for each $X, Y \in Ob(\boldsymbol{C})$, a collection $Hom_{\boldsymbol{C}}(X, Y)$ of morphisms from X to Y;*
- *for each $X, Y, Z \in ob(\boldsymbol{C})$, a function*

$$\begin{aligned} Hom_{\boldsymbol{C}}(Y, Z) \times Hom_{\boldsymbol{C}}(X, Y) &\to Hom_{\boldsymbol{C}}(X, Z) \\ (g, f) &\mapsto g \circ f, \end{aligned}$$

  called composition;

- *for each $X \in Ob(\boldsymbol{C})$, an element $1_X \in Hom_{\boldsymbol{C}}(X, X)$, called the identity on X,*

such that they satisfy the following conditions:

- *for each $f \in Hom_{\boldsymbol{C}}(X, Y), g \in Hom_{\boldsymbol{C}}(Y, Z)$ and $h \in Hom_{\boldsymbol{C}}(Z, U)$, we have $(h \circ g) \circ f = h \circ (g \circ f)$;*
- *for each $f \in Hom_{\boldsymbol{C}}(X, Y)$, we have $f \circ 1_X = f = 1_Y \circ f$.*

Example 45. The category whose objects are sets and morphisms are functions between sets, denoted by **Set** is a pervasive example.

Example 46. The category whose objects are groups and morphisms are group homomorphisms, denoted by **Grp** is another common example of a category.

Example 47. The category whose objects are vector spaces over a given field $\mathbb{F}$ and morphisms are linear maps between vector spaces, denoted by $\mathbf{Vect}_{\mathbb{F}}$ is a well-known category from linear algebra.

Example 48. A category with wide spread relevance for semantics of programming languages is the category **DCPO**, whose objects are dcpos (Definition 39), and whose morphisms are Scott-continuous functions (Definition 44).

Definition 65 (Opposite Category). *Given a category $\boldsymbol{C}$, its opposite category $\boldsymbol{C}^{op}$, is the category with the same objects as $\boldsymbol{C}$, but reversed morphisms, i.e., for a morphism $f \in Hom_{\boldsymbol{C}}(X, Y)$ in the category $\boldsymbol{C}$, the corresponding morphism in $\boldsymbol{C}^{op}$ is $f^{op} \in Hom_{\boldsymbol{C}^{op}}(Y, X)$. Note that composition is reversed as expected, i.e. $(g \circ f)^{op} = f^{op} \circ g^{op}$.*

Example 49. Given a partially ordered set $(S, \leq)$, consider the category $\mathbf{POrd}_S$ whose objects are the elements of S and if we have $x \leq y$, for $x, y \in S$, we have a corresponding morphism $f : x \to y \in \mathbf{POrd}_S$. Then the corresponding opposite category $\mathbf{POrd}_S^{op}$, is again a partial order on S, which is the *dual* to $\leq$. More precisely, in the opposite category, we have $x \leq^{op} y$ iff $y \leq x$.

Remark 20. The concept of the opposite in category theory is reminiscent of the principle of duality in mathematics. Intuitively, it says that every mathematical construct has a dual, which can be obtained by "reversing" arrows. As we proceed further, many examples of dual constructions will appear.

Remark 21. Note that the opposite of the opposite category, gives us the original category, i.e. $(\mathbf{C}^{op})^{op} \cong \mathbf{C}$.

With a clear idea of what a category is, we can now define notions of some special morphisms which correspond to the universal constructions of isomorphims, surjective and injective maps in mathematics respectively. These definitions also serve as the first example among many of the concept of duality within category theory.

Definition 66 (Isomorphism). *Given a category $\boldsymbol{C}$, a morphism $f \in Hom_{\boldsymbol{C}}(X, Y)$ for objects $X, Y \in Ob(\boldsymbol{C})$ is said to be an isomorphism if there exists a morphism $g \in Hom_{\boldsymbol{C}}(Y, X)$ such that $f \circ g = 1_Y$ and $g \circ f = 1_X$. In this case we denote g by f^{-1}.*

Definition 67 (Monomorphism). *Given a category $\boldsymbol{C}$, a morphism $f \in Hom_{\boldsymbol{C}}(X, Y)$ for objects $X, Y \in Ob(\boldsymbol{C})$ is said to be a monomorphism if for any object $Z \in Ob(\boldsymbol{C})$, and morphisms $h_1, h_2 : Z \to X$, if $f \circ h_1 = f \circ h_2$, then $h_1 = h_2$.*

Definition 68 (Epimorphism). *Given a category $\boldsymbol{C}$, a morphism $f \in Hom_{\boldsymbol{C}}(X, Y)$ for objects $X, Y \in Ob(\boldsymbol{C})$ is said to be an epimorphism if for any object $Z \in Ob(\boldsymbol{C})$, and morphisms $h_1, h_2 : Y \to Z$, if $h_1 \circ f = h_2 \circ f$, then $h_1 = h_2$.*

The fact that the two notions of morphisms as defined above are dual to each other is formally stated in the next lemma.

Lemma 2. *Given a category $\boldsymbol{C}$, a morphism $f \in \boldsymbol{C}$ is a monomorphism iff it is an epimorphism in $\boldsymbol{C}^{op}$.*

Example 50. In the category **Set**, monomorphisms are exactly the injective functions and epimorphisms are exactly the surjective functions.

Building on the idea that category theory provides insight into mathematical structures through their *interactions* with each other, we now define a map between two categories.

Definition 69 (Functor). *Given two categories $\boldsymbol{C}$, and $\boldsymbol{D}$, a functor $F : \boldsymbol{C} \to \boldsymbol{D}$ consists of:*

- *a function $Ob(\boldsymbol{C}) \to Ob(\boldsymbol{D})$, written as $X \mapsto F(X)$;*
- *for each $X, Y \in \boldsymbol{C}$, a function $Hom_{\boldsymbol{C}}(X, Y) \to Hom_{\boldsymbol{D}}(F(X), F(Y))$ written as $f \mapsto F(f)$,*

satisfying the following axioms:

- *$F(g \circ f) = F(g) \circ F(f)$ for $f \in Hom_{\boldsymbol{C}}(X, Y), g \in Hom_{\boldsymbol{C}}(Y, Z)$;*
- *$F(1_X) = 1_{F(X)}$ for $X \in Ob(\boldsymbol{C})$.*

Example 51. For any category $\mathbf{C}$, the identity functor $id_{\mathbf{C}} : \mathbf{C} \to \mathbf{C}$ is defined as simply the identity map on all objects and morphisms.

Example 52. Some very pervasive examples of functors are forgetful functors. In this example we list some of the most common ones.

- For categories $\mathbf{Grp}, \mathbf{Set}$, the forgetful functor $U : \mathbf{Grp} \to \mathbf{Set}$, is defined by "forgetting" the group structure on a given group G and the homomorphism structure of a morphism f, i.e., $U(G)$ is simply the set of elements of G, and $U(f)$ is simply the function f acting on elements of a set.
- For categories $\mathbf{Vect}_{\mathbb{F}}, \mathbf{Set}$, the forgetful functor $U : \mathbf{Vect}_{\mathbb{F}} \to \mathbf{Set}$, is defined by forgetting the vector space structure on a vector space V and the linear structure of a morphism f, i.e., $U(V)$ is simply the set of elements of V, and $U(f)$ is the function f acting on elements of a set.

Any functor $F : \mathbf{C} \to \mathbf{D}$ is also often reffered to as a *covariant* functor to emphasize that it is from a category $\mathbf{C}$ and not $\mathbf{C}^{op}$. The next definition defines when a functor is said to be in the reverse direction.

Definition 70 (Contravariant functor). *Given categories $\boldsymbol{C}$, and $\boldsymbol{D}$ a contravariant functor is a functor $F : \boldsymbol{C}^{op} \to \boldsymbol{D}$.*

Example 53. A very important example of a contravariant functor appears in the form of the construction of a dual vector space in linear algebra. Recall that the dual of a vector space V denoted by V^* is the vector space of linear functionals on V. Further, notice that any linear map $f : V \to W$ between vector spaces V, W, induces a corresponding linear map between their duals, but in the opposite direction, i.e. $f^* : W^* \to V^*$ is given by $f^*(w) := w \circ f$. Hence, we have the contravariant functor, $(_)^* : \mathbf{Vect}_{\mathbb{F}}^{op} \to \mathbf{Vect}$, defined by the dual construction described above.

Example 54. An example of a contravariant functor which draws inspiration from a very important construction in the mathematical formulation of quantum mechanics is the ℓ^2 functor. Constructed using its namesake, its a functor from the opposite category $\mathbf{Bij}^{op}$ of sets and bijective functions between them to the category **Unitary**, of Hilbert spaces and unitary maps between them. It maps a set X to the Hilbert space $\ell^2(X)$, of square summable complex-valued functions on X (Example 54). For a bijective map $f : X \to Y \in Hom_{\mathbf{Bij}}(X, Y)$, we define $\ell^2(f) : \ell^2(X) \to \ell^2(Y)$ as $\ell^2(f)(\phi)(y) := \phi(f^{op}(y))$.

The universal notions of injectivity and surjectivity can be extended to functors like we did for morphisms above.

Definition 71. *A functor $F : \boldsymbol{C} \to \boldsymbol{D}$ is said to be faithful (resp. full) if for all $X, Y \in Ob(\boldsymbol{C})$, the function*

$$Hom_{\boldsymbol{C}}(X, Y) \to Hom_{\boldsymbol{D}}(F(X), F(Y))$$
$$f \mapsto F(f)$$

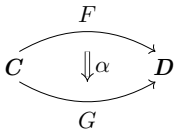
is injective (resp. surjective).

Definition 72 (Subcategory). *A category $\boldsymbol{C}$ is said to be a subcategory of a category $\boldsymbol{D}$, if $Ob(\boldsymbol{C})$ is a sub-class of $Ob(\boldsymbol{D})$ and for any $X, Y \in Ob(\boldsymbol{C})$, $Hom_{\boldsymbol{C}}(X, Y)$ is a sub-class of $Hom_{\boldsymbol{D}}(X, Y)$. Furthermore, a subcategory is said to be full, if for all $X, Y \in Ob(\boldsymbol{C})$, $Hom_{\boldsymbol{C}}(X, Y) = Hom_{\boldsymbol{D}}(X, Y)$*

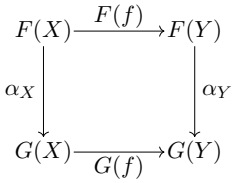
Remark 22. The image of a category under a functor need not be a subcategory of the codomain of the functor.

As we have seen so far, category theory emphasized interactions through functions. We will see now that such interactions need not be limited to objects and categories, but can be extended to functors as well. Indeed, interactions between functors have proven to be particularly useful for defining denotational semantics of programming languages.

Definition 73 (Natural Transformation). *Given two categories $\boldsymbol{C}, \boldsymbol{D}$ and functors $F, G : \boldsymbol{C} \to \boldsymbol{D}$ a natural transformation $\alpha : F \Rightarrow G$, often represented by the diagram,*



is a family of morphisms $\{\alpha_X : F(X) \to G(X)\}_{X \in \boldsymbol{C}}$ in $\boldsymbol{D}$ such that for every $f \in Hom_{\boldsymbol{C}}(X, Y)$ the following diagram commutes:

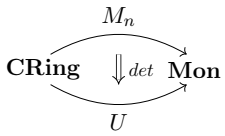


These maps α_X are called components of α.

Example 55. For a discrete category $\mathbf{C}$ (i.e., a category with only identity morphisms of objects as morphisms), a category $\mathbf{D}$ and two functors $F, G : \mathbf{C} \to \mathbf{D}$, a natural transformation α between F, G is simply given by the collection of maps $\{\alpha_X : F(X) \to G(X)\}_{X \in \mathbf{C}}$. These maps trivially satisfy the commutativity condition since the only morphisms in $\mathbf{C}$ are the identity maps.

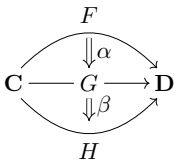
Example 56. Given a natural number n the determinant of an $n \times n$ matrix M over a commutative ring R can be understood to be a natural transformation. Consider the category **CRing**, of commutative rings and ring homomorphisms between them and the category **Mon** of monoids and monoid homomorphisms. Then for any fixed n the set of $n \times n$ matrices with entries from a commutative ring R form a monoid $M_n(R)$. This map gives us a functor $M_n : \mathbf{CRing} \to \mathbf{Mon}$. Moroever, since any commutative ring R is a monoid with the the multiplication operator, we have the forgetful functor $U : \mathbf{CRing} \to \mathbf{Mon}$. The family of maps $\{det_R : M_n(R) \to U(R)\}_{R \in \mathbf{CRing}}$

gives rise to a natural transformation,

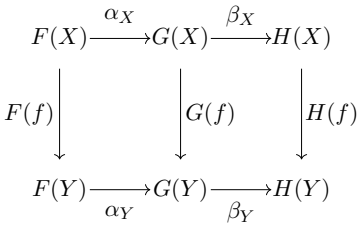


since the properties $det_R(XY) = det_R(X)det_R(Y)$ and $det_R(I) = 1$, make the determinant map into a monoid homomorphism.

Remark 23. Natural transformations can be intuitively viewed as generalizations of morphisms in a category. To understand this, first consider two categories $\mathbf{C}, \mathbf{D}$. Then, we can construct a category $[\mathbf{C}, \mathbf{D}]$ called the *functor category*, whose objects are functors from $\mathbf{C}$ to $\mathbf{D}$ and whose morphisms are natural transformations between them. This makes sense, since natural transformations can be composed in a very intuitive way. Consider two natural transformations $\alpha : F \Rightarrow G$, $\beta : G \Rightarrow H$, then the composition $\beta \circ \alpha : F \Rightarrow H$, denoted by the diagram,



is simply the family $\{\beta_X \circ \alpha_X : F(X) \Rightarrow H(X)\}_{X \in \mathbf{C}}$ of composition of morphisms in $\mathbf{D}$ such that the following diagram commutes:



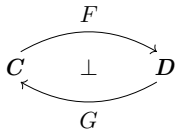
A functor category of functors from $\mathbf{C}$ to $\mathbf{D}$ is also often denoted by $\mathbf{D}^{\mathbf{C}}$.

Natural transformations allowed us to study interactions between two functors in the "same" direction. But what if we wanted to understand how functors in "opposite" directions interact? This leads us to the next definition of an *adjunction*, which is arguably one of the central ideas in category theory.

Definition 74 (Adjunction). *Given two categories $\boldsymbol{C}, \boldsymbol{D}$ and functors $\boldsymbol{C} \underset{G}{\overset{F}{\rightleftarrows}} \boldsymbol{D}$, F is the left adjoint of the functor G (resp. G is the right adjoint of the functor F), written as $F \dashv G$, if*

$$Hom_{\boldsymbol{D}}(F(X), Y) \cong Hom_{\boldsymbol{C}}(X, G(Y))$$

is a natural bijection for all objects $X \in Ob(\boldsymbol{C})$, $Y \in Ob(\boldsymbol{D})$. Then $F \dashv G$ is said to be an adjunction, often denoted by the following diagram:



Remark 24. (a) "Naturality" in Definition 74, refers to the choice of an isomorphism $\cong$ in Definition 74 such that it satisfies a naturality axiom. Here we explain this axiom. Let $\hat{(\cdot)}, \bar{(\cdot)}$ denote the two directions of the bijective correspondence in Definition 74. Specifically, we have,

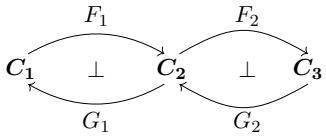
$$f \in Hom_{\mathbf{D}}(F(X), Y) \mapsto \hat{f} \in Hom_{\mathbf{C}}(X, G(Y))$$
$$\bar{g} \in Hom_{\mathbf{D}}(F(X), Y) \hookleftarrow g \in Hom_{\mathbf{C}}(X, G(Y))$$

Note that $\bar{\hat{f}} = f$ and $\hat{\bar{g}} = g$. Under the naturality axiom, this correspondence is expected to satisfy the following two conditions:

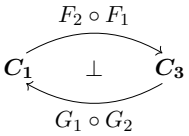
- For $F(X) \xrightarrow{f} Y \xrightarrow{h} Y'$, we have $\widehat{h \circ f} = G(h) \circ \hat{f}$;
- For $X' \xrightarrow{h'} X \xrightarrow{g} G(Y)$, we have $\overline{g \circ h'} = \bar{g} \circ F(h')$.

(b) Given a functor F, if it's left (resp. right) adjoint exists, it is unique up to a natural isomorphism.

(c) Like natural transformations, adjunctions can be composed. The following adjunctions,



give rise to the composite adjunction:



An adjunction results in the natural transformations:

- $\eta : id_{\mathbf{C}} \Rightarrow G \circ F$ called the *unit* of the adjunction;
- $\varepsilon : F \circ G \Rightarrow id_{\mathbf{D}}$ called the *counit* of the adjunction;

such that for any $X \in Ob(\mathbf{C}), Y \in Ob(\mathbf{D})$ we have,

$$id_{F(X)} = \varepsilon_{F(X)} \circ F(\eta_X) \qquad id_{G(Y)} = G(\varepsilon_Y) \circ \eta_{G(Y)}.$$

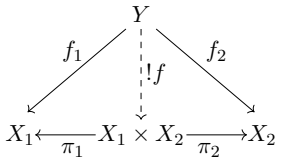
Example 57. Recall the forgetful functor $U : \mathbf{Vect}_{\mathbb{F}} \to \mathbf{Set}$ from Example 52. This functor has a left adjoint $F : \mathbf{Set} \to \mathbf{Vect}_{\mathbb{F}}$ known as the *free* construction. Given a set $S \in Ob(\mathbf{Set})$, $F(S)$ is defined to be the set of all formal $\mathbb{F}$-linear combinations of elements of S, i.e., $\Sigma_{s \in S} \alpha_s s$, where each α_s is a scalar from $\mathbb{F}$ and only finitely many of them are non-zero. Addition of two elements and multiplication by a scalar are defined in the obvious way. $F \dashv U$, amounts to saying that defining a linear map $F(S) \to V$ is equivalent to defining a function $S \to U(V)$. It can be easily checked that this turns out to be the case. Note that in general, many other algebraic structures can be viewed as free constructions. In all these cases, generally the free construction is a left adjoint to the forgetful functor that also arises out of it.

Definition 75 (Initial and Terminal Object). *Given a category $\boldsymbol{C}$, an object $0 \in Ob(\boldsymbol{C})$ is initial (resp. $1 \in Ob(\boldsymbol{C})$ is terminal) if for every $X \in Ob(\boldsymbol{C})$, there is a unique map $0 \to X$ (resp. $X \to 1$).*

Remark 25. Initial and terminal objects in any category are unique up to a unique isomorphism.

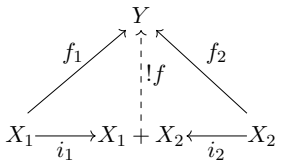
So far all the constructs we introduced were about interactions between categories. Now we introduce specific instances of the concept of *limits*, which deal with what goes on inside a category.

Definition 76 (Product). *Given a category $\boldsymbol{C}$ and $X_1, X_2 \in Ob(\boldsymbol{C})$, the product of X_1 and X_2 is an object $X_1 \times X_2$ and morphisms π_1, π_2 called projections, such that for every $Y \in Ob(\boldsymbol{C})$ and morphisms $f_1 : Y \to X_1, f_2 : Y \to X_2$, there is a unique morphism $f : Y \to X_1 \times X_2$ such that the following diagram commutes:*



The morphism f is usually denoted by $\langle f_1, f_2 \rangle$.

Definition 77 (Coproduct). *Given a category $\boldsymbol{C}$ and $X_1, X_2 \in Ob(\boldsymbol{C})$, the coproduct of X_1 and X_2 is an object $X_1 + X_2$ and morphisms i_1, i_2 called injections, such that for every $Y \in Ob(\boldsymbol{C})$ and morphisms $f_1 : X_1 \to Y, f_2 : X_2 \to Y$, there is a unique morphism $f : X_1 + X_2 \to Y$ such that the following diagram commutes:*



The morphism f is usually denoted by $[f_1, f_2]$.

Note that a product and coproduct are dual concepts, as stated in the next lemma.

Lemma 3. *Given a category $\boldsymbol{C}$, for an object X and morphisms f_1, f_2, (X, f_1, f_2) is a product iff (X, f_1^{op}, f_2^{op}) is a coproduct in $\boldsymbol{C}^{op}$.*

Example 58. Given two sets $X, Y \in \mathbf{Set}$, their product exists, and is given by the cartesian product $X \times Y$, along with the usual projection maps p_1, p_2. Similarly, their coproduct also exists and is given by their tagged disjoint union, defined as $X + Y := \{(x, 0) \mid x \in X\} \sqcup \{(y, 1) \mid y \in Y\}$ along with the expected inclusion maps i_1, i_2.

Example 59. Given two dcpos $P, Q \in \mathbf{DCPO}$, their product exists, and the dcpo is given by the cartesian product $P \times Q$, of the underlying set, and the order structure is defined as $(p_1, q_1) \leq (p_2, q_2)$ iff $p_1 \leq p_2$ in P and $q_1 \leq q_2$ in Q. The usual projection maps defined on this structure are Scott-continuous. The coproduct also exists in this category. The order structure on the set $P + Q$ (same as the underlying set for the coproduct in **Set**) is given by $(p_1, 0) \leq (p_2, 0)$ iff $p_1 \leq p_2$ in P and $(q_1, 1) \leq (q_2, 1)$ iff $q_1 \leq q_2$ in Q. There is no relation between the elements of different components, i.e., there is no ordering between elements of P and Q. The usual inclusions are Scott-continous since, like the projections they are monotonic and preserve joins of directed sets.

Example 60. Given two categories $\mathbf{C}, \mathbf{D}$, their product $\mathbf{C} \times \mathbf{D}$ is a category. A functor defined on this category is called a *bifunctor*.

Remark 26. The notion of products and co-products can be extended to arbitrarily many objects in a natural way:

- Given a category C, an index set I and a family of objects $\{X_i\}_{i \in I}$, a product of $\{X_i\}_{i \in I}$ consists of an object P and a family of maps $\{P \xrightarrow{p_i} X_i\}_{i \in I}$, such that for any object Y and family of maps $\{Y \xrightarrow{f_i} X_i\}_{i \in I}$, there exists a unique map $f : Y \to P$ such that $p_i \circ f = f_i$ for all $i \in I$.
- Given a category C, an index set I and a family of objects $\{X_i\}_{i \in I}$, a coproduct of $\{X_i\}_{i \in I}$ consists of an object P and a family of maps $\{X_i \xrightarrow{p_i} P\}_{i \in I}$, such that for any object Y and family of maps $\{X_i \xrightarrow{f_i} Y\}_{i \in I}$, there exists a unique map $f : P \to Y$ such that $f \circ p_i = f_i$ for all $i \in I$.

The idea of interpreting Types of a language as objects of a category, and Terms as morphisms gave rise to the *Curry-Howard-Lambek* correspondence. In order to be suitable for such an interpretation, a category needs to be *cartesian closed*. This property allows one to treat function spaces as "first-class" citizens of a category (i.e., objects), and is reminiscent of the functional style of porgramming, which allows functions to be treated as values which can be manipulated. As the name, suggests a cartesian category comes equipped with the structure of a product, mimicking the cartesian product of sets.

Definition 78 (Cartesian category). *A category $\boldsymbol{C}$ is said to be cartesian if,*

- *$\boldsymbol{C}$ has a terminal object* 1*;*
- *binary products for any pair of objects X, Y.*

In order to intepret programming languages meaningfully in categories, there is yet another property that a category needs to have. The property of being *closed* explores the structure on function spaces of a category, through their "similarity" to its objects.

The idea described above, give rise to the notion of *cartesian closed categories*, which are basically synonymous with denotational models of the simply typed λ-calculus.

Definition 79 (Cartesian closed category (CCC)). *A category $\boldsymbol{C}$ is said to be cartesian closed if,*

- *$\boldsymbol{C}$ has a terminal object* 1*;*
- *binary products for any pair of objects X, Y;*

such that for any object $X \in Ob(\boldsymbol{C})$, the map $(_ \times X) : \boldsymbol{C} \to \boldsymbol{C}$ is a left adjoint (functor).

The right adjoint of the functor $_ \times X$ defined in Definition 79 is denoted by $[X \to _] : \mathbf{C} \to \mathbf{C}$. As the notation suggests, it is this functor that is used to construct the *exponential object* for any pair of objects. In fact, the exponential object for any two objects X, Y is defined as $Y^X := [X \to Y]$. Moreover, in **CCC**s, the exponential object comes equipped with the *evaluation map*, which is a morphism defined as,

$$eval_{X,Y} : Y^X \times X \to Y$$

The exponential object with the evalutaion map satisfies the *universal property*, that given a morphism $h : Z \times X \to Y$, there is a unique morphism $h^\lambda : Z \to Y^X$ such that $eval_{X,Y} \circ (h^\lambda \times id_X) = h$. This morphism h^λ is known as the *curried* version of h.

Example 61. The category **Set** is cartesian closed, with the exponential object Y^X simply being the set $\{f \mid f : X \to Y\}$, for sets X, Y.

With the basics in place, the denotational semantics for the simply typed λ-calculus, with syntax given in Figure 3.2 can now be defined. Types are interpreted as objects in a relevant **CCC**, and typing judgements in Figure 3.3 as morphisms between them. The interpretations of Types are given below, and

$$[\![\mathrm{I}]\!] := 1 \qquad [\![\mathrm{T}_1 \Rightarrow \mathrm{T}_2]\!] := [\![\mathrm{T}_2]\!]^{[\![\mathrm{T}_1]\!]}$$

the interpretations of typing judgements are given in Figure 3.4.

$$\begin{aligned}
[\![\Gamma \vdash M : \mathrm{T}]\!] &\in Hom_{\mathbf{C}}([\![\Gamma]\!], [\![\mathrm{T}]\!]) \\
[\![\Gamma, x : \mathrm{T} \vdash \mathrm{T}]\!] &:= [\![\Gamma]\!] \times [\![\mathrm{T}]\!] \xrightarrow{\pi_2} [\![\mathrm{T}]\!] \\
[\![\Gamma \vdash * : \mathrm{I}]\!] &:= [\![\Gamma]\!] \xrightarrow{!} 1 \\
[\![\Gamma \vdash \lambda x.M : \mathrm{T}_1 \Rightarrow \mathrm{T}_2]\!] &:= [\![\Gamma, x : \mathrm{T}_1 \vdash M : \mathrm{T}_2]\!]^{\lambda} \\
[\![\Gamma \vdash M\ N : \mathrm{T}_2]\!] &:= eval_{[\![\mathrm{T}_1]\!],[\![\mathrm{T}_2]\!]} \circ \langle [\![\Gamma \vdash M : \mathrm{T}_1 \Rightarrow \mathrm{T}_2]\!], [\![\Gamma \vdash N : \mathrm{T}_1]\!] \rangle
\end{aligned}$$

Figure 3.4: Interpretation of typing judgements for the simply typed λ-calculus.

The behaviour of programs as defined by the operational semantics is tied to their interpretations in the denotational semantics through *soundness* and *completeness* results, which often take the form of the lemmas stated below.

Lemma 4 (Soundness). *For a typing judgement* $\Gamma \vdash M : \mathrm{T}$*, if* $M \to N$*, then* $[\![\Gamma \vdash M : \mathrm{T}]\!] = [\![\Gamma \vdash N : \mathrm{T}]\!]$.

Lemma 5 (Completeness). *Given two typing judgements* $\Gamma \vdash M : \mathrm{T}$, $\Gamma \vdash N : \mathrm{T}$*, if* $[\![\Gamma \vdash M : \mathrm{T}]\!] = [\![\Gamma \vdash N : \mathrm{T}]\!]$ *, then* $M \sim N$*. Here,* $\sim$ *is the reflexive, symmetric and transitive closure of the relation* $\to$*.*

Remark 27. Note that in general, completeness is known as *adequacy* in programming language terminology. Adequacy is harder to prove than soundness, and need not always hold.

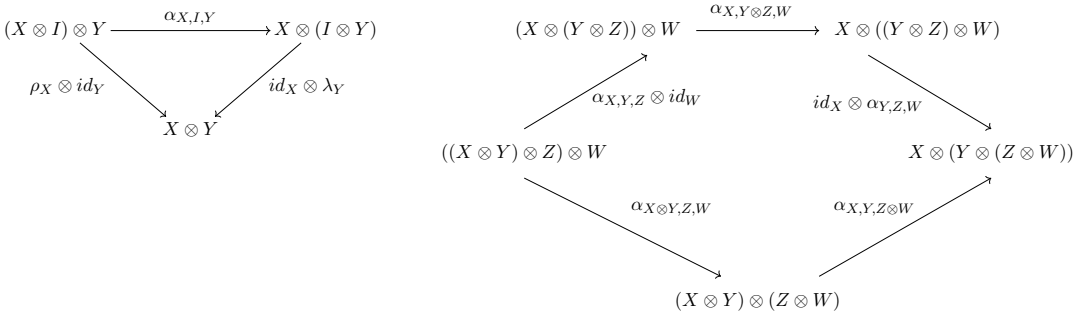
With the crux of the Curry-Howard correspondence in place, we now introduce more general versions of cartesian closed categories, which play a crucial role in the categorical approach to quantum computing. In mathematics, a *monoid* over a set is defined via a binary relation resembling *multiplication*. A similar generalization of a cartesian category, is a monoidal category. The *monoidal product* (also known as the *tensor product*) is a generalization of the cartesian product.

Definition 80 (Monoidal category). *A category* $\boldsymbol{C}$ *is said to be monoidal if it is equipped with,*

- *a bifunctor* $\otimes : \boldsymbol{C} \times \boldsymbol{C} \to \boldsymbol{C}$*;*
- *an object* I *called the unit;*
- *for all objects* X, Y, Z*, a natural isomorphism* $\alpha_{X,Y,Z} : (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$ *known as the associator;*

- *for all objects X, a natural isomorphism $\lambda_X : I \otimes X \to X$ known as the left unitor;*
- *for all objects X, a natural isomorphism $\rho_X : X \otimes I \to X$ known as the right unitor;*

such that for all $W, X, Y, Z \in Ob(\boldsymbol{C})$ the diagrams



commute.

The commutative diagrams shown above are known as the *triangle identity* for unitors and associators and the *pentagon identity* for associators. These equations are together known as *coherence conditions*, and ensure that the monoidal product is well behaved with respect to the unitors and associators.

Example 62. The category **Mon** of monoids and monoid homomorphisms is a monoidal category, with the tensor product given by the direct product of monoids and the tensor unit given by the trivial monoid, i.e., any single element set $\{e\}$ equipped with the product $e \cdot e = e$.

Example 63. The category **Hilb** of Hilbert spaces and linear maps between them is a monoidal category, with the tensor product given by the tensor product of Hilbert spaces, and the tensor unit given by the one-dimensional Hilbert space $\mathbb{C}$.

As one might expect, there are special functors between monoidal categories, which preserve their monoidal structure.

Definition 81 (Monoidal functor). *Given two monoidal categories $(\boldsymbol{C}, \otimes_{\boldsymbol{C}}, I_{\boldsymbol{C}})$ and $(\boldsymbol{D}, \otimes_{\boldsymbol{D}}, I_{\boldsymbol{D}})$ a functor $F : \boldsymbol{C} \to \boldsymbol{D}$ is said to be monoidal if,*

- *there is a natural transformation,*

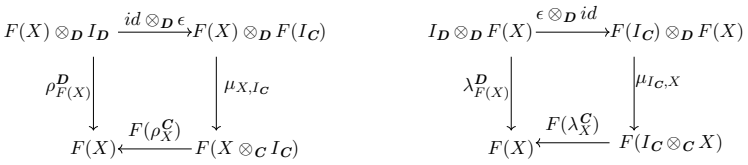
$$\mu_{X,Y} : F(X) \otimes_{\boldsymbol{D}} F(Y) \to F(X \otimes_{\boldsymbol{C}} Y)$$

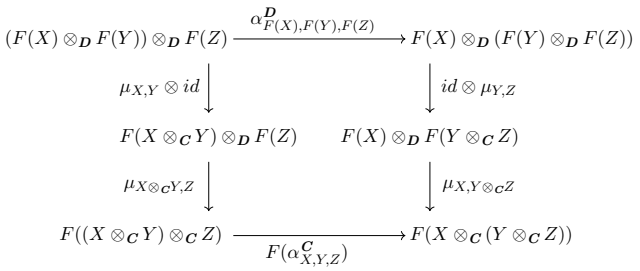
  for all $X, Y \in Ob(\boldsymbol{C})$;

- *and a morphism,*

$$\epsilon : I_{\boldsymbol{D}} \to F(I_{\boldsymbol{C}});$$

such that the diagrams,





commute for all $X, Y, Z \in Ob(\boldsymbol{C})$.

Example 64. The free construction, which often gives us a functor from **Set** to appropriate categories of mathematical objects, often turns out to be a monidal functor with respect to the corresponding monoidal structure on these categories. A common instance is the free construction of a vector space. The functor $F : \mathbf{Set} \to \mathbf{Vect}_{\mathbb{F}}$, defined as $F(X) := \mathbb{F}(X)$ (i.e., the free vector space on X) and $F(f)(\sum_i a_i x_i) := \sum_i a_i f(x_i)$ is monoidal, with respect to the cartesian product on **Set** and the tensor product on $\mathbf{Vect}_{\mathbb{F}}$. In fact, its adjoint, the forgetful functor $U : \mathbf{Vect}_{\mathbb{F}} \to \mathbf{Set}$ is also a monoidal functor.

Definition 82 (Strong monoidal functor). *A monoidal functor* $F : \boldsymbol{C} \to \boldsymbol{D}$ *is said to be strong if* ϵ *and* $\mu_{X,Y}$ *are isomorphisms for all* $X, Y \in Ob(\boldsymbol{C})$.

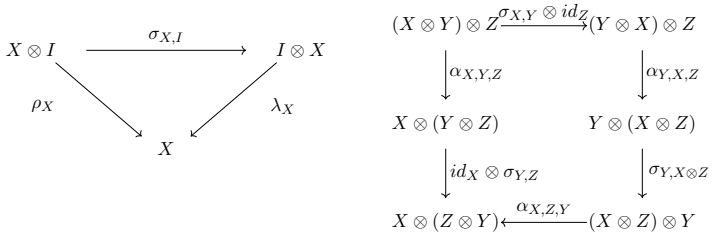
Example 65. The free construction functor $F : \mathbf{Set} \to \mathbf{Vect}_{\mathbb{F}}$ described in Example 64 is strong monoidal. However, its adjoint, the forgetful functor is not. Another important example is the functor $\ell^2 : \mathbf{Bij}^{op} \to \mathbf{Unitary}$ described in Example 54 is strong monoidal, with respect to the cartesian product on **Set** and the tensor product of Hilbert spaces on **Unitary**.

Definition 83 (Strict monoidal functor). *A monoidal functor* $F : \boldsymbol{C} \to \boldsymbol{D}$ *is said to be strict if* ϵ *and* $\mu_{X,Y}$ *are identity morphisms for all* $X, Y \in Ob(\boldsymbol{C})$.

Example 66. The identity functor on the category **Mon** of monoids and monoid homomorphisms is strict with respect to the cartesian product of monoids and the unit being the trivial monoid $\{e\}$.

A *symmetric monoidal category* is a particualr kind of *monoidal category* in which the tensor product "commutes" upto isomorphism.

Definition 84 (Symmetric monoidal category). *A symmetric monoidal category is a monoidal category* $\boldsymbol{C}$ *along with a natural isomorphism* $\sigma_{X,Y} : X \otimes Y \to Y \otimes X$ *known as the symmetry morphism, such that for all* $X, Y, Z \in Ob(\boldsymbol{C})$ *the diagrams,*



commute and for any $X, Y \in Ob(\boldsymbol{C})$, $\sigma_{Y,X} \circ \sigma_{X,Y} = id_{X \otimes Y}$.

Remark 28. Any category with finite products (Remark 26) is a symmetric monoidal category. In particular, this means that $\mathbf{Set}, \mathbf{Vect}_{\mathbb{F}}$ are symmetric monoidal. However, note that the standard construction of a monoidal product in $\mathbf{Vect}_{\mathbb{F}}$ is given by the *tensor product* of vector spaces, and not the direct sum construction, which is the usual notion of *product* in this category.

Example 67. The categories **Mon**, **Hilb** are symmetric monoidal.

Definition 85 (Symmetric monoidal closed category). *A symmetric monoidal closed category* $\boldsymbol{C}$ *is a symmetric monoidal category, such that for every* $X \in Ob(\boldsymbol{C})$ *the functor* $_ \otimes X$ *has a right adjoint* $[X, _]$.

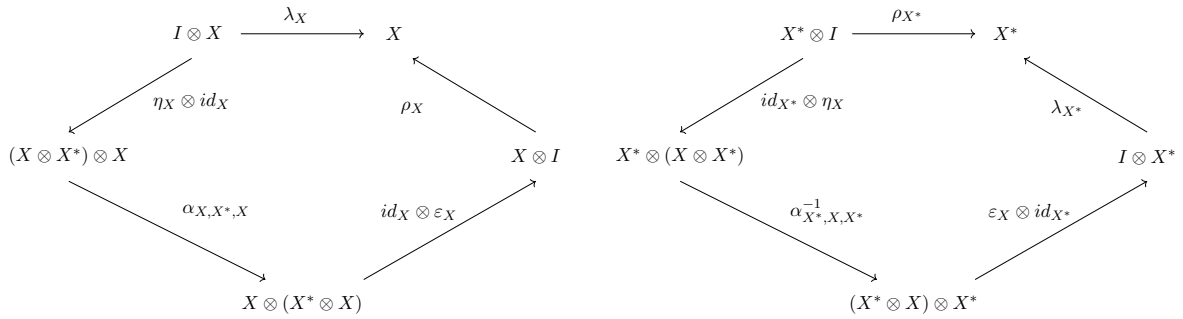
Example 68. The categories $\mathbf{Set}, \mathbf{Vect}_{\mathbb{F}}$ are symmetric monoidal closed with respect to the cartesian product of sets and the tensor product of vector spaces respectively.

A monoidal category can be understood to arise out of the multiplication operation that can be defined on a set. What if the existence of inverses under such an operation was guaranteed? Dual objects can be understood through two mathematical lenses. One is the generalization of the notion of an invertible object under the monoidal product. The other is the generalization of function spaces (*dual* of a vector space), in a way so that they interact well with the monoidal product. We saw this interaction between the two structures via adjoints, while defining *currying* using *closed* categories. The existence of a dual in both these senses is important enough to warrant a name for symmetric monoidal categories where this is always guaranteed. *Compact* categories are symmetric monoidal categories whose objects always have a dual (the same object is a dual in both the senses described above). They are important for quantum mechanics because they provide a model for the notion of duality and adjoints from quantum mechanics in categories. Before we introduce them formally, we first define the *dual* of an object in the monoidal sense. Duals can be understood to resemble "weak inverses" under the monoidal product, so that resulting morphisms that arise can be used to interpret operations like state preparation in quantum mechanics.

Definition 86 (Dual object). *Given a monoidal category* $\boldsymbol{C}$, *an object* $X \in Ob(\boldsymbol{C})$ *is said to be dualizable, if there is an object* X^* *and,*

- *a morphism* $\eta_X : I \to X \otimes X^*$ *called the unit (coevaluation);*
- *a morphism* $\varepsilon_X : X^* \otimes X \to I$ *called the counit (evaluation);*

which make the diagrams,



commute. In this case, X is the right dual of X^*, and X^* is the left dual of X.

Definition 87 (Rigid monoidal category). *A monoidal category $\boldsymbol{C}$ is said to be rigid if every $X \in Ob(\boldsymbol{C})$, has a left dual and a right dual.*

Remark 29. If an object X is both a left and a right dual to an object Y, it is simply known as a dual of Y. In a symmetric monoidal category the left and the right dual coincide.

Definition 88 (Compact category). *A rigid monoidal category $\boldsymbol{C}$ is said to be compact if it is symmetric monoidal.*

In general, as stated previously, there are two possible notions of a dualizable object. One is in the sense of closedness, which relates the dual of an object to a function space. The other is in the sense of invertibility of the monoidal product. In general, these two different ideas lead to different kinds of dual which do not always coincide. However, in the case of a compact category, it turns out that they do. The functor $[X, _] := _ \otimes X^*$ turns out to be the right adjoint to the tensor functor $_ \otimes X$ (*i.e*, $_ \otimes X \dashv [X, _]$), which is a sufficient condition for $Hom(X, Y)$ to be an object of the category for any two objects X, Y. Hence, compact categories are also referred to as *compact closed categories*.

Example 69. The category $\mathbf{FdVect}_{\mathbb{C}}$ of finite-dimensional complex vector spaces and linear maps between them is a compact category. The categorical dual of a vector space V is given by its algebraic dual V^*, which is the vector space of linear functionals on V. The monoidal product is given by the tensor product of vector spaces and the monoidal unit is given by $\mathbb{C}$.

Example 70. The category **FdHilb** of finite-dimensional Hilbert spaces and linear maps between them is a compact category. The categorical dual of a Hilbert space H is given by its algebraic dual H^*, which is the vector space of linear functionals on H. Let $\overline{H}$ denote the *complex conjugate* Hilbert space of H, which has the same underlying set as H, but whose scalar multiplication is anti-linear, i.e $\alpha \cdot_{\overline{H}} h := \overline{\alpha} \cdot_H h$ and the inner product is given by $\langle v, w \rangle_{\overline{H}} := \overline{\langle v, w \rangle_H}$. There is a natural vector space isomorphism $\Phi : H^* \to \overline{H}$, which associates a vector $v_\phi \in \overline{H}$ to each $\phi \in H^*$. Using this isomorphism, the inner product on H^* is defined as $\langle \phi, \psi \rangle_{H^*} := \langle v_\psi, v_\phi \rangle_H$. The monoidal product is given by the tensor product of Hilbert spaces and the monoidal unit is given by the Hilbert space $\mathbb{C}$.

3.6 Linear Logic

The relationship between logic and programming languages can be considered as old as the history of theoretical computer science itself. Formally, this relationship is highlighted clearly through the *Curry-Howard* correspondence, which views propositions as types and proofs as programs.

The Curry-Howard correspondence provides a powerful lens through which one can design safe, expressive, and reliable languages. This correspondence equates logical propositions to types, proofs to programs and then logical connectives to type constructors. For example, the logical connective of implication $A \Rightarrow B$ corresponds to the function type $A \to B$, where the types A, B correspond to their namesake propositions respectively. Similarly, conjunction of propositions $A \wedge B$ corresponds to the product type $A \times B$ and disjunction of propositions $A \vee B$ corresponds to the type $A + B$. Finally, False ($\bot$) corresponds to empty types (i.e., a type without any values) and True ($\top$) corresponds to the unit type. Proceeding in this fashion, it is easy to infer that typing judgements correspond to permissible introduction and elimination rules of logic. Under this correspondence, the simply typed lambda calculus introduced in §3.4 corresponds to *intuitionistic propositional logic*. More generally, functional programming languages correspond to intuitionistic logic, and resource sensitive languages correspond to *linear logic*.

Unlike *classical logic*, intuitionistic logic takes a constructionist view of mathematics, i.e., any logical statement must be constructively proved to be true or false. Any existential claim is considered true in intuitionistic logic only if, an instance is specifically provided. Moreover, the *law of the excluded middle* (i.e., $P \vee \neg P$) is not generally considered valid in intuitionistic logic. Similarly, double negation elimination (i.e. $\neg\neg P \implies P$) is also not generally considered valid in this setting.

Linear logic is a refinement of both classical and intuitionistic logic, that treats propositions as resources- emphasizing how many times a hypothesis is used, along with whether it is true. Linear logic was introduced by Jean-Yves Girard in 1987. It has since then had seismic impact in the fields of proof theory, programming languages and more recently quantum computation. A detailed treatment of logic and its relation to theoretical computer science can be found in [GLT89]. In both classical and intuitionistic logic, assumptions can be used as often as needed. In contrast, linear logic prohibits this. All assumptions must be used exactly once, unless explicitly stated. Therefore, in linear logic the rules of *contraction* (copying) and *weakening* (deleting) are not permissible, without the use of modal operators like $!, ?$. This immediately makes it a fit for quantum computation, since the no cloning theorem reflects the lack of a rule for contraction. Using a linear type system for designing a quantum programming language would prevent one from copying quantum data, accidentally erasing or overwriting quantum states and finally adhering to entanglement constraints. Indeed, Quipper, a domain specific language of Haskell uses linear types to model quantum data flow. The quantum lambda calculus, which we discuss at length in Chapter 4 is a variant of the simply typed lambda calculus for quantum computation based on a linear type system. As expected, the variables A, B, C are used to range over formulae and proof rules corresponding to the particular logic are used to derive sequents of the form,

$$A_1, A_2, \ldots, A_m \vdash B_1, B_2 \ldots B_n.$$

Such a sequent represents a formula expressing the fact that the conjunction of the formulae $A_1, A_2, \ldots, A_m$ implies the disjunction of the formulae $B_1, B_2, \ldots B_n$. The variables Γ, Δ represent *multisets* of formulae. Note that the presence of contraction and weakening allows one to use *sets* of formulae instead of multisets. However, for the linear fragment of any logic presented, the distinction is necessary, and so we assume without loss of generality that these symbols range over multisets of formulae. Now we briefly discuss Classical (Linear) Logic, before delving into Intuitionistic Linear Logic, whose variant IMALL (Intuitionistic Multiplicative Additive Linear Logic) appears in the work presented in Chapter 6. Note that the formulae of Classical Logic are given by the grammar,

$$A \quad ::= \quad 1 \mid A \wedge A \mid 0 \mid A \vee A \mid A \implies A.$$

The proof rules of Classical Logic in a Gentzen style representation are given in Figure 3.5.

$$\frac{}{A \vdash A}\ Axiom$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma', \Gamma \vdash \Delta', \Delta}\ Cut$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ Weakening_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ Weakening_R$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ Contraction_L \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ Contraction_R$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ \wedge_{L_1} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ \wedge_{L_2} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'}\ \wedge_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, 1 \vdash \Delta}\ 1_L \qquad \frac{}{\vdash 1}\ 1_R$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \implies B \vdash \Delta, \Delta'}\ \implies_L \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \implies B, \Delta}\ \implies_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \vee B \vdash \Delta, \Delta'}\ \vee_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee_{R_1} \quad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee_{R_2}$$

$$\frac{}{0 \vdash}\ 0_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash 0, \Delta}\ 0_R$$

Figure 3.5: Proof rules (Classical Logic).

Note that this representation of Classical Logic uses the implication instead of the negation. In this system we can define negation as $\neg A := A \implies 0$. Alternatively, we can replace the rules for implication with those for negation, and define implication as $A \implies B := \neg A \vee B$. Both these representations of Classical Logic are equivalent.

The fundamental idea in the proof-as-program analogy, results in the reduction of a proof (cut-elimination) to correspond to the evaluation of a program. One would then ask if the same reduced proof is always obtained independent of the choice of reductions. Unfortunately, it turns out the cut-elimination is non-deterministic in Classical Logic in general, and hence this property does not hold. Because of this, Classical Logic has no non-trivial sound denotational semantics.

The workaround to this limitation leads us to Intuitionistic Logic as well as (Classical) Linear Logic. There are two ways to fix this issue:

- The right hand side of each sequent is subject to the restriction that it contains exactly one formula. This gives rise to Intuitionistic Logic which we will discuss soon.
- Contraction and Weakening are marked explicitly using the modalities ! and ? on the formulae. This in turn gives rise to Classical Linear Logic which we briefly discuss next.

Classical Linear Logic is comprised of the following formulae:

$$A \ ::= \ \top \mid A \otimes A \mid \bot \mid A \parr A \mid A \multimap A \mid !A \mid ?A \mid A \,\&\, A \mid 1 \mid A \oplus A \mid 0.$$

The proof rules for Classical Linear logic are given in Figure 3.6. These rules are used to derive sequents of the form,

$$A_1, A_2, \ldots, A_m \vdash B_1, B_2, \ldots B_n.$$

The !-modality corresponds to contraction and weakening on the left side of the sequent, whereas the modality ?-modality corresponds to contraction and weakening on the right side of the sequent. This means that $!A$ can be used freely like a classical resource, and $?A$ its dual, allows *contexts* to freely use (i.e., discard or copy A). The connectives $\bigoplus, \&, \top, 0$ correspond to the additive fragment of the logic and the system obtained after excluding these connectives is the multiplicative fragment. Since all formulae can be used exactly once unless explicitly stated, various connectives imply different available choices. In the multiplicative fragment, & and $\otimes$ have dual meanings: $A \parr B$ denotes a formula where a proof for both A and B are available, but exactly one can be used, whereas $A \otimes B$ denotes a formula where a proof for both A and B is available and both must be used. In contrast to this, in the additive fragment we have & and $\oplus$ which represent internal (you choose) and external choices (context chooses) reflecting dual non-determinism. $A \multimap B$ represents linear multiplication, i.e., A can be used exactly once to produce a proof for B. $1, \bot$ represent empty resource, absurdity respectively in the multiplicative fragment. Likewise, $0, \top$ represent impossible and trivial respectively in the additive fragment. Negation can be defined in this sytem as $\neg A := A \multimap \bot$. Since this is an extension of classical logic, $\neg\neg A$ is equivalent to A. The connectives $\otimes, \parr$ are De Morgan duals, and so are $\oplus, \&$. Note that although this system can now be given a non-trivial sound denotational semantics via *coherence spaces*, Classical Linear Logic still does not satisfy the *Church-Rosser* property, i.e., if a proof tree π reduces to two trees π', π'', we cannot in general guarantee in this system that they will both reduce further to a common proof tree $\tilde{\pi}$.

With the formal framework for Classical Logic and its Linear variant laid out, we now discuss the same for Intuitionistic Logic. As stated earlier, Intuitionistic Logic differs from Classical Logic in its constructionist view of mathematical logic. The issues that arise in Classical Logic, as a suitable logic for the proofs-as-programs paradigm, are resolved here, and hence it has a clear correspondence with the simply typed lambda calculus under the *Curry-Howard* isomorphism. We present the proof rules for this logic in the Natural Deduction style, which associates to each connective an *introduction* rule and an *elimination* rule. The proof rules are presented in Figure 3.7, and are used to derive sequents of the form,

$$A_1, A_2 \ldots, A_m \vdash B.$$

While contraction and weakening rules are explicitly a part of the Gentzen style formulation, in the Natural Deduction style they are admissible. This presentation of Intuitionistic Logic satisfies the Church-Rosser property, which means that for any two reductions of a proof tree π to π' and π'', there is a proof tree $\tilde{\pi}$ to which both of these proof trees reduce. Moreover, it also satisfies the *strong normalization* property, i.e., all reduction sequences issued from a given proof are of finite length. These two properties together imply that every proof tree π reduces to a unique proof which can no longer be further reduces. This is known as the *normal form* of that proof, and via the Curry-Howard isomorphism, this corresponds to unique normal forms for the lambda calculus semantics.

Now we discuss Intuitionistic Linear Logic, whose formulae are the same as Classical Linear Logic, except the connectives $\bot, \parr$ and ? are ommitted. Again, the proof rules are presented in Natural Deduction style in Figure 3.8, and used to prove sequents of the form,

$$A_1, A_2, \ldots, A_m \vdash B.$$

Similar to Intuitionistic Logic, Intuitionistic Linear Logic satisfies the Church-Rosser property and strong normalization, and therefore corresponds to a linear lambda calculus under the Curry-Howard isomorphism. This isomorphism gives rise to a natural question. Since the quantum

$$\frac{}{A \vdash A}\ Axiom$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma', \Gamma \vdash \Delta', \Delta}\ Cut$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \otimes_L \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ \otimes_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta}\ \top_L \qquad \frac{}{\vdash \top}\ \top_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ \parr_L \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \parr_R$$

$$\frac{}{\bot \vdash}\ \bot_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\ \bot_R$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \multimap B \vdash \Delta, \Delta'}\ \multimap_L \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \multimap_R$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ !_L \qquad \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}\ !_R$$

$$\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\ ?_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}\ ?_R$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\ Weakening_L \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, ?A}\ Weakening_R$$

$$\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}\ Contraction_L \qquad \frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}\ Contraction_R$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}\ \&_{L_1} \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}\ \&_{L_2} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \,\&\, B, \Delta}\ \&_R$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}\ \oplus_L \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \oplus_{R_1} \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ \oplus_{R_2}$$

$$\frac{}{\Gamma, 0 \vdash \Delta}\ 0_L \qquad \frac{}{\Gamma \vdash 1, \Delta}\ 1_R$$

Figure 3.6: Proof rules (Classical Linear Logic).

$$\frac{}{A_1, A_2, \ldots, A_m \vdash A_q}\ Axiom$$

$$\frac{}{\Gamma \vdash 1}\ 1_I \qquad \frac{\Gamma \vdash 0}{\Gamma \vdash C}\ 0_E$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge_I \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge_{E_1} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge_{E_2}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \implies B}\ \implies_I \qquad \frac{\Gamma \vdash A \implies B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \implies_E$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{I_1} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee_{I_2} \qquad \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee_E$$

Figure 3.7: Proof rules (Intuitionistic Logic).

$$\frac{}{A \vdash A}\ Axiom$$

$$\frac{}{\vdash \top}\ \top_I \qquad \frac{\Gamma \vdash \top \quad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash A}\ \top_E$$

$$\frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\ \otimes_I \qquad \frac{\Gamma \vdash A \otimes B \quad \Gamma', A, B \vdash C}{\Gamma, \Gamma' \vdash C}\ \otimes_E$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap_I \qquad \frac{\Gamma \vdash A \multimap B \quad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash B}\ \multimap_E$$

$$\frac{\Gamma_1 \vdash !A_1, \ldots, \Gamma_n \vdash !A_n \quad !A_1, \ldots, !A_n \vdash B}{\Gamma_1, \Gamma_2, \ldots, \Gamma_n \vdash !B}\ Prom. \qquad \frac{\Gamma \vdash !B}{\Gamma \vdash B}\ Dereliction$$

$$\frac{\Gamma \vdash !A \quad \Gamma', !A, !A \vdash B}{\Gamma, \Gamma' \vdash B}\ Contraction \qquad \frac{\Gamma \vdash !A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash B}\ Weakening$$

$$\frac{\Gamma_1 \vdash A_1, \ldots, \Gamma_n \vdash A_n}{\Gamma_1, \ldots, \Gamma_n \vdash 1}\ 1_I \qquad \frac{\Gamma_1 \vdash A_1, \ldots, \Gamma_n \vdash A_n \quad \Delta \vdash 0}{\Gamma_1, \ldots, \Gamma_n, \Delta \vdash C}\ 0_E$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \&_I \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A}\ \&_{E_1} \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B}\ \&_{E_2}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus_{I_1} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus_{I_2} \qquad \frac{\Gamma \vdash A \oplus B \quad \Gamma', A \vdash C \quad \Gamma', B \vdash C}{\Gamma, \Gamma' \vdash C}\ \oplus_E$$

Figure 3.8: Proof rules (Intuitionistic Linear Logic).

lambda calculus is an extension of the linear lambda calculus, is there a similar linear logic system that corresponds to it, giving us a quantum variant of the Curry-Howard isomorphism? The type system of the quantum lambda calculus is based on Intuitionistic Linear Logic. However, the quantum lambda calculus only deals with classical control. Can such a correspondence be extended to calculi which combine both quantum and classical control? We attempt to answer this question in Chapter 6.

Ending this chapter with what it began, looks like the right place to stop. We began with the rich framework of the *Curry-Howard-Lambek* correspondence. The beginning of this section addressed the relationship between logic and programming languages. Now, we bring in the third element of this correspondence, within the setting of Linear logic. Intuitionistic logic corresponds to Cartesian Closed categories (CCC), with products in CCCs corresponding to conjunction and exponentials corrresponding to implication. Similarly, the Multiplicative fragment of Linear logic (MLL), corresponds to symmetric monoidal closed categories. The monoidal product of a symmetric monoidal closed category corresponds to the tensor of MLL. Similarly, the linear implication $\multimap$ of MLL, corresponds to the internal hom of SMCCs. Finally, the duality between the $\parr$ and the $\otimes$ in MLL, is expressed in SMCCs, by interpreting the $\parr$ as a dual to the $\otimes$ using the concept of dual objects.

4

Quantum Programming Languages

In this chapter, we discuss all the past research towards developing a programming language for quantum computing. We begin with a brief introduction to the history of quantum programming languages, and then go on to discuss various works such as variants of the λ-calculus designed for quantum computing, circuit description languages, and linear algebraic calculi. We also discuss the two notions of control flow, which offer a broad and informal classification of all these languages. This literature review with a lens of control flow is essential to understand the novelty of the work described in this thesis.

The idea of a *universal quantum computer* was first proposed by David Deutsch in [D85]. He extended the classical *Church-Turing* thesis by claiming that every finitely realizable physical system can be simulated by a universal quantum computer. He also introduced the quantum analogue of the Turing Machine, i.e., the Quantum Turing Machine, which operated on qubits instead of bits and accounted for the central quantum principles of unitary evolution, entanglement and superposition. A universal QTM can simulate any physical system that is based on the laws of quantum mechanics, much like its classical analogue which can simulate any other classical Turing Machine. This work laid the first foundation for quantum computing, and paved the way for the celebrated Shor's algorithm and other more general circuit based quantum computation. It also laid the foundation for the notion of a quantum programming language. [K96] published in 1996, authored by Knill was the first work that laid the groundwork for writing quantum pseudocode and thus birthing a general guideline for the properties a quantum programming language should possess.

The first proper quantum programming language was proposed by Bernhard Ömer in 1998, and appeared in [O03] as a part of his PhD thesis. His language *QCL* is a high-level implemented language, which has a classical sublanguage with complete expressive power. It supports key features such as ensuring reversibility of user-defined quantum operations and management of memory used as "scratch space" for computations. Building on this, [BCS03], proposed a quantum programming architecture with an intermediate assembly like language. The syntax of the language is similar to C++, and unlike QCL, it suggests a design to embed quantum computations into existing classical computing environments. In this work, quantum operators are treated like modules, with entire libraries dedicated to various quantum functionalities. A more theoretical approach was taken in designing qGCL (quantum guarded command language), introduced in [SZ00]. It is a quantum version of the guarded command language introduced by Dijkstra, and works as a specification language for quantum algorithms. Among all the works introduced so far, this is the only language with a formal operational semantics.

A common feature that all these works possess is that they are *imperative* languages. As

opposed to this, the work of this thesis has a distinct *logical* and *functional* flavour. Hence, we discuss all relevant *functional* languages in this chapter. The next section describes the first functional programming language due to Peter Selinger.

4.0.1 Towards a quantum programming language

In this section we give an overview of the ideas introduced in [S04].

The syntax of the language is diagrammatic in nature, and is defined using *quantum flow charts* (QFCs). These diagrams resemble control flow graphs often associated to lower level imperative languages, but differ in a key aspect. The "flow" is governed by a more functional approach, i.e., action of commands is dictated through their modification of an *input* resulting in an *output*, as opposed to them updating global variables. A simple example of such a control flow diagram dealing with only classical information is given in Figure 4.1. The control flow in QFCs is from top to bottom. The edges are labelled by a set of typed variables, which are to be understood as the ones being available at that point in the program. At any point, the *state* of the program is represented by the pair (e, ρ), where e an edge of the flow chart can be intuitively understood to be the program counter, and ρ is an assignment of values to the variables that constitute e. The example in Figure 4.1 is a particularly simple purely classical example of a

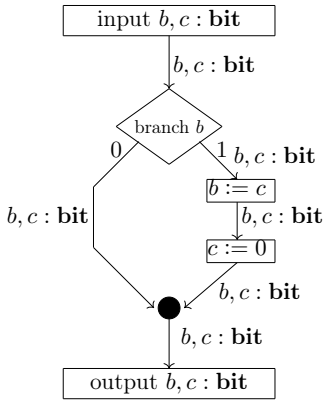


Figure 4.1: A simple quantum control flow.

QCF. Throughout the execution of the program, the two variables of type **bit** (booleans), are not discarded, and hence remain on the edges of the whole diagram. It represents a conditional statement, which modifies the bit value stored in c subject to conditions on the value of b, and returns both the bits. This QCF can be interpreted probabilistically, which aligns better with expression of quantum programs. In order to see this, note that the diagram in Figure 4.1 can be "expanded", so that edges are no longer labelled by variables, but represent the conditional path that the program execution takes depending on the input value of the variables. This view connects the control flow of the program to its classical state. Now we can view the program as a *function* that takes as input a probability distribution on the possible input values, and outputs the corresponding distribution on the possible output values. In the context of the example in Figure 4.1, the program can be understood to represent a function F, that takes as input probabilities A, B, C, D of input values $\mathbf{00}, \mathbf{01}, \mathbf{10}, \mathbf{11}$ respectively, and outputs the probabilities

of the output values. We have $F(A, B, C, D) = F(A + C, B, D, 0)$. In order to incorporate this view, while preventing a blow up in the size of the diagram (which is inevitable with the control flow approach), edges of the diagrams are additionally annotated with tuples of probabilities, i.e., an edge labelled with n booleans, is also annotated with a tuple of 2^n probabilities instead of just 1 probability. The ith coordinate of the tuple, is the probability assigned to the ith state, under the lexicographic ordering of the 2^n possible states that a n bit system can take.

Note that it is important to highlight the distinction between an edge *label* and its *annotation*. The label is a typing judgement which forms a part of the syntax of the language, and the annotation is a tuple of probabilities, which constitute the semantics of the language. There are various rules that dictate the semantics in the language, corresponding to bit allocation, assignment, permutation, and discarding. We omit these rules here.

Quantum control flow charts are a variant of the classical ones described above, with a new *ground* type **qbit**, and corresponding two new operations: *unitary transformations* and *measurement*. The language models unitary transformation as a fixed set of *constants* and crucially relies on the following result, to approximate *any* unitary transformation via these constants:

Proposition 1 (Solovay, Kitaev). *Let $n \geq 2$. for any unitary matrix $S \in \mathbb{C}^{2^n \times 2^n}$ and any $\epsilon > 0$, there exists a unitary matrix S' and a unit complex number λ such that $\|S - \lambda S'\| < \epsilon$ such that S' can be written as a product of matrices of the form $I \otimes A \otimes J$, where I, J are identity matrices of appropriate dimensions and A is one of the gates H ($Hadamard$), V_c ($controlled\ phase\ gate$) or X ($CNOT\ gate$).*

For a single qubit unitary transformation S, a node labelled $q* = S$, denotes the application of S on the qubit q. Similarly, for a n-qubit unitary transformation S, a node labelled $q_1, q_2, \ldots, q_n* = S$, denotes the application of S on the qubits $q_1, q_2, \ldots q_n$. The output of a quantum flow chart is in general a *mixed state*, and hence the semantics of a quantum flow chart can be represented as a function from mixed states to mixed states. Using density matrices to represent mixed states, the semantics of a quantum flow chart are seen as functions acting on density matrices. The language follows the convention that all density matrices are normalised. This has several desirable consequences, the most notable being that each edge becomes a linear function of the input. This in particular, immediately implies that the *no-cloning* property for quantum data is satisfied by the semantics. Note that in contrast to the purely classical flow chart, the annotations of the edges for the quantum flow chart now take the shape of a density matrix. The diagrams corresponding to the allocation of a qubit, its measurement, and the application of a unitary transformation are discussed in detail in [S04] and omitted here.

Moreover, the language supports recursion through *loops*, which are represented in the diagrammatic syntax. We omit the details here. A construct called a *procedure* is a particular fragment of the flow chart with a name and a type and can be understood as a function or a method from an imperative language. Despite its functional flavour, the language doesn't yet support higher order terms. It is a statically-typed first order language which has served as an important stepping stone for other languages that came up later to build on. In order to model procedure calls, the concept of *context extension* is incorporated in flow charts. Context extension refers to the addition of dummy variables to flow charts. Again the semantics of context extension are discussed in detail in the paper and omitted here. Context extension along with a similar treatment as loops enables the language to provide support for recursive procedures as well. Nodes representing recursive procedures are denoted using a different box.

The formal denotational semantics for flow charts are laid down using a special kind of monoidal categories. In order to construct the category formally, consisting of arrows from matrix tuples to matrix tuples, the notion of a *signature* is introduced. A signature is defined to be a

list of non-zero natural numbers $\sigma = n_1, \ldots, n_s$. Associating to each signature, a complex vector space,

$$V_\sigma = \mathbb{C}^{n_1 \times n_1} \times \ldots \times \mathbb{C}^{n_s \times n_s}$$

gives us a vector space V_σ whose elements are tuples of matrices, with dimensions determined by σ. In this setting the type **bit** has signature $1, 1$, whereas the type **qbit** has signature 2. A category V is defined as the category with signatures as objects, and morphisms from σ to σ' to be complex linear functions $F : V_\sigma \to V_{\sigma'}$. In this setting, any morphism F which is completely positive and trace non-increasing is known as a *superoperator*.

The category V and its subcategory Q with superoperators as morphisms satisfy various properties, which ultimately leads one to show that the denotational semantics of the flow charts defined using these categories turns out to be *fully-abstract*, i.e., two program fragments are denotationally equivalent iff they are indistinguishable in the context of any larger program. In this setting flow charts are interpreted as morphisms acting on signatures representing typing-contexts. Addition of a variable to a context is represented by using the monoidal product ($\otimes$) in the category. Moreover, vertical and horizontal composition are interpreted using composition of morphisms and the co-product ($\oplus$) from the category.

Finally, the paper describes *QPL* a language with a more formal syntax, and a formal type system as opposed to the diagrammatic approach taken previously. QPL supports all constructs described in its diagrammatic and more intuitive counterpart. An excerpt of its syntax, is given in Figure 4.2.

$(Variables)$	x, y, z		
$(Procedure\ variables)$	X, Y, Z		
$(Terms)$	P, Q	$::=$	**new bit** $b := 0$ | **new qbit** $q := 0$ | **discard** x
			| $q_1, \ldots, q_n* = S$ | **skip** | $P; Q$ | **measure** q **then** P **else** Q
			| **while** b **do** P
			| **proc** $X : \Gamma \to \Gamma'\{P\}$ **in** Q | $y_1, y_2 \ldots, y_m = X(x_1, x_2, \ldots, x_n)$

Figure 4.2: QPL syntax (excerpt).

Typing rules are laid down using *procedure contexts* (denoted by Π), which is similar to typing contexts, except it consists solely of *procedure variables* and corresponding *procedure types*. A *typing judgement* is an expression of the form,

$$\Pi \vdash \langle \Gamma \rangle P \langle \Gamma' \rangle,$$

where Π is a procedure context and Γ, Γ' are typing contexts. The above judgement is to be understood as P being a well-typed term under the procedure context Π, that acts on a set of variables Γ, and transforms them into a set of variables Γ'. An excerpt of the typing rules is given in Figure 4.3.

$$\frac{}{\Pi \vdash \langle \Gamma \rangle \textbf{new bit } b := 0 \langle b : \textbf{bit}, \Gamma \rangle}\ newbit$$

$$\frac{}{\Pi \vdash \langle \Gamma \rangle \textbf{new qbit } q := 0 \langle q : \textbf{qbit}, \Gamma \rangle}\ newqbit$$

$$\frac{}{\Pi \vdash \langle x : t, \Gamma \rangle \textbf{discard } x \langle \Gamma \rangle}\ discard$$

Figure 4.3: Typing judgements (excerpt).

To summarize, this work serves as an important stepping stone for designing quantum programming languages. It describes a first fully abstract denotational semantics for a first order quantum functional language which supports classical recursion. The work also discusses practical challenges that could arise if one were to implement this language. Thus, it describes both a theoretical framework expressive enough to write important quantum algorithms like the QFT (Quantum Fourier Transform), and gives insight into practical challenges that await implementation of a quantum programming language. Many works that we discuss further ahead in this chapter build on ideas presented here.

4.0.2 Quantum λ-calculi

The quantum λ-calculus is a framework that is an extension of the classical λ-calculus, which incorporates the principles of quantum mechanics into a functional programming language to support quantum computation. One of the earliest works towards designing such a lambda calculus is described in [V04], which introduces an untyped quantum lambda calculus, as an extension of the classical untyped lambda calculus. The calculus introduced in [V04] is often referred to as λ_q, and extends the syntax of the classical lambda calculus, by addition of explicit construct for *linearity* of terms, and modelling of unitary operations as constants. Linearity of terms is introduced to ensure that quantum data does not violate the *no-cloning* theorem. Notably, however λ_q does not model measurement, and relies on the principle of *deferred measurement*, to express quantum algorithms as measurement-free with delayed measurements. The paper proves a formal result on the computational equivalence of λ_q to a quantum Turing machine. We omit the details here, and instead focus on the typed quantum lambda calculus introduced in [SV09].

The quantum lambda calculus introduced in [SV06] is arguably the most influential quantum variant of the simply typed lambda calculus, with various extensions and denotational semantics proposed for it later on. Similar to its untyped counterpart described in [V04], this variant also incorporates linearity of terms in its syntax. This is achieved by adopting a linear typing system. However, unlike λ_q, this linear quantum lambda calculus also models measurement as an explicit term in its syntax, thus successfully supporting both quantum and classical computation. The grammar dictating the type system for the quantum lambda calculus is given by,

$$\text{qType } A, B \ ::= \ \ \alpha \ \mid \ X \ \mid \ !A \ \mid \ (A \multimap B) \ \mid \ \top \ \mid \ (A \otimes B),$$

where α ranges over the type constant *bit*, *qbit*, and X ranges over a countable set of type variables. All types are assumed to be linear, as reflected in the linear function type $A \multimap B$ and the linear product type $A \otimes B$. This design choice immediately prevents duplication functions, and ensures that the no-cloning principle is not violated. $\top$ represents the linear unit type. Hence, by default a type A is assumed to be non-duplicable, and duplicable terms are given by the type $!A$. In addition, there is the subtyping relation $<:$, used to indicate that $n-$fold composition with the !, is always a subtype of a smaller number of compositions of !, and other relations one would expect the types $\multimap, \otimes$ to satisfy. The syntax for the terms is an extension of the intuitionistic lambda calculus with three quantum operations, *new*, *meas* and U (which indicates builtin n-ary unitary gates). *new* models the creation of a qubit from a bit value and is typed as a duplicable function type from *bit* to *qbit*. *meas* models measurement of a single qubit, and is again typed as a duplicable function type from *qbit* to *bit*. Finally, U represents an inbuilt (a constant in the language) n-ary qubit gate which is again typed as a duplicable function type from the n-fold product of qubits to itself. The syntax for terms of the quantum lambda calculus is given in Figure 4.4.

The *program state* in this syntax is represented using a triple $[Q, L, M]$, where,

$$
\begin{array}{lll}
(Variables) & x, y, z, f & \\
(Terms) & M, N \quad ::= & x \mid M\ N \mid \lambda x.M \\
 & & \mid \mathit{if}\ M\ \mathit{then}\ N\ \mathit{else}\ P \mid 0 \mid 1 \\
 & & \mid \mathit{meas} \mid \mathit{new} \mid U \mid * \\
 & & \mid \langle M, N \rangle \mid \mathit{let}\ \langle x, y \rangle = M\ \mathit{in}\ N
\end{array}
$$

Figure 4.4: Terms of the quantum lambda calculus.

- Q is a normalised vector of $\bigotimes_{i=0}^{n-1} \mathbb{C}^2$, for some $n \geq 0$;
- M is a term;
- L is a linking function mapping the distinct free variables in M (denoted by $FV(M)$) to distinct integers in $\{0, \ldots, n-1\}$.

A program state $[Q, L, M]$ is said to be *well-typed* of type B if $\Delta \rhd M : B$ is derivable for $\Delta = \{x : \mathit{qbit} \mid x \in FV(M)\}$. In this case, $[Q, L, M] : B$ is said to be valid. Operational semantics of this calculus is defined probabilistically using the above description of a program state. A call-by-value evaluation strategy is adopted. A reduction rule takes the form $[Q, L, M] \to_p [Q', L', M']$, to mean that the program state $[Q, L, M]$ reduces to the program state $[Q', L', M']$ with probability p. An excerpt of the reduction rules, consisting of those pertaining to quantum operations are described in Figure 4.5.

$$
\begin{aligned}
[Q, L, U\ \langle x_{j_1}, \ldots, x_{j_n} \rangle] &\to_1 [Q', L, \langle x_{j_1}, \ldots, x_{j_n} \rangle] \\
[\alpha\,|Q_0\rangle + \beta\,|Q_1\rangle\,, L, \mathit{meas}\ x_i] &\to_{|\alpha|^2} [Q_0, L', 0] \\
[\alpha\,|Q_0\rangle + \beta\,|Q_1\rangle\,, L, \mathit{meas}\ x_i] &\to_{|\beta|^2} [Q_0, L', 1] \\
[Q, L, \mathit{new}\ 0] &\to_1 [Q \otimes |0\rangle\,, L', x_{n+1}]
\end{aligned}
$$

Figure 4.5: Reduction rules (an excerpt).

The paper proves that the operational semantics satisfy all desirable properties such as type-safety and progress. Furthermore, it discusses a type inference algorithm as well. An extension of this calculus with recursion and sum types is discussed in [SV09]. Additionally, it also discusses extending the type system with inductively defined infinite data types such as lists. It also modifies the notion of a program state to a quantum closure by replacing the linking function with a list of variables, whose order matches with that in which they appear in the term.

More importantly, [SV09] describes a denotational semantics for a core fragment (excludes sum types and recursion) of the quantum lambda calculus they describe. This is done using three separate abstract categories for computations, quantum values and classical values named $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ respectively. Specifically, $\mathcal{B}$ represents the class of quantum computations, which need to be evaluated and result in a probability distribution over values upon evaluation. $\mathcal{C}$ represents the category of quantum values, and therefore in particular those which are typeable by *only* linear types. Finally, $\mathcal{B}$ represents the category of values which are typeable by *only* duplicable types and duplicable variables. Various important properties of these abstract categories are proven and a soundness and completeness result ties the semantics defined using them to an *axiomatic equivalence* relation.

The calculus described in [SV06, SV09] has proven to be seminal in introducing a quantum version of the classical simply typed lambda calculus. Recently, an untyped variant of this calculus and a corresponding rewriting theory inspired from probabilistic rewrite systems has been discussed in [FLV24]. Different denotational models have been described for variants of the calculus described in these papers. Now, we briefly discuss these models.

[PSV14] describes a denotational model of the full quantum lambda calculus described in [SV09] using a category of natural numbers and completely positive maps $F : \mathbb{C}^{n^2} \to \mathbb{C}^{m^2}$ for natural numbers n, m called **CPM**. [SV08] provides a fully abstract model for a finitary, strictly linear version of the quantum lambda calculus using **CPM**. However, **CPM** cannot represent infinite data types, or non-linear functions. [PSV14] describes a suitable extension of the category, to obtain an adequate denotational model for the finitary version of the quantum lambda calculus.

[CDW19, CD19, D20] describe one of the first *fully abstract* denotational models for the full quantum lambda calculus described in [SV09] using *game semantics*. The game semantics approach models computations through an interactive view. The idea is to model any computation as an interaction between a Player (the program) and their Opponent (the environment). In this setting, moves correspond to computational steps. Types are thus viewed as games, and programs are viewed as *strategies*-permissible ways for a Player to respond to an Opponent's moves. The works described in these papers, imposes the structure of a compact closed category on concurrent games and strategies, and interprets the quantum lambda calculus in this category, called **QCG**. We omit the details of the model here.

[CW16] describes a more physically intuitive denotational semantics for the recursion-free version of the quantum lambda calculus with finite data types. It lays down an adequate model of the quantum lambda calculus using two different categories of von Neumann algebras and adjunctions between them. The model is based on the Heisenberg view of quantum mechanics.

4.0.3 Linear algebraic calculi & Proof languages

Some of the first works to discuss the idea of a lambda calculus endowed with a vector space structure on terms were by Arrighi & Dowek [AD05, AD06] and Lionel Vaux [V07a, V09].

The line of work by Arrighi & Dowek in [AD05, AD06] discussed a rewrite system for an untyped version of the lambda-calculus, by augmenting the syntax for terms with terms of the form $\alpha t + \beta u$, where α, β are scalars from an appropriate ring. Additionally, they discuss several possible rewrite systems for the scalars, and discuss how these rewrite systems interact with those for terms of the calculus. This research culminated in the work *Lineal* [AD17], where the authors proposed an untyped calculus, with rewrite systems adopting the call-by-value (CBV) strategy so that the resulting calculus is a higher-order language expressive enough to encode linear operators. The authors discuss the prospect of encoding quantum computing in Lineal, by formally proving that quantum circuits can be encoded in the language. They also prove the system to be confluent for a combination of rewrite systems, for the scalars and the terms of the language. Confluence of such systems was a major challenge that they resolved in this work. However, the calculi proposed in these works are untyped, and incorporating a type system with support for types beyond the simply-typed lambda calculi remains an interesting question.

Lionel Vaux proposed a typed version of such an algebraic calculus, adopting the call-by-name (CBN) strategy for reduction. He extended the line of work for algebraic lambda calculi by proving normalization for the simply typed algebraic lambda calculus he proposed. Furthermore, he tied the algebraic linearity of the syntax to the resource sensitivity introduced in Linear logic [G87] by Girard. He observed that the resource sensitive (linear in the sense of Linear logic)

use of variables in β-reduction also obeys the laws of linearity associated to vector spaces. Thus, his work proved to be seminal in tying together programming languages, logic and linear algebra. The calculus he introduced is general and is expressive enough to encode linear, probabilistic as well as quantum computation. The type system he introduced is however simple, and does not incorporate sum types or product types.

[ADPTV14] provides a great summary of various algebraic $\lambda-$calculi, with various reduction strategies. They compare the works discussed above, and their equivalence (or not) in terms of their expressive power. All the works discussed here however, are at best, simply typed. The question of augmenting the simply typed algebraic lambda calculus with sum types or product types is not discussed. More recently, [DGMV19] presents a realizability semantics for a linear algebraic lambda calculus. This work is particularly relevant to quantum computation as it introduces a type system which captures the notion of "unit" vectors (i.e. normalized vectors with norm 1 in the ℓ^2-norm) as well as that of unitarity of linear operators. Furthermore, it discusses a typed lambda calculus which not only has support for linear combination of terms, but also introduces sum types and product types. Using the realizability semantics model for this linear algebraic calculus, they define the semantics of a quantum lambda calculus that they call λ_Q and introduce in one of the latter sections of the paper. The operational semantics for λ_Q is defined using quantum configurations like those for the quantum lambda calculi discussed in §4.0.2. Notably, however neither the linear algebraic lambda calculus nor λ_Q support the measurement operation. [DM22] builds on this work by introducing a linear algebraic lambda calculus called Lambda$-S_1$, which incorporates modalities on types introduced in [DGMV19], into a lambda calculus, by modifying the notion of orthogonality of terms first introduced in [DGMV19]. [DM22] additionally defines a denotational model for Lambda$-S_1$. It is a calculus focused on pure state quantum computation, with quantum control.

[DD24] discusses a linear algebraic calculus which also serves as a proof language for IMALL- the Intuitionistic Multiplicative and Additive fragment of Linear Logic. The type system incorporates the connectives $\multimap, \&, \oplus$ and $\otimes$ from Linear Logic, and corresponding proof terms. Additionally, it supports linear combinations of terms through the symbols $+, \bullet$. Commutation with these symbols is ensured for some terms through the reduction relation defined for this calculus. The system is called the $\mathcal{L}^{\mathcal{S}}$-calculus, and algebraic linearity of function application is an important result of the paper. Furthermore, the authors prove that quantum computation can be encoded in this calculus, by providing an appropriate encoding of vectors and matrices as terms of the calculus. Again, this work does not support the measurement operation. Note that measurement cannot be simply encoded into linear algebraic calculi where all functions are inherently linear, since measurement when viewed as a linear operator on vector spaces fails to be linear.

The works [DCD17, DCDR19] are the first works to resolve the question of incorporating measurement in linear algebraic calculi. They do so by introducing a new modality $\mathcal{S}$ in the type system for their calculus called Lambda$-\mathcal{S}$. Intuitively, for a type A, the type $\mathcal{S}(A)$ represents terms which are a linear combination of *base terms* of type A. Base terms are defined in the grammar for the syntax of the language they introduce. These papers explain an appropriate notion of duality, according to which $\mathcal{S}$ is a dual to the !- modality from Intuitionistic Linear Logic. [DCM20, DCM19, DCM24] provide different denotational models for Lambda-S.

[DCD23] introduces a new connective $\odot$ in Intuitionistic Propositional logic and provides a proof language for this fragment. They also discuss a probabilistic denotational model. However, the connection of the denotational model to quantum computing is not clear. A variant of the proof language in [DD24] augmented with the $\odot$ connective, called the $\mathcal{L}\odot$ calculus is discusses in the latter sections of the same work.

4.0.4 Symmetric pattern-matching and quantum control

In this section we discuss the work from [SVV18], and then its adaptation which formed Section 2 in [DLPZ25]. We discuss all the technical details of this section, since [DLPZ25] is a combination of the work done there and Chapter 5 of this thesis.

[SVV18] first discusses what it means for control flow in a program to be quantum, and then ventures into possible "levels" of quantum control. The most primitive being standard conditional statements, and the most complete being an appropriate notion of "quantum recursion". The paper first describes a preliminary first order language with a construct called an *Iso*, on which the entire development to define and implement quantum control rests. Building on this language, it develops a complete reversible functional language which incorporates quantum control in its entirety, i.e., also supports "quantum recursion". The syntax for this language is given in Figure 4.6. As the name suggests, an Iso is a term construct meant to represent reversible oper-

$$
\begin{array}{lllll}
(Variables) & x, y, z, f & & & \\
(Val\ \&\ term\ types) & a, b & ::= & \mathrm{I} \mid a \oplus b \mid a \otimes b \mid [a] \\
(Iso\ types) & \mathrm{T} & ::= & a \leftrightarrow b \mid (a \leftrightarrow b) \to \mathrm{T} \\
 & & & & \\
(Values) & v & ::= & () \mid x \mid \mathit{inl}(v) \mid \mathit{inr}(v) \mid \langle v_1, v_2 \rangle \\
(Products) & p & ::= & () \mid x \mid \langle p_1, p_2 \rangle \\
(Extended\ Values) & e & ::= & v \mid \text{let } p_1 = \omega\ p_2 \text{ in } e \\
(Isos) & \omega & ::= & \{\mid v_1 \leftrightarrow e_1 \mid v_2 \leftrightarrow e_2 \cdots \} \mid \lambda f.\omega \mid \\
 & & & \mu f.\omega \mid f \mid \omega_1\ \omega_2 \\
(Terms) & t & ::= & () \mid x \mid \mathit{inl}(t) \mid \mathit{inr}(t) \mid \langle t_1, t_2 \rangle \mid \\
 & & & \omega\ t \mid \text{let } p = t_1 \text{ in } t_2
\end{array}
$$

Figure 4.6: Syntax.

ations, not necessarily quantum in nature. The typing judgements and the construction for such a term, is inspired from the pattern-matching constructs presented in functional programming languages. A predicate called *Orthogonal Decomposition*, is defined to intuitively "decompose" the type space into "orthogonal" subspaces. The core concept used here is that of linear independence and spanning sets for vector spaces. Isos are denoted by the Greek letter ω, and the formation rule for such a construct is given as follows:

$$
\frac{\begin{array}{lll}
\Delta_1 \vdash_v v_1 : a \cdots \Delta_n \vdash_v v_n : a & \forall i \neq j, v_i \perp v_j & dim(a) = n \\
\Delta_1 \vdash_v v_1' : b \cdots \Delta_n \vdash_v v_n' : b & \forall i \neq j, v_i' \perp v_j' & dim(b) = n
\end{array}}{\vdash_\omega \{\mid v_1 \leftrightarrow v_1' \mid v_2 \leftrightarrow v_2' \cdots \} : a \leftrightarrow b}
$$

The above rule constructs an Iso ω, which maps a value v_i to v_i', and conversely its *inverse* maps v_i' to v_i. Its invertibility is guaranteed by two requirements in the rule, one is the *orthogonality* of values given by $v_i \perp v_j$ for all $i \neq j$, and the other is exhaustivity of the type spaces, given by the requirements $dim(a) = n$ and $dim(b) = n$. Orthogonality of values is defined by a set of rules, some of which are given in Figure 4.7. Using these rules, a predicate called *orthogonal decomposition* denoted by OD is defined on sets of values. We omit it here, and describe a detailed modified version that we use in [DLPZ25]. $dim(\cdot)$ denotes the dimension of the type space and is defined inductively and represents the number of elements in the type for finite types. The syntax for the higher order reversible programming language defined in the paper is given in Figure 4.6.

$$\frac{}{inl(v_1) \perp inr(v_2)} \qquad \frac{}{inr(v_1) \perp inl(v_2)}$$

$$\frac{v_1 \perp v_2}{inr(v_1) \perp inr(v_2)} \qquad \frac{v_1 \perp v_2}{inl(v_1) \perp inl(v_2)} \qquad \frac{v_1 \perp v_2}{\langle v, v_1\rangle \perp \langle v', v_2\rangle} \qquad \frac{v_1 \perp v_2}{\langle v_1, v\rangle \perp \langle v_2, v'\rangle}$$

Figure 4.7: Orthogonality of values.

Finally, the operational semantics are defined in a small-step style using the notion of variable valuation, which is a partial map from a finite set of variables to a set of values. Using these rules an execution of the inverse of an Iso ω can also be defined. The definition is a bit subtle when higher order terms are involved, and so we omit it here. This is formally stated and proven as a theorem.

Theorem 5. *The inversion operation is well-typed, in the sense that if* $f_1 : a_1 \leftrightarrow b_1, \ldots, f_n : a_n \leftrightarrow b_n \vdash_\omega \omega : \mathrm{T}$ *then we also have* $f_1 : b_1 \leftrightarrow a_1, \ldots, f_n : b_n \leftrightarrow a_n \vdash_\omega \omega^{-1} : \mathrm{T}^{-1}$.

Moreover, another theorem captures the fact that this inverse operationally behaves as expected.

Theorem 6. *Consider a well-typed, closed iso* $\vdash_\omega \omega : a \leftrightarrow b$*, and suppose that* $\vdash_v v : a$ *and that* $\vdash_v w : b$*, then* $\omega^{-1}\ (\omega\ v) \to_* v$ *and* $\omega\ (\omega^{-1} w) \to_* w$.

4.0.5 A calculus for quantum control

Here we describe Section 2 of [DLPZ25] in detail. All the details are also given in [L24]. [DLPZ25] is a combination of Chapter 3 from [L24] and Chapter 5 of this thesis. The syntax of the language described in [L24] is a smaller set of the syntax of the language in [SVV18], and hence supports a weaker version of quantum control. However, the operational semantics are described as an equational theory, and unlike [SVV18], this language has a denotational semantics which is shown to be sound and complete with respect to the equational theory. The formation rules that define the predicate *orthogonal decomposition*, ensure non-overlapping patterns as in [SVV18]. There are two *kinds* of types in this language. The first kind, referred to as *pure quantum types* in [DLPZ25] are defined by the following grammar:

$$\boldsymbol{Q} \ ::= \ \boldsymbol{I} \mid \boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2 \mid \boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2 \mid \mathbf{qnat}.$$

These can be understood as quantum analogues of standard types, with qubits being represented by the type $\mathbf{qbit} := \boldsymbol{I} \oplus \boldsymbol{I}$. $\mathbf{qbit}^n$ represents the tensor product $\mathbf{qbit} \otimes \cdots \otimes \mathbf{qbit}$ of n qubits. **qnat** is a ground type of *quantum natural numbers*, and this addition allows one to abstract away from quantum circuits, to a more high-level approach to writing quantum programs.

The second kind of types denoted by $U(\boldsymbol{Q}_1, \boldsymbol{Q}_2)$ are used to represent unitary function spaces with domain represented by a pure quantum type $\boldsymbol{Q}_1$ and the codomain represented by a pure quantum type $\boldsymbol{Q}_2$. They are referred to as *unitary types* in [DLPZ25]. The term grammar has syntactic support for superposition and is given in Figure 4.8. The syntax for terms can be understood as follows. The most general kinds of terms in this language are the pure terms, and symbols like $\boldsymbol{t}$ are used to range over them. The symbols $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ denote variables. $\mathbf{inl}_{\boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2}\ \boldsymbol{t}$ and $\mathbf{inr}_{\boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2}\ \boldsymbol{t}$ represent sum type injections, and, for brevity, writing the subscripts is omitted. The term $\boldsymbol{t}_1 \otimes \boldsymbol{t}_2$ is used for forming pairs of terms. The terms $\mathbf{0}$ and $\mathbf{S}\ \boldsymbol{t}$ represent the qnat $|0\rangle$ and the qnat $|n+1\rangle$, provided that $\boldsymbol{t}$ represents the qnat $|n\rangle$. Here, $|\underline{\mathbf{n}}\rangle := \mathbf{S^n}\ \mathbf{0}$ and $\left|\underline{\boldsymbol{t}+\mathbf{n}}\right\rangle := \mathbf{S^n}\ \boldsymbol{t}$,

$$
\begin{array}{rlll}
(\textbf{\textit{Basis Terms}}) & \boldsymbol{b} & ::= & \boldsymbol{x} \mid * \mid \mathbf{inl}\ \boldsymbol{b} \mid \mathbf{inr}\ \boldsymbol{b} \mid \boldsymbol{b}_1 \otimes \boldsymbol{b}_2 \mid \mathbf{0} \mid \mathbf{S}\ \boldsymbol{b} \\
(\textbf{\textit{Values}}) & \boldsymbol{v} & ::= & \sum_{i\in\mathcal{I}} \boldsymbol{\alpha}_i \cdot \boldsymbol{b}_i \\
(\textbf{\textit{Unitaries}}) & u & ::= & \{ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1 \ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n \ \} \mid u_1 \oplus u_2 \mid u_1 \otimes u_2 \mid u_2 \circ u_1 \mid u^* \mid \mathrm{ctrl}\ u \\
(\textbf{\textit{Terms}}) & \boldsymbol{t} & ::= & \boldsymbol{x} \mid * \mid \mathbf{inl}\ \boldsymbol{t} \mid \mathbf{inr}\ \boldsymbol{t} \mid \boldsymbol{t}_1 \otimes \boldsymbol{t}_2 \mid \mathbf{0} \mid \mathbf{S}\ \boldsymbol{t} \mid u\,\boldsymbol{t} \mid \textstyle\sum_{i\in\mathcal{I}} \boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i
\end{array}
$$

Figure 4.8: The term grammar.

for $\mathbf{n} \in \mathbb{N}$. The term $u\,\boldsymbol{t}$ represents the application of a unitary u to a term $\boldsymbol{t}$. Superpositions are represented by the term $\sum_{i\in\mathcal{I}} \boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i$. Conditions that ensure normalisation of these terms are defined further along in this section. In Figure 4.8, the ***Basis Terms*** (ranged over by symbols $\boldsymbol{b}, \boldsymbol{b}_i$) are the simplest kind of pure terms. The ***Values***, ranged over by $\boldsymbol{v}$ are the normal forms in this language. They are of the form $\sum_{i\in\mathcal{I}} \boldsymbol{\alpha}_i \cdot \boldsymbol{b}_i$, where $\boldsymbol{\alpha}_i \in \mathbb{C}$, $\boldsymbol{b}_i$ are basis terms, and $\mathcal{I}$ is a finite index set. $\mathcal{I}$ is assumed to be a totally ordered set to ensure existence of unique normal forms. The expressions $\boldsymbol{\alpha}_1 \cdot \boldsymbol{b}_1 + \cdots + \boldsymbol{\alpha}_n \cdot \boldsymbol{b}_n$ and $\sum_{i\in\{1,\ldots,n\}} \boldsymbol{\alpha}_i \cdot \boldsymbol{b}_i$ are used interchangeably. The formation conditions ensure that all terms in the sums are pairwise distinct, orthogonal, and of the same type, and thus well-formed values correspond to *normalised* vectors. The equational theory for this language ensures that every term $\boldsymbol{t}$ can be rewritten to a normalised value, $\boldsymbol{v}$.

Example 71. The basis term $*\colon \boldsymbol{I}$ represents the complex scalar $1 \in \mathbb{C}$. The basis terms that represent the computational basis for qubits can be defined by $|\mathbf{0}\rangle := \mathbf{inl}\ *\colon \mathbf{qbit}$ and $|\mathbf{1}\rangle := \mathbf{inr}\ *\colon \mathbf{qbit}$. Other basis terms that can be defined include $|\mathbf{00}\rangle := |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle : \mathbf{qbit}^2$ and $|\mathbf{11}\rangle := |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle : \mathbf{qbit}^2$. If $\boldsymbol{b}$ is a basis term, it is often written as $|\boldsymbol{b}\rangle$ to distinguish it from other (non-basis) terms, e.g., $|\underline{\mathbf{2}}\rangle : \mathbf{qnat}$ represents the qnat $|2\rangle \in \ell^2(\mathbb{N})$. Single-qubit states may be defined by $|\pm\rangle := (1/\sqrt{2} \cdot |\mathbf{0}\rangle \pm 1/\sqrt{2} \cdot |\mathbf{1}\rangle) : \mathbf{qbit}$. (Entangled) quantum states can be defined, e.g., $\mathbf{Bell} := (1/\sqrt{2} \cdot |\mathbf{00}\rangle + 1/\sqrt{2} \cdot |\mathbf{11}\rangle) : \mathbf{qbit}^2$. Linear combinations of qnats can also be written in the language, *i.e.* $(1/\sqrt{2} \cdot |\underline{\mathbf{0}}\rangle + 1/\sqrt{6} \cdot |\underline{\mathbf{1}}\rangle + 1/\sqrt{3} \cdot |\underline{\mathbf{2}}\rangle) : \mathbf{qnat}$.

Unitaries u of Figure 4.8 are based on *unitary pattern-matching* $\{ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1 \ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n \ \}$. This allows one to build unitary maps out of basis terms and values: every *closed* basis term $\boldsymbol{b}_i$ is mapped to the closed value $\boldsymbol{v}_i$; basis terms with free variables $\boldsymbol{b}_i$ determine a mapping through the use of substitution with respect to the corresponding value $\boldsymbol{v}_i$, wherein both $\boldsymbol{b}_i$ and $\boldsymbol{v}_i$ have the same free variables occurring within them, and the induced mapping is determined by performing all possible substitutions of the free variables on both sides with the same closed basis terms. The formation conditions ensure that the basis terms $\boldsymbol{b}_i$ (resp. the values $\boldsymbol{v}_i$) determine an ONB for the type $\boldsymbol{Q}_1$ (resp. $\boldsymbol{Q}_2$) and therefore $\{ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1 \ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n \ \}$ determines a unitary map. There is syntactic sugar for quantum if statements, written **qif**. If u_1 is of the form $\{ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1 \ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n \ \}$ and u_2 of the form $\{ \mid \boldsymbol{b}'_1 \mapsto \boldsymbol{v}'_1 \ \ldots \mid \boldsymbol{b}'_m \mapsto \boldsymbol{v}'_m \ \}$ then $\mathbf{qif}\ \boldsymbol{x}\ \mathbf{then}\ \boldsymbol{x} \otimes u_2\ \mathbf{else}\ \boldsymbol{x} \otimes u_1$ is short for the unitary:

$$
\left\{ \begin{array}{lcl} |\mathbf{0}\rangle \otimes \boldsymbol{b}_1 & \mapsto & |\mathbf{0}\rangle \otimes \boldsymbol{v}_1 \\ \vdots & & \\ |\mathbf{0}\rangle \otimes \boldsymbol{b}_n & \mapsto & |\mathbf{0}\rangle \otimes \boldsymbol{v}_n \end{array} \right. , \left. \begin{array}{lcl} |\mathbf{1}\rangle \otimes \boldsymbol{b}'_1 & \mapsto & |\mathbf{1}\rangle \otimes \boldsymbol{v}'_1 \\ \vdots & & \\ |\mathbf{1}\rangle \otimes \boldsymbol{b}'_m & \mapsto & |\mathbf{1}\rangle \otimes \boldsymbol{v}'_m \end{array} \right\}
$$

Example 72. The unitaries Had and CNOT below encode the standard Hadamard and CNOT gates, respectively. CNOT makes use of some simple pattern-matching on its last line: it can match either with $|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle$ or with $|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle$. The unitary $\mathrm{Had}_{\mathbf{qnat}}$ defines an extension of the Hadamard gate on the space $\ell^2(\mathbb{N})$.

$$\mathrm{Had} : U(\mathbf{qbit}, \mathbf{qbit})$$
$$\mathrm{Had} := \left\{ \begin{array}{lcl} |\,|\mathbf{0}\rangle & \mapsto & {}^1\!/\!\sqrt{2} \cdot |\mathbf{0}\rangle + {}^1\!/\!\sqrt{2} \cdot |\mathbf{1}\rangle \\ |\,|\mathbf{1}\rangle & \mapsto & {}^1\!/\!\sqrt{2} \cdot |\mathbf{0}\rangle - {}^1\!/\!\sqrt{2} \cdot |\mathbf{1}\rangle \end{array} \right\}$$

$$\mathrm{CNOT} : U(\mathbf{qbit}^2, \mathbf{qbit}^2)$$
$$\mathrm{CNOT} := \left\{ \begin{array}{lcl} |\,|\mathbf{1}\rangle \otimes |\mathbf{0}\rangle & \mapsto & |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \\ |\,|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle & \mapsto & |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle \\ |\,|\mathbf{0}\rangle \otimes |\boldsymbol{x}\rangle & \mapsto & |\mathbf{0}\rangle \otimes |\boldsymbol{x}\rangle \end{array} \right\}$$

$$\mathrm{Had}_{\mathbf{qnat}} : U(\mathbf{qnat}, \mathbf{qnat})$$
$$\mathrm{Had}_{\mathbf{qnat}} := \left\{ \begin{array}{lcl} |\,|\underline{\mathbf{0}}\rangle & \mapsto & {}^1\!/\!\sqrt{2} \cdot |\underline{\mathbf{0}}\rangle + {}^1\!/\!\sqrt{2} \cdot |\underline{\mathbf{1}}\rangle \\ |\,|\underline{\mathbf{1}}\rangle & \mapsto & {}^1\!/\!\sqrt{2} \cdot |\underline{\mathbf{0}}\rangle - {}^1\!/\!\sqrt{2} \cdot |\underline{\mathbf{1}}\rangle \\ |\,|\underline{\boldsymbol{x}+\mathbf{2}}\rangle & \mapsto & |\underline{\boldsymbol{x}+\mathbf{2}}\rangle \end{array} \right\}$$

Furthermore, the controlled Hadamard operator can be represented in this syntax by ctrl Had and the Toffoli, with ctrl CNOT. One can also easily define permutations. This becomes important once we introduce the calculus in Chapter 5. For instance, the following unitary

$$\begin{aligned} &\mathrm{SWAP} : U(\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2, \boldsymbol{Q}_2 \otimes \boldsymbol{Q}_1) \\ &\mathrm{SWAP} := \{\ |\ \boldsymbol{x} \otimes \boldsymbol{y} \ \mapsto\ \boldsymbol{y} \otimes \boldsymbol{x}\ \} \end{aligned}$$

defines a unitary swap operator where the types $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ may be chosen arbitrarily.

The remaining expressions for forming unitaries are easy to understand: $u_2 \circ u_1$ represents composition of unitaries, u^* represents the adjoint of a unitary, $u_1 \otimes u_2$ and $u_1 \oplus u_2$ represent tensor products and direct sums of unitaries, and finally, ctrl u represents a qubit-controlled unitary operator. For simplicity, we sometimes write $\{\ldots \mid \boldsymbol{b}_i \mapsto \boldsymbol{v}_i \ldots\}$ as a shorthand notation for the unitary $\{\ |\ \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1\ \ldots |\ \boldsymbol{b}_n \mapsto \boldsymbol{v}_n\ \}$, when n is clear from context or unimportant. We also point out that unitaries depend only on values and basis terms and that general pure terms cannot be used for the construction of unitaries.

Formation Conditions. In order to guarantee (linear-algebraic) normalisation for quantum terms and unitarity for functions, this work formulates: (1) a suitable *orthogonality relation* $\perp \subseteq \boldsymbol{Terms} \times \boldsymbol{Terms}$, which is used as part of the formation condition of some terms; (2) additional predicates $\mathrm{ONB}_{\boldsymbol{Q}}$ and $\mathrm{ONB}^{val}_{\boldsymbol{Q}}$ which ensure that a set of basis terms/values of type $\boldsymbol{Q}$ determines an ONB which is used as part of the formation conditions for the atomic introduction rule of unitaries.

The relation $\perp$ is defined as the smallest binary relation $\perp \subseteq \boldsymbol{Terms} \times \boldsymbol{Terms}$ that is closed under the rules in Figure 4.9. Quantum terms that are orthogonal via this relation are orthogonal in a denotational sense as well. This relation allows us to enforce the orthogonality of terms appearing within a superposition $\sum_{i \in \mathcal{I}} (\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)$. Under the assumption that all terms $\boldsymbol{t}_i$ represent normalised vectors that are pairwise orthogonal, the norm of the superposition is easy to calculate (statically), and it is given by $\sum_{i \in \mathcal{I}} |\boldsymbol{\alpha}_i|^2$. This justifies the formulation of the introduction rule for superposition (Figure 4.11) and guarantees that the vector represented by this superposition of terms is normalised.

Example 73. The rules for the orthogonality relation are not restricted to closed quantum terms, e.g., $\mathbf{0} \perp \mathbf{S}\,\boldsymbol{x}$ and $\mathbf{inl}\,\boldsymbol{x} \perp \mathbf{inr}\,\boldsymbol{y}$. One can also show that $|+\rangle \perp |-\rangle$ (defined in Example 71). However, we cannot show that Had $|\mathbf{0}\rangle \perp |-\rangle$ even though Had $|\mathbf{0}\rangle$ is equal to $|+\rangle$ with respect to our equational logic and also with respect to our denotational semantics (both introduced later). The reason for this is that the orthogonality relation is to be defined *statically*.

In order to determine whether a finite set S of (potentially open) basis terms determines an ONB for a type $\boldsymbol{Q}$, we introduce the *orthonormal basis predicates* $\mathrm{ONB}_{\boldsymbol{Q}}(S)$ in Figure 4.10,

$$\frac{}{\mathbf{inl}\ t_1 \perp \mathbf{inr}\ t_2} \qquad \frac{}{\mathbf{inr}\ t_1 \perp \mathbf{inl}\ t_2} \qquad \frac{t_1 \perp t_2}{\mathbf{inl}\ t_1 \perp \mathbf{inl}\ t_2} \qquad \frac{t_1 \perp t_2}{\mathbf{inr}\ t_1 \perp \mathbf{inr}\ t_2}$$

$$\frac{t_1 \perp t_2}{t \otimes t_1 \perp t' \otimes t_2} \qquad \frac{t_1 \perp t_2}{t_1 \otimes t \perp t_2 \otimes t'} \qquad \frac{}{\mathbf{0} \perp \mathbf{S}\ t} \qquad \frac{}{\mathbf{S}\ t \perp \mathbf{0}}$$

$$\frac{t_1 \perp t_2}{\mathbf{S}\ t_1 \perp \mathbf{S}\ t_2} \qquad \frac{t_1 \perp t_2}{u\ t_1 \perp u\ t_2} \qquad \frac{\forall i.\ t \perp t_i}{t \perp \sum_{i\in\mathcal{I}}(\alpha_i \cdot t_i)} \qquad \frac{\forall i.\ t_i \perp t}{\sum_{i\in\mathcal{I}}(\alpha_i \cdot t_i) \perp t}$$

$$\frac{\forall i.\ t \perp t_i \qquad \alpha_* = 0 \qquad t_* = t}{t \perp \sum_{i\in\mathcal{I}\cup\{*\}}(\alpha_i \cdot t_i)} \qquad \frac{\forall i.\ t_i \perp t \qquad \alpha_* = 0 \qquad t_* = t}{\sum_{i\in\mathcal{I}\cup\{*\}}(\alpha_i \cdot t_i) \perp t}$$

$$\frac{\forall j \neq k.\ t_j \perp t_k \qquad \mathcal{J},\mathcal{K} \subseteq \mathcal{I} \qquad \sum_{i\in\mathcal{J}\cup\mathcal{K}} \overline{\alpha_i}\beta_i = 0}{\sum_{j\in\mathcal{J}}(\alpha_j \cdot t_j) \perp \sum_{k\in\mathcal{K}}(\beta_k \cdot t_k)}$$

Figure 4.9: Rules for the orthogonality relation.

$$\frac{}{\mathrm{ONB}_{\boldsymbol{Q}}(\{x\})} \qquad \frac{}{\mathrm{ONB}_{\boldsymbol{I}}(\{*\})} \qquad \frac{\mathrm{ONB}_{\boldsymbol{Q}_1}(S) \qquad \mathrm{ONB}_{\boldsymbol{Q}_2}(T)}{\mathrm{ONB}_{\boldsymbol{Q}_1\oplus\boldsymbol{Q}_2}(\{\mathbf{inl}\ b \mid b \in S\} \cup \{\mathbf{inr}\ b \mid b \in T\})}$$

$$\frac{\mathrm{ONB}_{\boldsymbol{Q}_i}(\pi_i(S)) \qquad \forall b \in \pi_i(S),\ \mathrm{ONB}_{\boldsymbol{Q}_j}(S^i_b)}{\mathrm{ONB}_{\boldsymbol{Q}_1\otimes\boldsymbol{Q}_2}(S)}\ i,j \in \{1,2\}, i \neq j$$

$$\frac{\mathrm{ONB}_{\mathbf{qnat}}(S)}{\mathrm{ONB}_{\mathbf{qnat}}(\{\mathbf{0}\} \cup \{\mathbf{S}\ b \mid b \in S\})} \qquad \frac{(\alpha_{b,b'})_{(b,b')\in S\times S} \text{ is a unitary matrix} \quad \mathrm{ONB}_{\boldsymbol{Q}}(S)}{\mathrm{ONB}^{val}_{\boldsymbol{Q}}(\{\sum^*_{b'\in S}\ \alpha_{b,b'} \cdot b' \mid b \in S\})}$$

Figure 4.10: Orthonormal basis predicates.

originally known as *orthogonal decomposition* [SVV18]. Note that there is a mistake with the rule for the tensor product in the original paper, *e.g.* it does not allow for the typing of a CNOT.

Given a type $\boldsymbol{Q}$ and a finite set of basis terms S, we say that S determines an *orthonormal basis of* $\boldsymbol{Q}$, which we write as $\mathrm{ONB}_{\boldsymbol{Q}}(S)$, if this can be derived via the rules in Figure 4.10. We now explain some of the notation used therein. If S is a set of the form $\{b_1 \otimes b'_1, \ldots, b_n \otimes b'_m\}$ then we define $\pi_1(S) := \{b_1, \ldots, b_n\}$ and $\pi_2(S) := \{b'_1, \ldots, b'_m\}$. In this situation, we also define $S^1_b := \{b' \mid b \otimes b' \in S\}$ and $S^2_{b'} := \{b \mid b \otimes b' \in S\}$. Furthermore, the ONB predicate implies pairwise orthogonality of the associated values. Note that the predicate $\mathrm{ONB}_{\boldsymbol{Q}}(S)$ is defined for a finite set S consisting of *basis* terms. But, in order to be able to define unitaries that make use of superposition, we also work with other ONBs. Given a finite set S consisting of *values*, we

$$\frac{\Vdash u\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2) \quad \boldsymbol{\Gamma}\vdash \boldsymbol{t}\colon \boldsymbol{Q}_1}{\boldsymbol{\Gamma}\vdash u\,\boldsymbol{t}\colon \boldsymbol{Q}_2} \qquad \frac{\forall k,\ \boldsymbol{\Gamma}\vdash \boldsymbol{t}_k\colon \boldsymbol{Q} \quad \forall k\neq j,\ \boldsymbol{t}_k\perp \boldsymbol{t}_j \quad \sum_{i\in\mathcal{I}}|\boldsymbol{\alpha}_i|^2=1}{\boldsymbol{\Gamma}\vdash \sum_{i\in\mathcal{I}}\boldsymbol{\alpha}_i\cdot\boldsymbol{t}_i\colon \boldsymbol{Q}}$$

$$\frac{\begin{array}{ll}\forall i,\ \boldsymbol{\Gamma}_i\vdash \boldsymbol{b}_i\colon \boldsymbol{Q}_1 & \mathrm{ONB}_{\boldsymbol{Q}_1}\{\boldsymbol{b}_1,\dots,\boldsymbol{b}_n\}\\ \forall i,\ \boldsymbol{\Gamma}_i\vdash \boldsymbol{v}_i\colon \boldsymbol{Q}_2 & \mathrm{ONB}^{val}_{\boldsymbol{Q}_2}\{\boldsymbol{v}_1,\dots,\boldsymbol{v}_n\}\end{array}}{\Vdash \{\ |\ \boldsymbol{b}_1\mapsto\boldsymbol{v}_1\ \dots\ |\ \boldsymbol{b}_n\mapsto\boldsymbol{v}_n\ \}\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2)} \qquad \frac{\Vdash u_1\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2) \quad \Vdash u_2\colon U(\boldsymbol{Q}_3,\boldsymbol{Q}_4)}{\Vdash u_1\oplus u_2\colon U(\boldsymbol{Q}_1\oplus\boldsymbol{Q}_3,\boldsymbol{Q}_2\oplus\boldsymbol{Q}_4)}$$

$$\frac{\Vdash u_1\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2) \quad \Vdash u_2\colon U(\boldsymbol{Q}_3,\boldsymbol{Q}_4)}{\Vdash u_1\otimes u_2\colon U(\boldsymbol{Q}_1\otimes\boldsymbol{Q}_3,\boldsymbol{Q}_2\otimes\boldsymbol{Q}_4)} \qquad \frac{\Vdash u_1\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2) \quad \Vdash u_2\colon U(\boldsymbol{Q}_2,\boldsymbol{Q}_3)}{\Vdash u_2\circ u_1\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_3)}$$

$$\frac{\Vdash u\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2)}{\Vdash u^*\colon U(\boldsymbol{Q}_2,\boldsymbol{Q}_1)} \qquad \frac{\Vdash u\colon U(\boldsymbol{Q},\boldsymbol{Q})}{\Vdash \mathrm{ctrl}\ u\colon U(\mathbf{qbit}\otimes\boldsymbol{Q},\mathbf{qbit}\otimes\boldsymbol{Q})}$$

Figure 4.11: Formation rules for terms and unitaries (excerpt).

say that S determines an ONB of type $\boldsymbol{Q}$, and we write $\mathrm{ONB}^{val}_{\boldsymbol{Q}}(S)$ to indicate this, whenever this can be derived via the rules in Figure 4.10. We also introduce syntactic sugar: the value $\sum^*_{i\in\mathcal{I}}\boldsymbol{\alpha}_i\cdot\boldsymbol{b}_i$ is the finite linear combination where elements with scalar 0 are omitted. In other words, the indices $i\in\mathcal{I}$ such that $\boldsymbol{\alpha}_i=0$ are ignored, e.g., if $\boldsymbol{\alpha}_1=1$ and $\boldsymbol{\alpha}_2=0$, $\sum^*_{i\in\{1,2\}}\boldsymbol{\alpha}_i\cdot\boldsymbol{b}_i$ is the value $\sum_{i\in\{*\}}1\cdot\boldsymbol{b}_1$.

Example 74. For **qbit**, the rules from Figure 4.10 show that $\mathrm{ONB}_{\mathbf{qbit}}(\{|\mathbf{0}\rangle, |\mathbf{1}\rangle\})$ and that $\mathrm{ONB}^{val}_{\mathbf{qbit}}(\{|+\rangle, |-\rangle\})$ hold (with $|\pm\rangle$ defined in Example 71). We also have that $\mathrm{ONB}_{\mathbf{qbit}}(\{\boldsymbol{x}\})$, where $\boldsymbol{x}$ is a variable. In other words, we determine an ONB for the type **qbit** by substituting $\boldsymbol{x}$ with all possible closed computational *basis* terms of type **qbit** (see Figure 4.8). For **qnat**, any set S with the property that $\mathrm{ONB}_{\mathbf{qnat}}(S)$ holds, must contain a term that is not closed. We have that $\mathrm{ONB}_{\mathbf{qnat}}(\{|\underline{\mathbf{x}}\rangle\})$, $\mathrm{ONB}_{\mathbf{qnat}}(\{|\underline{\mathbf{0}}\rangle, |\underline{\boldsymbol{x}+\mathbf{1}}\rangle\})$, and $\mathrm{ONB}_{\mathbf{qnat}}(\{|\underline{\mathbf{0}}\rangle, |\underline{\mathbf{1}}\rangle, |\underline{\boldsymbol{x}+\mathbf{2}}\rangle\})$ hold true and determine the same ONB for the type **qnat** (it represents the computational basis of $\ell^2(\mathbb{N})$). The set used in the last predicate allows us to conclude that the predicate,

$$\mathrm{ONB}^{val}_{\mathbf{qnat}}\big(\{{}^1\!/\!\sqrt{2}\cdot|\underline{\mathbf{0}}\rangle+{}^1\!/\!\sqrt{2}\cdot|\underline{\mathbf{1}}\rangle, {}^1\!/\!\sqrt{2}\cdot|\underline{\mathbf{0}}\rangle-{}^1\!/\!\sqrt{2}\cdot|\underline{\mathbf{1}}\rangle, |\underline{\boldsymbol{x}+\mathbf{2}}\rangle\}\big)$$

holds. This last set of values is used to type $\mathrm{Had}_{\mathbf{qnat}}$ from Example 72.

Judgements for pure quantum terms have the form $\boldsymbol{\Gamma}\vdash\boldsymbol{t}\colon\boldsymbol{Q}$ where $\boldsymbol{\Gamma}$ stands for a linear context of the form $\boldsymbol{\Gamma}:=\boldsymbol{x}_1\colon\boldsymbol{Q}_1,\dots,\boldsymbol{x}_n\colon\boldsymbol{Q}_n$. The formation rules for terms are presented in Figure 4.11; the first two lines are standard for a linearly-typed term calculus. The rule for superposition depends on the orthogonality relation $\perp$, and the premise ensures that the resulting superposition corresponds to a normalised vector. This is made precise later (Theorem 7). Finally, the formation rule for $u\,\boldsymbol{t}$, i.e. the application of a unitary u to a term $\boldsymbol{t}$, is akin to function application in lambda-calculi. However, the complexity is primarily relegated to the judgement $\Vdash u\colon U(\boldsymbol{Q}_1,\boldsymbol{Q}_2)$, which ensures the unitarity of u.

The formation rules for unitaries are presented in Figure 4.11. In the introduction rule for atomic unitaries $\{\ |\ \boldsymbol{b}_1\mapsto\boldsymbol{v}_1\ \dots\ |\ \boldsymbol{b}_n\mapsto\boldsymbol{v}_n\ \}$, we have that the potentially open basis terms $\{\boldsymbol{b}_1,\dots,\boldsymbol{b}_n\}$ (resp. values $\{\boldsymbol{v}_1,\dots,\boldsymbol{v}_n\}$) determine an ONB for $\boldsymbol{Q}_1$ (resp. $\boldsymbol{Q}_2$) in the sense explained above. Then the resulting bijective mapping that associates $\boldsymbol{b}_i$ to $\boldsymbol{v}_i$ uniquely determines a unitary. The formation rules for the remaining unitaries are straightforward, and they align easily with the mathematical intuition that motivates them. We also wish to note that unitaries may depend on basis terms and values, but not on general terms (see Figure 4.8).

Denotational Semantics. The denotational semantics of the language are laid out using two categories, **Isometry**- the category of Hilbert spaces and isometries, and **Unitary**- the category of Hilbert spaces and unitaries. For the category **Hilb** - the category of Hilbert spaces and bounded linear maps, we have that the following inclusion relations hold: **Unitary** $\subseteq$ **Isometry** $\subseteq$ **Hilb**.

The interpretation of a pure quantum type $\boldsymbol{Q}$ is a Hilbert space $[\![\boldsymbol{Q}]\!]$, i.e., an object of the category **Isometry**, and the interpretation of a unitary type $U(\boldsymbol{Q}_1, \boldsymbol{Q}_2)$ is the set $[\![U(\boldsymbol{Q}_1, \boldsymbol{Q}_2)]\!]$ of unitaries, $[\![\boldsymbol{Q}_1]\!] \to [\![\boldsymbol{Q}_2]\!]$, which are morphisms in **Unitary**.

A linear context $\boldsymbol{\Gamma} := \boldsymbol{x}_1 : \boldsymbol{Q}_1, \boldsymbol{x}_2 : \boldsymbol{Q}_2, \ldots, \boldsymbol{x}_n : \boldsymbol{Q}_n$ is interpreted in the usual way $[\![\boldsymbol{\Gamma}]\!] := [\![\boldsymbol{Q}_1]\!] \otimes \cdots \otimes [\![\boldsymbol{Q}_n]\!]$. The interpretation of a judgement $\boldsymbol{\Gamma} \vdash \boldsymbol{t} \colon \boldsymbol{Q}$ is given by a morphism $[\![\boldsymbol{\Gamma} \vdash \boldsymbol{t} \colon \boldsymbol{Q}]\!] \in \mathbf{Isometry}([\![\boldsymbol{\Gamma}]\!], [\![\boldsymbol{Q}]\!])$. The notations $[\![\boldsymbol{t}]\!]$ and $[\![\boldsymbol{\Gamma} \vdash \boldsymbol{t} : \boldsymbol{Q}]\!]$ are used interchangeably. For closed quantum terms $\cdot \vdash \boldsymbol{t} \colon \boldsymbol{Q}$, the empty context is interpreted as $\mathbb{C}$ (the tensor unit in **Isometry**) so that $[\![\cdot \vdash \boldsymbol{t} \colon \boldsymbol{Q}]\!](1)$ can be identified with a normalised vector.

The basis terms $\boldsymbol{b}$ (Figure 4.8) have standard formation conditions. They depend neither on unitaries, nor on sums, so if $\boldsymbol{\Gamma} \vdash \boldsymbol{b} \colon \boldsymbol{Q}$ is a well-formed basis term, then $[\![\boldsymbol{b}]\!]$ is an isometry; moreover, for any pure quantum type $\boldsymbol{Q}$, the set $\{[\![\boldsymbol{b}]\!](1) \mid \cdot \vdash \boldsymbol{b} \colon \boldsymbol{Q}\}$ is an ONB for $[\![\boldsymbol{Q}]\!]$. This is proven formally as a proposition.

Proposition 2. *If $\boldsymbol{\Gamma} \vdash \boldsymbol{b} \colon \boldsymbol{Q}$ is a well-formed basis term, then $[\![\boldsymbol{b}]\!]$ is an isometry. Moreover, for any pure quantum type $\boldsymbol{Q}$, the set $\{[\![\boldsymbol{b}]\!](1) \mid \cdot \vdash \boldsymbol{b} \colon \boldsymbol{Q}\}$ is an ONB for the Hilbert space $[\![\boldsymbol{Q}]\!]$.*

Every value $\boldsymbol{v}$ is of the form $\sum_{i \in \mathcal{I}} \boldsymbol{\alpha}_i \cdot \boldsymbol{b}_i$, but the sum is subject to strict admissibility conditions imposed by the orthogonality relation $\perp$ from 4.9. The denotational significance of this relation is provided by the statement: given $\boldsymbol{\Gamma}_1 \vdash \boldsymbol{t}_1 \colon \boldsymbol{Q}$ and $\boldsymbol{\Gamma}_2 \vdash \boldsymbol{t}_2 \colon \boldsymbol{Q}$, if $\boldsymbol{t}_1 \perp \boldsymbol{t}_2$ then $[\![\boldsymbol{t}_1]\!]^* [\![\boldsymbol{t}_2]\!] = 0_{[\![\boldsymbol{\Gamma}_2]\!], [\![\boldsymbol{\Gamma}_1]\!]}$; which is central in proving that the interpretation of values is well-defined as isometries.

Proposition 3. *Given $\boldsymbol{\Gamma}_1 \vdash \boldsymbol{t}_1 \colon \boldsymbol{Q}$ and $\boldsymbol{\Gamma}_2 \vdash \boldsymbol{t}_2 \colon \boldsymbol{Q}$, if $\boldsymbol{t}_1 \perp \boldsymbol{t}_2$ then $[\![\boldsymbol{t}_1]\!]^* [\![\boldsymbol{t}_2]\!] = 0_{[\![\boldsymbol{\Gamma}_2]\!], [\![\boldsymbol{\Gamma}_1]\!]}$.*

Moreover, if $\boldsymbol{\Gamma} \vdash \boldsymbol{v} \colon \boldsymbol{Q}$ is a well-formed value, then $[\![\boldsymbol{\Gamma} \vdash \boldsymbol{v} \colon \boldsymbol{Q}]\!] \in \mathbf{Isometry}([\![\boldsymbol{\Gamma}]\!], [\![\boldsymbol{Q}]\!])$. Unitaries are interpreted as morphisms in **Unitary**. The most challenging unitaries to interpret are the unitary pattern-matchings, constructed through the use of basis terms and values (but not general terms) which we already know are interpreted as isometries. Another key role is played by the $\mathrm{ONB}_{\boldsymbol{Q}}$ and $\mathrm{ONB}^{val}_{\boldsymbol{Q}}$ predicates: they can be understood as determining suitable ONBs of the corresponding types. The denotational significance of these predicates is that they give us a resolution of the identity at type $\boldsymbol{Q}$, as made precise by the following proposition.

Proposition 4. *Let $\boldsymbol{\Gamma}_1 \vdash \boldsymbol{v}_1 \colon \boldsymbol{Q}, \ldots, \boldsymbol{\Gamma}_n \vdash \boldsymbol{v}_n \colon \boldsymbol{Q}$ be well-formed values. If $\mathrm{ONB}^{\mathrm{val}}_{\boldsymbol{Q}}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$, then $\mathrm{id}_{[\![\boldsymbol{Q}]\!]} = \sum_{i=1}^{n} [\![\boldsymbol{v}_i]\!] \circ [\![\boldsymbol{v}_i]\!]^*$.*

Pattern-matching is then interpreted by $[\![\Vdash \{\ldots \mid \boldsymbol{b}_i \mapsto \boldsymbol{v}_i \ldots\}]\!] := \sum_i [\![\boldsymbol{v}_i]\!] \circ [\![\boldsymbol{b}_i]\!]^*$, and the previous proposition ensures it is unitary. Finally, we can show that the interpretation of all pure quantum terms is well-defined as isometries.

Theorem 7. *If $\boldsymbol{\Gamma} \vdash \boldsymbol{t} \colon \boldsymbol{Q}$, then $[\![\boldsymbol{t}]\!] \in \mathbf{Isometry}([\![\boldsymbol{\Gamma}]\!], [\![\boldsymbol{Q}]\!])$.*

Moreover, any canonical symmetric monoidal isomorphism can be represented by the unitaries of our language. The canonical symmetric monoidal isomorphisms are simply those that can be defined using the structure of a symmetric monoidal category only, *i.e.* isomorphisms that are given by: left and right unitors, symmetric swaps, associators, closed under composition and tensor products. This result is important for the development in Chapter 5.

Proposition 5. *Let $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ be two pure types, and let $f\colon [\![\boldsymbol{Q}_1]\!] \to [\![\boldsymbol{Q}_2]\!]$ be a canonical symmetric monoidal isomorphism. Then, there exists a unitary $\Vdash u\colon U(\boldsymbol{Q}_1, \boldsymbol{Q}_2)$ such that $[\![u]\!] = f$.*

The interpretations of terms and unitaries are given in Figure 4.12.

$$
\begin{aligned}
[\![\boldsymbol{\Gamma} \vdash \mathit{inl}\boldsymbol{t}\colon \boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2]\!] &:= \mathrm{inl} \circ [\![\boldsymbol{t}]\!] & [\![\boldsymbol{x}\colon \boldsymbol{Q} \vdash \boldsymbol{x}\colon \boldsymbol{Q}]\!] &:= \mathrm{id}\\
[\![\boldsymbol{\Gamma} \vdash \mathit{inr}\boldsymbol{t}\colon \boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2]\!] &:= \mathrm{inr} \circ [\![\boldsymbol{t}]\!] & [\![\cdot \vdash *\colon \boldsymbol{I}]\!] &:= \mathrm{id}\\
[\![\boldsymbol{\Gamma} \vdash \boldsymbol{t}_1 \otimes \boldsymbol{t}_2\colon \boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2]\!] &:= [\![\boldsymbol{t}_1]\!] \otimes [\![\boldsymbol{t}_2]\!] & [\![\cdot \vdash \mathbf{0}\colon \mathbf{qnat}]\!] &:= (1 \mapsto |0\rangle)\\
[\![\boldsymbol{\Gamma} \vdash \mathbf{S}\ \ \boldsymbol{t}\colon \mathbf{qnat}]\!] &:= s \circ [\![\boldsymbol{t}]\!] & [\![\boldsymbol{\Gamma} \vdash u\,\boldsymbol{t}\colon \boldsymbol{Q}]\!] &:= [\![u]\!] \circ [\![\boldsymbol{t}]\!]\\
\left[\!\!\left[\boldsymbol{\Gamma} \vdash \textstyle\sum_{i\in\mathcal{I}} \boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i\colon \boldsymbol{Q}\right]\!\!\right] &:= \textstyle\sum_{i\in\mathcal{I}} \boldsymbol{\alpha}_i [\![\boldsymbol{t}_i]\!] & &
\end{aligned}
$$

$$
\begin{aligned}
[\![\Vdash \{\ldots \mid \boldsymbol{b}_i \mapsto \boldsymbol{v}_i \ldots\}]\!] &:= \textstyle\sum_i [\![\boldsymbol{v}_i]\!] \circ [\![\boldsymbol{b}_i]\!]^* & [\![\Vdash u_1 \oplus u_2]\!] &:= [\![u_1]\!] \oplus [\![u_2]\!]\\
[\![\Vdash u_1 \otimes u_2]\!] &:= [\![u_1]\!] \otimes [\![u_2]\!] & [\![\Vdash u_2 \circ u_1]\!] &:= [\![u_2]\!] \circ [\![u_1]\!]\\
[\![\Vdash \mathrm{ctrl}\ u]\!] &:= |0\rangle\langle 0| \otimes \mathrm{id} + |1\rangle\langle 1| \otimes [\![u]\!] & [\![\Vdash u^*]\!] &:= [\![u]\!]^*
\end{aligned}
$$

Figure 4.12: Interpretation of pure quantum terms and unitaries.

Equational Theory. In this section we describe the equational theory which defines the operational semantics of the language. Given two pure quantum terms $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$, $\boldsymbol{\Gamma} \vdash \boldsymbol{t}_1 = \boldsymbol{t}_2\colon \boldsymbol{Q}$ indicates that $\boldsymbol{t}_1$ equals $\boldsymbol{t}_2$ as terms of type $\boldsymbol{Q}$ in context $\boldsymbol{\Gamma}$. As usual if $\boldsymbol{\Gamma} \vdash \boldsymbol{t}_1 = \boldsymbol{t}_2\colon \boldsymbol{Q}$, the rules imply that $\boldsymbol{\Gamma} \vdash \boldsymbol{t}_1\colon \boldsymbol{Q}$ and $\boldsymbol{\Gamma} \vdash \boldsymbol{t}_2\colon \boldsymbol{Q}$. The rules of the equational theory are divided into four groups. The first one is concerned with the usual rules for reflexivity, symmetry and transitivity. The second set handles linear algebraic identities, such as exchanging sums and the linearity of injections and unitaries. Congruence rules comprise the third group, e.g., $\boldsymbol{\Gamma} \vdash \mathbf{S}\,\boldsymbol{t}_1 = \mathbf{S}\,\boldsymbol{t}_2\colon \mathbf{qnat}$ given that $\boldsymbol{\Gamma} \vdash \boldsymbol{t}_1 = \boldsymbol{t}_2\colon \mathbf{qnat}$. Last but not least are the *computational* rules, which, read left to right, can be seen as providing an operational semantics. The most important rule and our version of β-reduction, is the rule for unitary pattern-matching applied to a basis term.

$$
\frac{\cdot \vdash \boldsymbol{b}'\colon \boldsymbol{Q}_1 \qquad \Vdash \{\ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1\ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n\ \}\colon U(\boldsymbol{Q}_1, \boldsymbol{Q}_2) \qquad \mathtt{s}(\boldsymbol{b}_j) = \boldsymbol{b}'}{\cdot \vdash \{\ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1\ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n\ \}\,\boldsymbol{b}' = \mathtt{s}(v_j)\colon \boldsymbol{Q}_2}
$$

The formation conditions on unitary pattern-matching ensure that $\mathrm{ONB}_{\boldsymbol{Q}_1}(\{\boldsymbol{b}_i\}_i)$. With the latter, and given $\cdot \vdash \boldsymbol{b}'\colon \boldsymbol{Q}_1$, there exists a unique substitution $\mathtt{s}$ and a unique $\boldsymbol{b}_j$ such that $\mathtt{s}(\boldsymbol{b}_j) = \boldsymbol{b}'$. The resulting term of the rule is then the same substitution applied to the corresponding output $\boldsymbol{v}_j$. Such substitutions were first defined in [SVV18], and can be deduced via an inference system.

Proposition 6. *Given $\cdot \vdash \boldsymbol{t}\colon \boldsymbol{Q}$, there is a unique value $\boldsymbol{v}$ s.t. $\cdot \vdash \boldsymbol{t} = \boldsymbol{v}\colon \boldsymbol{Q}$.*

This proposition shows that pure terms have a unique *normal form*. It is crucial for the proof of the completeness theorem, and comes in handy in the rest of the paper, especially to define the reduction system for the main calculus (Chapter 5,§5.2.1).

The main equational theory for the language is given in Figure 4.16, with the rules of reflexivity, symmetry and transitivity in Figure 4.13, linear algebraic identities in Figure 4.15, and congruence identities in Figure 4.14.

$$\frac{\Gamma \vdash t : Q}{\Gamma \vdash t = t : Q} \qquad \frac{\Gamma \vdash t_1 = t_2 : Q}{\Gamma \vdash t_2 = t_1 : Q} \qquad \frac{\Gamma \vdash t_1 = t_2 : Q \qquad \Gamma \vdash t_2 = t_3 : Q}{\Gamma \vdash t_1 = t_3 : Q}$$

Figure 4.13: Basic equational rules.

Remark 30. Among the equational rules presented in this section, only the equations in Figure 4.16 provide an operational account of the language. They can be seen as a reduction system from left to right. On the other hand, the equations in Figure 4.15 show that the algebraic power of Hilbert spaces and isometries can be captured within the type system.

$$\frac{\Gamma \vdash t_1 = t_2 : Q_1}{\Gamma \vdash \mathbf{inl}\, t_1 = \mathbf{inl}\, t_2 : Q_1 \oplus Q_2} \qquad \frac{\Gamma \vdash t_1 = t_2 : Q_2}{\Gamma \vdash \mathbf{inr}\, t_1 = \mathbf{inr}\, t_2 : Q_1 \oplus Q_2}$$

$$\frac{\Gamma \vdash t_1 = t_2 : Q_1 \qquad \Gamma' \vdash t : Q_2}{\Gamma, \Gamma' \vdash t_1 \otimes t = t_2 \otimes t : Q_1 \otimes Q_2} \qquad \frac{\Gamma \vdash t_1 = t_2 : Q_2 \qquad \Gamma' \vdash t : Q_1}{\Gamma, \Gamma' \vdash t \otimes t_1 = t \otimes t_2 : Q_1 \otimes Q_2}$$

$$\frac{\Gamma \vdash t_1 = t_2 : \mathbf{qnat}}{\Gamma \vdash \mathbf{S}\, t_1 = \mathbf{S}\, t_2 : \mathbf{qnat}} \qquad \frac{\Gamma \vdash t_1 = t_2 : Q_1 \qquad \Vdash u : U(Q_1, Q_2)}{\Gamma \vdash u t_1 = u t_2 : Q_2}$$

$$\frac{\Gamma \vdash \sum_i (\alpha_i \cdot t_i) : Q \qquad \forall i, \Gamma \vdash t_i = t_i' : Q}{\Gamma \vdash \sum_i (\alpha_i \cdot t_i) = \sum_i (\alpha_i \cdot t_i') : Q}$$

Figure 4.14: Congruence equational rules.

Finally, the denotational semantics are cemented with the equational theory, through a soundness and completeness result.

Theorem 8 (Completeness). $\cdot \vdash t_1 = t_2 : Q$ *iff* $[\![\cdot \vdash t_1 : Q]\!] = [\![\cdot \vdash t_2 : Q]\!]$.

4.0.6 Other languages

[PPRZ20] extends QPL, a quantum programming language introduces in [S04], with inductive data types, becoming one of the first quantum programming languages to do so. Further, it provides a denotational semantics based on W^*-algebras (Definition 51), and proves that this denotational semantics is adequate with respect to the operational semantics. Since it is an extension of QPL, the syntax is an extended version of the one discussed in §4.0.1. [TA24] is another quantum programming language which introduces recursive types. Notably, the language discussed in [PPRZ20] is an extension of the QPL, and hence introduces computations in the syntax using the syntactic construct of procedures, while [TA24] is an extension of the quantum lambda calculus with recursive types, and thus also incorporates lambda abstractions into its

$$\frac{\ell\colon \{1,\dots,n\} \to \{1,\dots,n\} \text{ is a bijection} \qquad \boldsymbol{\Gamma} \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q}}{\boldsymbol{\Gamma} \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i) = \sum_i(\boldsymbol{\alpha}_{\ell(i)} \cdot \boldsymbol{t}_{\ell(i)})\colon \boldsymbol{Q}}$$

$$\frac{\boldsymbol{\Gamma} \vdash \sum_{i=1}^{n}(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q} \qquad \boldsymbol{\alpha}_n = 0}{\boldsymbol{\Gamma} \vdash \sum_{i=1}^{n}(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i) = \sum_{i=1}^{n-1}(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q}} \qquad \frac{\boldsymbol{\Gamma} \vdash \boldsymbol{t}\colon \boldsymbol{Q}}{\boldsymbol{\Gamma} \vdash \sum_{i=1}^{1}(1 \cdot \boldsymbol{t}) = \boldsymbol{t}\colon \boldsymbol{Q}}$$

$$\frac{\boldsymbol{\Gamma} \vdash \sum_i \left(\boldsymbol{\alpha}_i \cdot \sum_j(\boldsymbol{\beta}_{ij} \cdot \boldsymbol{t}_j)\right) : \boldsymbol{Q}}{\boldsymbol{\Gamma} \vdash \sum_i \left(\boldsymbol{\alpha}_i \cdot \sum_j(\boldsymbol{\beta}_{ij} \cdot \boldsymbol{t}_j)\right) = \sum_j((\sum_i \boldsymbol{\alpha}_i\boldsymbol{\beta}_{ij}) \cdot \boldsymbol{t}_j)\colon \boldsymbol{Q}}$$

$$\frac{\boldsymbol{\Gamma} \vdash \sum_{ij}(\boldsymbol{\alpha}_i\boldsymbol{\beta}_{ij} \cdot \boldsymbol{t}_{ij})\colon \boldsymbol{Q}}{\boldsymbol{\Gamma} \vdash \sum_i \left(\boldsymbol{\alpha}_i \cdot \sum_j(\boldsymbol{\beta}_{ij} \cdot \boldsymbol{t}_{ij})\right) = \sum_{ij}(\boldsymbol{\alpha}_i\boldsymbol{\beta}_{ij} \cdot \boldsymbol{t}_{ij})\colon \boldsymbol{Q}}$$

$$\frac{\Vdash u\colon U(\boldsymbol{Q}_1, \boldsymbol{Q}_2) \qquad \boldsymbol{\Gamma} \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q}_1}{\boldsymbol{\Gamma} \vdash u\sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i) = \sum_i(\boldsymbol{\alpha}_i \cdot u\boldsymbol{t}_i)\colon \boldsymbol{Q}_2}$$

$$\frac{\boldsymbol{\Gamma} \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q}_1}{\boldsymbol{\Gamma} \vdash \mathbf{inl}\ \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i) = \sum_i(\boldsymbol{\alpha}_i \cdot \mathbf{inl}\ \boldsymbol{t}_i)\colon \boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2}$$

$$\frac{\boldsymbol{\Gamma}_1 \vdash \boldsymbol{t}\colon \boldsymbol{Q}_1 \qquad \boldsymbol{\Gamma}_2 \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q}_2}{\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2 \vdash \boldsymbol{t} \otimes (\sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)) = \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t} \otimes \boldsymbol{t}_i)\colon \boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2}$$

$$\frac{\boldsymbol{\Gamma}_1 \vdash \boldsymbol{t}\colon \boldsymbol{Q}_2 \qquad \boldsymbol{\Gamma}_2 \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \boldsymbol{Q}_1}{\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2 \vdash (\sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)) \otimes \boldsymbol{t} = \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i \otimes \boldsymbol{t})\colon \boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2}$$

$$\frac{\boldsymbol{\Gamma} \vdash \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i)\colon \mathbf{qnat}}{\boldsymbol{\Gamma} \vdash \mathbf{S}\ \sum_i(\boldsymbol{\alpha}_i \cdot \boldsymbol{t}_i) = \sum_i(\boldsymbol{\alpha}_i \cdot \mathbf{S}\ \boldsymbol{t}_i)\colon \mathbf{qnat}}$$

Figure 4.15: Vector space and linear applications equational rules.

syntax. [TA24] also defines a denotational model for their calculus, which they call the Quantum FPC (Fixpoint Calculus) based on the category of modules over superoperators. They show this model to be fully abstract with respect to the operational semantics, which is defined using

$$\frac{\vdash \boldsymbol{b}' : \boldsymbol{Q}_1 \qquad \Vdash \{ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1 \ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n \} : U(\boldsymbol{Q}_1, \boldsymbol{Q}_2) \qquad \mathtt{s}(\boldsymbol{b}_j) = \boldsymbol{b}'}{\vdash \{ \mid \boldsymbol{b}_1 \mapsto \boldsymbol{v}_1 \ \ldots \mid \boldsymbol{b}_n \mapsto \boldsymbol{v}_n \} \ \boldsymbol{b}' = \mathtt{s}(\boldsymbol{b}_j) : \boldsymbol{Q}_2}$$

$$\frac{\vdash u\boldsymbol{b} = \boldsymbol{v} : \boldsymbol{Q}_2}{\vdash u^{-1}\boldsymbol{v} = \boldsymbol{b} : \boldsymbol{Q}_1} \qquad \frac{\Vdash u_1 : U(\boldsymbol{Q}_1, \boldsymbol{Q}_2) \qquad \Vdash u_2 : U(\boldsymbol{Q}_2, \boldsymbol{Q}_3) \qquad \boldsymbol{\Gamma} \vdash \boldsymbol{b} : \boldsymbol{Q}_1}{\boldsymbol{\Gamma} \vdash (u_2 \circ u_1)\boldsymbol{b} = u_2(u_1\boldsymbol{b}) : \boldsymbol{Q}_3}$$

$$\frac{\Vdash u_1 : U(\boldsymbol{Q}_1, \boldsymbol{Q}_3) \qquad \Vdash u_2 : U(\boldsymbol{Q}_2, \boldsymbol{Q}_4) \qquad \vdash \boldsymbol{b}_1 : \boldsymbol{Q}_1 \quad \vdash \boldsymbol{b}_2 : \boldsymbol{Q}_2}{\vdash (u_1 \otimes u_2)\ (b_1 \otimes \boldsymbol{b}_2) = (u_1\boldsymbol{b}_1) \otimes (u_2\boldsymbol{b}_2) : \boldsymbol{Q}_3 \otimes \boldsymbol{Q}_4}$$

$$\frac{\Vdash u_1 : U(\boldsymbol{Q}_1, \boldsymbol{Q}_3) \qquad \Vdash u_2 : U(\boldsymbol{Q}_2, \boldsymbol{Q}_4) \qquad \vdash \boldsymbol{b} : \boldsymbol{Q}_1}{\vdash (u_1 \oplus u_2)(\mathbf{inl}\ \boldsymbol{b}) = \mathbf{inl}\ (u_1\boldsymbol{b}) : \boldsymbol{Q}_3 \oplus \boldsymbol{Q}_4}$$

$$\frac{\Vdash u_1 : U(\boldsymbol{Q}_1, \boldsymbol{Q}_3) \qquad \Vdash u_2 : U(\boldsymbol{Q}_2, \boldsymbol{Q}_4) \qquad \vdash \boldsymbol{b} : \boldsymbol{Q}_2}{\vdash (u_1 \oplus u_2)\ (\mathbf{inr}\ \boldsymbol{b}) = \mathbf{inr}\ (u_1\boldsymbol{b}) : \boldsymbol{Q}_3 \oplus \boldsymbol{Q}_4}$$

$$\frac{\Vdash u : U(\boldsymbol{Q}, \boldsymbol{Q}) \qquad \vdash \boldsymbol{b} : \boldsymbol{Q}}{\vdash (\mathrm{ctrl}\ u)\ ((\mathbf{inl}\ *) \otimes \boldsymbol{b}) = (\mathbf{inl}\ *) \otimes \boldsymbol{b} : (I \oplus I) \otimes \boldsymbol{Q}}$$

$$\frac{\Vdash u : U(\boldsymbol{Q}, \boldsymbol{Q}) \qquad \vdash \boldsymbol{b} : \boldsymbol{Q}}{\vdash (\mathrm{ctrl}\ u)\ ((\mathbf{inr}\ *) \otimes \boldsymbol{b}) = (\mathbf{inr}\ *) \otimes (u\ b) : (I \oplus I) \otimes \boldsymbol{Q}}$$

Figure 4.16: Computational equational rules of the pure subsystem.

quantum configurations discussed in §4.0.2.

[JKLMZ22] further extends the type system for purely quantum types with recursive types, and provides a language that supports hybrid quantum and classical computation. It supports both classical and quantum recursive types, with an operational semantics defined using quantum configurations, similar to the quantum lambda calculi discussed in §4.0.2. This work also defines an adequate denotational model based on enrichment of an appropriate category of von Neumann algebras (Definition 24) on an appropriate category of DCPOs (Definition 39). and monads.

Finally, [VLRH23] is a recent work which also combines quantum and classical control. It is a compilation system, with no formal operational semantics. The syntax achieves this unification through a syntax based on a try-ctach mechanism, with two different higher-order types representing quantum and classical computations respectively. Projective measurement is implemented as a program using this try catch mechanism. The work also describes a denotational model using Kraus operators. However, this work does not support recursive types.

5

Combining quantum and classical control

In this chapter, we describe one of the original contributions of this thesis. The idea is to design a programming language that combines quantum and classical control, thus enabling writing programs which allow quantum and classical data to interact.The way this is achieved is by "embedding" the quantum control fragment, into the main calculus through the introduction of a new modality $\mathcal{B}(\cdot)$, in the type system. We embed the quantum control fragment, which is the language described in §4.0.4 of Chapter 4. The need for such a language is best understood

$$\begin{aligned} &\lambda y.let\ \mathcal{B}(x \otimes xs) = Tof(y) \\ &\quad in\ case\ meas(x)\ \ of \\ &\{inl(*) \to \mathcal{B}(\text{CNOT})(xs), \\ &\quad inr(*) \to xs\} \end{aligned}$$

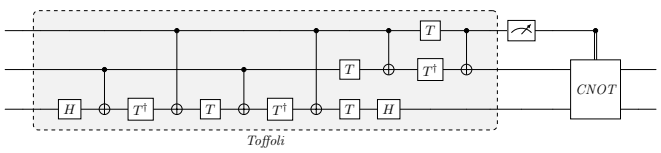


Figure 5.1: A program and the corresponding circuit.

through an example. Consider the circuit in Figure 5.1. It uses the well-known *Toffoli* gate, which would have to be decomposed as a large circuit like the one in the figure. However, the program representing this circuit can instead be written as a simple lambda term in our language given in Figure 5.1, where $Tof := \mathcal{B}(\text{ctrl CNOT})$. To highlight this further note that the controlled Hadamard can be defined similarly to the Toffoli in our language, but a typical classically controlled quantum programming language would require two H gates, six T gates and one $CNOT$ gate. Consider yet another well known quantum gate, known as the *Fredkin* gate. The Fredkin gate represents the controlled SWAP gate, and the simplest version acts on 3 qubits. Similar to the Toffoloi gate, it finds widespread use in various quantum algorithms. Any current implementation of the Fredkin gate requires a circuit composed of one S gate, two H gates, seven T gates and eight $CNOT$ gates. Once again, this is for the simplest case of 3 qubits. Whereas, in our language, the Fredkin gate can be written as a single unitary for any n.

Thus, our language offers an ease of writing programs by abstracting away quantum circuits. We aim to shift focus from drawing quantum circuits, which is a more *low-level* approach to writing code, to writing programs directly using syntax, which is a more *high-level* approach. In classical (non-quantum) programming, we can use high-level languages to write algorithms without having to specify boolean circuits or to encode integers using tuples of bits/booleans.

With our paper, we are trying to advance quantum programming abstractions towards such a direction. For example, an individual qnat $|n\rangle$ can be used instead of a tensor of qubits $|q_1 q_2 ... q_k\rangle$ whose binary encoding corresponds to n. We view boolean circuits and binary encodings of integers as low-level and likewise for their quantum counterparts. Our aim is to develop higher-level quantum abstractions.

Because of the potential presence of quantum entanglement, it is impossible (in general) to decompose a quantum state into a non-trivial tensor product of two smaller quantum states. Indeed, in the quantum lambda-calculus (QLC) [PSV14], which is based on classical control, the type **qbit** of qubits is an opaque type in the sense that there are no *closed* values of such a type. Program states may be described via *quantum configurations* which are triples $(|\psi\rangle, \ell, M)$, where $|\psi\rangle \in \mathbb{C}^{2^n}$ is a *pure* quantum state (possibly entangled); M is a program possibly containing free variables of type **qbit**; ℓ is a *linking* function that maps the free quantum variables of M to appropriate components of the state $|\psi\rangle$. It is possible to also view ℓ as a unitary permutation acting on $|\psi\rangle$ and this view is important for our development. Such configurations, even though they are not part of the user-facing syntax, allow us to reason about entangled states and the operational semantics are described via a relation $(|\psi\rangle, \ell, M) \to_p (|\psi'\rangle, \ell', M')$, with p the probability of reduction.

The main idea that allows us to combine both quantum and classical control is to modify the quantum configurations described above by replacing the quantum state $|\psi\rangle$ with a suitable *pure quantum term* $\boldsymbol{t}$ and to replace the linking function ℓ with a suitable *unitary permutation* u_σ, where both $\boldsymbol{t}$ and u_σ are syntactic constructs from our pure quantum language that may be assigned types.

In order to accurately model the situation with the QLC, the term $\boldsymbol{t}$ has to correspond to a *normalised* vector and we replace the unitary constants from the QLC with programmable unitary constructs, so the pure subsystem also has to ensure their *unitarity*. The pure subsystem from Chapter 4, §4.0.4 is based on ideas described in [SVV18], with an additional introduction of an *equational theory* for the pure subsystem, which has unique normal forms and a *denotational semantics* for it that is *sound and complete* with respect to the equational theory.

Finally, we integrate the pure quantum subsystem into our main calculus, which allows us to describe both quantum and classical information and, therefore also classical control. This is a variant of the QLC with the addition of a type Nat for classical natural numbers and the addition of a modality $\mathcal{B}(\boldsymbol{Q})$ which allows us to view pure quantum types $\boldsymbol{Q}$ as types of mixed state quantum operations in the Heisenberg picture. The intuition behind this is the following: mixed states in the Schrödinger picture can be seen as CPTP maps[1] $(1 \mapsto \rho) \colon \mathbb{C} \to \mathcal{T}(H)$ whereas mixed states in the Heisenberg picture are given by functionals of the form $\mathrm{tr}(\rho -) \colon \mathcal{B}(H) \to \mathbb{C}$, both of which are determined by a choice of density operator $\rho \colon H \to H$ (see [C14, Section 7] for detail on this matter). We motivate the choice for working in the Heisenberg picture in our work and this is the justification for the symbol we use for the modality. This modality allows us to replace the *constants* for state preparation and unitaries acting on qubits in the QLC with *terms and expressions* from the pure subsystem acting on more general quantum types beyond qubits. The operational semantics are described via a suitable adaptation of the aforementioned quantum configurations. The denotational semantics follows previous work on quantum programming semantics based on von Neumann algebras [CW16, PPRZ20, JKLMZ22] and we show, in addition, that the assignment $\mathcal{B}(-)$ may be extended to a strict monoidal functor (between the relevant categories of Hilbert spaces and von Neumann algebras) that is crucial for

[1] CPTP stands for 'completely positive trace preserving' and $\mathcal{T}(H)$ and $\mathcal{B}(H)$ stand for the trace-class and bounded linear operators, respectively, on a Hilbert space H.

our denotational semantics.

In the next section we very briefly recall the quantum control fragment before going on to describe the main calculus which achieves this combination.

5.1 Quantum Control Fragment

The quantum control fragment consists of the language described in §4.0.4 of Chapter 4, based on a pattern-matching syntax. The syntax for this fragment is represented using bold small-case Latin letters ($\boldsymbol{t}$), to highlight the fact that such terms belong to a different fragment of the calculus. It is a first-order language, with its operational semantics defined using an equational theory. The denotational semantics is laid out using appropriate categories of Hilbert spaces and linear operators acting on them. The equational theory is proven to be sound and complete with respect to the denotational interpretation. This fragment models pure-state quantum computation.

5.2 The main calculus

This section lays down the syntax and semantics of the main calculus, which combines the quantum control fragment with a classically controlled layer, a variant of a call-by-value linear lambda-calculus. The subsystem which we presented in the previous section is designed to represent pure quantum information and computation, whereas in this section, we are concerned with mixed quantum information and computation. Our denotational approach towards this uses von Neumann algebras, which provide an appropriate mathematical setting to achieve this treatment. Operationally, this is achieved by adapting the configurations (also called closures) $(|\psi\rangle, \ell, M)$ from an effectful quantum lambda-calculus, e.g., [CW16, SV09], by replacing the quantum state $|\psi\rangle$ with a pure term $\boldsymbol{t}$, replacing the linking function ℓ with a unitary u that can be described via our syntax and which represents a suitable permutation, and finally by defining suitable formation conditions for the configuration. We refer to this system as the "main calculus", because it allows us to integrate the pure quantum subsystem within it, but it also allows us to represent classical information, quantum measurements and mixed-state quantum computation.

5.2.1 Syntax

The *types* of the main calculus, ranged over by capital Latin letters (e.g., A, B) are given in Figure 5.2 and we now explain how they should be understood. We write I for the unit type, $\mathrm{A} + \mathrm{B}$ for sum types (which allow us to handle classical control), $\mathrm{A} \otimes \mathrm{B}$ for pair types, $\mathrm{A} \multimap \mathrm{B}$ for higher-order linear function types, and $!\mathrm{A}$ for exponential types (in the sense of linear logic). Nat is a ground type representing classical natural numbers and $\mathcal{B}(\boldsymbol{Q})$ is the type that represents *mixed state* quantum computation on the Hilbert space determined by the type $\boldsymbol{Q}$. As we see later, this type plays an important role by introducing quantum and probabilistic effects into the system. The modality $\mathcal{B}$ is inspired from the denotational model, where $\mathcal{B}(H)$ represents the von Neumann algebra of bounded operators on the Hilbert space H.

Terms (and values) of the calculus are described in Figure 5.2, and are either standard linear lambda-calculus terms, terms representing natural numbers, or terms that we introduce to model the interaction of the classical control fragment with the quantum control fragment. *zero* and *succ* are constructors for natural numbers and $\mathit{match}\, L\, \mathit{with}\, \{\mathit{zero} \to M,\ \mathit{succ}(x) \to N\}$ is the corresponding pattern-match. The term $\mathit{pure}(\boldsymbol{t})$ models the preparation of a pure quantum state represented by the quantum term $\boldsymbol{t}$ (from the quantum control fragment). Measurement is

(Vars)	x, y, z		
(Types)	A, B	$::=$	$\mathrm{I} \mid \mathrm{A} + \mathrm{B} \mid \mathrm{A} \otimes \mathrm{B} \mid !\mathrm{A} \mid \mathrm{A} \multimap \mathrm{B} \mid \mathrm{Nat} \mid \mathcal{B}(\boldsymbol{Q})$
(Terms)	L, M, N	$::=$	$* \mid x \mid \mathit{inl}(M) \mid \mathit{inr}(N) \mid \mathit{case}\, L\, \mathit{of}\, \{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\}$
			$\mid M \otimes N \mid \mathit{let}\, x \otimes y = M\ \mathit{in}\ N \mid \mathit{lift}(M) \mid \mathit{force}(M)$
			$\mid \lambda x.M \mid M\, N$
			$\mid \mathit{zero} \mid \mathit{succ}(M) \mid \mathit{match}\, L\, \mathit{with}\, \{\mathit{zero} \to M,\ \mathit{succ}(x) \to N\}$
			$\mid \mathit{pure}(\boldsymbol{t}) \mid \mathit{meas}(M) \mid \mathcal{B}(u)(M)$
			$\mid \mathit{let}\, \mathcal{B}(z) = M\ \mathit{in}\ N \mid \mathit{let}\, \mathcal{B}(x \otimes y) = M\ \mathit{in}\ N$
(Values)	V, W	$::=$	$* \mid x \mid \mathit{inl}(V) \mid \mathit{inr}(W) \mid V \otimes W \mid \mathit{lift}(M) \mid \lambda x.M$
			$\mid \mathit{zero} \mid \mathit{succ}(V)$

Figure 5.2: Syntax of the classically controlled system.

modelled by the term $\mathit{meas}(M)$. The term $\mathcal{B}(u)(M)$ represents the application of the unitary u to a term M. Additionally, we have two syntactic constructs to model the isomorphism between the types $\mathcal{B}(\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2)$ and $\mathcal{B}(\boldsymbol{Q}_1) \otimes \mathcal{B}(\boldsymbol{Q}_2)$: the term $\mathit{let}\, \mathcal{B}(z) = M\ \mathit{in}\ N$ allows one to construct a term of type $\mathcal{B}(\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2)$ from a term of type $\mathcal{B}(\boldsymbol{Q}_1) \otimes \mathcal{B}(\boldsymbol{Q}_2)$ and the term $\mathit{let}\, \mathcal{B}(x \otimes y) = M\ \mathit{in}\ N$ deals with the opposite direction. This isomorphism arises because the functor $\mathcal{B}(\cdot)$ used in the denotational model is *strict monoidal*. Both constructs are necessary to allow the language to manipulate composed quantum systems and also terms possibly representing *entangled states*.

Typing contexts (ranged over by symbols $\Delta, \Sigma_1, \Sigma_2$) are finite sequences $\Delta := x_1 : \mathrm{A}_1, \ldots, x_n : \mathrm{A}_n$ mapping variables to types. We use the notation $!\Delta$ for contexts of the form $x_1 : !\mathrm{A}_1, \ldots, x_n : !\mathrm{A}_n$. *Typing judgements* have the form $\Delta \vdash M : \mathrm{A}$ and follow a linear typing discipline to deal with quantum data. The typing rules are given in Figure 5.3 where, as usual, Σ_1, Σ_2 denotes the disjoint union of Σ_1 and Σ_2. Classical bits correspond to the type $\mathit{bit} := I + I$ and mixed state qubits have type $\mathit{qbit} := \mathcal{B}(\mathbf{qbit})$. The rule for the term $\mathit{pure}(\boldsymbol{t})$ introduces a state $\boldsymbol{t}$ from the quantum control fragment as a term of type $\mathcal{B}(\boldsymbol{Q})$ into the main calculus. The typing rule for measurement maps a term M of quantum type $\mathcal{B}(\boldsymbol{Q})$ to a term $\mathit{meas}(M)$ of classical type $\overline{\boldsymbol{Q}}$, where the type $\overline{\boldsymbol{Q}}$ is defined inductively on the structure of the pure quantum type $\boldsymbol{Q}$:

$$\overline{\boldsymbol{I}} := I, \quad \overline{\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2} := \overline{\boldsymbol{Q}_1} \otimes \overline{\boldsymbol{Q}_2}, \quad \overline{\boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2} := \overline{\boldsymbol{Q}_1} + \overline{\boldsymbol{Q}_2}, \quad \overline{\mathbf{qnat}} := \mathrm{Nat}.$$

This gives the classical analogue of the type $\boldsymbol{Q}$, e.g., $\overline{\mathbf{qbit}} = \mathit{bit}$. For a quantum basis term $\boldsymbol{b}$ of type $\boldsymbol{Q}$, $\overline{\boldsymbol{b}}$ (defined below) denotes its translation to a classical value of type $\overline{\boldsymbol{Q}}$, which we use to represent measurement outcomes.

$$\overline{*} := *, \quad \overline{\boldsymbol{b}_1 \otimes \boldsymbol{b}_2} := \overline{\boldsymbol{b}_1} \otimes \overline{\boldsymbol{b}_2}, \quad \overline{\mathit{inl}\boldsymbol{b}} := \mathit{inl}(\overline{\boldsymbol{b}}), \quad \overline{\mathbf{0}} := \mathit{zero}, \quad \overline{\mathbf{S}\, \boldsymbol{b}} := \mathit{succ}(\overline{\boldsymbol{b}}).$$

$$\frac{!\Delta, x:\mathrm{A}, y:\mathrm{B}, \Sigma \vdash M:\mathrm{C}}{!\Delta, y:\mathrm{B}, x:\mathrm{A}, \Sigma \vdash M:\mathrm{C}}\, ex \qquad \frac{}{!\Delta \vdash *:\mathrm{I}}\,\mathrm{I}_i \qquad \frac{}{!\Delta, x:\mathrm{A} \vdash x:\mathrm{A}}\, ax$$

$$\frac{\Delta \vdash M:\mathrm{A}}{\Delta \vdash \mathit{inl}(M):\mathrm{A}+\mathrm{B}}\,\oplus_{i_1} \qquad \frac{\Delta \vdash N:\mathrm{B}}{\Delta \vdash \mathit{inr}(N):\mathrm{A}+\mathrm{B}}\,\oplus_{i_2}$$

$$\frac{!\Delta, \Sigma_1 \vdash L:\mathrm{A}+\mathrm{B} \qquad !\Delta, \Sigma_2, x:\mathrm{A} \vdash M:\mathrm{C} \qquad !\Delta, \Sigma_2, y:\mathrm{B} \vdash N:\mathrm{C}}{!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{case}\, L\, \mathit{of}\{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\}:\mathrm{C}}\,\oplus_e$$

$$\frac{!\Delta, \Sigma_1 \vdash M:\mathrm{A} \quad !\Delta, \Sigma_2 \vdash N:\mathrm{B}}{!\Delta, \Sigma_1, \Sigma_2 \vdash M \otimes N:\mathrm{A}\otimes\mathrm{B}}\,\otimes_i \qquad \frac{!\Delta, \Sigma_1 \vdash M:\mathrm{A}\otimes\mathrm{B} \quad !\Delta, \Sigma_2, x:\mathrm{A}, y:\mathrm{B} \vdash N:\mathrm{C}}{!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\, x\otimes y = M\ \mathit{in}\ N:\mathrm{C}}\,\otimes_e$$

$$\frac{!\Delta \vdash M:\mathrm{A}}{!\Delta \vdash \mathit{lift}(M):!\mathrm{A}}\,!_i \qquad \frac{\Delta \vdash M:!\mathrm{A}}{\Delta \vdash \mathit{force}(M):\mathrm{A}}\,!_e$$

$$\frac{\Delta, x:\mathrm{A} \vdash M:\mathrm{B}}{\Delta \vdash \lambda x.M:\mathrm{A}\multimap\mathrm{B}}\,\multimap_i \qquad \frac{!\Delta, \Sigma_1 \vdash M:\mathrm{A}\multimap\mathrm{B} \quad !\Delta, \Sigma_2 \vdash N:\mathrm{A}}{!\Delta, \Sigma_1, \Sigma_2 \vdash M\, N:\mathrm{B}}\,\multimap_e$$

$$\frac{}{!\Delta \vdash \mathit{zero}:\mathrm{Nat}}\,\mathrm{Nat}_0 \qquad \frac{\Delta \vdash M:\mathrm{Nat}}{\Delta \vdash \mathit{succ}(M):\mathrm{Nat}}\,\mathrm{Nat}_s$$

$$\frac{!\Delta, \Sigma_1 \vdash L:\mathrm{Nat} \quad !\Delta, \Sigma_2 \vdash M:\mathrm{A} \quad !\Delta, \Sigma_2, x:\mathrm{Nat} \vdash N:\mathrm{A}}{!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{match}\, L\, \mathit{with}\{\mathit{zero} \to M,\ \mathit{succ}(x) \to N\}:\mathrm{A}}\,\mathrm{Nat}_e$$

$$\frac{\cdot \vdash \boldsymbol{t}:\boldsymbol{Q}}{!\Delta \vdash \mathit{pure}(\boldsymbol{t}):\mathcal{B}(\boldsymbol{Q})}\,\mathcal{B}_i$$

$$\frac{\Delta \vdash M:\mathcal{B}(\boldsymbol{Q})}{\Delta \vdash \mathit{meas}(M):\overline{\boldsymbol{Q}}}\,\mathcal{B}_e \qquad \frac{\Vdash u:U(\boldsymbol{Q}_1,\boldsymbol{Q}_2) \qquad !\Delta, \Sigma_1 \vdash M:\mathcal{B}(\boldsymbol{Q}_1)}{!\Delta, \Sigma_1 \vdash \mathcal{B}(u)(M):\mathcal{B}(\boldsymbol{Q}_2)}\,u_e$$

$$\frac{!\Delta, \Sigma_1 \vdash M:\mathcal{B}(\boldsymbol{Q}_1)\otimes\mathcal{B}(\boldsymbol{Q}_2) \qquad !\Delta, \Sigma_2, z:\mathcal{B}(\boldsymbol{Q}_1\otimes\boldsymbol{Q}_2) \vdash N:\mathrm{A}}{!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\,\mathcal{B}(z) = M\ \mathit{in}\ N:\mathrm{A}}\,\mathcal{B}(\otimes)_i$$

$$\frac{!\Delta, \Sigma_1 \vdash M:\mathcal{B}(\boldsymbol{Q}_1\otimes\boldsymbol{Q}_2) \qquad !\Delta, \Sigma_2, x:\mathcal{B}(\boldsymbol{Q}_1), y:\mathcal{B}(\boldsymbol{Q}_2) \vdash N:\mathrm{A}}{!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\,\mathcal{B}(x\otimes y) = M\ \mathit{in}\ N:\mathrm{A}}\,\mathcal{B}(\otimes)_e$$

Figure 5.3: Typing rules for terms.

5.3 Operational Semantics

Quantum configurations. We lay down the operational semantics of the calculus using an adaptation of the notion of a *quantum configuration* from quantum lambda-calculi ([CW16, SV09]).

Definition 89 (Quantum Configuration). *Let* $\mathtt{Conf}$ *be the set of* quantum configurations $\mathcal{C} \triangleq (\boldsymbol{t}, u_\sigma, M)$, *where:*

- $\boldsymbol{t} \in \boldsymbol{Terms}$ *represents a* pure quantum state*;*
- $u_\sigma \in \boldsymbol{Unitaries}$ *represents a symmetric monoidal isomorphism;*
- $M \in$ Terms *is a term from the main calculus.*

In the above definition, we can think of the quantum term $\boldsymbol{t}$ as representing a quantum state, because the quantum control fragment ensures that $\boldsymbol{t}$ can be rewritten (in its equational theory)

to a unique quantum value (Proposition 6). Next, u_σ permutes the components of $\boldsymbol{t}$ so that they appear in the same order as variables that represent them in M. We slightly abuse terminology and refer to u_σ as a permutation, but it is meant to be understood as a canonical symmetric monoidal isomorphism (see Proposition 5). If we fix the domain and codomain of u_σ, then the essential data is given by a choice of permutation σ and this makes it easier to understand how each unitary acts on the coordinates, and the contextual rules formulation becomes precise. Note that, thanks to Proposition 5, the pure quantum syntax is expressive enough to represent all the monoidal permutations that we need in the sequel.

Example 75. Consider the quantum configuration $(\boldsymbol{t}, u_{id}, x \otimes y)$, where we have $\boldsymbol{t} := * \otimes (1/\sqrt{2} \cdot |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle + 1/\sqrt{2} \cdot |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle)$ and $u_{id} := \{| * \otimes \boldsymbol{w} \otimes \boldsymbol{z} \mapsto \boldsymbol{w} \otimes \boldsymbol{z}\}$ is a unitary with type $\Vdash u_{id} \colon U(\boldsymbol{I} \otimes \mathbf{qbit} \otimes \mathbf{qbit}, \mathbf{qbit} \otimes \mathbf{qbit})$. It conveys the information that the variable x in $x \otimes y$ keeps track of the first qubit in $\boldsymbol{t}$ and that the variable y keeps track of the second one. Whereas in the configuration $(\boldsymbol{t}, u_{swap}, x \otimes y)$ with $u_{swap} := \{| * \otimes \boldsymbol{w} \otimes \boldsymbol{z} \mapsto \boldsymbol{z} \otimes \boldsymbol{w}\}$, the variable x keeps track of the second qubit and y of the first one. Note that, due to quantum entanglement, the term $\boldsymbol{t}$ cannot be decomposed into a non-trivial tensor product, and the variables x and y should be thought of as identifying qubit components of $\boldsymbol{t}$, instead of storing quantum data themselves.

A quantum configuration $\mathcal{V} = (\boldsymbol{v}, u_\sigma, V)$, when both the quantum term $\boldsymbol{v}$ and term V are values, is called a *value configuration*. For example, $(|0\rangle, u_{id}, x)$ is a value configuration. A quantum configuration $\mathcal{SV} = (\boldsymbol{v}, u_\sigma, M)$, where $\boldsymbol{v}$ is a quantum value, is a *semi-value configuration*. We write `VConf` (resp. `SVConf`) for the set of (resp. semi-)value configurations. The reason we introduce semi-value configurations, is because they allow us to define our operational semantics for the main calculus modulo equality of the pure quantum terms $\boldsymbol{t}$ (analogous to the quantum lambda-calculus). It is already proven that values $\boldsymbol{v}$ are normal forms for the pure terms (Proposition 6), so this makes them a natural choice.

Definition 90. *A quantum configuration* $(\boldsymbol{t}, u_\sigma, M)$ *is* well-formed *with type* A*, written as* $(\boldsymbol{t}, u_\sigma, M) \colon \mathrm{A}$ *if the following judgments can be derived:*

- $\cdot \vdash \boldsymbol{t} \colon \boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_n$;
- $\cdot \Vdash u_\sigma \colon U(\boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_n, \boldsymbol{Q}'_1 \otimes \cdots \otimes \boldsymbol{Q}'_m)$;
- $x_1 : \mathcal{B}(\boldsymbol{Q}'_1), \ldots, x_m : \mathcal{B}(\boldsymbol{Q}'_m) \vdash M : \mathrm{A}$.

We write $\mathrm{WF}_X(A)$ for the set of well-formed configurations in X with type A, where $X \in \{\mathtt{Conf}, \mathtt{VConf}, \mathtt{SVConf}\}$. Intuitively, a well-formed total configuration is *total* when the entire pure state represented by $\boldsymbol{t}$ is "covered" by variables in M. Non-total configurations are only needed for some technical reasons in a few proofs.

Example 76. The configuration $((1/\sqrt{2} \cdot |\mathbf{00}\rangle + 1/\sqrt{2} \cdot |\mathbf{11}\rangle) \otimes |\mathbf{0}\rangle, u_{\mathrm{id}}, x \otimes y)$ with:

- $\cdot \vdash (1/\sqrt{2} \cdot |\mathbf{00}\rangle + 1/\sqrt{2} \cdot |11\rangle) \otimes |\mathbf{0}\rangle \ : \mathbf{qbit}^3$;
- $\cdot \Vdash u_{\mathrm{id}} \colon U(\mathbf{qbit}^3, \mathbf{qbit}^3)$;
- $x : \mathcal{B}(\mathbf{qbit}^2), y : \mathit{qbit} \vdash x \otimes y : \mathcal{B}(\mathbf{qbit}^2) \otimes \mathit{qbit}$.

In this configuration, u_{id} is just the identity, x points to the Bell state in $\boldsymbol{t}$ whereas y points to $|\mathbf{0}\rangle$. We have $n = 3, m = 2$, with $\boldsymbol{Q}_1 = \boldsymbol{Q}_2 = \boldsymbol{Q}_3 = \mathbf{qbit}$, $\boldsymbol{Q}'_1 = \mathbf{qbit}^2$ and $\boldsymbol{Q}'_2 = \mathbf{qbit}$. The configuration is well-formed.

The above example highlights the need for m and n (from Definition 90) to be different values. The intuition behind this difference is that the unitary u_σ partitions the n quantum types into m blocks, where each block is represented by a variable in M. Furthermore, in situations where we wish to combine two such blocks which are not next to each other in the tensor expression of the n types, we allow the possibility of permuting these blocks and merging them so that now two blocks that were initially represented by two variables are represented by one. This is formally expressed in the reduction rules we define next. The role of u_σ in a well-formed configuration is illustrated in Figure 5.4. The *small-step reduction* $\cdot \to_{\cdot} \cdot \subseteq \texttt{Conf} \times [0,1] \times \texttt{Conf}$ is the relation defined by the rules of Figure 5.6: the reduction $\mathcal{C} \to_p \mathcal{C}'$ holds when the quantum configuration $\mathcal{C}$ reduces to $\mathcal{C}'$ with probability $p \in [0,1]$.

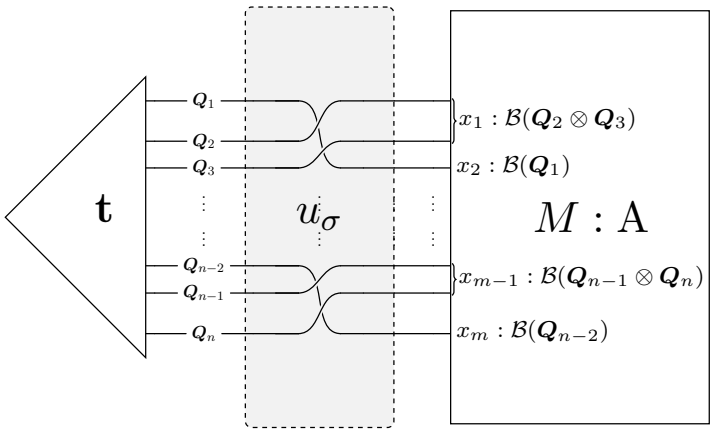


Figure 5.4: Graphical representation of a well-formed quantum configuration.

Additionally, we need to consider configurations where the entire state $\boldsymbol{t}$ is not covered by the variables appearing in M when proving subject reduction. Hence, we define a *well-formed* quantum configuration with *auxiliary types* to account for such configurations. A quantum configuration $(\boldsymbol{t}, u_\sigma, M)$ is said to be *well-formed with type* A *and auxiliary quantum data* of types $[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$, written $(\boldsymbol{t}, u_\sigma, M) \colon \mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$, if the following hold:

- $\cdot \vdash \boldsymbol{t} \colon \boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_n$ can be derived;
- $\cdot \Vdash u_\sigma \colon U(\boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_n, \boldsymbol{Q}''_1 \otimes \cdots \otimes \boldsymbol{Q}''_{m+k})$ can be derived;
- $x_1 : \mathcal{B}(\boldsymbol{Q}''_1), \ldots, x_m : \mathcal{B}(\boldsymbol{Q}''_m) \vdash M : \mathrm{A}$ can be derived;
- $[\boldsymbol{Q}''_1, \ldots, \boldsymbol{Q}''_{m+k}] = [\boldsymbol{Q}''_1, \ldots, \boldsymbol{Q}''_m, \boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ as lists of pure quantum types.

$\mathrm{WF}_{\texttt{Conf}}(\mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k])$ denotes the set of well-formed configurations wrt type A and auxiliary quantum types $[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$.

The *small-step reduction* $\cdot \to_{\cdot} \cdot \subseteq \texttt{Conf} \times [0,1] \times \texttt{Conf}$ is the relation defined by the rules of Figure 5.6: $\mathcal{C} \to_p \mathcal{C}'$ holds when the quantum configuration $\mathcal{C}$ reduces to $\mathcal{C}'$ with probability $p \in [0,1]$. Let $\mathrm{WF}_{\texttt{Conf}}(\mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k])$ denote the set of well-formed configurations wrt type A and auxiliary quantum types $[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$. The reduction rules along with the unitaries required to state them are given in Figure 5.6. $\{\boldsymbol{b}_{ij}\}_{i,j}$ stand for basis terms and s in the reduction rule for $\mathit{meas}(x)$ corresponds to the position of $\boldsymbol{b}_i$ in the tensor product appearing in the quantum term. All sums are finite. The unitary $u_{\sigma_{gather}}$ appearing in the reduction rule for the term *let* $\mathcal{B}(z) = x \otimes y$ *in* N merges two consecutive variables into one block, so that now a single

variable can represent it. Similarly, in the reduction rule for the term $\mathit{let}\ \mathcal{B}(x \otimes y) = z\ \mathit{in}\ N$, the unitary $u_{\sigma_{divide}}$ partitions a block into two, so that two variables can represent them. The two rules are symmetric as expected, since these terms only allow switching between types $\mathcal{B}(\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2)$ and $\mathcal{B}(\boldsymbol{Q}_1) \otimes \mathcal{B}(\boldsymbol{Q}_2)$. Unitaries applied to the reducts of the rules in Figure 5.6 are given in Figure 5.5. As is evident in Figure 5.5, we consider more than m variables, to account for auxiliary wires. Intuitively, the grouping and separation of variables in unitaries $u_{\sigma_{gather}}, u_{\sigma_{divide}}$ happens for the last few of the m variables which appear in the term M, and the wires after these m variables in the graphical representation (see Figure 5.5) are auxiliary wires. This is done so that the unitaries are also applicable for configurations which are not total. The last rule of Figure 5.6 is a *contextual rule* which holds for any *evaluation context* E. Evaluation contexts are defined by the following grammar:

$$\begin{aligned} E \quad := \quad & [.] \mid E \otimes N \mid V \otimes E \mid \mathit{let}\ x \otimes y = E\ \mathit{in}\ N \mid \mathit{inl}(E) \mid \mathit{inr}(E) \\ & \mid \mathit{case}\, E\, \mathit{of}\{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\} \mid \mathit{force}(E) \mid \mathit{succ}(E) \mid E\, N \\ & \mid V\, E \mid \mathit{meas}(E) \mid \mathcal{B}(u)(E) \mid \mathit{let}\ \mathcal{B}(z) = E\ \mathit{in}\ N \mid \mathit{let}\ \mathcal{B}(x \otimes y) = E\ \mathit{in}\ N \end{aligned}$$

One of the premises of the contextual rule states that if σ is a permutation which does not act on the same set of indices as σ_1, σ_2, then we can form a "union" of these permutations by composing an *extension* of the two. For a permutation $\sigma : S \to S$, we can define its extension $\sigma^{ext} : S \sqcup S' \to S \sqcup S'$ by $\sigma^{ext}(s) := \sigma(s)$, if $s \in S$, $\sigma^{ext}(s') := s'$, if $s' \in S'$. We now introduce the reduction relation we use to define the operational semantics. This relation can be intuitively understood as a wrapper over the small-step reduction defined above. It facilitates rewriting of the quantum state to a desirable form when needed to perform a small-step reduction.

Definition 91. *The* reduction relation $\cdot \rightsquigarrow_{\cdot} \cdot \subseteq \mathtt{SVConf} \times [0,1] \times \mathtt{SVConf}$ *is defined in the following way: we write* $(\boldsymbol{v}, u_\sigma, M) \rightsquigarrow_p (\boldsymbol{v}', u_{\sigma'}, M')$ *whenever*

$$\frac{\cdot \vdash \boldsymbol{v} = \boldsymbol{t} : \boldsymbol{Q} \qquad (\boldsymbol{t}, u_\sigma, M) \to_p (\boldsymbol{t}', u_{\sigma'}, M') \qquad \cdot \vdash \boldsymbol{t}' = \boldsymbol{v}' : \boldsymbol{Q}'}{(\boldsymbol{v}, u_\sigma, M) \rightsquigarrow_p (\boldsymbol{v}', u_{\sigma'}, M')}$$

The intuition behind the above definition is that it allows us to reason modulo equality of the pure terms $\boldsymbol{t}$, as a consequence of Proposition 6. Next we state the standard results that a coherent operational semantics for a programming language is expected to satisfy. We show that well-formedness is preserved by the reduction relation $\cdot \rightsquigarrow_{\cdot} \cdot$. In order to do so, we first state and prove some preliminary lemmas.

Lemma 6 (Substitution). *For a value* V*, if* $!\Delta, \Sigma_1 \vdash V : \mathrm{A}$ *and for a term* M*,* $!\Delta, \Sigma_2, x : \mathrm{A} \vdash M : \mathrm{B}$ *then* $!\Delta, \Sigma_1, \Sigma_2 \vdash M[V/x] : \mathrm{B}$.

Proof. Proof is by induction on the typing judgement for M.

- $!\Delta, x : \mathrm{A}, y : \mathrm{B}, \Sigma_1 \vdash M : \mathrm{C}$: Since this judgement is derived by applying the rule *ex* from Figure 5.3, the claim follows directly using the induction hypothesis.
- $!\Delta \vdash * : I$: The claim for this case follows vacuously.
- $!\Delta, x : \mathrm{A} \vdash x : \mathrm{A}$: In this case $M[V/x] = V$ and $\Sigma_1 = \emptyset$. Hence, the claim follows from the typing derivation for V.
- $\Delta \vdash \mathit{inl}(()M) : A + B$ or $\Delta \vdash \mathit{inr}(()M) : A + B$: These cases follows from the induction hypothesis and the observation that $\mathit{inl}(()M)[V/x] = \mathit{inl}(()M[V/x])$ and $\mathit{inr}(()M)[V/x] = \mathit{inr}(()M[V/x])$.

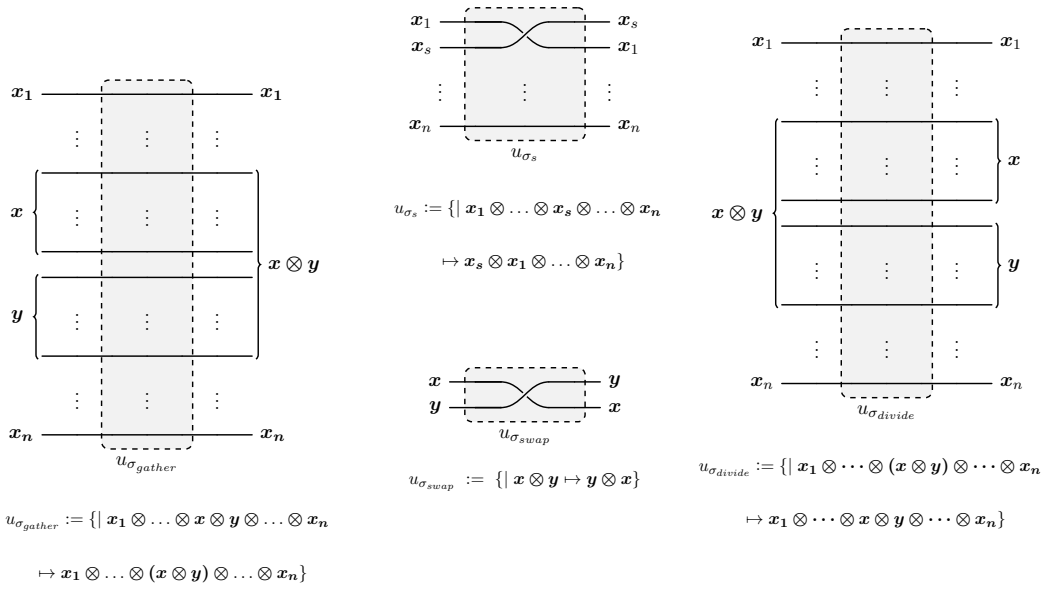


Figure 5.5: Unitary permutations for reduction rules.

$$
\begin{aligned}
(\boldsymbol{t}, u_\sigma, \mathit{case}\, \mathit{inl}(V)\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \ \mathit{inr}(y){\to}N\}) &\to_1 (\boldsymbol{t}, u_\sigma, M[V/x]) \\
(\boldsymbol{t}, u_\sigma, \mathit{case}\, \mathit{inr}(V)\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \ \mathit{inr}(y){\to}N\}) &\to_1 (\boldsymbol{t}, u_\sigma, N[V/y]) \\
(\boldsymbol{t}, u_\sigma, \mathit{let}\ x \otimes y = V \otimes W\ \ \mathit{in}\ M) &\to_1 (\boldsymbol{t}, u_\sigma, M[V/x, W/y]) \\
(\boldsymbol{t}, u_\sigma, \mathit{force}(\mathit{lift}(M))) &\to_1 (\boldsymbol{t}, u_\sigma, M) \\
(\boldsymbol{t}, u_\sigma, (\lambda x.M)\, V) &\to_1 (\boldsymbol{t}, u_\sigma, M[V/x]) \\
(\boldsymbol{t}, u_\sigma, \mathit{match}\, \mathit{zero}\, \mathit{with}\, \{\mathit{zero}{\to}M,\ \ \mathit{succ}(x){\to}N\}) &\to_1 (\boldsymbol{t}, u_\sigma, M) \\
(\boldsymbol{t}, u_\sigma, \mathit{match}\, \mathit{succ}(V)\ \mathit{with}\, \{\mathit{zero}{\to}M,\ \ \mathit{succ}(x){\to}N\}) &\to_1 (\boldsymbol{t}, u_\sigma, N[V/x]) \\
(\boldsymbol{t}, u_\sigma, \mathit{pure}(\boldsymbol{t}')) &\to_1 (\boldsymbol{t} \otimes \boldsymbol{t}', u_{\sigma_{swap}} \circ (u_\sigma \otimes u_{id}), x) \\
(\Sigma_i p_i{\cdot}\Sigma_j \boldsymbol{\alpha}_{ij}{\cdot}\boldsymbol{b}'_{ij} \otimes \cdots \otimes \boldsymbol{b}_i \otimes \cdots, u_{\sigma_s}, \mathit{meas}(x)) &\to_{|p_k|^2} (\Sigma_j \boldsymbol{\alpha}_{kj}{\cdot}\boldsymbol{b}'_{kj} \otimes \cdots, u_{id}, \overline{\boldsymbol{b}_k}) \\
(\boldsymbol{t}, u_\sigma, \mathcal{B}(u)(z)) &\to_1 ((u^*_\sigma \circ (u \otimes id) \circ u_\sigma)\, \boldsymbol{t}, u_\sigma, z) \\
(\boldsymbol{t}, u_\sigma, \mathit{let}\ \mathcal{B}(z) = x \otimes y\ \ \mathit{in}\ N) &\to_1 (\boldsymbol{t}, u_{\sigma_{gather}} \circ u_\sigma, N) \\
(\boldsymbol{t}, u_\sigma, \mathit{let}\ \mathcal{B}(x \otimes y) = z\ \ \mathit{in}\ N) &\to_1 (\boldsymbol{t}, u_{\sigma_{divide}} \circ u_\sigma, N)
\end{aligned}
$$

$$
\frac{(\boldsymbol{t}_1, u_{\sigma_1^{ext}}, M_1) \to_p (\boldsymbol{t}_2, u_{\sigma_2^{ext}}, M_2) \qquad dom(\sigma) \cap (dom(\sigma_1) \cup dom(\sigma_2)) = \emptyset}{(\boldsymbol{t}_1, u_{\sigma^{ext} \circ \sigma_1^{ext}}, E[M_1]) \to_p (\boldsymbol{t}_2, u_{\sigma^{ext} \circ \sigma_2^{ext}}, E[M_2])}
$$

Figure 5.6: Reduction rules.

- $!\Delta, \Sigma_1, \Sigma'_1 \vdash \mathit{case}\, L\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \ \mathit{inr}(y){\to}N\}$: There are two cases to consider. The first case is when z is not a linear variable, in which case the claim follows from the induction hypothesis and the observation that $\mathit{case}\, L\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \ \mathit{inr}(y){\to}N\}[V/z] = \mathit{case}\, L[V/z]\ \mathit{of}\{\mathit{inl}(x){\to}M[V/z],\ \ \mathit{inr}(y){\to}N[V/z]\}$. In the case that z is a linear variable, we again consider the separate cases when z appears in L or M, N. Both these cases follow from the induction hypothesis and the observations that $\mathit{case}\, L\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \mathit{inr}(y){\to}N\}[V/z] = \mathit{case}\, L[V/z]\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \ \mathit{inr}(y){\to}N\}$ and $\mathit{case}\, L\ \mathit{of}\{\mathit{inl}(x){\to}M,\ \ \mathit{inr}(y){\to}N\}[V/z] =$

case L *of*$\{inl(x)\rightarrow M[V/z],\ inr(y)\rightarrow N[V/z]\}$ respectively.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash M \otimes N$: Again there are two cases here. If x is not a linear variable, the claim follows from the induction hypothesis and the observation that $(M \otimes N)[V/x] = M[V/x] \otimes N[V/x]$. In the case when x is a linear variable, we have further two cases, depending on whether x appears in M or N. Both the cases follow from the induction hypothesis and the observation that $(M \otimes N)[V/x]$ is equal to $M[V/x] \otimes N$ or $M \otimes N[V/x]$ respectively.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash$ *let* $y \otimes z = M$ *in* N: Here two there are cases depending on the linearity of the variable x. If x is not linear, then the claim follows from the observation *let* $y \otimes z = M$ *in* $N[V/x] =$ *let* $y \otimes z = M[V/x]$ *in* $N[V/x]$ and the induction hypothesis. When x is linear, again there are two cases depending on whether x appears in M or N. Both these cases follow from the observation that *let* $y \otimes z = M$ *in* $N[V/x]$ is equal to *let* $y \otimes z = M[V/x]$ *in* N or *let* $y \otimes z = M$ *in* $N[V/x]$ respectively and the induction hypothesis.

- $!\Delta \vdash$ *lift*(M) :!A: This case follows from the induction hypothesis and the observation that *lift*$(M)[V/x] =$ *lift*$(M[V/x])$.

- $\Delta \vdash$ *force*(M) : A: This case follows from the induction hypothesis and the observation that *force*$(M)[V/x] =$ *force*$(M[V/x])$.

- $\Delta \vdash \lambda y.M$: A $\multimap$ B: This case follows from the induction hypothesis and the observation that $(\lambda y.M)[V/x] = \lambda y.M[V/x]$.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash M\ N : B$: There are two cases here. In the case when x is not linear, the claim follows from the induction hypothesis and the observation that $M\ N[V/x] = M[V/x]\ N[V/x]$. If x is linear, there are two cases depending on whether x appears in M or N. Both the cases, follow from the observation that $(M\ N)[V/x]$ is equal to $M[V/x]\ N$ or $M\ N[V/x]$ respectively and induction hypothesis.

- $!\Delta \vdash$ *zero* : Nat: This case follows vacuously.

- $\Delta \vdash$ *succ*$(()M)$: Nat: This case follows from the induction hypothesis and the observation that *succ*$(()M)[V/x] =$ *succ*$(()M[V/x])$.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash$ *match* L *with*$\{zero\rightarrow M,\ succ(x)\rightarrow N\}$: A: This case follows using a similar argument as that for the case when the last rule used to derive the typing judgement for M is $\oplus_e$ from Figure 5.3.

- $!\Delta \vdash$ *pure*$(\boldsymbol{t}) : \mathcal{B}(\boldsymbol{Q})$: This case follows vacuously.

- $\Delta \vdash$ *meas*$(M) : \overline{\boldsymbol{Q}}$: This case follows from the induction hypothesis and the observation that *meas*$(M)[V/x] =$ *meas*$(M[V/x])$.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash \mathcal{B}(u)(M) : \mathcal{B}(\boldsymbol{Q}_2)$: This case follows from the induction hypothesis and the observation that $\mathcal{B}(u)(M)[V/x] = \mathcal{B}(u)(M[V/x])$.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash$ *let* $\mathcal{B}(z) = M$ *in* N: There are two cases depending on the linearity of x. If x is not linear then the claim follows from the induction hypothesis and the observation that *let* $\mathcal{B}(z) = M$ *in* $N[V/x] =$ *let* $\mathcal{B}(z) = M[V/x]$ *in* $N[V/x]$. If x is linear there are two cases depending on whether x appears in M or N. Both follow using arguments similar to those presented previously.

- $!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\ \mathcal{B}(y \otimes z) = M\ \mathit{in}\ N$: This case follows using a similar argument to the one presented above.

□

Lemma 7 (Substitution for Contexts). *If* $!\Delta, \Sigma_1 \vdash M : A$ *and* $!\Delta, \Sigma_1, \Sigma_2 \vdash E[M] : B$*, then for* $!\Delta, \Sigma_1' \vdash M' : A$*, we have* $!\Delta, \Sigma_1', \Sigma_2 \vdash E[M'] : B$*.*

Proof. Follows from Lemma 6. □

Lemma 8 (Preservation for translation). *If* $\cdot \vdash \boldsymbol{v} : \boldsymbol{Q}$ *then* $\cdot \vdash \overline{\boldsymbol{v}} : \overline{\boldsymbol{Q}}$*.*

Proof. Proof is by induction on the rules for the translation. □

Theorem 9 (Subject Reduction). *For any* $\mathcal{C} \in \mathrm{WF}_{\mathcal{C}onf}(\mathrm{A}[\boldsymbol{Q}_1', \ldots, \boldsymbol{Q}_k'])$*, if* $\mathcal{C} \rightsquigarrow_p \mathcal{C}'$*, for some probability* $p \in [0, 1]$*, then* $\mathcal{C}' \in \mathrm{WF}_{\mathcal{C}onf}(\mathrm{A}[\boldsymbol{Q}_1', \ldots, \boldsymbol{Q}_k'])$*.*

Proof. Note that it suffices to prove that the result holds for the relation $\cdot \to_. \cdot$. Proof is by induction on $\cdot \to_. \cdot$:

- $(\boldsymbol{t}, u_\sigma, \mathit{case}\ \mathit{inl}(V)\ \mathit{of}\{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\}) \to_1 (\boldsymbol{t}, u_\sigma, M[V/x])$ or
  $(\boldsymbol{t}, u_\sigma, \mathit{case}\ \mathit{inr}(V)\ \mathit{of}\{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\}) \to_1 (\boldsymbol{t}, u_\sigma, N[V/y])$: We prove the first case, the other case follows analogously. Note that the reduction rule does not affect any auxiliary types, since the pure state $\boldsymbol{t}$ and the permutation unitary u_σ remain unaffected. Now, the claim simply follows from Lemma 6.

- $(\boldsymbol{t}, u_\sigma, \mathit{let}\ x \otimes y = V \otimes W\ \mathit{in}\ M) \to_1 (\boldsymbol{t}, u_\sigma, M[V/x, W/y])$: Again this rule does not affect any auxiliary types, since the pure state $\boldsymbol{t}$ and the permutation unitary u_σ remain unaffected. Again, the claim follows from Lemma 6.

- $(\boldsymbol{t}, u_\sigma, \mathit{force}(\mathit{lift}(M))) \to_1 (\boldsymbol{t}, u_\sigma, M)$: Again this rule does not affect any auxiliary types, thus it is enough to show that for the valid judgement $\Delta \vdash \mathit{force}(\mathit{lift}(M)) : A$, $\Delta \vdash M : A$ is valid. However, this follows from the rules $!_i, !_e$ in Figure 5.3.

- $(\boldsymbol{t}, u_\sigma, (\lambda x.M)\ V) \to_1 (\boldsymbol{t}, u_\sigma, M[V/x])$: Since here too the auxiliary types are unchanged, the claim follows from the rules $\multimap_i, \multimap_e$ in Figure 5.3.

- $(\boldsymbol{t}, u_\sigma, \mathit{match}\ \mathit{zero}\ \mathit{with}\ \{\mathit{zero} \to M,\ \mathit{succ}(x) \to N\}) \to_1 (\boldsymbol{t}, u_\sigma, M)$ or
  $(\boldsymbol{t}, u_\sigma, \mathit{match}\ \mathit{succ}(V)\ \mathit{with}\ \{\mathit{zero} \to M,\ \mathit{succ}(x) \to N\}) \to_1 (\boldsymbol{t}, u_\sigma, N[V/x])$: We show the first case, the second case follows analogously. Since, the auxiliary types remain unchanged, the claim again follows from Lemma 6.

- $(\boldsymbol{t}, u_\sigma, \mathit{pure}(\boldsymbol{t'})) \to_1 (\boldsymbol{t} \otimes \boldsymbol{t'}, u_{\sigma_{swap}} \circ (u_\sigma \otimes id), x)$: Since $(\boldsymbol{t}, u_\sigma, \mathit{pure}(\boldsymbol{t'}))$ is a well-formed configuration with auxillary types $\boldsymbol{Q}_1', \ldots, \boldsymbol{Q}_k'$, we have that $\cdot \vdash \boldsymbol{t} : \boldsymbol{Q}$, $\cdot \Vdash u_\sigma : U(\boldsymbol{Q}, \boldsymbol{Q}_1' \otimes \ldots \otimes \boldsymbol{Q}_k')$ and $\cdot \vdash \mathit{pure}(\boldsymbol{t'}) : \mathcal{B}(\boldsymbol{Q'})$ are valid. To see that the resulting configuration is well-formed with auxiliary types $\boldsymbol{Q}_1', \ldots, \boldsymbol{Q}_k'$, note that the judgements $\cdot \boldsymbol{t} \otimes \boldsymbol{t'} : \boldsymbol{Q} \otimes \boldsymbol{Q'}$, $\cdot \Vdash u_{\sigma_{swap}} \circ (u_\sigma \otimes id) : U(\boldsymbol{Q} \otimes \boldsymbol{Q'}, \boldsymbol{Q}_1' \otimes, \ldots, \boldsymbol{Q}_k')$ and $x : \mathcal{B}(\boldsymbol{Q'}) \vdash x : \mathcal{B}(\boldsymbol{Q'})$ are valid.

- $(\Sigma_i p_i \cdot \Sigma_j \boldsymbol{\alpha}_{ij} \cdot \boldsymbol{b}_{ij}' \otimes \cdots \otimes \boldsymbol{b}_i \otimes \cdots, u_{\sigma_s}, \mathit{meas}(x)) \to_{|p_k|^2} (\Sigma_j \boldsymbol{\alpha}_{kj} \cdot \boldsymbol{b}_{kj}' \otimes \cdots, u_{id}, \overline{\boldsymbol{b}_k})$: This case follows easily from Lemma 8 and the definition of a well-formed configuration.

- $(\boldsymbol{t}, u_\sigma, \mathcal{B}(u)(z)) \to_1 ((u_\sigma^* \circ (u \otimes id) \circ u_\sigma)\, \boldsymbol{t}, u_\sigma, z)$: Again this case, follows using a similar argument to the one for configurations with the term $\mathit{pure}(\boldsymbol{t'})$.

- $(\boldsymbol{t}, u_\sigma, \mathit{let}\ \mathcal{B}(z) = x \otimes y\ \mathit{in}\ N) \to_1 (\boldsymbol{t}, u_{\sigma_{gather}} \circ u_\sigma, N)$ or
  $(\boldsymbol{t}, u_\sigma, \mathit{let}\ \mathcal{B}(x \otimes y) = z\ \mathit{in}\ N) \to_1 (\boldsymbol{t}, u_{\sigma_{divide}} \circ u_\sigma, N)$: Both the cases follow directly from the definition of $u_{\sigma_{gather}}$ and $u_{\sigma_{divide}}$.

□

Next, we show that progress and strong normalisation hold. More specifically, a well-formed semi-value configuration $\mathcal{C}$ is either a value configuration or it reduces to a finite number of semi-value configurations with total probability 1, *i.e.* if we have not reached a normal form then the probability to be stuck is 0.

Theorem 10. *For a quantum configuration $\mathcal{C} \in \mathrm{WF}_{\mathtt{SVConf}}(\mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k])$, there are two possibilities:*

- *either $\mathcal{C} \in \mathtt{VConf}$,*
- *or there exists $\mathcal{C}_1 \in \mathrm{WF}_{\mathtt{SVConf}}(\mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k])$ such that $\mathcal{C} \rightsquigarrow_{p_1} \mathcal{C}_1$.*

Let $\{\mathcal{C}_i\}_{i \in I}$ be the set of all such distinct (i.e., not α-equivalent) configurations. Then, the set I is finite, and $\Sigma_{i \in I} p_i = 1$.

Proof. Let $\mathcal{C} = (\boldsymbol{v}, u_\sigma, M)$ be an arbitrary well-formed semi-value quantum configuration. Proof is by induction on the form of M:

- M is $*$: In this case $\mathcal{C}$ is a well-formed value configuration.
- M is x: In this case, since $\mathcal{C}$ is well-formed, x is of type $\mathcal{B}(\boldsymbol{Q})$ for some pure type $\boldsymbol{Q}$. Hence, it is a value configuration.
- M is $\mathit{inl}(M)$ or $\mathit{inr}(M)$: We prove the claim for the first case, the second case follows analogously. The claim follows from the induction hypothesis applied to the configuration $(\boldsymbol{v}, u_\sigma, M)$ and an application of the ocntextual reduction rule.
- M is $\mathit{case}\, L\, \mathit{of}\, \{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\}$: Without loss of generality we assume that L is a value. Since $\mathcal{C}$ is a well-formed semi-value configuration, L is either os the form $\mathit{inl}(V)$ or $\mathit{inr}(V)$, for some value V. In this case $\mathcal{C}$ reduces further under $\rightsquigarrow$.
- M is $M \otimes N$: Without loss of generality we assume that M is a value. If N is also a value then $\mathcal{C}$ is a well-formed value configuration and we are done. Otherwise, the claim follows from the induction hypothesis for M and the contextual reduction rule.
- M is $\mathit{let}\ x \otimes y = M\ \mathit{in}\ N$: Again without loss of generality, we assume that M is a value. Since $\mathcal{C}$ is well-formed, it follows that M is of the form $V \otimes W$ for values V, W. In this case $\mathcal{C}$ reduces further under $\rightsquigarrow$.
- M is $\mathit{lift}(M)$ or $\mathit{force}(M)$: Both the cases follow from an application of the induction hypothesis and the contextual reduction rule.
- M is $\lambda x.M$: In this case $\mathcal{C}$ is a well-formed value configuration.
- M is $M\ N$: Without loss of generality we assume that both M, N are values. In this case the reduction rule for application of lambda terms apllies, and hence $\mathcal{C}$ reduces further.
- M is *zero*: In this case $\mathcal{C}$ is a well-formed value configuration.

- M is $succ(M)$: This case follows by an application of the induction hypothesis to M and the contextual reduction rule.
- M is $match\, L\, with\, \{zero \to M,\ succ(x) \to N\}$: Without loss of generality we assume that L isi a value. Since $\mathcal{C}$ is well-formed it follows that L is of the form $zero$ or $succ(V)$, for some value V. In both these cases, $\mathcal{C}$ reduces further.
- M is $pure(\boldsymbol{t})$: In this case $\mathcal{C}$ reduces further to a value configuration following Proposition 6.
- M is $meas(M)$: Without loss of generality we assume M is a value. Since $\mathcal{C}$ is well-formed, it follows that M is of the form x with type $\mathcal{B}(\boldsymbol{Q})$ for some pure quantum type $\boldsymbol{Q}$. Now the result follows from an application of the reduction rule for $meas(x)$ and the fact that all terms from the pure fragment are normalized,i.e. they have norm 1 in the euclidean norm.
- M is $\mathcal{B}(u)(M)$: Without loss of generality, we assume thet M is a value. Again since $\mathcal{C}$ is well-formed, it follows that M is of the form x of type $\mathcal{B}(\boldsymbol{Q})$ for some pure quantum type $\boldsymbol{Q}$. Now the claim follows by an application of a reduction rule for $\rightsquigarrow$.
- M is $let\ \mathcal{B}(z) = M\ in\ N$ or $let\ \mathcal{B}(x \otimes y) = M\ in\ N$: Both the cases follow from arguments similar to the ones presented above.

□

Strong Normalisation In order to prove strong normalisation for the relation $\rightsquigarrow$, we define another relation $\longrightarrow$ on Terms of the main calculus, and prove that it strongly normalises. We also show that strong normalisation of $\longrightarrow$ implies strong normalisation of $\rightsquigarrow$.

Definition 92 (Term reduction). $\longrightarrow \in \text{Terms} \times [0,1] \times \text{Terms}$ *is defined as follows:*

$$\frac{(\boldsymbol{v}_1, u_{\sigma_1}, M_1) \rightsquigarrow_p (\boldsymbol{v}_2, u_{\sigma_2}, M_2)}{M_1 \longrightarrow_p M_2}$$

We additionally have a contextual reduction rule for the relation $\longrightarrow$ defined as:

$$\frac{M \longrightarrow_p M'}{E[M] \longrightarrow_p E[M']}$$

Definition 93 (Well-typed term). *We say a term $M \in \text{Terms}$ is well-typed of type $A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ if there exist $\boldsymbol{t}, u_\sigma$ such that $(\boldsymbol{t}, u_\sigma, M)$ is a well-formed configuration of type $A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$. We denote such a term by $M : A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$.*

Let $\mathcal{SN}_C$ denote the set of well-formed strongly normalizing quantum configurations under the relation $\rightsquigarrow$. Let $\mathcal{SN}_T$ denote the set of strongly normalizing terms from the main calculus under the relation $\longrightarrow$. We denote a reduction path $M \longrightarrow_{p_0} M_1 \longrightarrow_{p_1} \ldots \longrightarrow_{p_n} V$ by $M \longrightarrow_* V$.

Lemma 9. *For $M \in \mathcal{SN}_T$, if $\boldsymbol{v}, u_\sigma$ are such that $\mathcal{C} = (\boldsymbol{v}, u_\sigma, M)$ is well-formed then $\mathcal{C} \in \mathcal{SN}_C$.*

Proof. Follows directly from the definition of the sets $\mathcal{SN}_T, \mathcal{SN}_C$ and the relation $\longrightarrow_{\text{cot}}$. □

To prove strong normalisation of $\longrightarrow$, corresponding to each type we define the following sets:

- $\mathcal{S}_T[I] := \{M | M \in \mathcal{SN}_T \text{ and } M \longrightarrow_* V \implies V = * : I[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ for some } k\}$
- $\mathcal{S}_T[A + B] := \{M | M \in \mathcal{SN}_T \text{ and } M \longrightarrow_* V \implies V = \mathit{inl}(V) : A + B[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ or } V = \mathit{inr}(W) : A + B[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ for some } k\}$
- $\mathcal{S}_T[A \otimes B] := \{M | M \in \mathcal{SN}_T \text{ and } M \longrightarrow_* V \implies V = V \otimes W : A \otimes B[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ for some } k\}$
- $\mathcal{S}_T[!A] := \{M | M \in \mathcal{SN}_T \text{ and } M \longrightarrow_* V \implies V = \mathit{lift}(M') :!A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ for some } k \text{ and } \mathit{force}(V) \in \mathcal{S}_T[A]\}$
- $\mathcal{S}_T[\text{Nat}] := \{M | M \in \mathcal{SN}_T \text{ and } M \longrightarrow_* V \implies V = \mathit{zero} : \text{Nat}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ or } V = \mathit{succ}(V) : \text{Nat}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ for some } k\}$
- $\mathcal{S}_T[A \multimap B] := \{M | M \in \mathcal{SN}_T \text{ and } \forall N \in \mathcal{S}_T[A] \ , \ MN \in \mathcal{S}_T[B]\}$
- $\mathcal{S}_T[\mathcal{B}(\boldsymbol{Q})] := \{M | M \in \mathcal{SN}_T \text{ and } M \longrightarrow_* V \implies V = x : \mathcal{B}(\boldsymbol{Q})[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \text{ for some } k\}$

Definition 94 (Coherent configurations). *A quantum configuration $(\boldsymbol{t}, u_\sigma, M)$ is said to be coherent with respect to a typing context Γ with type $A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ if the following hold:*

- $\cdot \vdash \boldsymbol{t} : \boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_n$ *can be derived;*
- $\cdot \Vdash u_\sigma : U(\boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_n, \boldsymbol{Q}''_1 \otimes \cdots \otimes \boldsymbol{Q}''_{m+k})$ *can be derived;*
- $\Gamma, x_1 : \mathcal{B}(\boldsymbol{Q}''_1), \ldots, x_m : \mathcal{B}(\boldsymbol{Q}''_m) \vdash M : \text{A}$ *can be derived;*
- $[\boldsymbol{Q}''_1, \ldots, \boldsymbol{Q}''_{m+k}] = [\boldsymbol{Q}''_1, \ldots, \boldsymbol{Q}''_m, \boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ *as lists of pure quantum types.*

Definition 95 (Coherent term). *A term M is said to be coherent with respect to a typing context Γ with type $A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ if there exist $\boldsymbol{t}, u_\sigma$ such that $(\boldsymbol{t}, u_\sigma, M)$ is a coherent configuration with respect to Γ.*

Remark 31. Note that when $\Gamma = \emptyset$ a coherent configuration is just a well-typed configuration. Similarly, in this case a coherent term is just a well-typed term.

Definition 96 (Substitution function). *A substitution function is a map $l : \textsc{Vars} \to \textsc{Values}$. For a term M, Ml denotes the term obtained after substituting the free variables in M with their image under l.*

For a typing context Γ, we define a set $\mathcal{S}_C[\Gamma] := \{l | \mathit{dom}(l) = \mathit{dom}(\Gamma) \text{ and } x : A \in \Gamma \implies lx \in S_T[A]\}$.

Now we prove a series of results, which lead to Theorem 15.

Theorem 11 (Progress). *For $M : A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$, there are two possibilities:*

- *either $M \in \textsc{Values}$,*
- *or there exists $M_1 \in A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ such that $M \longrightarrow_{p_1} M_1$.*

Let $\{M_i\}_{i \in I}$ be the set of all such distinct (i.e., not α-equivalent) terms. Then, the set I is finite, and $\Sigma_{i \in I} p_i = 1$.

Proof. Follows from Theorem 10. □

Theorem 12 (Subject Reduction). *For any $M : A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$, if $M \longrightarrow_p M'$, for some probability $p \in [0,1]$, then $M' : A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$.*

Proof. Follows from Theorem 9. □

Lemma 10. *For a type A, if $M \in \mathcal{S}_T[A]$ then $M \in \mathcal{SN}_T$.*

Proof. Follows from the definition of the sets $\mathcal{S}_T[A]$ for any type A. □

Lemma 11. *For terms M, M', if $M \in \mathcal{S}_T[A]$ for some A, and $M \longrightarrow_p M'$ then $M' \in S_T[A]$.*

Lemma 12. *If $M[V/x] \in \mathcal{S}_T[A]$, then $(\lambda x.M)\, V \in \mathcal{S}_T[A]$.*

Proof. Note that there is only one possible reduction path that can be issued from $(\lambda x.M)\, V$. This path begins at $M[V/x]$. Since $M[V/x] \in \mathcal{S}_T[A]$, this reduction path is finite. Hence, $(\lambda x.M)\ V \in \mathcal{S}_T[A]$. □

Theorem 13. *If M is a coherent term with respect to Γ of type $A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]$ then $\forall l \in \mathcal{S}_C[\Gamma], Ml \in \mathcal{S}_T[A]$.*

Proof. Proof is by induction on the formation rule for the term M. We present some important cases here:

- $\lambda x.M$:
  We need to show that $\forall l \in \mathcal{S}_C[\Gamma],\ Ml \in \mathcal{S}_T[A \multimap B]$.
  Note that $(\lambda x.M)l = \lambda x.Ml$ up to $\alpha-$equivalence. Suffices to check that for $N \in \mathcal{S}_T[A]$ $(\lambda x.Ml)\, N \in \mathcal{S}_T[B]$. Since $N \in \mathcal{S}_T[A]$, all reductions from this term terminate in finitely many steps to a value of the form V. For each such reduction, we can apply the contextual rule for the relation $\longrightarrow$ to obtain that $(\lambda x.Ml)\, N$ reduces in each case to a term of the form $(\lambda x.Ml)V$ in finitely many steps. Note that from Lemma 12, it suffices to show that $(Ml)[V/x] \in \mathcal{S}_T[B]$. We know by subject reduction that $(Ml)[V/x]$ is a well-typed term. It follows that $l' = l \cup \{x \mapsto V\} \in \mathcal{S}_C[\Gamma, x : A]$. Hence from IH, it follows that $Ml' \in \mathcal{S}_T[B]$. But $Ml' = M(l \cup \{x \mapsto V\} = Ml[V/x]$. Hence, $Ml[V/x] \in \mathcal{S}_T[B]$.

- MN:
  We need to show that for any $l \in \mathcal{S}_C[\Gamma], (MN)l \in \mathcal{S}_T[A]$.
  By the induction hypothesis, we know that $Ml \in \mathcal{S}_T[B \multimap A]$, and $Nl \in \mathcal{S}_T[B]$. By the definition of $\mathcal{S}_T[B \multimap A]$ it follows that $(Ml)(Nl) \in \mathcal{S}_T[A]$. However, note that $(Ml)(Nl) = (M\, N)l$. Thus, we have shown that $(M\, N)l \in \mathcal{S}_T[A]$ as desired.

- $\mathit{pure}(\boldsymbol{t}')$:
  We need to show that for any $l \in \mathcal{S}_C[\Gamma]$, $\mathit{pure}(\boldsymbol{t}')l \in \mathcal{S}_T[\mathcal{B}(\boldsymbol{Q})]$.
  This follows in a straightforward manner since $(\mathit{pure}(\boldsymbol{t}')l = \mathit{pure}(\boldsymbol{t}'))$, and from the reduction rules in Figure 5.6, it follows that there are $\boldsymbol{v}, u_\sigma$ such that $(\boldsymbol{v}, u_\sigma, \mathit{pure}(\boldsymbol{t}'))$ reduces to a value configuration of the form $(\boldsymbol{v}', u_\sigma, x)$. Hence $M \longrightarrow_1 x$ and we have that $\mathit{pure}(\boldsymbol{t}') \in \mathcal{S}_T[\mathcal{B}(\boldsymbol{Q})]$.

- $\mathit{meas}(M)$:
  We need to show that for any $l \in \mathcal{S}_C[\Gamma]$, $\mathit{meas}(M)l \in \mathcal{S}_T[\overline{\boldsymbol{Q}}]$.
  Note that $\mathit{meas}(M)l = \mathit{meas}(Ml)$. Now, applying the induction hypothesis, we have that $Ml \in \mathcal{S}_T[\mathcal{B}(\boldsymbol{Q})]$. Hence, there exists a k such that Ml reduces to a term of the form x in at most k reduction steps. Applying the contextual reduction rule above we get that $\mathit{meas}(M)$

reduces to a term of the form $meas(x)$ in at most k steps. Finally, applying the reduction rule for measurement from Figure 5.6 gives us a value configuration of the desired form. Hence, we have that $meas(M)l \in \mathcal{S}_T[\overline{\boldsymbol{Q}}]$.

- $\mathcal{B}(u)(M)$:
  This case follows by an argument similar to the case for measurement.

□

Theorem 14. $M : \mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k] \implies M \in \mathcal{SN}_T$.

Proof. We know from Theorem 13 , that for all $l \in \mathcal{S}_C[\emptyset], Ml \in \mathcal{S}_T[A]$. But if $\Gamma = \emptyset, Ml = M$. Hence $M \in \mathcal{S}_T[A]$. From Lemma 10 it follows that $M \in \mathcal{SN}_T$. □

Theorem 15. $\mathcal{C} \in \mathrm{WF}_{\mathit{SVConf}}(\mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]) \implies \mathcal{C} \in \mathcal{SN}_C$.

Proof. Follows from Lemma 9 and Theorem 14. □

5.4 Denotational Semantics

In this section we describe a denotational model for the main calculus. Before going on to describe the denotational semantics, we first describe the categories that we use, and their structure relevant to our interpretation. We lay down the model using opposite categories of von Neumann algebras (Definition 24) and completely positive (sub)unital maps (abbreviated as **NCPSU**) and von Neumann algebras and multiplicative, involutive and unital maps (abbreviated as **NMIU**). Additionally we use the categories **Isometry**, and **Unitary** (§4.0.4), for the semantics of the quantum control fragment. A central contribution of this chapter is the functor that glues these categories, thus allowing us to model the interaction between the quantum control fragment and the main calculus denotationally. This interaction is summarized succinctly in Figure 5.7. Recall

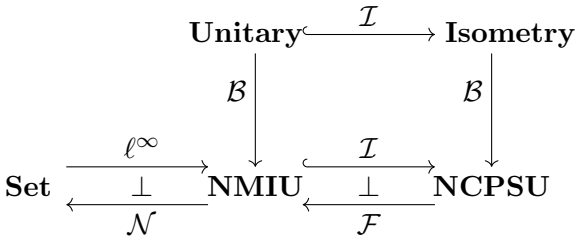


Figure 5.7: Categorical relations.

that a von Neumann algebra M is commutative if for every $x, y \in M$, we have that $xy = yx$. Such von Neumann algebras can be used to represent classical information. The Banach space $\ell^\infty(X) := \{f \colon X \to \mathbb{C} \mid \sup_{x \in X} |f(x)| < \infty\}$, for any set X, equipped with the supremum norm, can be identified with a commutative von Neumann algebra on the Hilbert space $\ell^2(X)$ via the isometric embedding given by $m_X \colon \ell^\infty(X) \to \mathcal{B}(\ell^2(X))$ and $m_X(f) = m_f$, where $m_f \colon \ell^2(X) \to \ell^2(X)$ is the bounded linear operator defined by $m_f(\sum_{x \in X} \alpha_x |x\rangle) := \sum_{x \in X} f(x)\alpha_x |x\rangle$. Then the image of this isometric embedding $m_X(\ell^\infty(X)) \subseteq \mathcal{B}(\ell^2(X))$ is a *commutative* von Neumann subalgebra of $\mathcal{B}(\ell^2(X))$. Here we abuse notation and we identify $\ell^\infty(X)$ with the von Neumann

algebra $m_X(\ell^\infty(X))$, which hopefully should not lead to confusion. Banach space with respect to the $\ell^\infty-$norm defined by $\|f\| := \sup_{x\in X} |f(x)|$.

Furthermore, elements $f \in \ell^\infty(X)$ can be identified with sequences of complex numbers $(\alpha_x)_{x\in X}$ indexed by elements of x via the correspondence $\alpha_x = f(x)$. An important special case for our development includes the von Neumann algebra $\ell^\infty(\mathbb{B})$, where $\mathbb{B} = \{true, false\}$, which can be identified with $\mathbb{C} \oplus \mathbb{C}$ and which corresponds to the interpretation for the type of (classical) bits in our language. Since we can safely identify $\mathcal{B}(\ell^2(\mathbb{B}))$ with the space $\mathrm{M}_2(\mathbb{C})$, then the image of $\ell^\infty(\mathbb{B})$ can be identified with the diagonal matrices of $\mathrm{M}_2(\mathbb{C})$, which are well-known to commute under multiplication. Another important example is given by the space $\ell^\infty(\mathbb{N})$ which we use to interpret the type Nat of classical natural numbers in our language. In this case, a natural number $n \in \mathbb{N}$ corresponds to the vector $(0 \cdots 0\ 1\ 0\ 0 \cdots)$ of $\ell^\infty(\mathbb{N})$ which has the number 1 at the n-th position and it is 0 otherwise.

Now we recall some of the definitions laid out in Chapter 3, §3.2, which are necessary for the categories **NCPSU**, **NMIU**. Let M and N be two von Neumann algebras (not necessarily over the same Hilbert space). A linear function $\varphi\colon M \to N$ is called *multiplicative/involutive/unital* if it preserves multiplication/involution/the unit of M, respectively. A linear map $\varphi\colon M \to N$ is called *subunital* if $\varphi(1_M) \leq 1_N$. When a map $\varphi\colon M \to N$ is both positive and subunital, then it restricts to a map $[0,1]_M \to [0,1]_N$ and we say that φ is *normal* whenever this restriction is a Scott-continuous map between the two dcpo's. A map $\varphi\colon M \to N$ is called an *NMIU*-map, if it is normal, multiplicative, involutive, and unital. If N is a von Neumann algebra (over H), then we can equip the set $\mathrm{M}_n(N)$ of $n \times n$ matrices with entries in N with the structure of a von Neumann algebra (over $\oplus_{i=1}^n H$). If $\varphi\colon N \to N'$ is a linear map between von Neumann algebras, then we can define a linear map $\varphi^{(n)}\colon \mathrm{M}_n(N) \to \mathrm{M}_n(N')$, for every $n \in \mathbb{N}$ by using φ componentwise. The map $\varphi\colon N \to N'$ is *completely positive* if $\varphi^{(n)}\colon \mathrm{M}_n(N) \to \mathrm{M}_n(N')$ is positive for every $n \in \mathbb{N}$. A map $\varphi\colon M \to N$ is called an *NCP(S)U*-map, if it is normal, completely positive, and (sub)unital.

The two main notions of morphisms that we use in our semantics are given by the NMIU and NCPSU maps. Every NMIU-map is also NCP(S)U, but not vice-versa (e.g., state preparation), hence NMIU-maps give a stronger notion of morphism. We write **NMIU** $\subseteq$ **NCPSU** for the *opposite* (sub)categories with objects von Neumann algebras and with NMIU/ NCPSU maps as morphisms between them. There are several reasons for working with the opposite categories:

- they allow us to reason appropriately about certain physical processes in the Heisenberg picture (as it is customary with von Neumann algebras), rather than in the Schrödinger one;

- the aforementioned opposite categories enjoy several categorical properties that we use in our semantics that are not known to hold for their non-opposite counterparts (e.g., monoidal closure, coproducts).

Symmetric Monoidal Structure. The categories **NMIU** and **NCPSU** have a symmetric monoidal structure when equipped with the *spatial tensor product* [K17, Prop 7.2]: the tensor unit can be identified with $\mathbb{C}$, and the tensor product of von Neumann algebras $M \subseteq B(H)$ and $N \subseteq B(K)$, written $M \otimes N$, is the smallest von Neumann subalgebra of $\mathcal{B}(H \otimes K)$ that contains the algebraic tensor product $M \odot N$ of the underlying vector spaces. The subcategory inclusion $\mathcal{I}\colon$ **NMIU** $\hookrightarrow$ **NCPSU** is a *strict* symmetric monoidal functor. Moreover, the category **NMIU** is a *closed* symmetric monoidal category and we write N^{*M} for its internal hom [K17].

Coproducts. Both categories **NMIU** and **NCPSU** have coproducts that are described in the same way: the initial object is the von Neumann algebra on the zero-dimensional Hilbert space 0 and the coproduct is given by direct sum with respect to the supremum norm [T12, Definition 3.4]. More specifically, if $M \subseteq \mathcal{B}(H)$ and $N \subseteq \mathcal{B}(K)$ are von Neumann algebras, then $M \oplus N$ is a Banach space (equipped with the supremum norm) which can be isometrically embedded into $\mathcal{B}(H \oplus K)$.

Linear Exponential. The assignment $\ell^\infty(-)$ can be extended to a functor by observing that for any function $f\colon X \to Y$ we get an NMIU morphism $(g \mapsto g \circ f)\colon \ell^\infty(Y) \to \ell^\infty(X)$ so that $\ell^\infty(f) := (g \mapsto g \circ f)^{\mathrm{op}}\colon \ell^\infty(X) \to \ell^\infty(Y)$ in **NMIU**. The adjunctions in Figure 5.7 are symmetric monoidal (see [CW16]) and this allows us to interpret the linear exponential, written as "!", following [BW96, B94]. We write $\mathcal{L} := \mathcal{I} \circ \ell^\infty$ and $\mathcal{R} := \mathcal{N} \circ \mathcal{F}$, so that $! := \mathcal{L}\mathcal{R}\colon$ **NCPSU** $\to$ **NCPSU**. We also write $\eta : Id \Rightarrow \mathcal{R}\mathcal{L}$ for the unit of the adjunction and $\mu : \mathcal{L}\mathcal{R} \Rightarrow Id$ for its counit.

Kleisli Exponential. Our language is call-by-value and we use Kleisli exponentials to interpret higher-order function types, so we now outline this construction. The inclusion functor $\mathcal{I}$ has a right adjoint $\mathcal{F}$ (Figure 5.7), so we have a natural bijection $\Phi_1\colon \mathbf{Hom}_{\mathbf{NCPSU}}(\mathcal{I}X, Y) \cong \mathbf{Hom}_{\mathbf{NMIU}}(X, \mathcal{F}Y)$. The category **NMIU** is a closed symmetric monoidal category, so we also have another natural bijection $\Phi_2\colon \mathbf{Hom}_{\mathbf{NMIU}}(X \otimes Z, Y) \cong \mathbf{Hom}_{\mathbf{NMIU}}(X, Y^{*Z})$. Consequently, for each morphism $f : X \otimes Z \to Y \in$ **NCPSU**, we have a morphism $\Phi_2(\Phi_1(f)) : X \to (\mathcal{F}Y)^{*Z} \in$ **NMIU**. Hence, we define the *evaluation map* $\epsilon_{Y,Z} : (\mathcal{F}Y)^{*Z} \otimes Z \to Y$ to be the image of $\mathrm{id} : \mathcal{F}Y^{*Z} \to \mathcal{F}Y^{*Z}$ under the inverse of the above construction $\epsilon_{Y,Z} := (\Phi_2 \circ \Phi_1)^{-1}(\mathrm{id})$.

The $\mathcal{B}$ functor. In order to model the interaction between the pure quantum subsystem and the main calculus, we define $\mathcal{B} :$ **Isometry** $\to$ **NCPSU** (see Figure 5.7) and show it is a functor. For an isometry $f\colon H_1 \to H_2$, the map $(f^* \circ (-) \circ f)$ is an NCPSU morphism $B(H_2) \to B(H_1)$, so by defining $\mathcal{B}(f) := (f^* \circ (-) \circ f)^{op}$, we get that $\mathcal{B}(f)\colon \mathcal{B}(H_1) \to \mathcal{B}(H_2)$ in **NCPSU**.

Lemma 13. *The map $\mathcal{B} :$* **Isometry** $\to$ **NCPSU** *as defined above is a functor.*

Proof. We first need to show that if f is an isometry then the function $\phi \mapsto f^* \circ \phi \circ f$ is a ncpu map from $B(H_2)$ to $B(H_1)$. Note that for $\phi = Id_{B(H_2)}$ we have,

$$f^* \circ (Id) \circ f = f^* \circ f = Id_{B(H_1)}$$

Hence the map is unital. To show that it is completely positive, recall that every von Neumann algebra is a C^*-algebra, and hence it suffices to check an equivalent condition for complete positivity of maps defined on C^*-algebra. Hence we need to show that for any $\{\phi_i\}_{i\in\mathbb{N}} \in B(H_2)$ and $\{\psi_i\}_{i\in\mathbb{N}} \in B(H_1)$, $\Sigma_{i,j}\psi_i^*(f^* \circ (\phi_i^*\phi_j) \circ f)\psi_j$ is positive. However we have,

$$\Sigma_{i,j}\psi_i^*(f^* \circ (\phi_i^*\phi_j) \circ f)\psi_j = (\Sigma_i\phi_i \circ f \circ \psi_i)^*(\Sigma_i\phi_i \circ f \circ \psi_i)$$

Hence,the above summation is positive.

To show that $\psi := f^* \circ (-) \circ f\colon B(H_2) \to B(H_1)$ is normal, it suffices to show that ψ has a predual $\tau\colon T(H_1) \to T(H_2)$. But the predual is obviously $\tau = f \circ (-) \circ f^*$. Indeed, one can easily see that τ is positive and therefore bounded with respect to the trace norm (it is also trace preserving). To check that τ is indeed the predual, we simply have to verify that for any $t \in T(H_1)$ and $b \in B(H_2)$ we have that

$$\mathrm{tr}(\psi(b)t) = \mathrm{tr}(b\tau(t)).$$

Indeed we have

$$\mathrm{tr}(\psi(b)t) = \mathrm{tr}(f^*bft) = \mathrm{tr}(bftf^*) = \mathrm{tr}(b\tau(t))$$

as required, so that ψ is indeed normal.

Every other property of a functor is trivially true. □

We remark that this functor restricts to a functor $\mathcal{B}$: **Unitary** $\to$ **NMIU** and we show that the functor $\mathcal{B}$: **Isometry** $\to$ **NCPSU** is strict monoidal.

Proposition 7. *The functor* $\mathcal{B}$: **Isometry** $\to$ **NCPSU** *is strict monoidal.*

Proof. From *eqn*(10) on page 185 in [T12], it follows that $\mathcal{B}(H_1) \otimes \mathcal{B}(H_2) \cong_{\mu_{H_1,H_2}} \mathcal{B}(H_1 \otimes H_2)$. Here μ_{H_1,H_2} is just the identity. Then for Hilbert spaces H_1, H_2, K_1, K_2 and isometries f, g we need to show the following:

$$\mathcal{B}(f) \otimes \mathcal{B}(g) = \mathcal{B}(f \otimes g)$$

First we show this for the generators of the von Neumann algebra $\mathcal{B}(H_1) \otimes B(H_2)$ which are of the form $\phi_1 \otimes \phi_2$. Note that,

$$\begin{aligned}
\mathcal{B}(f) \otimes \mathcal{B}(g)(\phi_1 \otimes \phi_2) &= (f^* \circ \phi_1 \circ f) \otimes (g^* \circ \phi_2 \circ g) \\
&= (f^* \otimes g^*) \circ (\phi_1 \otimes \phi_2) \circ (f \otimes g) \\
&= (f \otimes g)^* \circ (\phi_1 \otimes \phi_2) \circ (f \otimes g) \\
&= \mathcal{B}(f \otimes g)(\phi_1 \otimes \phi_2)
\end{aligned}$$

This argument extends to linear combinations of elements of the form $\phi_1 \otimes \phi_2 \in \mathcal{B}(H_1) \otimes \mathcal{B}(H_2)$. Recalling the definition of the spatial tensor product, note that $(\mathcal{B}(H_1) \odot \mathcal{B}(H_2))''$ is the closure of $\mathcal{B}(H_1) \otimes \mathcal{B}(H_2)$ in the ultraweak topology. Hence we have that the other elements of this von Neumann algebra are limits of such linear combinations in the ultraweak topology. Let X be a directed set and $\phi = lim_{x \in X} \phi_x$ the limit of a directed net $\{\phi_x\}_{x \in X}$, where the above equation holds for ϕ_x for all x. Then we have the following:

$$\begin{aligned}
\mathcal{B}(f) \otimes \mathcal{B}(g)(\phi) &= \mathcal{B}(f) \otimes \mathcal{B}(g)(lim_{x \in X} \phi_x) && (5.1)\\
&= lim_{x \in X} \mathcal{B}(f) \otimes \mathcal{B}(g)(\phi_x) && (5.2)\\
&= lim_{x \in X} \mathcal{B}(f \otimes g) && (5.3)\\
&= \mathcal{B}(f \otimes g)(lim_{x \in X} \phi_x) && (5.4)\\
&= \mathcal{B}(f \otimes g)(\phi) && (5.5)
\end{aligned}$$

In the above calculation, (3.2) holds, because normal maps are continuous with respect to the ultraweak topology, and the tensor product of normal maps is a normal map. Similarly, (3.4) holds because composition of normal maps is a normal map. □

With all the relevant categorical structure in place, we now define the denotational semantics for the main calculus using the data provided above.

Interpretation of Types. We interpret types as von Neumann algebras (Figure 5.8). The types corresponding to a linear call-by-value lambda-calculus are interpreted in the standard way. The type for natural numbers is interpreted as the commutative von Neumann algebra $\ell^\infty(\mathbb{N})$, i.e $[\![\mathrm{Nat}]\!] := \ell^\infty(\mathbb{N})$ which unsurprisingly corresponds to applying the left adjoint to $\mathbb{N}$. The interpretation of the type $\mathcal{B}(\boldsymbol{Q})$ uses the $\mathcal{B}$ functor that allows us to incorporate pure quantum primitives into our semantics.

$$\llbracket \mathrm{I} \rrbracket := \mathbb{C} \qquad \llbracket \mathrm{A} + \mathrm{B} \rrbracket := \llbracket \mathrm{A} \rrbracket \oplus \llbracket \mathrm{B} \rrbracket \qquad \llbracket \mathrm{A} \otimes \mathrm{B} \rrbracket := \llbracket \mathrm{A} \rrbracket \otimes \llbracket \mathrm{B} \rrbracket$$
$$\llbracket !\mathrm{A} \rrbracket := !\llbracket \mathrm{A} \rrbracket \qquad \llbracket \mathrm{A} \multimap \mathrm{B} \rrbracket := \llbracket \mathrm{A} \rrbracket \multimap \llbracket \mathrm{B} \rrbracket \qquad \llbracket \mathrm{Nat} \rrbracket := \ell^\infty(\mathbb{N})$$
$$\llbracket \mathcal{B}(\boldsymbol{Q}) \rrbracket := \mathcal{B}\llbracket \boldsymbol{Q} \rrbracket$$

Figure 5.8: Interpretation of types.

Interpretation of Typing judgements. A typing context $\Gamma := x_1 \colon \mathrm{A}_1, \ldots, x_n \colon \mathrm{A}_n$ is interpreted in the usual way as $\llbracket \Gamma \rrbracket := \llbracket \mathrm{A}_1 \rrbracket \otimes \cdots \otimes \llbracket \mathrm{A}_n \rrbracket$. A typing judgment of the form $\Delta \vdash M \colon \mathrm{A}$ is interpreted as a morphism $\llbracket \Delta \vdash M \colon \mathrm{A} \rrbracket \colon \llbracket \Delta \rrbracket \to \llbracket \mathrm{A} \rrbracket$ in **NCPSU** and we often abbreviate this by writing $\llbracket M \rrbracket$. Before we lay down the interpretation of typing judgements, we introduce the notation used therein.

We use the symbol $\diamond$ for the canonical discarding map that may be defined at types !A and non-linear contexts !Δ. For a non-linear context !Δ we may define a copy (or diagonal) map $d_{!\Delta} \colon \llbracket !\Delta \rrbracket \to \llbracket !\Delta \rrbracket \otimes \llbracket !\Delta \rrbracket$ in the standard way and also the map *split* defined as,

$$\mathit{split} := \left(\llbracket !\Delta, \Sigma_1, \Sigma_2 \rrbracket \xrightarrow{d \otimes \mathrm{id} \otimes \mathrm{id}} \llbracket !\Delta, !\Delta, \Sigma_1, \Sigma_2 \rrbracket \xrightarrow{\cong} \llbracket !\Delta, \Sigma_1 \rrbracket \otimes \llbracket !\Delta, \Sigma_2 \rrbracket \right).$$

Given a non-linear context $!\Delta := \{x_1 \colon !\mathrm{A}_1, \ldots, x_n \colon !\mathrm{A}_n\}$, the *promotion* map is defined as usual by $\rho_{!\Delta} := \llbracket !\Delta \rrbracket \xrightarrow{\cong} \mathcal{L}(\mathcal{R}\llbracket \mathrm{A}_1 \rrbracket \times \cdots \times \mathcal{R}\llbracket \mathrm{A}_n \rrbracket) \xrightarrow{\mathcal{L}\eta} !\mathcal{L}(\mathcal{R}\llbracket \mathrm{A}_1 \rrbracket \times \cdots \times \mathcal{R}\llbracket \mathrm{A}_n \rrbracket) \xrightarrow{\cong} !\llbracket !\Delta \rrbracket$. We write $\gamma_{A,B} \colon A \otimes B \to B \otimes A$ for the monoidal symmetry. The successor map is given by $s \colon \llbracket \mathrm{Nat} \rrbracket \to \llbracket \mathrm{Nat} \rrbracket :: (x_1, x_2, \ldots) \mapsto (0, x_1, x_2, \ldots)$. The full interpretation of typing judgements is given in Figure 5.9.

Note that the interpretation of the measurement operation crucially relies on the following theorem.

Theorem 16. *For every pure quantum type $\boldsymbol{Q}$, there exist a set X, an NMIU isomorphism $\alpha_{\overline{\boldsymbol{Q}}} : \llbracket \overline{\boldsymbol{Q}} \rrbracket \cong \ell^\infty(X)$, and an isometric isomorphism $\beta_{\boldsymbol{Q}} : \llbracket \boldsymbol{Q} \rrbracket \cong \ell^2(X)$. Moreover, $\forall \boldsymbol{b} \in$ **Basis Terms**, $\exists x \in X$ s.t. $\beta_{\boldsymbol{Q}}(\llbracket \boldsymbol{b} \rrbracket) = \alpha_{\overline{\boldsymbol{Q}}}(\llbracket \overline{\boldsymbol{b}} \rrbracket) = |x\rangle$.*

Proof. (Sketch) Proof is by induction on the form of $\boldsymbol{Q}$.

- $\boldsymbol{Q} = \boldsymbol{I}$:

  $$\llbracket \boldsymbol{I} \rrbracket = \mathbb{C} \cong \ell^2(\{*\}) \quad \llbracket \overline{\boldsymbol{I}} \rrbracket = \mathbb{C} \cong \ell^\infty(\{*\})$$

  Moreover , in this case we have $\boldsymbol{b} = *$, and $\overline{\boldsymbol{b}} = *$. We also know that $\llbracket * \rrbracket = \llbracket \overline{\boldsymbol{b}} \rrbracket = 1 \in \mathbb{C}$. However, the scalar 1 can be identified with $|*\rangle$. Hence the second statement of the theorem also holds.

- $\boldsymbol{Q} = \mathbf{qnat}$:

  $$\llbracket \mathbf{qnat} \rrbracket = \ell^2(\mathbb{N}) \quad \llbracket \overline{\mathbf{qnat}} \rrbracket = \llbracket \mathrm{Nat} \rrbracket = \ell^\infty(\mathbb{N})$$

  In this case the isomorphisms are identity maps, and we know that $\{\boldsymbol{b}_i\}_{i \in \mathbb{N}} = \{|n\rangle\}_{n \in \mathbb{N}}$ because of Figure 4.9. Similarly, $\{\overline{\boldsymbol{b}_i}\}_{i \in \mathbb{N}} = \{|n\rangle\}_{n \in \mathbb{N}}$. Hence, the second statement of the theorem also holds.

$$[\![\Gamma, y : \mathrm{B}, x : \mathrm{A}, \Sigma \vdash M : \mathrm{C}]\!] := [\![\Gamma, x : \mathrm{A}, y : \mathrm{B}, \Sigma \vdash M : \mathrm{C}]\!] \circ (id_{[\![\Gamma]\!]} \otimes \gamma \otimes id_{[\![\Sigma]\!]})$$

$$[\![!\Delta \vdash * : \mathrm{I}]\!] := [\![!\Delta]\!] \xrightarrow{\diamond_{!\Delta}} [\![\mathrm{I}]\!]$$

$$[\![!\Delta, x : \mathrm{A} \vdash x : \mathrm{A}]\!] := [\![!\Delta]\!] \otimes [\![\mathrm{A}]\!] \xrightarrow{\diamond_{!\Delta} \otimes id} [\![\mathrm{I}]\!] \otimes [\![\mathrm{A}]\!] \cong [\![\mathrm{A}]\!]$$

$$[\![\Delta \vdash \mathit{inl}(N) : \mathrm{A} + \mathrm{B}]\!] := i_1 \circ [\![\Delta \vdash N : \mathrm{A}]\!]$$

$$[\![\Delta \vdash \mathit{inr}(M) : \mathrm{A} + \mathrm{B}]\!] := i_2 \circ [\![\Delta \vdash M : \mathrm{B}]\!]$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{case}\, L\, \mathit{of}\{\mathit{inl}(x) \to M,\ \mathit{inr}(y) \to N\} : \mathrm{C}]\!]$$
$$:= [[\![!\Delta, \Sigma_2, x : \mathrm{A} \vdash M : \mathrm{C}]\!], [\![!\Delta, \Sigma_2, y : \mathrm{B} \vdash N : \mathrm{C}]\!]] \circ \cong \circ ([\![!\Delta, \Sigma_1 \vdash L : \mathrm{A} + \mathrm{B}]\!] \otimes id) \circ \mathit{split}$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash M \otimes N : \mathrm{A} \otimes \mathrm{B}]\!] := ([\![!\Delta, \Sigma_1 \vdash M : \mathrm{A}]\!] \otimes [\![!\Delta, \Sigma_2 \vdash N : \mathrm{B}]\!]) \circ \mathit{split}$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\ x \otimes y = M\ \mathit{in}\ N : \mathrm{C}]\!]$$
$$:= [\![!\Delta, x : \mathrm{A}, y : \mathrm{B}, \Sigma_2 \vdash N : \mathrm{C}]\!] \circ ([\![!\Delta, \Sigma_1 \vdash M : \mathrm{A} \otimes \mathrm{B}]\!] \otimes id) \circ \mathit{split}$$

$$[\![!\Delta \vdash \mathit{lift}(M) : !\mathrm{A}]\!] := ![\![!\Delta \vdash M : \mathrm{A}]\!] \circ \rho_{!\Delta}$$

$$[\![\Delta \vdash \mathit{force}(M) : \mathrm{A}]\!] := \mu \circ [\![\Delta \vdash M : !\mathrm{A}]\!]$$

$$[\![\Delta \vdash \lambda x.M : \mathrm{A} \multimap \mathrm{B}]\!] := \Phi_2 \circ \Phi_1([\![\Delta, x : \mathrm{A} \vdash M : \mathrm{B}]\!])$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash M\, N : \mathrm{B}]\!] := \epsilon_{\mathrm{B},\mathrm{A}} \circ ([\![!\Delta, \Sigma_1 \vdash M : \mathrm{A} \multimap \mathrm{B}]\!] \otimes [\![!\Delta, \Sigma_2 \vdash N : \mathrm{A}]\!]) \circ \mathit{split}$$

$$[\![!\Delta \vdash \mathit{zero} : \mathrm{Nat}]\!] := [\![!\Delta]\!] \xrightarrow{\diamond_{!\Delta}} [\![I]\!] \xrightarrow{i_0} [\![\mathrm{Nat}]\!]$$

$$[\![\Delta \vdash \mathit{succ}(M)]\!] := s \circ [\![\Delta \vdash M : \mathrm{Nat}]\!]$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{match}\, L\, \mathit{with}\{\mathit{zero} \to M,\ \mathit{succ}(x) \to N\} : \mathrm{A}]\!]$$
$$:= [[\![!\Delta, \Sigma_2 \vdash M : \mathrm{A}]\!], [\![!\Delta, \Sigma_2, x : \mathrm{Nat} \vdash N : \mathrm{A}]\!]] \circ \cong \circ (id \otimes [\![!\Delta, \Sigma_1 \vdash L : \mathrm{Nat}]\!]) \circ \mathit{split}$$

$$[\![!\Delta \vdash \mathit{pure}(\boldsymbol{t}) : \mathcal{B}(\boldsymbol{Q})]\!] := \mathcal{B}[\![\boldsymbol{t}]\!] \circ \diamond_{!\Delta}$$

$$\left[\!\left[\Delta \vdash \mathit{meas}(M) : \overline{\boldsymbol{Q}}\right]\!\right] := m^{\boldsymbol{Q}} \circ [\![\Delta \vdash M : \mathcal{B}(\boldsymbol{Q})]\!]$$

$$[\![\Delta \vdash \mathcal{B}(u)(M) : \mathcal{B}(\boldsymbol{Q}_2)]\!] := \mathcal{B}[\![u]\!] \circ [\![\Delta \vdash M : \mathcal{B}(\boldsymbol{Q}_1)]\!]$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\ \mathcal{B}(z) = M\ \mathit{in}\ N : \mathrm{A}]\!]$$
$$:= [\![!\Delta, \Sigma_2, z : \mathcal{B}(\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2) \vdash N : \mathrm{A}]\!] \circ ([\![!\Delta, \Sigma_1 \vdash M : \mathcal{B}(\boldsymbol{Q}_1) \otimes \mathcal{B}(\boldsymbol{Q}_2)]\!] \otimes id) \circ \mathit{split}$$

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash \mathit{let}\ \mathcal{B}(x \otimes y) = M\ \mathit{in}\ N : \mathrm{A}]\!]$$
$$:= [\![!\Delta, \Sigma_2, x : \mathcal{B}(\boldsymbol{Q}_1), y : \mathcal{B}(\boldsymbol{Q}_2) \vdash N : \mathrm{A}]\!] \circ ([\![!\Delta, \Sigma_1 \vdash M : \mathcal{B}(\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2)]\!] \otimes id) \circ \mathit{split}$$

Figure 5.9: Interpretation of typing judgements.

- $\boldsymbol{Q} = \boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2$:
  By induction hypothesis we have that $\alpha_{\overline{\boldsymbol{Q}_1}} : \left[\!\left[\overline{\boldsymbol{Q}_1}\right]\!\right] \to \ell^\infty(X_1)$ and $\beta_{\boldsymbol{Q}_1} : [\![\boldsymbol{Q}_1]\!] \to \ell^2(X_1)$ for some set X_1. Furthermore, for all i, we have the relation $\beta_{\boldsymbol{Q}_1}([\![\boldsymbol{b_i}]\!]) = \alpha_{\overline{\boldsymbol{Q}_1}}(\left[\!\left[\overline{\boldsymbol{b_i}}\right]\!\right]) = |x_i\rangle$ for $x_i \in X_1$. Similarly, for $\boldsymbol{Q}_2$ we have, $\alpha_{\overline{\boldsymbol{Q}_2}} : \left[\!\left[\overline{\boldsymbol{Q}_2}\right]\!\right] \to \ell^\infty(X_2)$ and $\beta_{\boldsymbol{Q}_2} : [\![\boldsymbol{Q}_2]\!] \to \ell^2(X_2)$ for some set X_2. We also have that $\beta_{\boldsymbol{Q}_2}([\![\boldsymbol{b'_i}]\!]) = \alpha_{\overline{\boldsymbol{Q}_2}}(\left[\!\left[\overline{\boldsymbol{b'_i}}\right]\!\right]) = |x'_i\rangle$ for $x'_i \in X_2$. Consider the following:

$$[\![\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2]\!] = [\![\boldsymbol{Q}_1]\!] \otimes [\![\boldsymbol{Q}_2]\!] \cong_{\beta_{\boldsymbol{Q}_1} \otimes \beta_{\boldsymbol{Q}_2}} \ell^2(X_1) \otimes \ell^2(X_2) \cong \ell^2(X_1 \times X_2)$$

$$\left[\!\left[\boldsymbol{Q}_1 \otimes \boldsymbol{Q}_2\right]\!\right] = \left[\!\left[\boldsymbol{Q}_1\right]\!\right] \otimes \left[\!\left[\boldsymbol{Q}_2\right]\!\right] \cong_{\alpha_{\overline{\boldsymbol{Q}_1}} \otimes \alpha_{\overline{\boldsymbol{Q}_2}}} \ell^\infty(X_1) \otimes \ell^\infty(X_2) \cong \ell^\infty(X_1 \times X_2)$$

The second statement holds because $\beta_{\boldsymbol{Q}_1\otimes\boldsymbol{Q}_2}(\left[\!\left[\boldsymbol{b}_i\otimes\boldsymbol{b}'_j\right]\!\right]) = \beta_{\boldsymbol{Q}_1}([\![\boldsymbol{b}_i]\!])\otimes\beta_{\boldsymbol{Q}_2}(\left[\!\left[\boldsymbol{b}'_j\right]\!\right]) = |x_i\rangle\otimes\left|x'_j\right\rangle$. Similar statement holds for $\{\overline{b_i}\}_{i\in I}$.

- $\boldsymbol{Q} = \boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2$:
  By induction hypothesis we have that $\left[\!\left[\overline{\boldsymbol{Q}_1}\right]\!\right] \cong_{\alpha_{\overline{\boldsymbol{Q}_1}}} \ell^\infty(X_1)$ and $[\![\boldsymbol{Q}_1]\!] \cong_{\beta_{\boldsymbol{Q}_1}} \ell^2(X_1)$ for some set X_1. Similarly, for $\boldsymbol{Q}_2$ we have, $\left[\!\left[\overline{\boldsymbol{Q}_2}\right]\!\right] \cong_{\alpha_{\overline{\boldsymbol{Q}_2}}} \ell^\infty(X_2)$ and $[\![\boldsymbol{Q}_2]\!] \cong_{\beta_{\boldsymbol{Q}_2}} \ell^2(X_2)$ for some set X_2. Consider the following:

$$[\![\boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2]\!] = [\![\boldsymbol{Q}_1]\!] \oplus [\![\boldsymbol{Q}_2]\!] \cong_{\beta_{\boldsymbol{Q}_1}\oplus\beta_{\boldsymbol{Q}_2}} \ell^2(X_1)\oplus\ell^2(X_2) \cong \ell^2(X_1 \sqcup X_2)$$

$$\left[\!\left[\overline{\boldsymbol{Q}_1 \oplus \boldsymbol{Q}_2}\right]\!\right] = \left[\!\left[\overline{\boldsymbol{Q}_1}\right]\!\right] \oplus \left[\!\left[\overline{\boldsymbol{Q}_2}\right]\!\right] \cong_{\alpha_{\overline{\boldsymbol{Q}_1}}\oplus\alpha_{\overline{\boldsymbol{Q}_2}}} \ell^\infty(X_1)\oplus\ell^\infty(X_2) \cong \ell^\infty(X_1 \sqcup X_2)$$

The second statement holds because $\beta_{\boldsymbol{Q}_1\oplus\boldsymbol{Q}_2}([\![inl\boldsymbol{b}_i]\!]) = inl(\beta_{\boldsymbol{Q}_1}([\![\boldsymbol{b}_i]\!])) = |x_i\rangle^{ext}$. Here $|x_i\rangle^{ext}$ is just the extension of the functional $|x_i\rangle$ defined on X_1 to $X_1 \sqcup X_2$ by defining it to be identically zero on whole of X_2. Same is true for the right injection. Similar statement holds for $\{\overline{b_i}\}_{i\in I}$.

Note that this argument works because we deal with separable Hilbert spaces only, and hence the set $\{\boldsymbol{b}_i\}_{i\in I}$ is always countable. □

Using the above theorem, measurement is interpreted as the map $m^{\boldsymbol{Q}} : \mathcal{B}([\![\boldsymbol{Q}]\!]) \to \left[\!\left[\overline{\boldsymbol{Q}}\right]\!\right]$ defined as $m^{\boldsymbol{Q}} := \alpha^{op}_{\overline{\boldsymbol{Q}}} \circ m^{op}_X \circ \mathcal{B}(\beta_{\boldsymbol{Q}})$.

Interpretation of Quantum Configurations. A well-formed i*total* configuration $(\boldsymbol{t}, u_\sigma, M)$ with $\cdot \vdash \boldsymbol{t} \colon \boldsymbol{Q}$, $\Vdash u_\sigma \colon (\boldsymbol{Q}, \boldsymbol{Q}'_1 \otimes \cdots \otimes \boldsymbol{Q}'_m)$, and $x_1 \colon \mathcal{B}(\boldsymbol{Q}'_1), \ldots, x_m \colon \mathcal{B}(\boldsymbol{Q}'_m) \vdash M \colon \mathrm{A}$ is interpreted as the NCPSU morphism

$$[\![(\boldsymbol{t}, u_\sigma, M) : \mathrm{A}]\!] := \left(\mathbb{C} \xrightarrow{\mathcal{B}([\![u_\sigma \boldsymbol{t}]\!])} \begin{array}{c} \mathcal{B}(\boldsymbol{Q}'_1 \otimes \cdots \otimes \boldsymbol{Q}'_m) \\ = \\ \mathcal{B}(\boldsymbol{Q}'_1) \otimes \cdots \otimes \mathcal{B}(\boldsymbol{Q}'_m) \end{array} \xrightarrow{[\![M]\!]} \mathrm{A} \right).$$

We naturally extend the interpretation defined for a total configuration above to a configuration with auxiliary quantum types as follows:

$$\begin{array}{ccc} \mathbb{C} & \xrightarrow{[\![(\boldsymbol{t}, u_\sigma, M) : \mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]]\!]} & [\![\mathrm{A}]\!] \otimes \mathcal{B}(\boldsymbol{Q}'_1) \otimes \cdots \otimes \mathcal{B}(\boldsymbol{Q}'_k) \\ \downarrow{\scriptstyle \mathcal{B}([\![u_\sigma \boldsymbol{t}]\!])} & := & \uparrow{\scriptstyle [\![x_1 \colon \mathcal{B}(\boldsymbol{Q}''_1), \ldots, x_m \colon \mathcal{B}(\boldsymbol{Q}''_m) \vdash M \colon A]\!] \otimes id} \\ \mathcal{B}(\boldsymbol{Q}''_1 \otimes \ldots \otimes \boldsymbol{Q}''_{m+k}) & \xrightarrow{\cong} & \mathcal{B}(\boldsymbol{Q}''_1) \otimes \cdots \otimes \mathcal{B}(\boldsymbol{Q}''_m) \otimes \mathcal{B}(\boldsymbol{Q}'_1) \otimes \cdots \otimes \mathcal{B}(\boldsymbol{Q}'_k) \end{array}$$

5.4.1 Soundness and Adequacy

We can now show that our semantic interpretation is sound with respect to single-step reduction. In order to do so, we first show that the denotational interpretations are preserved under substitution.

Lemma 14 (*Substitution*). *Given typing judgements* $!\Delta, \Sigma_1 \vdash V : A$,$!\Delta, \Sigma_2, x : A \vdash M : B$ *and* $!\Delta, \Sigma_1, \Sigma_2 \vdash M[V/x] : B$ *(Lemma 6), the following equation holds:*

$$[\![!\Delta, \Sigma_1, \Sigma_2 \vdash M[V/x] : B]\!] = [\![!\Delta, \Sigma_2, x : A \vdash M : B]\!] \circ ([\![!\Delta, \Sigma_1 \vdash V : A]\!] \otimes id) \circ split$$

Proof. Proof is by induction on the form of the term M. □

The next lemma is a consequence of the one above.

Lemma 15. *The following equations hold for valid typing judgements:*

$$\begin{aligned}
[\![\Delta \vdash (\lambda x.M)V : A]\!] &= [\![\Delta \vdash M[V/x] : A]\!] \\
[\![\Delta \vdash let\ x \otimes y = V \otimes W\ in\ M : A]\!] &= [\![\Delta \vdash M[V/x, W/y] : A]\!] \\
[\![\Delta \vdash case\, inl(V)\ of\{inl(x) \to M,\ inr(y) \to N\} : A]\!] &= [\![\Delta \vdash M[V/x] : A]\!] \\
[\![\Delta \vdash case\, inr(W)\ of\{inl(x) \to M,\ inr(y) \to N\} : A]\!] &= [\![\Delta \vdash M[W/y] : A]\!] \\
[\![\Delta \vdash force(lift(M)) : A]\!] &= [\![\Delta \vdash M : A]\!] \\
[\![\Delta \vdash match\, zero\, with\, \{zero \to M,\ succ(x) \to N\} : A]\!] &= [\![\Delta \vdash M : A]\!] \\
[\![\Delta \vdash match\, succ(V)\ with\, \{zero \to M,\ succ(x) \to N\} : A]\!] &= [\![\Delta \vdash N[V/x] : A]\!]
\end{aligned}$$

Proof. Follows from Lemma 14. □

Finally, we prove another additional lemma before stating and proving soundness for the small-step reduction.

Lemma 16. *For a quantum type* $\boldsymbol{Q}$ *and finitely many basis terms* $\{\boldsymbol{b_i}\}_{i \in I}$ *of* $\boldsymbol{Q}$*, the following equation holds:*

$$\mathcal{B}^r([\![\Sigma_i p_i \boldsymbol{b_i}]\!]) \circ \mathcal{B}^r(\beta_{\boldsymbol{Q}}) \circ m_X \circ \alpha_{\overline{\boldsymbol{Q}}} = \Sigma_i |p_i|^2 \overline{[\![\boldsymbol{b_i}]\!]}$$

provided that $\cdot \vdash \Sigma_i p_i \boldsymbol{b_i}$*. Here* $\mathcal{B}^r$: **Isometry** $\to$ $\boldsymbol{W}$ *is defined as* $\mathcal{B}^r(H) := \mathcal{B}(H)$ *on objects, and for a morphism* $f \in$ **Isometry**, $\mathcal{B}^r(f)(\phi) := f^* \circ \phi \circ f$ *and* **NCPSU** $= \boldsymbol{W}^{op}$*.*

Proof. It follows from the definition of the functor $\mathcal{B}^r$, that it suffices to show the following:

$$\mathcal{B}^r([\![\boldsymbol{b_i}]\!]) \circ \mathcal{B}^r(\beta_{\boldsymbol{Q}}) \circ m_X \circ \alpha_{\overline{\boldsymbol{Q}}} = \overline{[\![\boldsymbol{b_i}]\!]} \quad \forall i$$

This follows from the definition of $\mathcal{B}^r, m_X$, Theorem 16 and the observation that $\{\boldsymbol{b_i}\}_{i \in I} = \{|x\rangle\}_{x \in X}$. □

Proposition 8. *For a well-formed quantum configuration* $\mathcal{C} : A[\boldsymbol{Q'}_1, \ldots, \boldsymbol{Q'}_k]$*, we have* $[\![\mathcal{C} : A[\boldsymbol{Q'}_1, \ldots, \boldsymbol{Q'}_k]]\!] = \Sigma_{\mathcal{C}'} p [\![\mathcal{C}' : A[\boldsymbol{Q'}_1, \ldots, \boldsymbol{Q'}_k]]\!]$.

Proof. Note that, the proof essentially reduces to proving the statement for the relation $\to_p$, since $\rightsquigarrow_p$ holds after we rewrite the first component. More precisely, if $\mathcal{C}$ doesn't reduce further, the statement holds trivially. If it does, then since the equational theory for the quantum control fragment is complete(Theorem 8), it suffices to show the statement for $\to_p$. We can assume without loss of generality that $\mathcal{C} = (\boldsymbol{t}, u_\sigma, M)$ for some quantum value $\boldsymbol{t}$, some unitary u_σ and

some term M. Now the proof is by induction on the form of the term M. Note that if M is a value, then the above equation holds trivially. Let $\mathcal{C} = (\boldsymbol{t}, u_\sigma, M\,N)$, where M is not a value. Then we know from the contextual rule fo Figure 5.6 that the only possible reductions take the form $(\boldsymbol{t}, u_\sigma, M\,N) \to_{p_i} (\boldsymbol{t}_i, u_{\sigma_i}, M_i\,N)$ whenever $(\boldsymbol{t}, u_\sigma, M) \to_{p_i} (\boldsymbol{t}_i, u_{\sigma_i}, M_i)$. By the induction hypothesis , we know that $[\![(\boldsymbol{t}, u_\sigma, M)]\!] = \Sigma_i p_i [\![(\boldsymbol{t}_i, u_{\sigma_i}, M_i)]\!]$. Hence we have the following:

$$
\begin{aligned}
&[\![(\boldsymbol{t}, u_\sigma, M\,N) : \mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]]\!] \\
&= [\![x_1, \ldots, x_m : \mathcal{B}(\boldsymbol{Q}_m) \vdash M\,N : \mathrm{A}]\!] \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots} \circ \mathcal{B}([\![u_\sigma \boldsymbol{t}]\!]) \\
&= ((\epsilon_{\mathrm{A,B}} \circ ([\![x_1, \ldots, x_j \vdash M : \mathrm{B} \multimap \mathrm{A}]\!] \otimes [\![x_{j+1}, \ldots \vdash N : \mathrm{B}]\!])) \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \circ \mathcal{B}([\![\boldsymbol{v}]\!]) \\
&= ((\epsilon_{\mathrm{A,B}} \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \circ (id_{\mathrm{B}\multimap\mathrm{A}} \otimes [\![x_{j+1}, \ldots \vdash N : \mathrm{B}]\!] \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \\
&\quad \circ ([\![x_1, \ldots, x_j \vdash M : \mathrm{B} \multimap \mathrm{A}]\!] \otimes id_{\mathcal{B}(\boldsymbol{Q}_{j+1})\ldots} \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \circ \mathcal{B}([\![u_\sigma \boldsymbol{t}]\!]) \\
&= ((\epsilon_{\mathrm{A,B}} \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \\
&\quad \circ (id_{\mathrm{B}\multimap\mathrm{A}} \otimes [\![x_{j+1}, \ldots \vdash N : \mathrm{B}]\!] \otimes id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \circ (\Sigma_i p_i [\![(\boldsymbol{t}_i, u_{\sigma_i}, M_i)]\!]) \otimes (id_{\mathcal{B}(\boldsymbol{Q}'_1)\ldots}) \\
&= \Sigma_i p_i [\![(\boldsymbol{t}_i, u_{\sigma_i}, M_i N)]\!]
\end{aligned}
$$

The case when $M\,N = (\lambda x.M')\,V$ is a consequence of 14. The other cases in the induction step for $M\,N$ follow similarly. We now prove the induction step for the term $\mathit{pure}(\boldsymbol{t})$. From Figure 5.6, it follows that we need this equation to hold:

$$[\![(\boldsymbol{t}, u_\sigma, \mathit{pure}(\boldsymbol{t}')) : \mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}'})[\boldsymbol{Q}_{\boldsymbol{t}}]]\!] = [\![(\boldsymbol{t} \otimes \boldsymbol{t}', u_{\sigma_{swap}} \circ (u_\sigma \otimes u_{id}), x) : \mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}'})[\boldsymbol{Q}_{\boldsymbol{t}}]]\!]$$

Note the following:

$$
\begin{aligned}
[\![(\boldsymbol{t}, u_\sigma, \mathit{pure}(\boldsymbol{t}')) : \mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}'})[\boldsymbol{Q}_{\boldsymbol{t}}]]\!] &= ([\![\cdot \vdash \mathit{pure}(\boldsymbol{t}') : \mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}'})]\!] \otimes id_{\mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}})}) \circ \mathcal{B}([\![u_\sigma \boldsymbol{t}]\!]) \\
&= (\mathcal{B}([\![\boldsymbol{t}']\!]) \otimes id_{\mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}})}) \circ \mathcal{B}([\![u_\sigma \boldsymbol{t}]\!]) \\
&= (id_{\mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}'})} \otimes id_{\mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}})}) \circ \mathcal{B}([\![u_{\sigma_{swap}} \circ (u_\sigma \otimes u_{id})(\boldsymbol{t} \otimes \boldsymbol{t}')]\!]) \\
&= ([\![x]\!] \otimes id_{\mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}})}) \circ \mathcal{B}([\![u_{\sigma_{swap}}(u_\sigma \boldsymbol{t}) \otimes \boldsymbol{t}']\!]) \\
&= [\![(\boldsymbol{t} \otimes \boldsymbol{t}', u_{\sigma_{swap}} \circ (u_\sigma \otimes u_{id}), x) : \mathcal{B}(\boldsymbol{Q}_{\boldsymbol{t}'})[\boldsymbol{Q}_{\boldsymbol{t}}]]\!]
\end{aligned}
$$

The proof for the induction step for the term $\mathit{meas}(M)$ follows from Lemma 16. Rest of the cases follow using arguments similar to the above cases. □

We can now show that our semantic interpretation is sound with respect to single step reduction.

Theorem 17 (Soundness). *For $\mathcal{C} \in \mathrm{WF}_{\mathtt{SVConf}}(A[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k])$, if $\mathcal{C} \notin \mathtt{VConf}$, then*

$$[\![\mathcal{C} : \mathrm{A}]\!] = \sum_{\mathcal{C} \rightsquigarrow_p \mathcal{C}'} p [\![\mathcal{C}' : \mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k]]\!],$$

where the sum ranges over all possible reducts $\mathcal{C} \rightsquigarrow_p \mathcal{C}'$.

Proof. Follows from Proposition 8. □

Here we show that strong adequacy holds for our denotational semantics with respect to the operational semantics. The *net probability* of a reduction $R = \mathcal{C} \rightsquigarrow_{p_0} \mathcal{C}_1 \rightsquigarrow_{p_1} \rightsquigarrow_{p_2} ..$ is defined as $P(R) := \Pi_i p_i$. The set of reductions R from a configuration $\mathcal{C}_1$ to $\mathcal{C}_2$ is denoted by $\mathit{Red}(\mathcal{C}_1, \mathcal{C}_2)$. Now for a configuration $\mathcal{C}$ the probability that it reaches a value configuration $\mathcal{V}$ in at most n steps is defined as $P(\mathcal{C} \to_{\leq n} \mathcal{V}) := \sum_{R \in \mathit{Red}(\mathcal{C}, \mathcal{V})} P(R)$.

Theorem 18 (Strong Adequacy). *For* $\mathcal{C} \in \mathrm{WF}_{\mathcal{SVConf}}(\mathrm{A}[\boldsymbol{Q}'_1, \ldots, \boldsymbol{Q}'_k])$,

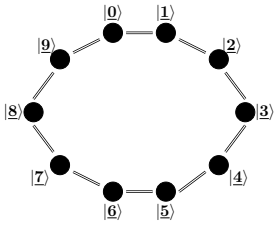
$$[\![\mathcal{C}]\!] = \sum_{\mathcal{V} \in \mathcal{VConf}} P(\mathcal{C} \to_* \mathcal{V})[\![\mathcal{V}]\!],$$

where $P(\mathcal{C} \to_* \mathcal{V})$ *indicates the overall probability that* $\mathcal{C}$ *reduces to* $\mathcal{V}$.

Proof. The result is a consequence of Theorem 14 and Theorem 17. □

5.5 Example: Quantum Walk Search

Given a graph with some marked vertices, a quantum walk search is the quantum analogue of a random walk [CCDFGS03], which looks for these marked vertices. Each node in the graph represents the state of two quantum registers. A condition on the application of a coin operator on the first register decides the direction in which a walker should take the next step. A step in this direction is represented by the application of a unitary on the second register. After a given number of steps, measurement is performed on the second register to reveal whether the walker has arrived on a marked node. Here, we present a k-step quantum walk on a cycle with 10 nodes for simplicity in our language. Representing the second register with qubits, in this case, would require 4 qubits, whereas we represent the state of the second register with a single **qnat**. In the unitary $\mathrm{CNOT}_{\mathbf{qnat}}$ below, we use some (hopefully obvious) syntactic sugar with the expressions involving **n** in order to avoid writing all the cases; the letter **n** here should not be seen as a variable.



$$u_1 := \left\{ \begin{array}{lcll} |\underline{\mathbf{2n}}\rangle & \mapsto & |\underline{\mathbf{2n{+}1}}\rangle, & \mathbf{n} \leq 4 \\ |\underline{\mathbf{2n{+}1}}\rangle & \mapsto & |\underline{\mathbf{2n}}\rangle, & \mathbf{n} \leq 4 \\ |\underline{\mathbf{y{+}10}}\rangle & \mapsto & |\underline{\mathbf{y{+}10}}\rangle & \end{array} \right\} \quad u_2 := \left\{ \begin{array}{lcll} |\underline{\mathbf{0}}\rangle & \mapsto & |\underline{\mathbf{9}}\rangle & \\ |\underline{\mathbf{2n{+}2}}\rangle & \mapsto & |\underline{\mathbf{2n{+}1}}\rangle, & \mathbf{n} \leq 3 \\ |\underline{\mathbf{2n{+}1}}\rangle & \mapsto & |\underline{\mathbf{2n{+}2}}\rangle, & \mathbf{n} \leq 3 \\ |\underline{\mathbf{9}}\rangle & \mapsto & |\underline{\mathbf{0}}\rangle & \\ |\underline{\mathbf{y{+}10}}\rangle & \mapsto & |\underline{\mathbf{y{+}10}}\rangle & \end{array} \right\}$$

$$\mathrm{CNOT}_{\mathbf{qnat}} : U(\mathbf{qbit} \otimes \mathbf{qnat}, \mathbf{qbit} \otimes \mathbf{qnat})$$
$$\mathrm{CNOT}_{\mathbf{qnat}} := \mathbf{qif}\ \boldsymbol{x}\ \mathbf{then}\ \boldsymbol{x} \otimes u_1\ \mathbf{else}\ \boldsymbol{x} \otimes u_2$$

We define the unitary S corresponding to one step of the walk as $S := \mathrm{CNOT}_{\mathbf{qnat}} \circ (\mathrm{Had} \otimes \mathrm{Id})$. Finally, the k-step walk can be represented in our language as follows, where S^k represents k compositions of S with itself:

$$\mathrm{walk} : (\mathcal{B}(\mathbf{qbit} \otimes \mathbf{qnat}) \multimap \mathit{qbit} \otimes \mathrm{Nat})$$
$$\mathrm{walk} := \lambda z.\mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = \mathcal{B}(S^k)(z)\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2)$$

We now exhibit how the lambda expression for the quantum walk would reduce in our semantics when evaluated at the quantum state $|\mathbf{0}\rangle \otimes |\underline{\mathbf{0}}\rangle$ for $k = 1$. For the rest of this example, let $M := \mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = \mathcal{B}(S)(z)\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2)$ and $\boldsymbol{v} := |\mathbf{0}\rangle \otimes |\underline{\mathbf{0}}\rangle$. The following reductions can be derived:

$$(*, u_{id}, \mathit{pure}(\boldsymbol{v})) \rightsquigarrow_1 (* \otimes \boldsymbol{v}, u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), x)$$
$$(*, u_{id}, (\lambda z.M)\ \mathit{pure}(\boldsymbol{v})) \rightsquigarrow_1 (* \otimes \boldsymbol{v}, u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), M[x/z])$$

It also holds that

$$(* \otimes \boldsymbol{v}, u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), \mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = \mathcal{B}(S)(x)\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2))$$
$$\rightsquigarrow_1 (u(* \otimes \boldsymbol{v}), u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), \mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = x\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2))$$

$$\dfrac{\dfrac{(*, u_{id}, \mathit{pure}(\boldsymbol{v})) \rightsquigarrow_1 (* \otimes \boldsymbol{v}, u_{\sigma_{swap} \circ (u_{id} \otimes u_{id})}, x)}{(*, u_{id}, (\lambda z.M)\mathit{pure}(\boldsymbol{v})) \rightsquigarrow_1 (* \otimes \boldsymbol{v}, u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), M[x/z])}}{(* \otimes \boldsymbol{v}, u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), \mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = \mathcal{B}(S)(x)\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2)) \rightsquigarrow_1 (u(* \otimes \boldsymbol{v}), u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), \mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = x\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2))}$$

The unitary u in the above derivation is given by $u := (u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}))^* \circ (S \otimes id) \circ (u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}))$. The quantum term $u(* \otimes \boldsymbol{v})$ rewrites to $\boldsymbol{v}' := {}^1\!/\!_{\sqrt{2}} \cdot (* \otimes |\mathbf{0}\rangle \otimes |\underline{\mathbf{1}}\rangle + * \otimes |\mathbf{1}\rangle \otimes |\underline{\mathbf{9}}\rangle)$. We define $\mathit{nine} := \mathit{succ}^9(\mathit{zero})$. Thus, the following,

$$\begin{aligned} &(\boldsymbol{v}', u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id}), \mathit{let}\ \mathcal{B}(x_1 \otimes x_2) = x\ \mathit{in}\ x_1 \otimes \mathit{meas}(x_2)) \\ &\rightsquigarrow_1 (\boldsymbol{v}', u_{\sigma_{divide}} \circ (u_{\sigma_{swap}} \circ (u_{id} \otimes u_{id})), x_1 \otimes \mathit{meas}(x_2)) \\ &\begin{cases} \rightsquigarrow_{1/2} (* \otimes |\mathbf{0}\rangle, u_{\sigma_{swap}}, x_1 \otimes \mathit{succ}(\mathit{zero})) \\ \rightsquigarrow_{1/2} (* \otimes |\mathbf{1}\rangle, u_{\sigma_{swap}}, x_1 \otimes \mathit{nine}) \end{cases} \end{aligned}$$

reductions hold.

5.6 Related Work

Here, we discuss some of the relevant work from the literature on quantum programming languages, before proceeding to the technical material of the work. We begin by commenting on semantic approaches to classical control. In [PSV14], the authors describe a QLC where the higher-order primitives are interpreted using techniques from the semantics of quantitative models of linear logic. In [CDW19, CD19, D20], the author(s) work with QLCs whose denotational semantics is given via game semantics. More recently, in [TA24], the authors have shown how to interpret a QLC with recursive types using presheaves that are enriched over a suitable base category and they also prove full abstraction. The approach that we use in this paper is based on von Neumann algebras which have been previously used for the semantics of quantum programming languages [CW16, JKLMZ22, PPRZ20]. In particular, if we disregard the interaction between the pure subsystem and the main calculus, then our denotational model for the main calculus coincides with the one in [CW16]. The reason that we choose von Neumann algebras over the other approaches is because we think this gives the model which is the closest to mathematical physics out of the ones mentioned. Furthermore, our primary focus is not on developing the semantics of the classical paradigm; instead it is on the *interaction* between the pure quantum control subsystem and the main calculus based on classical control. Another distinguishing feature of our system, compared to the ones above, is the addition of types such as $\mathcal{B}(\mathbf{qnat})$ which allow us to describe states in infinite-dimensional Hilbert spaces, such as $\ell^2(\mathbb{N})$, and which we do not think can be easily included using the other approaches.

For the quantum control paradigm, relevant work includes [DGMV19, DM22, M22, DCM22, AD17]. What all these approaches have in common is that they have specific terms in their syntax for representing superposition. However, these approaches do not ensure unitarity of the relevant function spaces, in general. Our approach, instead, is based on related work first described in [SVV18]. One of the most distinctive features of this approach to quantum control is that the terms that introduce superposition are subject to strict formation conditions that ensure (linear algebraic) normalisation of the corresponding vectors. Furthermore, this approach to quantum control also imposes strict admissibility criteria on the unitary function spaces that

* These languages admit general recursion.

can be formed through the type system and this ensures unitarity of the relevant expressions. Because of these considerations, we choose this approach to model quantum control in our pure subsystem. However, we have made some changes, in particular, we replace the operational semantics from these works with an equational theory instead. This works better for our approach to *combining* quantum and classical control within the quantum configurations (in QLCs the quantum data is considered modulo equality). We continue building on this development by describing a denotational semantics that is sound and complete with respect to the equational theory. This is the first such result for languages with quantum control.

Another paper which combines quantum and classical control is Qunity [VLRH23]. However, compared to our work, Qunity does not have an operational semantics, but instead there is a compilation procedure to quantum circuits. Furthermore, the denotational semantics of Qunity: (1) does not ensure unitarity and normalisation of the pure primitives in its language, whereas ours does; (2) does not ensure trace-preservation of the mixed primitives in the Schrödinger picture, whereas ours does indeed ensure unitality of the mixed primitives in the Heisenberg picture (which corresponds to trace-preservation under the Heisenberg-Schrödinger duality). Other differences include that Qunity is restricted to finite-dimensional quantum data and they do not have higher-order lambda abstractions for dealing with mixed-state primitives. Instead, Qunity relies on a try-catch mechanism to combine quantum and classical control, whereas our approach is based on quantum configurations.

5.7 Conclusion

We described a programming language which has support for both pure state quantum computation and mixed state quantum computation. We began by recalling the pure quantum subsystem (§5.1), for which there is an equational theory which is complete w.r.t. its denotational semantics. Then, we described the main calculus (§5.2.1) which uses a new adaptation of quantum configurations (§5.3), which we used to define the operational semantics of the language. Here, the interaction between the pure quantum subsystem and the main calculus becomes apparent. We showed that our denotational semantics (§5.4) has a clear and appropriate interpretation as channels in the Heisenberg picture of quantum mechanics and we showed it is sound and adequate with respect to the operational semantics (§5.4).

Although our language has support for Nat, an inductive type, an obvious extension would be to include more general inductive types. Additionally, support for higher-order *pure* quantum computation is another feature that would be interesting to add in a future work.

Another matter that could be tackled as future work is recursion. In the pure system, adding general recursion is an open and difficult problem [L24, Section 5.2]. In the main calculus, our denotational model supports recursion for first-order quantum functions [PPRZ20, JKLMZ22]. However, our model does not support general recursion when quantum lambda abstractions are involved. There are other denotational models that support general recursion and quantum lambda abstractions, but they do not support types like qnat that correspond to $\ell^2(\mathbb{N})$, instead they support recursive types that intuitively correspond to infinite classical combinations of spaces involving finite-dimensional Hilbert spaces, which is not the same as $\ell^2(\mathbb{N})$. A model for quantum lambda abstractions, qnat and general recursion is an open problem to the best of our knowledge.

6

A linear algebraic λ-calculus combining quantum and classical control

6.1 Introduction

In parallel with the development of quantum programming languages, a complementary line of research seeks to understand the logical structure underlying quantum computation. Just as intuitionistic logic, λ-calculus, and Cartesian closed categories are linked via the Curry–Howard–Lambek correspondence [H80, L80, SU06], one may ask whether there exists a natural logical system whose proof language corresponds to quantum computation. This perspective motivates the study of *quantum computational logic*, where proofs denote quantum processes and the type system reflects the constraints and structure of quantum theory.

From this viewpoint, the challenge is not to build yet another quantum programming language, but to construct a logical framework with a well-defined proof theory that aligns with the operational and denotational semantics of quantum computation. Such a framework should ideally support superposition and linear combinations natively, provide a proof-theoretic foundation for unitary transformations and entangled systems, and admit a natural account of measurement and mixed states.

This chapter presents a proof language for Intuitionistic Multiplicative Additive Linear Logic (IMALL), extended with a modality $\mathcal{B}$ that captures the transition from pure to mixed states. Our system is based on quantum control and supports both pure and mixed-state computation within a unified logical setting, with a compositional and adequate semantics.

Our main contributions are as follows:

- We define the logic $\mathcal{B}$-IMALL by extending IMALL with a modality $\mathcal{B}$ in a *transparent* way: every provable formula in $\mathcal{B}$-IMALL remains provable in IMALL when occurrences of $\mathcal{B}$ are erased.

- The proof language supports linear combinations of terms through the operators $\cdot$ and $+$, which preserve the structure of a complex vector space (Corollary 1). Our system is thus *algebraically linear* in a precise sense.

- The cut-elimination process is defined via a composite reduction relation, obtained by interleaving a linear-algebraic equivalence ($\leftrightarrows$) with a deterministic evaluation ($\rightarrow$), as formalised in §6.2. This approach combines ideas from other frameworks: the relation $\rightarrow$ captures the interaction between algebraic constructors and logical connectives, as in

the $\odot$ and $\mathcal{L}\odot$ calculi [DD24, DCD23], and is here extended to handle the $\mathcal{B}$ modality. The symmetric relation $\leftrightarrows$ ensures canonical vector forms, in the style of algebraic λ-calculi [AD17, V09, ADPTV14].

- Measurement is expressible as a program, not as a constant: rather than requiring a dedicated syntactic construct with external semantics, it emerges naturally as a proof term within the language.
- The language avoids the need for quantum lambda configurations (QLCs) [SV06, DLPZ25, CDW19, TA24], by encoding both program and quantum state uniformly as terms.
- We provide a denotational model inspired by [CHK16]. Our model defines a functor $\mathcal{B}$ mapping pure-state morphisms to completely positive maps. A similar functor also denoted with the same symbol first appeared in [DLPZ25]. This model validates the design of the language and supports an adequacy theorem.
- Finally, we show that every linear transformation over $\mathbb{C}^{2^n}$ can be encoded in the pure fragment of our language (§6.5), and demonstrate the expressiveness of the language through examples such as quantum teleportation and the quantum switch—highlighting its ability to naturally capture paradigmatic quantum phenomena.

Taken together, these results yield a novel proof language that can express both pure and mixed-state quantum computation, grounded in a transparent logical system with a compositional and adequate semantics.

Compared to previous systems, our approach treats quantum computation not merely as a programming discipline but as a logical theory with intrinsic proof-theoretic content. Rather than relying on external control structures or dedicated constants, we encode quantum data, operations, and measurements uniformly as logical terms. This design bridges foundational insights from logic with the practical demands of quantum computation.

6.2 Syntax and Derivation rules

The calculus $\mathcal{L}_{\mathcal{B}}$ is an extension of the linear lambda calculus, viewed as a proof system for an extension of IMALL. This extension consists of adding a new modality, $\mathcal{B}$, to the language, which enables the embedding of pure-state quantum computations into the mixed-state fragment of the calculus. As a result, the language is divided into two fragments: one for pure-state quantum computation and another for mixed-state quantum computation. Accordingly, the syntax for the logic $\mathcal{B}$-IMALL is given in Figure 6.1.

We denote pure types by bold capital Latin letters, and mixed types by regular capital Latin letters.

Notice that, by removing the modality $\mathcal{B}$ and ignoring the distinction between bold and non-bold connectives, the two fragments collapse into IMALL. For example, $\top \oplus \mathbf{1}$ is not a valid proposition in $\mathcal{B}$-IMALL, whereas $\mathcal{B}(\top) \oplus 1$ is. Removing the modality from the latter, and identifying $\mathbf{1}$ with 1, yields the syntactically valid IMALL proposition $\top \oplus 1$.

The proposition $\mathcal{B}(\boldsymbol{P})$ represents a mixed-state quantum computation on the Hilbert space represented by the proposition $\boldsymbol{P}$. The notation $\mathcal{B}(\cdot)$ is inspired by our denotational model, where $\mathcal{B}(H)$ denotes the C^*-algebra of linear operators on a finite-dimensional Hilbert space H.

The proof terms of the calculus are also divided into two fragments: one for pure-state quantum computation, corresponding to pure types, and another for mixed-state quantum computation, corresponding to mixed ones. Pure and mixed variables–denoted by $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ and x, y, z,

(Vars)	$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, x, y, z$		
(Pure Types)	$\boldsymbol{P}$	::=	$\top \mid \mathbf{1} \mid \boldsymbol{P} \multimap \boldsymbol{P} \mid \boldsymbol{P} \,\&\, \boldsymbol{P} \mid \boldsymbol{P} \otimes \boldsymbol{P}$
(Mixed Types)	A	::=	$0 \mid 1 \mid A \multimap A \mid A \otimes A \mid A \oplus A \mid \mathcal{B}(\boldsymbol{P})$
(Pure Terms)	$\boldsymbol{m}$	::=	$\boldsymbol{x} \mid \boldsymbol{m} + \boldsymbol{m} \mid \boldsymbol{a} \cdot \boldsymbol{m} \mid \langle\rangle \mid * \mid \boldsymbol{\delta_1(m,m)} \mid \boldsymbol{\lambda x.m} \mid \boldsymbol{m\ m}$ $\mid \boldsymbol{\langle m,m\rangle} \mid \boldsymbol{\delta^1_{\&}(m,x.m)} \mid \boldsymbol{\delta^2_{\&}(m,x.m)} \mid \boldsymbol{m \otimes m} \mid \boldsymbol{\delta_\otimes(m,xy.m)}$
(Mixed Terms)	m	::=	$x \mid m + m \mid p \cdot m \mid \delta_0(m) \mid * \mid \delta_1(m,m) \mid \lambda x.m \mid m\ m$ $\mid m \otimes m \mid \delta_\otimes(m, xy.m) \mid inl(m) \mid inr(m)$ $\mid \delta_\oplus(m, x.m, y.m) \mid \mathcal{B}(\boldsymbol{t}) \mid \tau(m)$
(Pure Values)	$\boldsymbol{v_b}$	::=	$* \mid \langle\rangle \mid \boldsymbol{x} \mid \boldsymbol{v_n} \otimes \boldsymbol{v_n}$
	$\boldsymbol{v_n}$	::=	$\boldsymbol{\lambda x.t} \mid \boldsymbol{\langle v,v\rangle} \mid \boldsymbol{v_b}$
	$\boldsymbol{v}$	::=	$\boldsymbol{v_n} \mid \sum_{i=1}^n \boldsymbol{a_i \cdot v_{bi}} + \sum_{j=1}^l \boldsymbol{v_{bj}} \mid \sum_{i=1}^n \boldsymbol{0 \cdot v_{bi}}$
(Mixed Values)	v_b	::=	$* \mid x \mid inl(v_n) \mid inr(v_n) \mid v_n \otimes v_n \mid \mathcal{L}(u)$
	v_n	::=	$\lambda x.m \mid v_b$
	v	::=	$v_n \mid \sum_{i=1}^n p_i \cdot v_{bi} + \sum_{j=1}^l v_{bj} \mid \sum_{i=1}^n 0 \cdot v_{bi}$

Figure 6.1: Syntax.

respectively–belong to an enumerable set Vars. The α-equivalence relation and the free variables of a term are defined as usual. Terms are defined modulo α-equivalence. Closed terms are the ones that do not contain free variables.

Most terms in $\mathcal{L}_\mathcal{B}$ are standard for the linear lambda calculus. As before, to highlight the distinction between pure and mixed terms, pure terms are represented by bold letters $\boldsymbol{m}, \boldsymbol{n}, \boldsymbol{o}$, while mixed terms are represented by the letters m, n, o. Each construction corresponds to a rule of the logic.

Starting with the pure fragment, the symbols $+$ and $\cdot$ denote sums and scalar multiplication, which correspond to the following rules introduced in [DD24].

$$\frac{\boldsymbol{\Gamma} \vdash \boldsymbol{P} \quad \boldsymbol{\Gamma} \vdash \boldsymbol{P}}{\boldsymbol{\Gamma} \vdash \boldsymbol{P}} \qquad \frac{\boldsymbol{\Gamma} \vdash \boldsymbol{P}}{\boldsymbol{\Gamma} \vdash \boldsymbol{P}}$$

Scalars in the pure fragment range over the field of complex numbers and are denoted by the letters $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$. We work with unnormalized vectors, since the techniques required to restrict the calculus to normalized vectors are similar to those used in [DGMV19], and would make the calculus even more complex.

The other constructions are standard:

- The connective $\top$ has only an introduction rule, as usual, which is $\langle\rangle$.
- The connective $\mathbf{1}$ has an introduction rule, given by $*$, and also an elimination rule, $\boldsymbol{\delta_1(m,n)}$, intended to be understood as sequencing. While not always necessary, it is useful in the presence of scalars.
- The connective $\multimap$ has the standard introduction and elimination rules: $\boldsymbol{\lambda x.m}$ and $\boldsymbol{m\ n}$, respectively.
- The connective $\&$ has an introduction rule given by $\boldsymbol{\langle m,n\rangle}$, and, since we present all rules in their extended version, it has the extended elimination rule $\boldsymbol{\delta^i_{\&}(m,x.n)}$ for $i \in \{1,2\}$.

- The connective $\otimes$ has an introduction rule given by $\boldsymbol{m \otimes n}$, and an elimination rule given by $\boldsymbol{\delta_{\otimes}(m, xy.n)}$.

Similarly, the symbols $+$ and $\cdot$ in the mixed fragment denote sums and scalar multiplication. However, scalars in the mixed fragment range over the non-negative real numbers and are denoted by the letters p, q. Since, we work with unnormalized vectors in the mixed state setting as well, valid linear combinations of terms can be understood to be a generalization of probability distributions.

The connectives 1, $\multimap$, and $\otimes$ are completely analogous to those in the pure fragment. The connectives 0 and $\oplus$ are standard: 0 has only an elimination rule, as usual, given by $\delta_0(m)$; $\oplus$ has two introduction rules, $inl(m)$ and $inr(n)$, and an elimination rule given by $\delta_{\oplus}(m, x.n, y.o)$.

Finally, there are two other constructions specific to the mixed fragment, which do not correspond to any logical connective: The modality $\mathcal{B}(\boldsymbol{m})$, which embeds pure terms into the mixed fragment. It has an introduction rule that allows the inclusion of closed pure terms. Thus, a pure term $\boldsymbol{m}$ can be embedded as $\mathcal{B}(\boldsymbol{m})$. The construction $\tau(m)$ is a *coercion* operator that enables switching from the proposition $\mathcal{B}(\boldsymbol{P}) \otimes \mathcal{B}(\boldsymbol{Q})$ to the proposition $\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q})$. This operator is crucial for the interaction between the pure and mixed fragments.

The pure and mixed values in this calculus are as given in Figure 6.1. Here, in the summations, $\boldsymbol{v_{bi}, v_{bj}}$ (resp. v_{bi}, v_{bj}) are assumed to be distinct values, and $\boldsymbol{a_i \neq 0, 1}$ (resp. $p_i > 0, p_i \neq 1$). From now on, we will always assume that the summation symbol $\sum$ is used in the context of values, and that the summands are distinct.

Pure values $\boldsymbol{v}$ are composed of linear combinations of the values $\boldsymbol{v_b}$, and of values $\boldsymbol{v_n}$, which are not linear combinations. The values $\boldsymbol{v_n}$ correspond to the connectives $\top$, $\boldsymbol{\multimap}$, and $\boldsymbol{\&}$, for which linear combinations commute with the introduction rules and can be distributed internally. For example, the term $\langle \boldsymbol{m}_1, \boldsymbol{n}_1 \rangle + \langle \boldsymbol{m}_2, \boldsymbol{n}_2 \rangle$ can be rewritten as $\langle \boldsymbol{m}_1 + \boldsymbol{m}_2, \boldsymbol{n}_1 + \boldsymbol{n}_2 \rangle$. In contrast, the values $\boldsymbol{v_b}$ correspond to the connectives $\mathbf{1}$ and $\otimes$, where such commutation is not possible. For instance, the term $(\boldsymbol{m}_1 \otimes \boldsymbol{n}_1) + (\boldsymbol{m}_2 \otimes \boldsymbol{n}_2)$ cannot be rewritten by commuting $+$ with $\otimes$.

Similarly, mixed values v are composed of linear combinations of the values v_b, and of values v_n, which are not linear combinations. Henceforth, we may use the letters M, N, O to denote *both* mixed and pure terms, the letters U, V, W to denote both mixed and pure values, and S, T to denote both mixed and pure types. Additionally, we use the Greek letters α, β to denote scalars from either fragment. For simplicity, we may use the same symbols for connectives in both fragments when using capital letters. For example, we may write $\otimes$ to refer to either $\otimes$ or $\otimes$, depending on context. Similarly, we may write x instead of $\boldsymbol{x}$, $*$ instead of $\boldsymbol{*}$, and so on, when the distinction is clear.

Whenever we use metavariables like M, N, or S in place of concrete terms or types, we assume that all components belong to the same fragment. For instance, since $\boldsymbol{\delta_1(t}, r)$ is not a valid term—because $\boldsymbol{\delta_1}$ and $\boldsymbol{t}$ belong to the pure fragment and r to the mixed one—expressions like $\delta_{\mathbf{1}}(M, N)$ implicitly assume that M, N, and the connective itself all belong to the same fragment.

The logic $\mathcal{B}$-IMALL is defined by the syntax given above, together with a set of derivation rules, presented in Figure 6.2. The rules are formulated in natural deduction style, with proof terms annotating the derivations.

A context is a list of variable declarations, where each variable is associated with a type. A pure context is denoted by $\boldsymbol{\Gamma}$ or $\boldsymbol{\Delta}$, while a mixed context is denoted by Γ or Δ. As before, we simply write Γ or Δ when the distinction is clear or when referring to both fragments. The notation Γ, Δ denotes the concatenation of the two disjoint contexts Γ and Δ.

The names of the rules include a superscript ^{p} indicating it belongs to the pure fragment or a

$$\frac{}{x : S \vdash x : S}\ \text{ax} \qquad \frac{\Gamma \vdash M : S \quad \Gamma \vdash N : S}{\Gamma \vdash M + N : S}\ \text{sum} \qquad \frac{\Gamma \vdash M : S}{\Gamma \vdash \alpha \cdot M : S}\ \text{prod}(\alpha)$$

$$\frac{}{\boldsymbol{\Gamma \vdash \langle\rangle : \top}}\ \top_i^p \qquad \frac{\Gamma \vdash m : 0}{\Gamma, \Delta \vdash \delta_0(m) : A}\ 0_e^m \qquad \frac{}{\vdash * : 1}\ 1_i \qquad \frac{\Gamma \vdash M : 1 \quad \Delta \vdash N : S}{\Gamma, \Delta \vdash \delta_{\mathbf{1}}(M, N) : S}\ 1_e$$

$$\frac{\Gamma, x : S \vdash M : T}{\Gamma \vdash \lambda x.M : S \multimap T}\ \multimap_i \qquad \frac{\Gamma \vdash M : S \multimap T \quad \Delta \vdash N : S}{\Gamma, \Delta \vdash M\ N : T}\ \multimap_e$$

$$\frac{\boldsymbol{\Gamma \vdash m : P \quad \Gamma \vdash n : Q}}{\boldsymbol{\Gamma \vdash \langle m, n\rangle : P \,\&\, Q}}\ \&_i^p$$

$$\frac{\boldsymbol{\Gamma \vdash m : P \,\&\, Q \quad \Delta, x : P \vdash n : R}}{\boldsymbol{\Gamma, \Delta \vdash \delta^{\mathbf{1}}_{\&}(m, x.n) : R}}\ \&_{e_1}^p \qquad \frac{\boldsymbol{\Gamma \vdash m : P \,\&\, Q \quad \Delta, x : Q \vdash n : R}}{\boldsymbol{\Gamma, \Delta \vdash \delta^{\mathbf{2}}_{\&}(m, x.n) : R}}\ \&_{e_2}^p$$

$$\frac{\Delta \vdash m : A}{\Delta \vdash \mathit{inl}(m) : A \oplus B}\ \oplus_{i1}^m \qquad \frac{\Delta \vdash n : B}{\Delta \vdash \mathit{inr}(n) : A \oplus B}\ \oplus_{i2}^m$$

$$\frac{\Delta \vdash m : A \oplus B \quad \Gamma, x : A \vdash n : C \quad \Gamma, y : B \vdash o : C}{\Delta, \Gamma \vdash \delta_{\oplus}(m, x.n, y.o) : C}\ \oplus_e^m$$

$$\frac{\Delta \vdash M : S_1 \quad \Gamma \vdash N : S_2}{\Delta, \Gamma \vdash M \otimes N : S_1 \otimes S_2}\ \otimes_i \qquad \frac{\Delta \vdash M : S_1 \otimes S_2 \quad \Gamma, x : S_1, y : S_2 \vdash N : T}{\Delta, \Gamma \vdash \delta_{\otimes}(M, xy.N) : T}\ \otimes_e$$

$$\frac{\boldsymbol{\vdash m : P}}{\boldsymbol{\vdash \mathcal{B}(m) : \mathcal{B}(P)}}\ \mathcal{B}_i^m$$

$$\frac{\Gamma \vdash m : \mathcal{B}(\boldsymbol{P \multimap Q}) \quad \Delta \vdash n : \mathcal{B}(\boldsymbol{P})}{\Gamma, \Delta \vdash m\ n : \mathcal{B}(\boldsymbol{Q})}\ \mathcal{B}(\multimap)_e^m \qquad \frac{\Gamma \vdash m : \mathcal{B}(\boldsymbol{P}) \otimes \mathcal{B}(\boldsymbol{Q})}{\Gamma \vdash \tau(m) : \mathcal{B}(\boldsymbol{P \otimes Q})}\ \mathcal{B}(\otimes)^m$$

$$\frac{\Delta, y : S_1, x : S_2, \Gamma \vdash M : T}{\Delta, x : S_2, y : S_1, \Gamma \vdash M : T}\ \text{ex}$$

Figure 6.2: Derivation rules.

superscript ^{m} for the mixed fragment. The absence of a superscript indicates that the rule applies to both fragments. In such cases, we may optionally add the superscript later when specifying one of the fragments.

All the rules are standard, except for the rules sum and prod(α), which correspond to sums and scalar multiplication, respectively, and the rules related to the $\mathcal{B}$ modality. Notice that the rules sum and prod(α) do not change the provability associated with formulae.

The rules related to $\mathcal{B}$ are the following:

- The rule $\mathcal{B}_i^m$ allows the introduction of *closed* pure terms into the mixed fragment. There are two reasons for this being restricted to closed terms. First, the design and construction of proof terms for propositions $\mathcal{B}(\boldsymbol{P})$ are inspired by the denotational model, where $\mathcal{B}$ is interpreted as a functor between two categories. As such, it cannot be composed directly with morphisms from the domain category. Second, allowing open proof terms of propositions $\boldsymbol{P}$ to be promoted to proof terms of $\mathcal{B}(\boldsymbol{P})$ would permit arbitrary transformations from pure to mixed propositions to be expressed as valid terms in $\mathcal{L}_{\mathcal{B}}$. This goes against the intended design, which views the pure fragment as a subsystem of $\mathcal{L}_{\mathcal{B}}$ dedicated to pure-state quantum computation. In contrast, the mixed fragment is regarded as the "main" language of the calculus, with the pure fragment *included* into it to provide expressive power for

encoding pure-state quantum computation.

- The rule $\mathcal{B}(\multimap)^m_e$ allows the elimination of an embedded pure arrow.
- The rule $\mathcal{B}(\otimes)^m$ is a coercion rule. It allows one to transition from the proposition $\mathcal{B}(\boldsymbol{P}) \otimes \mathcal{B}(\boldsymbol{Q})$ to the less informative proposition $\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q})$—just as type-casting an integer to a real "forgets" its integer nature.

6.3 Operational Semantics

The operational reduciton rules for $\mathcal{L}_\mathcal{B}$ are defined using a relation $\cdot \hookrightarrow \cdot \subseteq (\textsc{Pure Terms} \sqcup \textsc{Mixed Terms}) \times (\textsc{Pure Terms} \sqcup \textsc{Mixed Terms})$. $\hookrightarrow$ (Definition 97), is formulated in terms of two auxiliary relations: an equivalence relation $\leftrightarrows$, capturing basic algebraic identities between proof terms, and a cut-reduction relation $\rightarrow$. Contexts are defined by the following grammar:

$$\begin{aligned} K := [.] \mid K \mathbin{+} M \mid \alpha \cdot K \mid \delta_0(K) \mid \delta_{\mathbf{1}}(K, M) \mid K\ M \mid V\ K \\ \mid \langle K, \boldsymbol{m} \rangle \mid \langle \boldsymbol{v}, K \rangle \mid \boldsymbol{\delta^1_\&}(K, \boldsymbol{x.m}) \mid \boldsymbol{\delta^2_\&}(K, \boldsymbol{x.t}) \\ \mid \mathit{inl}(K) \mid \mathit{inr}(K) \mid \delta_\oplus(K, x.m, y.m) \\ \mid K \otimes M \mid V \otimes K \mid \delta_\otimes(K, xy.M) \mid \mathcal{B}(K) \mid \tau(K) \end{aligned}$$

The equivalence relation $\leftrightarrows$ on terms is defined in Figure 6.3 (top part). The rules are divided into two groups: the left column corresponds to the rules usually seen in algebraic lambda calculi [ADPTV14, AD17, V09, DGMV19], while the right column corresponds to rules involving the $\otimes$ connective.

The relation $\leftrightarrows$ gives rise to the possibility of defining a canonical form for terms. To this end, we introduce the notion of *algebraic form* of a term, which is a linear combination of distinct *base forms* of terms. These concepts are defined by the following grammar. As before, the symbols correspond to both fragments unless otherwise specified. We recall that, as per our earlier convention, all summations are assumed to involve pairwise distinct summands.

$$\begin{aligned} &\mathfrak{b} := x \mid \langle\rangle \mid \delta_0(\mathfrak{a}) \mid * \mid \delta_{\mathbf{1}}(\mathfrak{a}, \mathfrak{a}) \mid \lambda x.\mathfrak{a} \mid \mathfrak{a}\ \mathfrak{a} \mid \langle \mathfrak{a}, \mathfrak{a} \rangle \mid \boldsymbol{\delta^i_\&}(\mathfrak{a}, \boldsymbol{x.\mathfrak{a}}) && \text{(Base Form)} \\ &\qquad \mid \mathit{inl}(\mathfrak{a}) \mid \mathit{inr}(\mathfrak{a}) \mid \delta_\oplus(\mathfrak{a}, x.\mathfrak{a}, y.\mathfrak{a}) \mid \mathfrak{b} \otimes \mathfrak{b} \mid \delta_\otimes(\mathfrak{a}, xy.\mathfrak{a}) \mid \mathcal{B}(\mathfrak{a}) \mid \tau(\mathfrak{a}) \\ &\mathfrak{a} := \textstyle\sum_{i=1}^n 0 \cdot \mathfrak{b}_i \mid \sum_{i=1}^n \alpha_i \cdot \mathfrak{b}_i\ (\alpha_i \neq 0) && \text{(Algebraic Form)} \end{aligned}$$

Note that the choice of a symmetric relation and the use of algebraic forms are made purely for convenience. The rules could be oriented, following [ADPTV14], so that canonical forms coincide with normal forms.

Lemma 17 (Uniqueness). *If $\Gamma \vdash M : T$, then there is a unique algebraic form $\mathfrak{a}$ such that $M \leftrightarrows \mathfrak{a}$. We denote the algebraic form of M by $\mathfrak{a}(M)$.*

Proof. Proof is by induction on M:

- M is $\boldsymbol{x}$ or x: In these cases $\mathfrak{a}(M) = 1 \cdot M$.
- M is $\langle\rangle$: Again in this case $\mathfrak{a}(M) = 1 \cdot M$.
- M is $*$ or $*$: In these cases, $\mathfrak{a}(M) = 1 \cdot M$.
- M is $\boldsymbol{\delta_1(m, n)}$: By induction hypothesis $\boldsymbol{m}, \boldsymbol{n}$ have unique algebraic forms $\mathfrak{a}(\boldsymbol{m}), \mathfrak{a}(\boldsymbol{n})$. We have, $\mathfrak{a}(M) = 1 \cdot \boldsymbol{\delta_1(\mathfrak{a}(m), \mathfrak{a}(n))}$.

- M is $\boldsymbol{\lambda x.m}$: By induction hypothesis, $\boldsymbol{m}$ has a algebraic form $\mathfrak{a}(\boldsymbol{m})$. We define, $\mathfrak{a}(M) = 1 \cdot \boldsymbol{\lambda x.}\mathfrak{a}(m)$. Since $\mathfrak{a}(\boldsymbol{m})$ is unique, so is $\mathfrak{a}(M)$.
- M is $\boldsymbol{m\ n}$: We have, $\mathfrak{a}(\boldsymbol{m\ n}) = \mathbf{1} \cdot \mathfrak{a}(\boldsymbol{m})\ \mathfrak{a}(\boldsymbol{n})$.
- M is $\boldsymbol{m \otimes n}$: By the induction hypothesis, we have $\mathfrak{a}(\boldsymbol{m}) = \sum_{i=1}^{m} \boldsymbol{a_i} \cdot \mathfrak{b}^i$, $\mathfrak{a}(\boldsymbol{n}) = \sum_{j=1}^{n} \boldsymbol{a_i} \cdot \mathfrak{b}^j$. Hence from transitivity and contextual rules it follows that $M \leftrightarrows (\sum_{i=1}^{m} \boldsymbol{a_i} \cdot \mathfrak{b}^i) \otimes (\sum_{j=1}^{n} \boldsymbol{a_i} \cdot \mathfrak{b}^j)$. Following the rules for rewriting under $\leftrightarrows$ in Figure 6.3, we have, $M \leftrightarrows \sum_{i=1}^{m} \sum_{j=1}^{n} \boldsymbol{a_i a_j} \cdot \boldsymbol{\mathfrak{b}^i} \otimes \boldsymbol{\mathfrak{b}^j}$. Now, renaming indices and rewriting if needed using the rule $\alpha \cdot M + \beta \cdot M \leftrightarrows (\alpha + \beta) \cdot M$, gives us the desired result.
- M is $\boldsymbol{\delta_{\otimes}(m, xy.n)}$: In this case we have $\mathfrak{a}(M) = \mathbf{1} \cdot \boldsymbol{\delta_{\otimes}(\mathfrak{a}(m), xy.}\mathfrak{a}(\boldsymbol{n}))$.
- M is $\langle \boldsymbol{m, n} \rangle$: We have, $\mathfrak{a}(M) = \mathbf{1} \cdot \langle \mathfrak{a}(\boldsymbol{m}), \mathfrak{a}(\boldsymbol{n}) \rangle$.
- M is $\boldsymbol{\delta^i_{\&}(m, x.n)}$: Follows using the same argument as the cases above.
- M is $\boldsymbol{m + n}$: By induction hypothesis we have $\mathfrak{a}(\boldsymbol{m}) = \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \mathfrak{b}^i$ and $\mathfrak{a}(\boldsymbol{n}) = \sum_{j=1}^{k} \boldsymbol{a_j} \cdot \mathfrak{b}^j$. Hence, $M \leftrightarrows \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \mathfrak{b}^i + \sum_{j=1}^{k} \boldsymbol{a_j} \cdot \mathfrak{b}^j$. Now, rewriting using commutativity and the rule $\alpha \cdot M + \beta \cdot M \leftrightarrows (\alpha + \beta) \cdot M$ gives us the result.
- M is $\boldsymbol{a \cdot m}$: We have $\mathfrak{a}(\boldsymbol{m}) = \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \mathfrak{b}^i$ by the induction hypothesis. Hence, $M \leftrightarrows \boldsymbol{a} \cdot \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \mathfrak{b}^i$. Now rewriting using distributivity, commutativity and the rule $\alpha \cdot (\beta \cdot M) \leftrightarrows \alpha\beta \cdot M$, gives us the result.
- M is $\mathcal{B}(\boldsymbol{m})$: In this case we have, $\mathfrak{a}(M) = 1 \cdot \mathcal{B}(\mathfrak{a}(\boldsymbol{m}))$.
- M is $\tau(m)$: In this case we have, $\mathfrak{a}(M) = 1 \cdot \tau(\mathfrak{a}(m))$.

The remaining cases for the mixed fragment follow using similar arguments as those above. □

The rewrite relation $\rightarrow$ is also defined in Figure 6.3. As usual, we write $(N/x)M$ for the substitution of N for x in M. In the rules involving summations, we assume—as per our earlier convention—that all summands are pairwise distinct. The rules are divided into three groups:

- Rules (6.1) to (6.7) correspond to cut-elimination steps: an introduction rule followed by an elimination rule for each connective. The only non-standard case is the rule for 1 (and $\mathbf{1}$), which carries a scalar α in the elimination.
- Rules (6.8) to (6.12), consist of commutation rules that allow the proof constructors $+$ and $\cdot$ to commute with each connective. In some cases, commutation applies to introduction rules (for $\top$, $\multimap$, and $\&$); in others, to elimination rules (for $\oplus$ and $\otimes$). There is no commutation rule for $\overline{o}$, and no need for one for 1, which is already handled by Rule (6.1). The last two rules of this group, Rules (6.13) and (6.14), are not strictly necessary, as they are subsumed by Rule (6.11). However, they are useful for obtaining better normal forms. For instance, the term $inl(p \cdot *) + inl(q \cdot *)$ reduces, using $\rightarrow$ and $\leftrightarrows$, to $(p + q) \cdot inl(*)$.
- Rule (6.15) is a coercion rule to transition from proposition $\mathcal{B}(\boldsymbol{P}) \otimes \mathcal{B}(\boldsymbol{Q})$ to proposition $\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q})$. Rule (6.16) is a commutation rule for the coercion operator τ. Rule (6.17) allows a scalar to commute with the $\mathcal{B}$ operator. Note that $\mathcal{B}$ is not a linear operator. Intuitively, if $\boldsymbol{v_b}$ represents a quantum state in vector form, then $\mathcal{B}(\boldsymbol{v_b})$ represents the same state in its density matrix form. As a consequence, the scalar $\boldsymbol{a}$ is multiplied by $\overline{\boldsymbol{a}}$ when commuting with $\mathcal{B}$. Finally, Rule (6.18) distributes applications over the $\mathcal{B}$ operator.

$$1 \cdot M \leftrightarrows M$$
$$M_1 + M_2 \leftrightarrows M_2 + M_1$$
$$M_1 + (M_2 + M_3) \leftrightarrows (M_1 + M_2) + M_3$$
$$\alpha \cdot M + \beta \cdot M \leftrightarrows (\alpha + \beta) \cdot M$$
$$\alpha \cdot (\beta \cdot M) \leftrightarrows \alpha\beta \cdot M$$
$$\alpha \cdot (M_1 + M_2) \leftrightarrows \alpha \cdot M_1 + \alpha \cdot M_2$$
$$\text{If } M \neq 0 \cdot M',\ M + 0 \cdot N \leftrightarrows M$$

$$M_1 \otimes (\alpha \cdot M_2) \leftrightarrows \alpha \cdot M_1 \otimes M_2$$
$$(\alpha \cdot M_1) \otimes M_2 \leftrightarrows \alpha \cdot M_1 \otimes M_2$$
$$(M_1 + M_2) \otimes N \leftrightarrows M_1 \otimes N + M_2 \otimes N$$
$$M \otimes (N_1 + N_2) \leftrightarrows M \otimes N_1 + M \otimes N_2$$

$$\delta_{\mathbf{1}}(\alpha \cdot *, M) \to \alpha \cdot M \tag{6.1}$$
$$(\lambda x.M)\ V \to (V/x)M \tag{6.2}$$
$$\boldsymbol{\delta^1_{\&}(\langle v, w\rangle, x.t) \to (v/x)t} \tag{6.3}$$
$$\boldsymbol{\delta^2_{\&}(\langle v, w\rangle, y.t) \to (w/y)t} \tag{6.4}$$
$$\delta_{\oplus}(inl(v), x.m, y.n) \to (v/x)m \tag{6.5}$$
$$\delta_{\oplus}(inr(v), x.m, y.n) \to (v/y)n \tag{6.6}$$
$$\delta_{\otimes}(V_n \otimes W_n, xy.M) \to (V_n/x, W_n/y)M \tag{6.7}$$

$$\boldsymbol{a \cdot \langle\rangle \to \langle\rangle} \tag{6.8}$$
$$\textstyle\sum_{i=1}^n \alpha_i \cdot \lambda x.M_i \to \lambda x.(\sum_{i=1}^n \alpha_i \cdot M_i) \tag{6.9}$$
$$\textstyle\sum_{i=1}^n \boldsymbol{a_i \cdot \langle v_i, w_i\rangle} \to \boldsymbol{\langle \sum_{i=1}^n a_i \cdot v_i, \sum_{i=1}^n a_i \cdot w_i\rangle} \tag{6.10}$$
$$\textstyle\delta_{\oplus}(\sum_{i=1}^n p_i \cdot v_{bi}, x.o, y.s) \to \sum_{i=1}^n p_i \cdot \delta_{\oplus}(v_{bi}, x.o, y.s) \tag{6.11}$$
$$\textstyle\delta_{\otimes}(\sum_{i=1}^n \alpha_i \cdot V_{bi}, xy.M) \to \sum_{i=1}^n \alpha_i \cdot \delta_{\otimes}(V_{bi}, xy.M) \tag{6.12}$$
$$\textstyle inl(\sum_{i=1}^n p_i \cdot v_{bi}) \to \sum_{i=1}^n p_i \cdot inl(v_{bi}) \tag{6.13}$$
$$\textstyle inr(\sum_{i=1}^n p_i \cdot v_{bi}) \to \sum_{i=1}^n p_i \cdot inr(v_{bi}) \tag{6.14}$$

$$\boldsymbol{\tau(\mathcal{B}(v) \otimes \mathcal{B}(w)) \to \mathcal{B}(v \otimes w)} \tag{6.15}$$
$$\textstyle\tau(\sum_{i=1}^n p_i \cdot v_{bi}) \to \sum_{i=1}^n p_i \cdot \tau(v_{bi}) \tag{6.16}$$
$$\boldsymbol{\text{If } a \neq 1,\ \mathcal{B}(a \cdot v_b) \to |a|^2 \cdot \mathcal{B}(v_b)} \tag{6.17}$$
$$\textstyle(\sum_{i=1}^n p_i \cdot \mathcal{B}(\boldsymbol{v_i}))\ (\sum_{j=1}^k q_j \cdot \mathcal{B}(\boldsymbol{w_j})) \to \sum_{i,j} p_i q_j \cdot \mathcal{B}(\boldsymbol{v_i\ w_j}) \tag{6.18}$$

$$\frac{M \leftrightarrows N}{K[M] \leftrightarrows K[N]} \qquad \frac{M \to N}{K[M] \to K[N]}$$

Figure 6.3: Reduction relations $\leftrightarrows$ and $\to$.

Definition 97 ($\hookrightarrow$). *The relation $\hookrightarrow$ is defined as:*

$$\frac{M \to N}{M \hookrightarrow N} \qquad \frac{M \leftrightarrows V}{M \hookrightarrow V} \qquad \frac{M \not\to \quad M \leftrightarrows \mathfrak{a}(M) \quad \mathfrak{a}(M) \to N}{M \hookrightarrow N}$$

We denote by $\equiv$ the reflexive, transitive, and symmetric closure of $\hookrightarrow$.

Intuitively, $\hookrightarrow$ interleaves $\leftrightarrows$ and $\to$: it rewrites any term via $\leftrightarrows$ until it either reduces under $\to$ or reaches normal form. For instance, the term $\delta_{\oplus}(inl(*) + inl(*), x.m, y.n)$ does not reduce under

$\to$ alone (because $inl(*) + inl(*)$ is not a value), but rewriting it via $\leftrightarrows$ to $\delta_{\oplus}(2 \cdot inl(*), x.m, y.n)$ makes it reducible. Each step of $\leftrightarrows$ is a purely algebraic manipulation. In the following section, we show that $\hookrightarrow$ satisfies the standard properties of a well-behaved reduction system.

6.3.1 Correctness

We now state and prove the standard correctness theorems. First we show that the relation $\hookrightarrow$ never gets stuck.

Theorem 19 (Progress). *$\vdash M : T$ implies $T \neq \overline{o}$, and either M is a value V, or there exists N such that $M \hookrightarrow N$.*

Proof. The $\leftrightarrows$ is only used to rewrite terms when $\to$ gets stuck, if the reduct is not in its algebraic form. We prove progress by induction on M. M cannot be a variable since it is closed, and if If M is a value, we are done. Otherwise, we have the following cases:

- M is $\boldsymbol{m_1 + m_2}$: By induction hypothesis, either $\boldsymbol{m_1}, \boldsymbol{m_2}$ reduce further or they are values. If they reduce further then so does M by the contextual reduction rules. If they are values we do a case analysis based on the proposition $\boldsymbol{P}$ of M. If $\boldsymbol{P = P_1 \,\&\, P_2}, \boldsymbol{P_1 \multimap P_2}$, then $\boldsymbol{m_1}, \boldsymbol{m_2}$ are $\boldsymbol{\langle u_1, u_2 \rangle}, \boldsymbol{\langle v_1, v_2 \rangle}$ or $\boldsymbol{\lambda x.n_1}, \boldsymbol{\lambda x.n_2}$ respectively. In these cases M reduces further under $\to$ after a rewrite of M to $\mathfrak{a}(M)$. Now, if $\boldsymbol{P = 1}, \boldsymbol{P_1 \otimes P_2}$, then they are of the form $*, \boldsymbol{a \cdot *}$ or $\sum_{i=1}^{n} \boldsymbol{a_i \cdot v_b^i}$ as in the syntax. In all these cases, M does not reduce under $\to$ and hence a rewrite of M under $\leftrightarrows$ gives us the desired result. Hence, in all possible cases M reduces under $\hookrightarrow$.

- M is $\boldsymbol{a \cdot n}$: Again by induction hypothesis, $\boldsymbol{n}$ reduces further until it is a value, in which case so does M. Now we assume $\boldsymbol{n}$ is a value. We do a case analysis based on the proposition $\boldsymbol{P}$ of M. If $\boldsymbol{P = P_1 \,\&\, P_2}, \boldsymbol{P_1 \multimap P_2}$, then $\boldsymbol{n}$ is $\boldsymbol{\langle v_1, v_2 \rangle}$ or $\boldsymbol{\lambda x.n'}$ respectively. In these cases M reduces further under $\to$ to take the form $\boldsymbol{v}$ defined in the syntax. Now, if $\boldsymbol{P = 1}, \boldsymbol{P_1 \otimes P_2}$, then it is of the form $*, \boldsymbol{a \cdot *}$ or $\sum_{i=1}^{n} \boldsymbol{a_i \cdot v_b^i}$ as in the syntax. In both these cases, M does not reduce under $\to$ and hence a rewrite of M under $\leftrightarrows$ gives us the desired result. Hence, in all possible cases M reduces under $\hookrightarrow$.

- M is $\boldsymbol{\delta_1(m_1, m_2)}$: By induction hypothesis, $\boldsymbol{m_1}$ reduces until it is a value, in which case so does M. Now, we assume that $\boldsymbol{m_1}$ is a value. It follows that $\boldsymbol{m_1}$ is $*$ or $\boldsymbol{a \cdot *}$. Hence, it follows that $\boldsymbol{m}$ reduces further either through a potential rewrite of $\boldsymbol{m_1}$ under $\leftrightarrows$ or by reduction under $\to$.

- M is $\boldsymbol{m_1\ m_2}$: Again by induction hypothesis both $\boldsymbol{m_1}, \boldsymbol{m_2}$ reduce until they are values, in which case so does M. Now, we assume both $\boldsymbol{m_1}, \boldsymbol{m_2}$ are values. Hence, by the rules given in Figure 6.2 it follows that $\boldsymbol{m_1}$ is $\boldsymbol{\lambda x.n}$. Therefore, M reduces under $\to$.

- M is $\boldsymbol{\langle m_1, m_2 \rangle}$: By induction hypothesis and contextual reduction rules it follows that either M is of the form $\boldsymbol{\langle v_1, v_2 \rangle}$ for values $\boldsymbol{v_1}, \boldsymbol{v_2}$ or it reduces further under $\to$.

- M is $\boldsymbol{\delta^1_{\&}(m_1, x.m_2)}$ or $\boldsymbol{\delta^2_{\&}(m_1, x.m_2)}$: By induction hypothesis, either $\boldsymbol{m_1}$ reduces or it is a value. If it reduces then so does M by the contextual reduction rules. If it is a value, then by the rules given in Figure 6.2 it follows that $\boldsymbol{m_1}$ is $\boldsymbol{\langle v_1, v_2 \rangle}$. Hence, M reduces further in both the cases.

- M is $\boldsymbol{m_1} \otimes \boldsymbol{m_2}$: By induction hypothesis and contextual reduction rules it follows that either M is of the form $\boldsymbol{v_1} \otimes \boldsymbol{v_2}$ for values $\boldsymbol{v_1}, \boldsymbol{v_2}$ or it reduces further under $\rightarrow$. In the first case a rewrite using Lemma 17 gives us a value of the form $\boldsymbol{v}$ from the syntax.

- M is $\boldsymbol{\delta_{\otimes}(m_1, xy.m_2)}$: By induction hypothesis, $\boldsymbol{m_1}$ reduces further until it is a value, in which case so does M. Now we assume $\boldsymbol{m_1}$ is a value. From the rules in Figure 6.2 it follows that $\boldsymbol{m_1}$ is of the form $\boldsymbol{v_n^1} \otimes \boldsymbol{v_n^2}$ or a linear combination of such values. Hence, following a potential rewrite under $\leftrightarrows$, M reduces in both the cases under $\rightarrow$.

- M is $m_1 + m_2$: Again by induction hypothesis, m_1, m_2 reduce further until they are values, in which case so does M. Hence we assume that both m_1, m_2 are values. Now this case follows using an argument similar to the above case and rewriting using the rules for $\leftrightarrows$ defined in Figure 6.3.

- M is $p \cdot n$: By induction hypothesis, n reduces further until it is a value, in which case so does M. Now we assume n is a value, then n takes one of the forms of v from the syntax. In all the cases, depending on the form the value takes, M reduces further under $\rightarrow$ through a rewrite under $\leftrightarrows$ (when M is of proposition $A \multimap B$) or it is itself equal a value.

- M is $\delta_0(n)$: By induction hypothesis, n reduces until it is a value. Hence, so does M by the contextual rules. Now, we assume n is a value. But by induction hypothesis, n can never be a value.

- M is $\delta_{\boldsymbol{1}}(m_1, m_2)$: By induction hypothesis, either m_1 reduces or it is a value. If it reduces then so does M by the contextual reduction rules. If it is a value, then by the rules given in Figure 6.2 it follows that m_1 is either $*$, or $p \cdot *$. In all cases after a potential rewrite to $\mathfrak{a}(m_1)$, M reduces further.

- M is $m_1\ m_2$: Again by induction hypothesis m_1, m_2 reduce further until they are values, in which case so does M. Now, we assume both m_1, m_2 are values. We do a case analysis on the proposition of m_1. First we consider the case when m_1 is of the proposition $A \multimap B$. Then it follows that m_1 is $\lambda x.n$, In this M reduces further. Now we consider the case when m_1 is of proposition $\mathcal{B}(\boldsymbol{P_1} \multimap \boldsymbol{P_2})$. Then it follows that m_1 is $\mathcal{B}(\boldsymbol{m_1})$ or $\sum_{i=1}^{n} p_i \cdot \mathcal{B}(\boldsymbol{v_i})$. Similarly, m_2 is also of the form $\mathcal{B}(\boldsymbol{v})$, or $\sum_{i=1}^{n} p_i \cdot \mathcal{B}(\boldsymbol{u_i})$. This gives us 4 possible cases for M. All cases reduce using the rules in Figure 6.3 after potential rewrites under $\leftrightarrows$.

- M is $inl(n)$ or $inr(n)$: By induction hypothesis, n reduces further until it is a value, in which case so does M. Now we assume n is a value. If n is of the form v_b, then in both the cases M is a value. Otherwise, n can be rewritten to its algebraic form as a linear combination of v_b^is, so that M reduces further.

- M is $\delta_{\oplus}(m, x.n, y.o)$: By induction hypothesis m reduces further until it is a value, in which case so does M. Now we assume m is a value. Then by Lemma 17 it follows that m can be rewritten to the form $\sum_{i=1}^{n} p_i \cdot v_b^i$. Hence, M reduces further.

- M is $m_1 \otimes m_2$: Again by induction hypothesis, m_1, m_2 reduce further until they are values, in which case so does M. Now, we assume m_1, m_2 are values, in which case M is of the form $v_1 \otimes v_2$ for values v_1, v_2. A potential rewrite under $\leftrightarrows$ gives us a value of the form v from the syntax.

- M is $\delta_{\otimes}(m_1, x.m_2)$: By induction hypothesis m_1 reduces further until it is value, in which case so does M. Now we assume m_1 is a value. Then by Lemma 17 it follows that m_1 can be rewritten to the form $\sum_{i=1}^{n} p_i \cdot v_b^i$. Hence, M reduces further.
- M is $\mathcal{B}(\boldsymbol{m})$: Again by induction hypothesis, $\boldsymbol{m}$ reduces further until it is a value, in which case so does M.
- M is $\tau(n)$: By induction hypothesis, n reduces further until it is a value, in which case so does M. Now, we assume that n is a value. Then by Lemma 17 it follows that n can be rewritten to the form $\sum_{i=1}^{n} p_i \cdot v_b^i$. Hence, M reduces further.

□

In order to show that the relation $\hookrightarrow$ is preserves types, we first state and prove the customary *substitution lemma*.

Lemma 18 (Substitution lemma). *If* $\Gamma, x : T \vdash M : S$ *and* $\Delta \vdash N : T$*, then* $\Gamma, \Delta \vdash (N/x)M : S$. □

Proof. By induction on the derivation of M.

- $\boldsymbol{\Delta, x : P, y : Q, \Gamma \vdash m : R}$: This case follows directly from the induction hypothesis.
- $\boldsymbol{\Gamma \vdash \langle\rangle : \top}$: This case follows vacuously.
- $\boldsymbol{\Gamma, \Delta \vdash \delta_0(m') : P}$: We consider two cases. First case is when the variable being substituted is in $\boldsymbol{\Gamma}$. Then the claim follows by using the induction hypothesis on the term $\boldsymbol{m'}$ and the observation that $\boldsymbol{(n/x)\delta_0(m') = \delta_0((n/x)m')}$. In the other case, the claim follows since $\boldsymbol{(n/x)\delta_0(m') = \delta_0(m')}$.
- $\boldsymbol{x : P \vdash x : P}$: In this case, $\boldsymbol{\Gamma = \emptyset}$, $\boldsymbol{P = Q}$ and $\boldsymbol{(n/x)m = n}$ and the claim follows.
- $\boldsymbol{\vdash * : 1}$: This case follows vacuously.
- $\boldsymbol{\Gamma, \Delta \vdash \delta_1(m_1, m_2) : P}$: We consider two cases, one when the variable appears in $\boldsymbol{m_1}$ and the other when it appears in $\boldsymbol{m_2}$. Both follow by application of induction hypothesis on $\boldsymbol{m_1, m_2}$ respectively and the observation that in one case $\boldsymbol{(n/x)\delta_1(m_1, m_2) = \delta_1((n/x)m_1, m_2)}$ and in the other case $\boldsymbol{(n/x)\delta_1(m_1, m_2) = \delta_1(m_1, (n/x)m_2)}$.
- $\boldsymbol{\Gamma \vdash \lambda x.m' : P \multimap Q}$: This case follows using the induction hypothesis on $\boldsymbol{m'}$ and the observation that $\boldsymbol{(n/x)\lambda y.m' = \lambda y.(n/x)m'}$.
- $\boldsymbol{\Gamma, \Delta \vdash m_1\ m_2 : Q}$: Again we consider two cases, one when the variable appears in $\boldsymbol{m_1}$ and the other when the variable appears in $\boldsymbol{m_2}$. Now the claim follows by applying induction hypothesis on $\boldsymbol{m_1, m_2}$ in both the cases respectively and the observation that in one case $\boldsymbol{(n/x)(m_1\ m_2) = ((n/x)m_1)\ m_2}$ and in the other $\boldsymbol{(n/x)(m_1\ m_2) = m_1\ ((n/x)m_2)}$.
- $\boldsymbol{\Gamma \vdash \langle m_1, m_2\rangle : P \,\&\, Q}$: Follows by application of induction hypothesis on $\boldsymbol{m_1, m_2}$ respectively and the observation that $\boldsymbol{(n/x)\langle m_1, m_2\rangle = \langle (n/x)m_1, (n/x)m_2\rangle}$.
- $\boldsymbol{\Gamma, \Delta \vdash \delta^1_{\&}(m_1, x.m_2) : R}$ or $\boldsymbol{\Gamma, \Delta \vdash \delta^2_{\&}(m_1, x.m_2) : R}$: Again we consider two cases, one when the variable appears in $\boldsymbol{m_1}$ and the other when it appears in $\boldsymbol{m_2}$. The proof follows by applying the induction hypothesis on $\boldsymbol{m_1, m_2}$ respectively and the observation that $\boldsymbol{(n/x)\delta^i_{\&}(m_1, x.m_2)}$ is equal to $\boldsymbol{\delta^i_{\&}((n/x)m_1, x.m_2)}$ in one case and $\boldsymbol{(n/x)\delta^i_{\&}(m_1, x.m_2)}$ is equal to $\boldsymbol{\delta^i_{\&}(m_1, (n/x)x.m_2)}$ in the other.

- $\boldsymbol{\Delta, \Gamma \vdash m_1 \otimes m_2 : P \otimes Q}$: We consider two cases, one when the variable $\boldsymbol{x}$ appears in $\boldsymbol{m_1}$ and the other when $\boldsymbol{x}$ appears in $\boldsymbol{m_2}$ respectively. We apply induction hypothesis on the terms $\boldsymbol{m_1, m_2}$ in both the cases. The claim follows using the observation that $\boldsymbol{(n/x)(m_1 \otimes m_2) = (n/x)m_1 \otimes m_2}$ in one case and $\boldsymbol{(n/x)(m_1 \otimes m_2) = m_1 \otimes (n/x)m_2}$ in the other.

- $\boldsymbol{\Delta, \Gamma \vdash \delta_{\otimes}(m_1, xy.m_2) : R}$: We consider two cases, one when the variable appears in $\boldsymbol{m_1}$ and the other when it appears in $\boldsymbol{m_2}$. In both the cases we apply the induction hypothesis on the terms $\boldsymbol{m_1, m_2}$ respectively. Then again, the claim follows from the observation that $\boldsymbol{(n/x)\delta_{\otimes}(m_1, yz.m_2)}$ is equal to $\boldsymbol{\delta_{\otimes}((n/x)m_1, yz.m_2)}$ in one case and $\boldsymbol{(n/x)\delta_{\otimes}(m_1, yz.m_2)}$ is equal to $\boldsymbol{\delta_{\otimes}(m_1, (n/x)yz.m_2)}$ in the other.

- $\boldsymbol{\Gamma \vdash a \cdot m' : P}$: This case follows from the induction hypothesis applied to $\boldsymbol{m'}$ and the observation that $\boldsymbol{(n/x)(a \cdot m') = a \cdot (n/x)m'}$.

- $\boldsymbol{\Gamma \vdash m + n : P}$: This case follows from the induction hypothesis applied to $\boldsymbol{m_1, m_2}$ and the observation that $\boldsymbol{(n/x)(m_1 + m_2) = (n/x)m_1 + (n/x)m_2}$.

- $\Delta, x : B, y : A, \Gamma \vdash m : C$: This case follows directly from the induction hypothesis.

- $\vdash * : 1$: This case follows vacuously.

- $\Gamma, \Delta \vdash \delta_{\mathbf{1}}(m_1, m_2) : A$: We consider two cases, one when the variable appears in m_1 and the other when it appears in m_2. Both follow by application of induction hypothesis on m_1, m_2 respectively and the observation that in one case $(n/x)\delta_{\mathbf{1}}(m_1, m_2) = \delta_{\mathbf{1}}((n/x)m_1, m_2)$ and in the other case $(n/x)\delta_{\mathbf{1}}(m_1, m_2) = \delta_{\mathbf{1}}(m_1, (n/x)m_2)$.

- $\Delta \vdash \lambda x.m' : A \multimap B$: We consider two cases. One when the variable being substituted is x which is different from the one being abstracted as a lambda abstraction, and the other case when the variable being substituted is the one that is a part of the lambda abstraction. The latter case follows by the induction hypothesis applied on m'. The former case follows using the induction hypothesis on m' and the observation that $(n/x)\lambda y.m' = \lambda y.(n/x)m'$.

- $\Gamma, \Delta \vdash m_1\ m_2 : B$ or $\Gamma, \Delta \vdash m_1\ m_2 : \mathcal{B}(\boldsymbol{Q})$: Again we consider two cases, one when the variable appears in m_1 and the other when the variable appears in m_2. Now the claim follows by applying induction hypothesis on m_1, m_2 in both the cases respectively and the observation that in one case $(n/x)(m_1\ m_2) = ((n/x)m_1)\ m_2$ and in the other $(n/x)(m_1\ m_2) = m_1\ ((n/x)m_2)$. The other case follows vacuously since in this case both m_1, m_2 are closed.

- $\Delta, \Gamma \vdash m_1 \otimes m_2 : A \otimes B$: We consider two cases, one when the variable $\boldsymbol{x}$ appears in m_1 and the other when x appears in m_2 respectively. We apply induction hypothesis on the terms m_1, m_2 in both the cases. The claim follows using the observation that $(n/x)(m_1 \otimes m_2) = (n/x)m_1 \otimes m_2$ in one case and $(n/x)(m_1 \otimes m_2) = m_1 \otimes (n/x)m_2$ in the other.

- $\Delta, \Gamma \vdash \delta_{\otimes}(m_1, xy.m_2) : C$: We consider two cases, one when the variable appears in m_1 and the other when it appears in m_2. In both the cases we apply the induction hypothesis on the terms m_1, m_2 respectively. Then again, the claim follows from the observation that $(n/x)\delta_{\otimes}(m_1, yz.m_2)$ is equal to $\delta_{\otimes}((n/x)m_1, yz.m_2)$ in one case and $(n/x)\delta_{\otimes}(m_1, yz.m_2)$ is equal to $\delta_{\otimes}(m_1, (n/x)yz.m_2)$ in the other.

- $\Delta \vdash \mathit{inl}(m') : A \oplus B$ or $\Delta \vdash \mathit{inr}(m') : A \oplus B$: This case follows by applying the induction hypothesis on m' and the observation that $(n/x)\mathit{inl}(m') = \mathit{inl}((n/x)m')$. The case for $\mathit{inr}(m')$ follows similarly.

- $\Delta, \Gamma \vdash \delta_{\oplus}(m_1, x.m_2, y.m_3) : C$: We consider two cases, one when the variable appears in m_1 and the other when it appears in the terms m_2, m_3. Both the cases follow by applying the induction hypothesis on the terms m_1, m_2, m_3 respectively and the observation that $(n/x)\delta_{\oplus}(m_1, y.m_2, z.m_3)$ is equal to $\delta_{\oplus}((n/x)m_1, y.m_2, z.m_3)$ in one case and $(n/x)\delta_{\oplus}(m_1, y.m_2, z.m_3)$ is equal to $\delta_{\oplus}(m_1, (n/x)y.m_2, (N/x)z.m_3)$ in the other.

- $\vdash \mathcal{B}(\boldsymbol{m'}) : \mathcal{B}(\boldsymbol{P})$: This case follows vacuously since $\boldsymbol{m'}$ is a closed term.

- $\Gamma \vdash \tau(m') : \mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q})$: This case follows from the induction hypothesis and the observation that $(n/x)\tau(m') = \tau((n/x)m')$.

- $\Gamma \vdash p \cdot m' : A$: This case follows from the induction hypothesis applied to m' and the observation that $(n/x)(p \cdot m') = p \cdot (n/x)m'$.

- $\Gamma \vdash m_1 \mathbin{+} m_2 : A$: This case follows from the induction hypothesis applied to m_1, m_2 and the observation that $(n/x)(m_1 \mathbin{+} m_2) = (n/x)m_1 \mathbin{+} (n/x)m_2$.

□

Theorem 20 (Preservation). *$\Gamma \vdash M : T$ and $M \hookrightarrow N$ imply $\Gamma \vdash N : T$.*

Proof. Note that from the rules for $\leftrightarrows$ in Figure 6.3 and the rules sum and prod(α) from Figure 6.2 it follows that propositions are preserved by $\leftrightarrows$. Hence, it suffices to prove that $\rightarrow$ preserves propositions. We prove this using induction on the relation $\rightarrow$. We focus on the basic reduction rules in Figure 6.3 and omit the proof for contextual rules since they follow easily from the induction hypothesis.

- $M = \boldsymbol{\delta_1(a \cdot *, n)}$ and $N = \boldsymbol{a \cdot n}$. Since we have that $\vdash \boldsymbol{* : 1}$, if we have $\boldsymbol{\Gamma \vdash n : P}$, it follows from the derivation rules that $\boldsymbol{\Gamma \vdash a \cdot n : P}$.

- $M = \delta_{\mathbf{1}}(p \cdot *, n)$ and $N = a \cdot n$. Since we have that $\vdash a \cdot * : 1$, if we have $\cdot \vdash n : A$, it follows from the derivation rules that $\cdot \vdash a \cdot n : A$.

- $M = \boldsymbol{(\lambda x.m)v}$ and $N = \boldsymbol{(m_2/x)m_1}$, and the result holds by Lemma 18.

- $M = (\lambda x.m)\ v$ and $N = (v/x)m$. The result holds by Lemma 18.

- $M = \boldsymbol{\delta^1_{\&}(\langle v_1, v_2\rangle, x.n)}$ and $N = \boldsymbol{(v_1/x)n}$. The result holds by Lemma 18. The case for $\boldsymbol{\delta^2_{\&}(\langle v_1, v_2\rangle, x.n)}$ is similar.

- $M = \delta_{\oplus}(\mathit{inl}(v), x.m, y.n)$ or $M = \delta_{\oplus}(\mathit{inr}(v), x.m, y.n)$ and $N = (v/x)/m$ or $N = (v/y)n$ respectively. The result holds in both the cases by an application of Lemma 18.

- $M = \boldsymbol{\delta_{\otimes}(v^1_n \otimes v^2_n, xy.m')}$ and $N = \boldsymbol{(v^1_n/x, v^2_n/y)m'}$. The result follows by a double application of Lemma 18.

- $M = \delta_{\otimes}(v^1_n \otimes v^2_n, xy.n)$ and $N = (v^1_n/x, v^2_n/y)n$. The result holds by a double application of Lemma 18.

- $M = \boldsymbol{a \cdot \langle\rangle}$ and $N = \langle\rangle$. The claim follows from the rules $\top^p_i$, sum^p, and $\mathrm{prod}(a)^p$.

- $M = \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{\lambda x.m_i}$ and $N = \boldsymbol{\lambda x.} \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{m_i}$. The claim follows from the rules $\multimap_i^p$, sum^p, $\mathrm{prod}(a)^p$.

- $M = \sum_{i=1}^{n} p_i \cdot \lambda x.m_i$ and $N = \lambda x. \sum_{i=1}^{n} p_i \cdot m_i$. The claim follows from the rules $\multimap_i^m$, sum^m, $\mathrm{prod}(p)^m$.

- $M = \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \langle \boldsymbol{u_{i1}}, \boldsymbol{u_{i2}} \rangle$ and $N = \langle \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{u_{i1}}, \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{u_{i2}} \rangle$. The claim follows from the rules $\&_i^p$, sum^p and $\mathrm{prod}(a)^p$.

- $M = \delta_{\oplus}(\sum_{i=1}^{n} p_i \cdot v_{bi}, x.o, y.s)$ and $N = \sum_{i=1}^{n} p_i \cdot \delta_{\oplus}(v_{bi}, x.o, y.s)$. The claim follows from the rules $\oplus_e^m$, sum^m, $\mathrm{prod}(p)^m$.

- $M = \boldsymbol{\delta_{\otimes}}(\sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{v_b^i}, \boldsymbol{xy.n})$ and $N = \sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{\delta_{\otimes}}(\boldsymbol{v_b^i}, \boldsymbol{xy.n})$. The claim follows from the rules $\otimes_e^p$, sum^p, $\mathrm{prod}(a)^p$.

- $M = inl(\sum_{i=1}^{n} p_i \cdot v_{bi})$ and $N = \sum_{i=1}^{n} p_i \cdot inl(v_{bi})$. The claim follows from the rules $\oplus_e^{i1}$, sum^p and $\mathrm{prod}(p)^p$. The case for $inr(\sum_{i=1}^{n} p_i \cdot v_{bi})$ is similar.

- $M = \tau(\mathcal{B}(\boldsymbol{v}_1) \otimes \mathcal{B}(\boldsymbol{v}_2))$ and $N = \mathcal{B}(\boldsymbol{v_1} \otimes \boldsymbol{v_2})$. The claim follows from the rules $\mathcal{B}(\otimes)^m$, $\otimes_i^p$ and $\mathcal{B}_i^m$.

- $M = \tau(\sum_{i=1}^{n} p_i \cdot v_{bi})$ and $N = \sum_{i=1}^{n} p_i \cdot \tau(v_{bi})$. The claim follows from the rules $\mathcal{B}(\otimes)^m$, sum^m and $\mathrm{prod}(p)^m$.

- $M = \mathcal{B}(\boldsymbol{a} \cdot *)$ and $N = |\boldsymbol{a}|^2 \cdot \mathcal{B}(*)$. The claim follows by applying the rules $\mathbf{1}_i^p$, $\mathcal{B}_i^m$, $\mathrm{prod}(a)^p$ and $\mathrm{prod}(|a|^2)^m$.

- $M = (\sum_{i=1}^{n} p_i \cdot \mathcal{B}(\boldsymbol{v_i}))\ (\sum_{j=1}^{k} q_j \cdot \mathcal{B}(\boldsymbol{w_j}))$ and $N = \sum_{i,j} p_i q_j \cdot \mathcal{B}(\boldsymbol{v_i}\ \boldsymbol{w_j})$. The claim follows from the rules $\mathcal{B}(\multimap)_e^m$, sum^m and $\mathrm{prod}(p)^m$.

□

Strong Normalization Here we show that the relation $\hookrightarrow$ always terminates in finitely many steps. To this end, we first extend the relation with additional rules, and show that strong normalisation holds for this extended relation. Since the original relation is included in the extended one, this immediately implies that $\hookrightarrow$ is also strongly normalising.

The following rules are added to form the extended relation:

$$M + N \hookrightarrow M$$
$$M + N \hookrightarrow N$$
$$\alpha \cdot M \hookrightarrow M$$

The *length* of a strongly terminating term M, denoted $\ell(M)$, is the maximal length of any reduction sequence starting from M.

Lemma 19. *If M and N are strongly normalising, then so is $M + N$.*

Proof. We show that all one-step reducts of $M + N$ are strongly normalising, by induction on $\ell(M) + \ell(N)$, and then by structural induction on the size of the term.

If the reduction applies to either M or N, the claim follows directly from the induction hypothesis. Otherwise, the reduction applies to the outermost $+$. In that case, the possible reductions are:

$$\lambda x.m + \lambda x.n \hookrightarrow \lambda x.(1 \cdot m + 1 \cdot n)$$
$$\boldsymbol{\lambda x.m + \lambda x.n \hookrightarrow \lambda x.(1 \cdot m + 1 \cdot n)}$$
$$\boldsymbol{a \cdot \langle m_1, m_2 \rangle + b \cdot \langle m_1', m_2' \rangle \hookrightarrow \langle a \cdot m_1 + b \cdot m_1', a \cdot m_2 + b \cdot m_2' \rangle}$$
$$M + N \hookrightarrow M$$
$$M + N \hookrightarrow N$$

In all cases, the reduct either decreases the size of the term or is built from strongly normalising subterms. Hence, each reduct is strongly normalising by the induction hypothesis or by assumption. □

Lemma 20. *If M is strongly normalising, then so is $\alpha \cdot M$.*

Proof. As in the previous lemma, we prove that all one-step reducts of $\alpha \cdot M$ are strongly normalising, by induction on $\ell(M)$, and then on the size of the term.

If the reduction applies to M, the claim follows from the induction hypothesis. Otherwise, the reduction applies to the outermost scalar multiplication. The possible reductions are:

$$p \cdot \lambda x.m \hookrightarrow \lambda x.(p \cdot m)$$
$$\boldsymbol{a \cdot \lambda x.m \hookrightarrow \lambda x.(a \cdot m)}$$
$$\boldsymbol{a \cdot \langle\rangle \hookrightarrow \langle\rangle}$$
$$\boldsymbol{a \cdot \langle m_1, m_2 \rangle \hookrightarrow \langle a \cdot m_1, a \cdot m_2 \rangle}$$

In all cases, either the term decreases in size, or it is constructed from strongly normalising subterms. Thus, the result follows by induction or by the assumption that M is strongly normalising. □

We let **SN** denote the set of proof terms that terminate in finitely many steps.

Definition 98. *We inductively define the interpretation $\boldsymbol{SN}(T)$ of each proposition T as the set of strongly normalising proof terms of T, as follows:*

$$\begin{aligned}
\boldsymbol{SN}(\top) &:= \{\boldsymbol{m} \in \mathbf{SN}\} \\
\boldsymbol{SN}(\mathbf{1}) &:= \{\boldsymbol{m} \in \mathbf{SN}\} \\
\boldsymbol{SN}(\boldsymbol{P} \multimap \boldsymbol{Q}) &:= \{\boldsymbol{m} \in \mathbf{SN} \mid \boldsymbol{m} \hookrightarrow^* \lambda x.n \Rightarrow \forall \boldsymbol{v} \in \boldsymbol{SN}(\boldsymbol{P}),\ (\boldsymbol{v}/x)n \in \boldsymbol{SN}(\boldsymbol{Q})\} \\
\boldsymbol{SN}(\boldsymbol{P} \,\&\, \boldsymbol{Q}) &:= \{\boldsymbol{m} \in \mathbf{SN} \mid \boldsymbol{m} \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle \Rightarrow \boldsymbol{v_1} \in \boldsymbol{SN}(\boldsymbol{P}) \text{ and } \boldsymbol{v_2} \in \boldsymbol{SN}(\boldsymbol{Q})\} \\
\boldsymbol{SN}(\boldsymbol{P} \otimes \boldsymbol{Q}) &:= \{\boldsymbol{m} \in \mathbf{SN} \mid \boldsymbol{m} \hookrightarrow^* \boldsymbol{v_1} \otimes \boldsymbol{v_2} \Rightarrow \boldsymbol{v_1} \in \boldsymbol{SN}(\boldsymbol{P}) \text{ and } \boldsymbol{v_2} \in \boldsymbol{SN}(\boldsymbol{Q})\} \\
\boldsymbol{SN}(\overline{o}) &:= \{m \in \mathbf{SN}\} \\
\boldsymbol{SN}(1) &:= \{m \in \mathbf{SN}\} \\
\boldsymbol{SN}(A \multimap B) &:= \{m \in \mathbf{SN} \mid m \hookrightarrow^* \lambda x.m' \Rightarrow \forall v \in \boldsymbol{SN}(A),\ (v/x)m' \in \boldsymbol{SN}(B)\} \\
\boldsymbol{SN}(A \oplus B) &:= \left\{ m \in \mathbf{SN} \mid \begin{array}{l} m \hookrightarrow^* inl(v) \Rightarrow v \in \boldsymbol{SN}(A), \\ m \hookrightarrow^* inr(v) \Rightarrow v \in \boldsymbol{SN}(B) \end{array} \right\} \\
\boldsymbol{SN}(A \otimes B) &:= \{m \in \mathbf{SN} \mid m \hookrightarrow^* v \otimes w \Rightarrow v \in \boldsymbol{SN}(A) \text{ and } w \in \boldsymbol{SN}(B)\} \\
\boldsymbol{SN}(\mathcal{B}(\boldsymbol{P})) &:= \{m \in \mathbf{SN} \mid m \hookrightarrow^* \mathcal{B}(\boldsymbol{v}) \Rightarrow \boldsymbol{v} \in \boldsymbol{SN}(\boldsymbol{P})\}
\end{aligned}$$

Definition 99 (Definitions and notations). *We introduce some important terminology:*

- *A* substitution function *is a map* $\theta : \textsc{Vars} \to \textsc{Pure Values} \cup \textsc{Mixed Values}$. *Given a term* M*, we write* $\theta(M)$ *for the result of replacing all free variables in* M *by their image under* θ*.*

- *Given a context* Γ *and a substitution function* θ*, we say that* θ *is* valid for Γ*, written* $\theta \vDash \Gamma$*, if for all* $x : T \in \Gamma$*, we have* $\theta(x) \in \boldsymbol{SN}(T)$*.*

- *For a term* M*, we define the set of one-step reducts of* M *as:*

$$\mathit{Red}(M) := \{N \mid M \hookrightarrow N\}$$

- *A term is said to be* neutral *if the last rule used in its derivation is an elimination rule, or the rule* $\mathcal{B}(\otimes)^m$*.*

Lemma 21 (CR3). *Let* M *be a neutral proof term of* T*. If* $N \in \boldsymbol{SN}(T)$ *for all* $N \in \mathit{Red}(M)$*, then* $M \in \boldsymbol{SN}(T)$*.*

Proof. Since all one-step reducts of M are strongly normalising, all reduction sequences originating from M are finite. Thus, $M \in \mathbf{SN}$. Moreover, since M is neutral, it is not a value.

We proceed by case analysis on the proposition T:

- $T \in \{\mathbf{1}, 1, \top, \overline{\boldsymbol{o}}\}$: Immediate, as M strongly terminates by assumption and no further structure is required.

- $T = \boldsymbol{P} \multimap \boldsymbol{Q}$ (or $T = A \multimap B$): We show the pure case; the mixed case is analogous. Let $M \hookrightarrow N \hookrightarrow^* \lambda x.m$. Since $N \in \mathbf{SN}(T)$, we know that for all $\boldsymbol{v} \in \mathbf{SN}(\boldsymbol{P})$, we have $(\boldsymbol{v}/x)m \in \mathbf{SN}(\boldsymbol{Q})$. Thus, by definition of $\mathbf{SN}(\boldsymbol{P} \multimap \boldsymbol{Q})$, we conclude $M \in \mathbf{SN}(T)$.

- $T = \boldsymbol{P} \mathbin{\&} \boldsymbol{Q}$: In this case, $M \hookrightarrow N \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle$. Since $N \in \mathbf{SN}(T)$, we know that $\boldsymbol{v_1} \in \mathbf{SN}(\boldsymbol{P})$ and $\boldsymbol{v_2} \in \mathbf{SN}(\boldsymbol{Q})$. Therefore, $M \in \mathbf{SN}(\boldsymbol{P} \mathbin{\&} \boldsymbol{Q})$.

- $T = A \oplus B$: In this case, $M \hookrightarrow N \hookrightarrow^* \mathit{inl}(v)$ or $\mathit{inr}(v)$. Since $N \in \mathbf{SN}(T)$, we have either $v \in \mathbf{SN}(A)$ or $v \in \mathbf{SN}(B)$, respectively. Hence, $M \in \mathbf{SN}(A \oplus B)$.

- $T = \boldsymbol{P} \otimes \boldsymbol{Q}$ (or $T = A \otimes B$): Again, we show the pure case. Let $M \hookrightarrow N \hookrightarrow^* \boldsymbol{v_1} \otimes \boldsymbol{v_2}$. Since $N \in \mathbf{SN}(T)$, we know $\boldsymbol{v_1} \in \mathbf{SN}(\boldsymbol{P})$ and $\boldsymbol{v_2} \in \mathbf{SN}(\boldsymbol{Q})$. Thus, $M \in \mathbf{SN}(\boldsymbol{P} \otimes \boldsymbol{Q})$.

- $T = \mathcal{B}(\boldsymbol{P})$: In this case, $M \hookrightarrow N \hookrightarrow^* \mathcal{B}(\boldsymbol{v})$. Since $N \in \mathbf{SN}(T)$, we have $\boldsymbol{v} \in \mathbf{SN}(\boldsymbol{P})$. Hence, $M \in \mathbf{SN}(\mathcal{B}(\boldsymbol{P}))$. □

□

Lemma 22 (Adequacy of sum). *If* $M, N \in \boldsymbol{SN}(T)$*, then* $M + N \in \boldsymbol{SN}(T)$*.*

Proof. Proof is again by induction on the proposition T. Again by Lemma 19 we have that $M + N$ always terminates.

- T is $\mathbf{1}$ or 1: Follows directly from Lemma 19.

- T is $\boldsymbol{P \,\&\, Q}$: Since $M \mathbin{+} N$ terminates and T is $\boldsymbol{P \,\&\, Q}$, we have $M \mathbin{+} N \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle$. Hence, either $M \hookrightarrow^* \langle \boldsymbol{u_1}, \boldsymbol{u_2} \rangle$, $N \hookrightarrow^* \langle \boldsymbol{u_1'}, \boldsymbol{u_2'} \rangle$ with $\boldsymbol{u_1} \mathbin{+} \boldsymbol{u_1'} \hookrightarrow^* \boldsymbol{v_1}$ and $\boldsymbol{u_2} \mathbin{+} \boldsymbol{u_2'} \hookrightarrow^* \boldsymbol{v_2}$ or $M \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle$ or $N \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle$. The first case follows by the induction hypothesis and the assumption that $M, N \in \mathbf{SN}(T)$. The other two cases follow from the assumption $M, N \in \mathbf{SN}(T)$.

- T is $\boldsymbol{P_1 \multimap P_2}$ or $A \multimap B$: Since both the cases are similar, we present only one case here. Since $M \mathbin{+} N$ terminates and T is $\boldsymbol{P \multimap Q}$, we have $M \mathbin{+} N \hookrightarrow^* \boldsymbol{\lambda x.m}$. Now, either $M \hookrightarrow \boldsymbol{\lambda x.m_1}$, $M \hookrightarrow \boldsymbol{\lambda x.m_2}$ and $\boldsymbol{m}$ is of the form $\boldsymbol{m_1} \mathbin{+} \boldsymbol{m_2}$ or $M \hookrightarrow^* \boldsymbol{\lambda x.m}$ or $N \hookrightarrow^* \boldsymbol{\lambda x.m}$. The first case follows by an application of the induction hypothesis to $\boldsymbol{(v/x)m_1} \mathbin{+} \boldsymbol{m_2}$ and the assimption that $M, N \in \mathbf{SN}(T)$. The other two cases follow from the assumption $M, N \in \mathbf{SN}(T)$.

- T is $\boldsymbol{P \otimes Q}$ or $A \otimes B$: We present only one case here, the other case follows using a similar argument. Since, $M \mathbin{+} N$ terminates and T is $\boldsymbol{P \otimes Q}$ we have $M \mathbin{+} N \hookrightarrow^* \boldsymbol{v_1 \otimes v_2}$. Then either $M \hookrightarrow^* \boldsymbol{v_1 \otimes v_2}$ or $N \hookrightarrow^* \boldsymbol{v_1 \otimes v_2}$. Both cases follow from the assumption that $M, N \in \mathbf{SN}(T)$.

- T is $A \oplus B$: Since, $M \mathbin{+} N$ terminates and T is $A \oplus B$ we have $M \mathbin{+} N \hookrightarrow^* \mathit{inl}(v)$(or $\mathit{inr}(v)$). Then either $M \hookrightarrow^* \mathit{inl}(v)$(or $\mathit{inr}(v)$) or $N \hookrightarrow^* \mathit{inl}(v)$(or $\mathit{inr}(v)$). Both cases follow from the assumption that $M, N \in \mathbf{SN}(T)$.

- T is $\mathcal{B}(\boldsymbol{P})$: Since, $M \mathbin{+} N$ terminates and T is $\mathcal{B}(\boldsymbol{P})$ we have $M \mathbin{+} N \hookrightarrow^* \mathcal{B}(\boldsymbol{v})$. Then either $M \hookrightarrow^* \mathcal{B}(\boldsymbol{v})$ or $N \hookrightarrow^* \mathcal{B}(\boldsymbol{v})$. Both cases follow from the assumption that $M, N \in \mathbf{SN}(T)$.

□

Lemma 23 (Adequacy of prod(α)). *If $M \in \boldsymbol{SN}(T)$, then $\alpha \cdot M \in \boldsymbol{SN}(T)$.*

Proof. Proof is by induction on the proposition T. Note that by Lemma 20 we have that $\alpha \cdot M$ always terminates.

- T is $\mathbf{1}$ or 1: Directly follows from Lemma 20.

- T is $\boldsymbol{P \,\&\, Q}$: Since, $\alpha \cdot M$ terminates and T is $\boldsymbol{P \,\&\, Q}$ it follows that $\alpha \cdot M \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle$. Then either $M \hookrightarrow^* \langle \boldsymbol{u_1}, \boldsymbol{u_2} \rangle$ with $\alpha \cdot \boldsymbol{u_1} \hookrightarrow^* \boldsymbol{v_1}$ and $\alpha \cdot \boldsymbol{u_2} \hookrightarrow^* \boldsymbol{v_2}$ or $M \hookrightarrow^* \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle$. In the first case, the conclusion follows by an application of induction hypothesis to $\boldsymbol{u_1}, \boldsymbol{u_2}$ and $M \in \mathbf{SN}(T)$. In the second case, the conclusion follows from the fact that $M \in \mathbf{SN}(T)$.

- T is $\boldsymbol{P \multimap Q}$ or $A \multimap B$: We present only one case here, the other case follows similarly. Since $\alpha \cdot M$ terminates and T is $\boldsymbol{P \multimap Q}$, it follows that $\alpha \cdot M \hookrightarrow^* \boldsymbol{\lambda x.m}$. Now either $M \hookrightarrow^* \boldsymbol{\lambda x.n}$ and m is $\alpha \cdot \boldsymbol{n}$ or $M \hookrightarrow^* \boldsymbol{\lambda x.m}$. The first case follows from the induction hypothesis applied to $\boldsymbol{(v/x)n}$ for $\boldsymbol{v} \in \mathbf{SN}(\boldsymbol{P})$ and $M \in \mathbf{SN}(T)$. The latter case follows from $M \in \mathbf{SN}(T)$.

- T is $\boldsymbol{P \otimes Q}$ or $A \otimes B$: Again we present only one case here, the argument for the other case is similar. Since $\alpha \cdot M$ terminates and T is $\boldsymbol{P \otimes Q}$, we have $\alpha \cdot M \hookrightarrow^* \boldsymbol{v_1 \otimes v_2}$ and $M \hookrightarrow^* \boldsymbol{v_1 \otimes v_2}$. Now the claim follows since $M \in \mathbf{SN}(T)$.

- T is $A \oplus B$: Since $\alpha \cdot M$ terminates and T is $A \oplus B$, we have $\alpha \cdot M \hookrightarrow^* \mathit{inl}(v)$(or $\mathit{inr}(v)$) and $M \hookrightarrow^* \mathit{inl}(v)$(or $\mathit{inr}(v)$). Now the claim follows since $M \in \mathbf{SN}(T)$.

- T is $\mathcal{B}(\boldsymbol{P})$: Since $\alpha \cdot M$ terminates and T is $\mathcal{B}(\boldsymbol{P})$, we have $\alpha \cdot M \hookrightarrow^* \mathcal{B}(\boldsymbol{v})$ and $M \hookrightarrow^* \mathcal{B}(\boldsymbol{v})$. Now the claim follows since $M \in \mathbf{SN}(T)$.

□

Lemma 24 (Adequacy of 0_e^m). *If $m \in \boldsymbol{SN}(\overline{o})$ then $\delta_0(m) \in \boldsymbol{SN}(A)$.*

Proof. m always terminates. Any reduction acting on $\delta_0(m)$ can only reduce m, hence it always terminates. Since, $\delta_0(m)$ can never reduce to a closed value, it follows that $\delta_0(m) \in \mathbf{SN}(A)$. □

Lemma 25 (Adequacy of $\mathbf{1}_e^p$). *If $\boldsymbol{m_1} \in \boldsymbol{SN}(\mathbf{1})$ and $\boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{P})$, then $\boldsymbol{\delta_1(m_1, m_2)} \in \boldsymbol{SN}(\boldsymbol{P})$.*

Proof. We use induction on $\ell(\boldsymbol{m_1})$ and Lemma 21 to prove that $\boldsymbol{\delta_1(m_1, m_2)} \in \mathbf{SN}(\boldsymbol{P})$. From Lemma 21 it suffices to show that all one-step reducts of the term $\boldsymbol{\delta_1(m_1, m_2)}$ are in $\mathbf{SN}(\boldsymbol{P})$. Now if a reduction applies to $\boldsymbol{m_1}$, then by induction hypothesis the resulting term is in $\mathbf{SN}(\boldsymbol{P})$. Otherwise, $\boldsymbol{m_1}$ is of the form $\boldsymbol{a \cdot *}$, in which case, $\boldsymbol{\delta_1(m_1, m_2)} \hookrightarrow \boldsymbol{a \cdot m_2}$. Now again from Lemma 23 it follows that $\boldsymbol{a \cdot m_2} \in \mathbf{SN}(\boldsymbol{P})$. Hence, from Lemma 21 it follows that $\boldsymbol{\delta_1(m_1, m_2)} \in \mathbf{SN}(\boldsymbol{P})$. □

Lemma 26 (Adequacy of 1_e^m). *If $m_1 \in \boldsymbol{SN}(1)$ and $m_2 \in \boldsymbol{SN}(A)$, then $\delta_1(m_1, m_2) \in \boldsymbol{SN}(A)$.*

Proof. Proof for this lemma follows by the same argument as for Lemma 25. □

Lemma 27 (Adequacy of $\multimap_i^p$). *If $\boldsymbol{v} \in \boldsymbol{SN}(\boldsymbol{P})$ and $\boldsymbol{(v/x)m} \in \boldsymbol{SN}(\boldsymbol{Q})$ then $\boldsymbol{\lambda x.m} \in \boldsymbol{SN}(\boldsymbol{P} \multimap \boldsymbol{Q})$.*

Proof. Straightforward from the definition of $\mathbf{SN}(\boldsymbol{P} \multimap \boldsymbol{Q})$. □

Lemma 28 (Adequacy of $\multimap_i^m$). *If $v \in \boldsymbol{SN}(A)$ and $(v/x)m \in \boldsymbol{SN}(B)$ then $\lambda x.m \in \boldsymbol{SN}(A \multimap B)$.*

Proof. Straightforward from the definition of $\mathbf{SN}(A \multimap B)$. □

Lemma 29 (Adequacy of $\multimap_e^p$). *If $\boldsymbol{n} \in \boldsymbol{SN}(\boldsymbol{P})$ and $\boldsymbol{m} \in \boldsymbol{SN}(\boldsymbol{P} \multimap \boldsymbol{Q})$ then $\boldsymbol{m\ n} \in \boldsymbol{SN}(\boldsymbol{Q})$.*

Proof. Proof is by induction on $\ell(m) + \ell(n)$ and Lemma 21. If a reduction applies to either $\boldsymbol{m}$ or $\boldsymbol{n}$, then the resulting term is in $\mathbf{SN}(\boldsymbol{Q})$ by the induction hypothesis. Otherwise, both $\boldsymbol{m}$ and $\boldsymbol{n}$ are values and $\boldsymbol{m\ n}$ reduces to $\boldsymbol{(n/x)m'}$ for some term $\boldsymbol{m'}$. Since $\boldsymbol{m} \in \mathbf{SN}(\boldsymbol{P} \multimap \boldsymbol{Q})$, we have that $\boldsymbol{(n/x)m'} \in \mathbf{SN}(\boldsymbol{Q})$. Hence, all one step reducts of $\boldsymbol{m\ n} \in \mathbf{SN}(\boldsymbol{Q})$. Therefore, by Lemma 21 it follows that $\boldsymbol{m\ n} \in \mathbf{SN}(\boldsymbol{Q})$. □

Lemma 30 (Adequacy of $\multimap_e^m$). *If $n \in \boldsymbol{SN}(A)$ and $m \in \boldsymbol{SN}(A \multimap B)$ then $m\ n \in \boldsymbol{SN}(B)$.*

Proof. Analogous to the proof of Lemma 29. □

Lemma 31 (Adequacy of $\&_i^p$). *If $\boldsymbol{m_1} \in \boldsymbol{SN}(\boldsymbol{P})$ and $\boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{Q})$ then $\boldsymbol{\langle m_1, m_2 \rangle} \in \boldsymbol{SN}(\boldsymbol{P \& Q})$.*

Proof. Note that since a reduction can only apply to $\boldsymbol{\langle m_1, m_2 \rangle}$ if it applies to either $\boldsymbol{m_1}$ or $\boldsymbol{m_2}$, it follows that since $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ strongly terminate, so does $\boldsymbol{\langle m_1, m_2 \rangle}$. Moreover, we also have that $\boldsymbol{\langle m_1, m_2 \rangle} \hookrightarrow^* \boldsymbol{\langle v_1, v_2 \rangle}$ for $\boldsymbol{v_1} \in \mathbf{SN}(\boldsymbol{P}_1)$ and $\boldsymbol{v_2} \in \mathbf{SN}(\boldsymbol{P}_2)$, since $\boldsymbol{\langle m_1, m_2 \rangle} \hookrightarrow \boldsymbol{\langle m_1', m_2' \rangle} \implies \boldsymbol{m_1} \hookrightarrow \boldsymbol{m_1'}$ and $\boldsymbol{m_2} \hookrightarrow \boldsymbol{m_2'}$. □

Lemma 32 (Adequacy of $\&^p_{e1}$). *If $\boldsymbol{m_1} \in \boldsymbol{SN}(\boldsymbol{P} \,\&\, \boldsymbol{Q})$ and for all $\boldsymbol{v} \in \boldsymbol{SN}(\boldsymbol{P})$, $(\boldsymbol{v}/\boldsymbol{x})\boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{R})$, then $\boldsymbol{\delta^1_\&}(\boldsymbol{m_1}, \boldsymbol{x.m_2}) \in \boldsymbol{SN}(\boldsymbol{R})$.*

Proof. Proof is by induction on $\ell(\boldsymbol{m_1})$ and Lemma 21. If a reduction applies to $\boldsymbol{m_1}$, then we have that the resulting term is in $\mathbf{SN}(\boldsymbol{P} \,\&\, \boldsymbol{Q})$. If $\boldsymbol{m_1}$ is a value, then we have $(\boldsymbol{m_1}/\boldsymbol{x})\boldsymbol{m_2} \in \mathbf{SN}(\boldsymbol{R})$. Hence all one step reducts of $\boldsymbol{\delta^1_\&}(\boldsymbol{m_1}, \boldsymbol{x.m_2})$ are in $\mathbf{SN}(\boldsymbol{R})$. Therefore, the result follows from Lemma 21. □

Lemma 33 (Adequacy of $\&^p_{e1}$). *If $\boldsymbol{m_1} \in \boldsymbol{SN}(\boldsymbol{P} \,\&\, \boldsymbol{Q})$ and for all $\boldsymbol{v} \in \boldsymbol{SN}(\boldsymbol{Q})$, $(\boldsymbol{v}/\boldsymbol{x})\boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{R})$, then $\boldsymbol{\delta^2_\&}(\boldsymbol{m_1}, \boldsymbol{x.m_2}) \in \boldsymbol{SN}(\boldsymbol{R})$.*

Proof. Proof follows by the same argument as that for the proof of Lemma 32. □

Lemma 34 (Adequacy of $\oplus^m_{i1}$). *If $m \in \boldsymbol{SN}(A)$, then $inl(m) \in \boldsymbol{SN}(A \oplus B)$.*

Proof. Note that a reduction can only apply to $inl(m)$ through m. Hence, $inl(m)$ strongly terminates. Moreover, since $inl(m) \hookrightarrow inl(m') \implies m \hookrightarrow m'$, we have $inl(m) \in \mathbf{SN}(A \oplus B)$. □

Lemma 35 (Adequacy of $\oplus^m_{i2}$). *If $m \in \boldsymbol{SN}(B)$, then $inr(m) \in \boldsymbol{SN}(A \oplus B)$.*

Proof. This result follows using the same argument as the one for the proof of Lemma 34. □ □

Lemma 36 (Adequacy of $\oplus^m_e$). *If $m \in \boldsymbol{SN}(A \oplus B)$ and for all $m_1 \in \boldsymbol{SN}(A), m_2 \in \boldsymbol{SN}(B)$, we have $(m_1/x)n \in \boldsymbol{SN}(C)$ and $(m_2/y)o \in \boldsymbol{SN}(C)$, then $\delta_\oplus(m, x.n, y.o) \in \boldsymbol{SN}(C)$.*

Proof. Proof is by induction on $\ell(m)$ and an application of Lemma 21. If a reduction applies to m then the resulting term is in $\mathbf{SN}(A \oplus B)$ as a result of the induction hypothesis. If, m is a value, and the resulting term is of the form $(m_1/x)n$ or $(m_2/y)o$. Otherwise, m is of the form $\sum_{i=1}^n v_i$, where v_i is of the form $inl(v)$ or $inr(v)$. Then $\delta_\oplus(m, x.n, y.o)$ reduces to $\sum_{i=1}^n p_i \cdot \delta_\oplus(v_i, x.n, y.o)$. This term is in $\mathbf{SN}(A \oplus B)$ by the induction hypothesis, Lemma 23 and Lemma 22. Hence, all one-step reducts of $\delta_\oplus(m, x.n, y.o)$ are in $\mathbf{SN}(C)$. Therefore, the claim follows by Lemma 21. □ □

Lemma 37 (Adequacy of $\otimes^p_i$). *If $\boldsymbol{m_1} \in \boldsymbol{SN}(\boldsymbol{P})$ and $\boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{Q})$ then $\boldsymbol{m_1} \otimes \boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{P} \otimes \boldsymbol{Q})$.*

Proof. Note that since a reduction can only apply to $\boldsymbol{m_1} \otimes \boldsymbol{m_2}$ if it applies to either $\boldsymbol{m_1}$ or $\boldsymbol{m_2}$, it follows that since $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ strongly terminate, so does $\boldsymbol{m_1} \otimes \boldsymbol{m_2}$. Moreover, we also have that $\boldsymbol{m_1} \otimes \boldsymbol{m_2} \hookrightarrow^* \boldsymbol{v_1} \otimes \boldsymbol{v_2}$ for $\boldsymbol{v_1} \in \mathbf{SN}(\boldsymbol{P}_1)$ and $\boldsymbol{v_2} \in \mathbf{SN}(\boldsymbol{P}_2)$, since $\boldsymbol{m_1} \otimes \boldsymbol{m_2} \hookrightarrow \boldsymbol{m_1'} \otimes \boldsymbol{m_2'} \implies \boldsymbol{m_1} \hookrightarrow \boldsymbol{m_1'}$ and $\boldsymbol{m_2} \hookrightarrow \boldsymbol{m_2'}$. □

Lemma 38 (Adequacy of $\otimes^m_i$). *If $m \in \boldsymbol{SN}(A)$ and $n \in \boldsymbol{SN}(B)$ then $m \otimes n \in \boldsymbol{SN}(A \otimes B)$.*

Proof. The result follows using the same argument as the one for Lemma 38. □

Lemma 39 (Adequacy of $\otimes^p_e$). *If $\boldsymbol{m_1} \in \boldsymbol{SN}(\boldsymbol{P} \otimes \boldsymbol{Q})$ and for all $\boldsymbol{v_1} \in \boldsymbol{SN}(\boldsymbol{P}), \boldsymbol{v_2} \in \boldsymbol{SN}(\boldsymbol{Q})$ we have $(\boldsymbol{v_1}/\boldsymbol{x}, \boldsymbol{v_2}/\boldsymbol{y})\boldsymbol{m_2} \in \boldsymbol{SN}(\boldsymbol{R})$ then $\boldsymbol{\delta_\otimes}(\boldsymbol{m_1}, \boldsymbol{xy.m_2}) \in \boldsymbol{SN}(\boldsymbol{R})$.*

Proof. Again proof is by induction on $\ell(\boldsymbol{m_1})$ and an application of Lemma 21. If a reduction applies to $\boldsymbol{m_1}$ then the resulting term is in $\mathbf{SN}(\boldsymbol{P} \otimes \boldsymbol{Q})$ as a result of the induction hypothesis. If, $\boldsymbol{m_1}$ is a value, and the resulting term is of the form $(\boldsymbol{v_1}/\boldsymbol{x}, \boldsymbol{v_2}/\boldsymbol{y})\boldsymbol{m_2}$. Otherwise $\boldsymbol{m_1}$ is of the form $\sum_{i=1}^n \boldsymbol{a_i} \cdot \boldsymbol{v_n^1} \otimes \boldsymbol{v_n^2}$. In this case $\boldsymbol{\delta_\otimes}(\boldsymbol{m_1}, \boldsymbol{xy.m_2})$ reduces to $\sum_{i=1}^n \boldsymbol{a_i} \cdot \boldsymbol{\delta_\otimes}(\boldsymbol{v_n^1} \otimes \boldsymbol{v_n^2}, \boldsymbol{xy.m_2})$. By induction hypothesis, Lemma 23 and Lemma 22, this term is in $\mathbf{SN}(\boldsymbol{R})$. Hence, all one-step reducts of $\boldsymbol{\delta_\otimes}(\boldsymbol{m_1}, \boldsymbol{xy.m_2})$ are in $\mathbf{SN}(\boldsymbol{R})$. Therefore, the claim follows by Lemma 21. □

Lemma 40 (Adequacy of $\otimes_e^m$). *If $m_1 \in \boldsymbol{SN}(A \otimes B)$ and for all $m \in \boldsymbol{SN}(A), n \in \boldsymbol{SN}(B)$ $(m/x, n/y)m_2 \in \boldsymbol{SN}(C)$ then $\delta_\otimes(m_1, xy.m_2) \in \boldsymbol{SN}(C)$.*

Proof. The proof for this lemma follows using the same argument as the one for Lemma 39. □

Lemma 41 (Adequacy of $\mathcal{B}_i^m$). *If $\boldsymbol{m} \in \boldsymbol{SN}(\boldsymbol{P})$, then $\mathcal{B}(\boldsymbol{m}) \in \boldsymbol{SN}(\mathcal{B}(\boldsymbol{P}))$.*

Proof. Note that any reduction that applies to $\mathcal{B}(\boldsymbol{m})$ is through $\boldsymbol{m}$. Hence, $\mathcal{B}(\boldsymbol{m})$ strongly terminates. Since, $\mathcal{B}(\boldsymbol{m}) \hookrightarrow \mathcal{B}(\boldsymbol{m'}) \implies \boldsymbol{m} \hookrightarrow \boldsymbol{m'}$, we have $\mathcal{B}(\boldsymbol{m}) \in \mathbf{SN}(\mathcal{B}(\boldsymbol{P}))$. □

Lemma 42 (Adequacy of $\mathcal{B}(\multimap)_e^m$). *If $m \in \boldsymbol{SN}(\mathcal{B}(\boldsymbol{P} \multimap \boldsymbol{Q}))$ and $n \in \mathcal{B}(\boldsymbol{P})$ then $m\ n \in \boldsymbol{SN}(\mathcal{B}(\boldsymbol{Q}))$.*

Proof. Again, proof is by induction on $\ell(m)+\ell(n)$ and Lemma 21. We prove all one-step reducts of $m\ n$ are in $\mathbf{SN}(\mathcal{B}(\boldsymbol{Q}))$. If a reduction applies to either m or n, then the resulting term is in $\mathbf{SN}(\mathcal{B}(\boldsymbol{Q}))$ by induction hypothesis. Otherwise m is of the form $\sum_{i=1}^n p_i \cdot \mathcal{B}(\boldsymbol{\lambda x.m_i'})$ and n is of the form $\sum_{j=1}^k q_j \cdot \mathcal{B}(\boldsymbol{v_j})$. In this case, $m\ n$ reduces to $\sum_{i=1}^n \sum_{j=1}^k p_i q_j \cdot \mathcal{B}(\boldsymbol{\lambda x.m'\ n})$. Now the claim follows by Lemma 29, Lemma 23, Lemma 22 and the definition of $\mathbf{SN}(\mathcal{B}(\boldsymbol{P}))$ for any proposition $\boldsymbol{P}$. □

Lemma 43 (Adequacy of $\mathcal{B}(\otimes)^m$). *If $m \in \boldsymbol{SN}(\mathcal{B}(\boldsymbol{P}) \otimes \mathcal{B}(\boldsymbol{Q}))$, then $\tau(m) \in \boldsymbol{SN}(\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q}))$.*

Proof. Proof is by induction on $\ell(\boldsymbol{m})$ and Lemma 21. If a reduction applies to m then the resulting term is in $\mathbf{SN}(\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q}))$ by induction hypothesis. If m is a value of the form $\mathcal{B}(\boldsymbol{v_1}) \otimes \mathcal{B}(\boldsymbol{v_2})$ then the term reduces to $\mathcal{B}(\boldsymbol{v_1} \otimes \boldsymbol{v_2})$ which is in $\mathbf{SN}(\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q}))$, from Lemma 37 and $m \in \mathbf{SN}(\mathcal{B}(\boldsymbol{P}) \otimes \mathcal{B}(\boldsymbol{Q}))$. Otherwise m is of the form $\sum_{i=1}^n p_i \cdot v_i$, in which case one of the following rules may apply:

$$\tau(m) \hookrightarrow \textstyle\sum_{i=1}^n p_i \cdot \tau(v_i)$$
$$\tau(m) \hookrightarrow \tau(p_i \cdot v_i)$$

The resulting term in both the cases is in $\mathbf{SN}(\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q}))$ by induction hypothesis, Lemma 23 and Lemma 22. Hence, all one step reducts of $\tau(m)$ are in $\mathbf{SN}(\mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q}))$. Now the result follows by Lemma 21. □

Lemma 44 (Adequacy). *For the judgement $\Gamma \vdash M : T$, if $\theta \vDash \Gamma$ then $\theta(M) \in \boldsymbol{SN}(T)$.*

Proof. Proof is by induction on the derivation of $\Gamma \vdash M : T$.

- If the last rule is ax, $M = x$: Since $\theta \vDash \Gamma$, we have $\theta(x) \in \mathbf{SN}(T)$.
- If the last rule is sum, $M = M' + N'$: Since $\theta(M' + N') = \theta M' + \theta N'$, from the induction hypothesis and Lemma 22 it follows that $\theta M \in \mathbf{SN}(T)$.
- If the last rule is prod(α): M is $\alpha \cdot N$: Since $\theta(\alpha \cdot N) = \alpha \cdot (\theta N)$, from the induction hypothesis and Lemma 23 it follows that $\theta M \in \mathbf{SN}(T)$.
- If the last rule is $\langle\rangle_i^p$, M is $\langle\rangle$: Since M is closed and irreducible we have $\theta M = M$, and $M \in \mathbf{SN}(\top)$.
- If the last rule is 0_e^m, M is $\delta_0(m)$: Note that $\theta M = \delta_0(\theta m)$. Now the claim follows from the induction hypothesis applied to θm and Lemma 24.

- If the last rule is 1_i, M is $*$: Then M is closed irreducible proof term of 1. Hence, $\theta M = M$, and we have $\theta M \in \mathbf{SN}(T)$.

- If the last rule is 1_e, M is $\delta_{\mathbf{1}}(m, n)$: Note that $\theta\delta_{\mathbf{1}}(m, n) = \delta_{\mathbf{1}}(\theta m,\theta n)$. Now the claim follows from the induction hypothesis and Lemmas 25 and 26.

- If the last rule is $\multimap_i$, M is $\lambda x.N$: Note that $\theta(\lambda x.N) = \lambda x.(\theta N)$. We need to show that $\theta((V/x)N) \in \mathbf{SN}(T)$ for any $V \in \mathbf{SN}(S)$. However, note that $\theta((V/x)N) = \theta'(N)$ for a new substitution function, $\theta' = \theta \cup \{x \mapsto V\}$. Since $V \in \mathbf{SN}(S)$, we have $\theta' \vDash \Gamma, x : S$. Hence, by the induction hypothesis, it follows that $\theta'(M) \in \mathbf{SN}(T)$. Hence, the claim follows from Lemmas 27 and 28.

- If the last rule is $\multimap_e$, M is $M'\ N'$: Note that $\theta M = (\theta M')\ (\theta N')$. Now the claim follows using the induction hypothesis on the terms $\theta M', \theta N'$ and applying Lemma 29, Lemma 30.

- If the last rule is $\&_i^p$, M is $\langle \boldsymbol{m_1}, \boldsymbol{m_2}\rangle$: Note that $\theta M = \langle \theta\boldsymbol{m_1}, \theta\boldsymbol{m_2}\rangle$. Now the claim follows from the induction hypothesis and Lemma 31.

- If the last rule is $\&_{ei}^p$, M is $\boldsymbol{\delta^i_{\&}(m_1, x.m_2)}$: Thus, $\theta M = \boldsymbol{\delta^i_{\&}(\theta m_1, x.\theta m_2)}$. Now the claim follows from the induction hypothesis and Lemma 32, Lemma 33.

- If the last rule is $\oplus_{i1}^m$, M is $inl(m)$: In this case $\theta M = inl(\theta m)$. Now the claim follows from the induction hypothesis and Lemma 34.

- If the last rule is $\oplus_{i2}^m$, M is $inr(m)$: In this case $\theta M = inr(\theta m)$. Now the claim follows from the induction hypothesis and Lemma 35.

- If the last rule is $\oplus_e^m$, M is $\delta_\oplus(m, x.n, y.o)$: Thus, $\theta M = \delta_\oplus(\theta m, x.\theta n, y.\theta o)$. Now the claim follows from the induction hypothesis and Lemma 36.

- If the last rule is $\otimes_i$, M is $\boldsymbol{m}\otimes\boldsymbol{n}$ or $m\otimes n$: We prove one case, the other follows analogously. Note that $\theta M = \theta\boldsymbol{m} \otimes \theta\boldsymbol{n}$. Hence from the induction hypothesis it follows that $\theta\boldsymbol{m} \in \mathbf{SN}(\boldsymbol{P})$ and $\theta\boldsymbol{n} \in \mathbf{SN}(\boldsymbol{Q})$. Now the claim follows from Lemmas 37 and 38.

- If the last rule is $\otimes_e$, M is $\boldsymbol{\delta_\otimes(m, xy.n)}$ or $\delta_\otimes(m, xy.n)$: Again we prove one case, the other follows analogously. In this case $\theta M = \boldsymbol{\delta_\otimes(\theta m, xy.\theta n)}$. Hence the claim follows from the induction hypothesis and Lemmas 39 and 40.

- If the last rule is $\mathcal{B}_i^m$, M is $\mathcal{B}(\boldsymbol{m})$: Since $\boldsymbol{m}$ is a closed term $\theta\boldsymbol{m} = \boldsymbol{m}$, hence from the induction hypothesis and Lemma 41, $\theta M = \mathcal{B}(\boldsymbol{m}) \in \mathbf{SN}(\boldsymbol{P})$.

- If the last rule is $\mathcal{B}(\boldsymbol{\multimap})_e^m$, M is $m\ n$: Note that $\theta M = (\theta m)\ (\theta n)$. Now the claim follows using the induction hypothesis on the terms $\theta m, \theta n$ and applying Lemma 42.

- If the last rule is $\mathcal{B}(\otimes)^m$, M is $\tau(m)$: Claim follows by the observation that $\theta\tau(m) = \tau(\theta m)$, an application of induction hypothesis and Lemma 43.

- If the last rule is ex, then the claim directly follows from the induction hypothesis.

□

Theorem 21 (Strong Normalisation). *$\Gamma \vdash M : T$ implies M terminates.*

Proof. The result follows from Lemma 44, by using the identity function as the substitution function. □

Finally we show that the reduction relation $\hookrightarrow$ is confluent, i.e., two different reduction paths issued from a term M always terminate in the same value.

Theorem 22 (Confluence). *Let $\Gamma \vdash M : T$. If $M \hookrightarrow^* M_1$ and $M \hookrightarrow^* M_2$, then there exists a value V such that $M_1 \hookrightarrow^* V \ {}^*\!\hookleftarrow M_2$.*

Proof. The confluence of $\hookrightarrow$ follows from its structure as an interleaving of a deterministic reduction ($\to$) and the equivalence $\leftrightarrows$. The relation $\leftrightarrows$ is applied only when $\to$ is blocked, serving as an algebraic normalisation step to enable further reduction. By Theorems 17 and 19, the normal forms of $\hookrightarrow$ are unique, so any two reduction sequences from M must converge. □

6.4 Denotational Semantics

This section introduces the categorical model and subsequently the denotational semantics of $\mathcal{L}_{\mathcal{B}}$. We write **FHilb** for the category of finite-dimensional Hilbert spaces with linear maps as morphisms and we write **FC*** for the category of finite-dimensional C*-algebras and completely-positive maps as morphisms. We use **FHilb** to interpret the pure fragment of $\mathcal{L}_{\mathcal{B}}$ and **FC*** to interpret the mixed fragment. Finite-dimensional C*-algebras [T12] have sufficient structure to allow us to describe completely-positive maps and to model the quantum operations (also known as channels) that are used for mixed-state quantum computation in finite dimensions. Both categories enjoy similar categorical properties that we now recall and for which we use similar notation. See [HV19, CHK16] for more information. Throughout the remainder of the section, let X, Y, Z be finite-dimensional Hilbert spaces (C*-algebras).

Symmetric Monoidal Structure. The category **FHilb** (**FC***) has a symmetric monoidal structure with tensor unit given by the Hilbert space (C*-algebra) $\mathbb{C}$ and monoidal product given by the Hilbert space (C*-algebra) tensor product $X \otimes Y$. The left (right) unitors λ (ρ), the associator α, and symmetry σ natural isomorphisms are defined in the same way as for vector spaces.

Compact Closed Structure. The category **FHilb** (**FC***) is also compact closed. We write $\eta_X \colon \mathbb{C} \to X^* \otimes X$ for the unit and $\epsilon_X \colon X \otimes X^* \to \mathbb{C}$ for the counit of the compact closed structure, where X^* indicates the dual of X (in the sense of compact closure). It follows that **FHilb** (**FC***) is a closed symmetric monoidal category with internal hom given by $[X, Y] := X^* \otimes Y$. We write $\Phi \colon \mathbf{C}(X \otimes Y, Z) \cong \mathbf{C}(X, [Y, Z])$ for the currying natural isomorphism, where **C** stands for **FHilb** or **FC***, and we write $\mathrm{eval}_{X,Y} \colon [X, Y] \otimes X \to Y$ for the canonically induced evaluation morphism of **C**. Note that the categorical dual in the category **FC*** is a slightly modified version of the Banach space dual construction. The categorical dual X^* of a C*-algebra X is given by its opposite C*-algebra, which is a C*-algebra with the same underlying vector space structure and involution but with the multiplication reversed.

Finite Biproducts. The category **FHilb** (**FC***) also has finite biproducts. Binary biproducts are given by the Hilbert space (C*-algebra) direct sum $X \oplus Y$ and the zero object is given by the zero-dimensional Hilbert space (C*-algebra) 0. We write $!_{X,Y} \colon X \to Y$ for the unique morphism that factors through 0. We write $\pi_1 \colon X \oplus Y \to X$ and $\pi_2 \colon X \oplus Y \to Y$ for the canonical projections, which are defined as $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$. Given morphisms $f \colon Z \to X$ and $g \colon Z \to Y$ in **FHilb** (**FC***), $\langle f, g \rangle \colon Z \to X \oplus Y$ is the canonical morphism induced by the categorical product and defined by $\langle f, g \rangle(z) := (f(z), g(z))$. We write $i_1 \colon X \to X \oplus Y$ and $i_2 \colon Y \to X \oplus Y$ for the canonical coproduct injections defined by $i_1(x) := (x, 0)$ and $i_2(y) = (0, y)$.

Given morphisms $f\colon X \to Z$ and $g\colon Y \to Z$ in **FHilb** ($\mathbf{FC}^*$), we write $[f,g]\colon X \oplus Y \to Z$ for the canonical map induced by the couniversal property of $X \oplus Y$ and defined by $[f,g](x,y) = f(x) + g(y)$. Since **FHilb** ($\mathbf{FC}^*$) is symmetric monoidal closed, it follows that the monoidal product distributes over coproducts and we write $d_{X,Y,Z}\colon X \otimes (Y \oplus Z) \cong (X \otimes Y) \oplus (X \otimes Z)$ for the canonical natural isomorphism.

Linear combinations. The homsets $\mathbf{FHilb}(X,Y)$ are closed under finite $\mathbb{C}$-linear combinations, i.e. for $f_i \in \mathbf{FHilb}(X,Y)$, $\alpha_i \in \mathbb{C}$ complex scalars, $i \in I$ with I a finite set, $\sum_{i\in I} \alpha_i f_i \in \mathbf{FHilb}(H,K)$, where the sum is defined pointwise in the usual way. However, the homsets $\mathbf{FC}^*(X,Y)$ are *not* closed under such linear combinations, because if $f\colon X \to Y$ is a completely-positive map, the map $(-1)f$ need not be one. Instead, homsets in $\mathbf{FC}^*(X,Y)$ are closed under finite $\mathbb{R}_{\geq 0}$-linear combinations, again defined pointwise, so that $\sum_{i\in I} r_i f_i \in \mathbf{FC}^*(X,Y)$, where $r_i \geq 0$ for all $i \in I$.

Linear combinations behave well with these categorical constructions, as the next lemma shows.

Lemma 45. *Given compatible morphisms f, f', g, g', h in* **FHilb** ($\mathbf{FC}^*$) *and a scalar $a \in \mathbb{C}$ ($a \in \mathbb{R}_{\geq 0}$), the following equations hold:*

$$\begin{array}{rlrl}
\Phi(f \mathbin{+} g) & = \Phi(f) \mathbin{+} \Phi(g) & \Phi(a \cdot f) & = a \cdot \Phi(f) \\
\langle f,g\rangle \mathbin{+} \langle f',g'\rangle & = \langle f \mathbin{+} f', g \mathbin{+} g'\rangle & a \cdot \langle f,g\rangle & = \langle a \cdot f, a \cdot g\rangle \\
[f,g] \mathbin{+} [f',g'] & = [f \mathbin{+} f', g \mathbin{+} g'] & a \cdot [f,g] & = [a \cdot f, a \cdot g] \\
(f \mathbin{+} g) \otimes h & = f \otimes h \mathbin{+} g \otimes h & a \cdot (f \otimes h) & = (a \cdot f) \otimes h = f \otimes (a \cdot h) \\
h \otimes (f \mathbin{+} g) & = h \otimes f \mathbin{+} h \otimes g &&
\end{array}$$

Proof. Straightforward verification. □

Before we proceed further we recall the following example of a C*-algebra.

Example 77. If H is a Hilbert space, then the space $\mathcal{B}(H)$ of bounded linear operators on H, has the structure of a C*-algebra. The vector space structure is obvious, the involution is given by taking adjoints, the multiplication by composition of functions, the multiplicative unit is id_H and the norm is the usual operator norm. When H is finite-dimensional, as assumed in this paper, with dimension n, then $\mathcal{B}(H) \cong M_n$ as C*-algebras, so the two may be often identified.

The assignment $\mathcal{B}(-)$ from Example 77 can be extended functorially by defining $\mathcal{B}(f) := f(\cdot)f^\dagger$ for a linear map $f\colon H \to K$. This allows us to lift functions acting on pure states to ones acting on mixed states.

Lemma 46. *The assignment $\mathcal{B}(-)\colon$* **FHilb** $\to \mathbf{FC}^*$ *can be equipped with the structure of a strong monoidal functor. We write $\tau_{H,K}\colon \mathcal{B}(H) \otimes \mathcal{B}(K) \cong \mathcal{B}(H \otimes K)$ and $\tau_{\mathbb{C}}\colon \mathbb{C} \cong \mathcal{B}(\mathbb{C})$ for the obvious natural isomorphisms.* □

With the model in place, we now describe the denotational semantics of $\mathcal{L}_{\mathcal{B}}$. Pure and mixed types are interpreted as objects in **FHilb** and **FC***, respectively.

$$\begin{gathered}
[\![\top]\!] := 0_{\mathbf{FHilb}}, \quad [\![\mathbf{1}]\!] := \mathbb{C}, \quad [\![\boldsymbol{P} \multimap \boldsymbol{Q}]\!] := [[\![\boldsymbol{P}]\!], [\![\boldsymbol{Q}]\!]], \quad [\![\boldsymbol{P} \mathbin{\&} \boldsymbol{Q}]\!] := [\![\boldsymbol{P}]\!] \oplus [\![\boldsymbol{Q}]\!], \\
[\![\boldsymbol{P} \otimes \boldsymbol{Q}]\!] := [\![\boldsymbol{P}]\!] \otimes [\![\boldsymbol{Q}]\!], \quad [\![0]\!] := 0_{\mathbf{FC^*}}, \quad [\![1]\!] := \mathbb{C}, \quad [\![A \multimap B]\!] := [[\![A]\!], [\![B]\!]], \\
[\![A \oplus B]\!] := [\![A]\!] \oplus [\![B]\!], \quad [\![A \otimes B]\!] := [\![A]\!] \otimes [\![B]\!], \quad [\![\mathcal{B}(\boldsymbol{P})]\!] := \mathcal{B}([\![\boldsymbol{P}]\!]).
\end{gathered}$$

A pure context $\boldsymbol{\Gamma} = \boldsymbol{x_1 : P_1, \ldots, x_n : P_n}$ is interpreted as $[\![\boldsymbol{\Gamma}]\!] := [\![\boldsymbol{P}_1]\!] \otimes [\![\boldsymbol{P}_2]\!] \otimes \cdots \otimes [\![\boldsymbol{P}_n]\!]$. Similarly, a mixed context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is interpreted as $[\![\Gamma]\!] := [\![A_1]\!] \otimes [\![A_2]\!] \otimes \cdots \otimes [\![A_n]\!]$. Pure judgements $\boldsymbol{\Gamma} \vdash \boldsymbol{m : P}$ are interpreted as morphisms $[\![\boldsymbol{\Gamma} \vdash m : P]\!]\colon [\![\boldsymbol{\Gamma}]\!] \to [\![\boldsymbol{P}]\!]$ in

$$
\begin{aligned}
[\![x : S \vdash x : S]\!] &:= id \\
[\![\Gamma \vdash M + N : S]\!] &:= [\![M]\!] + [\![N]\!] \\
[\![\Gamma \vdash \alpha \cdot M : S]\!] &:= \alpha \cdot [\![M]\!] \\
[\![\boldsymbol{\Gamma} \vdash \langle\rangle : \top]\!] &:= ! \\
[\![\Gamma, \Delta \vdash \delta_0(m) : A]\!] &:= ! \circ ([\![m]\!] \otimes id) \\
[\![\vdash * : \mathbf{1}]\!] &:= id \\
[\![\Gamma, \Delta \vdash \delta_{\mathbf{1}}(M, N) : S]\!] &:= \lambda \circ ([\![M]\!] \otimes [\![N]\!]) \\
[\![\Gamma \vdash \lambda x.M : S \multimap T]\!] &:= \Phi([\![M]\!]) \\
[\![\Gamma, \Delta \vdash M\ N : T]\!] &:= \mathrm{eval} \circ ([\![M]\!] \otimes [\![N]\!]) \\
[\![\boldsymbol{\Gamma} \vdash \langle \boldsymbol{m}, \boldsymbol{n} \rangle : \boldsymbol{P}\ \&\ \boldsymbol{Q}]\!] &:= \langle [\![\boldsymbol{m}]\!], [\![\boldsymbol{n}]\!] \rangle \\
\left[\!\!\left[\boldsymbol{\Gamma}, \boldsymbol{\Delta} \vdash \boldsymbol{\delta}^i_{\&}(\boldsymbol{m}, \boldsymbol{x.n}) : \boldsymbol{R}\right]\!\!\right] &:= [\![\boldsymbol{n}]\!] \circ \sigma \circ (\pi_i \otimes id) \circ ([\![\boldsymbol{m}]\!] \otimes id) \\
[\![\Delta \vdash inl(m) : A \oplus B]\!] &:= i_1 \circ [\![m]\!] \\
[\![\Delta \vdash inr(n) : A \oplus B]\!] &:= i_2 \circ [\![n]\!] \\
[\![\Gamma, \Delta \vdash \delta_{\oplus}(m, x.n, y.o) : C]\!] &:= [[\![n]\!], [\![o]\!]] \circ d \circ \sigma \circ ([\![m]\!] \otimes id) \\
[\![\Gamma, \Delta \vdash M \otimes N : S_1 \otimes S_2]\!] &:= [\![M]\!] \otimes [\![N]\!] \\
[\![\Gamma, \Delta \vdash \delta_{\otimes}(M, xy.N) : T]\!] &:= [\![N]\!] \circ (id \otimes [\![M]\!]) \\
[\![\vdash \mathcal{B}(t) : \mathcal{B}(\boldsymbol{P})]\!] &:= \mathcal{B}([\![t]\!]) \\
[\![\Gamma, \Delta \vdash m\ n : \mathcal{B}(\boldsymbol{Q})]\!] &:= \mathcal{B}(\mathrm{eval}) \circ \tau \circ ([\![m]\!] \otimes [\![n]\!]) \\
[\![\Gamma \vdash \tau(m) : \mathcal{B}(\boldsymbol{P} \otimes \boldsymbol{Q})]\!] &:= \tau \circ [\![m]\!] \\
[\![\Delta, x : S_2, y : S_1, \Gamma \vdash M : T]\!] &:= [\![M]\!] \circ (id \otimes \sigma \otimes id)
\end{aligned}
$$

Figure 6.4: Interpretation of judgements.

FHilb and mixed judgements $\Gamma \vdash m : A$ are interpreted as morphisms $[\![\Gamma \vdash m : A]\!] \colon [\![\Gamma]\!] \to [\![A]\!]$ in **FC***. The denotational interpretation of judgements from both fragments is given in Figure 6.4. We sometimes write $[\![M]\!]$ as a shorthand for $[\![\Gamma \vdash M : T]\!]$. For simplicity, we often suppress some of the coherent natural isomorphisms related to the monoidal structure (e.g. the α monoidal associator).

Our interpretation is sound with respect to the cut-elimination process (Theorem 23), and complete (Theorem 24) with respect to a notion of contextual equivalence (Definition 101). Since this completeness is established with respect to a different relation than the one used for soundness, the result is usually referred to as *adequacy*.

Soundness In order to prove the soundness of our denotational intepretation with respect to the reduction relation $\hookrightarrow$, we first state and prove some lemmas, which show soundness of substitution.

Lemma 47 (Substitution). *Let* $\Gamma, x : S \vdash M : T$, $\Delta \vdash N : S$, *and* $\Gamma, \Delta \vdash (N/x)M : T$ *be valid judgements. Then:*

$$[\![\Gamma, \Delta \vdash (N/x)M : T]\!] = [\![\Gamma, x : S \vdash M : T]\!] \circ (id_{[\![\Gamma]\!]} \otimes [\![\Delta \vdash N : S]\!]).$$

Proof. By induction on the derivation of $\Gamma, x : S \vdash M : T$. We consider the last rule in the derivation.

(ax) $x : S \vdash x : S$: In this case $(N/x)M = N$, $T = S$, and $\Gamma = \emptyset$.

(sum) $\Gamma, x : S \vdash M_1 + M_2 : T$: This case follows directly from the induction hypothesis and the observation that $(N/x)(M_1 + M_2) = (N/x)M_1 + (N/x)M_2$.

(prod(α)) $\Gamma, x : S \vdash \alpha \cdot M : S$: This case also follows directly from the induction hypothesis and the observation that $(N/x)\alpha \cdot M = \alpha \cdot (N/x)M$.

($\top_i^p$) $\mathbf{\Gamma \vdash \langle\rangle : \top}$: This case follows from $!_{[\![\Gamma]\!]\otimes[\![\Delta]\!]} = !_{[\![\Gamma]\!]\otimes[\![S]\!]} \circ (id \otimes [\![\Delta \vdash N : \top]\!])$.

(0_e^m) $\Gamma, \Delta \vdash \delta_0(m) : A$: In this case, first observe that $(n/x)\delta_0(m) = \delta_0((n/x)m)$. Now the claim follows from induction hypothesis applied to m and the following computation:

$$\begin{aligned}
&[\![\Gamma, \Delta \vdash \delta_0((n/x)m) : A]\!] \\
&= !_{\overline{o}\otimes[\![\Gamma]\!],A} \circ ([\![(n/x)m]\!] \otimes id) \\
&= !_{\overline{o}\otimes[\![\Gamma]\!],A} \circ ([\![m]\!] \otimes id) \circ ([\![n]\!] \otimes id)
\end{aligned}$$

(1_i) $\vdash * : \mathbf{1}$: This case is impossible, since $\Gamma, x : S \neq \emptyset$.

(1_e) $\Gamma, x : S \vdash \delta_{\mathbf{1}}(M_1, M_2) : T$: In this case we have $(N/x)\delta_{\mathbf{1}}(M_1, M_2) = \delta_{\mathbf{1}}((N/x)M_1, (N/x)M_2)$. We consider the case when x appears in M_1, the other case follows using a similar argument. Let $\Gamma = \Gamma_1, \Gamma_2$ so that $\Gamma_1, x : S \vdash M_1 : \mathbf{1}$ and $\Gamma_2 \vdash M_2 : T$. Then we have, $[\![\Gamma, \Delta \vdash \delta_{\mathbf{1}}((N/x)M_1, M_2) : T]\!] = \lambda_T \circ ([\![(N/x)M_1]\!] \otimes [\![M_2]\!])$. Applying the induction hypothesis, we have that $[\![\Gamma_1, \Delta \vdash (N/x)M_1 : \mathbf{1}]\!] = [\![\Gamma_1, x : S \vdash M_1 : \mathbf{1}]\!] \circ (id_{[\![\Gamma_1]\!]} \otimes [\![\Delta \vdash N : \mathbf{1}]\!])$. Hence we have,

$$\begin{aligned}
&[\![\Gamma, \Delta \vdash \delta_{\mathbf{1}}((N/x)M_1, M_2) : T]\!] \\
&= \lambda_T \circ (([\![\Gamma_1, x : S \vdash M_1 : \mathbf{1}]\!] \circ (id_{[\![\Gamma_1]\!]} \otimes [\![\Delta \vdash N : S]\!])) \otimes [\![M_2]\!]) \\
&= \lambda_T \circ (([\![\Gamma_1, x : S \vdash M_1 : \mathbf{1}]\!] \otimes [\![M_2]\!]) \circ (id_{[\![\Gamma_1]\!]} \otimes [\![\Delta \vdash N : S]\!] \otimes id_{[\![\Gamma_2]\!]})) \\
&= [\![\Gamma, x : S \vdash \delta_{\mathbf{1}}(M_1, M_2) : T]\!] \circ (id_{[\![\Gamma]\!]} \otimes [\![\Delta \vdash N : S]\!])
\end{aligned}$$

($\multimap_i$) $\Gamma, x : S \vdash \lambda y.M : T \multimap R$: We have $(N/x)\lambda y.M = \lambda y.(N/x)M$. We use the induction hypothesis on $(N/x)M$ to obtain the following:

$$\begin{aligned}
&[\![\Gamma, \Delta \vdash \lambda y.(N/x)M : T \multimap R]\!] \\
&= \Phi([\![\Gamma, \Delta, y : T \vdash (N/x)M : R]\!]) \\
&= \Phi([\![\Gamma, y : T, x : S \vdash M : R]\!] \circ (id_{[\![\Gamma, y:T]\!]} \otimes [\![\mathbf{\Delta \vdash N : S}]\!])) \\
&= [\![\Gamma, x : S \vdash \lambda y.m : T \multimap R]\!] \circ (id_{[\![\Gamma]\!]} \otimes [\![\Delta \vdash n : S]\!])
\end{aligned}$$

($\multimap_e$) $\Gamma, x : S \vdash M_1\ M_2 : T$: We have $(N/x)(M_1\ M_2) = (N/x)M_1\ (N/x)M_2$. We consider the case when x appears in M_1, the other case follows using a similar argument. Again we let $\Gamma = \Gamma_1, \Gamma_2$ so that $\Gamma_1, x : S \vdash M_1 : R \multimap T$, and $\Gamma_2 \vdash M_2 : R$. Let f, g denote

$[\![\Gamma_1, x : S \vdash M_1 : R \multimap T]\!]$, and $[\![\Gamma_2 \vdash M_2 : R]\!]$ respectively.

$$\begin{aligned}
&[\![\Gamma, \Delta \vdash (N/x)M_1\ M_2 : T]\!] \\
&= eval_{T,R} \circ ([\![\Gamma_1, \Delta \vdash (N/x)M_1 : R \multimap T]\!] \otimes [\![\Gamma_2 \vdash M_2 : R]\!]) \\
&= eval_{T,R} \circ ((f \circ (id_{[\![\Gamma_1]\!]} \otimes [\![\Delta \vdash N : S]\!])) \otimes g) \\
&= eval_{T,R} \circ (f \otimes g) \circ (id_{[\![\Gamma_1]\!],\Gamma_2} \otimes [\![\Delta \vdash N : S]\!]) \\
&= [\![\Gamma, x : S \vdash M_1\ M_2 : T]\!] \circ (id_\Gamma \otimes [\![\Delta \vdash N : S]\!])
\end{aligned}$$

$(\&_i^p)$ $\boldsymbol{\Gamma, x : R \vdash \langle m_1, m_2\rangle : P \& Q}$: Note that we have $\boldsymbol{(n/x)\langle m_1, m_2\rangle = \langle (n/x)m_1, (n/x)m_2\rangle}$. We use the induction hypothesis for both the terms $\boldsymbol{(n/x)m_1}, \boldsymbol{(n/x)m_2}$. We have:

$$\begin{aligned}
&[\![\boldsymbol{\Gamma \vdash (n/x)\langle m_1, m_2\rangle : P \,\&\, Q}]\!] \\
&= \langle [\![\boldsymbol{\Gamma \vdash (n/x)m_1 : P}]\!], [\![\boldsymbol{\Gamma \vdash (n/x)m_2 : Q}]\!]\rangle \\
&= \langle [\![\boldsymbol{\Gamma, x : R \vdash m_1 : P}]\!] \circ (id_\Gamma \otimes [\![\boldsymbol{\Delta \vdash n : R}]\!]), \\
&\qquad\qquad [\![\boldsymbol{\Gamma, x : R \vdash m_2 : Q}]\!] \circ (id_\Gamma \otimes [\![\boldsymbol{\Delta \vdash n : R}]\!])\rangle \\
&= \langle [\![\boldsymbol{\Gamma, x : R \vdash m_1 : P}]\!], [\![\boldsymbol{\Gamma, x : R \vdash m_2 : Q}]\!]\rangle \circ (id_\Gamma \otimes [\![\boldsymbol{\Delta \vdash n : R}]\!]) \\
&= [\![\boldsymbol{\Gamma, x : R \vdash \langle m_1, m_2\rangle : P \,\&\, Q}]\!] \circ (id_\Gamma \otimes [\![\boldsymbol{\Delta \vdash n : R}]\!])
\end{aligned}$$

$(\&_{ei}^p)$ $\boldsymbol{\Gamma, x : P' \vdash \delta^i_{\&}(m', y.n') : P}$: We present the case where $\boldsymbol{x}$ appears in $\boldsymbol{m'}$ first. Let $\boldsymbol{\Gamma = \Gamma_1, \Gamma_2}$, so that $\boldsymbol{\Gamma_1, x : P' \vdash m' : Q \,\&\, R}$ and $\boldsymbol{\Gamma_2, y : Q \vdash n' : P}$ are valid. We have:

$$\begin{aligned}
&\left[\!\!\left[\boldsymbol{\Gamma, \Delta \vdash \delta^i_{\&}((n/x)m', y.n') : P}\right]\!\!\right] \\
&= [\![\boldsymbol{n'}]\!] \circ (\pi_i \otimes id_{\boldsymbol{\Gamma_2}}) \circ ([\![\boldsymbol{(n/x)m'}]\!] \otimes id_{\boldsymbol{\Gamma_2}}) \circ (id_{[\![\boldsymbol{\Gamma_1}]\!]} \otimes \sigma_{[\![\boldsymbol{\Gamma_2}]\!],[\![\boldsymbol{\Delta}]\!]}) \\
&= [\![\boldsymbol{n'}]\!] \circ (\pi_i \otimes id_{\boldsymbol{\Gamma_2}}) \circ ([\![\boldsymbol{m'}]\!] \otimes id_{\boldsymbol{\Gamma_2}}) \circ (id_{\boldsymbol{\Gamma_1}} \otimes [\![\boldsymbol{n}]\!] \otimes id_{\boldsymbol{\Gamma_2}}) \\
&\qquad\qquad \circ (id_{[\![\boldsymbol{\Gamma_1}]\!]} \otimes \sigma_{[\![\boldsymbol{\Gamma_2}]\!],[\![\boldsymbol{\Delta}]\!]}) \\
&= [\![\boldsymbol{n'}]\!] \circ (\pi_i \otimes id_{\boldsymbol{\Gamma_2}}) \circ ([\![\boldsymbol{m'}]\!] \otimes id_{\boldsymbol{\Gamma_2}}) \circ (id_{\boldsymbol{\Gamma}} \otimes [\![\boldsymbol{n}]\!]) \\
&= \left[\!\!\left[\boldsymbol{\Gamma, x : P' \vdash \delta^i_{\&}(m', y.n') : P}\right]\!\!\right] \circ (id_{\boldsymbol{\Gamma}} \otimes [\![\boldsymbol{n}]\!])
\end{aligned}$$

For the case when $\boldsymbol{x}$ appears in $\boldsymbol{n'}$, again let $\boldsymbol{\Gamma = \Gamma_1, \Gamma_2}$ so that $\boldsymbol{\Gamma_1 \vdash m' : Q \,\&\, R}$, $\boldsymbol{\Gamma_2, \Delta \vdash (n/x)n' : P}$ are valid. We have $\boldsymbol{\Delta \vdash n : P'}$. It suffices to show that $((\pi_1 \circ [\![\boldsymbol{n}]\!]) \otimes id_{[\![\boldsymbol{\Gamma_2, P'}]\!]}) \circ (id_{[\![\boldsymbol{\Gamma}]\!]} \otimes [\![\boldsymbol{n}]\!]) = (id_{[\![\boldsymbol{Q,\Gamma_2}]\!]} \otimes [\![\boldsymbol{n}]\!]) \circ ((\pi_1 \circ [\![\boldsymbol{m'}]\!]) \otimes id_{[\![\boldsymbol{\Gamma_2,\Delta}]\!]})$. But this holds, because $\boldsymbol{m', n}$ are independent of each other, and act on disjoint contexts.

$(\oplus_{i1}^m)$ $\Gamma, x : C \vdash inl(m') : A \oplus B$: We have $(n/x)inl(m') = inl((n/x)m')$. We apply the induction hypothesis on $(n/x)m'$. Hence,

$$\begin{aligned}
&[\![\Gamma, \Delta \vdash (n/x)inl(m') : A \oplus B]\!] \\
&= i_1 \circ ([\![\Gamma, \Delta \vdash (n/x)m' : A]\!]) \\
&= i_1 \circ [\![\Gamma, x : C \vdash m' : A]\!] \circ (id_\Gamma \otimes [\![\Delta \vdash n : C]\!]) \\
&= [\![\Gamma, x : C \vdash inl(m') : A \oplus B]\!] \circ (id_\Gamma \otimes [\![\Delta \vdash n : C]\!])
\end{aligned}$$

$(\oplus_{i2}^m)$ $\Gamma, x : C \vdash inr(m') : A \oplus B$: This case follows using the same argument as that for the above case.

$(\oplus_e^m)$ $\Gamma, x : A' \vdash \delta_\oplus(m', y.n', z.o') : C$: Suppose x appears in m'. In this case $(n/x)\delta_\oplus(m', y.n', z.o') = \delta_\oplus((n/x)m', y.n', z.o')$. Let $\Gamma = \Gamma_1, \Gamma_2$ so that $\Gamma_1, x : A' \vdash m' : A \oplus B$, $\Gamma_2, y : A \vdash n' : C$ and $\Gamma_2, z : B \vdash o' : C$ are valid. Let f, g, h denote $[\![\Gamma_2, y : A \vdash n' : C]\!]$, $[\![\Gamma_2, z : B \vdash o' : C]\!]$, $[\![\Gamma_1, x : A' \vdash m' : A \oplus B]\!]$ respectively. We have:

$$\begin{aligned}
&[\![\Gamma \vdash \delta_\oplus((n/x)m', y.n', z.o' : C)]\!] \\
&= [f, g] \circ d \circ \sigma \circ ((h \circ (id \otimes [\![n]\!])) \otimes id) \\
&= [f, g] \circ d \circ \sigma \circ (h \otimes id) \circ (id \otimes [\![n]\!] \otimes id) \\
&= [f, g] \circ d \circ \sigma \circ (h \otimes id) \circ (id \otimes [\![n]\!])
\end{aligned}$$

Suppose x appears in n', o'. Then the term $(n/x)\delta_\oplus(m', y.n', z.o')$ is equal to $\delta_\oplus(m', y.(n/x)n', z.(n/x)o')$ Again let $\Gamma = \Gamma_1, \Gamma_2$ so that $\Gamma_1 \vdash m : A \oplus B$, $\Gamma_2, x : A', y : A \vdash n : C$ and $\Gamma_2, x : A', z : B \vdash o : C$ are valid. Let f, g, h denote $[\![\Gamma_2, y : A, x : A' \vdash n' : C]\!]$, $[\![\Gamma_2, z : B, x : A' \vdash o' : C]\!]$, $\Gamma_1 \vdash m' : A \oplus B$ respectively. We use the induction hypothesis on $(n/x)n', (n/x)o'$. We have:

$$\begin{aligned}
&[\![\Gamma_1, \Gamma_2, \Delta \vdash \delta_\oplus(m', y.(n/x)n', z.(n/x)o') : C]\!] \\
&= [f \circ (id \otimes [\![n]\!]), g \circ (id \otimes [\![n]\!])] \circ d \circ \sigma \circ (h \otimes id) \\
&= [f, g] \circ (id \otimes [\![n]\!]) \circ d \circ \sigma \circ (h \otimes id) \\
&= [f, g] \circ d \circ \sigma \circ (h \otimes id) \circ id \otimes [\![n]\!]
\end{aligned}$$

Note that the last equality holds because m', n are independent terms and acting on disjoint contexts, therefore the corresponding morphisms commute.

$(\otimes_i)$ $\Gamma, x : S \vdash M_1 \otimes M_2 : T_1 \otimes T_2$: Without loss of generality assume x appears in M. The case when x appears in N follows using a similar argument. In this case we have, $(N/x)(M_1 \otimes M_2) = (N/x)M_1 \otimes M_2$. Let $\Gamma = \Gamma_1, \Gamma_2$ so that $\Gamma_1, x : S \vdash M_1 : T_1$ and $\Gamma_2 \vdash M_2 : T_2$ are valid. We use the induction hypothesis for $(N/x)M_1$. Let f, f', g denote $[\![\Gamma_1, \Delta \vdash (N/x)M_1 : T_1]\!]$, $[\![\Gamma_1, x : S \vdash M_1 : T_1]\!]$, $[\![\Gamma_2 \vdash M_2 : T_2]\!]$ respectively.

$$\begin{aligned}
&[\![\Gamma_1, \Gamma_2, \Delta \vdash (N/x)M_1 \otimes M_2 : T_1 \otimes T_2]\!] \\
&= (f \otimes g) \circ (id_{\Gamma_1} \otimes \sigma_{[\![\Gamma_2]\!], [\![\Delta]\!]}) \\
&= ((f' \circ (id_{\Gamma_1} \otimes [\![N]\!])) \otimes g) \circ (id_{\Gamma_1} \otimes \sigma_{[\![\Gamma_2]\!], [\![\Delta]\!]}) \\
&= (f' \otimes g) \circ (id_{\Gamma_1} \otimes [\![N]\!] \otimes id_{\Gamma_2}) \circ (id_{\Gamma_1} \otimes \sigma_{[\![\Gamma_2]\!], [\![\Delta]\!]}) \\
&= (f' \otimes g) \circ (id_\Gamma \otimes [\![N]\!])
\end{aligned}$$

$(\otimes_e)$ $\Gamma, x : S \vdash \delta_\otimes(M', yz.N') : T$: Again there are two cases here. First we consider the case when x appears in M'. Applying the induction hypothesis on M', we have the following:

$$\begin{aligned}
&[\![\Gamma, \Delta \vdash \delta_\otimes((N/x)M', yz.N') : T]\!] \\
&= [\![N']\!] \circ (id \otimes [\![(N/x)M']\!]) \\
&= [\![N']\!] \circ (id \otimes ([\![M']\!] \circ (id \otimes [\![N]\!]))) \\
&= [\![N']\!] \circ (id \otimes [\![M']\!]) \circ (id \otimes [\![N]\!]) \\
&= [\![\delta_\otimes(M', yz.N')]\!] \circ (id \otimes [\![N]\!])
\end{aligned}$$

Now we consider the case when x appears in N'. Applying the induction hypothesis on N', we have the following:

$$\begin{aligned}&[\![\Gamma, \Delta \vdash \delta_{\otimes}((N/x)M', yz.N') : T]\!]\\ &= [\![(N/x)N']\!] \circ (id \otimes [\![M']\!])\\ &= [\![N']\!] \circ (id \otimes [\![N]\!]) \circ (id \otimes [\![M']\!])\\ &= [\![N']\!] \circ (id \otimes [\![M']\!]) \circ (id \otimes [\![N]\!])\end{aligned}$$

Note that the last equality holds because M', N are independent terms and acting on disjoint contexts, therefore the corresponding morphisms commute.

$(\mathcal{B}_i^m)$ $\vdash \mathcal{B}(\boldsymbol{m}) : \mathcal{B}(\boldsymbol{P})$: This case is impossible, since $\Gamma, x : S \neq \emptyset$.

$(\mathcal{B}(\multimap)^m)$ $\Gamma, x : A \vdash m'\ n' : \mathcal{B}(\boldsymbol{Q})$: Again we consider two cases, one when x appears in m' and the other when it appears in n'. First we consider the case when it appears in m'. Applying the induction hypothesis to m', we have the following:

$$\begin{aligned}&[\![\Gamma, \Delta \vdash (n/x)m'\ n' : \mathcal{B}(\boldsymbol{Q})]\!]\\ &= \mathcal{B}(eval) \circ \tau \circ ([\![(n/x)m']\!] \otimes [\![n']\!])\\ &= \mathcal{B}(eval) \circ \tau \circ (([\![m']\!] \circ (id \otimes [\![n]\!])) \otimes [\![n']\!])\\ &= \mathcal{B}(eval) \circ \tau \circ ([\![m']\!] \otimes [\![n']\!]) \circ (id \otimes [\![n]\!])\end{aligned}$$

Similarly, for the case when x appears in n', we have the following:

$$\begin{aligned}&[\![\Gamma, \Delta \vdash (n/x)m'\ n' : \mathcal{B}(\boldsymbol{Q})]\!]\\ &= \mathcal{B}(eval) \circ \tau \circ ([\![m']\!] \otimes [\![(n/x)n']\!])\\ &= \mathcal{B}(eval) \circ \tau \circ ([\![m']\!] \otimes ([\![n']\!] \circ (id \otimes [\![n]\!])))\\ &= \mathcal{B}(eval) \circ \tau \circ ([\![m']\!] \otimes [\![n']\!]) \circ (id \otimes [\![n]\!])\end{aligned}$$

$(\mathcal{B}(\otimes)^m)$ $\Gamma, x : A \vdash \tau(m') : B$: This case follows easily from the observation that $(n/x)\tau(m') = \tau((n/x)m')$ and the application of the induction hypothesis on m'.

(ex) $\Gamma, x : S_2, y : S_1 \vdash M : T$. Direct from the induction hypothesis.

□

Lemma 48. *The following equations hold for valid judgements:*

1. $[\![\Gamma \vdash \delta_{\mathbf{1}}(\alpha \cdot *, M) : T]\!] = [\![\Gamma \vdash \alpha \cdot M : T]\!]$
2. $[\![\Gamma \vdash (\lambda x.M)\ N : T]\!] = [\![\Gamma \vdash (N/x)M : T]\!]$
3. $\boldsymbol{[\![\Gamma \vdash \delta^i_{\&}(\langle m_1, m_2\rangle, x.n) : P]\!] = [\![\Gamma \vdash (m_i/x)n : P]\!]}$
4. $[\![\Gamma \vdash \delta_{\oplus}(inl(m), x.n, y.o) : A]\!] = [\![\Gamma \vdash (m/x)n : A]\!]$
5. $[\![\Gamma \vdash \delta_{\oplus}(inr(m), x.n, y.o) : A]\!] = [\![\Gamma \vdash (m/y)o : A]\!]$
6. $[\![\Gamma \vdash \delta_{\otimes}(M_1 \otimes M_2, xy.N) : T]\!] = [\![\Gamma \vdash (M_1/x, M_2/y)N : T]\!]$

Proof. Most cases follow from Lemma 47:

1. $$\begin{aligned}\llbracket \Gamma \vdash \delta_{\mathbf{1}}(\alpha \cdot *, M) : T \rrbracket &= \lambda_T \circ (\alpha \cdot id_I \otimes \llbracket \Gamma \vdash M : T \rrbracket) \\ &= \lambda_T \circ (id_I \otimes \alpha \cdot \llbracket \Gamma \vdash M : T \rrbracket) \\ &= \alpha \cdot \llbracket \Gamma \vdash M : T \rrbracket \\ &= \llbracket \Gamma \vdash \alpha \cdot M : T \rrbracket\end{aligned}$$

2. Let $\Gamma = \Gamma_1, \Gamma_2$ with $\Gamma_1 \vdash \lambda x.M : S \multimap T$ and $\Gamma_2 \vdash N : S$. Let f, g denote the morphisms $\llbracket \Gamma_1, x : S \vdash M : T \rrbracket$, $\llbracket \Gamma_2 \vdash N : S \rrbracket$ respectively.

$$\begin{aligned}\llbracket \Gamma \vdash (\lambda x.M)\ N : T \rrbracket &= eval_{T,S} \circ (\Phi(f) \otimes g) \\ &= \varepsilon_{\llbracket S \rrbracket} \circ (\Phi(f) \otimes id_{\llbracket S \rrbracket}) \circ (id_{\llbracket \Gamma_1 \rrbracket} \otimes g) \\ &= \varepsilon_{\llbracket S \rrbracket} \circ _ \otimes \llbracket S \rrbracket(\Phi(f)) \circ (id_{\llbracket \Gamma_1 \rrbracket} \otimes g) \\ &= f \circ (id_{\llbracket \Gamma_1 \rrbracket} \otimes g) \\ &= \llbracket \Gamma \vdash (N/x)M : T \rrbracket\end{aligned}$$

3. Let $\boldsymbol{\Gamma = \Gamma_1, \Gamma_2}$ with $\boldsymbol{\Gamma_1 \vdash \langle m_1, m_2 \rangle : Q \,\&\, R}$ and $\boldsymbol{\Gamma_2, x : Q \vdash n : P}$.

$$\begin{aligned}&\left\llbracket \boldsymbol{\Gamma \vdash \delta^i_{\&}(\langle m_i, m_2 \rangle, x.o) : P} \right\rrbracket \\ &= \llbracket \boldsymbol{\Gamma_2, x : Q \vdash n : P} \rrbracket \circ (\pi_i \otimes id) \circ (\llbracket \boldsymbol{\Gamma_1 \vdash \langle m_1, m_2 \rangle : Q \,\&\, R} \rrbracket \otimes id) \\ &= \llbracket \boldsymbol{\Gamma_2, x : Q \vdash n : P} \rrbracket \circ (\llbracket \boldsymbol{\Gamma_1 \vdash m_i : Q} \rrbracket \otimes id) \circ \sigma_{\llbracket \boldsymbol{\Gamma_1} \rrbracket, \llbracket \boldsymbol{\Gamma_2} \rrbracket} \\ &= \llbracket \boldsymbol{\Gamma \vdash (m_i/x)o : P} \rrbracket\end{aligned}$$

4. Let $\Gamma = \Gamma_1, \Gamma_2$, with $\Gamma_1 \vdash inl(m) : B \oplus C$, $\Gamma_2, x : B \vdash n : A$ and $\Gamma_2, y : C \vdash o : A$. Let f, g denote the morphisms $\llbracket \Gamma_2, x : B \vdash n : A \rrbracket$, $\llbracket \Gamma_2, y : C \vdash o : A \rrbracket$ respectively.

$$\begin{aligned}\llbracket \Gamma_1, \Gamma_2 \vdash \delta_{\oplus}(inl(m), x.n, y.o) : C \rrbracket &= [f, g] \circ d \circ \sigma \circ (\llbracket inl(m) \rrbracket \otimes id) \\ &= [f, g] \circ d \circ \sigma \circ (i_1 \otimes id) \circ (\llbracket m \rrbracket \otimes id) \\ &= [f, g] \circ d \circ (id \otimes i_1) \circ \sigma \circ (\llbracket m \rrbracket \otimes id) \\ &= f \circ \sigma \circ (\llbracket m \rrbracket \otimes id) \\ &= \llbracket \Gamma_1, \Gamma_2 \vdash (m/x)n : A \rrbracket\end{aligned}$$

5. This case is analogous to the previous one.

6. Let $\Gamma = \Gamma_1, \Delta_1, \Delta_2$ with $\Gamma_1, x : S_1, x : S_2 \vdash N : T$, $\Delta_1 \vdash M_1 : S_1$ and $\Delta_2 \vdash M_2 : S_2$.

$$\begin{aligned}&\llbracket \Gamma_1, \Delta_1, \Delta_2 \vdash \delta_{\otimes}(M_1 \otimes M_2, xy.N) : T \rrbracket \\ &= \llbracket \Gamma_1, x : S_1, x : S_2 \vdash N : T \rrbracket \circ (id \otimes \llbracket M_1 \rrbracket \otimes \llbracket M_2 \rrbracket) \\ &= \llbracket \Gamma \vdash (M_1/x, M_2/y)N : T \rrbracket\end{aligned}$$

□

Theorem 23 (Soundness). *If $M \hookrightarrow^* N$ then $\llbracket M \rrbracket = \llbracket N \rrbracket$.*

Proof. By induction on the reduction relation. First, notice that if $M \leftrightarrows N$, then $\llbracket M \rrbracket = \llbracket N \rrbracket$, by Lemma 45. Thus, we only need to show the theorem for the case $M \to N$ (see Figure 6.3).

- Rules (6.1) to (6.7), are covered by Lemma 48

- Rules (6.8) to (6.14) are direct consequence of Lemma 45
- Rule (6.15): We have $[\![\cdot \vdash \tau(\mathcal{B}(\boldsymbol{v} \otimes \boldsymbol{w}))]\!] = \tau \circ [\![\cdot \vdash \mathcal{B}(\boldsymbol{v} \otimes \boldsymbol{w})]\!] = [\![\cdot \vdash \mathcal{B}(\boldsymbol{v})]\!] \otimes [\![\cdot \vdash \mathcal{B}(w)]\!] = [\![\cdot \vdash \mathcal{B}(\boldsymbol{v}) \otimes \mathcal{B}(w)]\!]$.
- Rule (6.16): We have $[\![\Gamma \vdash \tau(\sum_{i=1}^n p_i \cdot v_{bi})]\!] = \tau \circ \sum_{i=1}^n p_i \cdot [\![v_{bi}]\!] = \sum_{i=1}^n p_i \tau \circ [\![v_{bi}]\!] = [\![\sum_{i=1}^n p_i \cdot \tau(v_{bi})]\!]$, because τ is a completely positive isomorphism.
- Rule (6.17): $$\begin{aligned}[\![\vdash \mathcal{B}(\boldsymbol{a} \cdot \boldsymbol{v_b}) : \mathcal{B}(\boldsymbol{P})]\!] &= \mathcal{B}(\boldsymbol{a} \cdot [\![\boldsymbol{v_b}]\!]) \\ &= \boldsymbol{a} \cdot [\![\boldsymbol{v_b}]\!] \circ _ \circ (\boldsymbol{a} \cdot [\![\boldsymbol{v_b}]\!])^* \\ &= \boldsymbol{a} \cdot [\![\boldsymbol{v_b}]\!] \circ _ \circ \boldsymbol{a^*} \cdot [\![\boldsymbol{v_b}]\!]^* \\ &= \boldsymbol{aa^*} \cdot [\![\boldsymbol{v_b}]\!] \circ _ \circ [\![\boldsymbol{v_b}]\!]^* \\ &= |\boldsymbol{a}|^2 \cdot \mathcal{B}(\boldsymbol{v_b}) \\ &= [\![\vdash |\boldsymbol{a}|^2 \cdot \mathcal{B}(\boldsymbol{v_b}) : \mathcal{B}(\boldsymbol{P})]\!]\end{aligned}$$
- Rule (6.18): Since the rule $\mathcal{B}_i^m$ applies only to closed terms, the terms in this rule are necessarily closed. Moreover, by Theorem 19, the first values must be lambda terms. To simplify the proof, we first consider the case $n = k = 1$ and $p_1 = q_1 = 1$, and then generalise the result.
  - Let $f = [\![\vdash \boldsymbol{\lambda x.m : P \multimap Q}]\!]$, and $g = [\![\vdash \boldsymbol{v : P}]\!]$. We have,
  $$\begin{aligned}[\![\vdash \mathcal{B}(\boldsymbol{\lambda x.m})\ \mathcal{B}(\boldsymbol{v}) : \mathcal{B}(\boldsymbol{Q})]\!] &= \mathcal{B}(eval_{[\![\boldsymbol{P}]\!],[\![\boldsymbol{Q}]\!]}) \circ \tau \circ (\mathcal{B}(f) \otimes \mathcal{B}(g)) \\ &= \mathcal{B}(eval_{[\![\boldsymbol{P}]\!],[\![\boldsymbol{Q}]\!]} \circ (f \otimes g)) \\ &= \mathcal{B}([\![\vdash \boldsymbol{(\lambda x.m)\ v : Q})]\!]\end{aligned}$$
  - In the general case, we have $M = (\sum_{i=1}^n p_i \cdot \mathcal{B}(\boldsymbol{m_i}))\ (\sum_{j=1}^k q_j \cdot \mathcal{B}(\boldsymbol{m'_j}))$ and $N = \sum_{i,j} \mathcal{B}(\boldsymbol{m_i\ m'_j})$. The equality of the denotations of these two terms follows from the case above and Lemma 45.
- The contextual rules are straightforward and we omit them.

□

Definition 100 (Elimination context). *An elimination context E is a term with exactly one free variable, denoted by $[.]$, defined by the following grammar:*

$$\begin{aligned}E &:= [.] \mid \delta_0(E) \mid \delta_{\mathbf{1}}(E, M) \mid E\ M \\ &\mid \boldsymbol{\delta^1_\&(E, x.m)} \mid \boldsymbol{\delta^2_\&(E, x.m)} \mid \delta_\oplus(E, x.m, y.n) \mid \delta_\otimes(E, xy.M)\end{aligned}$$

The substitution of $[.]$ by a term M in E is denoted by $E[M]$.

Definition 101 (Contextual equivalence). *Two terms M and N are contextually equivalent ($M \sim N$) if, for every elimination context $[.] \vdash E : T$, with $T \in \{\mathbf{1}, 1, \mathcal{B}(\mathbf{1})\}$, there exists a value V such that $E[M] \hookrightarrow^* V$ iff $E[N] \hookrightarrow^* V$.*

Theorem 24 (Adequacy). *If $[\![\vdash M : T]\!] = [\![\vdash N : T]\!]$ then $M \sim N$.*

Proof. Let $[.] \vdash E : T$, with $T \in \{\mathbf{1}, 1, \mathcal{B}(\mathbf{1})\}$. Case analysis.

- Let $T = \mathbf{1}$ or 1: By Theorem 19, we have $E[M] \hookrightarrow^* \alpha \cdot *$ and $E[N] \hookrightarrow^* \beta \cdot *$. Then, by Theorem 23 and Lemma 47,

$$[\![\vdash \alpha \cdot * : 1]\!] = [\![\vdash E[M] : 1]\!] = [\![\vdash E[N] : 1]\!] = [\![\vdash \beta \cdot * : 1]\!]$$

  Thus, $\alpha = \beta$.

- Let $T = \mathcal{B}(\mathbf{1})$: By Theorem 19, we have $E[M] \hookrightarrow^* p \cdot \mathcal{B}(*)$ and $E[N] \hookrightarrow^* q \cdot \mathcal{B}(*)$. Then, by Theorem 23 and Lemma 47,

$$\begin{aligned}[\![\vdash p \cdot \mathcal{B}(*) : \mathcal{B}(\mathbf{1})]\!] &= [\![\vdash E[M] : \mathcal{B}(\mathbf{1})]\!] \\ &= [\![\vdash E[N] : \mathcal{B}(\mathbf{1})]\!] \\ &= [\![\vdash q \cdot \mathcal{B}(*) : \mathcal{B}(\mathbf{1})]\!]\end{aligned}$$

  Thus, $p = q$.

Therefore, $M \sim N$. □

6.5 Encoding vectors and matrices

The fact that our calculus is linear in a linear algebraic sense follows as a straightforward corollary of adequacy (Theorem 24).

Corollary 1 (Linearity). *Let $x : S \vdash M : T$, $\Delta \vdash N_1 : S$, $\Delta \vdash N_2 : S$, and $\Delta \vdash N : S$ (all in the same fragment), then $(\lambda x.M)\ (\alpha \cdot N_1 + \beta \cdot N_2) \sim \alpha \cdot (\lambda x.M)\ N_1 + \beta \cdot (\lambda x.M)\ N_2$.*

Remark that Corollary 1 excludes $\mathcal{B}$, sice those terms are always closed.

Let Q be the set of closed proof terms $\boldsymbol{m}$ of $\mathbf{1} \,\&\, \mathbf{1}$, modulo the equivalence relation $\equiv$. From now on, we write **qbit** for $\mathbf{1} \,\&\, \mathbf{1}$.

Theorem 25 (One-to-one correspondence [DD24, Lemmas 3.7 and 3.8]). *The set Q forms a finite-dimensional vector space, with vector addition and scalar multiplication given by $+$ and $\cdot$, respectively.*

For every element $[\boldsymbol{m}] \in Q$, there is a vector $\underline{[\boldsymbol{m}]} \in \mathbb{C}^2$. Conversely, for any vector $\mathbf{v} \in \mathbb{C}^2$, there is a closed term $\vdash \overline{\mathbf{v}} : \boldsymbol{qbit}$. Moreover, this correspondence preserves the structure: $\underline{[\boldsymbol{m_1} + \boldsymbol{m_2}]} = \underline{[\boldsymbol{m_1}]} + \underline{[\boldsymbol{m_2}]}$ and $\underline{[\boldsymbol{a} \cdot \boldsymbol{m}]} = \boldsymbol{a} \cdot \underline{[\boldsymbol{m}]}$. □

Example 78 (Qubit encoding). A single qubit $a\,|0\rangle + b\,|1\rangle \in \mathbb{C}^2$ is encoded by the term $\langle \boldsymbol{a} \cdot *, \boldsymbol{b} \cdot * \rangle$. In particular, we define the basis vectors as: $|0\rangle := \langle *, \mathbf{0} \cdot * \rangle$ and $|1\rangle := \langle \mathbf{0} \cdot *, * \rangle$. An n-qubit is a vector in $\mathbb{C}^{2^n}$, and it is encoded as a linear combination of n-fold tensor products of 1-qubit encodings. As usual, we write $|b_1 \cdots b_n\rangle$ for the n-qubit $|b_1\rangle \otimes \cdots \otimes |b_n\rangle$. For example, the 2-qubit entangled state $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ is encoded as: $\vdash \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot |00\rangle + \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot |11\rangle : \mathbf{qbit} \otimes \mathbf{qbit}$, that is, $\vdash \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot (\langle *, \mathbf{0} \cdot * \rangle \otimes \langle *, \mathbf{0} \cdot * \rangle) + \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot (\langle \mathbf{0} \cdot *, * \rangle \otimes \langle \mathbf{0} \cdot *, * \rangle) : (\mathbf{1} \,\&\, \mathbf{1}) \otimes (\mathbf{1} \,\&\, \mathbf{1})$.

Remark 32. In [DD24], 2^n-dimensional vectors are encoded as proof terms of $\mathbf{1}^{\&2^n}$. Here we prefer to use the tensor notation, which provides finer control over term structure—in particular, allowing for a more direct encoding of constructs such as the Quantum Switch (see Example 80).

We first show how to encode 2×2 matrices (Theorem 26), and then generalise the construction to $n \times n$ matrices (Corollary 2).

Theorem 26 (2×2 **matrices [DD24, Theorem 3.10]).** *Let M be a 2×2 complex matrix. Then there exists a closed proof term $\boldsymbol{f}$ of $\mathbf{qbit} \multimap \mathbf{qbit}$ such that for any vector $\mathbf{v} \in \mathbb{C}^2$, we have: $M\mathbf{v} = \underline{\boldsymbol{f}(\overline{\mathbf{v}})}$.*

Proof. Let $M := a\,|0\rangle\langle 0| + b\,|0\rangle\langle 1| + c\,|1\rangle\langle 0| + d\,|1\rangle\langle 1|$ be a 2×2 matrix, and let $\boldsymbol{f} := \boldsymbol{\lambda x.\delta^1_{\&}(x, x_1.f_1 x_1) + \delta^2_{\&}(x, x_2.f_2 x_2)}$, where $\boldsymbol{f_1} := \boldsymbol{\lambda x.\delta_1(x, \langle a \cdot *, c \cdot *\rangle)}$ and $\boldsymbol{f_2} := \boldsymbol{\lambda x.\delta_1(x, \langle b \cdot *, d \cdot *\rangle)}$. Let $\mathbf{v} := a_1\,|0\rangle + b_1\,|1\rangle \in \mathbb{C}^2$. Notice that $\boldsymbol{f}(\overline{\mathbf{v}}) \hookrightarrow^* \boldsymbol{f_1(a_1 \cdot *) + f_2(b_1 \cdot *)} \hookrightarrow^* \langle \boldsymbol{(a_1 a + b_1 b) \cdot *, (a_1 c + b_1 d) \cdot *} \rangle$. Thus, $M\mathbf{v} = \underline{\boldsymbol{f}(\overline{\mathbf{v}})}$. □ □

Corollary 2. *Let M be a linear transformation on $\mathbb{C}^{2^n}$. Then there exists a closed term $\boldsymbol{f}$ such that: $\underline{\boldsymbol{f}(\overline{\mathbf{v}})} = M\mathbf{v}$ for all $\mathbf{v} \in \mathbb{C}^{2^n}$.*

Proof. Let $\boldsymbol{f_1}$ and $\boldsymbol{g_1}$ encode two matrices M_1 and N_1 respectively. Thethe tensor product $M_1 \otimes N_1$ can be encoded by: $\boldsymbol{\lambda z.\, \delta_{\otimes}(z, xy.f_1 x \otimes g_1 y)}$. Thus, the result follows from Theorem 26 and the fact that any linear transformation on a finite-dimensional space $V \otimes W$ can be written as a linear combination of operators of the form $f \otimes g$, with $f : V \to V$ and $g : W \to W$. □ □

Example 79 (CNOT gate). The $CNOT$ gate can be written as a linear combination of tensor products of Pauli matrices X, Z, and the identity I: $\frac{1}{2}(I \otimes I + Z \otimes I + I \otimes X - Z \otimes X)$. Let $\boldsymbol{f_x}$, $\boldsymbol{f_z}$ be the proof terms representing X and Z, respectively. Then, by Corollary 2, a proof term representing the $CNOT$ gate for proposition $\mathbf{qbit} \otimes \mathbf{qbit} \multimap \mathbf{qbit} \otimes \mathbf{qbit}$ is:
$\boldsymbol{\lambda z.\delta_{\otimes}\Big(z, xy.\frac{1}{2} \cdot (x \otimes y) + \frac{1}{2} \cdot (f_z\ x \otimes y) + \frac{1}{2} \cdot (x \otimes f_x\ y) + \frac{-1}{2} \cdot (f_z\ x \otimes f_x\ y)\Big)}$.

Example 80 (Quantum switch). The quantum switch is a higher-order construction that applies two proofs $\boldsymbol{f, g : P \multimap P}$ in opposite order depending on the value of a control qubit. It can be represented in $\mathcal{L}_{\mathcal{B}}$as the proof term:
$\boldsymbol{\lambda h.\lambda z.\delta_{\otimes}\Big(h, fg.\delta_{\otimes}(z, xy.\delta^1_{\&}(x, x_1.x_1 \otimes f(g\,y)) + \delta^2_{\&}(x, x_2.x_2 \otimes g(f\,y)))\Big)}$
proving the proposition $\boldsymbol{((P \multimap P) \otimes (P \multimap P)) \multimap (\mathbf{qbit} \otimes P) \multimap (\mathbf{qbit} \otimes P)}$.

The mixed-state fragment allows us to represent quantum measurements. Let $m\colon M_2 \to M_2$ be the measurement map

$$m(\rho) := \textstyle\sum_{i=1}^{n} P_i \rho P_i^{\dagger},$$

determined by a choice of positive elements $P_i \in M_2$ that sum to the identity, i.e. a POVM. Let $\boldsymbol{m_i}$ denote the encoding in $\mathcal{L}_{\mathcal{B}}$ of the linear map P_i. Then $\mathcal{B}(\boldsymbol{m_i})$ encodes the map $P_i(\cdot)P_i^{\dagger}$ in a way that allows us to recover its action on positive elements (which is sufficient considering our choice of morphisms). Therefore the encoding of m in $\mathcal{L}_{\mathcal{B}}$ is

$$\vdash \mathcal{B}(\boldsymbol{m_1}) + \mathcal{B}(\boldsymbol{m_2}) + \cdots + \mathcal{B}(\boldsymbol{m_n}) : \mathcal{B}(\mathbf{qbit} \multimap \mathbf{qbit}).$$

This construction naturally generalises to measurements on finitely many qubits by applying Corollary 2.

Example 81 (Measurement in the computational basis). Consider the 2×2 projection matrices $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$. Let $\boldsymbol{m_1, m_2}$ denote the encoding of these matrices in $\mathcal{L}_{\mathcal{B}}$. Then, the measurement of a single qubit in the computational basis is given by the term $\vdash \mathcal{B}(\boldsymbol{m_1}) + \mathcal{B}(\boldsymbol{m_2}) : \mathcal{B}(\mathbf{qbit} \multimap \mathbf{qbit})$. Consider, for example, the term $|+\rangle := \langle \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot *, \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot * \rangle$. Then:

$$\begin{aligned}(\mathcal{B}(\boldsymbol{m_1}) + \mathcal{B}(\boldsymbol{m_2}))\,\mathcal{B}(|+\rangle) &\hookrightarrow \mathcal{B}(\boldsymbol{m_1}\ |+\rangle) + \mathcal{B}(\boldsymbol{m_2}\ |+\rangle) \\ &\hookrightarrow^* \mathcal{B}\big(\langle \tfrac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot *, \mathbf{0} \cdot * \rangle\big) + \mathcal{B}\big(\langle \mathbf{0} \cdot *, \tfrac{\mathbf{1}}{\sqrt{\mathbf{2}}} \cdot * \rangle\big).\end{aligned}$$

This is, as expected, the representation of the density matrix $\frac{1}{2} \cdot |0\rangle\langle 0| + \frac{1}{2} \cdot |1\rangle\langle 1|$.

Example 82 (Bell measurement). The Bell basis is given by $\{\beta_{00}, \beta_{01}, \beta_{10}, \beta_{11}\}$, with $\beta_{ij} := CNOT((H\,|i\rangle)\otimes|j\rangle)$. Let $\boldsymbol{m}_{ij}$ be the encoding of the projector $|\beta_{ij}\rangle\langle\beta_{ij}|$. Then, Bell measurement is represented by:

$$\vdash \mathcal{B}(\boldsymbol{m}_{00}) + \mathcal{B}(\boldsymbol{m}_{01}) + \mathcal{B}(\boldsymbol{m}_{10}) + \mathcal{B}(\boldsymbol{m}_{11}) : \mathcal{B}(\mathbf{qbit} \otimes \mathbf{qbit} \multimap \mathbf{qbit} \otimes \mathbf{qbit})$$

Example 83 (Teleportation). Quantum teleportation transfers the state of a qubit $|\psi\rangle$ using shared entanglement and classical communication. Let $\boldsymbol{f_i}$ be the proof terms representing the unitary corrections U_i, and $\boldsymbol{h_i}$ the proof terms for the projectors P_i used in a Bell measurement. Then, consider the proof term:

$$\begin{aligned}\boldsymbol{U} :=& \mathcal{B}(\boldsymbol{\lambda z.\delta_{\otimes}(z, xy.f_1\ x \otimes h_1\ y)}) + \mathcal{B}(\boldsymbol{\lambda z.\delta_{\otimes}(z, xy.f_2\ x \otimes h_2\ y)}) + \\ & \mathcal{B}(\boldsymbol{\lambda z.\delta_{\otimes}(z, xy.f_3\ x \otimes h_3\ y)}) + \mathcal{B}(\boldsymbol{\lambda z.\delta_{\otimes}(z, xy.f_4\ x \otimes h_4\ y)})\end{aligned}$$

of the proposition $\mathcal{B}(\mathbf{qbit} \otimes (\mathbf{qbit} \otimes \mathbf{qbit})) \multimap \mathcal{B}(\mathbf{qbit} \otimes (\mathbf{qbit} \otimes \mathbf{qbit}))$. Teleportation is then encoded as the proof term for the proposition $\mathcal{B}(\mathbf{qbit}) \multimap \mathcal{B}(\mathbf{qbit} \otimes \mathbf{qbit} \otimes \mathbf{qbit})$ given by

$$\text{Telep} := \lambda z.\, \boldsymbol{U}\ (\tau(\mathcal{B}(\beta_{00}) \otimes z)).$$

Here, τ is used to encode the Bell state and the state to be teleported into a single density matrix.

6.6 Related Work

Our approach offers three major advantages over the typical approach adopted by quantum lambda calculi (QLCs) [SV06, DLPZ25, CDW19, TA24]. First, our approach enables a richer linear-algebraic structure: instead of treating unitary transformations as fixed constants, they can be defined and composed as first-class terms, allowing precise descriptions of quantum operations as linear maps. Second, it allows the expression of genuinely quantum control flows, such as the *quantum switch* [OCB12], where the order of operations is conditioned on the state of a qubit—a behaviour not efficiently representable in QLCs. Third, quantum control dispenses with the need for an external global configuration—such as the "quantum configurations" in the QLC—to carry quantum state alongside the program. In our system, both state and program are encoded in the same proof language, and measurement is not a fixed constant with special semantics, but a proof term built from the same algebraic structure as the rest of the language.

Unlike the algebraic λ-calculi [AD17, V09, ADPTV14], our language integrates both pure and mixed-state computation. The syntax accommodates superpositions and measurements within the same framework.

Finally, unlike the Lambda-S family of calculi [DCDR19, DCM20, DGMV19, DM22], which introduce a new logic based on a dedicated modality S to manage quantum superpositions, our system builds directly on the well-established logic IMALL. This ensures that the logical foundations are familiar, modular, and reusable, while still being expressive enough to represent both pure and mixed-state computation. The modality $\mathcal{B}$ embeds the pure fragment into the mixed one, allowing transitions from pure-state vectors to density matrices. This resolves a limitation of earlier approaches using the $\odot$ connective [DCD23], which were well suited to probabilistic settings but lacked a clear quantum denotational semantics [DCM24].

6.7 Conclusion

We have introduced a proof language for Intuitionistic Multiplicative Additive Linear Logic (IMALL), extended with a modality $\mathcal{B}$ to integrate both pure and mixed-state quantum computation in a unified setting. Our calculus departs from programming-centric paradigms by treating quantum computation as a proof-theoretic discipline: terms are logical proofs, and the interplay between algebraic structure and proof-theoretic structure enforces quantum constraints naturally.

The system enjoys strong meta-theoretic properties, is supported by a denotational model validating its semantics, and allows the representation of interesting quantum phenomena—as exemplified by the quantum switch and quantum teleportation—entirely within the logic. Future work includes extending the model to the full Intuitionistic Linear Logic (ILL) and clarifying the connection with Lambda-S, whose semantics is based on the same adjunction used to interpret the exponential modality of linear logic [DCM20, DM22].

7

Conclusion

In this thesis, we attempt to lay down logical and theoretical foundations addressing the question of unifying quantum and classical control flow in programs within one framework. This is reflected in the two works described here. We also address this question denotationally, by using mathematical structures well-suited to the Heisenberg view of quantum mechanics. Adopting this picture, as opposed to the Schrödinger view has a clear advantage, in that it is supported by existing mathematical literature to express both classical and quantum computation and their interaction. We hope that this inspires future work to incorporate *complete* quantum control with classical control, using the Heisenberg view of quantum mechanics and its formulation. The logical aspect of achieving this combination is achieved by defining a suitable extension of IMALL, and defining a proof language for this logic with support for both quantum superpositions and measurement. An important caveat here is that measurement is natively expressed in this proof language as a program, as opposed to a ground term.

Chapter 5 introduces a quantum lambda calculus which combines quantum and classical control, by embedding a purely quantum control fragment into a quantum lambda calculus. The design is guided by the denotational model, which embeds the categories **Unitary** and **Isometry** of Hilbert spaces into categories **NCPSU** and **NMIU** of von Neumann algebras. Operationally, we accomplish this by adapting the notion of quantum configurations first introduced in [SV06]. The quantum control fragment, supports a type for quantum natural numbers, as a prototype for more general quantum inductive types which can be easily incorporated into this subsystem. The question of further supporting recursion in the lambda calculus, remains an interesting future work.

Chapter 6 introduces a logic $\mathcal{B}-$IMALL, an extension of the Intuitionistic Multiplicative and Additive fragment of Linear Logic, to support quantum computation, along with a corresponding proof language. We define a denotational model for this language, and show adequacy. Moreover, the linearity of the operational semantics is obtained as a natural consequence of this adequacy result. We show that the language is expressive enough, by providing an encoding of quantum computation in the language. This work builds on the work introduced in Chapter 5, by using a similar modality. However, the operational semantics is greatly simplified by doing away with quantum configurations. Moreover, the syntax is reminiscent of linear algebraic constructs, with explicit support for linear combination of terms. Beyond this, an important contribution is that we are able to write measurement as a program as opposed to forcing it as a ground term construct of the syntax. An interesting future work is the exploration of interaction of this logic with the !-modality, by adding support for non-linear types.

Bibliography

[B82] Paul Benioff. Quantum mechanical Hamiltonian models of Turing machines Springer, 1982.

[S94] Peter Shor. Algorithms for quantum computation: discrete logarithms and factoring IEEE Computer Society, 1994.

[G96] Lov Grover. A fast quantum mechanical algorithm for database search ACM, 1996.

[AC04] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols IEEE, 2004.

[CD08] Bob Coecke and Ross Duncan. Interacting quantum observables Springer, 2008.

[CD08a] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics IOP Publishing, 2008.

[B14] Miriam Backens. The ZX calculus is complete for stablizer quantum mechanics IOP Publishing, 2014.

[HK71] Kenneth Hoffmann and Ray Alden Kunze. Linear algebra Prentice-Hall Hoboken, NJ, 1971.

[C10] Bob Coecke. Quantum Computer Science 2010.

[CW16] Kenta Cho and Abraham Westerbaan. Von Neumann algebras form a model for the quantum lambda calculus arXiv preprint arXiv:1603.02133, 2016.

[SV09] Peter Selinger and Benoît Valiron and others. Quantum Lambda Calculus Semantic techniques in quantum computation, 2009.

[SV06] Peter Selinger and Benoît Valiron. A lambda calculus for quantum computation with classical control Cambridge University Press, 2006.

[L14] Tom Leinster. Basic Category Theory Cambridge University Press, 2014.

[P91] Benjamin C. Pierce. Basic category theory for computer scientists MIT Press, 1991.

[HV19] Chris Heunen and Jamie Vicary. Categories for Quantum Theory: an Introduction Oxford University Press, 2019.

[NC10] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information Cambridge University Press, 2010.

[T12] Masemichi Takesaki. Theory of Operator Algebras I Springer New York, 2012.

[M89] Sidney Morris. Topology without tears University of New England, 1989.

[D85] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer The Royal Society London, 1985.

[K96] Emmanuel Knill. Conventions for Quantum pseudocode External distribution, 1996.

[O03] Bernhard Ömer. Structured Quantum Programming Technische Universität Wien, 2003.

[BCS03] Stefano Bettelli, Tommaso Calarco and Luciano Serafini. Toward an architecture for quantum programming Springer, 2003.

[SZ00] Jeff Sanders and Paolo Zuliani. Quantum programming Springer, 2000.

[S212] Shôichirô Sakai. C*-algebras and W*-algebras Springer Science & Business Media, 2012.

[S04] Peter Selinger. Towards a quantum programming language Cambridge University Press, 2004.

[PRZ17] Jennifer Paykin, Robert Rand and Steve Zdancewic. QWIRE : a core language for quantum circuits ACM, 2017.

[SVV18] Amr Sabry, Benoît Valiron and Juliana Kaizer Vizzotto. From symmetric pattern-matching to quantum control Springer, 2018.

[DLPZ25] Kinnari Dave, Louis Lemonnier, Romain Péchoux and Vladimir Zamdzhiev. Combining quantum and classical control:syntax, semantics and adequacy Springer, 2025.

[L24] Louis Lemonnier. The Semantics of Effects: Centrality, Quantum Control and Reversible Recursion arXiv preprint arXiv:2406.07216, 2024.

[K17] Andre Kornell. Quantum collections International Journal of Mathematics, 2017.i

[BW96] Nick Benton and Philip Wadler. Linear Logic, Monads and the Lambda Calculus IEEE Computer Society, 1996.

[B94] Nick Benton A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models (Extended Abstract) Springer, 1994.

[DD24] Alejandro Díaz-Caro and Gilles Dowek. A linear linear lambda-calculus Mathematical Structures in Computer Science, 2024.

[ADPTV14] Ali Assaf, Alejandro Díaz-Caro, Simon Perdrix, Christine Tasson and Benoît Valiron. Call-by-value, call-by-name and the vectorial behaviour of the algebraic λ-calculus Logical Methods in Computer Science, 2014.

[AD17] Pablo Arrighi and Gilles Dowek. Lineal: A Linear Algebraic Lambda-Calculus Logical Methods in Computer Science, 2017.

[V09] Lionel Vaux The Algebraic Lambda-Calculus Mathematical Structures in Computer Science, 2009.

[HV19] Chris Heunen and Jamie Vicary. Categories for quantum theory: an introduction Oxford University Press, 2019.

[CHK16] Bob Coecke, Chris Heunen and Aleks Kissinger. Categories of quantum and classical channels Springer, 2016.

[S05] Barry Simon. Trace ideals and their applications American Mathematical Society, 2005.

[GLT89] Jean Yves Girard, Yves Lafont and Paul Taylor. Proofs and Types Cambridge University Press, 1989.

[G87] Jean Yves Girard. Linear Logic Elsevier, 1987.

[V04] André Van Tonder. A lambda calculus for quantum computation SIAM Journal on Computing, 2004.

[V09] Lionel Vaux. The algebraic lambda calculus Cambridge University Press, 2009.

[CDW19] Pierre Clairambault, Marc De Visme and Glynn Winskel. Game semantics for quantum programming ACM New York, 2019.

[CD19] Pierre Clairambault and Marc De Visme. Full abstraction for the quantum lambda-calculus ACM New York, 2019.

[D20] Marc De Visme. Quantum Game Semantics Université de Lyon, 2020.

[PSV14] Michele Pagani, Peter Selinger and Benoît Valiron. Applying quantitative semantics to higher-order quantum computing ACM, 2014.

[SV08] Peter Selinger and Benoît Valiron. On a fully abstract model for a quantum linear functional language Elsevier, 2008.

[VLRH23] Michael Hicks, Liyi Li, Robert Rand and Finn Voichick. Qunity: A Unified Language for Quantum and Classical Computing ACM, 2023.

[PPRZ20] Romain Péchoux, Simon Perdrix, Mathys Rennela and Valdimir Zamdzhiev. Quantum Programming with Inductive Datatypes: Causality and Affine Type Theory Springer, 2020.

[JKLMZ22] Xiaodong Jia, Andre Kornell, Bert Lindenhovius , Michael W. Mislove and Vladimir Zamdzhiev. Semantics for variational Quantum programming ACM, 2022.

[TA24] Takeshi Tsukada and Kazuyuki Asada. Enriched Presheaf Model of Quantum FPC ACM 2024.

[DGMV19] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel and Benoît Valiron. Realizability in the Unitary Sphere IEEE, 2019.

[DM22] Alejandro Díaz-Caro and Octavio Malherbe. Quantum Control in the Unitary Sphere: Lambda-S1 and its Categorical Model Logical Methods in Computer Science, 2022.

[M22] Pablo Andrés-Martínez. Unbounded loops in quantum programs: categories and weak-while loops University of Edinburgh, 2022.

[DCM22] Alejandro Díaz-Caro and Octavio Malherbe. Semimodules and the (syntactically-)linear lambda calculus arXiv, 2022.

[AD17] Pablo Arrighi and Gilles Dowek. Lineal: A linear-algebraic Lambda-calculus Logical Methods in Computer Science, 2017.

[DCM20] Alejandro Díaz-Caro and Octavio Malherbe. A Categorical Construction for the Computational Definition of Vector Spaces Springer, Netherlands, 2020.

[OCB12] Nicolas Brunner, Fabio Costa and Ognyan Oreshkov. Quantum correlation with no causal order Nature Publishing Group, 2012.

[DCDR19] Alejandro Díaz-Caro, Gilles Dowek and Juan Pablo Rinaldi. Two linearities for quantum computing in the lambda calculus Elsevier, 2019.

[DCM19] Alejandro Díaz-Caro and Octavio Malherbe. A concrete categorical semantics of Lambda-S Elsevier, 2019.

[DCM24] Alejandro Díaz-Caro and Octavio Malherbe. A concrete model for a typed linear algebraic calculus Cambridge University Press, 2024.

[DCD17] Alejandro Díaz-Caro and Gilles Dowek. Typing quantum superpositions and measurement Springer, 2017.

[DCD23] Alejandro Díaz-Caro and Gilles Dowek. A new connective in natural deduction, and its application to quantum computing Elsevier, 2023.

[DCM24] Alejandro Díaz-Caro and Octavio Malherbe. The Sup Connective in IMALL: A Categorical Semantics arXiv, 2024.

[C14] Kenta Cho. Semantics for a Quantum Programming Language by Operator Algebras Open Publishing Association, 2014.

[H80] William Alvin Howard. The Formulae-as-Types Notion of Construction Academic Press, 1980.

[L80] Joachim Lambek. From Lambda Calculus to Cartesian Closed Categories Academic Press, 1980.

[SU06] Morten Heine Sørensen and Paweł Urzyczyn. Lectures on the Curry-Howard Isomorphism Elsevier, 2006.

[AD05] Pablo Arrighi and Gilles Dowek. A computational definition of the notion of vectorial space Elsevier, 2005.

[AD06] Pablo Arrighi and Gilles Dowek. Linear-algebraic lambda-calculus: higher-order, encodings and confluence arxiv, 2006.

[V07a] Lionel Vaux. On linear combinations of λ-terms Springer, 2007.

[CCDFGS03] Andrew Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann and Daniel Spielman. Exponential algorithmic speedup by a quantum walk ACM, 2003.

[GLRSV13] Alexander Green, Peter Lumsdaine, Neil Ross, Peter Selinger and Benoît Valiron. Quipper: a scalable quantum programming language ACM, 2013.

[FKRS20] Peng Fu, Kohei Kishida, Neil Ross and Peter Selinger. A tutorial introduction to quantum circuit programming in dependently typed Proto-Quipper Springer, 2020.

[BBMV20] Benjamin Bichsel, Maximilian Baader, Timon Gehr and Martin Vechev. Silq: A high-level quantum language with safe uncomputation and intuitive semantics ACM, 2020.

[OTCH20] Calista Carey, Austin Garcia, Andrew Hancock, AJ Hanus, Jake Shendenhelm, Akhilesh Tyagi and Victory Omole. Cirq: A Python framework for creating, ediitng and invoking quantum corcuits arXiv, 2020.

[WVN19] Yehuda Naveh, Rod Van Meter and Robert Wille. IBM's Qiskit tool chain: Working with and developing for real quantum computers IEEE, 2019.

[BISG18] Shahnawaz Ahmed, Vishnu Ajith, Sohaib Alam, Guillermo Alonso-Linaje, Ville Bergholm, Christian Gogolin, Izaac Josh and Maria Schuld. PennyLane: Automatic differentiation of hybrid quantum-classical computations arXiv, 2018.

[FLV24] Claudia Faggian, Gaetan Lopez and Benoît Valiron. A Rewriting theory for Quantum Lambda Calculus arXiv, 2024.

Résumé

Mots-clés: Informatique quantique, langages de programmation, logique, sémantique, théorie des catégories, systèmes de types.

Cette thèse vise principalement à établir les fondements théoriques nécessaires à la conception d'un langage de programmation complet pour l'informatique quantique. L'idée qui a inspiré toutes les contributions de cette thèse est de produire dans le cadre quantique un développement similaire à celui utilisé pour les langages de programmation et la logique dans le paradigme classique. En nous appuyant sur les travaux préparatoires de la correspondance *Curry-Howard-Lambek*, nous essayons d'approfondir les trois pierres angulaires de cette correspondance : les langages de programmation, la logique et la théorie des catégories, en nous concentrant sur la combinaison de deux paradigmes et en considérant des programmes avec des flots de contrôle quantique et classique. Nous avons essayé d'aller plus loin. Nous avons également abordé les deux visions complémentaires de la mécanique quantique, celles de Heisenberg et de Schrödinger, et adopté l'une d'elles pour définir une sémantique physiquement intuitive pour les langages dont nous discutons.

Cette thèse apporte deux contributions principales à la littérature existante sur les langages de programmation quantiques. Nous introduisons tout d'abord un lambda calcul quantique qui combine le contrôle quantique et classique, dont la conception s'appuie sur la structure mathématique de sa sémantique dénotationnelle physiquement intuitive. La sémantique opérationnelle de ce langage est basée sur une adaptation des configurations quantiques que nous introduisons, et est liée au modèle dénotationnel par un résultat d'adéquation. Le langage fournit également un prototype de types inductifs quantiques à travers le type *qnat*, un type pour les entiers naturels quantiques. Le modèle dénotationnel de ce calcul est basé sur les algèbres de von Neumann, qui sont au cœur de la formulation mathématique de la vision de Heisenberg de la mécanique quantique.

La deuxième contribution est un langage de preuve pour le fragment IMALL de la logique linéaire qui s'appuie sur certaines observations clés du calcul présenté ci-dessus. Ce langage est l'un des premiers à pouvoir prendre en charge des programmes combinant des constructions telles que le quantum switch et la mesure. À cet égard, ce travail constitue une amélioration significative par rapport au calcul précédent, car il prend en charge une version plus fine du contrôle quantique, permettant ainsi l'écriture d'algorithmes tels que le quantum switch. La syntaxe du langage est par nature algébrique et linéaire, permettant d'écrire des combinaisons linéaires de termes. Une conséquence majeure, avec la sémantique de la modalité $\mathcal{B}$ qu'il introduit, est que la mesure n'est plus une construction de base du langage. Elle peut désormais être écrite comme un programme parmi d'autres dans le langage. Enfin, la sémantique dénotationnelle de ce travail est basée sur les algèbres C* - une généralisation des algèbres de von Neumann. Le modèle dénotationnel s'avère adéquat pour la sémantique opérationnelle, dont la linéarité en est une conséquence naturelle. Enfin, nous concluons cette thèse en présentant les limites de ces travaux et les orientations possibles pour des travaux futurs.

Abstract

Keywords: Quantum Computing, Programming Languages, Logic, Semantics, Category Theory, Type systems.

The focus of this thesis is on laying the theoretical foundation for designing an all-encompassing programming language for quantum computing. The idea inspiring all the contributions of this thesis was to mirror the development of programming languages and logic done for classical computing for its quantum counterpart. Following the groundwork of the *Curry-Howard-Lambek* correspondence, we try to delve into the three cornerstones of this correspondence - programming languages, logic and category theory with a focus on combining two paradigms of control flow in programs - quantum and classical into one framework. We have tried to achieve more. We have also touched upon the two complementary view of quantum mechanics- Heisenberg and Schrödinger, and adopted one view to define a physically intuitive semantics for the languages we discuss.

There are two main contributions that this thesis makes to the existing literature on quantum programming languages. We first introduce a quantum lambda calculus which combines quantum and classical control, whose design is guided by the mathematical structure of its physically intuitive denotational semantics. The operational semantics for this language, is based on an adaptation of quantum configurations which we introduce, and is tied to the denotational model through an adequacy result. The language also provides a prototype of quantum inductive types through the type *qnat*- a type for quantum natural numbers. The denotational model for this calculus is based on von Neumann algebras- which are at the heart of the mathematical formulation of the Heisenberg view of quantum mechanics.

The second contribution is a proof language for the IMALL fragment of Linear logic which builds on some key observations from the calculus introduced above. This language is one of the first such works to be able to support programs which combine constructs such as the quantum switch with measurement. In this aspect, this work is a strict improvement on the previous calculus, since it supports a more fine grained version of quantum control, thus enabling expression of algorithms such as the well-known quantum switch. The syntax is also linear algebraic in nature, supporting linear combinations of terms. A key consequence of this, along with the semantics of the $\mathcal{B}$ modality that it introduces is that measurement is no longer a ground construct of the language. In fact, it can now be written as just another program in the language. Finally, the denotational semantics for this work is based on C*-algebras - a generalization of von Neumann algebras. The denotational model is shown to be adequate for the operational semantics, whose linearity turns out to be a natural consequence of this. Finally, we conclude this thesis with limitations of these works and possible directions for future work.