String Diagrams for Quantum Foundations, Computing and Natural Language Processing

Muhammad Hamza Waseem
Magdalen College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2024

Acknowledgements

علم کے دریا سے نکلے غوطہ زن گوہر بدست

وائے محرومی خزف چینِ لبِ ساحل ہوں میں

"From the sea of knowledge, divers emerge with pearls in hand.
Alas, O deprivation! I am but a mere collector of pebbles on the shore."

Like Iqbal, I feel like an ordinary pebble-collector at the end of my DPhil. And yet, the chance even to touch the shore of the sea of knowledge has been an extraordinary privilege. No one writes a thesis alone. This is my attempt to acknowledge the debts I owe to the people and communities who have supported my journey.

I cannot imagine writing this acknowledgements section without mentioning the profound impact Sabieh Anwar had on my academic path. About a decade ago, I was a freshman who had reluctantly chosen electrical engineering, more out of pragmatism and societal pressure than genuine passion, which for me lay in physics. As I learned about Sabieh's trajectory, I realised it was possible to pivot toward physics. In my second year of undergrad, I had the privilege of working with Sabieh on science outreach and popularisation. Later, I managed to undertake a physics thesis under his supervision, despite bureaucratic challenges at my home institute, since Sabieh was at a different university. That experience, I believe, was the defining factor that eventually led me to Oxford for a DPhil in Physics.

While looking for DPhil projects and research groups, I ran into some scepticism at Oxford, mainly because I did not hold a formal degree in physics. But Alexy Karenowska saw things differently. Having made the transition from engineering to physics herself, she offered me a place in her group. Looking back, I think I hit the jackpot. With Alexy, I found not just an excellent and supportive research supervisor, but also someone who genuinely appreciated my passion for science outreach and teaching. Throughout my DPhil, I collaborated with her on several related projects.

In Alexy's group, I was originally meant to study the quantum properties of electronic spin waves through experiments. But then the Covid-19 pandemic hit, and all the lab work became nearly impossible. To make things worse, I caught Covid myself and was ill for almost six months; this was well before vaccines were around. Travel restrictions meant I was stuck in Pakistan for over a year.

Still, I kept up with teaching and outreach remotely. Through it all, Alexy, my co-supervisor John Gregg, and my College Advisor Giles Barr were consistently encouraging. John and

Giles, especially, made sure I always had teaching opportunities at Magdalen College. That eventually led to a Lectureship, something I am incredibly grateful for. I additionally owe a lot to my students at Oxford, who constantly challenged my understanding of physics and may have taught me more than I taught them.

When lockdowns eased, I realised my heart was not really in experimental physics anymore. I felt a strong pull toward theoretical work, especially in quantum foundations. That is when I reached out to Bob Coecke. He had just left academia, but generously agreed to supervise me at Quantinuum, a quantum computing company. I have learned so much from Bob not just in physics, maths, and philosophy, but also in how to build community and collaborate. He introduced me to the world of string diagrams and helped me see the world through a relational lens.

I am truly grateful to Alexy for backing my transition completely and continuing to believe in me throughout. This thesis is the result of work guided by both Alexy and Bob, and I really could not have asked for better supervisors.

I have likewise been deeply fortunate to have wonderful mentors beyond research. The late Vicky Neale gave me thoughtful and generous feedback on my teaching philosophy and practice. I benefited a great deal from co-teaching with the late Derek Goldrei, whose tutorials were masterclasses in mathematical clarity and wit. Sian Tedaldi always had my back on the outreach front, and Soufia Siddiqi was someone I could turn to for academic, semi-academic, and even non-academic advice whenever I needed it.

Over the course of my DPhil, I got to work on a project that aimed to make quantum physics more accessible. In the process, I gained insight into inclusive teaching from Selma Coecke, Caterina Puca, and Lia Yeh. I am thankful as well to the late Ian Shipsey for supporting the project within the Department of Physics and the MPLS Division.

To all my collaborators and co-authors: thank you for everything you have taught me. I have also had many inspiring conversations with colleagues and friends. I am probably forgetting some names, but I especially want to thank Johannes Fankhauser, Maria Violaris, Nicetu Tibau Vidal, Lia Yeh, Sadia Naeem, and Robin Lorenz. I am equally appreciative of my research group colleagues, Finlay Ryburn, Sally Lord, and Will Henderson. My work thrived in the intellectually rich and welcoming environment of the Quantinuum office in Oxford. Every visit left me with fresh ideas and motivation.

My sincere appreciation to Sean Tull, Harny Wang, Jonathon Liu, and Vincent Wang-Maścianica for reading parts of this thesis and offering helpful feedback. I am especially indebted to Anna Pearson for reading the whole thing and offering thoughtful, constructive critique throughout.

I was fortunate to be financially supported by the Rhodes Scholarship for the first three years of my DPhil. I am grateful to the Rhodes Trust and the selection committee for believing in me.

The Rhodes community gave me so much: space for reflection, learning, and friendship. Special thanks to the Rhodes House staff, especially Mary Eaton (Registrar) and Maureen

Freed (Scholar Wellbeing Adviser). To my fellow Rhodes Scholars and friends—Khansa Maria, Arslan Chaudhry, Sana Naeem, Siobhan Tobin, Namrata Ramesh, Terence Tsui, Krishnendu Ray, and Suhas Mahesh—thank you for the deep, thoughtful, and often heartwarming conversations. You have all left a mark on me.

Magdalen College supported me in all kinds of ways: great opportunities for teaching and outreach, amazing library access, and the ever-welcoming Dining Hall and Old Kitchen Bar. I feel honoured to have been part of this college, which also helped finance the latter years of my DPhil.

On a personal note, I have been blessed with the steady support of friends throughout this journey.

Mohsin Javed has been a constant through all my highs and lows in Oxford. His advice and insight, both intellectual and emotional, have helped me in countless ways. I am also thankful to Sarah Salim, Amal Javed, and Zaha Javed for their enduring friendship and love.

Abdul Afzal encouraged me to dig into philosophy and the foundations of quantum physics. Much of my philosophical education started with his book recommendations. Our wide-ranging discussions on consciousness, politics, love, and friendship were always a source of joy and insight.

My friendship with Ayesha Ali deeply enriched my perspective on life and relationships, and truly transformed me as a person. I cannot thank her enough for seeing light and goodness in me, especially at times when I struggled to see them myself.

Hassaan Saleem has also been a great friend: always encouraging, occasionally irritating, and often hilarious. I am glad we have stayed connected despite being on opposite sides of the Atlantic.

Mahdi Murtaza made the latter half of my time at Magdalen especially fun. Thanks for dragging me to formal dinners and Magdalen Iftars, and for showing up with home-cooked food.

I am truly thankful to Duaa Jamshaid for her kindness, her encouragement, our many conversations on physics and society, and for always cheering me on. I am beyond grateful for your support and friendship.

Over these years, I was fortunate to spend time with friends in Pakistan—Khadija Maryam, Shahryar Khan, Faizan-e-Ilahi, Tooba Malik, Shumail Hassan, Waqar Ali, and Mohsina Asif—and those in Oxford: Rishika Sahgal, Ninad Rajgopal, Bhavna Verma, Vikaran Khanna, and Shai de Vries. Thank you for looking after me and being such a meaningful part of this journey.

And of course, my friends from the Khwarizmi Science Society, Salman Mahmood Qazi, Nimra Khurram, and Charisma Wafee, thank you for keeping me in the loop on all things science, science communication, and occasionally science miscommunication during my visits back to Pakistan.

But perhaps the greatest privilege of all has been having a family that is unconditionally behind me in my quest for knowledge. Without the prayers and sacrifices of my mother, Saima, and my father, Waseem, and their constant encouragement, I would not have achieved

anything worthwhile. My brothers, Hammad and Haider, have celebrated my journey every step of the way. My Dadi and Phupho have always been a source of strength and love. And I am sure my late Dada would have been proud to see this chapter of my life.

The city of Oxford has been a wellspring of dreams, ideas, and inspiration. These years have been the happiest of my life so far, and I cannot leave without leaving a piece of my heart behind. Oxford has truly become my second home. In the words of Ghalib, I say to this city:

کم نہیں جلوہ گری میں، ترے کوچے سے بہشت

یہی نقشہ ہے ولے اس قدر آباد نہیں

"Not less in splendour is Paradise than thy street.
This very design exists, indeed, yet not so flourishing."

Abstract

Applied category theory provides powerful mathematical tools for modelling processes and their composition. Symmetric monoidal categories, which involve series and parallel composition, are particularly well-suited for describing the composition of processes in space and time. Also called process theories, they admit string diagrams, which constitute a visually intuitive, mathematically rigorous, expressive and flexible syntax that is applicable to wide-ranging scientific domains.

In this thesis, we employ string diagrams to investigate a selection of topics in the areas of quantum foundations, computing, and natural language processing. We report three main contributions:

- We formalise constructor theory as a process theory. In the context of quantum physics, we also demonstrate the conflict between constructor-theoretic principles of locality and composition. Moreover, we argue that if the principle of locality is rejected, categorical quantum mechanics (CQM) can be conceived as a constructor theory of quantum physics.

- We develop a formalism for wave-based logic circuits with phase encoding. We motivate the formalism using the example of spin-wave circuits, and then demonstrate its utility in design, analysis and optimisation of Boolean logic circuits.

- We investigate the elimination of inter-language grammatical bureaucracy in the distributional compositional circuits (DisCoCirc) framework. In particular, we develop a hybrid grammar for a restricted fragment of the Urdu language, and show that Urdu text endowed with this hybrid grammar maps surjectively to DisCoCirc text circuits. Furthermore, we show that for the same language fragment, Urdu and English text circuits become the same up to gate-level translation.

The aforementioned work supports the view that a process-relational outlook in science is well-supported by applied category-theoretic tools, particularly string diagrams.

Contents

1 Introduction

'... the aim of science is not things themselves... but the relations between things; outside those relations there is no reality knowable.'

Henri Poincaré,
Science and Hypothesis [1, p. xxiv]

Relationalism and process ontology

Mainstream Western science subscribes to the metaphysical doctrine that reality is fundamentally a collection of static objects whose dynamic aspects are ontologically derivative [2]. The origins of this worldview can be traced back to the pre-Socratics, especially Democritus and Parmenides.

According to Democritus, things are *essentially* clusters of indivisible, eternal particles called 'atoms' [3]. Democritus's doctrine is still reflected today in different forms of reductionism. Some of its examples are reducing physical phenomena to elementary particles; dividing biological systems into organs, tissues, and cells; and describing mathematical structures using set-theoretic foundations.

Democritus's atomism was a response to—or, more precisely, a refinement of—Parmenides's teaching [3] according to which reality is static; any change or motion in it is illusory [4]. Democritus taught that the properties of things are determined by how their constituents (*i.e.* atoms) are arranged. While the atoms are unchanging, their spatial rearrangements result in changing the appearance and properties of things. In this sense, all change is essentially the rearrangement of atoms in space. At any moment, a particular atomic arrangement constitutes the *now*. Stacking these *now*s in a sequence yields (the illusion of) temporal change.

It is worth noting that theories in physics more or less follow the aforementioned conceptual paradigm. First, they describe the state of a physical system at a particular instant; then, they track its time-evolution. The former is *kinematics*, and the latter is *dynamics*. A *state* is a particular arrangement of all the constituent entities that make the system. A *dynamical law*[1] characterises how the states are stacked up in time.

According to Heraclitus, a contemporary of Parmenides, everything in nature undergoes a constant flux. Reality is essentially composed of processes instead of static things [5]. This is in stark contrast to the stasis of Parmenides. Heraclitus is considered the founding father of process ontology in the Western philosophical tradition.

In modern philosophy, Leibniz's relational approach presents a perspective in which the relations between things in space and time are considered more fundamental than the spatio-temporal properties of things [6]. Heraclitus's process ontology and Leibniz's relationalism inspired Whitehead's research programme [7], the aim of which was to describe all natural phenomena in terms of processes and their relations, rather than objects or entities that are independent of one another [8]. Schrödinger refined this approach by pointing out that some relationships give rise to novel phenomena that cannot be strictly reduced to the properties of relata [9].

For over 2000 years, developments in science have been underpinned by a worldview that is Parmenidean and Democritean in spirit. Even though this approach has been remarkably successful, quantum physics has exposed its limitations. In particular, it has long been well known that it is hard to reconcile the following two aspects of quantum theory with the prevailing outlook: (1) the existence of two distinct dynamical laws that depend on the presence or absence of interaction (*i.e.* the measurement problem); and (2) the impossibility to reduce all phenomena to the properties of individual entities (*i.e.* entanglement). Merely philosophical problems for around a century, the advent of quantum computing and quantum technologies has breathed new life into these issues. Therefore, the time is ripe to give a fair chance to a process-relational view of reality as advocated by Whitehead and Schrödinger; and see how far it can take us. The mathematical foundation of this research area is provided by a relatively new branch of mathematics called applied category theory.

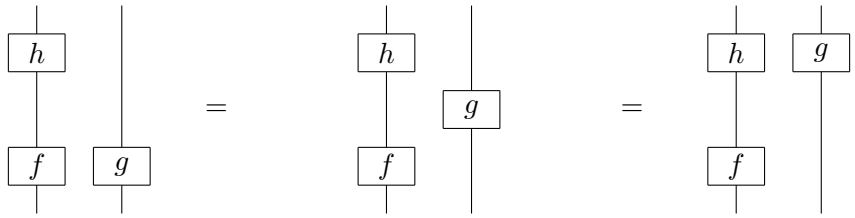
[1] Quantum mechanics does not strictly fit into this paradigm precisely because it has two dynamical laws. This is usually called the *measurement problem* of quantum mechanics.

Applied category theory provides effective mathematical tools to model processes and how they compose [10]. Symmetric monoidal categories, which involve series and parallel composition, are especially relevant in the context of process-relational science [11], being particularly well-suited for describing the composition of processes in space and time. Very aptly called process theories, they admit string diagrams [12–14], a syntax that is not only visually intuitive but also mathematically rigorous.

What are string diagrams?

String diagrams are two-dimensional graphical representations of morphisms (processes) in symmetric monoidal categories [14, 15]. They are a rigorous, intuitive and completely diagrammatic alternative to symbolic algebra that capture the compositionality of processes (denoted by boxes) between systems (denoted by wires). They are a very expressive language—flexible and versatile enough to be used at different levels of abstraction.

As a syntax, two-dimensional string diagrams incorporate the topology of one-dimensional symbolic expressions, thereby turning abstract and usually complex algebraic transformations into simple, fun, and intuitive diagram rewrite rules. An example diagrammatic calculation of 'sliding boxes' is given below.

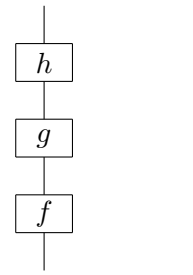


$$(h \otimes id) \circ (f \otimes g) \;=\; (h \otimes id) \circ (id \otimes g) \circ (f \otimes id) \;=\; (h \otimes g) \circ (f \otimes id)$$

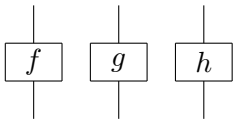
The diagrams in this chapter must be read from bottom to top and left to right. The algebraic expression corresponding to each diagram is written below it.

When moving from a symbolic algebraic syntax to string diagrams, some essential and non-trivial equations between algebraic expressions become redundant. In other words, these equations come for free in the string-diagrammatic universe and, therefore, there is no need to explicitly impose them. For instance, associativity, $(h \circ g) \circ f = h \circ (g \circ f)$ and $(f \otimes g) \otimes h = f \otimes (g \otimes h)$, is implicit in string diagrams. The algebraic expressions on both

sides of each equation map to the same diagram, resulting in a tautology.

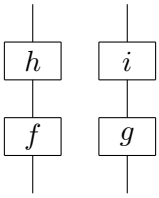


$$(h \circ g) \circ f = h \circ (g \circ f)$$



$$(f \otimes g) \otimes h = f \otimes (g \otimes h)$$

Another equation that becomes redundant in string diagrams is the *(middle-two-)interchange law*: $(h \otimes i) \circ (f \otimes g) = (h \circ f) \otimes (i \circ g)$. Both left- and right-hand sides map to the following diagram.



In short, string diagrams not only provide simpler and more intuitive counterparts to algebraic calculations, but also phase out some algebraic equations altogether.

String diagrams for the sciences

A precursor to string diagrams appeared in the work of Richard Feynman in 1949 [16, 17]. String-diagrammatic notation was invented by Roger Penrose [12] in order to keep track of indices of tensors. Günter Hotz independently invented string diagrams in a categorical setting to model series and parallel composition of digital circuits [18]. Hotz's thesis [18] appeared earlier than Penrose's paper [12], but Penrose was reportedly already using string diagrams during his graduate studies in the late 1950s [19]. The connection between categories and string diagrams was formalised by André Joyal and Ross Street in the 1990s [13]. Since then, they have been applied in a wide range of scientific domains.

Particularly in the last two decades, string diagrams have been used as a syntactic foundation for wide-ranging research domains of science, engineering, and technology [15]. For example: quantum physics [20, 21], quantum computing [22–24], electrical circuit theory [25], digital circuits [26, 27], photonics [28, 29], linguistics [30–32], control systems [33, 34], causal models [35], probability theory [36, 37], machine learning [38, 39], game theory [40–42] etc. Moreover, the string diagrammatic approach to quantum theory,

dubbed *Quantum Picturalism* [43], has found success in education both at university [22] and high school [23] levels. In fact, the results of a pedagogical study [44] demonstrate that the Quantum Picturalism approach is particularly effective in teaching university-level quantum theory to high school students [45].

To sum up, a process-relational outlook goes hand in hand with a mathematical language that is applicable to wide-ranging scientific areas. Additionally, by replacing symbolic reasoning with diagrammatic reasoning, this language makes science much more intuitive and accessible.

This thesis

This thesis is an exercise in *applied* process-relational philosophy with string diagrams as the tool of choice. It aims to add to the growing body of evidence supporting the view that there is a great deal of mileage in using string diagrams as the standard mathematical formalism in the sciences. Reported here are three strands of research, revolving around the following questions.

- The relatively new research area of constructor theory [46] bears family resemblance with the now well-established area of process theories. How are these two programmes related? Are the principles of constructor theory consistent with one another? Do these principles agree with the physical theories that they aim to characterise? Can quantum physics be formulated as a constructor theory? Can categorical quantum mechanics (CQM)—a full-blown process-theoretic formulation of quantum theory—be conceived as a constructor theory?

- Wave-based computation involves encoding information in some property of waves and performing computation via wave interference [47]. How can string diagrams be used to develop a formalism for this computing paradigm? How can this formalism be used for design, analysis and optimisation of logic circuits in wave-based technologies?

- Distributional compositional circuits (DisCoCirc) is a framework for modelling the grammar and meaning of natural languages in process-theoretic terms [31]. For a particular language, how can an arbitrary piece of text be mapped to DisCoCirc text

circuits? Does DisCoCirc eliminate grammatical differences between different natural languages?

It is hoped that this thesis makes some progress towards answering—or at least elucidating the nuances of—the above questions.

1.1 Main Results

The main contributions to the field described in this thesis are as follows. Except where explicitly mentioned, all the work is solely that of the author.

- Section 3.2 presents constructor theory as a process theory. It shows that if the theory of conceivable tasks is taken to be **Rel**, then for a given choice of substrates, the set of possible tasks forms a sub-SMC of **Rel**.[2]

- Section 3.3 gives an overview of the principle of locality and how it is instantiated in the Deutsch-Hayden formulation of quantum theory [49]. Then it uses the Deutsch-Hayden approach to demonstrate the conflict between the principles of locality and composition.

- Section 3.4 discusses the overlap between constructor theory and process theories with reference to quantum physics. It argues that if the principle of locality is rejected, CQM can be conceived as a constructor theory of quantum physics. This argument is supported by multiple examples of constructor-theoretic ideas in quantum physics and computation.

- Section 4.3 develops a string-diagrammatic formalism for representing and reasoning with wave-based logic circuits that use phase encoding. Using spin-wave systems as an example of wave-based computation platform, Section 4.4 shows how to apply this formalism for design, analysis, and optimisation of Boolean logic circuits.[3]

[2] This work is in collaboration with Stefano Gogioso, Vincent Wang-Maścianica, Carlo Maria Scandolo, and Bob Coecke. It was originally published as a part of the article *Constructor Theory as Process Theory* [48]. The author of this thesis is the third author of that publication. The specific contributions made by the author to this work are detailed in this thesis.

[3] This work, in collaboration with Alexy Karenowska, was originally published in the article *Wave-based Computation with String Diagrams* [50]. The author of this thesis is the first author of the publication and played a leading role in all aspects of the work.

- Section 5.5 develops a hybrid grammar for a restricted fragment of Urdu language.[4]
- Section 5.6 describes text diagrams and text circuits for Urdu, and shows that there exists a surjective map from the set of all Urdu text endowed with hybrid grammar to the set of all Urdu text circuits. Moreover, for the restricted language fragment considered in this thesis, a clear isomorphism between the hybrid grammars for English (developed in [32]) and Urdu is demonstrated. Using this isomorpism, it is shown that the text circuits for English and Urdu become the same, up to translating the labels on the gates.[5]

Apart from the aforementioned contributions and publications, the author of this thesis co-authored the following two papers and a book as well. These publications are:

- Quantum Picturalism: Learning Quantum Theory in High School [44]
- Making the Quantum World Accessible to Young Learners through Quantum Picturalism: An Experimental Study [45]
- Quantum Mechanics in the Single-Photon Laboratory [52] (and its second edition [53])

[4]This work is in collaboration with Vincent Wang-Maścianica, Jonathon Liu, and Bob Coecke. It first appeared in the article *Language-independence of DisCoCirc's Text Circuits: English and Urdu* [51]. The author of this thesis is the first author of the publication and played a leading role in all aspects of the work.

[5]This work is in collaboration with Vincent Wang-Maścianica, Jonathon Liu, and Bob Coecke. It first appeared in the article *Language-independence of DisCoCirc's Text Circuits: English and Urdu* [51]. The author of this thesis is the first author of the publication and played a leading role in all aspects of the work.

2

Quantum Theory from Processes and Composition

'I would like to make a confession which may seem immoral:
I do not believe absolutely in Hilbert space any more.'

John von Neumann,
letter to Garrett Birkhoff, 1935 [54, p. 1]

This chapter provides the necessary background for the thesis, beginning with process theories and string diagrams, and leading into categorical quantum mechanics.

2.1 Introduction

Categorical quantum mechanics (CQM) is a process-theoretic framework to represent, characterise and reason about quantum processes, their composition and their interaction [22, 55]. It was introduced in 2004 [55]. There are a number of ways to think of CQM. Initially, it was conceived as a high-level language [22] on top of Hilbert space [56], analogous to a high-level programming language, with the matrices of Hilbert space being the analogue to machine code. Alternatively, it can be seen as a new quantum formalism [57], as desired by von Neumann [54, 58], but one that takes compositionality as its first-class citizen—something that had been advocated for by Schrödinger in 1935 [9, 59]. There is also a reconstruction theorem supporting this view [57]. Finally, thanks to an equivalence between diagrams and categories [12–14], CQM can be seen as an intuitive graphical substitute for the Hilbert space formalism [24, 43, 60] that transparently reveals new aspects of quantum theory that are deeply buried in the traditional formalism. Since diagrams, foregrounding

the composition and interaction of quantum processes are at the heart of CQM, it is also called Quantum Picturalism (QP) [43, 44].

This *diagrammatic turn* in quantum physics has become very successful in quantum technologies. For example, it has bridged research disciplines like quantum computation and computational linguistics [61–64], and led to advances in quantum circuit optimisation [65, 66] and compilation [67, 68], quantum error correction [69, 70], quantum natural language processing [62], quantum machine learning [71, 72], measurement-based quantum computing [73, 74] and optical quantum computing [28, 29].

Within the education sector, the QP approach has been used to teach a university level postgraduate course at the University of Oxford for the last 10+ years [22]. More recently, with the publication of the book *Quantum in Pictures* [23], the QP approach has made the teaching and learning of quantum theory accessible to a wider audience with very few mathematical prerequisites. A recent educational experiment [44] showed that quantum theory can be successfully taught to high schoolers (*i.e.* students aged 16-18) using the QP approach.[1]

The aim of this chapter is to provide background material for the remainder of the thesis. It serves as a self-contained introduction to CQM.[2] Since CQM is a process theory, we begin by introducing process theories and string diagrams before presenting the key components of CQM. This material lays the foundation for the chapters that follow. This chapter is organised as follows. Section 2.2 introduces process theories from both category-theoretic and string-diagrammatic perspectives. Sections 2.3, 2.4 and 2.5 provide a gentle introduction to CQM.

2.2 Process Theory

Definition 2.2.1 (Definition 3.1 in [22])**.** A *process theory* is defined to consist of:

- a collection of systems (or system-types, or interfaces) S represented by wires,
- a collection of processes P represented by boxes, and where each process in P has a number of input wires and a number of output wires taken from S, and

[1] 54 randomly selected students from the UK attended two-hour online classes, along with tutorials, weekly for eight weeks, and were then assessed using Oxford postgraduate quantum physics exam questions. The results showed that over 80% students passed, with about half earning a distinction. See Ref. [45] for more details.

[2] For a more detailed exposition, the reader is referred to the textbook [22]; most of the definitions and explanations in this chapter are based on that text.

- a means of composing processes, represented by wiring boxes together, and the result of doing so should again be a process in P.

In other words, a process theory is a collection of processes that can be composed in a meaningful sense; *i.e.* the composition of processes in the process theory should be a process in the same theory.

From a category-theoretic perspective, a process theory is a *strict symmetric monoidal category*. Essentially, this is a symbolic way to axiomatise series composition $\circ$ and parallel composition $\otimes$. First we define a strict monoidal category.

Definition 2.2.2 (Definition 3.45 in [22])**.** A *strict monoidal category* $\mathcal{C}$ consists of:

- a collection ob($\mathcal{C}$) of *objects*
- for every pair of objects A, B, a set $\mathcal{C}(A, B)$ of *morphisms*
- for every object A, a special identity morphism: $1_A \in \mathcal{C}(A, A)$
- a sequential composition operation for morphisms: $\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$
- a parallel composition operation for objects: $\otimes : \text{ob}(\mathcal{C}) \times \text{ob}(\mathcal{C}) \to \text{ob}(\mathcal{C})$
- a unit object: $I \in \text{ob}(\mathcal{C})$
- and a parallel composition operation for morphisms:

$$\otimes : \mathcal{C}(A, B) \times \mathcal{C}(C, D) \to \mathcal{C}(A \otimes C, B \otimes D)$$

satisfying the following conditions:

- $\otimes$ is associative and unital on objects: $(A \otimes B) \otimes C = A \otimes (B \otimes C); A \otimes I = A = I \otimes A$
- $\otimes$ is associative and unital on morphisms: $(f \otimes g) \otimes h = f \otimes (g \otimes h); f \otimes 1_I = f = 1_I \otimes f$
- $\circ$ is associative and unital on morphisms: $(h \circ g) \circ f = h \circ (g \circ f); 1_B \circ f = f = f \circ 1_A$
- $\otimes$ and $\circ$ satisfy the *interchange law*: $(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2)$

Compare the above definition with the definition of process theory (Definition 2.2.1). *Types* or *system-types* in process theory correspond to objects in category theory; *processes* correspond to *morphisms*. The set $\mathcal{C}(A, B)$ therefore should be considered the set of all processes with input type A and output type B. A member of this set can be denoted by either $f \in \mathcal{C}(A, B)$ or $f : A \to B$. A collection of such sets with sequential composition gives a *category*. A *monoidal category* is a category with both sequential and parallel composition. The 'strict' part of the above definition will be explained shortly. Adding a swap morphism to a monoidal category makes it symmetric.

Definition 2.2.3 (Definition 3.46 in [22])**.** A *strict symmetric monoidal category* is a strict monoidal category with a *swap morphism*: $\sigma_{A,B} : A \otimes B \to B \otimes A$ defined for all objects A, B, satisfying:

$$\sigma_{B,A} \circ \sigma_{A,B} = 1_{A \otimes B} \qquad \sigma_{A,I} = 1_A$$

$$(f \otimes g) \circ \sigma_{A,B} = \sigma_{B',A'} \circ (g \otimes f)$$

$$(1_B \otimes \sigma_{A,C}) \circ (\sigma_{A,B} \otimes 1_C) = \sigma_{A,B \otimes C}$$

A monoidal category is called strict when $\otimes$ is associative and unital exactly. On the other hand, in the non-strict monoidal category, associativity and unitality of $\otimes$ are isomorphisms rather than identities.

Definition 2.2.4 (Definition 3.47 in [22])**.** An object A is *isomorphic* to an object B, denoted $A \cong B$ if and only if there exists a pair of morphisms:

$$f : A \to B \qquad f^{-1} : B \to A$$

such that:

$$f^{-1} \circ f = 1_A \qquad f \circ f^{-1} = 1_B$$

The morphism f is then called an *isomorphism*.

Non-strict monoidal categories are the more common species of monoidal categories. Their unitality and associativity of parallel composition are isomorphisms rather than identities:

$$(A \otimes B) \otimes C \cong A \otimes (B \otimes C) \qquad A \otimes I \cong A \cong I \otimes A$$

These are called *structural isomorphisms* and must be included in the definition of a category in order to define non-strict monoidal category. Additionally, a number of equations called *coherence equations* must be added to the definition to ensure that the behaviour of structural isomorphisms is sensible when they are composed. The following theorem gives the relation between strict and non-strict (symmetric) monoidal categories.

Theorem 2.2.5 (Coherence theorem; Theorem 3.48 in [22])**.** *Every (symmetric) monoidal category $\mathcal{C}$ is equivalent to a strict (symmetric) monoidal category $\mathcal{C}'$.*

Equivalence between two categories mean that they are the same for all practical purposes. The above theorem justifies considering $(A \otimes B) \otimes C$, $A \otimes (B \otimes C)$ and $A \otimes B \otimes C$ the same. In other words, we do not need to worry about parentheses in associativity of $\otimes$. Likewise, the theorem justifies considering $A \otimes I$, A and $I \otimes A$ the same.

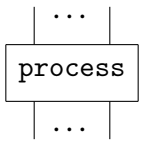
Definition 2.2.6. String diagrams are two-dimensional graphical representations of morphisms (processes) in symmetric monoidal categories.[3]

The following theorem justifies using string diagrams for symmetric monoidal categories.

Theorem 2.2.7 (Theorem 3.49 in [22])**.** *[String] diagrams are sound and complete for symmetric monoidal categories. That is, two morphisms f and g are provably equal using the equations of a symmetric monoidal category if and only if they can be expressed as the same [string] diagram.*

2.2.1 Processes and Composition

Definition 2.2.8. A *process* can be defined as anything that has zero or more inputs and zero or more outputs. A process is diagrammatically represented as a box with wires.



[3] In the literature, the term *string diagrams* is used for various kinds of diagrams corresponding to different notions of monoidal categories with some additional structure [14, 15]. For instance, in the book [22], the term *circuit diagrams* is used for symmetric monoidal categories to disambiguate from the term *string diagrams*, reserved for symmetric monoidal categories with some additional structure. In this thesis, to avoid potential confusion and to maintain consistency, we use the term *string diagrams* for symmetric monoidal categories. Specifically, we adopt the following correspondence:

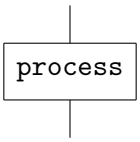
string diagrams $\leftrightarrow$ process theory $\leftrightarrow$ symmetric monoidal category

When needed, we shall introduce additional structure to the string diagrams.

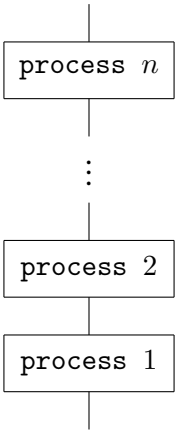
The wires represent the input and output systems or system-types. The input types are below the box while the output types are above the box.

Convention 2.2.9. Throughout this thesis—except in Chapter 5—all diagrams should be read from bottom to top and left to right.

Example 2.2.10. A single-input, single-output process can be represented as

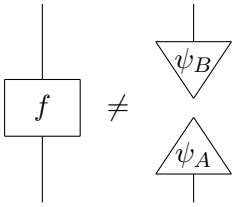


Composition is an important part of process theory. Once there are two or more processes, how are they to be composed? Diagrammatically, the series composition of n single-input, single-output processes is given by



This kind of composition usually corresponds to composition in time.

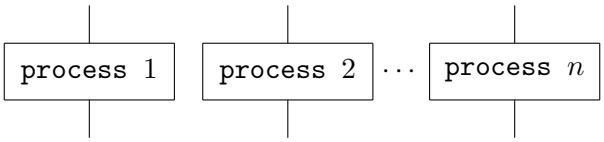
For a process theory to be non-trivial, there must be at least one process f that is non-separable (in time):



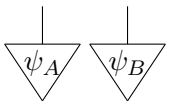
This equation means that the single-input, single-output process f is not equal to a series composition of single-input, zero-output (ψ_A) and zero-input, single-output (ψ_B) processes. In other words, the whole f is not completely describable in terms of its parts composed in series. This is called a *non-trivial series composition*.

If this condition is not satisfied by a process theory, there can be no interesting behaviour in it since each process gives the same output regardless of the input.

The parallel composition of n single-input, single-output processes is given by



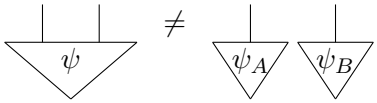
This kind of composition usually corresponds to composition in space. A simpler example of parallel composition would be that of two zero-input, single-output processes



These two processes could, for example, be wavefunctions of two quantum systems. Following Schrödinger's observation that entanglement is '*the* characteristic trait of quantum mechanics' [9],

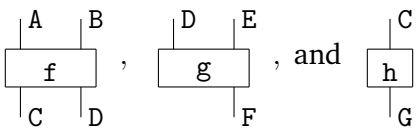
$$|\psi_{AB}\rangle \neq |\psi_A\rangle \otimes |\psi_B\rangle\,,$$

we get a *non-trivial parallel composition*: a situation in which the whole cannot be completely described in terms of the parts composed in parallel.

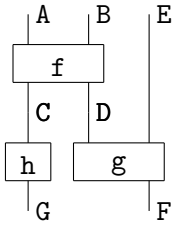


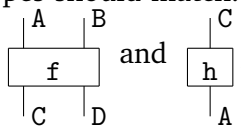
2.2.2 Diagrams

As described earlier, a process is diagrammatically represented by a box with wires. Composition of these boxes results in bigger diagrams. To put it differently, diagrams comprise boxes and wires, labelled by processes and system-types respectively [22]. For instance, the three processes

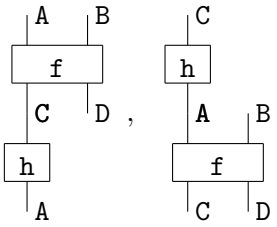


can be composed to form the diagram

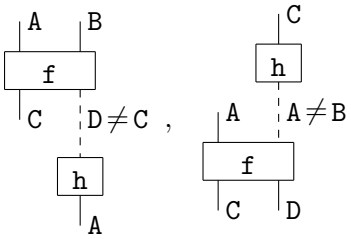


When composing processes, system-types should match. Otherwise, the diagrams are not defined. For example, for the processes 

and , the following diagrams are defined:

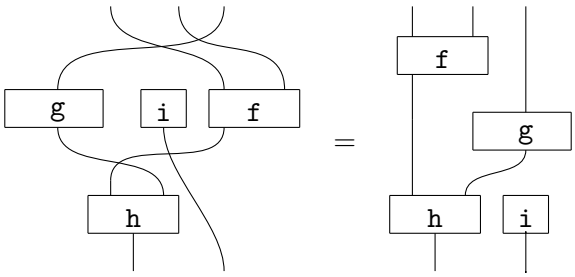


whereas the following are undefined:



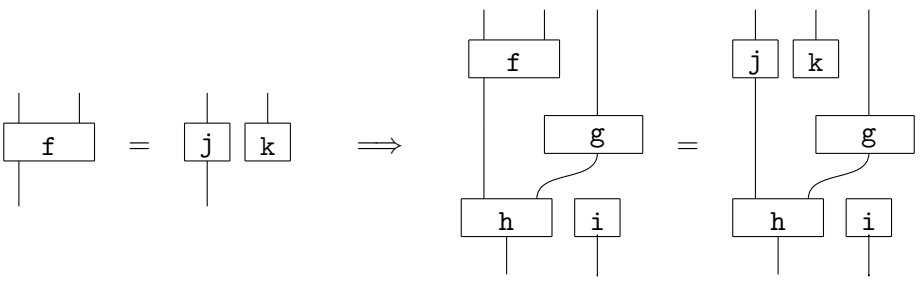
Generally, we shall skip wire labels unless there is an ambiguity.

There are two ways to get new diagram equations from old ones. First, two diagrams are equal as long as they contain the same processes that are connected in the same way, and have the same order of input and output wires. In other words, deforming a diagram while retaining its connectivity and the order of its wires results in the same diagram. For example,



Convention 2.2.11. This property of diagrams—only connectivity matters (OCM)—will be frequently used in this thesis. In this context, the retention of the order of input and output wires is considered part of OCM.

Second, diagrams can be substituted like algebra. For example,



2.2.3 Special Processes

Some special kinds of processes appear a lot in process theories; these include states, effects and numbers [22].

Definition 2.2.12. A *state* is a process with no input. For example, the state

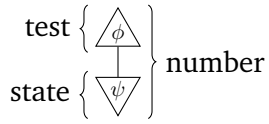
has no input and a single output. This diagram is interpreted as the preparation of a system in a particular state, in a context in which we are not interested in what came before the preparation.

Definition 2.2.13. An *effect* (or a *test*) is a process with no output. For example, the effect

has a single input and no output. This diagram is interpreted as a test of whether the system is in a particular state, in a context in which we are not interested in what happens after the test.

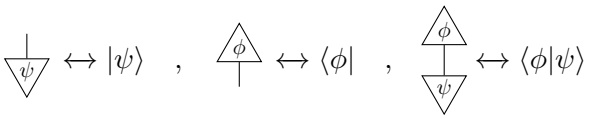
Definition 2.2.14. A number (or a scalar) is a process with no input and no output.

A number can be obtained by composing a state ψ and an effect ϕ.



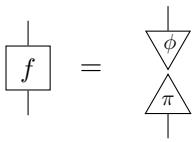
This is known as the *generalised Born rule*.

In the context of quantum theory, diagrams of states, tests and numbers have the following correspondence with the *Dirac notation*:

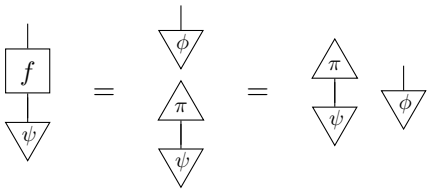


2.2.4 Separability

Definition 2.2.15 (Definition 4.5 in [22])**.** A process f is $\circ$-*separable* if it can be expressed as the series composition of a state ϕ and an effect π:

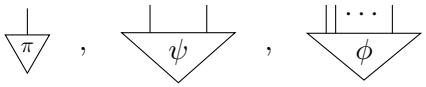


When this process is applied to any state ψ, the same state ϕ (up to a scalar) is obtained as the output:

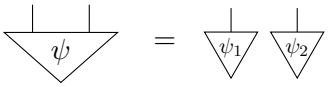


In Dirac notation, the above calculation is $f\,|\psi\rangle = |\phi\rangle\,\langle\pi|\psi\rangle = \langle\pi|\psi\rangle\,|\phi\rangle$.

States can be in one or more than one system, denoted by the number of wires:



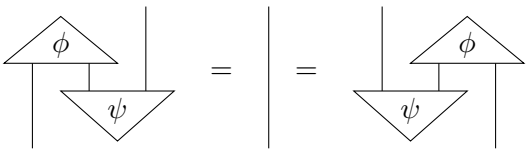
Definition 2.2.16 (Definition 4.2 in [22])**.** A two-system or bipartite state ψ is called $\otimes$-*separable* if it can be expressed as the parallel composition of two one-system states ψ_1 and ψ_2:



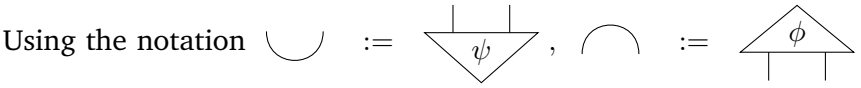
In Dirac notation, this equation corresponds to $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. This is the case if ψ_1 and ψ_2 are completely independent of each other, which is always the case in classical physics.

States that are not $\otimes$-separable are called *non-separable states*.

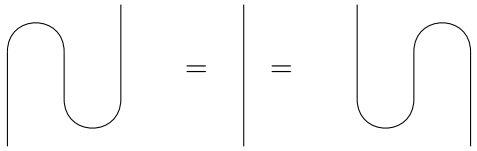
Definition 2.2.17. A two-system state ψ is called a *cup state* if there is a two-system effect ϕ, called a *cap effect*, such that the following equations hold.



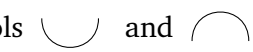
(2.1)

Convention 2.2.18. For each system-type, we fix a specific pair of cup state ψ and cap effect ϕ satisfying Eq. (2.1). Using the notation 

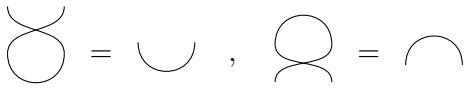
Eq. (2.1) becomes:



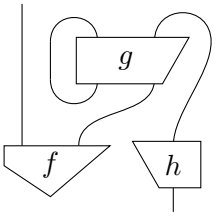
These are sometimes called the *snake equations*.

In diagrams, the symbols  and always denote the chosen cup and cap for the type of the wires they are attached to; they are not intended to range over all possible cup and cap morphisms.

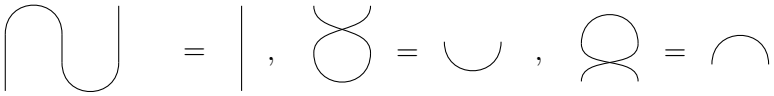
We additionally assume that every system-type is self-dual, so both legs of each cup or cap carry the same type. Under this assumption, we impose the further structural choice that cups and caps are symmetric:



Remark 2.2.19. Process theories that have cup states and cap effects may have diagrams in which wires connect outputs to inputs of the same process.

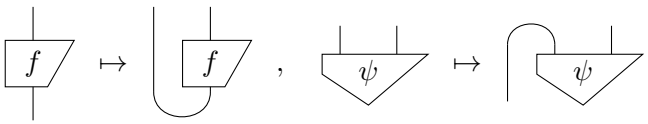


The equations characterising cup states and cap effects

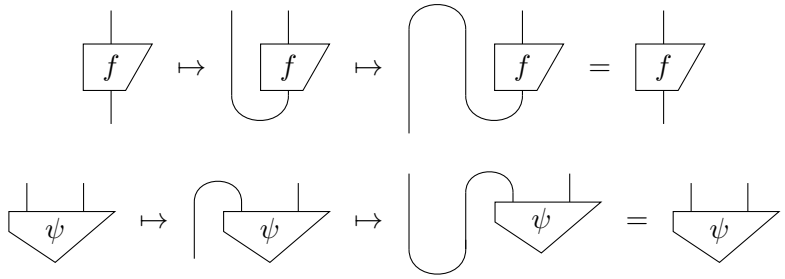


are called the *yanking equations*.

Definition 2.2.20. In string diagrams with cups and caps satisfying yanking equations, there exist states that are duals of processes and vice versa. This bijective correspondence between processes and states is called the *process-state duality*. For example, for single-input, single-output processes and bipartite states:

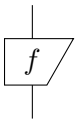


Applying the process-state duality twice, we get back the original states and processes, thanks to the snake equations:

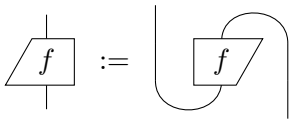


Convention 2.2.21. In the above diagrams, we have used asymmetrical shapes—not rectangles and triangles anymore—for the process and state diagrams. This convention will help visually distinguish between a process and its transpose, adjoint and conjugate processes, described below.

Definition 2.2.22 (Definition 4.23 in [22])**.** The *transpose* f^T of a process f

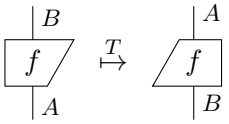


is defined by bending the input and output wires in the opposite directions.

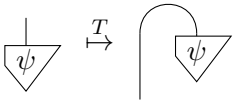


(2.2)

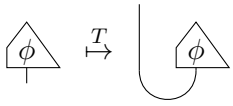
In other words, a cup-and-cap pair are needed to obtain the transpose of the process f. Another way to think of the transpose is that it is a process rotated by $180°$, as is also implied by the notation on the left-hand side above.



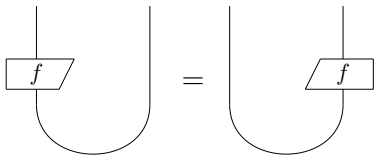
Examples 2.2.23. The transpose of a state is obtained by bending its output wire to make it an input wire.



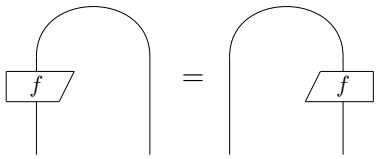
The transpose of an effect is obtained by bending its input into an output.



Boxes can be slid along wires and along cups and caps [22].

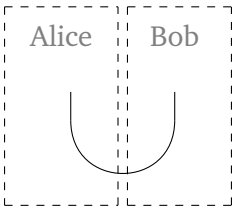


(2.3)

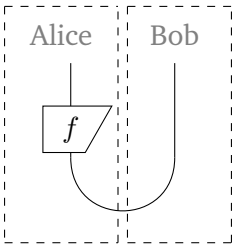


(2.4)

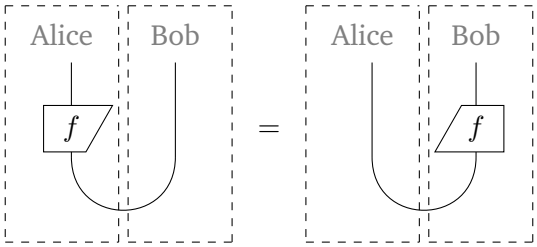
Remark 2.2.24. There is an operational way to think of a process and its transpose. Consider a cup state shared between two parties Alice and Bob that are at different spatial locations.



Suppose Alice applies the process f to her system. The diagram becomes

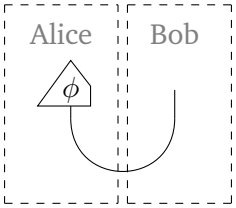


Using Eq. (2.3), we have

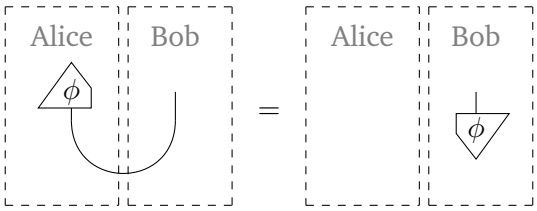


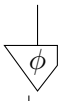
This equation can be interpreted as follows: Alice applying the process f to her system is operationally the same as Bob applying the process f^T to his system.

As another example, suppose Alice applies an effect to test whether her system is in the state 

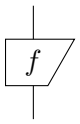


Using Eq. (2.3) again, we have

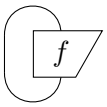


This equation implies that if Alice and Bob share a cup state, Alice successfully testing her system to be in the state  is operationally the same as Alice having no system and Bob's system being in the state  .

Definition 2.2.25 (Definition 4.34 in [22])**.** For a process f

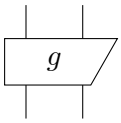


with the same input and output type, the *trace* is defined as

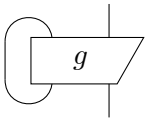


The trace maps a process to a number.

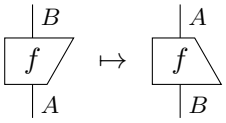
Definition 2.2.26 (Definition 4.34 in [22])**.** For a process g



with one of the inputs (say, the first one) having the same type as one of the outputs (say, the first one), the *partial trace* is defined as

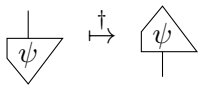


Definition 2.2.27. [22, p. 104] The *adjoint* $f^\dagger$ of f is given by its vertical reflection



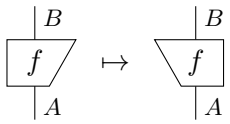
The adjoint of a state is the effect that tests for it.

Example 2.2.28. The adjoint of the state ψ yields the effect that tests for ψ.

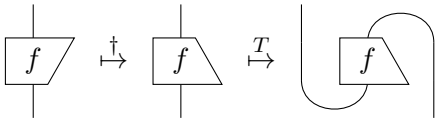


In Dirac notation, it is $(|\psi\rangle)^\dagger = \langle\psi|$.

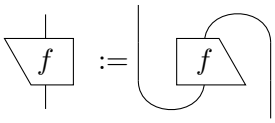
Definition 2.2.29. [22, p. 143] The *conjugate* $\bar{f}$ of f is given by its horizontal reflection



Remark 2.2.30. The conjugate can be obtained by taking the transpose of the adjoint of f,

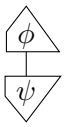


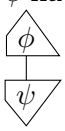
where

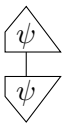


or by taking the adjoint of the transpose of f; *i.e.* $\bar{f} = (f^\dagger)^T = (f^T)^\dagger$.

Using adjoints of processes, the inner product can be defined.

Definition 2.2.31 (Definition 4.48 in [22])**.** If two states ψ and ϕ have the same type, their *inner product* is given by 

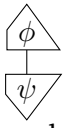
. The states are *orthogonal* if 

$= \quad 0$. A state ψ is *normalised* if 

$=$
.

The empty diagram, denoted by a dashed box, represents the number 1.

In Dirac notation, the inner product of states $|\psi\rangle$ and $|\phi\rangle$ is given by the 'Dirac braket' $\langle\phi|\psi\rangle$. For orthogonal states, $\langle\phi|\psi\rangle = 0$ and for a normalised state $|\psi\rangle$, $\langle\psi|\psi\rangle = 1$.

The diagram 

represents testing 'whether the state ψ is the state ϕ'. In other words, the inner product gives the overlap between the states ψ and ϕ in the form of a number. If there is no overlap, the inner product is 0 and the states are orthogonal. For a normalised state, the inner product with itself is 1.

Definition 2.2.32. An *orthonormal basis (ONB)* is a set of states

$$\mathcal{A} \;:=\; \left\{ \boxed{i} \right\}_i$$

that is orthonormal, *i.e.,* the states obey the equation

$$\boxed{j}\;\boxed{i} \;=\; \delta_i^j$$

and form a basis—the minimal set of states such that for all processes f and g:

$$\left(\forall\, \boxed{i} \;:\; \boxed{g}\;\boxed{i} \;=\; \boxed{f}\;\boxed{i} \right) \;\Longrightarrow\; \boxed{f} \;=\; \boxed{g}$$

In terms of any ONB $\left\{ \boxed{i} \right\}_i$, the identity wire decomposes as follows:

$$\Big| \;=\; \sum_i \boxed{i}\;\boxed{i}$$

and from this decomposition, we can obtain the decomposition of cups and caps, respectively, as

$$\cup \;=\; \sum_i \boxed{i}\;\boxed{i} \quad \text{and} \quad \cap \;=\; \sum_i \boxed{i}\;\boxed{i} \;.$$

Definition 2.2.33 (Definitions 4.54 and 4.56 in [22])**.** A process U is an *isometry* if it obeys the left equation only, and is a *unitary* if it also obeys the right equation:

$$\boxed{U}\;\boxed{U} \;=\; \Big| \quad , \quad \boxed{U}\;\boxed{U} \;=\; \Big| \tag{2.5}$$

Algebraically, the left equation is $U^\dagger U = I$ whereas the right equation is $UU^\dagger = I$.

The *inverse* of a unitary process $\boxed{U}$ is $\boxed{U}$.

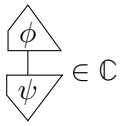
2.3 Towards a Quantum Process Theory

Having discussed process theories and the various string-diagrammatic definitions of 'quantum-like' concepts, one may be tempted to formulate quantum theory as a process theory of **linear maps**.

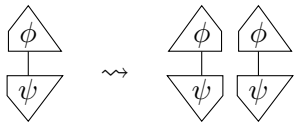
2.3.1 Linear Maps

Consider the process theory of **linear maps** in which system-types are finite-dimensional Hilbert spaces and processes are complex linear maps.

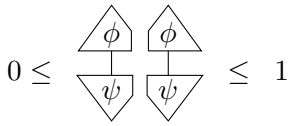
In this process theory, the numbers are complex. This implies that the generalised Born rule in this theory yields complex numbers.



Therefore, the 'Born rule' for linear maps does not give a probability, since a probability must be a real number. This problem can be resolved by multiplying the number with its conjugate.

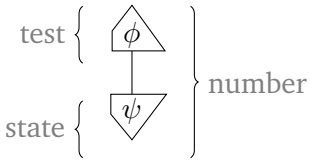


In Dirac notation, this is $\langle\phi|\psi\rangle \rightsquigarrow \overline{\langle\phi|\psi\rangle}\,\langle\phi|\psi\rangle = |\langle\phi|\psi\rangle|^2$.
For normalised states ψ and ϕ,



thereby giving the standard Born rule of quantum theory.

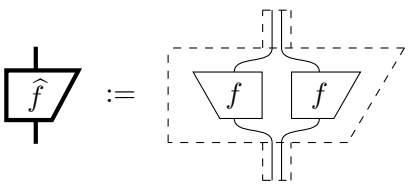
Fixing the numbers by 'doubling' leads to a discrepancy since the states and effects were described with respect to 'un-doubled' numbers.



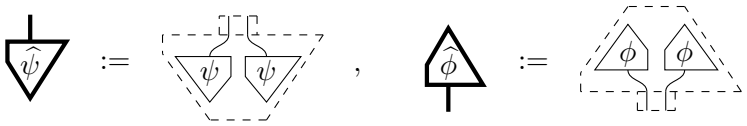
We need a new process theory in which doubling is consistently incorporated rather than introduced at the end in an ad hoc manner.

2.3.2 Doubling

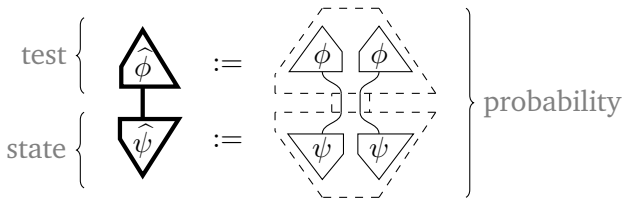
Doubling all the processes in the theory of **linear maps**,



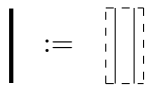
we get the process theory of **pure quantum maps.** In this process theory, states and effects are respectively given by



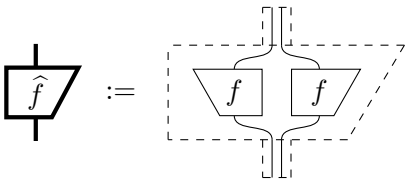
and they are related to probability by the Born rule as follows:



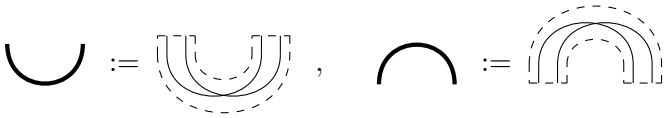
Definition 2.3.1 (Definition 6.8 in [22])**.** The process theory of **pure quantum maps** has doubled finite-dimensional Hilbert spaces $\hat{H} = \mathcal{H} \otimes \mathcal{H}$ as system-types



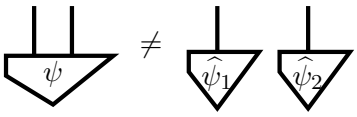
and doubled linear maps (called *pure quantum maps*) as processes



In this process theory, there are doubled cups and caps, related to single cups and caps as follows:



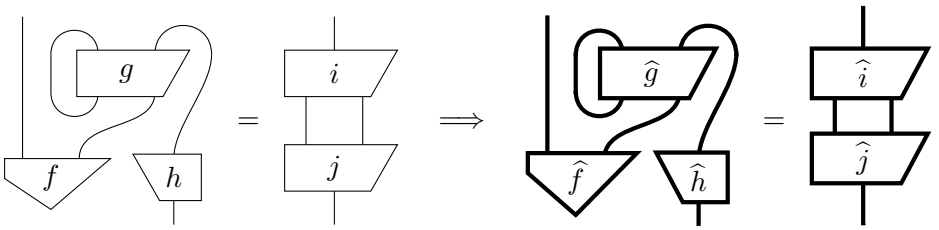
Definition 2.3.2 (Definition 6.15 in [22])**.** An *entangled state* is a pure quantum state that is not $\otimes$-separable. A bipartite entangled state is given by



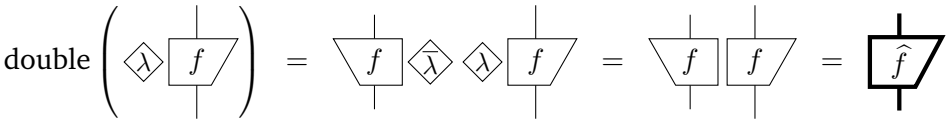
Example 2.3.3. A well-known example of an entangled state is the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, diagrammatically given by a doubled cup multiplied by a scalar.

The corresponding Bell test is given in terms of a doubled cap.

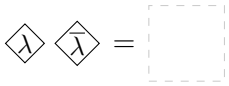
Remark 2.3.4. [22, p. 630] Doubling in a process theory preserves diagrams and diagrammatic equations



but it does not preserve global phases

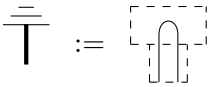


since

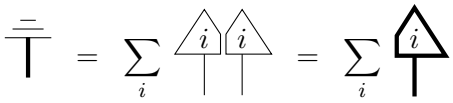


where $\lambda = e^{i\theta}$, the global phase.

Definition 2.3.5 (Definition 6.29 in [22])**.** Using doubling, we can define a process called *discarding*, by connecting the two single wires in a doubled wire.

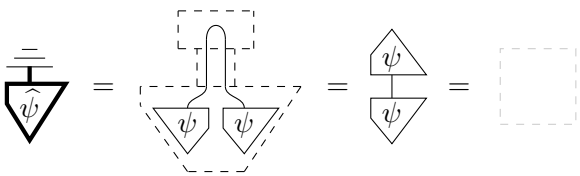


Notice that the discarding map is actually a cap in disguise, which means it has the same decomposition in terms of an ONB:



(2.6)

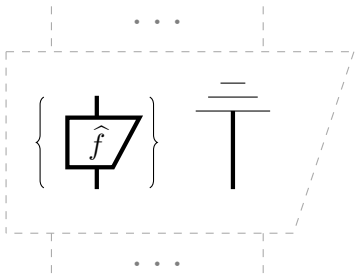
The discarding operation gets rid of a system. For instance, for a normalised state $\widehat{\psi}$, it gives



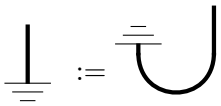
The discarding operation is a linear map but it is not a pure quantum map, since it cannot be obtained by doubling a linear map. It is a useful process to have in a quantum process theory. It is particularly important when we want to ignore or destroy (a part of) a system.

2.3.3 Quantum Maps

Definition 2.3.6 (Definition 6.44 in [22])**.** The process theory of **quantum maps** includes doubled finite-dimensional Hilbert spaces as system-types, and diagrams of pure quantum maps and discarding as processes.



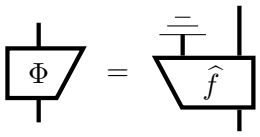
In this theory, we get the maximally mixed state by taking the transpose of the discarding process.



Definition 2.3.7 (Definition 6.34 in [22])**.** The *maximally mixed state* $\frac{|0\rangle\langle 0|+|1\rangle\langle 1|}{2}$ is given by

Any impure quantum map can be described in terms of a bigger pure map and discarding. This is achieved through a technique called purification.

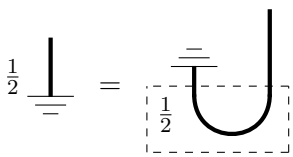
Definition 2.3.8 (Definition 6.47 in [22])**.** Any quantum map Φ, which may or may not be impure, can be expressed as a combination of a pure map $\widehat{f}$ and discarding :



(2.7)

If such a relation exists between the processes $\widehat{f}$ and Φ, $\widehat{f}$ is called a *purification* of Φ.

Example 2.3.9. The Bell state is a purification of the maximally mixed state.

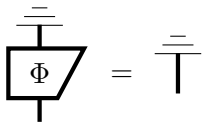


This equation implies that if one of the systems of the Bell state is discarded, the other system becomes a maximally mixed state.

2.3.4 Causality

Causality is defined in terms of the discarding operation.

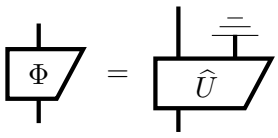
Definition 2.3.10 (Definition 6.52 in [22])**.** A process ϕ is *causal* if it obeys the equation



i.e., performing a process and then discarding it is essentially the same as plain discarding. That is, if the output of a process is discarded, it is as if the process never happened.

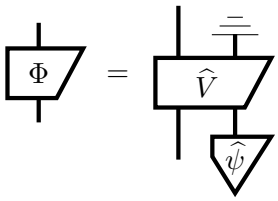
Here, a causal process means one that can be deterministically realised in the physical world. Discarding is the only causal quantum effect [22], others are non-deterministic.

Theorem 2.3.11 (Theorem 6.61 in [22])**.** *For every causal quantum map Φ, there exists a purification by an isometry $\widehat{U}$:*



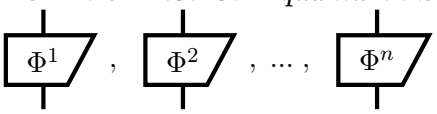
This is known as Stinespring dilation.

Remark 2.3.12. Another version of Stinespring dilation is as follows. For any causal quantum map ϕ, there exists a unitary $\widehat{V}$ that has a pure causal quantum state $\hat{\psi}$ as one of its inputs and one of its outputs is discarded [22].

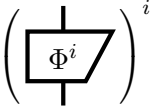


(2.8)

This version of Stinespring dilation is sometimes used to argue that the only physical processes in the real world are the pure, unitary processes, and what we observe are impure processes because we can access only a small part of the complete process.

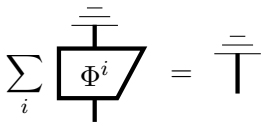
Definition 2.3.13. A *quantum instrument* is essentially a collection or list of quantum maps 

also denoted as



(2.9)

for which the following (generalised) causality condition holds:



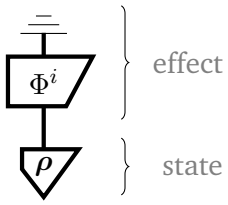
(2.10)

The quantum maps Φ^i in list (2.9) are called *branches*. If the quantum instrument includes only one branch, it is *deterministic*. If it includes more than one branch, it is *non-deterministic*. When a system is subjected to a quantum instrument, one of the branches is realised.

Remark 2.3.14. Note that the abovementioned quantum instrument is more precisely qualified as *uncontrolled*, because it does not depend on any outcome of earlier processes [22].

When a system $\boldsymbol{\rho}$ is subjected to a quantum instrument, the probability of occurrence of each branch Φ^i is given by the Born rule.

$P(\Phi^i|\boldsymbol{\rho})$:=

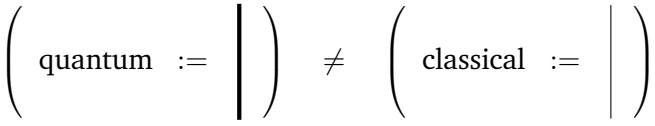


(2.11)

2.3.5 Classical Systems

Most discussions of quantum theory involve not only quantum systems but also classical systems and how quantum systems interact with classical data. This is especially the case when measurement or a controlled process is involved. In many accounts of measurement, classical data are obtained as a result of the measurement of quantum systems. In controlled processes, classical data control which quantum processes are applied, thereby affecting quantum systems.

To recap, doubling **linear maps**—the process theory of finite-dimensional Hilbert spaces and complex linear maps—resulted in the process theory of **pure quantum maps**. Introducing the discarding operation yielded impure quantum maps, leading to the theory of **quantum maps**. Now, adding single (*i.e.* non-doubled) wires/systems to **quantum maps** adds classical data to the process theory. The doubled wires denote finite-dimensional quantum systems while the single wires denote finite-dimensional classical systems/data.



Definition 2.3.15. *Classical states* are defined by fixing an ONB $\left\{ \boxed{i} \right\}_i$ and then taking the basis states as classical values of data. A classical state $\boxed{i}$ denotes 'providing the classical value i'.

Definition 2.3.16. The corresponding *classical effects* are given by $\boxed{i}$ which represents 'testing for the classical value i'.

This definition of classical states and effects is consistent with orthonormality:

$$\boxed{j}\,\boxed{i} \;=\; \delta_i^j$$

Convention 2.3.17. All the definitions in this section from this point onwards will be with respect to the same fixed ONB $\mathcal{A} \;:=\; \left\{ \boxed{i} \right\}_i$.

Example 2.3.18. For a classical *bit*, the ONB can be fixed as

$$\mathcal{B} \;:=\; \left\{ \boxed{0} \;,\; \boxed{1} \right\}$$

and the two basis states give classical values of data.

Definition 2.3.19. A *probability distribution* $\{p^i\}_i$ can be associated with the state

$$\boxed{p} \;:=\; \sum_i p^i \boxed{i}$$

The states $\boxed{i}$ then represent probability distributions called *point distributions*.

The state $\boxed{p}$ can be thought of as the most general state of a classical system, whose value is not exactly known and hence a probability distribution is needed to describe it.

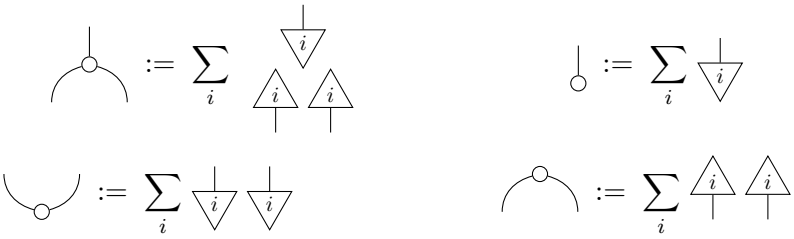
Definition 2.3.20 (Definition 8.17 in [22])**.** The *copy* and *delete* operations are defined respectively as

$$\curlyvee \;:=\; \sum_i \boxed{i}\,\boxed{i}\,\boxed{i} \qquad\qquad \mid \;:=\; \sum_i \boxed{i}$$

One can verify that

Note that the copy map works only for the ONB states. In other words, a state can be copied by the copy operation if and only if it is a basis state [22]. The general probability distribution state  cannot be copied.

Other important classical maps are as follows.

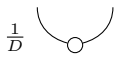


The first map is called *matching*. It takes two ONB states as input; if they are equal, it gives the same state as its output. Otherwise, it gives a zero. The second map gives a uniform probability distribution. The last two maps are the cup and cap in disguise (as can be checked from their ONB expression; they give the state and effect for two classical systems that are perfectly correlated).

Definition 2.3.21 (Definition 8.16 in [22])**.** The *uniform probability distribution* is defined as the state

where D represents the dimension of the ONB.

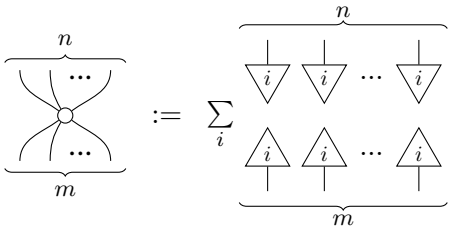
Definition 2.3.22. *Perfect classical correlations* are defined as the state



where D represents the dimension of the ONB.

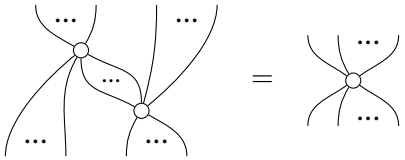
The aforementioned classical maps are special cases of a 'spider'.

Definition 2.3.23 (Definition 8.31 in [22])**.** A *spider* is defined as



(2.12)

In Dirac notation, the right hand side of the equation corresponds to $\sum_i |\underbrace{i \cdots i}_{n}\rangle \langle\underbrace{i \cdots i}_{m}|$. Spiders can be composed using the following rule.

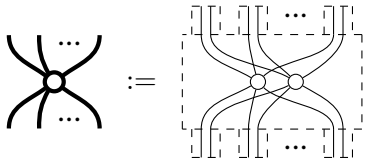


(2.13)

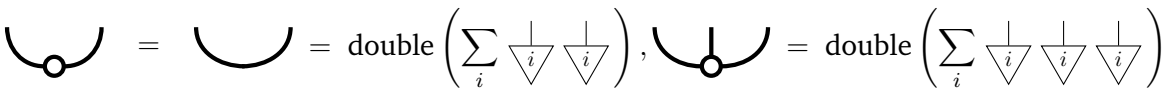
A spider can be thought of as a 'generalised wire'. A wire has a single input and a single output. A spider can have multiple inputs and outputs. The spider in the definition above has m inputs and n outputs.

2.3.6 Classical-quantum Interaction

Doubling *classical spiders* yields *quantum spiders*:

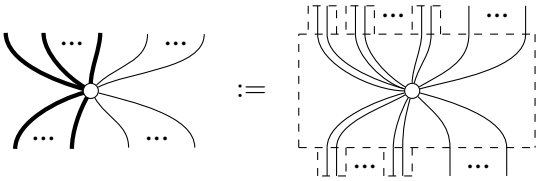


Key examples of quantum spiders are the Bell state and the Greenberger–Horne–Zeilinger (GHZ) state, respectively given by



In Dirac notation, these states are $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ and $\frac{|000\rangle+|111\rangle}{\sqrt{2}}$ respectively.

Just as classical spiders are used to represent and reason about classical systems, so quantum spiders do for quantum systems. Classical and quantum spiders combine to give rise to a more general form of a spider, called the *bastard spider* [22]:

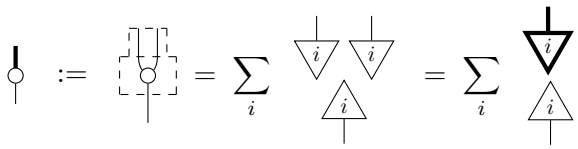


(2.14)

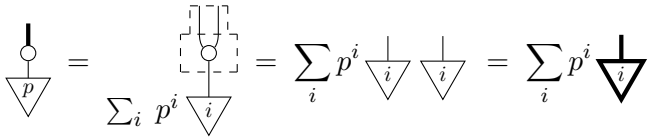
Remark 2.3.24. It is pertinent to note that, due to the subtle differences between classical and quantum spiders in diagrams, some extra care is warranted. Classical spiders are represented by thin wires and dots, whereas quantum spiders are depicted using thicker wires and dots. For instance, consider the bastard spider on the left-hand side of Eq. (2.14). The wires on the left are thick, and therefore quantum, while the wires on the right are thin, indicating they are classical. The dot is also classical.

Bastard spiders are important for representing and reasoning about interactions between classical and quantum systems. Three important examples in this regard are the *encode*, *measure* and *decoherence* processes.

Definition 2.3.25. The *encode* process encodes classical data into a quantum system.

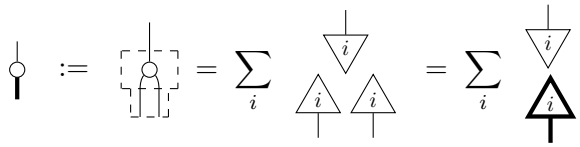


In other words, it takes a classical state as an input and gives a quantum state as an output. For instance, a probability distribution, which is a general classical state, is encoded as

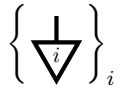


Note that the encode process is actually the copy process in disguise. Measuring is the adjoint of encoding and is therefore matching in disguise.

Definition 2.3.26. The *measure* process

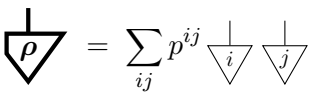


measures the state of a quantum system with respect to the basis

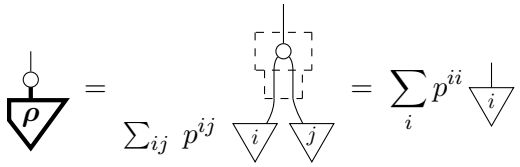


and the outcome (classical data) of the measurement is a probability distribution.

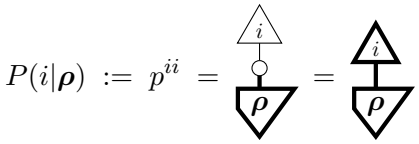
For example, consider the general quantum state



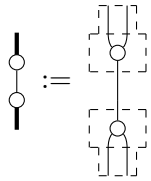
that is measured as follows:



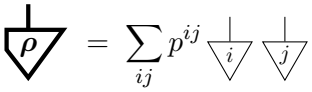
where the probabilities $\{p^{ii}\}_i$ are given by the Born rule:



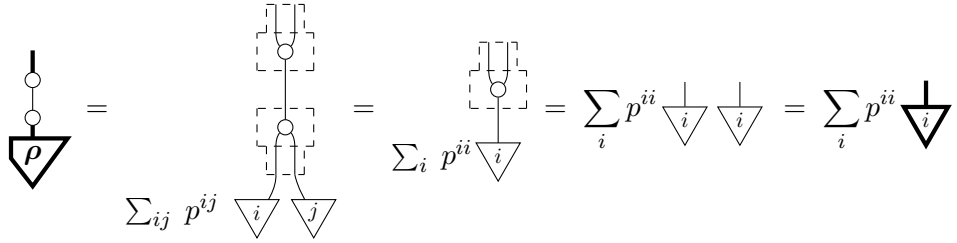
Definition 2.3.27 (Definition 8.61 in [22])**.** Decoherence can be defined as a measurement followed by encoding:



To illustrate, consider the general quantum state



that undergoes decoherence:



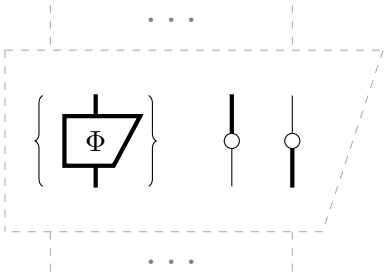
2.4 Categorical Quantum Mechanics

Armed with all of the diagrammatic tools we have discussed so far, we are now ready to jump into full-blown diagrammatic quantum theory.

2.4.1 Quantum Processes

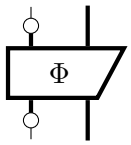
First, we describe a process theory that incorporates both quantum and classical maps.

Definition 2.4.1 (Definition 8.3 in [22])**.** The process theory of **classical-quantum maps** (abbreviated as **cq-maps**) comprises finite-dimensional Hilbert spaces and doubled finite-dimensional Hilbert spaces as system-types and the processes are diagrams composed of quantum maps, the encode process, and the measure process:

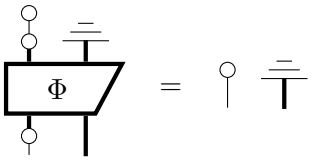


Requiring the processes in this theory to be causal gives us quantum theory.

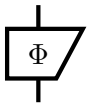
Definition 2.4.2 (Definition 8.10 in [22])**.** The theory of **quantum processes**, which is *quantum theory*, comprises **cq-maps**



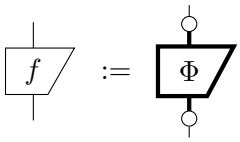
that are causal:



If we restrict this theory to processes with no classical inputs and outputs



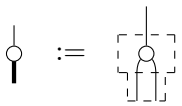
we get the process theory of **causal quantum maps**. Conversely, if we restrict to processes with no quantum inputs and outputs,



we get the process theory of causal classical maps. This theory is also called **classical processes** or **stochastic processes**.

2.4.2 Quantum Measurement

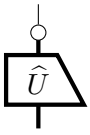
Having finally described a quantum process theory, we are now ready to look at the different notions of measurement that show up in standard quantum theory. Previously, we mentioned the simplest form of measurement, given by the process



(2.15)

defined with respect to a fixed ONB. A more general version of this process is obtained by composition with a unitary.

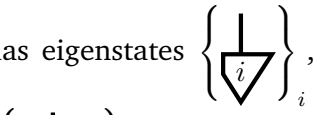
Definition 2.4.3. A *demolition ONB* measurement is given by

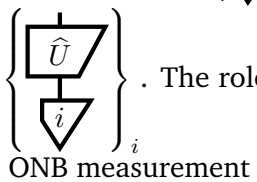


(2.16)

where $\widehat{U}$ is a unitary quantum process.

The measure process (2.15) is a special case of demolition ONB measurement in which the unitary process is the identity.

The *eigenstates* of ONB measurements are given by doubled ONB states. The ONB measurement (2.15) has eigenstates 

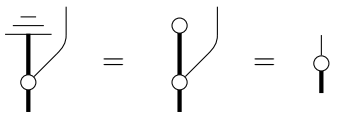
, whereas the more general measurement (2.16) has eigenstates 

. The role of the unitary $\widehat{U}$ is to obtain an arbitrary ONB measurement from the ONB measurement (2.15). Any state that is not an eigenstate of the measurement is called a *superposition* state with respect to that measurement [22].

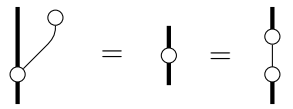
These measurements are called *demolition* measurements because the quantum state is destroyed as a result of these processes. A *non-demolition* measurement is one in which the quantum state is not destroyed. An example of a non-demolition measurement is


(2.17)

Remark 2.4.4. We get a demolition measurement by discarding the quantum output of a non-demolition measurement.

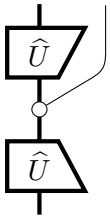


Conversely, discarding the classical outcome results in decoherence:



In other words, a non-demolition measurement followed by discarding the outcome of the measurement has the same effect on the quantum state as decoherence.

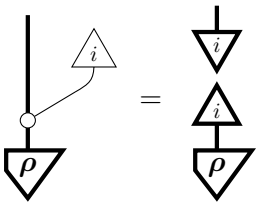
Definition 2.4.5. A *non-demolition measurement* is defined as



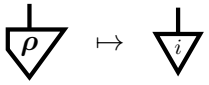
(2.18)

where $\widehat{U}$ is a unitary. The process (2.17) is a non-demolition measurement in which the unitary is the identity.

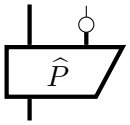
What happens to a quantum state after a non-demolition measurement, say the process (2.17), is performed and the classical outcome i is obtained? Diagrammatically, this non-demolition measurement of the quantum state $\boldsymbol{\rho}$ is given by

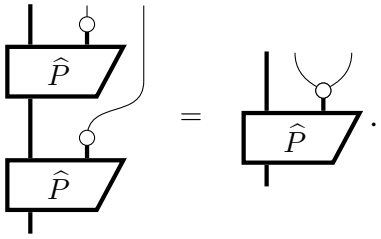


The measurement results in changing the state ρ to the state i. This is usually called a *collapse*:



Measurements based on spiders and unitaries only are known as *degenerate* [75]. There exists a more general variant of measurement, called a *von Neumann measurement* or *projective measurement*.

Definition 2.4.6. [75, p. 23] A (non-demolition) *von Neumann measurement* is defined as a quantum process 

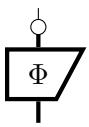
that obeys the equation 

.

In other words, making a von Neumann measurement twice is equal to making the measurement once and then copying the outcome of the measurement. This implies that once a von Neumann measurement is made, making the same measurement again will result in the same outcome, and the quantum state will stay the same. This characteristic of von Neumann measurements is commonly known as the *von Neumann projection postulate*. One can verify that measurements defined by processes (2.18) and (2.17) obey the projection postulate and are hence von Neumann measurements. If the quantum output of the von Neumann measurement is discarded, a demolition von Neumann measurement is obtained.

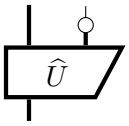
More generally, a demolition quantum measurement can be defined as a quantum process from a quantum input to a classical output.

Definition 2.4.7. A *demolition positive operator-valued measure (POVM) measurement* is defined as



(2.19)

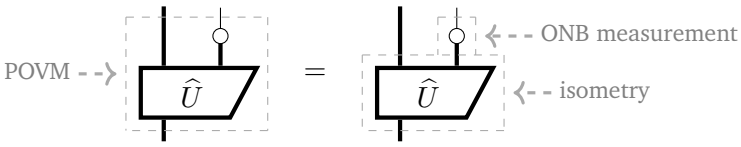
where Φ is a quantum process with pure quantum maps as branches. A *non-demolition POVM measurement* is obtained by purifying (2.19):



(2.20)

Here $\widehat{U}$ is an isometry.

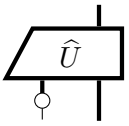
Remark 2.4.8. In fact, every non-demolition POVM measurement is obtained by composing one output of an isometry $\widehat{U}$ with an ONB measurement [22]:



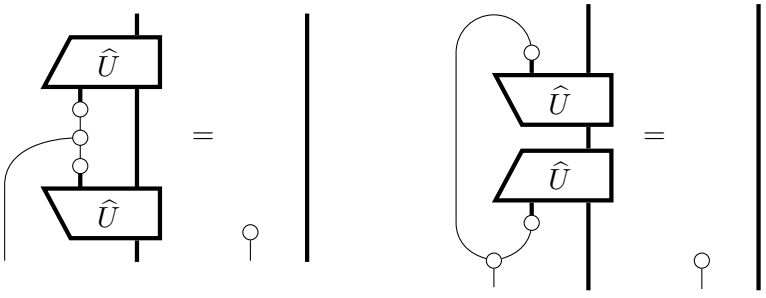
This is known as *Naimark dilation*.

So far, we have defined measurements and processes/operations that are independent of the earlier measurement outcomes. We can also define processes controlled by measurement outcomes or classical inputs; for instance, we can define controlled unitaries.

Definition 2.4.9. A *controlled unitary* is a quantum-classical process



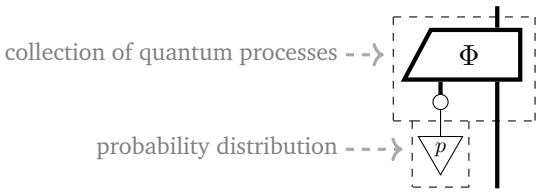
satisfying the equations:



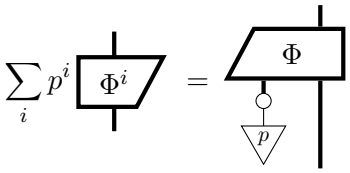
2.4.3 Mixtures

There are situations in which we are not certain about which one of a known set of possible quantum processes actually took place. In such a case, we can represent the set of processes as a single quantum process controlled by a classical input. This classical input is supplied by a probability distribution to take into account our ignorance.

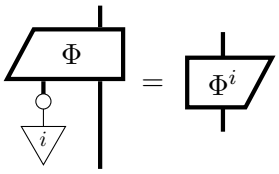
Definition 2.4.10. A *mixture* of quantum processes is defined as



In fact, we can represent any mixture of quantum processes like this [22]:



where $\{p^i\}_i$ represents a probability distribution.The components of the mixture can be obtained by plugging ONB states into the classical input [22]:



Remark 2.4.11. For a mixture (Definition 2.4.10), it is known that one of the processes from a certain collection occurs, but it is not known which one. The probability distribution supplied to the classical input reflects our lack of knowledge about which process actually took place. On the other hand, a quantum instrument with multiple branches (as described in Definition 2.3.13) is inherently non-deterministic—that is, the non-determinism does not stem from our ignorance.

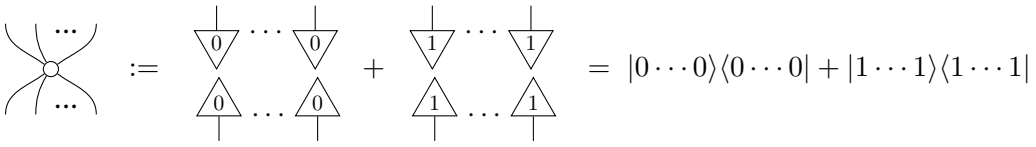
2.5 The ZX-calculus

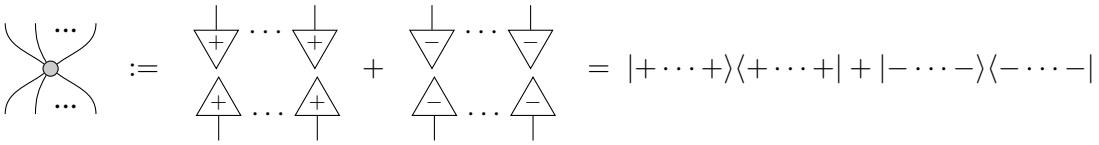
We have discussed most of the key concepts in quantum theory using our diagrammatic formalism. What is left is a description of *phases* and *complementarity*. Incorporating these aspects of quantum theory requires us to look inside the boxes of the process theory we have described. This granular description of quantum processes involves spiders corresponding to two complementary ONBs and 'decorated' with phases. In this section, we restrict the core of our discussion to qubits.

2.5.1 Complementarity

We saw that spiders were described with respect to a fixed ONB. For two complementary bases, we shall need spiders of two colours.

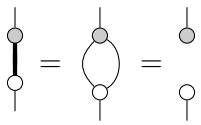
We fix the complementary bases $\{|0\rangle, |1\rangle\}$ and $\{|+\rangle, |-\rangle\}$ for the white and gray spiders respectively. This gives us





Since the chosen bases lie on the Z- and X-axes of the Bloch sphere, these are called Z and X bases; this is where the Z and X of the ZX-calculus come from. The corresponding spiders are called Z and X spiders. The diagrams made of Z and X spiders are called ZX-diagrams.

How do we make sense of complementarity using these two kinds of spiders? Intuitively, complementarity is the idea that if the information is encoded in one basis and measured in a complementary basis, there must be no transfer of information. It can be checked that this is the case for the Z and X spiders:



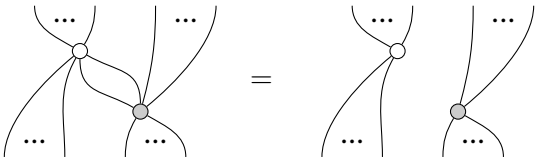
(2.21)

This is interpreted as follows: encoding in one basis and then measuring in a complementary basis allows for no flow of information.

In the last step of Eq. (2.21), the symbol $=$ should be read as equality up to a non-zero global scalar factor. As we shall be mostly concerned with how the processes are composed and not with the exact numbers, we ignore such global scalars in diagrammatic calculations.

Convention 2.5.1. In diagrammatic equations and calculations in what follows, all equalities are up to non-zero global scalars.

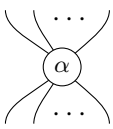
A more general version of the complementarity of Z and X spiders is given by



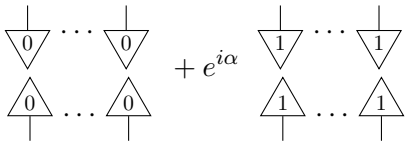
(2.22)

2.5.2 Phases

A *decorated spider* is a spider decorated with a phase. A general Z spider with the phase α is given by



$:=$

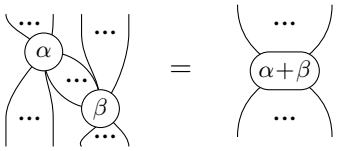


$= |0\cdots0\rangle\langle0\cdots0| + e^{i\alpha}\,|1\cdots1\rangle\langle1\cdots1|$

(2.23)

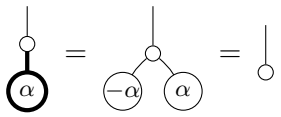
For instance, the decorated spider  corresponds to the state $|0\rangle + e^{i\alpha}\,|1\rangle$, and its conjugate is  $= |0\rangle + e^{-i\alpha}\,|1\rangle$.

The old spider fusion rule (2.13) does not apply to decorated spiders, for which there is the following fusion rule:

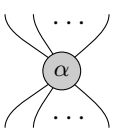


(2.24)

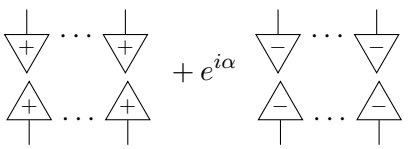
Roughly speaking, phase is the information in the quantum system that gets lost when a quantum state is measured. This can be verified for decorated spiders using the fusion rule (2.24):



A general X spider with the phase α is given by



$:=$

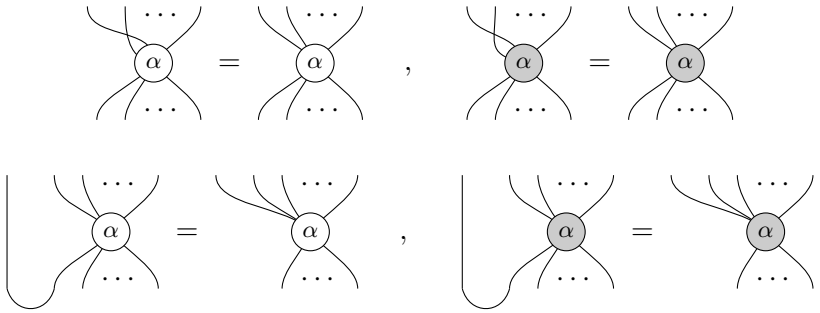


$= |+\cdots+\rangle\langle+\cdots+| + e^{i\alpha}\,|-\cdots-\rangle\langle-\cdots-|$

(2.25)

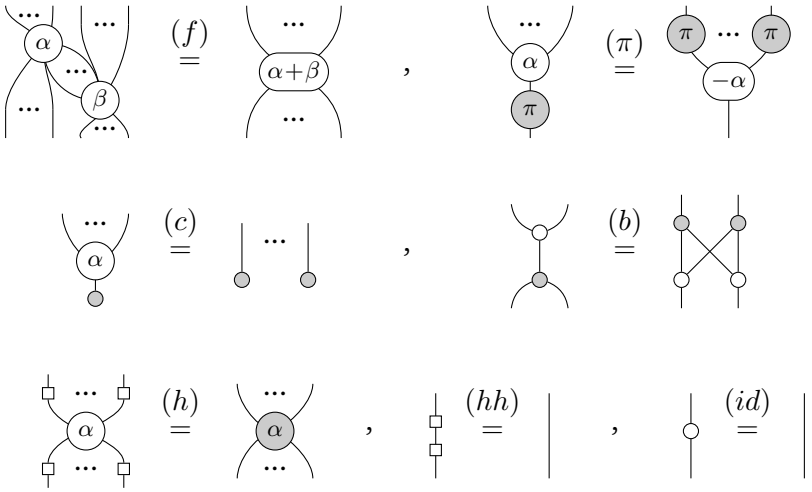
and it has a fusion rule similar to rule (2.24).

Remark 2.5.2. Representing spiders with round dots — rather than asymmetric boxes — makes intuitive sense, since spiders exhibit symmetries such as 'leg-swapping' and 'leg-flipping'.

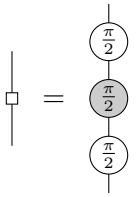


2.5.3 Rules of the ZX-calculus

We have seen some rules about how spiders of the same colour and different colours interact. Adding few more rules, we get the complete set for the stabiliser fragment of ZX-calculus [76]:



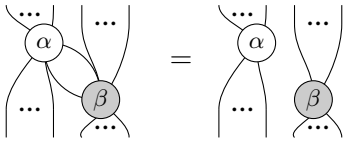
The letters above the equality signs are shorthand notation for the rule names and stand for spider (f)usion, (π)-commute, (c)opy, (b)ialgebra, (h)adamard, (hh)-cancellation and (id)entity. All these rules hold up to a global scalar factor and for phases $\alpha, \beta \in \{0, \pi/2, \pi, -\pi/2\}$. The Hadamard process is a convenient notation for a frequently-used ZX diagram:



and, as illustrated in the (h) rule above, changes spider colour. The (h) and (hh) rules imply that the above rules also hold for spiders with their colours inverted.

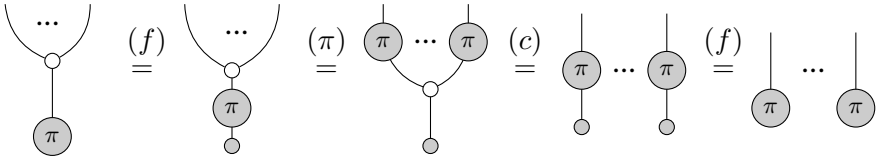
Complementarity (Eq. (2.21)) can be derived using the (f), (id), (b), and (c) rules. The

decorated-spider version of the general complementarity rule (Eq. (2.22)



(2.26)

can be derived using Eq. (2.21) and the (f)-rule. Another rule we shall use later is derived as follows:



(2.27)

A reader interested in a more thorough exposition of the ZX-calculus is referred to the comprehensive tutorial [76] or the textbook [77]; a high-level crash course for professionals can be found in Ref. [24]. Since 2023, there has been a book on the ZX-calculus targeted to the general public [23] as well!

It has been a century since the inception of quantum theory, whereas the ZX-calculus was proposed only in 2007 [78, 79]. When presenting the ZX-calculus as an alternative to the standard Hilbert space formalism, three desirable criteria are typically considered: *universality*, *soundness* and *completeness*. Universality requires that any linear map between qubits be representable as a ZX-diagram. Soundness means that any equality derivable in the ZX-calculus also holds in the Hilbert space formalism. The ZX-calculus is indeed both universal and sound for qubit quantum theory [78, 79].

Theorem 2.5.3. *The ZX-calculus is universal and sound for linear maps between qubits.*

Completeness of the ZX-calculus means that its rules are sufficient to derive any equality that can be derived using the Hilbert space formalism in qubit quantum mechanics. Proving completeness was a non-trivial task and it was achieved in multiple stages, starting with a small fragment of quantum theory and gradually increasing its size [80–83]. This culminated in the proof of completeness of the ZX-calculus for qubit quantum theory in 2017 [84–86].

Theorem 2.5.4. *The ZX-calculus is complete for linear maps between qubits.*

In other words, the ZX-calculus is as expressive as the Hilbert space formalism and hence is a viable substitute of the latter for qubit quantum theory. Research aimed at proving completeness of ZX and similar graphical calculi (such as ZW [87], ZH [88] and ZXW [89,

90]) for bigger fragments of quantum theory is in full swing. In fact, very recently, there has been a presentation of *finite-dimensional ZX-calculus* and its completeness for finite-dimensional Hilbert spaces [91].

The ZX-calculus has been extensively employed in quantum computing and technology research [77].[4] Some examples of research topics include quantum circuit optimisation [65, 66, 92, 93], quantum error correction [69, 94–96], measurement-based quantum computing [74, 97, 98], and fusion-based quantum computing [99, 100].

[4] As of writing this chapter, the website `https://zxcalculus.com/publications.html` lists 300+ papers related to the ZX-calculus.

3

Constructors, Processes, and Quantum Theory

> 'How often have I said to you that when you have eliminated the impossible, whatever remains, *however improbable*, must be the truth?'
>
> Sherlock Holmes to Dr Watson,
> The Sign of Four [101, p. 93]

The novel contributions of this chapter are presented in Sections 3.2, 3.3 and 3.4. Section 3.2 is adapted from the publication [48], based on work carried out in collaboration with Stefano Gogioso, Vincent Wang-Maścianica, Carlo Maria Scandolo, and Bob Coecke. The author of this thesis is the third author of the publication [48]. Only the parts of this work to which the author directly contributed are described here. Sections 3.3 and 3.4 are unpublished and represent original work carried out solely by the author of this thesis.

3.1 Introduction

Constructor theory [46, 102, 103] is a framework to formulate fundamental scientific theories in terms of the *possibility* and *impossibility* of *tasks*. It particularly aims to provide an alternative to the prevailing approach in physics that characterises physical phenomena in terms of initial conditions and dynamical laws. While the prevailing approach formulates theories by specifying a dichotomy between *what happens* and *what does not happen*, constructor theory seeks to do so by specifying a dichotomy between *what can happen* (*i.e.*, what is possible) and *what cannot happen* (*i.e.*, what is impossible) [102].[1]

[1] The eponymous popular science book [103] dubs constructor theory 'the science of can and can't'.

In constructor theory, tasks are defined as *transformations* between *systems*. Auxiliary systems may also be required as inputs by these transformations. For instance, consider the task of transforming black shoes into brown shoes. This task of painting shoes requires an amount of brown paint in addition to black shoes that are to be painted. The paint plays the role of a catalyst.

Tasks transform *states* of systems into other states, and *attributes* of systems, like the blackness of a shoe, into other attributes. In the physical world, the amount of brown paint is limited and the aforementioned task can be performed only a finite number of times before one runs out of brown paint. Ideally, if an unlimited amount of brown paint were available, the painting task could be performed as many times as needed. Such inexhaustible catalysts are called *constructors*.

A task is called *possible* if there is a constructor that makes the task performable an arbitrary number of times. Otherwise, the task is called *impossible*. Even though constructors and tasks are abstract, they offer explanatory value. The aim of constructor theory is to characterise physical theories in terms of which tasks are possible and which are impossible. Constructor theory has been described as a deeper (meta)theory than other theories of physics.[2] In the seminal article by David Deutsch, it is pitched as 'the ultimate generalisation of the theory of computation' [46, p. 12] and also a theory that could 'underlie all other theories including relativity and quantum theory' [46, p. 5]. The relationship between constructor theory and other theories (called *subsidiary theories* by Deutsch [46]) was described as follows:

"Other theories specify what substrates and tasks exist, and provide the multiplication tables for serial and parallel composition of tasks, and state that some of the tasks are impossible, and explain why. Constructor theory provides a unifying formalism in which other theories can do this, and its principles constrain their laws, and in particular, require certain types of task to be possible." [46, p. 5]

Being a metatheory, constructor theory is implementation-agnostic. That is, the user of constructor theory is free to pick a formal system of mathematics of their liking as a concrete language to interpret it. One such language is that of process theories. We contend that process theories represent a good choice of mathematical language to formally incarnate

[2] In an article in Quanta Magazine, it was called 'a master theory—a set of ideas so fundamental that all other theories would spring from it.' [104]

constructor theory because they are especially suitable for describing the composition of processes in space-time. Furthermore, they are expressible as string diagrams, enabling the user to work at an abstraction-level of their choosing.

This chapter is organised as follows. It comprises three main sections, briefly summarised below.

In Section 3.2, the ideas and jargon of constructor theory are formally interpreted using the string-diagrammatic language of process theories. The aim is to use string diagrams to bridge work between the research areas of constructor theory [46, 102, 105–107] and process theories [14, 22, 108–110]. The target is to provide a rigorous and intuitive mathematical language for constructor theory, which may help constructor theorists develop, express and communicate their ideas. This work is also intended to invite constructor theorists to the research area of process theories—potentially an extremely fruitful arena within which the ramifications of constructor theory can be explored.

In Section 3.3, it is argued that a constructor-theoretic formulation of non-relativistic quantum theory is afflicted with a fundamental problem: there is an inconsistency between the restrictions imposed by principles of locality and composition. Unless there is a fully compositional and local formulation of quantum theory (which, to the best of our knowledge, does not exist), either the principle of locality or the composition principle must be discarded from constructor theory.

Finally, in Section 3.4, it is demonstrated via examples that categorical quantum mechanics (CQM) is a viable constructor theory of non-relativistic quantum physics—one that is completely compositional but non-local.

Section 3.5 concludes the chapter.

3.2 Constructor Theory as a Process Theory

This section is adapted from our publication [48]. The expository presentation of this section is intended for those, particularly constructor theorists, who are not familiar with process theories. But it serves another purpose as well: it offers a dictionary to understand the mathematics of constructor theory, transliterated into diagrams with the fewest possible interpretational choices and without any bells and whistles.

3.2.1 Conceivable Tasks

From a constructor-theoretic point of view, a physical theory is characterised by answering the question 'which *tasks* can be performed within this physical theory?'. There are two notions of tasks in constructor theory: the abstract *conceivable tasks*, and the concrete *possible tasks*. Conceivable tasks are independent of particular scientific theories. They provide a formal setting for formulating principles or deriving constraints. Possible tasks, on the other hand, are dependent on the particular theory under discussion and are induced by constructors physically available to implement the tasks.

According to the seminal paper on constructor theory [46], it is required of conceivable tasks that they be composable in parallel and sequence (series).[3] In other words, conceivable tasks form a symmetric monoidal category (SMC). In the same paper, Deutsch used relations between sets to model tasks in constructor theory. The literature of constructor theory [46, 102, 105–107] has followed this modelling choice ever since. In our formalisation of constructor theory as a process theory, we also stick to this choice.

Remark 3.2.1. In this chapter, monoidal categories are taken to be *strict*.[4] Particularly, it is assumed that in a monoidal category **C**, the objects obj(**C**) form a strict monoid. For SMC **Rel**, this means a choice of a singleton set $1 := \{*\}$ that acts as a strict unit for the Cartesian product, *i.e.,* $X \times 1 = X = 1 \times X$.[5]

This also implies that we can write tuples without worrying about using parentheses or nesting. For instance, we can write triples as $X \times Y \times Z = \{(x, y, z) \mid x \in X, y \in Y, z \in Z\}$. It must be noted that strictness does not apply to symmetry isomorphisms; *i.e.,* for symmetry isomorphisms, we have $X \times Y \cong Y \times X$ but this does not imply $X \times Y = Y \times X$

Definition 3.2.2 (Definition 4.97 in [22])**.** A dagger symmetric monoidal category (†-SMC) is a symmetric monoidal category with a *dagger functor* † that

- does not alter objects: $A^\dagger := A$
- reverses morphisms: $(f : A \to B)^\dagger := f^\dagger : B \to A$

[3] "... first constructor-theoretic law of physics, which I shall call the *composition principle* – that *every regular network of possible tasks is a possible task*." [46, p. 4]

[4] Strict monoidal categories are defined in Definition 2.2.2.

[5] SMCs are defined in Definition 2.2.3.

- is involutive: $(f^\dagger)^\dagger = f$

- and respects the symmetric monoidal category structure:

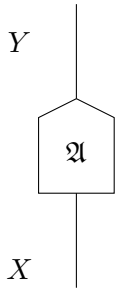
$$(g \circ f)^\dagger = f^\dagger \circ g^\dagger \qquad (f \otimes g)^\dagger = f^\dagger \otimes g^\dagger \qquad \sigma^\dagger_{A,B} = \sigma_{B,A}$$

In this work, the theory of *conceivable tasks* is taken to be **Rel**, which is the †-SMC of sets and relations. The components of this theory are described below.

A task or relation $\mathfrak{A} \subseteq X \times Y$ is denoted by $\mathfrak{A} : X \to Y$ where sets X and Y comprise legitimate input and output states respectively of the task. In order to avoid confusion, the pair or tuple notation is used for pairs or tuples of a Cartesian product, and the *maplet* notation is used for pairs of domain and codomain elements in a relation.

$$x \mapsto y \;\; :\equiv \;\; (x, y) \qquad\qquad x \overset{\mathfrak{A}}{\mapsto} y \;\; :\equiv \;\; (x, y) \in \mathfrak{A}$$

The $\mathfrak{A}$ symbol is omitted from $\overset{\mathfrak{A}}{\mapsto}$ when there is no ambiguity and the context is clear. Diagrammatically, this task can be represented as

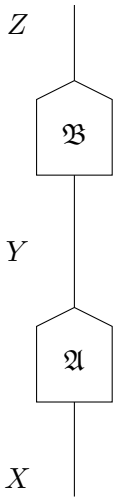


The arrow shape of the box represents the direction of the task from input to output.

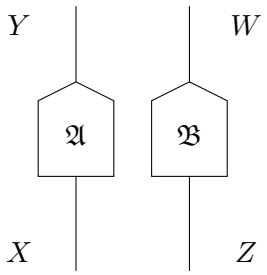
As mentioned before, conceivable tasks can be composed in sequence and parallel.

The *sequential composition* of tasks $\mathfrak{A} : X \to Y$ and $\mathfrak{B} : Y \to Z$, denoted by $\mathfrak{B} \circ \mathfrak{A} : X \to Z$, is defined as $\mathfrak{B} \circ \mathfrak{A} := \left\{ x \mapsto z \;\middle|\; \exists y \in Y.\, x \overset{\mathfrak{A}}{\mapsto} y \text{ and } y \overset{\mathfrak{B}}{\mapsto} z \right\}$. Here task $\mathfrak{A}$ happens first, followed

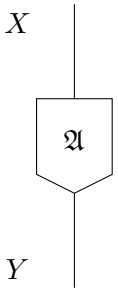
by $\mathfrak{B}$. Diagrammatically, this sequential composition is represented as



Using two sets of states X and Y, a composite set of states can be obtained using the Cartesian product $X \times Y$, as $X \times Y := \{(x, y) \mid x \in X \text{ and } y \in Y\}$. The *parallel composition* of two tasks $\mathfrak{A} : X \to Y$ and $\mathfrak{B} : Z \to W$, denoted by $\mathfrak{A} \times \mathfrak{B} : X \times Z \to Y \times W$, is defined as $\mathfrak{A} \times \mathfrak{B} := \left\{(x, z) \mapsto (y, w) \;\middle|\; x \overset{\mathfrak{A}}{\mapsto} y \text{ and } z \overset{\mathfrak{B}}{\mapsto} w\right\}$. Diagrammatically, the parallel composition is given by

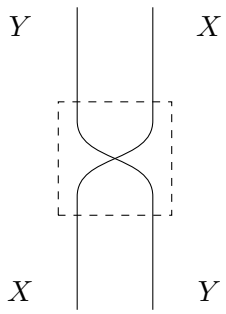


The *transpose* of a task $\mathfrak{A} : X \to Y$, denoted by $\mathfrak{A}^\dagger : Y \to X$, is defined as $\mathfrak{A}^\dagger := \left\{y \mapsto x \;\middle|\; x \overset{\mathfrak{A}}{\mapsto} y\right\}$ and diagrammatically represented as

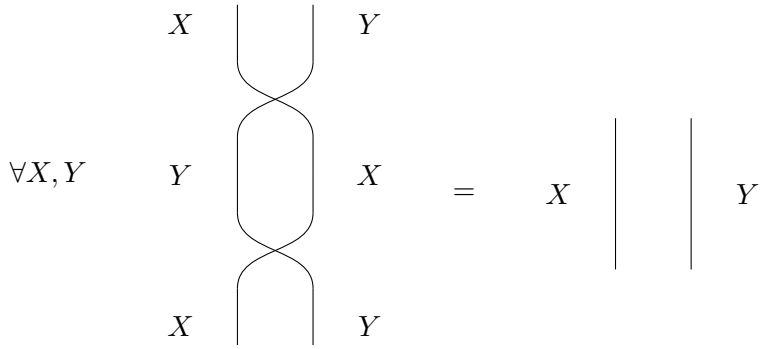


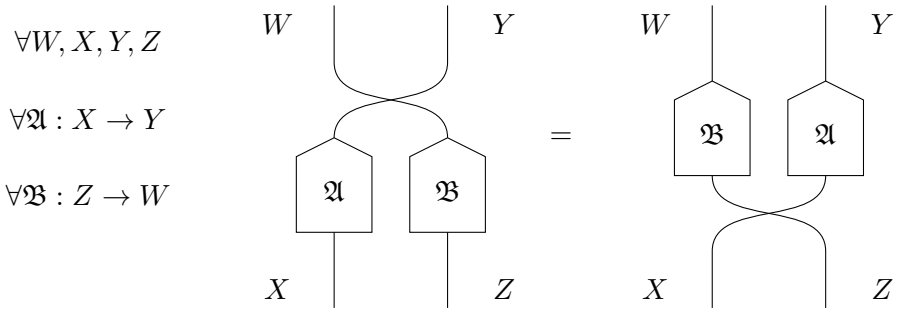
Symmetry isomorphisms, which are also known as *swaps*, are denoted by $\sigma_{X,Y} : X \times Y \overset{\cong}{\to} Y \times X$ and defined as $\sigma_{X,Y} := \{(x, y) \mapsto (y, x) \mid x \in X \text{ and } y \in Y\}$. Diagrammatically,

swaps are represented as



Swaps are a structural feature of the category and have the following properties.





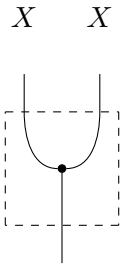
Thanks to swaps and their nice properties shown above, relations can be composed into acyclic networks, in which outputs of relations can be connected to inputs of other relations.

Remark 3.2.3. The sets X and Y are taken to be distinct for the sake of generality. Restricting these sets to be always equal means restricting the theory of conceivable tasks to be the †-SMC **EndoRel** of sets and endo-relations $R : X \to X$, which is a sub-†-SMC of **Rel**.

The *copy map*, denoted as $\delta_X : X \to X \times X$, is defined as follows on a set X:

$$\delta_X := \{x \mapsto (x, x) \mid x \in X\}$$

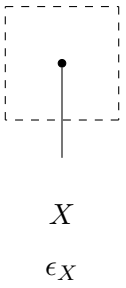
Diagrammatically, the copy map is given by



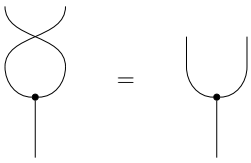
The *discarding map*, denoted as $\epsilon_X : X \to 1$, is defined as follows on a set X:

$$\epsilon_X := \{x \mapsto * \mid x \in X\}$$

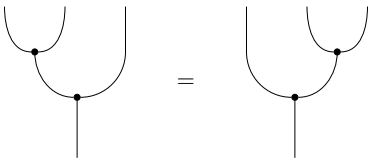
diagrammatically represented as



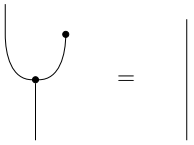
Copy and discarding maps have some useful properties. Copies are indistinguishable under swaps



and under repeated copying



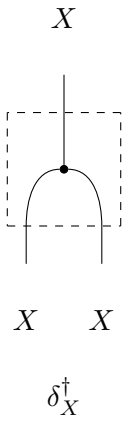
Moreover, copying and then discarding one of the copies results in the identity:



The transpose $\delta_X^\dagger : X \times X \to X$ of the copy map is a partial function that gives the common value of its inputs as the output when the inputs are equal, and is not defined otherwise.

$$\delta_X^\dagger := \{(x, x) \mapsto x \mid x \in X\}$$

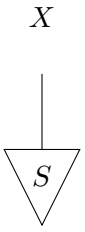
It is called the *match map*. Diagrammatically, it is represented as



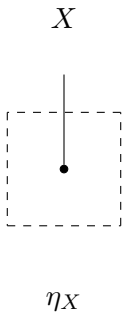
It is important to distinguish between states and attributes: considering the set X, its elements $x \in X$ are referred to as *states* while its subsets $S \subseteq X$ are referred to as *attributes.*.

Relations $S : 1 \to X$ can be identified with all possible *attributes* of states in X, *i.e.,* with all possible subsets $S \subseteq X$: $S \cong \{* \mapsto x \mid x \in S\}$.

In diagrams, states and attributes are represented by the same notation. This is because states $x \in X$ can be identified with singleton sets $\{x\} \subseteq X$. Thus, states or attributes are diagrammatically given by

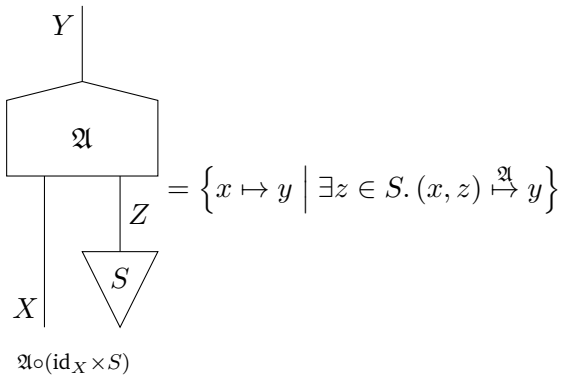


The transpose $\epsilon_X^\dagger : 1 \to X$ of the discarding map is the *trivial attribute*, corresponding to the subset $X \subseteq X$. It is defined as $\eta_X := \epsilon_X^\dagger = \{* \mapsto x \mid x \in X\}$ and is represented diagrammatically as

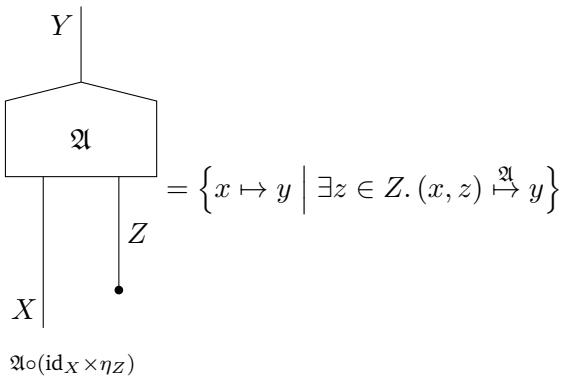


Tasks can be conditioned to specific input states using attributes.

Definition 3.2.4. Let $\mathfrak{A} : X \times Z \to Y$ be a task and let $S \subseteq Z$ be an attribute on states in Z. The task obtained by forgetting all information about the Z input of task $\mathfrak{A}$ except the fact that the input state has attribute S is the following *pre-conditioned task*.



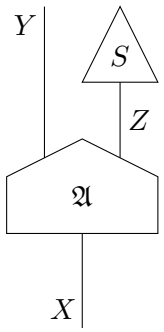
As a special case, the Z input can be discarded completely. This is obtained by pre-conditioning against the trivial attribute η_Z, as follows:



The relations $X \to 1$ are exactly the transposes $S^\dagger : X \to 1$ of the attributes $S : 1 \to X$. Explicitly, they are the constant partial functions with the attribute S as their domain: $S^\dagger := \{x \mapsto * \mid x \in S\}$. The transposes of attributes are called *tests*. Tasks can be conditioned to specific output states using tests.

Definition 3.2.5. Let $\mathfrak{A} : X \to Y \times Z$ be a task and let $S \subseteq Z$ be an attribute on states in Z. The task obtained by forgetting all information about the Z output of task $\mathfrak{A}$ except the fact

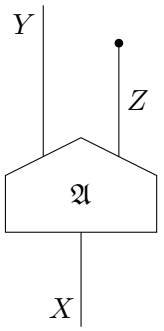
that the output state has attribute S is the following *post-conditioned task*.



$$= \left\{ x \mapsto y \;\middle|\; \exists z \in S.\, x \overset{\mathfrak{A}}{\mapsto} (y, z) \right\}$$

$(\mathrm{id}_Y \times S^\dagger) \circ \mathfrak{A}$

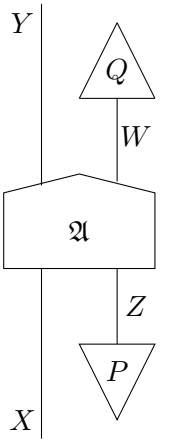
As a special case, the Z output can be discarded completely. This is obtained by post-conditioning against the trivial attribute η_Z, as follows:



$$= \left\{ x \mapsto y \;\middle|\; \exists z \in Z.\, x \overset{\mathfrak{A}}{\mapsto} (y, z) \right\}$$

$(\mathrm{id}_Y \times \epsilon_Z) \circ \mathfrak{A}$

Remark 3.2.6. A task $\mathfrak{A} : X \times Z \to Y \times W$ can simultaneously be pre-conditioned against an attribute $P \subseteq Z$ and post-conditioned against an attribute $Q \subseteq W$ as follows:



$$= \left\{ x \mapsto y \;\middle|\; \exists z \in P, w \in Q.\, (x, z) \overset{\mathfrak{A}}{\mapsto} (y, w) \right\}$$

$(\mathrm{id}_Y \times Q^\dagger) \circ \mathfrak{A} \circ (\mathrm{id}_X \times P)$

3.2.2 Possible Tasks

As mentioned earlier, conceivable tasks are theory-independent whereas possible tasks are theory-dependent. A choice of substrates within a theory of processes is needed to determine possible tasks.

Definition 3.2.7. A *choice of substrates*, represented as $(\mathbf{C}, \Sigma, \Gamma)$, consists of the following components:

1. A reference *theory of processes*, in the form of a strict SMC $\mathbf{C}$ with monoidal product $\otimes$ and unit I.[6] This specifies the theory of conceivable tasks.

2. A choice of *substrates*, in the form of a subset $\Sigma \subseteq \mathrm{obj}(\mathbf{C})$ of systems in the theory of processes.

3. A choice of *sets of substrate states*, in the form of a family $\Gamma = (\Gamma_{\mathtt{H}})_{\mathtt{H} \in \Sigma}$ where $\Gamma_{\mathtt{H}} \subseteq \mathrm{states}_{\mathbf{C}}(\mathtt{H})$ is a set of states in $\mathbf{C}$ for each substrate $\mathtt{H} \in \Sigma$.

The choice of substrates is required to be closed under parallel composition: $I \in \Sigma$ and $\mathtt{H} \otimes \mathtt{K} \in \Sigma$ for all $\mathtt{H}, \mathtt{K} \in \Sigma$. Moreover, the set of substrate states is required to respect parallel composition of substrates: $\Gamma_I = 1$ and $\Gamma_{\mathtt{H} \otimes \mathtt{K}} = \Gamma_{\mathtt{H}} \times \Gamma_{\mathtt{K}}$ for all $\mathtt{H}, \mathtt{K} \in \Sigma$.[7]

For two substrates $\mathtt{H}, \mathtt{K} \in \Sigma$, considering the tasks $\Gamma_{\mathtt{H}} \to \Gamma_{\mathtt{K}}$, one is interested in the question 'which of these tasks are possible within the given theory of processes?'. A task is possible if there exists a *constructor* that can enable the task to be performed. To elaborate and make this statement precise, we need the following definitions.

Definition 3.2.8. Let $(\mathbf{C}, \Sigma, \Gamma)$ be a choice of substrates and consider two substrates $\mathtt{H}, \mathtt{K} \in \Sigma$. A process $f : \mathtt{H} \to \mathtt{K}$ that maps states in $\Gamma_{\mathtt{H}}$ to states in $\Gamma_{\mathtt{K}}$ is called a *task-inducing* process:

$$\forall \rho \in \Gamma_{\mathtt{H}}.\ f(\rho) \in \Gamma_{\mathtt{K}}$$

The task *induced by* f is denoted by $\lfloor f \rfloor$:

$$\lfloor f \rfloor := \{\rho \mapsto f(\rho) \mid \rho \in \Gamma_{\mathtt{H}}\}$$

Definition 3.2.9. Let $(\mathbf{C}, \Sigma, \Gamma)$ be a choice of substrates and consider a task $\mathfrak{A} : \Gamma_{\mathtt{H}} \to \Gamma_{\mathtt{K}}$. The task $\mathfrak{A}$ is *possible* if there are:

[6] For instance, this could be the theory of finite-dimensional quantum systems and unitary transformations. However, as far as we know, the literature on constructor theory only considers **Rel**, in which the tensor product $\otimes$ is the Cartesian product. In contrast, quantum theory requires the tensor product to be the Kronecker product of Hilbert spaces.

[7] This requirement states that the state of a combined system is the ordered set of the subsystem states. This requirement on substrate states to be 'Cartesian' is called 'the principle of locality' [46, 102] in the literature of constructor theory. This requirement ensures that there is explicitly no entanglement arising out of parallel composition of substrates.

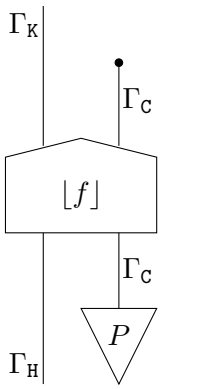
(i) a substrate $\mathtt{C}$ (acting as a *constructor* for the task),

(ii) an attribute $P \subseteq \Gamma_{\mathtt{C}}$ (singling out the relevant constructor states), and

(iii) a task-inducing process $f : \mathtt{H} \otimes \mathtt{C} \to \mathtt{K} \otimes \mathtt{C}$ (actually performing the task)

such that the following two conditions are satisfied:

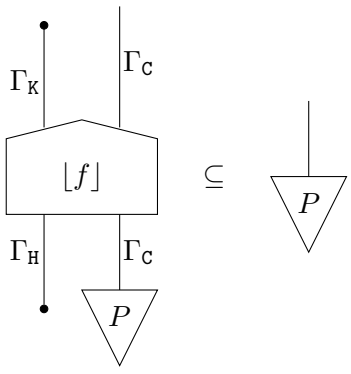
1. The task $\mathfrak{A}$ is obtained from the induced task $\lfloor f \rfloor$ by requiring that the input constructor state has attribute P and discarding the constructor output:

$\mathfrak{A} =$ 

$= \{\rho \mapsto \rho' \mid \exists \gamma \in P, \gamma' \in \Gamma_{\mathtt{C}}.\, f(\rho \otimes \gamma) = \rho' \otimes \gamma'\}$

$(\mathrm{id}_{\Gamma_{\mathtt{K}}} \times \epsilon_{\Gamma_{\mathtt{C}}}) \circ \lfloor f \rfloor \circ (\mathrm{id}_{\Gamma_{\mathtt{H}}} \times P)$

2. The attribute P is preserved by the induced task $\lfloor f \rfloor$. While a particular constructor state $\gamma \in P$ may be modified to become γ' by the underlying process of the induced task $\lfloor f \rfloor$, γ' remains a constructor state for the same induced task $\lfloor f \rfloor$, *i.e.* $\gamma' \in P$. In **Rel**, this constraint is equivalently expressed as the induced task $\lfloor f \rfloor$ sending the set of constructors P to a subset of itself, regardless of the input and output on the substrates $\mathtt{H}, \mathtt{K}$:



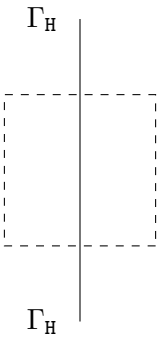
$(\epsilon_{\Gamma_{\mathtt{K}}} \times \mathrm{id}_{\Gamma_{\mathtt{C}}}) \circ \lfloor f \rfloor \circ (\eta_{\Gamma_{\mathtt{H}}} \times P) \;\subseteq\; P$

The set of possible tasks under the given choice of substrates is denoted by $(\mathbf{C}, \Sigma, \Gamma)^{\checkmark}$.

We give the main result of this section in the following proposition. The result shows for a given choice of substrates with **Rel** as the theory of conceivable tasks, the set of possible tasks forms a sub-SMC of **Rel**. In other words, the set of possible tasks is closed under composition in arbitrary (acyclic) networks.

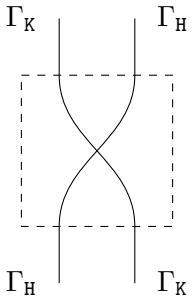
Proposition 3.2.10. For a given choice of substrates $(\mathbf{C}, \Sigma, \Gamma)$ with $\mathbf{C} = \mathbf{Rel}$, the set of possible tasks $(\mathbf{C}, \Sigma, \Gamma)^{\checkmark}$ forms a sub-SMC of **Rel**.

Proof. The identity tasks for all systems are made possible with the identity isomorphisms of **Rel**:



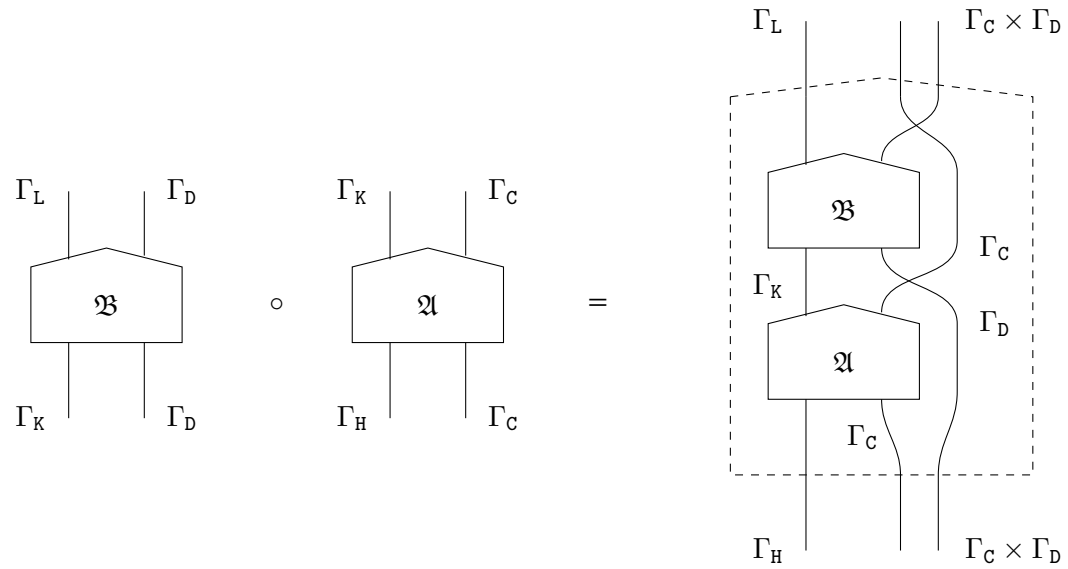
enabled by the trivial constructor $\mathtt{C} := I$.

Likewise, the swap tasks for all systems are made possible by the symmetry isomorphisms of **Rel**, using the trivial constructor $\mathtt{C} := I$:

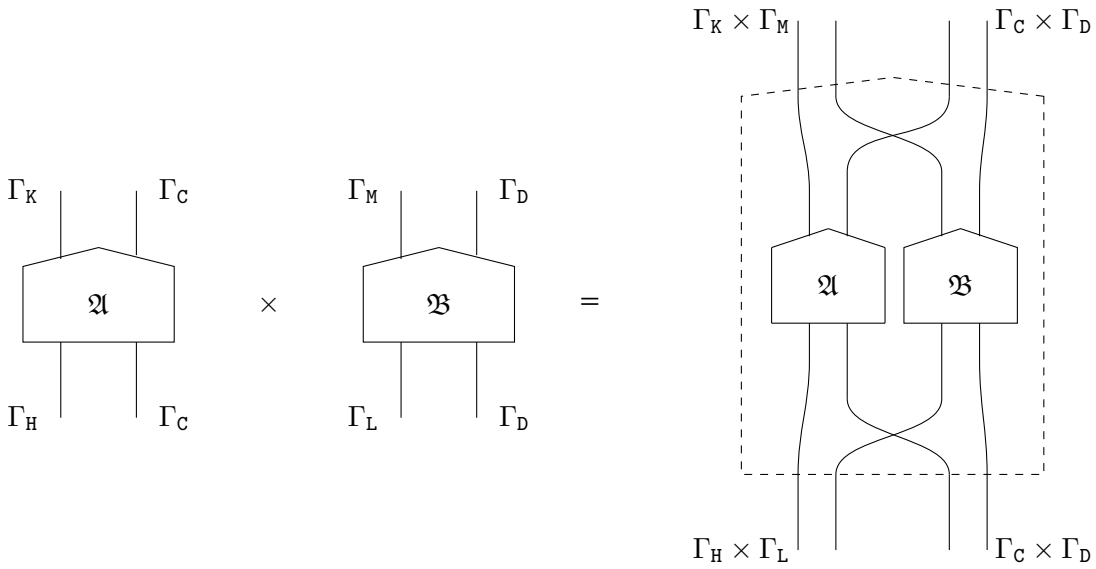


For two possible tasks $\mathfrak{A}$ and $\mathfrak{B}$, having constructors C and D respectively, the sequential

composition $\mathfrak{B} \circ \mathfrak{A}$ is made possible with the constructor $C \otimes D$, as follows:



For two possible tasks $\mathfrak{A}$ and $\mathfrak{B}$, having constructors C and D respectively, the parallel composition $\mathfrak{A} \times \mathfrak{B}$ is made possible with the constructor $C \otimes D$, as follows:



This concludes the proof. □

To recap, taking the theory of conceivable tasks to be **Rel**, we showed that, for a given choice of substrates, the set of possible tasks forms a sub-SMC of **Rel**. Our choice of **Rel** aligns with the constructor theory literature, wherein tasks are modelled by relations between sets [46, 102, 105–107]. While the result may not be surprising, it constitutes, to the best of our knowledge, the first process-theoretic formalisation of constructor theory.

3.2.3 Attributes as states

As originally proposed, constructor theory defined tasks as transformations of states [46]. Later literature defined tasks as acting on the attributes of states instead of on the underlying states [102, 105–107]. This was motivated by the idea that the abstract specification of (possible) tasks should be based on the observable 'macrostates' (*i.e.,* attributes, or subsets of a set) of a physical system, instead of on the unobservable 'microstates' (*i.e.,* states, or elements of a set) that constitute them. In Ref. [48], it was shown that the attribute-focused perspective is derivable from the state-focused perspective in a compositionally sound way via a suitable coarse-graining. We summarise the main insights here.[8]

A notion of 'coarse-graining' is defined for tasks. This coarse graining is in order to obtain tasks transforming attributes (macrostates) from tasks transforming states (microstates). The attributes are allowed to have non-trivial overlap. More specifically, the attributes do not need to form a partition, but nesting of attributes is not allowed. That is, no attribute can be a proper superset of another attribute.[9] With these requirements, coarse-grained tasks are defined and the following results are obtained.

First, it is proved that for any process theory of tasks, the coarse-grained tasks can also be arranged into a process theory. This implies that the tasks defined on attributes are just as compositionally sound as those defined on states. Second, the tasks defined on states can be compositionally embedded into the universe of coarse-grained tasks. This proves that coarse-grained tasks are a sound generalisation of the tasks originally defined on states. Third, the coarse-grained tasks can be embedded back into the universe of ordinary tasks. This shows that the ordinary tasks—*i.e.* those defined on states—are as expressive as coarse-grained tasks.

To summarise, a distinction between states (microstates) and attributes (macrostates) is made in the constructor theory literature for dealing with theories such as thermodynamics and information theory [102, 107]. In categorical semantics, as discussed above, this corresponds to the distinction between ordinary tasks defined on states and coarse-grained tasks defined on attributes. The distinction may or may not make sense in a physical setting. However, mathematically speaking, there is no distinction between the two approaches since they are equivalent as far as their expressivity is concerned.

[8] We do not include all details and provide only a summary, as the author of this thesis was not much involved in this part of the work.

[9] This ensures that the attributes are distinguishable in a strong sense, as required by constructor theory [102].

3.3 Locality in Quantum Theory

There is an overlap between the research areas of constructor theory and process theories as applied to quantum theory *i.e.* CQM. However, there also exists a bone of contention between the two areas: the principle of locality, which lies at the core of constructor theory [46, 102] but is rejected by CQM. In this section, we discuss the principle of locality and how it appears in the Deutsch-Hayden approach to quantum theory [49]. We also argue that Deutsch-Hayden locality is not compositional. This means that employing Deutsch-Hayden locality leads to a conflict between two main principles of constructor theory: composition and locality.

Constructor theory, since its conception, has stuck by 'the principle of locality' [46], according to which the state of a joint system is the ordered set of the states of its subsystems. This is implemented in categorical semantics by imposing the 'Cartesian' requirement on substrate states: $\Gamma_{\mathtt{H}\otimes\mathtt{K}} = \Gamma_{\mathtt{H}} \times \Gamma_{\mathtt{K}}$ for all $\mathtt{H}, \mathtt{K} \in \Sigma$ in Definition 3.2.7. The need for keeping locality in a physical theory is debatable. However, from a mathematical point of view, the generic tensor products for symmetric monoidal categories conservatively and very fruitfully generalise Cartesian products of sets. This is especially important in physical theories like quantum theory.

Below we provide more background on the principle of locality, and how it is made to work in quantum theory. Finally, we show how implementing the principle of locality renders quantum theory non-compositional.

3.3.1 The Principle of Locality

Constructor theory takes seriously Einstein's notion of locality [111], which comprises two components as described by Don Howard:

"The first, which I call the 'separability principle', asserts that any two spatially separated systems possess their own separate real states. The second, the 'locality principle' asserts that all physical effects are propagated with finite, subluminal velocities, so that no effects can be communicated between systems separated by a space-like interval." [112, p. 3]

The separability condition in constructor theory is stronger than the one in the passage quoted above. According to this condition, the whole is separable into independent parts, and these parts when combined make the whole. In other words, 'the whole is not more than the union of the parts' [113, p. 15].

Focusing on the separability part of Einstein's notion of locality, Deutsch posits that it is 'a necessary condition for tasks to be composable into networks' [46, p. 24] according to the composition principle [46]. In constructor-theoretic terms, separability means that the joint state of any two substrates can always be expressed as an ordered pair of individual states of the two substrates; in Definition 3.2.7, this was imposed as the following requirement on substrate states: $\Gamma_{\mathtt{H}\otimes\mathtt{K}} = \Gamma_{\mathtt{H}} \times \Gamma_{\mathtt{K}}$ for all $\mathtt{H}, \mathtt{K} \in \Sigma$.

Einstein locality (read: separability) is uncontroversial in classical physics. However, in the standard formulation of quantum theory, there are entangled states that are non-separable by definition. Deutsch and Hayden came up with a formulation of quantum theory that is arguably consistent with Einstein locality [49]. The existence of this so-called 'local' formulation [49, 114] is offered as an evidence of the compatibility of quantum theory with constructor theory. Below, we give a brief overview of this formulation [49].

3.3.2 Deutsch-Hayden Descriptors

The Deutsch-Hayden formulation of quantum theory describes qubits not in terms of states but in terms of *descriptors*. Roughly speaking, a descriptor of a qubit is a set of operators that determines all the observables of that system. Since the Deutsch-Hayden descriptors are based on the Heisenberg picture of quantum theory, we first describe how it differs from the more commonly used Schrödinger picture. Our exposition follows that of Ref. [115].

Consider a pure state $|\psi\rangle$. Such a state could be prepared by applying a unitary operation U to the state $|0\rangle$, where the latter is deemed an initial state. In standard quantum theory, the expectation of measurement outcomes is computed using the formula

$$\langle 0|U^\dagger O U|0\rangle$$

where $O \equiv \sum_i \lambda_i |\phi_i\rangle \langle\phi_i|$ represents a general observable. The set of vectors $\{|\phi_i\rangle\}_i$ represents the measurement basis and λ_i are the corresponding eigenvalues. The *Schrödinger picture* reads the above formula as the evolution of states while the observable does not change: $\left(\langle 0| U^\dagger\right) O \left(U |0\rangle\right)$. In other words, the state $|0\rangle$ evolves to the state $U|0\rangle$ while O does not evolve. In contrast, the *Heisenberg picture* views the observable as evolving whereas the state remains fixed: $\langle 0| \left(U^\dagger O U\right) |0\rangle$. That is, O changes to $U^\dagger O U$ while the state remains fixed to $|0\rangle$. As the state remains fixed, it is called the *reference vector*.

In short, the Heisenberg picture describes quantum systems not in terms of the evolution of their states but in terms of the evolution of their observables.

The Deutsch-Hayden approach describes a qubit via a mathematical object that encapsulates information about *all* the evolved observables [49, 115]. This task is made tractable by the fact that observables are linear operators, and linear operators form a vector space. As the evolution of the general observable O to $U^\dagger OU$ is linear, one needs to track the evolution of only the basis operators of O. That is, if $O = \sum_k a_k A_k$ where $\{A_k\}_k$ are the basis operators of O, we have $U^\dagger OU = \sum_k a_k U^\dagger A_k U$. Therefore, it is sufficient to track evolution of each basis operator A_k to determine the evolution of any observable by U.

Descriptor of a Single Qubit

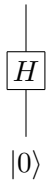
Observables of a single qubit are given by 2×2 matrices, for which a suitable basis is provided by the Pauli matrices along with the identity:

$$\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z) = \left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \right) \text{ and } \sigma_0 = \mathbb{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The evolution of $\mathbb{1}$, *i.e.*, $U^\dagger \mathbb{1} U = \mathbb{1}$ is trivial and can be ignored. Hence, one needs to keep track of the evolution of $\boldsymbol{\sigma}$ to determine any evolved observable of the single qubit. So, the *descriptor* of the single qubit after unitary evolution U is given by

$$\boldsymbol{q} = U^\dagger \boldsymbol{\sigma} U$$

Example 3.3.1. [115] Consider a circuit in which a state is initialised as $|0\rangle$ and then subjected to a Hadamard gate:



We consider the state $|0\rangle$ to be fixed and give a description of the quantum system in terms of the evolution of its descriptor. The unitary of the Hadamard is given by $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. The descriptor is initially $\boldsymbol{q}^0 = \boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$, which after the application of the Hadamard becomes $\boldsymbol{q}^1 = H^\dagger \boldsymbol{\sigma} H = H^\dagger (\sigma_x, \sigma_y, \sigma_z) H = (\sigma_z, -\sigma_y, \sigma_x)$. The evolution of the observable $|0\rangle\langle 0|$ can be determined using the descriptors. After the Hadamard is applied, the observable becomes

$$H^\dagger |0\rangle\langle 0| H = H^\dagger \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} H = H^\dagger \left(\frac{\mathbb{1} + \sigma_z}{2} \right) H = H^\dagger \left(\frac{\mathbb{1} + q_z^0}{2} \right) H = \frac{\mathbb{1} + q_z^1}{2} = \frac{\mathbb{1} + \sigma_x}{2}$$

The expectation of this observable can be calculated using the reference vector $|0\rangle$ as follows:

$$\langle 0| H^\dagger |0\rangle \langle 0| H |0\rangle = \langle 0| \frac{\mathbb{1} + \sigma_x}{2} |0\rangle = \frac{1}{2}$$

which is the probability of obtaining the measurement outcome $|0\rangle$.

Descriptor of n Qubits

Consider n qubits prepared in the state $|0\rangle^{\otimes n}$ and subjected to a unitary operation U. We are interested in describing the evolution of all the observables of this quantum system. For n qubits, the observables are given by $2^n \times 2^n = 4^n$ dimensional complex matrices. A suitable basis for these is given by products of Pauli operators

$$\mathcal{A} \equiv \{\sigma_{\mu_1} \otimes \sigma_{\mu_2} \otimes \cdots \sigma_{\mu_n} \mid \mu_i \in \{0, x, y, z\}\},$$

which are linearly independent and 4^n in number. This implies that by tracking the evolution of each operator in the basis $\mathcal{A}$ from $\sigma_{\mu_1} \otimes \sigma_{\mu_2} \otimes \cdots \sigma_{\mu_n}$ to $U^\dagger \sigma_{\mu_1} \otimes \sigma_{\mu_2} \otimes \cdots \sigma_{\mu_n} U$, one can determine the evolution of any observable.

Instead of tracking all the 4^n basis observables, one may track the evolution of the following set of observables

$$\boldsymbol{q}_i^0 = \mathbb{1}^{i-1} \otimes \boldsymbol{\sigma} \otimes \mathbb{1}^{n-i}, \qquad i = 1, \cdots, n,$$

where $\mathbb{1}^k$ represents the tensor product of k copies of the 2×2 identity matrix. $\boldsymbol{q}_i^0$ has three components for each i. These $3n$ observables defined above can be multiplied to obtain any of the 4^n basis observables [113, 115]. $\boldsymbol{q}_i^0$ is the *descriptor* of qubit i at time 0. The time-evolved descriptor at time t after unitary evolution is given by

$$\boldsymbol{q}_i^t = U^\dagger \boldsymbol{q}_i^0 U$$

The n-tuple with $\boldsymbol{q}_i^0$ as components is denoted as $\boldsymbol{q}^0$. It is the joint descriptor of n qubits. The operators in $\boldsymbol{q}^0$ satisfy the Lie algebra $\mathfrak{su}(2)^{\otimes n}$:

$$[q_{iw}^0, q_{jw'}^0] = 0 \qquad (i \neq j \text{ and } \forall w, w' \in \{0, x, y, z\})$$

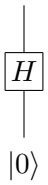
$$q_{ix}^0 q_{iy}^0 = i q_{iz}^0 \qquad \text{(and its cyclic permutations)}$$

$$(q_{iw}^0)^2 = \mathbb{1} \qquad (\forall w \in \{0, x, y, z\})$$

which is preserved by unitary evolution. These algebraic relations imply that for any i, there is a redundancy in the observables. That is, to completely determine any triple of observables $(q^0_{ix}, q^0_{iy}, q^0_{iz})$, one needs only two components as the third can be calculated from the other two using the algebraic relations. Therefore, one may avoid tracking the y component of each descriptor. Hence, one needs to track evolution of $2n$ observables to completely determine the evolution of an n-qubit system.

To summarise, for an n-qubit system, the Deutsch-Hayden approach focuses on tracking the evolution of *all* the observables from O to $U^\dagger OU$. This is achieved by tracking the evolution of $2n$ basis observables (each of dimension $2^n \times 2^n$) from $\boldsymbol{q}^0$ to $\boldsymbol{q}^t = U^\dagger \boldsymbol{q}^0 U$, from which the 4^n observables of the basis $\mathcal{A}$ can be determined. The 4^n observables can, in turn, be used to determine the evolution of any observable.

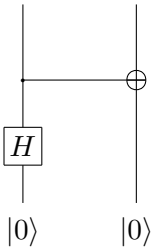
Example 3.3.2. Consider again the circuit of Example 3.3.1



where a state is initialised as $|0\rangle$ and then subjected to a Hadamard gate.

Based on the foregoing discussion, after fixing $|0\rangle$ as the reference vector, one needs to track evolution of only two basis observables (σ_x, σ_z) to fully characterise the circuit. In other words, $\boldsymbol{q}^0 = (\sigma_x, \sigma_z)$ represents the descriptor for the single qubit before the application of the Hadamard gate. After the Hadamard is applied, the descriptor becomes $\boldsymbol{q}^1 = H^\dagger(\sigma_x, \sigma_z)H = (\sigma_z, \sigma_x)$.

Example 3.3.3. Consider the following circuit, which is usually used to create the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$:



In this circuit, two qubits are initialised in the $|00\rangle$ state. The first qubit is subjected to a Hadamard gate. This is followed by an application of a CNOT gate to the two qubits, where the first qubit acts as the control and the second the target.

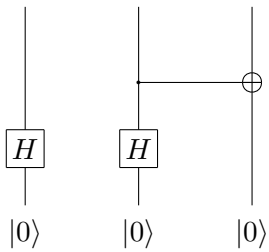
We fix $|00\rangle$ as the reference vector, and track evolution of two descriptors initially given by $\boldsymbol{q}_1^0 = (q_{1x}, q_{1z}) = (\sigma_x \otimes \mathbb{1}, \sigma_z \otimes \mathbb{1})$ and $\boldsymbol{q}_2^0 = (q_{2x}, q_{2z}) = (\mathbb{1} \otimes \sigma_x, \mathbb{1} \otimes \sigma_z)$. The joint description of the two systems is given by the Cartesian product of the descriptors of the two qubits $\boldsymbol{q}_1^0$ and $\boldsymbol{q}_2^0$, *i.e.,* $\boldsymbol{q}^0 = (\boldsymbol{q}_1^0, \boldsymbol{q}_2^0)$. After the application of the Hadamard but before that of the CNOT, the descriptors are given by $\boldsymbol{q}_1^1 = (H \otimes \mathbb{1})^\dagger(\sigma_x \otimes \mathbb{1}, \sigma_z \otimes \mathbb{1})(H \otimes \mathbb{1}) = (\sigma_z \otimes \mathbb{1}, \sigma_x \otimes \mathbb{1}) = (q_{1z}, q_{1x})$ and $\boldsymbol{q}_2^1 = (H \otimes \mathbb{1})^\dagger(\mathbb{1} \otimes \sigma_x, \mathbb{1} \otimes \sigma_z)(H \otimes \mathbb{1}) = (\mathbb{1} \otimes \sigma_x, \mathbb{1} \otimes \sigma_z) = (q_{2x}, q_{2z})$. As expected, there is no change in the descriptor of the second qubit because the Hadamard acts on the first qubit only. At this point, the joint description of the two systems is given by $\boldsymbol{q}^1 = (\boldsymbol{q}_1^1, \boldsymbol{q}_2^1)$. After the application of the CNOT gate, the unitary of which is $U_{\text{CNOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, the qubit descriptors are given by $\boldsymbol{q}_1^2 = (H \otimes \mathbb{1})^\dagger U_{\text{CNOT}}^\dagger(\sigma_x \otimes \mathbb{1}, \sigma_z \otimes \mathbb{1}) U_{\text{CNOT}}(H \otimes \mathbb{1}) = (\sigma_z \otimes \sigma_x, \sigma_x \otimes \mathbb{1}) = (q_{1z} q_{2x}, q_{1x})$ and $\boldsymbol{q}_2^2 = (H \otimes \mathbb{1})^\dagger U_{\text{CNOT}}^\dagger(\mathbb{1} \otimes \sigma_x, \mathbb{1} \otimes \sigma_z) U_{\text{CNOT}}(H \otimes \mathbb{1}) = (\mathbb{1} \otimes \sigma_x, \sigma_x \otimes \sigma_z) = (q_{2x}, q_{1x} q_{2z})$. The joint descriptor of the two-qubit system is $\boldsymbol{q}^2 = (\boldsymbol{q}_1^2, \boldsymbol{q}_2^2)$.

3.3.3 Locality and Compositionality

Here, we see how locality as implemented in the Deutsch-Hayden approach interacts with the composition of circuits.

Remark 3.3.4. In Examples 3.3.2 and 3.3.3, we characterised qubits in terms of their descriptors. Example 3.3.2 involved a single qubit and thus required only one descriptor, the unitary evolution of which was tracked. Example 3.3.3 was about two qubits and hence there were two descriptors to be tracked. Moreover, the descriptor of the joint two-qubit system was obtained by taking the Cartesian product of the individual descriptors of the two qubits.

Consider the parallel composition of the circuits from Examples 3.3.2 and 3.3.3.



If we want to characterise this circuit in terms of qubit descriptors, we cannot use those from Examples 3.3.2 and 3.3.3. The above circuit is a three-qubit system and requires that all the

descriptors be of dimension $2^3 \times 2^3$ whereas the descriptors in Examples 3.3.2 and 3.3.3 are of dimensions 2×2 and $2^2 \times 2^2$ respectively.

More explicitly, from Example 3.3.2, we have the evolved descriptor of the first qubit after the application of the Hadamard gate: $\boldsymbol{q}_1^2 = (\sigma_z, \sigma_x)$. From Example 3.3.3, we have the descriptor of the joint system comprising the second and third qubits after the application of the Hadamard and CNOT gates: $(\boldsymbol{q}_2^2, \boldsymbol{q}_3^2) = ((\sigma_z \otimes \sigma_x, \sigma_x \otimes \mathbb{1}), (\mathbb{1} \otimes \sigma_x, \sigma_x \otimes \sigma_z))$. The descriptor of the joint system comprising the three qubits after the application of all the unitary gates is *not* given by the tuple $(\boldsymbol{q}_1^2, \boldsymbol{q}_2^2, \boldsymbol{q}_3^2) = ((\sigma_z, \sigma_x), (\sigma_z \otimes \sigma_x, \sigma_x \otimes \mathbb{1}), (\mathbb{1} \otimes \sigma_x, \sigma_x \otimes \sigma_z))$. Instead, it is given by $((\sigma_z \otimes \mathbb{1} \otimes \mathbb{1}, \sigma_x \otimes \mathbb{1} \otimes \mathbb{1}), (\mathbb{1} \otimes \sigma_z \otimes \sigma_x, \mathbb{1} \otimes \sigma_x \otimes \mathbb{1}), (\mathbb{1} \otimes \mathbb{1} \otimes \sigma_x, \mathbb{1} \otimes \sigma_x \otimes \sigma_z))$, which is not the Cartesian product of the descriptors from Examples 3.3.2 and 3.3.3. In this sense, the descriptors of the Deutsch-Hayden approach are not compositional.

Example 3.3.3 and Remark 3.3.4 show that the descriptors of two systems can be composed using the Cartesian product only if they are defined with reference to the same circuit. If two circuits, the descriptors of which we have individually tracked, are composed in parallel, the Cartesian product of their descriptors does not give the descriptor of the overall circuit. In fact, as soon as the two circuits are put together in parallel, the old descriptors no longer apply to the individual circuits. Instead, one needs to define and track new descriptors. This shows that the descriptors are not compositional.

Remark 3.3.5. The presence of another circuit, or even just a qubit, changes the descriptors of the original circuit even if there is no interaction between the two circuits. In this sense, the Deutsch-Hayden descriptors are not *local*.

Remark 3.3.6. Two key principles of constructor theory are the composition principle and the principle of locality [46, 102]. The composition principle states that every regular network of possible tasks is a possible task, whereas the principle of locality states that the complete description of the system is given by the Cartesian product of the description of its parts. For the latter principle to be applicable to quantum theory, the Deutsch-Hayden formulation [49, 114] is quoted as a supporting framework. Our analysis in Example 3.3.3 and Remark 3.3.4 has shown that there is a clash between the two principles if the Deutsch-Hayden approach is used. In quantum theory, preparing qubits in $|0\rangle$ states and applying unitary operations on them, like we did in Examples 3.3.2 and 3.3.3, are possible tasks. The parallel composition

of these tasks, like the one in Remark 3.3.4, is a regular network and, hence, a possible task according to the composition principle. However, the example in Remark 3.3.4 shows that the locality principle is violated—the descriptor of the joint system is not obtainable by the Cartesian product of the descriptors of its parts.

Remark 3.3.7. Another way to describe the tension between composition and locality is as follows. Preparing circuits like those in Examples 3.3.2 and 3.3.3 are possible tasks. The 'states' of these tasks are given by the descriptors. The principle of locality states that the ordered tuple of these 'states' gives the 'state' of the joint task obtained by composing the individual tasks in parallel. However, taking the ordered tuple of the 'states' does not result in any legitimate 'state'.

The foregoing discussion shows that constructor theorists need to reject either of the two key principles, namely locality or composition, or find a completely local formulation of quantum theory that is compatible with the principle of composition. Otherwise, the claim that constructor theory is a deeper or more fundamental theory than quantum theory does not hold.

3.4 Constructors in Categorical Quantum Mechanics

The central idea in constructor theory is that of *possibility*, understood as what *could* happen. Constructor theory and process theories are alike in the sense that they have a common lineage of counter-factual reasoning, which can be traced back to the distinction between 'actuality' and 'potentiality' by Aristotle [116].

In constructor theory, theories are characterised by defining the dichotomy between possible and impossible tasks. The impossibility of cloning a general state has been suggested to yield quantum theory, at least in a broad sense [102]. In CQM [22, 23, 55], back in at least 2006, classicality was indeed defined by the ability to clone [117]. This led to the development of spiders [118] and the ZX-calculus [78, 79]. As discussed in Chapter 2, the latter is now a prominent formalism in quantum computing and technologies[10] as well as in quantum science education [44, 45].

When it is formulated as a concrete SMC, a process theory is about possible and impossible processes that obey the axioms of the corresponding category. The reconstructions of quantum

[10]The website `https://zxcalculus.com/publications.html` lists 300+ publications related to the ZX-calculus in some way.

theory in terms of process theories convert these categorical axioms into physical postulates that are considered reasonable [57, 119].

In the previous section, we showed that the Deutsch-Hayden formulation of quantum theory obeys Einstein locality at the expense of compositionality, thereby weakening the claim to fundamentality of constructor theory. Dispensing with the principle of locality in constructor theory, one can formulate quantum theory in constructor-theoretic terms. In this section, we show that CQM achieves just that. Here, we conceive CQM as a fully compositional theory of possible tasks in which parallel composition is given by the Kronecker product. We present examples and constructor-theoretic explanations of possible tasks in quantum theory and quantum computation within the diagrammatic language of CQM. Some of these examples correspond to those found in the literature of constructor theory.

Convention 3.4.1. The notation for CQM/ZX-diagrams in this section follows that introduced in Chapter 2. In diagrammatic equations and calculations, all equalities are up to non-zero global scalars.

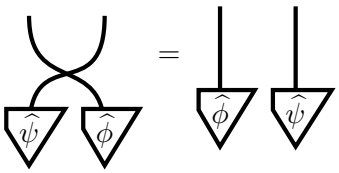
Remark 3.4.2. As discussed in Section 3.2.3, state-based and attribute-based approaches to constructor theory are equally expressive. Here, we stick to the state-based perspective.

3.4.1 Tasks with Trivial Constructors

We start with examples that involve trivial constructors. In Ref. [102], swapping two states is mentioned as an example of a reversible computation task.

Example 3.4.3. Swapping two quantum states is a possible task. Such a task comes for free in a process theory such as CQM. It is represented by

Applying this task to any two states $\widehat{\psi}$ and $\widehat{\phi}$ gives



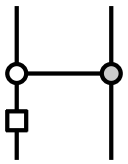
This task requires two substrates that are the two quantum systems whose states are to be swapped, but does not require any explicit constructor. In other words, this task is made possible by the trivial constructor I.

Example 3.4.4. Transforming a $\widehat{0}$ state into a $\widehat{1}$ state or vice versa is a possible task, performed by a NOT gate, represented by

This task requires no explicit constructor. That is, this task is made possible by the trivial constructor I.

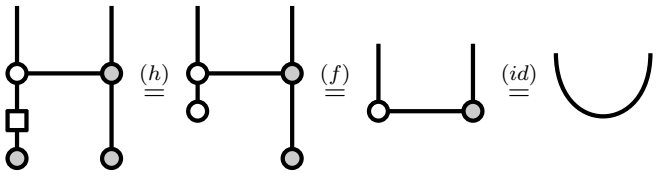
The Deutsch-Hayden approach [49] was originally introduced to provide a local formulation of scenarios involving entangled states, such as the Einstein-Podolsky-Rosen (EPR) experiment and the quantum teleportation protocol. In these scenarios, a Bell state is generated by preparing two systems, each in the $\widehat{0}$ state, and applying the Hadamard and CNOT gates to them. It is another example of a possible task that does not require an explicit constructor.

Example 3.4.5. The task of transforming two states $\widehat{0}$ and $\widehat{0}$ into the Bell state is given by



(3.1)

where the first system undergoes a Hadamard transformation. This is followed by the application of a CNOT gate to the two systems. The first system is the control whereas the second is the target. Supplying the input states to this task, we get



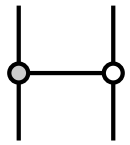
In CQM, the Bell state is non-separable and hence CQM does not obey the principle of locality. However CQM is a completely compositional theory, unlike the Deutsch-Hayden formulation of quantum theory [49]. This aspect shall be exemplified and further explained later.

Remark 3.4.6. Process (3.1) is itself a composite process. It is obtained by the parallel composition of the Hadamard and identity gates, followed by a serial composition with the CNOT gate.

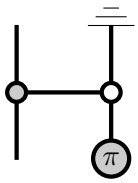
3.4.2 Tasks with Explicit Constructors

In Ref. [120], there is an example of a possible task requiring an explicit constructor, which we revisit using CQM below.

Example 3.4.7. Consider the task of transforming a $\widehat{0}$ state into a $\widehat{1}$ state using a CNOT gate. This task is made possible by the state $\widehat{1}$ which acts as a constructor.
Here, the CNOT gate acts as a task-inducing process.

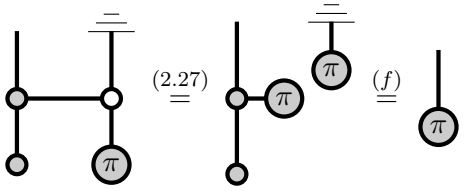


The first input is the target and takes the substrate state to be transformed. The second input is the control and takes the constructor state. The task is obtained from the task-inducing process by requiring that the input constructor state is $\widehat{1}$ and discarding the constructor output. This is given by the diagram

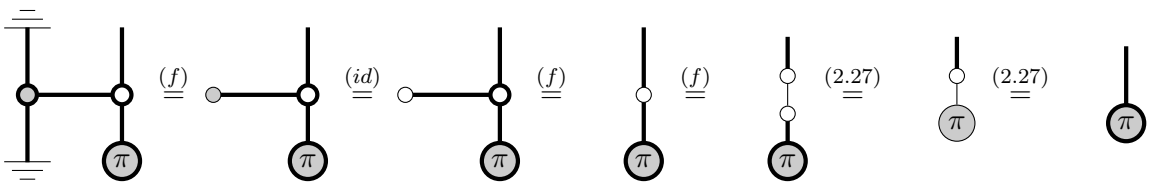


(3.2)

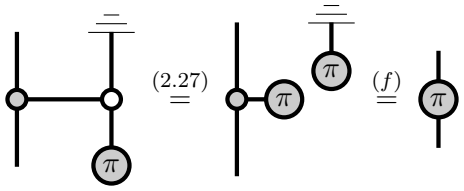
In agreement with the conditions of a possible task according to Definition 3.2.9, it can be verified that this task actually transforms a substrate state from $\widehat{0}$ into $\widehat{1}$



and that the constructor state is preserved by the induced task:

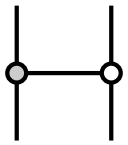


Remark 3.4.8. Task (3.2) can be simplified as follows

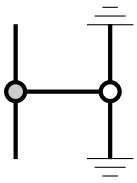


which gives the NOT gate, discussed in Example 3.4.4. This shows that the task of transforming $\widehat{0}$ to $\widehat{1}$ using a CNOT gate requires an explicit constructor while the same task does not require any explicit constructor when one uses a NOT gate. Once the constructor is supplied to a CNOT gate at the input and the constructor output is discarded, we get a NOT gate.

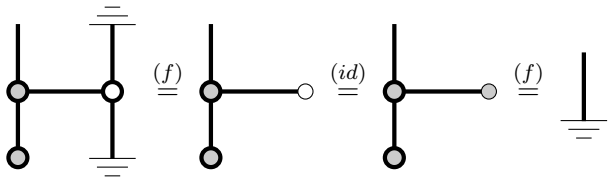
Example 3.4.9. Consider the task of transforming a $\widehat{0}$ state into the maximally mixed state. This task can be performed by a CNOT gate



and is made possible by a maximally mixed state acting as the constructor. Here, the CNOT gate is the task-inducing process and the required task is obtained by requiring that the input constructor state is the maximally mixed state and discarding the constructor output. This is given by the diagram

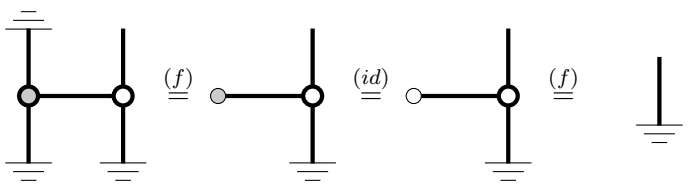


The conditions of a possible task according to Definition 3.2.9 can be verified. We plug the state $\widehat{0}$ at the task input and simplify



confirming that the task performs the required transformation.

To check whether the constructor is preserved by the task-inducing process, we have



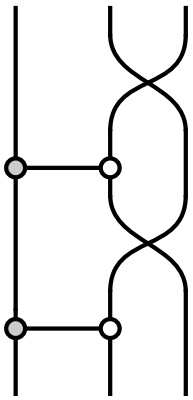
This shows that the constructor is preserved by the process, and hence the task under discussion is a possible task.

3.4.3 Composition of Tasks

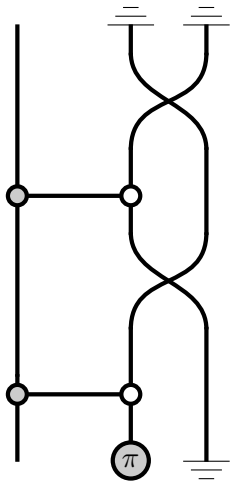
According to the principle of composition [46, 102], series and parallel compositions of possible tasks result in possible tasks. We provide examples of series and parallel composition below.

Example 3.4.10. Consider two possible tasks: (a) that of transforming a $\widehat{1}$ state into $\widehat{0}$ using a CNOT gate. This task is similar to that of Example 3.4.7, except that the input state to be transformed is $\widehat{1}$ instead of $\widehat{0}$; and (b) the task from Example 3.4.9, which transforms $\widehat{0}$ into the maximally mixed state.

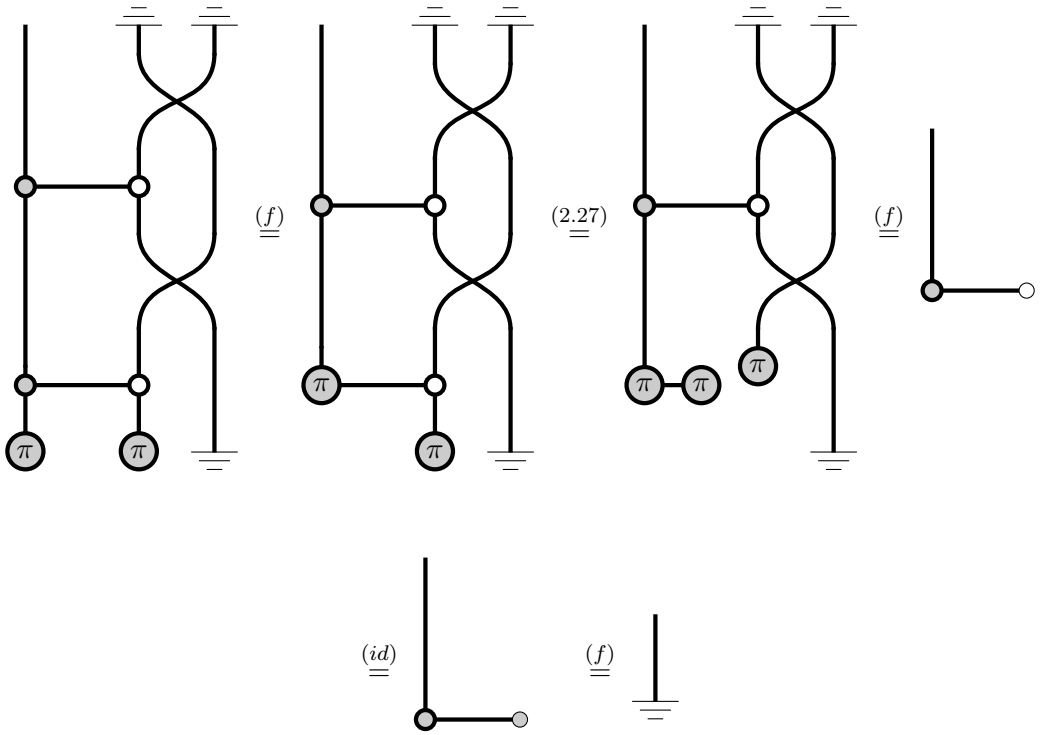
We are interested in the serial composition of task (a) followed by task (b). The serial composition of the corresponding task-inducing processes is given by



The composite task is obtained by supplying the required constructor inputs and discarding the constructor outputs.

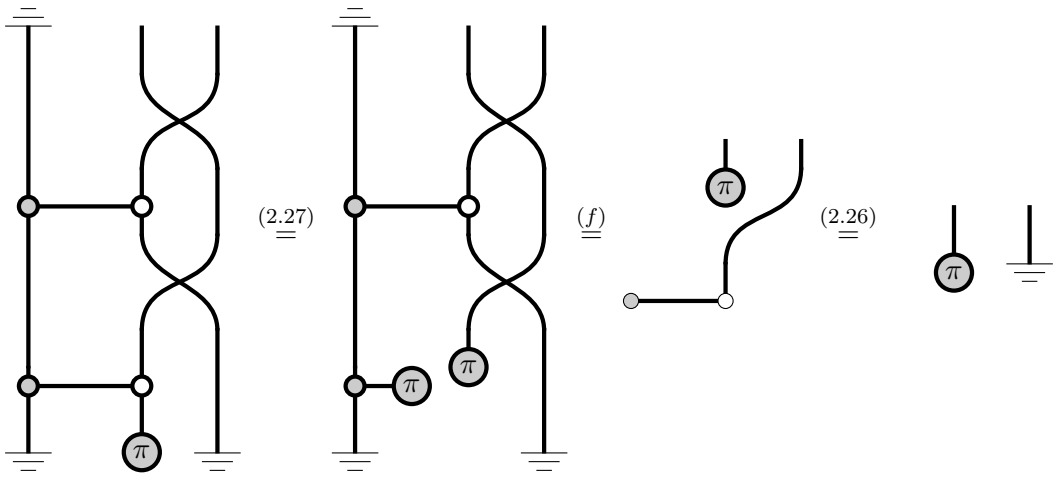


Applying the state $\widehat{1}$ as the input to this task, we have



which shows that the composite task transforms $\widehat{1}$ into the maximally mixed state.

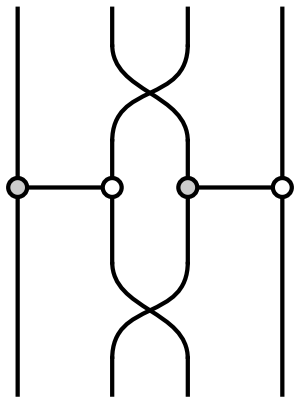
We can also check that the constructor for this task is preserved by the induced task:



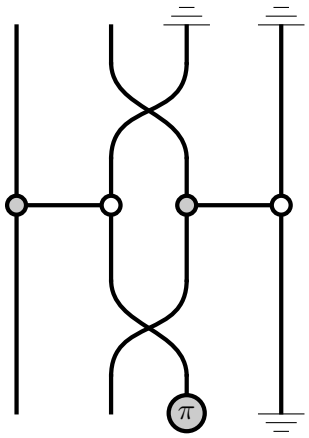
In this example, the constructor of the composite process is the tensor product of $\widehat{1}$ and the maximally mixed state.

Example 3.4.11. Consider the parallel composition of tasks in Examples 3.4.7 and 3.4.9. The

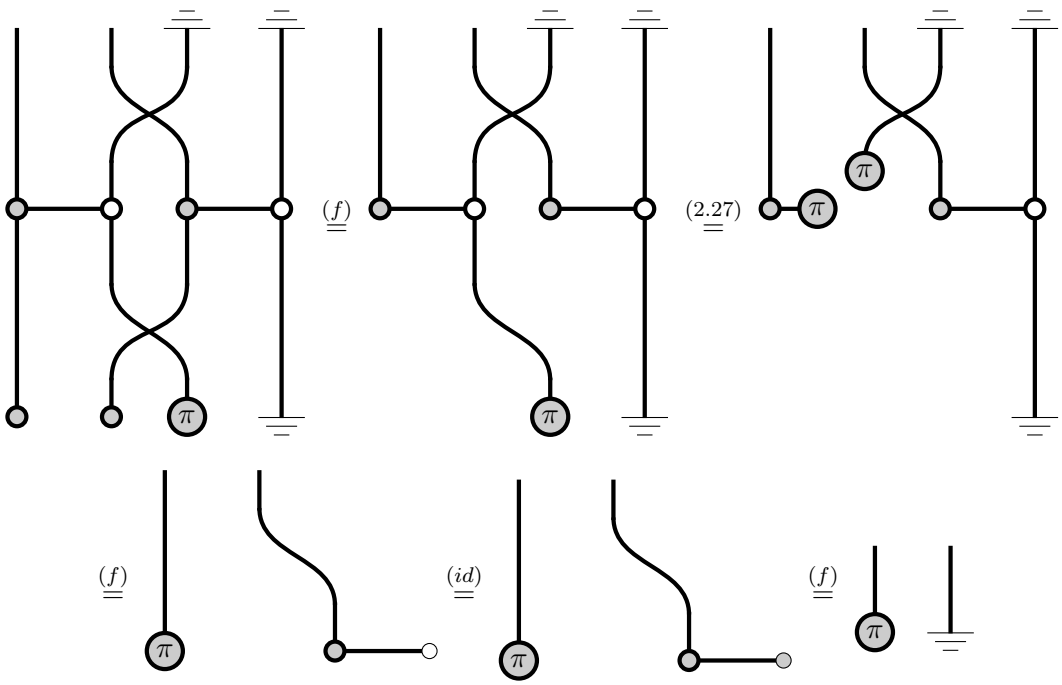
parallel composition of the corresponding task-inducing processes is given by



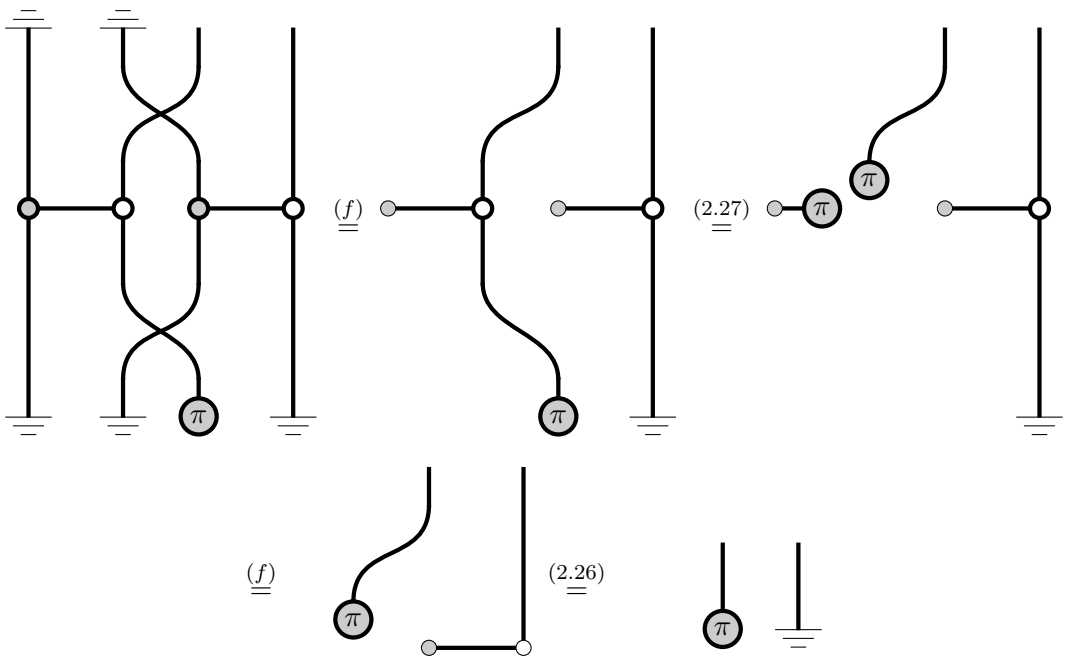
The composite task is obtained by supplying the required constructor inputs and discarding the constructor outputs.



To verify that the task performs the desired transformation, we plug the states $\widehat{0}$ and $\widehat{0}$ at the two inputs of the composite task and simplify the diagram as follows:



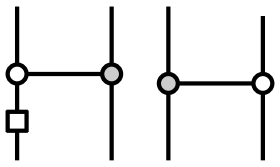
which is the desired output: the tensor product of $\widehat{1}$ and the maximally mixed state. To verify that the constructor for this task is preserved by the induced task, we apply maximally mixed states to the two inputs of the task and discard the task outputs.



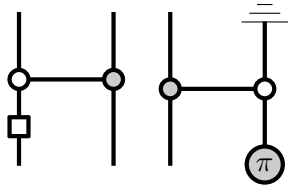
The simplified diagram is the tensor product of $\widehat{1}$ and the maximally mixed state—the constructor for this task. Hence, the constructor is preserved.

Tasks with explicit constructors can be composed with tasks with none. We give an example of one such parallel composition below.

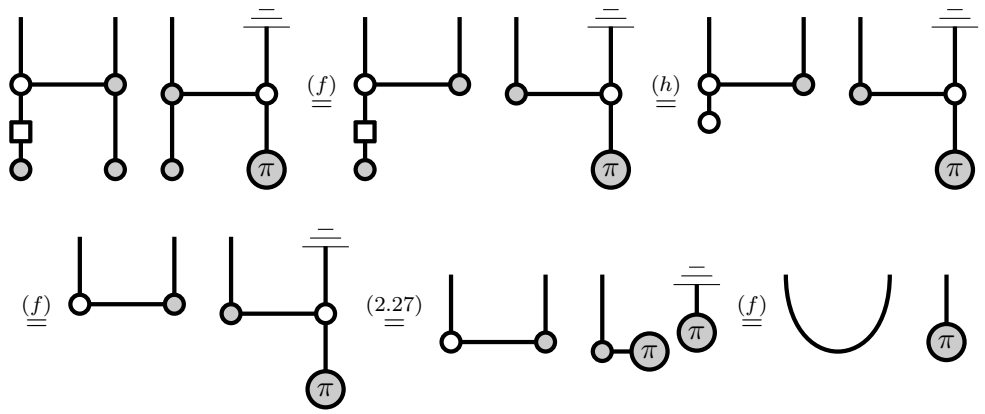
Example 3.4.12. Consider a parallel composition of tasks in Examples 3.4.5 and 3.4.7. The task-inducing process for this composite task is given by



The constructor for this task is the tensor product of the trivial constructor I and $\widehat{1}$ (*i.e.*, $I \otimes \widehat{1} = \widehat{1}$). The composite task is obtained by supplying the required state to the constructor input and discarding the constructor output. This is given by the parallel composition of diagrams (3.1) and (3.2):



Supplying states $\widehat{0}$, $\widehat{0}$, and $\widehat{0}$ to the three inputs, we have



which shows that the task transforms the states $\widehat{0}$, $\widehat{0}$, and $\widehat{0}$ into the tensor product of the Bell state and $\widehat{1}$.

Remark 3.4.13. The output of the task discussed in the above example is a parallel composition of the Bell state and $\widehat{1}$:

(3.3)

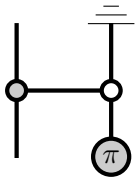
If one subscribes to the principle of locality in constructor theory [46, 102], the Bell state must be expressible in terms of two separable parts. Using the Deutsch-Hayden formalism [49] (which has been quoted in favour of the principle of locality [102]), one can indeed find such an expression for the Bell state. However, as soon as another quantum state is introduced in parallel to the Bell state (such as in diagram (3.3)), the old Deutsch-Hayden descriptors cannot be used anymore, and one needs to use new Deutsch-Hayden descriptors for the Bell state. In this sense, the so-called 'local' description of the Bell state is not compositional. In another sense, it is also non-local: the local descriptors of a system are dependent on the presence of other quantum systems which are not part of the local system to be described. We discussed this in detail in Section 3.3.3.

So, if we desire a compositionally sound constructor theory of quantum physics and quantum information/computation, we cannot use the Deutsch-Hayden formalism for quantum theory as it clashes with the principle of composition [46, 102]. In contrast, CQM offers a constructor theory of quantum physics but one which rejects the principle of locality, as it includes non-separable states such as the Bell state as legitimate inputs and outputs of possible tasks.

3.4.4 Reversibility of Tasks

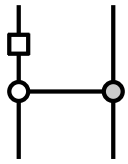
The tasks described in Examples 3.4.3, 3.4.4, 3.4.5, and 3.4.7 are reversible from a constructor-theoretic point of view. That is, the reverse of these tasks (transforming outputs of the original tasks into the corresponding inputs) are made possible by the constructors of the original tasks. Tasks in Examples 3.4.3, 3.4.4, and 3.4.5 are trivially reversible as the processes involved are reversible and do not require explicit constructors. The corresponding reverse tasks are obtained by vertically flipping the diagrams. We provide two examples of reversible tasks below: one with an explicit constructor and one with a trivial constructor.

Example 3.4.14. Consider Example 3.4.7, in which a task transforms a state $\widehat{0}$ into $\widehat{1}$. Its reverse task would transform $\widehat{1}$ into $\widehat{0}$, and is made possible by the same constructor as the one that enabled the original task. It is obtained by vertically flipping the task-inducing process (which changes inputs into outputs and vice versa) and then plugging the constructor state at the constructor input and discarding the constructor output.

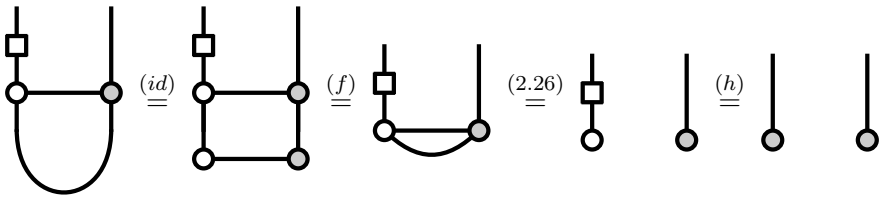


It can be verified that the above process achieves the desired task.

Example 3.4.15. Consider Example 3.4.5 in which a task transforms the tensor product of two states $\widehat{0}$ and $\widehat{0}$ into the Bell state. This task is made possible by the trivial constructor I. Its reverse task would transform the Bell state into the tensor product of $\widehat{0}$ and $\widehat{0}$, and is enabled by the same constructor I. This is obtained by vertically flipping the task-inducing process.

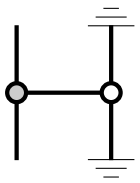


We verify that this process performs the desired reverse task:

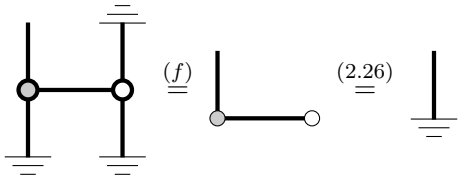


A constructor for a task is not necessarily a constructor for its reverse task. This is called *constructor-based irreversibility* [120, 121]. We provide an example of constructor-based irreversibility below.

Example 3.4.16. In Example 3.4.9, a task transforms $\widehat{0}$ into the maximally mixed state; this task is enabled by a maximally mixed state acting as the constructor. We can check whether the reverse task is made possible by the same constructor. We vertically flip the task-inducing process (which gives the same process as in the original example), plug the constructor state at the constructor input and discard the constructor output. We have



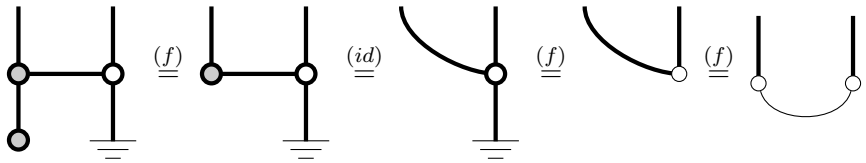
Applying a maximally mixed state as the input to this process, we have



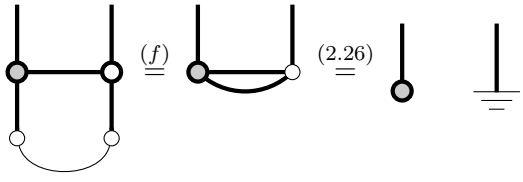
which is not the desired output: the state $\widehat{0}$. Therefore, this is an example of constructor-based irreversibility.

In the above example, the task is not reversible but the task-inducing process is reversible.

Remark 3.4.17. Note that in Example 3.4.9 and, therefore, the above example as well, the task-inducing process is reversible. If we forget the distinction between constructors and regular substrates, and treat $\widehat{0}$ and the maximally mixed state as merely two inputs of the task-inducing process, we get



Now, applying this output state as the input to the vertically-flipped task-inducing process, we have



This shows that the output of the reverse process is the input of the original process, confirming that the task-inducing process is reversible.

In summary, constructor-based irreversibility is the idea that even if the task-inducing process is reversible, the corresponding possible task may not be. A possible task is reversible if its constructor also makes its reverse task possible.

Remark 3.4.18. A foundational question in physics concerns how irreversibility arises from time-reversal symmetric physical laws. Conventional approaches to reconciling irreversibility with reversibility involve statistical mechanical methods [122, 123]. Constructor theory offers a fresh perspective on the emergence of irreversibility in terms of the possibility and impossibility of tasks. Task-inducing processes correspond to time-reversal symmetric physical laws, but that alone is not sufficient for the reversibility of tasks. If the constructor of a possible task does not also act as a constructor for its reverse task (meaning the latter task is impossible), then the task is irreversible—even if the underlying task-inducing process is reversible. Unlike the statistical approaches to irreversibility, which are restricted to macroscopic physics and rely on approximations, constructor-theoretic irreversibility is scale-independent and exact [120, 121].

3.5 Summary and Outlook

In this chapter, we presented categorical semantics for constructor theory. Our focus was on the desired mathematical foundations explained in Deutsch's seminal paper [46]. We took the theory of conceivable tasks to be **Rel**, and then showed that, for a given choice of substrates, the set of possible tasks forms a sub-SMC of **Rel**. While the result may not be surprising, it constitutes, to the best of our knowledge, the first process-theoretic formalisation of constructor theory. This formalisation offers a rigorous yet intuitive language to express and explore constructor-theoretic ideas.

We then identified a key inconsistency in the constructor-theoretic formulation of non-relativistic quantum theory: a conflict between the principles of locality and composition. Our analysis of the Deutsch–Hayden approach [49] demonstrated that locality comes at the cost of compositionality. Finally, we argued that CQM can be considered a viable constructor-theory

of quantum physics—one that is fully compositional but non-local—and illustrated this with examples grounded in quantum theory and computation.

The string diagrammatic syntax we employed to formally represent constructor theory can also be interpreted in other SMCs. This provides an opportunity to investigate and explore the ramifications of constructor theory beyond **Rel**, which has long been the default modelling choice in the constructor theory literature.

Our work suggests promising avenues for future research: namely, developing process-theoretic counterparts of constructor theories of information [102], thermodynamics [107], probability [106], and life [105], as well as exploring the connection between constructor theory and resource theories [124].

4

Waves, Interference, and Computation

> 'Civilization advances by extending the number of important operations which we can perform without thinking about them.'
>
> Alfred North Whitehead,
> An Introduction to Mathematics [125, p. 61]

This chapter is adapted from the publication [50], based on joint work with Alexy Karenowska. The author of this thesis is the first author of the publication [50], and played a leading role in all the work described in this chapter. The novel contributions are presented in Sections 4.3 and 4.4.

4.1 Introduction

The prevalent paradigm of classical computing technology is based on encoding information in electrical voltages or currents and performing computation by manipulating the movement of charge. The building block of classical computers is the complementary metal-oxide-semiconductor (CMOS) transistor. CMOS transistors can be employed as switches and amplifiers [126] with which one can physically implement logic gates to perform any Boolean operation [127, 128].

The practical success of conventional computing has relied on the continuing minituarisation of the CMOS technology [47, 129], increasing the packing density of transistors in integrated circuits (ICs). This trend of progressive miniaturisation and the corresponding increase in the number of transistors on ICs has been famously described in an observation [130, 131] now known as 'Moore's law'. The original observation stated that the number of transistors per circuit chip doubled every year.

Recently, there has been a realisation that Moore's law is not going to hold anymore because of fundamental physical scaling limits [47, 132, 133]. Therefore, there is interest in novel, 'beyond-CMOS' technologies and paradigms [134]. One such paradigm is wave-based computation, which is based on encoding information in the phase and/or amplitude of waves and performing computation through phase shifting and wave interference. There exist a number of physical platforms for implementing wave-based computation. These include optical computing [135–137], neuromorphic computing [138], and spin-wave computing [47].

Wave-based computation is a form of analogue computation, and therefore inherits the main limitations that have long affected analogue systems, such as limited precision, challenges in miniaturisation, poor scalability, restricted programmability, lack of robust memory, limited support for general-purpose computation, and most importantly, noise susceptibility and the unavailability of an effective approach for fault tolerance [139–142]. While recent advances in metamaterials [139] and metasurfaces [143] offer prospects for improving scalability and miniaturisation, these developments do not resolve the fundamental issue of noise susceptibility.

In analogue computing, information is represented by continuous physical quantities. These quantities are inherently sensitive to disturbances caused by thermal noise, device mismatch, and fabrication imperfections. Unlike digital computing, which repeatedly restores signals to well-defined states, analogue-only computing cannot remove noise in this way. As a result, small perturbations accumulate as signals propagate and interact, leading to progressively increasing errors over the course of a computation [139]. Noise therefore remains a fundamental limitation rather than an issue that can be eliminated through engineering alone.

Nevertheless, despite these constraints, wave-based computation remains an interesting and valuable paradigm to study. It offers potential advantages, including parallel computing [144], reversible computing [145, 146], and low energy consumption [147, 148], which may be beneficial for specific computational tasks. Recently, it was experimentally demonstrated that classical wave-based computing (with phase encoding) can execute certain quantum computing algorithms that do not require entanglement [149]. Although this approach cannot compete with quantum computing, it may still provide an efficient alternative to classical digital computing for certain computational tasks [149].

There have been a number of implementations of Boolean logic gates and circuits using different wave-based computing platforms. However, a general theoretical (syntactic) framework to design, analyse and optimise these circuits has thus far been lacking.

In this chapter, we offer such a syntactic framework. We present a string-diagrammatic formalism for wave-based computation with phase encoding. We motivate the approach and demonstrate its application with reference to spin-wave or 'magnonic' circuits. However, the same syntax is applicable to other wave-based technologies that use phase encoding [150–156]. Through the example of spin-wave circuits, we demonstrate the usage of the formalism in designing, analysing and simplifying Boolean logic circuits.

The remainder of the chapter is structured as follows. Section 4.2 provides a brief introduction to spin-wave computing. Section 4.3 presents a string-diagrammatic formalism for spin-wave Boolean circuits. Section 4.4 discusses its applications in the design, analysis, and optimisation of wave-based logic circuits. Section 4.5 offers concluding remarks.

4.2 Spin-wave Computing

Spin wave dynamics (also known as magnonics) is the study of *spin waves*—collective excitations of the electronic spin lattice of magnetic materials [157, 158]. Spin waves behave classically at room temperature but demonstrate quantum behaviour at millikelvin temperatures [159]. The spin-wave quantum is called the *magnon*. Magnonics research is concerned with generating, manipulating and detecting spin waves in order to characterise magnetic materials; or study interactions between magnons, other quasiparticles like photons or phonons, and superconducting qubits [148, 159–161]. Moreover, certain magnonic systems allow for the possibility of propagating spin waves that can be harnessed to design logic devices [161–164].

Magnonic or spin-wave logic devices rely on propagating spin waves, instead of electronic charges, as information carriers [47, 165]. Spin-wave computing is considered attractive for two reasons. Firstly, it offers generation of spin waves of gigahertz frequencies at nanometer wavelengths, leading to the prospect of miniaturised devices with high clock speeds [148]. Secondly, the propagation of spin waves through certain magnetic insulators involves relatively low energy dissipation [165], as compared to the movement of charges in CMOS devices.

There have been multiple implementations of spin-wave-based Boolean logic gates.[1] A NOT gate was first experimentally demonstrated in 2005 [166]. It was soon followed by implementations of XNOR and NAND gates [167]. As the NAND gate is a universal gate, its realisation opened the possibility of implementing any Boolean logical circuit within the spin-wave platform. In the last few years, designs of more complex spin-wave circuits have also been proposed [172–174], including half adders [175], full adders [176], and a 32-bit ripple-carry adder [177].

Wave-based computing, and especially spin-wave computing, offers a very economical implementation of the majority gate [163, 178–180]. The simplest majority gate has three Boolean inputs and one output. The output is equal to the value which is in majority at the input. In conventional CMOS-based computing, the implementation of a majority gate requires a combination of AND and OR gates, equivalent to ~ 10 transistors [148]. In contrast, (spin-)wave computing offers the majority gate as a primitive logic gate. Additionally, controlling one of the inputs of the majority gate results in an AND gate or an OR gate. Spin-wave computing also offers the XOR gate as a primitive. This can potentially replace the CMOS XOR gate that usually comprises multiple AND, OR and NOT gates, or equivalently ~ 8 transistors [148].

Although spin-wave computing shows great promise for energy-efficient and compact logic devices and circuits, several key hurdles remain. Like other forms of analogue computation [139, 140], it is inherently susceptible to noise, signal degradation, and accumulated errors, making reliable operation challenging [47]. Error correction is essential but difficult, particularly because it often involves non-linear processes that can significantly increase energy consumption, thereby undermining the benefits of spin-wave systems [181]. Cascading gates—necessary for constructing complex circuits—requires amplitude normalisation to preserve signal integrity. However, amplitude normalisation itself is an energy-intensive, non-linear process [182]. While spin-wave devices are amenable to miniaturisation and scaling, a major challenge at smaller dimensions is crosstalk noise [182]. Efforts are underway to overcome these barriers by integrating spin-wave computing with complementary platforms, including optics, superconducting qubits, and CMOS circuits [148, 160, 183]. Having

[1] Both Boolean [163, 166, 167] and non-Boolean [168–171] computing have been explored using spin-wave systems. In this chapter, we focus on the Boolean case.

acknowledged these experimental challenges, we now restrict our discussion to the modelling of spin-wave circuits for the remainder of this chapter.

We use string diagrams [12–14] to develop a formalism for wave-based Boolean logic circuits, based on the architecture of the spin-wave majority gate. However, the formalism can be applied to other physical platforms and technologies that are based on wave-based computation or majority gate networks [184–186], including quantum-dot cellular automata [150, 187], single-electron circuits [151], nanomagnet logic [154], graphene spin circuits [155], spin torque majority gates [152], molecular scale electronics [156], and DNA logic circuits [153, 188].

In wave-based computation, information is encoded in any measurable property of the wave. The encoding one chooses has a bearing on the physical structures one can use to perform computation, and on the robustness of signals [47]. In our work, we choose phase encoding as it allows composition of logic gates to form more complex circuits without requiring any components that are not wave-based. That is, phase-encoded wave-based circuits are compositional. Moreover, phase also tends to be favoured over the other common candidate, amplitude, because it is often more robust to noise [47]. The phase information of the waves is extracted either from time-domain measurements obtained using oscilloscopes with high sampling rates [178, 189], or from frequency-domain measurements performed with vector network analysers [47].

To illustrate phase encoding, consider a sinusoidal signal of the form $A\sin(\omega t+\phi)$, where A denotes the amplitude, ω the angular frequency, t the time, and ϕ the phase. Phase $\phi=0$ can be defined to represent the logical bit 0 and, likewise, $\phi=\pi$ to represent the logical bit 1. As an example, if a computation is performed in which two waves corresponding to the bit 1 and one wave corresponding to the bit 0 are superposed, the resultant $A\sin(\omega t+\pi)+A\sin(\omega t+\pi)+A\sin(\omega t)=A\sin(\omega t+\pi)$ corresponds to a logical 1. Note that each wave has the same amplitude A. If the output had a non-zero amplitude which was not A, it would be normalised to A before using it as an input to another gate. Amplitude normalisation is performed using devices like directional couplers [173] that exploit non-linear wave interactions [47].

In order to ensure the compositionality of circuits, situations involving total destructive interference must be prevented. For example, one configuration to avoid is a circuit that superposes the waves $A\sin(\omega t)$ and $A\sin(\omega t+\pi)$, as it produces an output with zero amplitude

and an undefined phase, making it unsuitable as an input to another circuit. Essentially, since inputs are normalised and only two phases are allowed in Boolean computation, total destructive interference can arise only if an even number of waves interfere. This is why, in majority gates, it is generally preferable to use an odd number of inputs, ensuring that interference of an even number of waves never occurs [47, 190].

In this work, we restrict ourselves to using only three-input majority gates for interfering waves. AND and OR gates are derivatives of the majority gate and thus also involve interference of three waves—more on this in Section 4.3.1. Because input amplitudes are always normalised, three-input majority gates preclude total destructive interference.

4.3 Wave-based Computation with String Diagrams

4.3.1 Logic Gates and Circuits

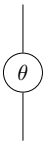
The basic structures through which waves or signals propagate in wave-based circuits are called *waveguides*. We denote a waveguide string-diagrammatically by a wire:

(4.1)

The wave is transmitted through the wire from bottom to top. A wire is essentially an *identity gate*.

Convention 4.3.1. In the remainder of the chapter, a wave $\sin(\omega t)$ (with amplitude $A = 1$ and phase $\phi = 0$) is assumed to enter each circuit from the bottom and exit at the top. If the exiting wave has the phase π, this corresponds to the logical output 1; if it is 0, to the logical output 0.

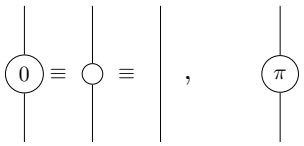
A waveguide may impose a *phase shift*, say θ, on the wave that is transmitted through it. This is denoted by a wire with the phase shift θ:



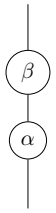
(4.2)

The phase shift has a single parameter $\theta \in \{0, \pi\}$ where $\theta = p\pi$ ($\theta = \pi$ corresponds to $p = 1$ and $\theta = 0$ corresponds to $p = 0$). A single phase shift along a wire initialises a Boolean logic

variable. Note that, in accordance with Convention 4.3.1, this initialisation of the variable is with respect to the reference wave $\sin(\omega t)$, which enters the circuit with phase 0. If $\theta = \pi$, the wave becomes $\sin(\omega t + \pi)$, and therefore bit 1 is initialised; conversely, if $\theta = 0$, bit 0 is initialised. Logical bits 0 and 1 are respectively represented as



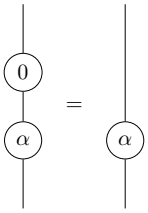
More complex circuits can be created by composing copies of circuits (4.1) and (4.2). For example, series or sequential composition of two copies of circuit (4.2) with phase shifts α $(= a\pi)$ and β $(= b\pi)$ respectively gives the diagram



(4.3)

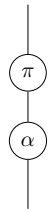
This circuit modifies the phase of the input wave by the shift α followed by the shift β.

Setting the second phase shift to 0, the circuit becomes



which is the series composition of the initialised variable α and an identity gate. This means that the circuit outputs α.

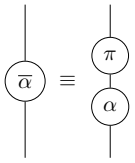
Conversely, setting the second phase shift of circuit (4.3) to π yields



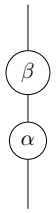
which is the series composition of the variable α and phase shift π. It is instructive to determine the input-output relations of this logic gate. Input $\alpha = 0$ gives the output π while input $\alpha = \pi$ yields the output 0. Therefore, the truth table for this gate is

a	out
0	1
1	0

which means that it is a NOT gate [166]. Symbolically, it can be denoted by a'. In other words, a' is the Boolean algebraic interpretation of this diagram. The NOT gate gives the complement of the input variable. An abbreviated version of the diagram for the complement (Comp) of the variable a is



Let's consider circuit (4.3) again

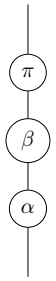


and find its truth table. It has two input variables α and β. Setting both inputs to 0 or π yields the output 0. Complementary inputs give the output π. Therefore, this circuit has the truth table

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

which is that of an exclusive OR (XOR) gate [167].

Composing a NOT gate (which is a π phase shift) in series with an XOR gate gives the diagram

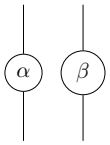


which has the truth table

a	b	out
0	0	1
0	1	0
1	0	0
1	1	1

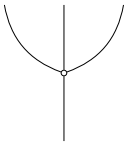
and therefore is an XNOR gate.

So far, we have discussed only the series composition of circuits. One can compose circuits in parallel too. For instance, two wires having phase shifts α and β can be composed in parallel as



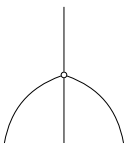
This circuit initialises two Boolean variables.

In phase-encoded wave-based computation, a wave signal may be required to be split or copied into multiple waveguides. For a wave copied into three waveguides, the *copy* operation is represented as



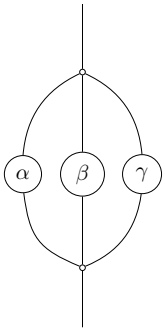
For an input wave entering the bottom wire, three copies are produced at the output.

Interference of waves is performed by combining signals from multiple waveguides into a single waveguide. Diagrammatically, it is given by the *merge* operation:



The output of a merge is obtained by adding the waves entering the three input wires and then normalising the resultant amplitude to 1.

We use only one-to-three copy and three-to-one merge operations here because, in addition to phase shifts, they are all we need to create all other standard logic gates. Three phase shifts (α, β, and γ) taken in parallel, along with a copy and a merge, give the circuit



(4.4)

In keeping with Convention 4.3.1, $\sin(\omega t)$ enters the input wire at the bottom, and is copied into three wires. The three waves are phase shifted by α, β, γ (corresponding to the logical

inputs a, b, c) and are then merged into the single output wire. The output is obtained by normalising the amplitude of the sum $\sin(\omega t + \alpha) + \sin(\omega t + \beta) + \sin(\omega t + \gamma)$. An output wave $\sin(\omega t)$ corresponds to logical 0 whereas $\sin(\omega t + \pi)$ corresponds to logical 1. The truth table for this gate is as follows:

a	b	c	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

i.e., it is a majority (MAJ) gate [163, 178, 179]. In Boolean algebra, it is interpreted as $(a \wedge b) \vee (b \wedge c) \vee (c \wedge a)$.

If one of the inputs, say c, is set to 0, the following truth table is obtained:

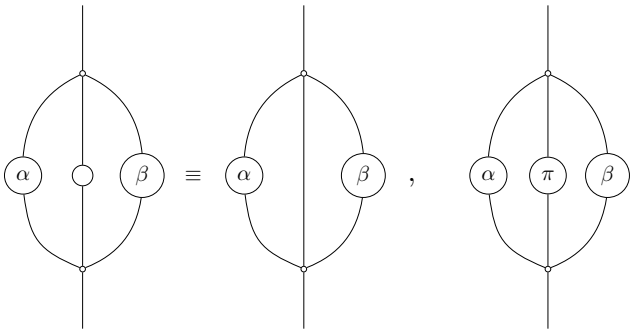
a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

That is, the resultant is effectively an AND gate [185].

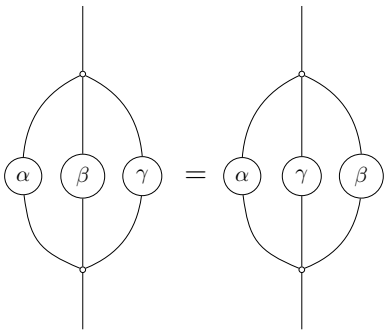
Setting the input c to 1 yields the truth table of an OR gate [185]:

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

In string diagrams, these gates are given by

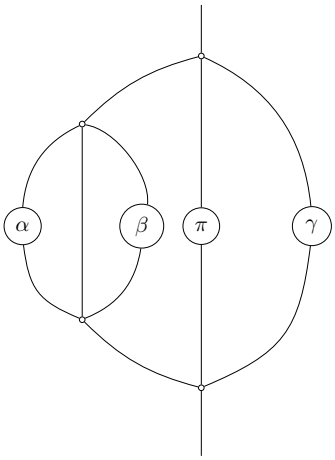


which are interpreted in Boolean algebra as $a \wedge b$ and $a \vee b$, respectively. Note that we have reordered the phases using the commutativity rule (CM):



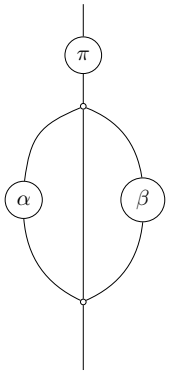
Convention 4.3.2. We follow the convention of drawing the input phase shifts of AND and OR gates on the sides. This helps to visually keep track of the input variables, especially in more complex diagrams.

More complex circuits can be obtained by composing logic gates. For instance, an AND gate and an OR gate can be composed such that the output of the AND gate is used as an input of the OR gate. Consider an AND gate $a \wedge b$ the output of which is used an input of an OR gate to implement the circuit $(a \wedge b) \vee c$. Diagrammatically, this circuit is obtained by plugging the $a \wedge b$ gate in as an input inside an $\vee$ gate with c as the other input:



Another example of composing logic gates is as follows. A NOT gate can be composed with an AND gate $a \wedge b$ to yield a circuit that inverts the output of the AND gate: $(a \wedge b)'$. This is actually a NAND gate. In diagrams, it is generated by composing a NOT gate (*i.e.*

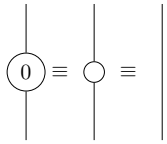
a π phase shift) in series with the AND gate:



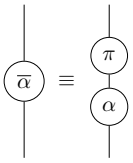
4.3.2 Diagrammatic Reasoning

For diagrammatic reasoning, we need to define a notion of equality and then a set of diagram substitution or rewrite rules based on this notion of equality. In this chapter, two diagrams are considered to be (operationally) equal if they represent the same logical operation, *i.e.* produce the same truth table. Using this notion of equality, we define rewrite rules as follows.

Two definitional rules were already discussed in the previous section: the identity (ID) rule and the complement (Comp) rule:

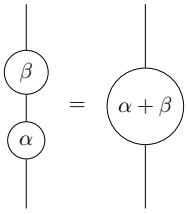


(ID)

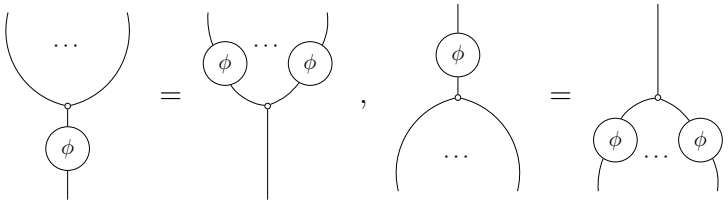


(Comp)

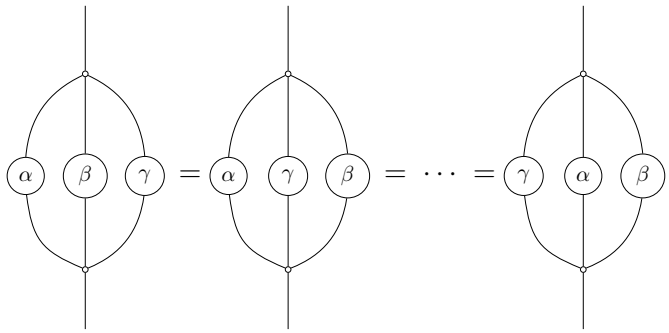
The remaining rules are the fusion (F) rule, the copy (C) rule, the commutativity (CM) rule, the distributivity (D) rule, the majority (M) rule, the associativity (A) rule, and the chopping (CH) rule:



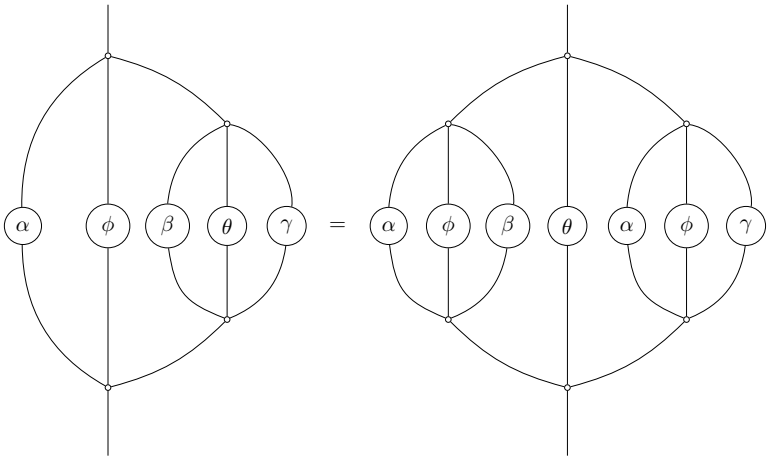
(F)



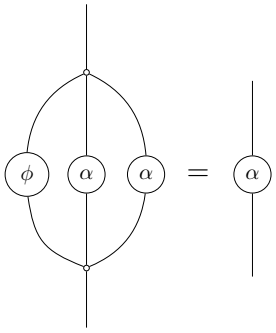
(C)



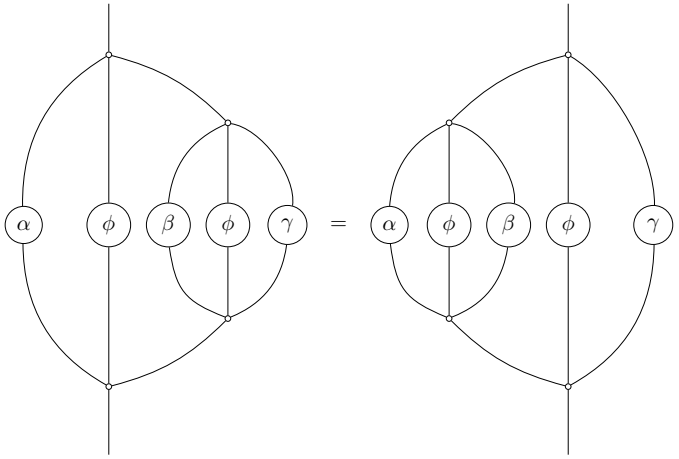
(CM)



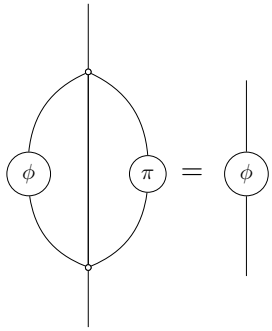
(D)



(M)



(A)

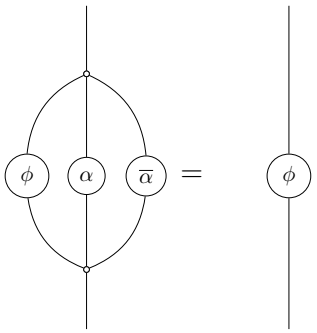


(CH)

where $\alpha, \beta, \gamma, \phi, \theta \in \{0, \pi\}$, $\phi \neq \theta$, and $+$ is the sum modulo 2π.

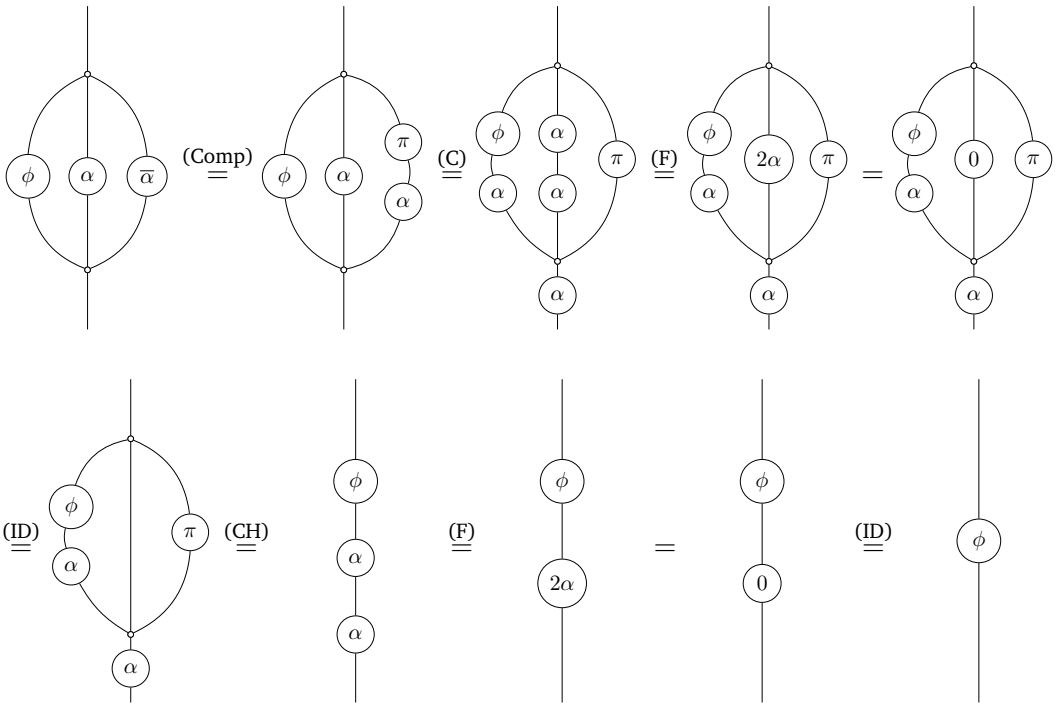
As an example of diagrammatic reasoning, we shall now use the rules defined above to derive another chopping rule (CH2) that will be of use later.

Proposition 4.3.3. For all $\phi, \alpha \in \{0, \pi\}$,



(CH2)

Proof.



□

An important question to ask is whether this string-diagrammatic formalism is universal, sound and complete for Boolean algebra. Here, universality means that the string diagrams can represent every valid Boolean algebraic expression; soundness means that any derivation in string diagrams is correct when interpreted as Boolean algebra; and completeness means that any valid Boolean algebraic equation can be derived using string diagrams.

Any Boolean algebraic expression comprises Boolean variables connected by $\wedge$, $\vee$ and $'$ operations. String diagrammatic formulae for these variables (phase shifts) and operations (logic gates) have already been demonstrated. Taking the $\wedge$ or $\vee$ of a Boolean expression with another one diagrammatically means nesting the corresponding diagrams in an AND or OR circuit. Taking the $'$ (negation) of an expression corresponds to composing the circuit with a NOT gate (*i.e.* a π phase shift). Our string-diagrammatic formalism is therefore universal by construction.

Proving soundness and completeness of this formalism for Boolean algebra will be the subject of future work. However, we offer some remarks below.

The axioms of Boolean algebra are

(a) $\forall x, y \in B,\ x \wedge y = y \wedge x$

(b) $\forall x, y \in B,\ x \vee y = y \vee x$

(c) $\forall x \in B,\ x \vee 0 = x$

(d) $\forall x \in B,\ x \wedge 1 = x$

(e) $\forall x \in B,\ x \vee x' = 1$

(f) $\forall x \in B,\ x \wedge x' = 0$

(g) $\forall x, y, z \in B,\ x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

(h) $\forall x, y, z \in B,\ x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

where $B = \{0, 1\}$.

Axioms (a) and (b) correspond diagrammatically to the (CM) rule, (c) and (d) to the (CH) rule, (e) and (f) to the (CH2) rule, and (g) and (h) to the (D) rule. The (CH2) rule is derived string-diagrammatically using the (CH) rule. Notice that associativity is not one of

the algebraic axioms since it can be derived from the eight above. In fact, its corresponding rule, (A), is derivable string-diagrammatically as well—see Appendix A for details. The rest of the rewrite rules are purely string-diagrammatic, *i.e.* they come for free with the formalism.

4.4 Application to Wave Logic Circuits

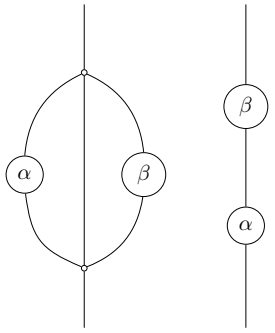
One can use the diagrammatic formalism for design, analysis and optimisation of wave-logic circuits. Some examples are presented below.

4.4.1 Design

Example 4.4.1. A half adder is characterised by the truth table:

a	b	$carry$	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The implementation of an AND gate has already been discussed. Hence the design of the carry circuit is straightforward. The algebraic expression of the sum circuit is that of an XOR gate, which (as discussed before) is implemented by composing phase shifts (corresponding to the input variables) in series. This means the complete half adder circuit is given by

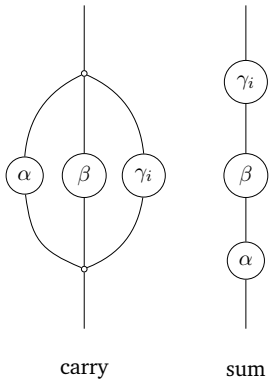


carry sum

Example 4.4.2. For a slightly more complex example, consider a full adder. Its truth table is as follows:

c_in	a	b	c_out	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The truth table shows that the c_out output is equal to that of the MAJ gate, whose implementation has already been discussed. On the other hand, the first half of sum (when $c\_in = 0$) is the output of an XOR gate for inputs a and b whereas the second half (when $c\_in = 1$) is its complement. This means that the sum circuit can be obtained by composing the XOR of a and b with the phase shift c_in in series. Hence, the full adder is given by the diagram



where we represent the phase shift corresponding to c_in by γ_i.

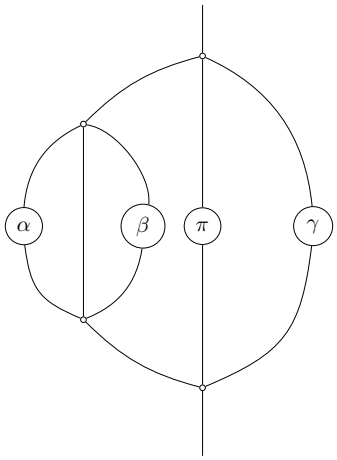
Remark 4.4.3. In diagrams such as the half adder and full adder, a Boolean parameter may appear in more than one location. This repetition is represented at the level of the diagram's interface: several phase shifts may carry the same label, indicating multiple uses of the same external Boolean input.

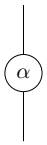
We do not realise these repetitions via the internal copy operation, because they do not correspond to any physical splitting of a wave inside the device; only to multiple references to the same external parameter, exactly as in ordinary Boolean algebraic syntax.

This light form of external bookkeeping does not affect the universality of the diagrammatic language. Universality here concerns the ability to represent any Boolean algebraic expression diagrammatically, and repeated variables in such expressions are realised by repeated labels.

Whenever internal duplication of a wave is required, the calculus already provides a copy operation; its absence in the half adder and full adder examples is simply a modelling choice that keeps the diagrams faithful to the physical architecture of those specific circuits.

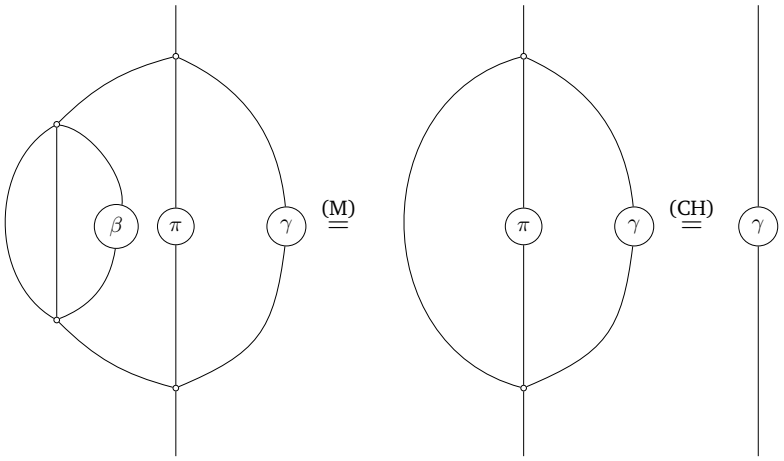
4.4.2 Analysis

Example 4.4.4. Considering the circuit 

, perhaps we are interested in its behaviour if the input a is 0. This corresponds to replacing 

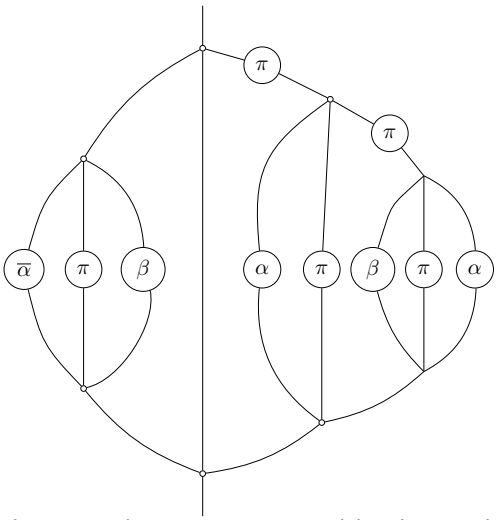
by in the circuit.

The resulting circuit can then be simplified using the rewrite rules, as follows:

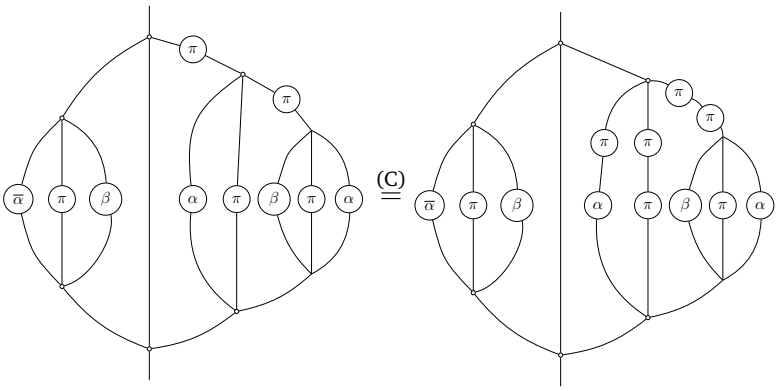


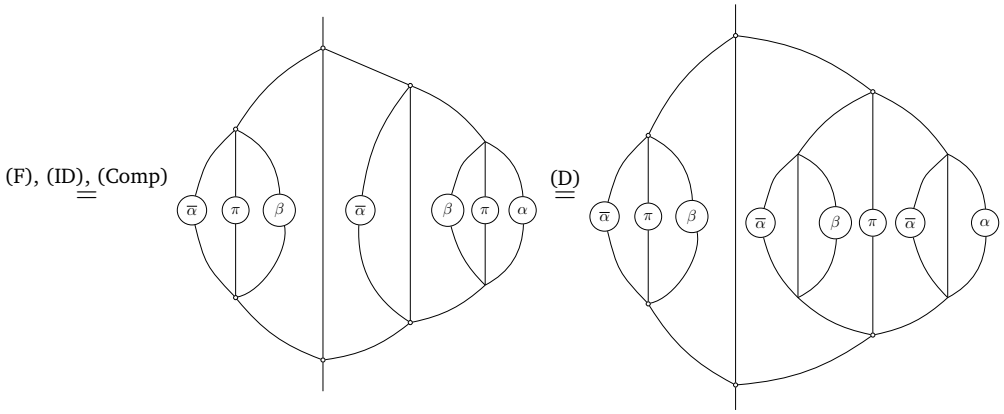
This shows that if input a is set to 0, the output of the circuit is given by the variable c, corresponding to the phase γ.

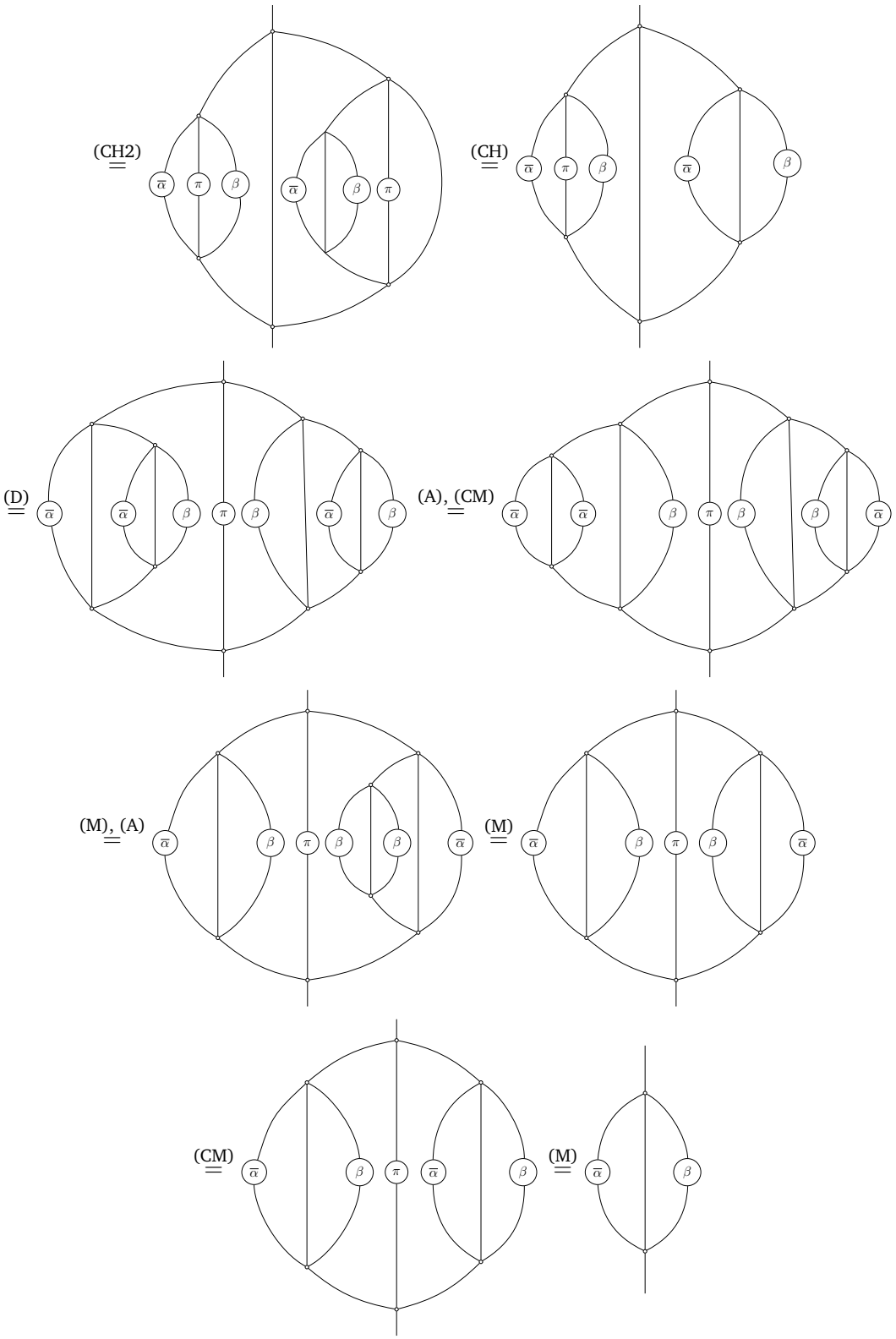
4.4.3 Optimisation

Example 4.4.5. Consider the circuit  for which an optimal implementation is desired. Such an implementation would achieve the same logical operation with the simplest possible circuit and the minimum possible number of logic gates.

The above circuit can be simplified using the diagram rewrite rules as follows.







Remark 4.4.6. When a logic circuit is required to be simplified, the conventional approach is to translate it into its corresponding symbolic algebraic expression, simplify the expression, and then translate the resultant expression back into the circuit.

$$\text{Circ}_1 \longrightarrow \text{Alg}_1 \longrightarrow \text{Alg}_2 \longrightarrow \text{Circ}_2$$

The string-diagrammatic formalism introduced in this chapter can be exploited to simplify circuits without translation into algebraic expressions. This means that the diagrams offer an alternative method for simplifying Boolean circuits.

$$\text{Circ}_1 \longrightarrow \text{Circ}_2$$

Remark 4.4.7. In fact, it is, in principle, possible to use the diagrams as an alternative to Boolean algebra: by mapping the expressions to diagrams, performing graphical simplification, and then converting the simplified diagrams into algebraic expressions again.

$$\text{Alg}_1 \longrightarrow \text{Circ}_1 \longrightarrow \text{Circ}_2 \longrightarrow \text{Alg}_2$$

Although as a syntax, Boolean algebra is more concise as compared with string diagrams, the latter are more amenable to visual intuition which is useful for a circuit designer or analyst.

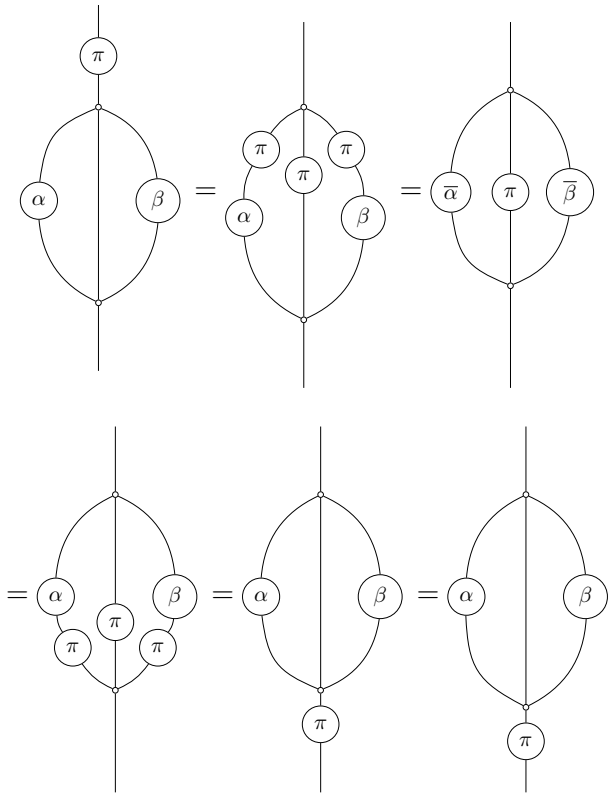
4.5 Summary and Outlook

In this chapter, we introduced a string-diagrammatic formalism for wave-based computation with phase encoding. Through the example of spin-wave circuits, we demonstrated the usage of the formalism in designing, analysing and simplifying Boolean logic circuits. The approach is device-independent and hence useful as a visual, intuitive aid to design, optimise and analyse circuits in different physical platforms that are based on wave-based or majority-logic-based computation [47, 135–138].

A key advantage of the approach lies in its immediate applicability to spin-wave logic circuits [47, 163], in the sense that the components of the circuit diagrams map directly to their physical counterparts. In other words, 'what you see is what you get'. The formalism, moreover provides a theoretical framework to represent and characterise spin-wave logic implementations such as those in Refs. [163, 166, 167, 178, 179].

In addition, diagrammatic rewrite rules allow graphical simplification of circuits which potentially lend themselves to automation and/or software-based optimisation. Indeed, broadly, the approach is very expressive in the flexibility it lends to circuit design and implementation. For instance, one may tweak a particular circuit to obtain the desired design (possibly subject to hardware constraints) by moving the phase shifts along the wires and across other phase shifts, and using copy and merge operations (applying the corresponding

rules). More concretely, if one is to compose a NOT gate with a circuit, it may be cascaded at the top or the bottom of the circuit (or the circuit may be modified internally in a more elaborate manner), giving the same logical outcome in either case.



An obvious avenue for future work is providing formal semantics for the string diagrams presented here, and proving that the formalism is sound and complete for Boolean algebra. Another line of research is to conceive of the formalism as a diagrammatic alternative to the traditional symbolic approach, wherein lies the possibility of a new axiomatisation of Boolean algebra.

This study focuses on the theoretical modelling of spin-wave circuits and introduces a formalism that abstracts away many hardware-level challenges. However, since spin-wave computing inherits key limitations of analogue systems—such as noise sensitivity, signal degradation, error accumulation, and energy-intensive error-correction mechanisms—these factors are not fully captured in the current framework. In particular, future extensions of the formalism may need to incorporate models for error correction, which often relies on non-linear processes that can compromise energy efficiency [181]. Similarly, cascading gates requires amplitude normalisation to maintain signal integrity, yet this too is a non-linear and energy-intensive operation [182]. Modelling these aspects—along with crosstalk

noise at small scales and potential integration with complementary platforms such as optics, superconducting qubits, or CMOS circuits [148, 160, 183]—will be crucial for improving the practical relevance of the formalism. Doing so would enhance its applicability to real-world implementations and support the development of more robust, scalable spin-wave computing systems.

5

Distributional Compositional Circuits for English and Urdu Text

> 'A sentence is not a state, but a process.... Text is a process that alters meanings of words.'
>
> Bob Coecke,
> The Mathematics of Text Structure [31, pp. 198–199]

This chapter is adapted from the publication [51], based on work carried out in collaboration with Vincent Wang-Maścianica, Jonathon Liu, and Bob Coecke. The author of this thesis is the first author of the publication [51], and played a leading role in all the work described in this chapter. The novel contributions are presented in Sections 5.5 and 5.6.

5.1 Introduction

Distributional Compositional Circuits (DisCoCirc) is a framework that incorporates both distributional (*i.e.* vectorial) and compositional aspects to model the meaning of natural languages [31]. DisCoCirc gives a process-theoretic account of language and builds further on the earlier work on the Categorical Distributional Compositional (DisCoCat) model for combining grammar and meaning [191, 192]. An important distinction between the two is that DisCoCirc is able to model not only the meaning of individual sentences, but also the interaction of sentences giving rise to the meaning of texts generally. The central idea is that the information associated with noun entities appearing in the text (encoded in circuit wires) is updated by sentences (modelled as gates) as the text progresses.

DisCoCirc admits a two-dimensional string-diagrammatic formalism, inspired by quantum circuits/networks, and therefore offers prospects for quantum-computational natural language

processing for texts comprising multiple sentences or paragraphs [193, 194]. It also has been applied to a number of problems including spatio-temporal models of language meaning [195], logical and conversational negation in natural language [196, 197], and solving logical puzzles [198, 199]. The relationship between DisCoCirc and discourse-representation theory [200] has also been briefly explored [32].

Recently, Wang-Maścianica, Liu and Coecke proposed a method for generating DisCoCirc diagrams for a significant fragment of English [32]. They started by creating a *hybrid grammar* for English text, incorporating *phrase structure*, *pronominal links*, and *phrase regions*.[1] It is called a hybrid grammar because it integrates key ideas from different grammatical formalisms. The main structure is based on transformational phrase structure grammars [201], with modifications inspired by pregroup grammars [202]. Pronominal links are derived from discourse representation theory [200], whereas phrase regions are based on dependency grammars [203].

These hybrid grammar representations of text were translated into DisCoCirc *text circuits*, via an intermediate structure called *text diagrams* that involve string diagrams. The entire translation process preserves the compositionality and connectedness of text meaning.

The same work describes how in the reverse direction, text circuits can be used as a generative grammar. For each freely generated text circuit, we can write some corresponding text. In this chapter, we use 'text' to refer not only to a string of words forming sentences, but to this string of words further endowed with a hybrid grammar structure. The text corresponding to a given text circuit is not unique owing to the loss of grammatical 'bureaucracy' [32] in the passage from one-dimensional syntax to two-dimensional text circuits.[2] Here, grammatical bureaucracy is used in a broad sense to refer to all of the stylistic choices one must make when communicating some desired meaning in the form of a text. That is, whenever two different texts communicate what is essentially the same meaning, we attribute the differences in

[1] To provide some intuition for these linguistic concepts, here are a few examples.
Consider the sentence "Alice likes Bob". The *phrase structure* provides a breakdown of the sentence into its basic constituents: "Alice" and "Bob" are noun phrases, while "likes" is a verb phrase.
If we have two sentences, "Alice likes Bob" and "Bob hates Charlie", with "Bob" referring to the same person, a *pronominal link* keeps track of this information. When the two sentences are combined into "Alice likes Bob who hates Charlie", the pronominal link resolves the reference of the relative pronoun "who", establishing a clear link back to its antecedent "Bob" and ensuring semantic coherence.
Finally, *phrase regions* help show how groups of words belong together. For instance, in the sentence "Dan sees Eve laugh", they make it clear that "Eve laugh" falls within the scope of "sees". In other words, the region headed by "sees" includes both "Eve" and "laugh", showing that Dan is observing the entire action of Eve laughing.

[2] Some of this was already observed in Ref. [204].

the structure of these texts to grammatical bureaucracy. Thus, grammatical bureaucracy includes syntactic rules like those governing word order, but also choices like the use of pronouns, whether to use a single long sentence with multiple clausal constructions or multiple short sentences, etc.

It was, in particular, suggested that because of eliminating grammatical bureaucracy and stylistic choices embedded in a particular natural language, text circuits are to some degree language-independent [32]. We take this suggestion seriously in this chapter, and show how DisCoCirc does indeed undo the grammatical bureaucracy related to word and phrase order for restricted fragments of English and Urdu. English and Urdu are both Indo-European languages and share similarities in grammatical rules and idioms. While they both follow the noun-verb distinction, they differ in sentence structure due to differences in word order. For instance, English uses a subject–verb–object (SVO) order, whereas Urdu follows a subject–object–verb (SOV) order.

Our main argument is structured as follows.

- In Ref. [32], a hybrid grammar was developed for English. A surjection from the set of all English text generated with the English hybrid grammar to the set of all English text circuits was demonstrated:

$$\text{English text} \twoheadrightarrow \text{English text circuits}$$

- In a similar vein, in this chapter, we describe how the hybrid grammar can be adapted for Urdu. We then provide rules for its translation into text diagrams and text circuits, which is essentially the same as in Ref. [32].[3] We show that this gives a surjective map from the set of all Urdu text generated with Urdu hybrid grammar to the set of all Urdu text circuits:

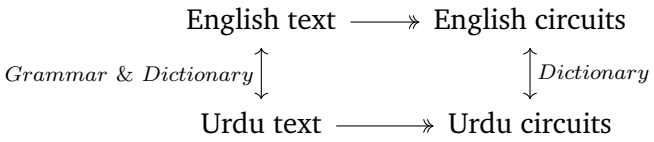
$$\text{Urdu text} \twoheadrightarrow \text{Urdu text circuits}$$

- For these restricted fragments, there is a clear isomorphism between the hybrid grammars for English and Urdu:

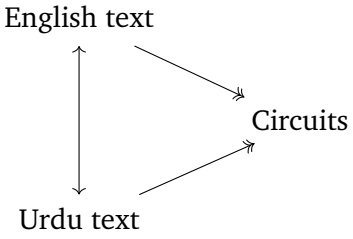
$$\text{English hybrid grammar} \simeq \text{Urdu hybrid grammar}$$

[3] We expect texts in other Indo-European languages to be amenable to translation into text diagrams and circuits. However, our approach may not be directly applicable to languages like Arabic, which lie outside this language family. We will return to this point later.

- Given the above correspondence, it turns out that text circuits for English and Urdu become the same, up to translating the labels on the gates. In other words, the following diagram commutes:



  Now, formally speaking, our texts consist of a hybrid grammar structure with labels (words) provided by a dictionary. If we restrict to just the grammatical structures and forget the language-specific word labels, the circuits for English and Urdu become literally the same. In this case, the above diagram reduces to:



The chapter is organised as follows. The hybrid grammar, text diagrams and text circuits for English are introduced in Sections 5.2, 5.3 and 5.4 respectively. A hybrid grammar for Urdu is introduced in Section 5.5, followed by the main results related to Urdu text diagrams and circuits in Section 5.6. Section 5.7 concludes the chapter.

5.2 A Hybrid Grammar for English Text

In this section, we introduce the hybrid (generative) grammar for English developed in Ref. [32]. We focus on modelling the language fragment that includes nouns, adjectives, verbs (intransitive and transitive), adverbs (modifying verbs), adpositions (for intransitive verbs), verbs with sentential complements, conjunctions, and pronominal links (subject relative pronouns and object relative pronouns). The technical terms will be explained later.

Disclaimer 5.2.1. For ease of analysis, the modelled language fragment is limited to propositional statements in a single tense. It is also assumed that adverbs and adjectives can be stacked indefinitely and appear to the left of their respective nouns and verbs. Additionally, determiners are not explicitly modelled.

In Ref. [32], first, *simple* sentences were modelled, containing verbs, adverbs, adjectives and adpositions. Then, *pronominal links* were introduced to account for recurring nouns and pronoun-referent pairs. Rewrite rules were introduced that allowed for the fusion of simple sentences into more complex ones, and the introduction of relative pronouns. Rules for modelling verbs with sentential complement and *phrase scope boundary* were introduced to accommodate compound sentences formed of components that are themselves sentences. We shall provide detailed descriptions and examples later.

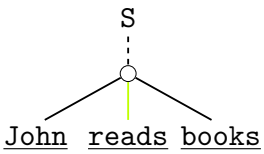
The hybrid grammar begins with a standard *phrase structure* grammar that generates the simple sentences, based on a *string rewrite system* with finitely many *production rules* of the form $\alpha \mapsto \beta$ [205]. These are valid transformations of strings of symbols represented by α and β. Individual symbols may be phrase components or entire words of the language we are modelling. In the latter case, the symbol is *terminal*, meaning no more rewrite rules can be further applied. In a grammar such as ours, a particular language comprises all strings of terminal symbols—forming grammatically correct sentences—that can be generated by applying finitely many production rules (associated with that language) to a start symbol, S. Only those productions that yield grammatical sentences are considered valid. For example, using the rules:

$$\begin{aligned} \texttt{S} &\mapsto \texttt{NP}_1 \cdot \texttt{TVP} \cdot \texttt{NP}_2 \\ \texttt{NP}_1 &\mapsto \underline{\texttt{John}} \\ \texttt{TVP} &\mapsto \underline{\texttt{reads}} \\ \texttt{NP}_2 &\mapsto \underline{\texttt{books}} \end{aligned}$$

where NP and TVP represent noun phrase and transitive verb phrase respectively, underlining denotes terminals, and $\cdot$ denotes string concatenation; we can generate the sentence <u>John reads books.</u>:

$$\begin{aligned} \texttt{S} &\mapsto \texttt{NP}_1 \cdot \texttt{TVP} \cdot \texttt{NP}_2 \\ &\mapsto \underline{\texttt{John}} \cdot \texttt{TVP} \cdot \texttt{NP}_2 \\ &\mapsto \underline{\texttt{John}} \cdot \underline{\texttt{reads}} \cdot \texttt{NP}_2 \\ &\mapsto \underline{\texttt{John}} \cdot \underline{\texttt{reads}} \cdot \underline{\texttt{books}} \end{aligned}$$

The rewriting of strings can be represented as two-dimensional tree diagrams, read from top to bottom. Our example sentence can be represented as



Convention 5.2.2. In this chapter, generative production rules $\alpha \mapsto \beta$ are depicted from top to bottom.

Disclaimer 5.2.3. We restrict to a version of hybrid grammar [32] generated by the explicitly context-free rules. A context-free rule is one that applies to an individual non-terminal symbol without depending on the surrounding symbols. We also exclude reflexive pronouns.

Symbolic labels can be dropped for intermediate edges and, instead, a colour coding can be used. See Table 5.1 for the colour coding employed in this chapter.

Grammatical type	Symbol/word	Colour code
Noun phrase	NP	
Intransitive verb phrase	IVP	
Transitive verb phrase	TVP	
Sentential complement verb	SCV	
Adjective	ADJ	
Adverb	ADV	
Conjunction	CNJ	
Adposition	ADP	
Copula	is (English), hai (Urdu)	
Phrase scope boundary	(,)	

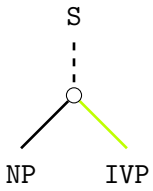
Table 5.1: Colour coding for grammatical types

In this section, a *phrase structure grammar* is introduced by providing the tree fragments for various grammatical types. We start with *simple* sentences, which include single verbs that do not take sentential complements.

5.2.1 Simple Sentences

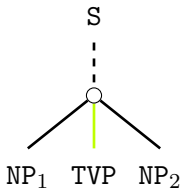
The *verb* in a simple sentence may be *intransitive* or *transitive*. If the verb is intransitive, the sentence comprises a noun phrase followed by an intransitive verb phrase. For

example, Bob laughs. The production rule for generating this sentence is $\mathtt{S} \mapsto \mathtt{NP} \cdot \mathtt{IVP}$ and the tree fragment is



(5.1)

On the other hand, if the verb is transitive, the sentence comprises a noun phrase, followed by a transitive verb phrase and another noun phrase. For instance, Alice loves Bob. The production rule for this case is $\mathtt{S} \mapsto \mathtt{NP}_1 \cdot \mathtt{TVP} \cdot \mathtt{NP}_2$ and the tree fragment is



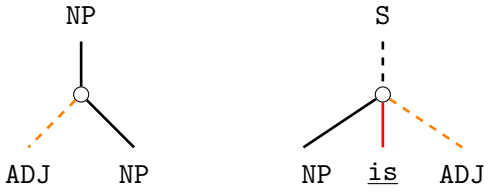
(5.2)

Remark 5.2.4. In the language fragment considered here, we require the subject and object of a transitive verb to be distinct.

Apart from the aforementioned production rules, there are terminal rules for each verb type: $\mathtt{IVP} \mapsto \underline{\mathtt{IV}}$ and $\mathtt{TVP} \mapsto \underline{\mathtt{TV}}$, respectively, where terminals are denoted by underlining.

Convention 5.2.5. From here onward, we do not explicitly show the terminal rules in the diagrams. The terminal symbols can be directly inferred from the finished derivations.

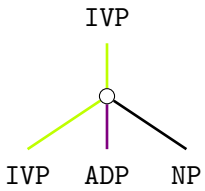
A simple sentence may consist of *adjectives* in two ways: adjectives can appear to the left of the noun phrase that they are modifying, for example, tall Bob; alternatively, a single adjective can appear to the right of a noun phrase after the copular verb is, for example, Bob is tall. The corresponding production rules are $\mathtt{NP} \mapsto \mathtt{ADJ} \cdot \mathtt{NP}$ and $\mathtt{NP} \mapsto \mathtt{NP} \cdot \underline{\mathtt{is}} \cdot \mathtt{ADJ}$, respectively. The tree fragments are as follows:



(5.3)

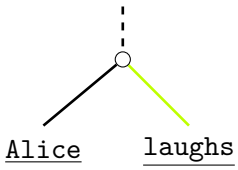
In simple sentences, *adverbs* can appear to the left of verbs. An example sentence is: `Alice happily talks.` For intransitive and transitive verb phrases, the adverb production rules are $\texttt{IVP} \mapsto \texttt{ADV} \cdot \texttt{IVP}$ and $\texttt{TVP} \mapsto \texttt{ADV} \cdot \texttt{TVP}$ respectively. The tree fragments are:

A simple sentence may contain an *adposition* that appears to the right of an intransitive verb phrase, followed by a noun phrase. For example, adding an adposition to the sentence `Alice smiles.`, we can get `Alice smiles at Bob.` The production rule for an adposition is $\texttt{IVP} \mapsto \texttt{IVP} \cdot \texttt{ADP} \cdot \texttt{NP}$, with the tree fragment given by

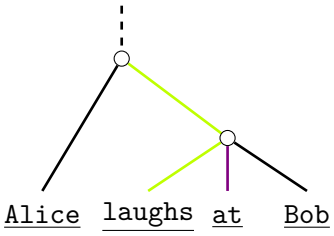


(5.4)

Example 5.2.6. Using the IVP-production rule yields $\texttt{NP} \cdot \texttt{IVP}$, corresponding to the grammatical structure of `Alice laughs.`



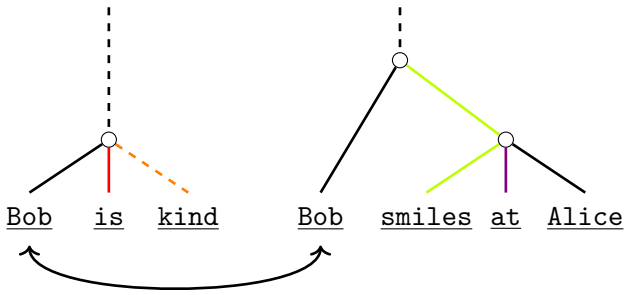
Applying the ADP-production rule to IVP—before the application of the terminal rule—gives the grammatical structure of `Alice laughs at Bob.`



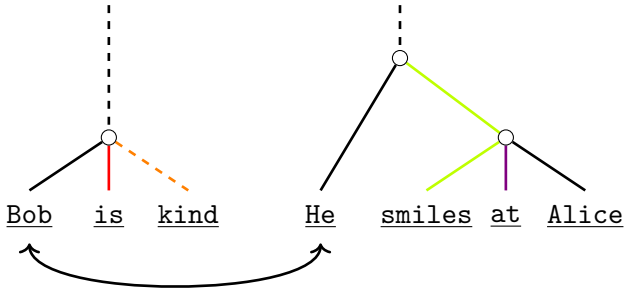
5.2.2 Compound Sentences

Compound sentences contain more than one verb. Compound sentences can be obtained by joining two simple sentences using *relative pronouns*. Pronoun information is modelled

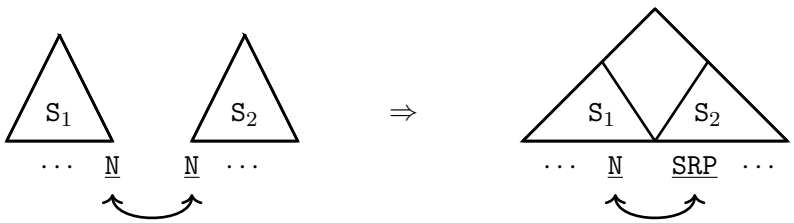
using *pronominal links* identifying nouns in the same or different sentences. These are represented by arrows pointing at the identified nouns drawn below the trees. For instance, the two instances of Bob in the sentences Bob is kind. and Bob smiles at Alice. can be identified as follows:



If there is a pronominal link, the later occurrences of a noun can be replaced by pronouns referring to the earlier occurrence.



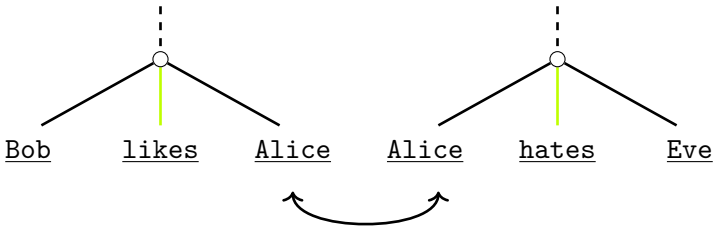
In certain settings of pronominal links, two consecutive sentences (*i.e.,* their parse trees) can be joined together using relative pronouns. Consider the configuration in which the subject noun of the second tree S_2 points to the object noun of the first tree S_1. Then, a *subject relative pronoun* can be used to replace the subject noun in the second tree S_2, and the two sentences can be fused. Diagrammatically, it can be represented by the rewrite rule:



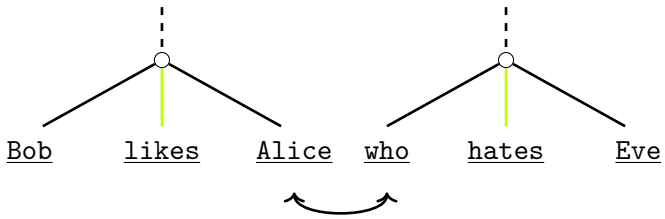
(5.5)

On the right hand side, the big triangle denotes the single sentence formed by the fusion of the two sentences. The above rewrite rule modifies phrase structure trees and hence is an example of a *transformational grammar rule*.

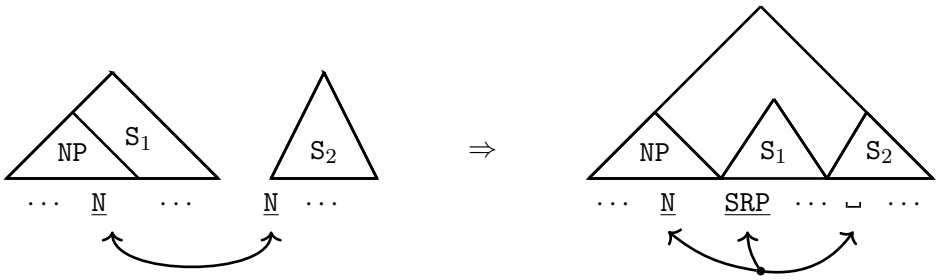
Example 5.2.7. In the sentences `Bob likes Alice.` and `Alice hates Eve.`, the two occurrences of `Alice` can be identified using a pronominal link.



The two trees can be joined by replacing the second occurrence of `Alice` with `who`. This results in the following diagram:



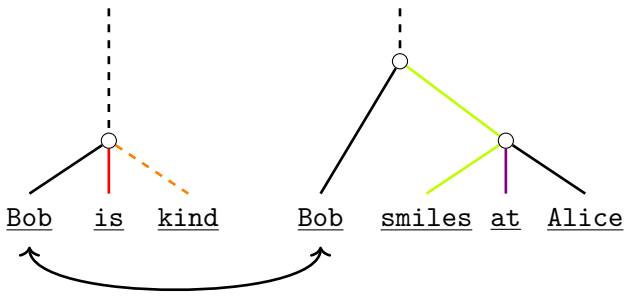
A subject relative pronoun can also be used to relate subject nouns of two sentences in the following way. If the subject noun of the second parse tree $\mathtt{S}_2$ refers to the subject noun of the first parse tree $\mathtt{S}_1$, the transformation rule yields: the noun phrase of $\mathtt{S}_1$, followed by $\mathtt{S}_1$ with its noun phrase replaced by the relative pronoun, followed by $\mathtt{S}_2$ without its subject noun. Diagrammatically, it can be denoted as



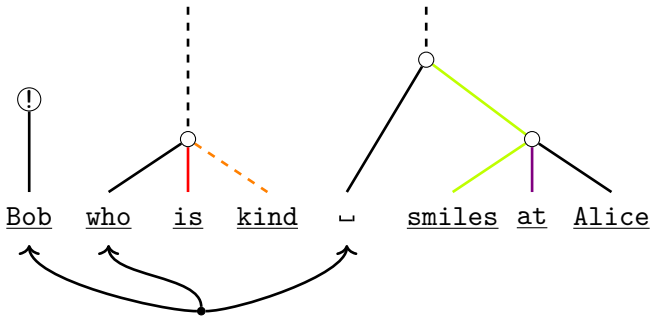
(5.6)

where the multiarrow identifies the pronominally linked nouns.

Example 5.2.8. Consider the following example again.

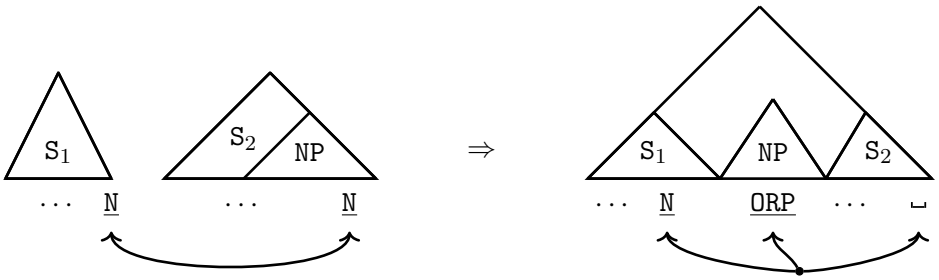


Applying the transformational rule, we get the sentence



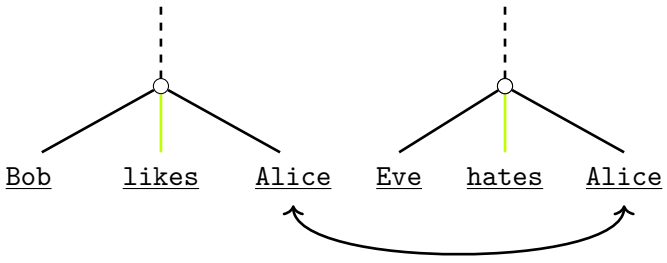
The artefacts ! (denoting isolated nouns) and ␣ (denoting blank labels) will be eliminated in the setting of text diagrams, introduced in Section 5.3.

If the object noun of the second sentence s_2 refers to the object noun of the first sentence s_1, the two sentences can be fused using an *object relative pronoun*. The transformational rule gives s_1, followed by the object relative pronoun, followed by s_2 with its object noun removed. Diagrammatically, it can be represented by

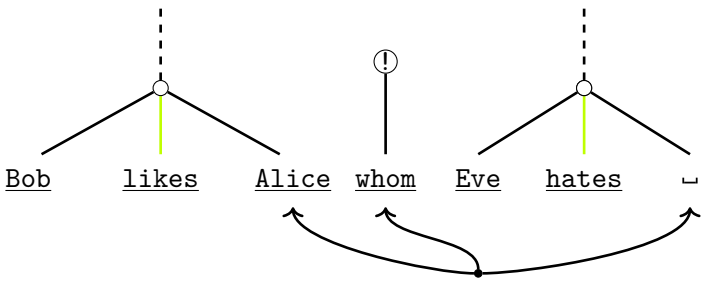


(5.7)

Example 5.2.9. Consider the sentences: Bob likes Alice. and Eve hates Alice. We identify the two occurrences of the object noun Alice using a pronominal link.



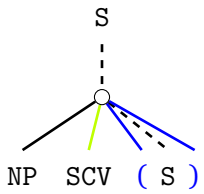
Applying the transformational rule, we have



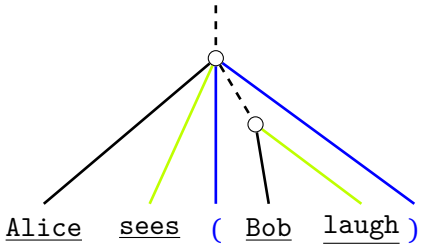
Remark 5.2.10. The language fragment considered in this chapter does not include reflexive pronouns. Therefore, we do not discuss intra-sentence pronominal links, introduced in Ref. [32].

Compound sentences can also be obtained by including in sentences phrases that are themselves sentences. Two such kinds of compound sentences are introduced here.

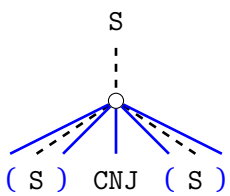
A compound sentence can be formed using a *verb with a sentential complement*. Examples of such verbs are `sees` and `thinks`. The compound sentence includes a noun phrase, followed by a verb, followed by a sentence that is the sentential complement of the verb. For example, in `Alice sees Bob laugh.`, the sentential complement is `Bob laughs.` The production rule is given by $\texttt{S} \mapsto \texttt{NP} \cdot \texttt{SCV} \cdot \texttt{(} \cdot \texttt{S} \cdot \texttt{)}$ where the types `(` and `)` represent the boundaries of the phrase scope. Its tree fragment is as follows.



Example 5.2.11. The tree diagram of the sentence `Alice sees Bob laugh.` is given by

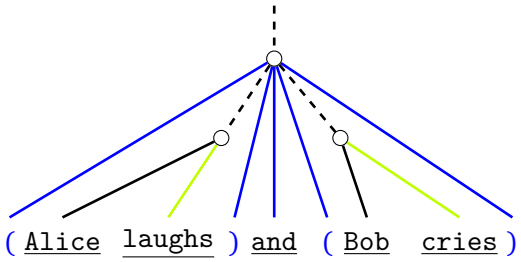


A compound sentence can also be formed by joining two sentences using *conjunctions*. The production rule is $\texttt{S} \mapsto \texttt{(} \cdot \texttt{S} \cdot \texttt{)} \cdot \texttt{CNJ} \cdot \texttt{(} \cdot \texttt{S} \cdot \texttt{)}$ and its tree fragment is given by.



Example 5.2.12. The phrase structure of the sentence `Alice laughs and Bob cries.` is as

follows.



5.3 Text Diagrams

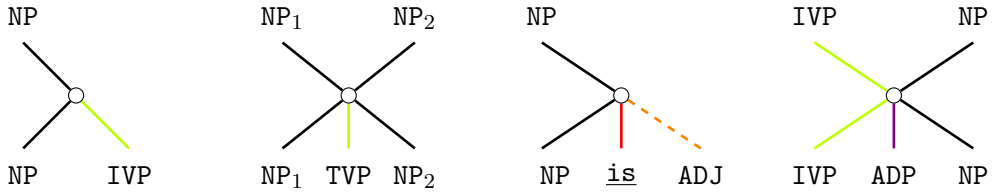
Text diagrams were introduced in [32] as an intermediate representation between hybrid grammar and text circuits. In a single mathematical framework, text diagrams capture the connectedness of meaning or grammatical structure, including not just the grammatical relations of simple sentences but also pronominal links as well as phrase scope.

Text diagrams do away with artefacts such as those handling pronominal links. Based on string diagrams [14, 22, 206], text diagrams offer a structural framework for boxes with inputs and outputs that allow for parallel and sequential composition.

Convention 5.3.1. In this chapter, all string diagrams must be read from top to bottom and left to right.

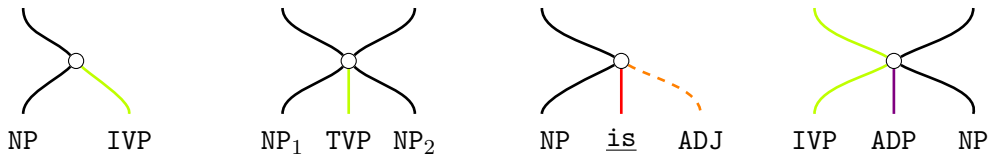
5.3.1 Simple Sentences

When moving from hybrid grammar to text diagrams, the phrase structure rules of the previous section are modified in the following way: each S-type is replaced by NP-types, the number of which depends on the sentence. It is ensured that every rule that includes NP-types has the same number of input and output NP-types. The modified rules, corresponding respectively to rules (5.1), (5.2), (5.3), and (5.4), are given below:

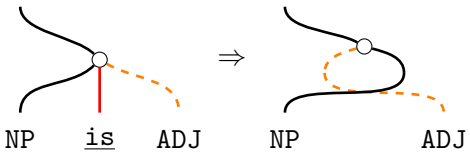


Convention 5.3.2. To guide the eye, we vertically align the ends of input wires with those of the corresponding output wires in text diagrams (and, later, in text circuits as well). Moreover,

we bend the wires to make them point upward or downward, reflecting their roles as inputs or outputs string-diagrammatically. We also drop labels that appear more than once.

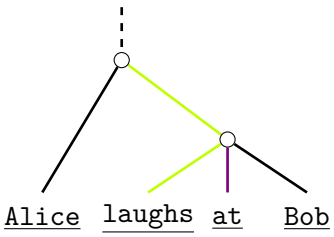


The copular verb is can be eliminated via the rewrite

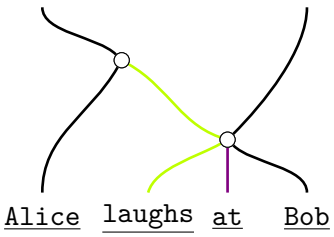


converting the postpositonal adjective into a prepositional one.

Example 5.3.3. The sentence Alice laughs at Bob. has the following grammatical structure



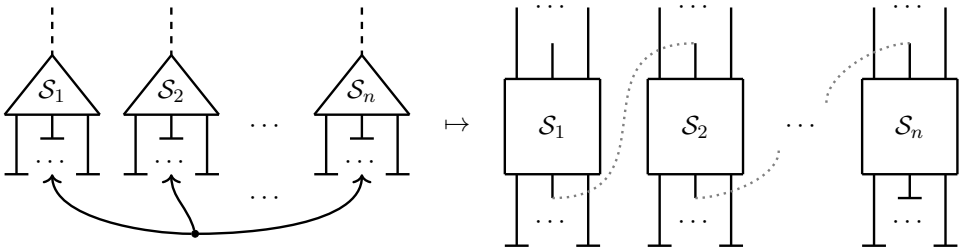
which, in text diagrams, is given by



This modification allows composition of tree fragments (while respecting the grammatical types). In text diagrams, pronominal links and phrase scope constructions are part of the same unified mathematical framework. Moreover, unlike in the hybrid grammar tree diagrams, wires in text diagrams may cross one another.

5.3.2 Compound Sentences

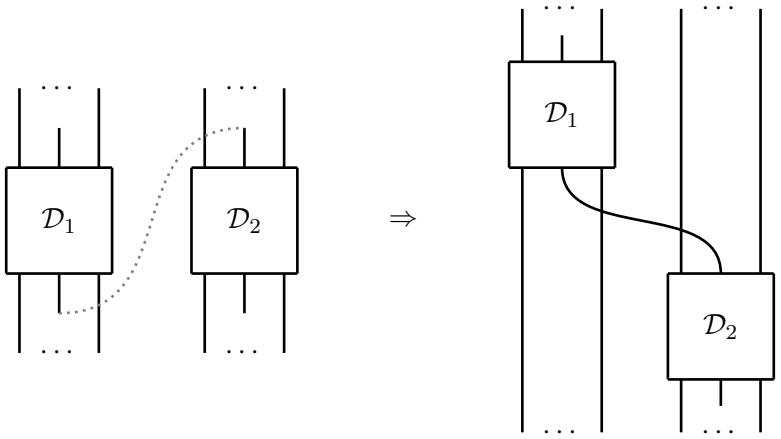
In the passage from hybrid grammar to text diagrams, the (multi)arrows become wires linking the pronominally-linked noun wires.



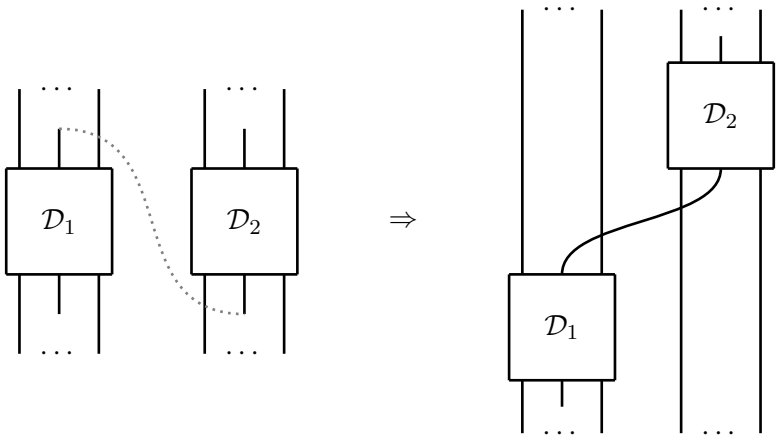
(5.8)

The pronominal link wires can be eliminated using the diagram rewrite rules:

This implies that for two text diagrams $\mathcal{D}_1$ and $\mathcal{D}_1$, we have the following rewrite rules.

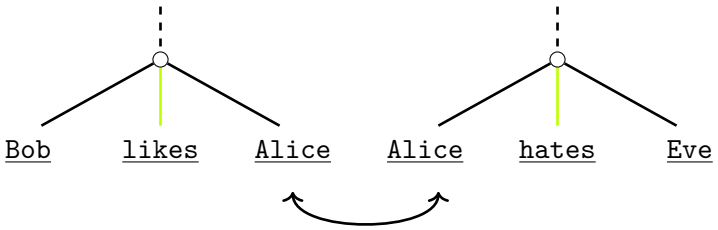


(5.9)

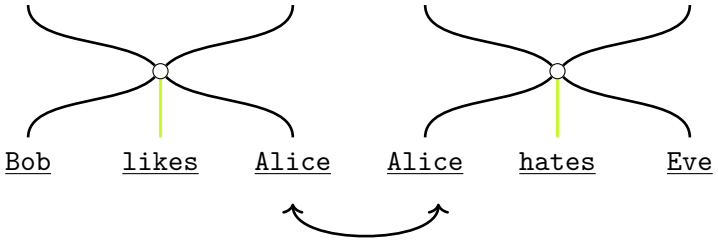


(5.10)

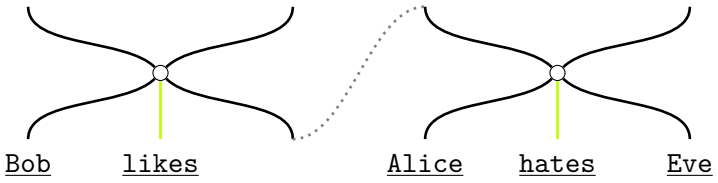
Example 5.3.4. Consider the following hybrid grammar text.



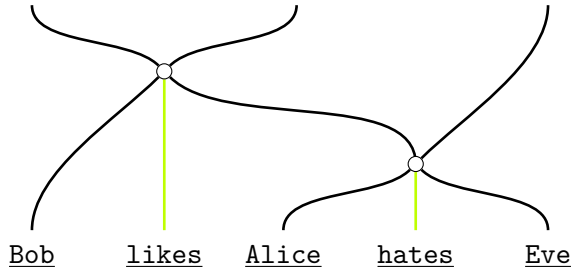
Each tree can be replaced with the corresponding text diagram.



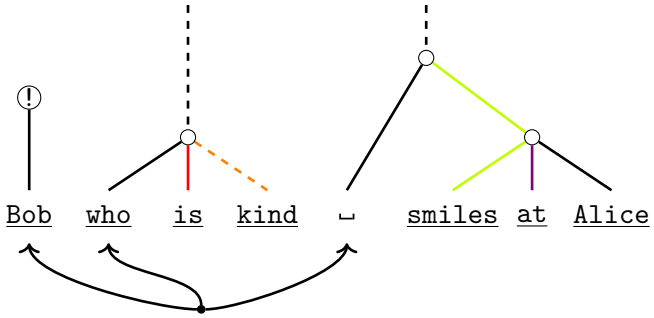
The pronominal link arrow can be replaced by a wire.



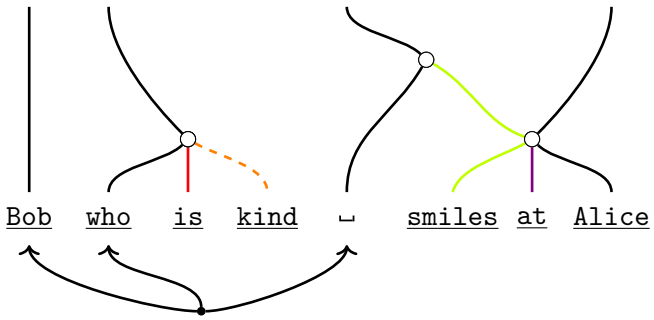
Notice that there is only one available output wire corresponding to Alice, and it is not labelled by the pronoun who. Using the rewrite rule for pronoun wire elimination, we get the following diagram.



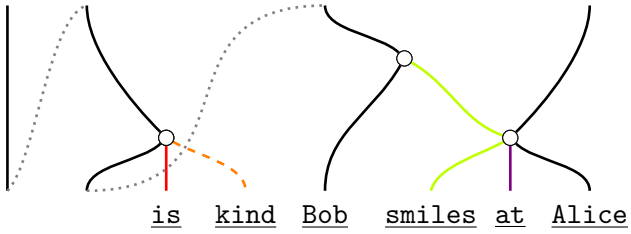
Example 5.3.5. Consider the following example sentence again.



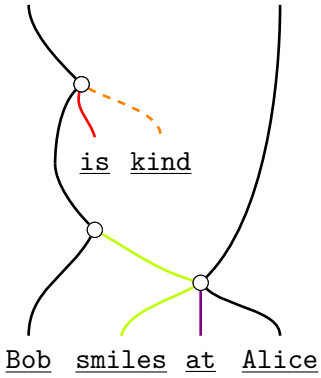
The artefacts ! and ␣ were required in the hybrid grammar to account for relative pronouns. In text diagrams, these artefacts are eliminated in the following way. Replacing the grammar trees with the corresponding text diagrams



and replacing the multiarrows with pronominal link wires, we get



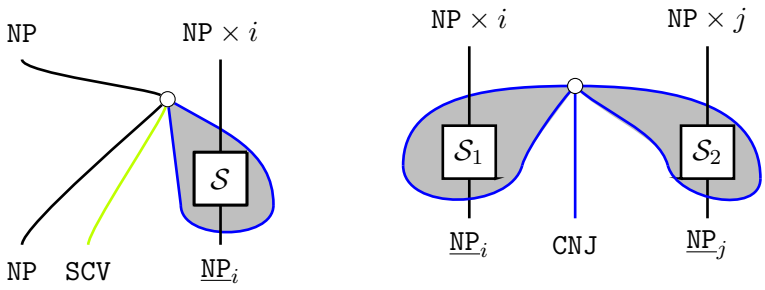
which can be rewritten as follows.



Moving from hybrid grammar to text diagrams, the sentence type S was replaced by a sentence-dependent number of NP wires. Therefore, string diagrams with regions [32] are needed to accommodate phrase scope.

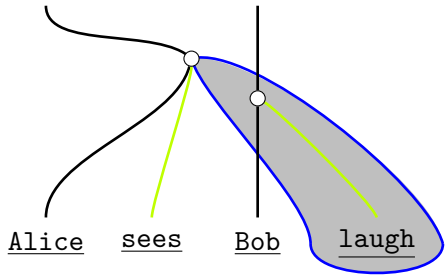
Convention 5.3.6. *Phrase regions* delineate phrase scope in text diagrams. These regions act as planar obstacles to everything except NP wires and pronominal link wires. This means that only NP and pronominal link wires can enter or exit the phrase regions. NP labels must always be placed outside the phrase regions.

The text diagrams corresponding to verbs with sentential complements and conjunctions are respectively given by

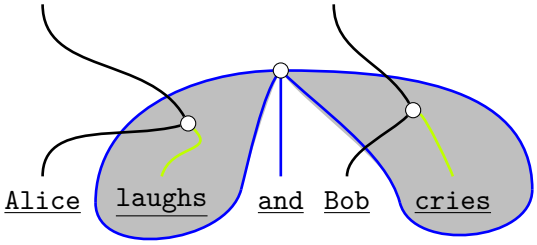


where i and j denote the number of NP wires.

Example 5.3.7. The sentence Alice sees Bob laugh. corresponds to the following text diagram.

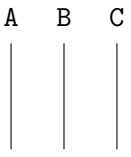


Example 5.3.8. The text diagram of the sentence Alice laughs and Bob cries. is as follows.



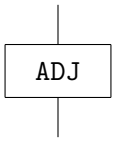
5.4 Text Circuits

Text circuits are composed of wires (or types), boxes (or first-order gates), and boxes with holes (or second-order gates). Nouns are represented by wires with labels:

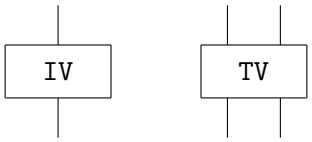


Here, each wire represents a distinct noun.

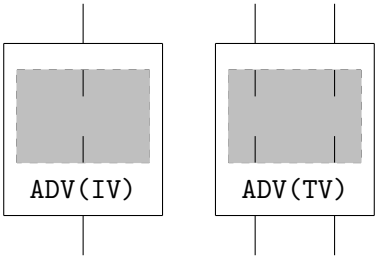
Adjectives are represented by single-input-single-output boxes or gates that update or act upon noun wires:



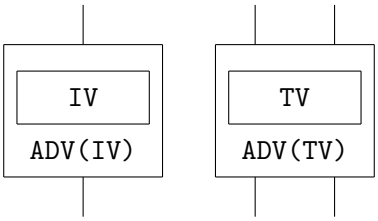
Like adjectives, intransitive verbs are also represented by single-input-single-output gates, whereas transitive verbs are given by double-input-double-output gates as they act on subject and object noun wires:



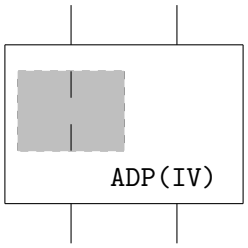
Adverbs modify verbs, which are themselves boxes. Hence, adverbs are represented by second-order gates, which are boxes with holes that are to be filled with verb boxes. The shape of the expected verb box is denoted by the wires inside the holes.



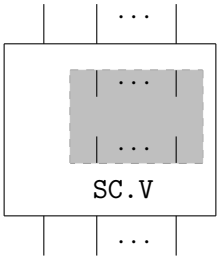
When the holes in the adverb boxes are filled, gates are obtained.



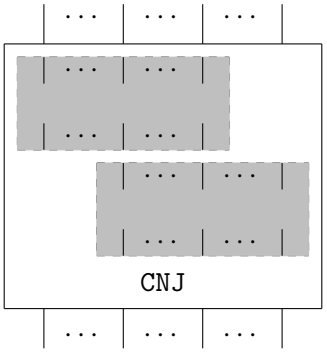
Adpositons also act upon verbs and add another noun wire to the right of the circuit. An adposition acting on an intransitive verb is given by a box with a hole expecting an intransitive verb circuit.



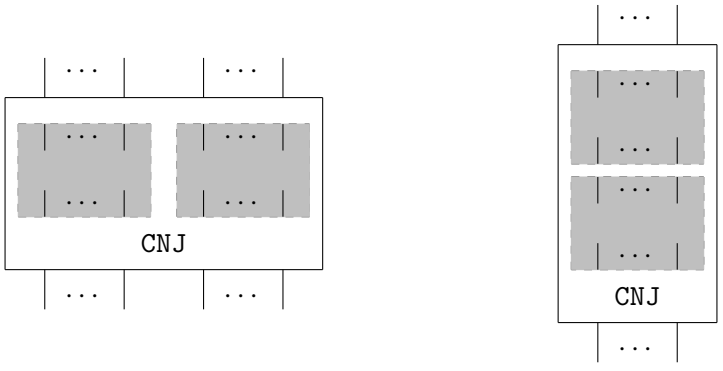
Verbs with sentential complements are represented by second-order gates that add a noun wire to the left of the circuit. Since the sentential complement can have an arbitrary number of noun wires, these gates form a class of boxes to account for the different number of wires.



For conjunctions as well, there is a class of second-order gates. The conjunction boxes have two holes that accept boxes/gates that may have some overlapping noun wires.



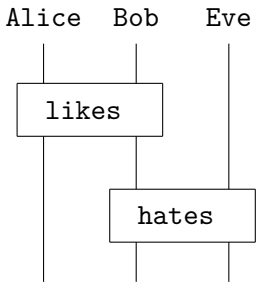
As special cases, the conjunction boxes may have disjoint or completely shared noun wires.



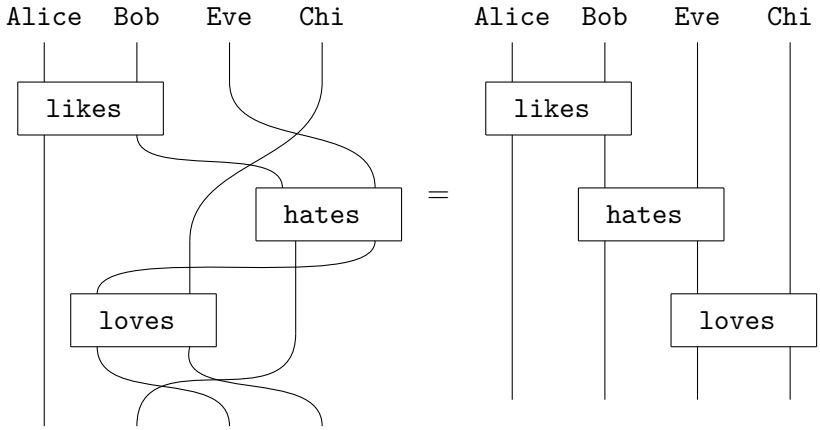
Gates can be composed in parallel or series/sequence to form *text circuits*. If no noun labels are shared between the gates, they are composed in parallel. Conversely, if the noun labels are shared, the gates are composed in sequence with the shared noun wires matched.

Example 5.4.1. The text `Alice likes Bob.  Bob hates Eve.` can be drawn as the follow-

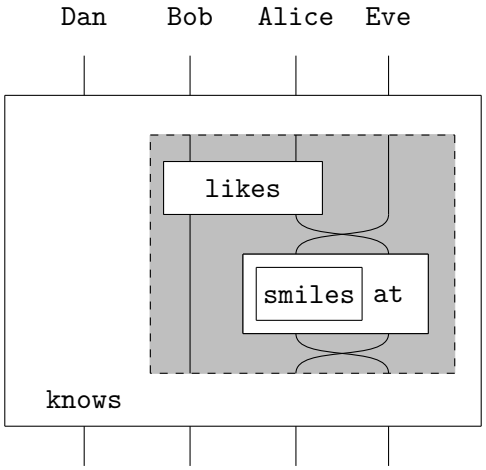
ing circuit.



Convention 5.4.2. In text circuits, nouns are represented by wires, which may cross or twist past one another. Two circuits are considered the same if they have the same connectivity of gates. This allows us to eliminate unnecessary wire twists to work with simpler diagrams. An example is as follows.



Example 5.4.3. The text Dan knows Bob likes Alice whom Eve smiles at. can be drawn as the following circuit.



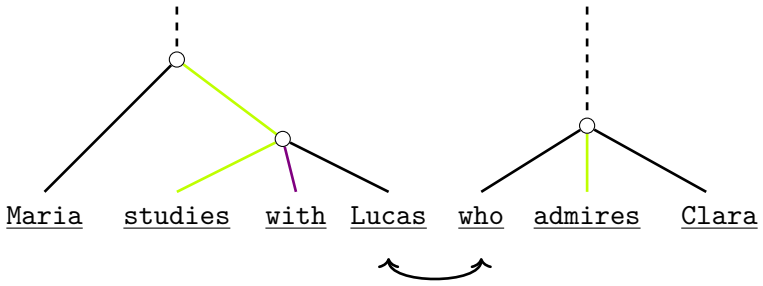
Thesis 5.4.4 (Text Circuit Thesis; Thesis 5.17 in [32])**.** Equal text circuits correspond to equal text meanings.

Theorem 5.4.5 (Text Circuit Theorem; Theorem 5.1 in [32]). *Let E denote the set of generators of the hybrid English grammar.*[4] *Let T_E denote the set of all English text constructed with grammar E, and let C_E denote the set of all text circuits for English. Then, there exists a surjection $T_E \twoheadrightarrow C_E$.*

Remark 5.4.6. The significance of Theorem 5.4.5 stems from Thesis 5.4.4. Theorem 5.4.5 implies that the text corresponding to a given text circuit may not be unique. According to Thesis 5.4.4, all texts corresponding to the same text circuit share the same meaning. In this sense, DisCoCirc text circuits eliminate differences between texts that convey the same meaning.

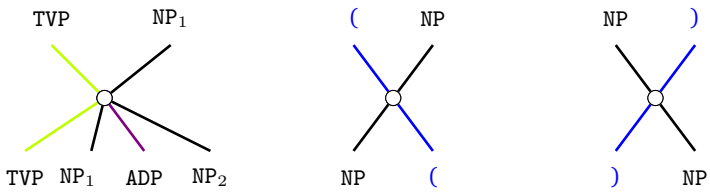
When two different texts express essentially the same meaning, we attribute their structural variation to grammatical bureaucracy. This includes stylistic choices such as the use of pronouns, the preference for a single long sentence with multiple clauses versus several shorter ones, and similar considerations. This grammatical bureaucracy is eliminated in the transition from the linear syntax of text to the two-dimensional structure of text circuits.

Example 5.4.7. Consider the following hybrid grammar text.



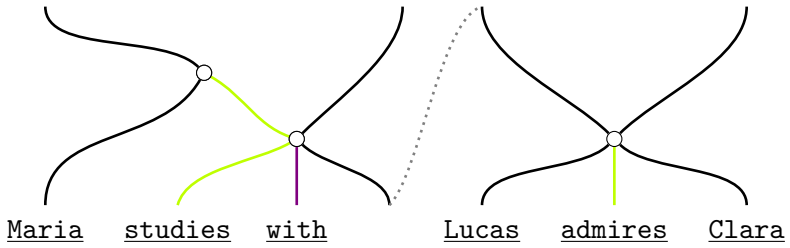
Replacing the grammar trees with the corresponding text diagrams, we have the following

[4] In this chapter, for the sake of simplicity, we focus on a restricted version of the hybrid grammar developed in Ref. [32]. We do not include reflexive pronouns. Moreover, the following three hybrid grammar rules are excluded from our analysis.

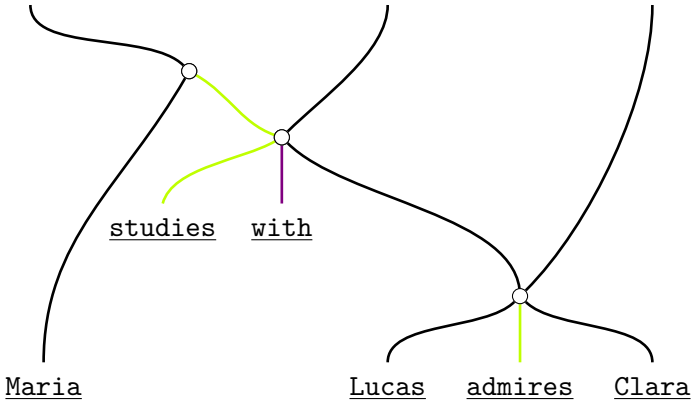


The first rule on the left involves an adposition (ADP) and a transitive verb phrase (TVP), while the remaining rules allow noun wires to exit phrase scope. Note that these rules are context-sensitive and do not apply to individual non-terminal symbols.

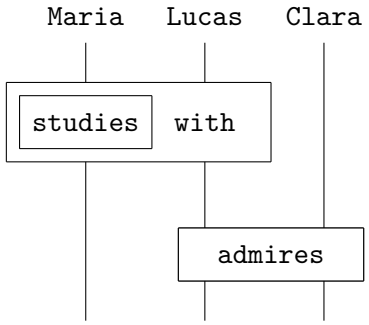
diagram.



Composing the pronominal link wires, we get



which can be expressed as gates and boxes to yield a circuit.

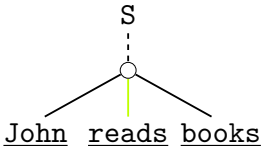


5.5 A Hybrid Grammar for Urdu Text

It was shown in Ref. [32] that English text generated from the English hybrid grammar is surjective to text circuits. In this chapter, we demonstrate a similar result for Urdu.

First, we develop a hybrid grammar for Urdu text. The language fragment we consider for Urdu is the same as that for English discussed in the previous sections.

Since a language is specified by set of production rules, different production rules lead to different languages. In English, the sentence `John reads books.` has the following structure.



Translating to Urdu, `John reads books.` becomes:

جان کتابیں پڑھتا ہے۔

It can be transliterated[5] into English as

John kitabein parhta hai (Urdu)

John books reads

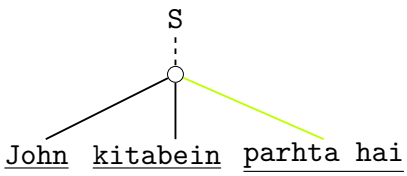
Using the production rules

$$\begin{aligned} \mathtt{S} &\mapsto \mathtt{NP}_1 \cdot \mathtt{NP}_2 \cdot \mathtt{TVP} \\ \mathtt{NP}_1 &\mapsto \underline{\mathtt{John}} \\ \mathtt{NP}_2 &\mapsto \underline{\mathtt{kitabein}} \\ \mathtt{TVP} &\mapsto \underline{\mathtt{parhta\ hai}} \end{aligned}$$

we can generate the sentence under discussion:

$$\begin{aligned} \mathtt{S} &\mapsto \mathtt{NP}_1 \cdot \mathtt{NP}_2 \cdot \mathtt{TVP} \\ &\mapsto \underline{\mathtt{John}} \cdot \mathtt{NP}_2 \cdot \mathtt{TVP} \\ &\mapsto \underline{\mathtt{John}} \cdot \underline{\mathtt{kitabein}} \cdot \mathtt{TVP} \\ &\mapsto \underline{\mathtt{John}} \cdot \underline{\mathtt{kitabein}} \cdot \underline{\mathtt{parhta\ hai}} \end{aligned}$$

Its tree diagram is given by

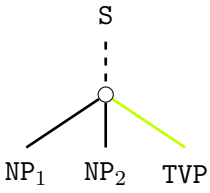


From this example, we can immediately spot an obvious difference between English and Urdu—the order of subject, verb and object: in Urdu, the verb is placed at the end of the sentence. This contrast plays a significant role in differentiating Urdu and English grammars.

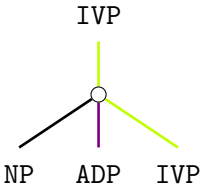
We now develop production rules and tree diagrams for the fragment of Urdu text (including verbs, adjectives, adverbs, adpositions, pronominal links, phrase scope) corresponding to that of English discussed in Section 5.2. In so doing, we realise that many of the rules and tree fragments are in fact the same. The ones that are different differ mainly in the relative placement of the verb. See Table 5.2. The differences are summarised as follows:

[5]Urdu script is written from right to left. Throughout this chapter, for ease of readability and linguistic analysis, we shall use English transliteration of Urdu text. However, to dispel any ambiguity which may arise from transliteration, we shall provide the Urdu script and the respective phrase meanings in English as well.

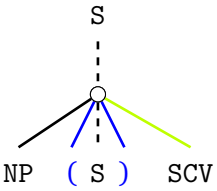
- In contrast to subject-verb-object (SVO) as in English, Urdu has the subject-object-verb (SOV) order.



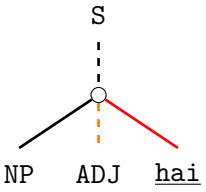
- The placement of adpositions differs from that in English; in Urdu, the verb simply comes last.



- Sentential complements precede verbs in Urdu.

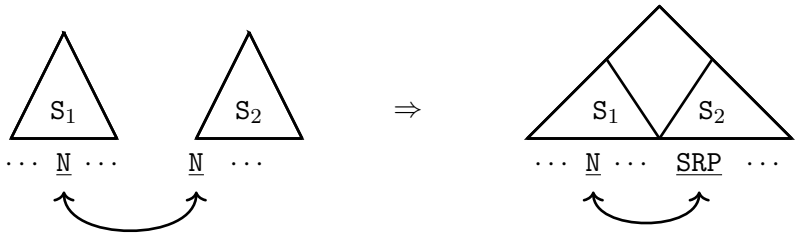


- In Urdu, the copula hai in the postpositional adjectival construction appears to the right of the adjective. On the other hand, in English, the copula is appears to the left of the adjective.



The transformational grammar rules for pronominal links are the same for English and Urdu in terms of the sentence-order, and the connectivity of pronominal links. However, there are differences arising from the different word order within individual simple sentences.

The counterpart of rewrite (5.5) for subject relative pronoun in Urdu is as follows.



(5.11)

Example 5.5.1. Consider the Urdu sentences

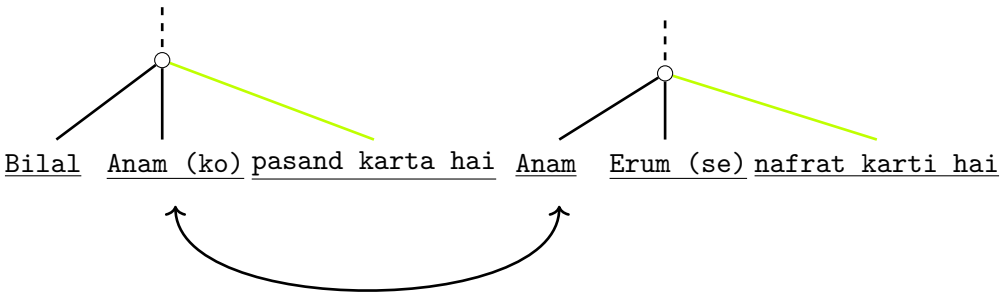
بلال انعم کو پسند کرتا ہے۔ انعم ارم سے نفرت کرتی ہے۔

transliterated as:

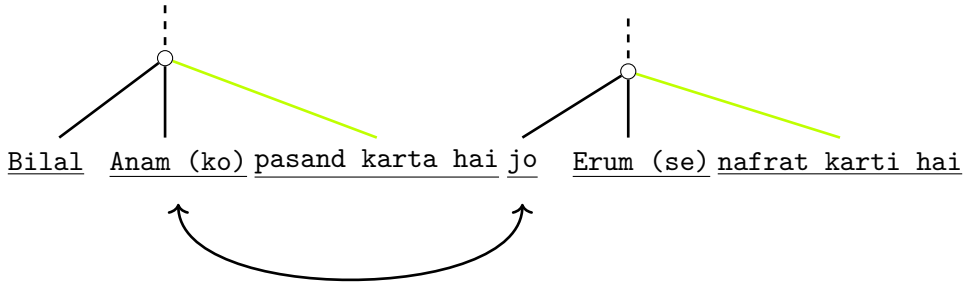
Bilal Anam (ko) pasand karta hai. Anam Erum (se) nafrat karti hai. (Urdu)

Bilal Anam likes. Anam Erum hates.

The two occurrences of Anam can be identified using a pronominal link.



The two trees can be joined by replacing the second occurrence of Anam with jo. This results in the following diagram:



corresponding to the Urdu sentence

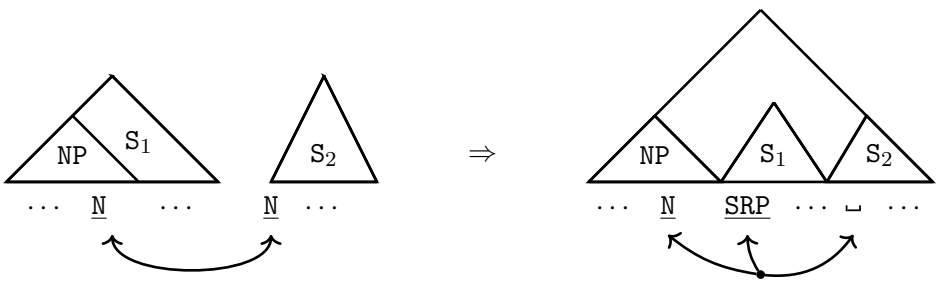
بلال انوم کو پسند کرتا ہے جو ارم سے نفرت کرتی ہے۔

transliterated as:

Bilal Anam (ko) pasand karta hai jo Erum (se) nafrat karti hai. (Urdu)

Bilal Anam likes who Erum hates.

The counterpart of rewrite (5.6) for subject relative pronoun in Urdu is given by



(5.12)

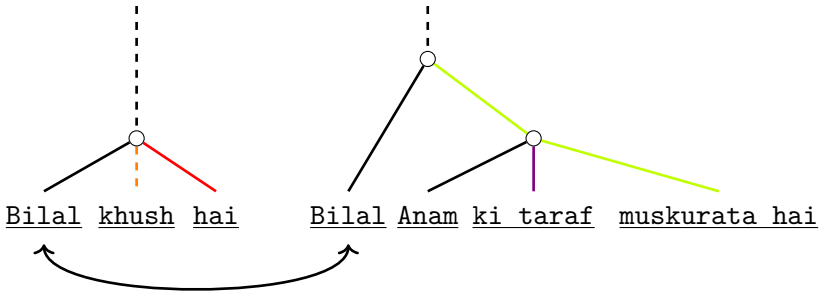
Example 5.5.2. Consider the Urdu sentences

بلال خوش ہے۔ بلال انعم کی طرف مسکراتا ہے۔

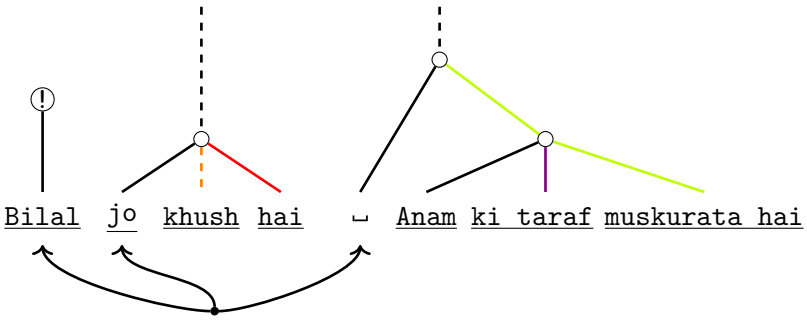
transliterated as:

Bilal	*khush*	*hai.*	*Bilal*	*Anam*	*ki taraf*	*muskurata hai.*	(Urdu)
Bilal	happy	is.	Bilal	Anam	at	smiles.	

The two occurrences of `Bilal` can be identified using a pronominal link.



Applying the transformational rule, we get the sentence



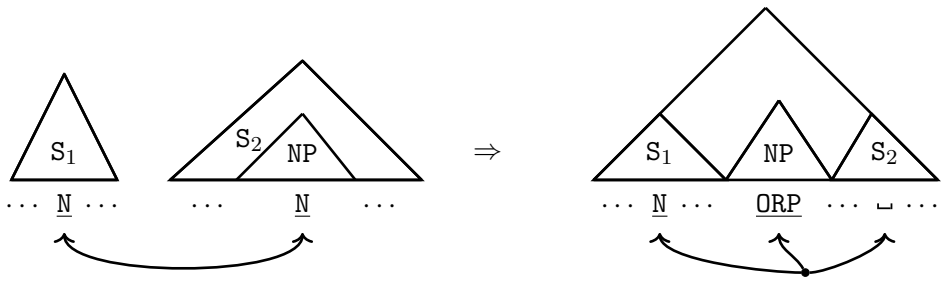
corresponding to the Urdu sentence

بلال جو خوش ہے انعم کی طرف مسکراتا ہے۔

transliterated as:

Bilal	*jo*	*khush*	*hai*	*Anam*	*ki taraf*	*muskurata hai.*	(Urdu)
Bilal	who	happy	is	Anam	at	smiles.	

The counterpart of rewrite (5.7) for object relative pronoun in Urdu is as follows.



(5.13)

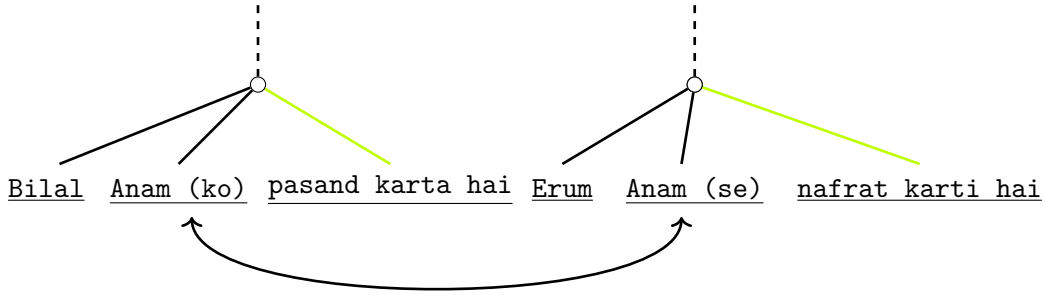
Example 5.5.3. Consider the Urdu sentences

بلال انعم کو پسند کرتا ہے۔ ارم انعم سے نفرت کرتی ہے۔

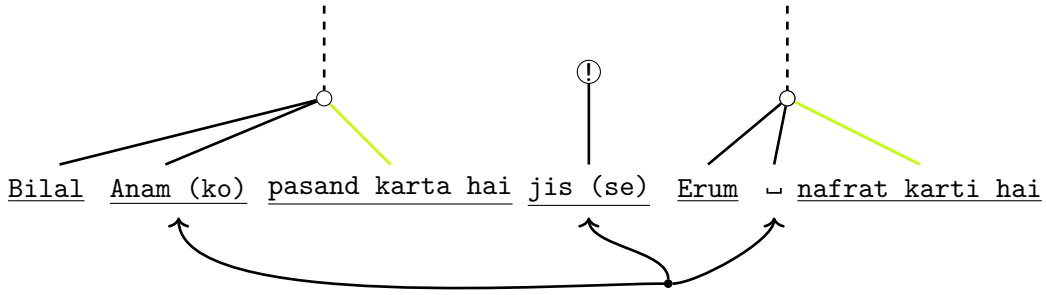
transliterated as:

Bilal	*Anam (ko)*	*pasand karta hai.*	*Erum*	*Anam (se)*	*nafrat karti hai.*	(Urdu)
Bilal	Anam	likes.	Erum	Anam	hates.	

The two occurrences of Anam can be identified using a pronominal link.



Applying the transformational rule, we get the sentence



corresponding to the Urdu sentence

بلال انعم کو پسند کرتا ہے جس سے ارم نفرت کرتی ہے۔

transliterated as:

Bilal	*Anam (ko)*	*pasand karta hai*	*jis (se)*	*Erum*	*nafrat karti hai.*	(Urdu)
Bilal	Anam	likes	whom	Erum	hates.	

Remark 5.5.4. Moving from the hybrid grammar to text diagrams (using (5.8), (5.9) and (5.10)), the connectivity of pronominal links is retained, and is the same for English and Urdu.

5.6 Text Diagrams and Circuits for Urdu Text

5.6.1 Main result

Let $\mathtt{E}$ denote the set of generators of the hybrid English grammar, summarised in Table 5.2. Let $\mathtt{T_E}$ denote the set of all English text constructed with grammar $\mathtt{E}$, and let $\mathtt{C_E}$ denote the set of all text circuits for English. Then, there exists a surjection $\mathtt{T_E} \twoheadrightarrow \mathtt{C_E}$ [32].

We have created a set of generators for Urdu grammar $\mathtt{U}$ which closely correspond with the English generators $\mathtt{E}$; see Table 5.2. Let $\mathtt{T_U}$ denote the set of all Urdu text generated with $\mathtt{U}$. Let $\mathtt{C_U}$ denote the set of all text circuits for Urdu. Then,

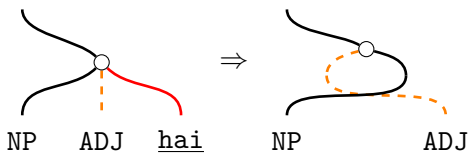
- there exists a surjection $\mathtt{T_U} \twoheadrightarrow \mathtt{C_U}$, and
- $\mathtt{C_U}$ is isomorphic to $\mathtt{C_E}$, i.e., $\mathtt{C_U} \xrightarrow{\simeq} \mathtt{C_E}$ (up to word translations at the gate level).

5.6.2 Urdu Text Surjects onto Circuits

The method of turning hybrid grammar trees into text diagrams is the same in English and Urdu: each component of a hybrid grammar tree is modified so that the number of $\mathtt{NP}$ wires for inputs and outputs is equal, and sentence types $\mathtt{S}$ are eliminated.

Table 5.2 illustrates the similarities and differences between Urdu and English grammar, as reflected in the hybrid grammar trees and text diagrams.

The same text diagram reductions hold in Urdu as those in English. The following is the reduction of a postpositional adjectival construction using a copula $\underline{\mathtt{hai}}(\simeq \underline{\mathtt{is}})$ to a prepositional adjective that does not require a copula:



Just as in the case of English hybrid grammar, in the passage to text diagrams, the pronominal-link (multi)arrows become wires (5.8) that can be eliminated using the diagram rewrite rules (5.9) and (5.10) (see Remark 5.5.4).

Lemma 5.6.1. Let $\mathtt{U}$ denote the set of generators of the hybrid Urdu grammar. Let $\mathtt{T_U}$ denote the set of all Urdu text constructed with grammar $\mathtt{U}$, and let $\mathtt{C_U}$ denote the set of all text circuits for Urdu. Then, there exists a surjection $\mathtt{T_U} \twoheadrightarrow \mathtt{C_U}$.

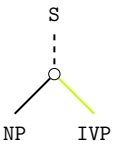
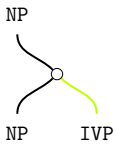
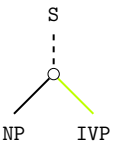
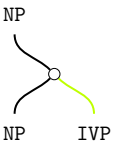
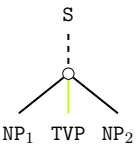
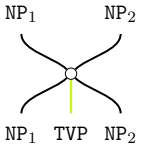
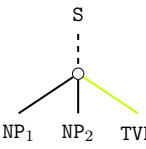
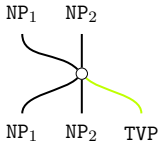
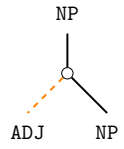
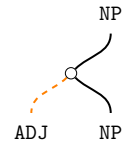
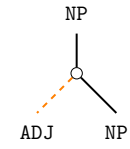
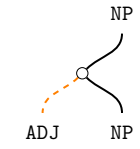
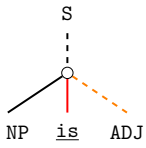
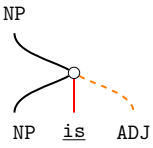
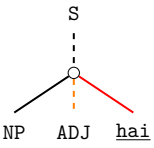
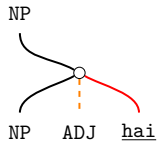
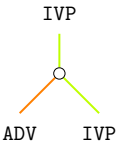
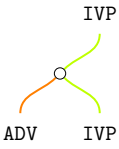
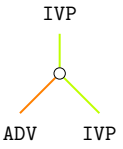
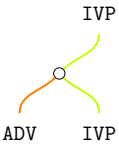
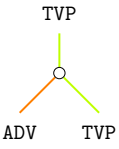
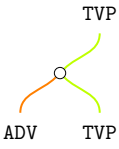
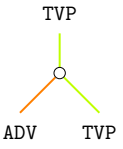
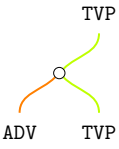
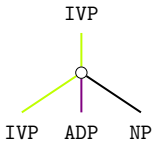
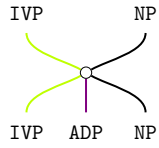
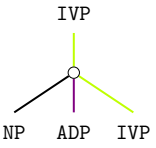
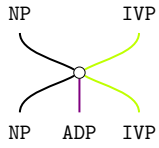
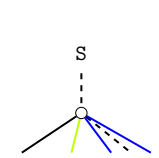
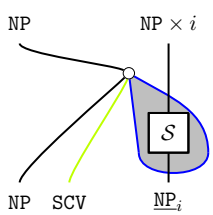
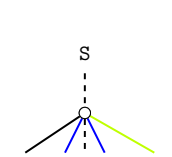
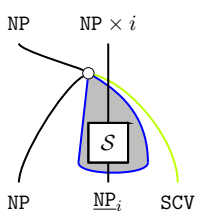
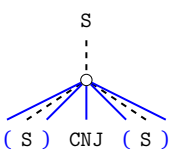
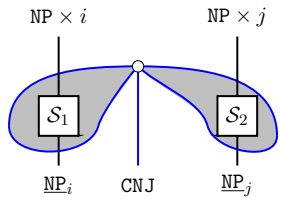
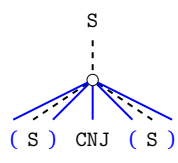
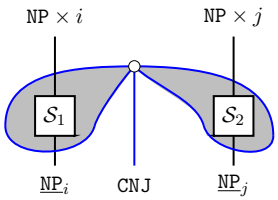
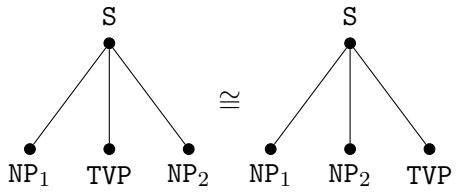
Rule	English grammar	English diagram	Urdu grammar	Urdu diagram
Intrans.Verb				
Trans.Verb				
Adjective(Pre.)				
Adjective(Post.)				
Adverb(IV)				
Adverb(TV)				
Adposition(IV)				
Sent.Comp.Verb				
Conjunction				

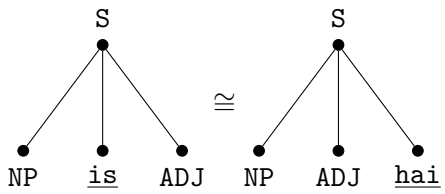
Table 5.2: Generators for hybrid Urdu and English grammar and the corresponding diagrams. Note the different constructions for English and Urdu in the rules labelled red: Trans.Verb, Adjective(Post.), Adposition(IV) and Sent.Comp.Verb.

Proof. Table 5.2 shows the generators of English and Urdu grammar, four of which are different for the two languages. Viewing the generators as rooted labelled trees (vertices are labels like NP or hai, and the root is the initial sentence type S), the generators of English and Urdu for each rule are isomorphic in the graph-theoretic sense.

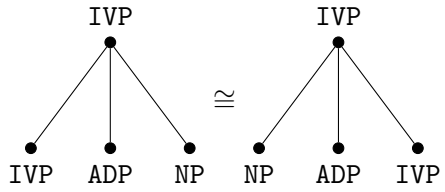
- Transitive verb



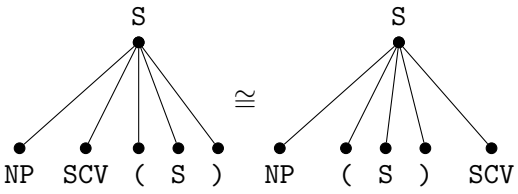
- Postpositional adjective (where hai($\cong$ is))



- Adposition



- Verb with a sentential complement



The transformational grammar rules for Urdu (5.11, 5.12, 5.13) and English ((5.5, 5.6, 5.7) are the same in terms of the connectivity of pronominal links and sentence-order. They differ with respect to the intra-sentence word orders. These differences are eliminated due to the isomorphisms of generators described above. Therefore, essentially the same transformational grammar rules and diagram rewrites apply to English and Urdu.

The aforementioned modifications translate the formal claim of Theorem 5.4.5 [32] for Urdu; the hybrid grammar text for Urdu surjects onto text circuits. □

Remark 5.6.2. The graph-theoretic isomorphism of the generators of English and Urdu grammars is not surprising, as both languages belong to the Indo-European language family [207]. For all the grammatical rules considered, the generators of both languages have the same types, although they may appear in different orderings. For contrast, consider null-subject languages, in which subjects can be omitted [208]. One example is Arabic, where pronouns are dropped because they can be inferred from verb morphology [209]. Therefore, the generators of Arabic grammar—or those of other null-subject languages—are not isomorphic to those of English and Urdu, even for the limited fragment of language considered in this chapter.

5.6.3 English and Urdu Give the Same Circuits

In the case that we only consider context-free generators, the desired isomorphism between English and Urdu circuits essentially follows from the isomorphism between the trees generated by the English and Urdu hybrid grammar.

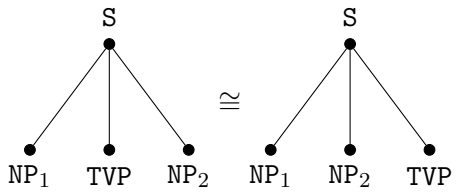
Lemma 5.6.3. Let $\mathtt{T}_{\mathtt{E}}$ denote the set of texts generated by the English production rules E, and let $\mathtt{T}_{\mathtt{U}}$ denote the set of texts generated by the Urdu production rules U. Viewing the syntax trees as rooted labelled trees (vertices are labels like NP or hai, and the root is the initial sentence type S), there is an isomorphism $\mathtt{T}_{\mathtt{E}} \simeq \mathtt{T}_{\mathtt{U}}$ in the graph-theoretic sense.

Proof. Essentially, the Urdu generators U correspond exactly to the English generators E, except some of them have the order of their outputs switched around.

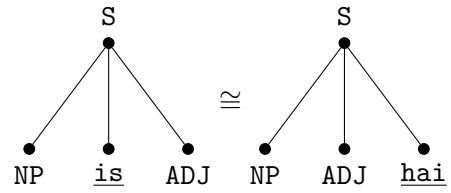
So, given an English syntax tree generated by some sequence of applications of production rules in E, to translate this to an Urdu syntax tree we simply apply the corresponding rules in U to the appropriate symbols. The exact proof of this is a simple induction. Note that in order to 'apply the right rule to the right symbol', we must track the identities of different symbols, *e.g.* distinguishing between different instances of NP in our string. This tracking can be done by numbering the symbols with indices.

Refer to Table 5.2. For the base case, the intransitive verb, the transitive verb, the adjective (postpositional), the sentential complement verb or the conjunction rule can be applied. Of these rules, two are exactly the same for English and Urdu. The rest of the rules generate syntax trees that when viewed as rooted labelled trees are isomorphic graph-theoretically.

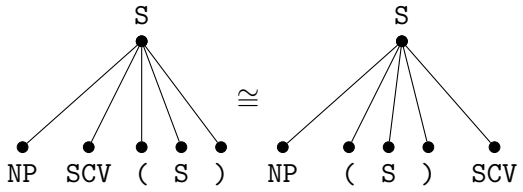
- Transitive verb



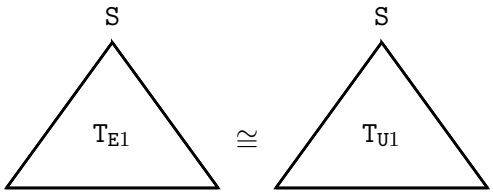
- Postpositional adjective (where hai(≅ is))



- Verb with a sentential complement

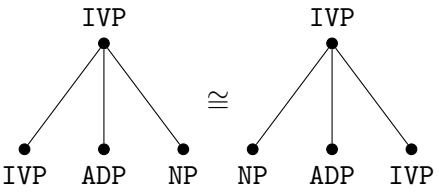


The inductive hypothesis is that each pair of English and Urdu syntax trees generated by n applications of corresponding pairs of production rules is isomorphic.



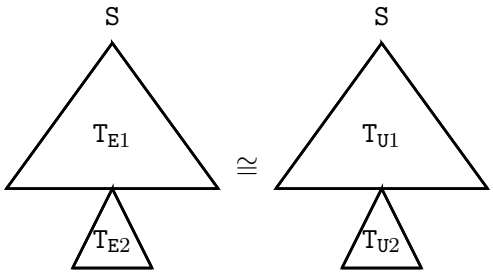
For the induction step, we apply another production rule to the syntax trees generated by the inductive hypothesis. Any of the possible production rules from Table 5.2 can be applied. All except four rules are different for English and Urdu. Of the four, three were shown to be isomorphic in the base case. The last one is also isomorphic in the same sense.

Adposition



Then, using the inductive hypothesis, English and Urdu syntax trees generated by $n + 1$

applications of production rules are isomorphic.



□

Lemma 5.6.4. $\mathtt{C_U}$ is isomorphic to $\mathtt{C_E}$, i.e., $\mathtt{C_U} \xrightarrow{\simeq} \mathtt{C_E}$ (up to word translations at the gate level).

Proof. According to Lemma 5.6.3, there is an isomorphism $\mathtt{T_E} \simeq \mathtt{T_U}$ between individual syntax trees. We introduce pronominal links and the rewrite rules ((5.5, 5.6, 5.7) for English; (5.11, 5.12, 5.13) for Urdu) that allow us to adjoin pronominally-linked trees into single sentences, thus attaining the full-blown hybrid grammar. The transformational rewrite rules for English and Urdu are the same in terms of the connectivity of pronominal links. The rewrite rules differ with respect to the word order in individual syntax trees. This difference is eliminated by Lemma 5.6.3. Therefore, the isomorphism between trees lifts to a kind of isomorphism between these full hybrid grammar structures.

Next, we convert hybrid grammar to text diagrams. We apply the rules in Table 5.2 for resolving the `S` type into the constituent `NP` wires and converting phrase scope into phrase regions. Then we turn the pronominal links into dashed wires (5.8) in preparation for composition. After performing the composition using the rewrite rule (5.9), we recover text diagrams.

The isomorphism of hybrid grammar structures simply lifts to an isomorphism of the structures at each intermediate step. At the step where we reach the level of text diagrams, the isomorphism becomes an exact equality (modulo the different word labels, and allowing a certain degree of topological deformation as we usually do in string diagrams).

With the (topologically) identical structure of the English and Urdu diagrams, we simply apply the same conversion map from text diagrams to text circuits to obtain the same text circuit up to word-translations at the level of individual gates. □

Example 5.6.5. Consider the English sentence `The young student who sees the honest teacher passionately teach smiles at him.` which we translate into the Urdu sentence

نوجوان طالب علم جو ایماندار استاد کو شوق سے پڑھاتے ہوئے دیکھتا ہے اس کی طرف مسکراتا ہے۔

transliterated as:

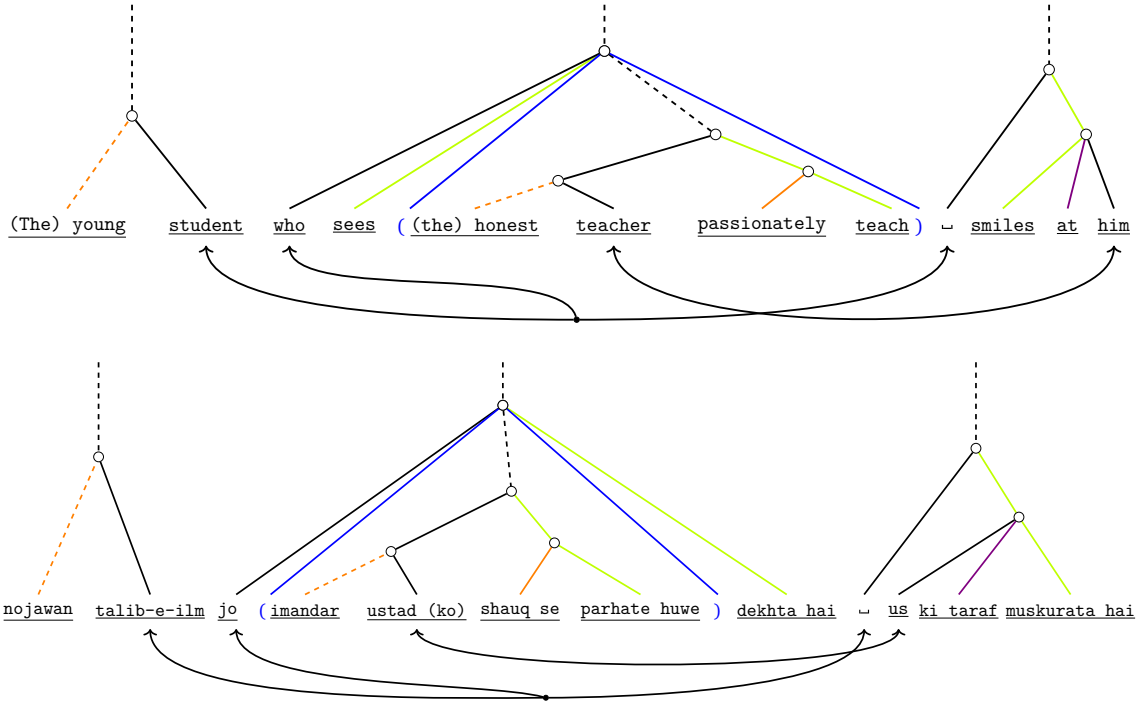
nojawan *talib-e-ilm* *jo* *imandar* *ustad (ko)* *shauq se* *parhate huwe* (Urdu)

(the) young student who (the) honest teacher passionately teach

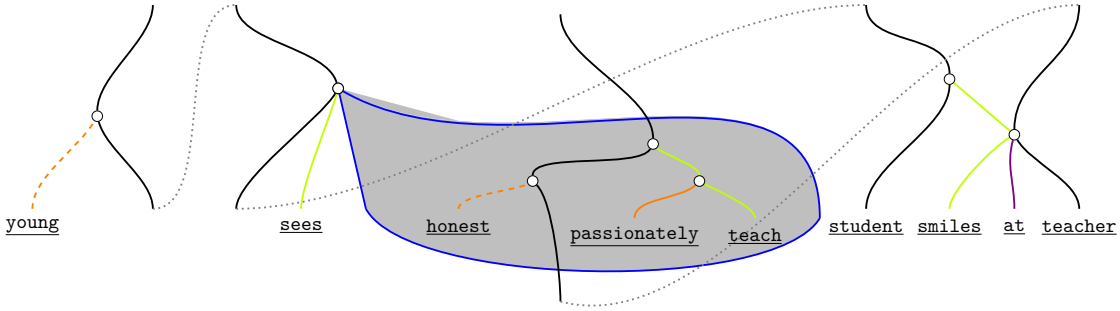
dekhta hai *us* *ki taraf* *muskurata hai* (Urdu)

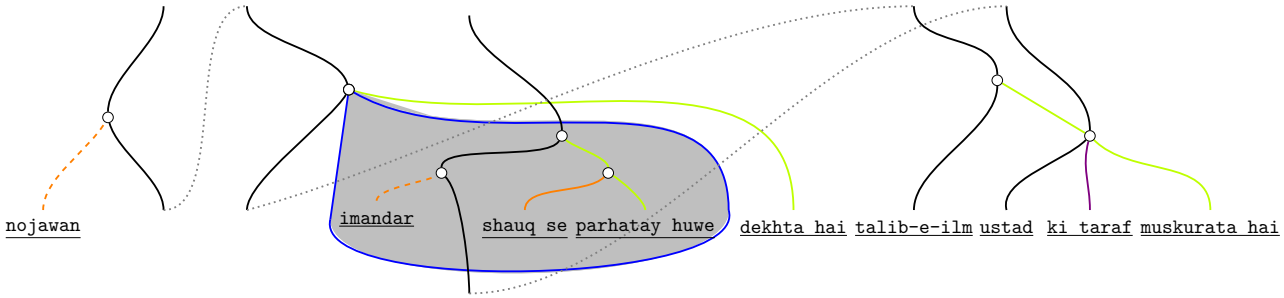
sees him at smiles

We start with the hybrid grammar structure of the English and Urdu sentences.

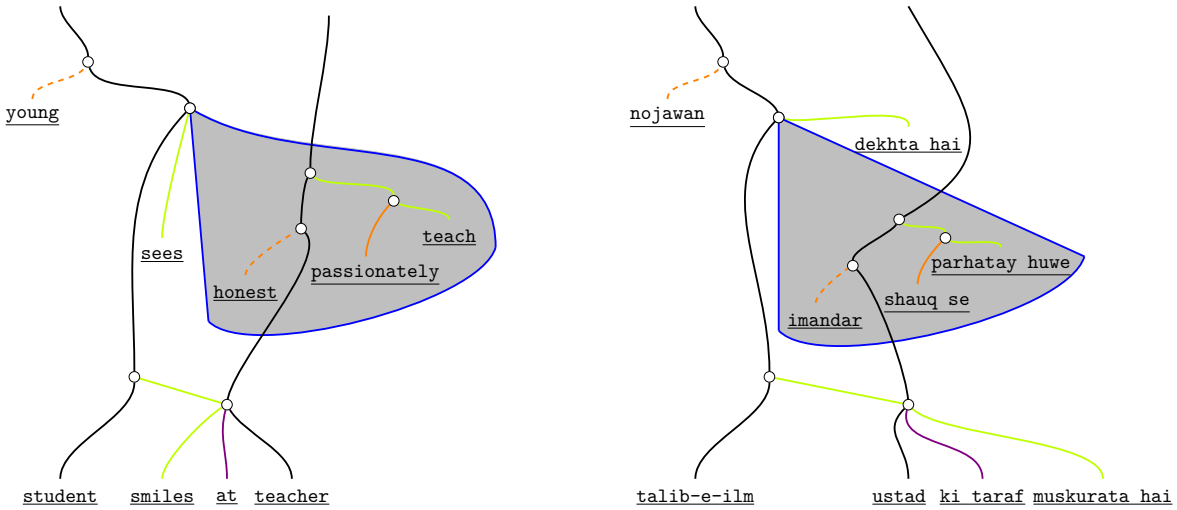


We replace the individual tree diagrams with the corresponding text diagrams, and the pronominal link (multi)arrows with dashed wires.

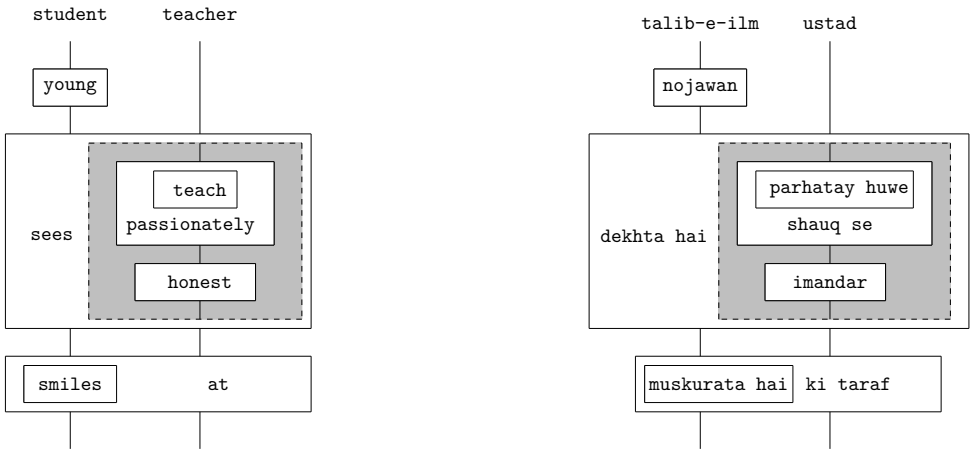




Eliminating the dashed wires with diagram rewrite rules, we obtain text diagrams for the the whole sentences.



At this point, we can see that the text diagrams for English and Urdu have the same topological structure. We, therefore, apply the same conversion map to express text diagrams in terms of gates and boxes, resulting in the same text circuits, up to gate-level translations.



Remark 5.6.6. The process from text to circuits can be (non-deterministically) reversed. This makes text circuits generative formalisms for English and Urdu text. Note that we are only concerned here with the fragment of language described in this chapter.

5.7 Summary and Outlook

In this chapter, we demonstrated how DisCoCirc text circuits eliminate grammatical bureaucracy based on word or phrase order for restricted fragments of English and Urdu. Building on Ref. [32], which established a surjection from English hybrid grammar to English text circuits, we adapted the hybrid grammar for Urdu and defined translation rules into text circuits that mirror those for English. This yields a surjective map from Urdu hybrid grammar to Urdu text circuits. For the language fragments considered, the English and Urdu hybrid grammars are isomorphic. Using this isomorphism, we show that their corresponding text circuits are structurally identical, differing only in gate labels.

A major difference between grammars in different natural languages arises from different word orderings for, for example, subject, verb and object. For instance, in English the usual ordering is subject-verb-object (SVO), whereas in Urdu, it is subject-object-verb (SOV). These differences, in turn, exist because human verbal communication is restricted to one dimension and different cultures and demographics made different stylistic choices as languages evolved [32]. But there is no such restriction on machines. Two-dimensional grammars such as ours may be a suitable abstraction of text for computers, particularly quantum computers [193, 194], and may prove advantageous for natural language processing tasks, such as machine translation. This chapter takes a step forward in this direction. Our work represents a first step, and there is work to be done to expand the fragments of natural language that we can handle.

A limitation of this work is that both English and Urdu belong to the Indo-European language family and share similar grammatical structures, including a clear noun–verb distinction [207]. While they differ in word order—English follows SVO and Urdu follows SOV—the generators of their grammars are graph-theoretically isomorphic for the fragment considered, making the main results of this work somewhat unsurprising. In contrast, extending this framework to typologically distinct languages, such as null-subject languages like Arabic [208, 209], presents a greater challenge. These languages allow subject omission due to rich verb morphology, resulting in grammatical generators that are not isomorphic to those of English or Urdu. Addressing such structural differences is an important direction for future work.

6
Conclusion

> 'The compartments into which human thought is divided are not so water–tight that fundamental progress in one is a matter of indifference to the rest.'
>
> Arthur Eddington,
> The Philosophy of Physical Science [210, p. 8]

In this chapter, we summarise the key contributions described in this thesis and discuss avenues for future research. At the end of this chapter, we make a case for the prospects of process-relational philosophy in science.

A process-theoretic approach to constructor theory and quantum physics

In Chapter 3, we describe three main contributions.

Firstly, we present categorical semantics for constructor theory while adhering to the desired mathematical foundations described in the seminal article [46] on constructor theory. Taking the theory of conceivable tasks to be the symmetric monoidal category (SMC) **Rel**, we show that, for a given choice of substrates, the set of possible tasks forms a sub-SMC of **Rel**.

Secondly, we argue that in the constructor-theoretic formulation of non-relativistic quantum theory, there is an inconsistency between the restrictions imposed by the principles of locality and composition. Constructor theorists argue that there exists a local formulation of quantum theory: the Deutsch-Hayden approach [49]. We demonstrate, using concrete examples, that this approach is not compositional.

Thirdly, we discuss that a key difference between constructor theory and process theories concerns the principle of locality. We argue that categorical quantum mechanics (CQM) can

be conceived as a constructor theory of quantum physics that is compositional but non-local.

Future work

We recommend several directions for future research:

- A key future direction is to develop process-theoretic formulations of constructor theories in domains such as information [102], thermodynamics [107], probability [106], and life [105]. This will clarify their foundational principles and interconnections, enabling a unified understanding of processes across these fields.

- We also propose investigating the relationship between constructor theory and resource theories [124, 211]. Exploring this connection could yield new insights into how resources are transformed and conserved in physical and informational processes, enriching both frameworks and opening new avenues for application.

- The string-diagrammatic syntax of process theories can also be interpreted in SMCs other than the one discussed in this thesis. This provides an opportunity to investigate and explore the ramifications of constructor theory beyond **Rel**, which has been the default modelling choice in the constructor theory literature so far [46, 121].

Wave-based computation in diagrams

In Chapter 4, we develop a string-diagrammatic formalism for wave-based logic circuits with phase encoding. The formalism is motivated with reference to spin-wave or 'magnonic' circuits. Through the example of spin-wave circuits, the usage of the formalism in designing, analysing and simplifying Boolean logic circuits is demonstrated.

Future work

Some directions for future research are outlined below:

- To extend this work, we would anticipate developing formal semantics for the string diagrams presented here, and proving that the formalism is sound and complete for Boolean algebra.

- Another avenue for future work is to conceive of the formalism as a diagrammatic alternative to symbolic Boolean algebra, wherein lies the possibility of a new axiomatisation of Boolean algebra.

- This work has proposed a formalism for the theoretical modelling of spin-wave circuits, which abstracts away many hardware-level complexities. However, spin-wave computing inherits core limitations of analogue systems, such as noise susceptibility, signal degradation, error propagation, and energy-intensive error correction. These issues are not fully addressed in the current framework. Future extensions may need to incorporate error-correction mechanisms—typically involving non-linear operations [181]—and amplitude normalization for cascaded logic gates, which is also non-linear [182]. Additionally, modelling crosstalk noise at smaller scales and integrating with platforms such as optical systems, superconducting qubits, or CMOS technologies [148, 160, 183] will be crucial for improving the framework's practical applicability. Addressing these aspects will enhance its relevance to real-world applications and contribute to more scalable and resilient spin-wave computing architectures.

DisCoCirc beyond English text

In Chapter 5, we show how word- or phrase-order-based grammatical bureaucracy is eliminated by DisCoCirc text circuits for restricted fragments of English and Urdu.

We describe a restricted version of the hybrid grammar, text diagrams and text circuits for English developed in Ref. [32]. According to Ref. [32], there is a surjection from the set of all English text generated with the English hybrid grammar to the set of all English text circuits. In a similar vein, in this thesis, we describe how the hybrid grammar can be adapted for Urdu. We then provide rules for its translation into text diagrams and text circuits, which are essentially the same as those in Ref. [32]. We show that this gives a surjective map from the set of all Urdu text generated with Urdu hybrid grammar to the set of all Urdu text circuits.

Furthermore, for the language fragments considered in this thesis, there is a clear isomorphism between the hybrid grammars for English and Urdu. Using this isomorpism, we show that text circuits for English and Urdu become the same, up to translation of gate-labels.

Future work

Our work can be extended in several directions:

- This thesis focused on a limited fragment of English and Urdu. Future work can extend this to broader language fragments by incorporating additional grammatical features such as multiple tenses, reflexive pronouns, determiners etc.

- The primary syntactic variation in the language fragments studied lies in word order: English follows SVO, while Urdu follows SOV.[1] Any language that differs from English or Urdu only in the word order of simple sentences can potentially be modelled using DisCoCirc. Hybrid grammars for such languages can be constructed to fit within the DisCoCirc framework. Future research can pursue both a theoretical characterisation of all possible DisCoCirc languages and an empirical investigation into which of them correspond to real-world languages. DisCoCirc, as a two-dimensional framework, abstracts away from the linear order of words and thus supports language-independent grammatical modelling.

- A limitation of this work is that both English and Urdu are Indo-European languages with comparable grammatical structures, such as clear noun-verb distinctions [207]. Although they differ in word order, their grammatical generators for the studied fragment are graph-theoretically isomorphic, resulting in English and Urdu texts mapping to the same circuits. Extending DisCoCirc to typologically distinct languages, such as null-subject languages like Arabic [208, 209], presents additional challenges. These languages allow subject omission due to rich verbal morphology, leading to non-isomorphic grammatical structures. Addressing such typological variation is essential for broadening the applicability of the DisCoCirc framework.

- The relationship between DisCoCirc and other linguistic formalisms—especially discourse representation theory [213], which shares conceptual similarities—requires further exploration. Some initial work in this direction appears in Ref. [32].

[1] Different natural languages can share the same grammatical structure, including word order. For instance, Hindi and Urdu have nearly identical grammars but different vocabularies [212]. The results obtained for Urdu in this thesis directly apply to Hindi for the considered fragment.

An invitation to process-relational philosophy

As mentioned in Chapter 1, this thesis can be considered a proof of concept of applied process-relational philosophy.

CQM, reviewed in Chapter 2, is a process-theoretic formulation of quantum physics. It suggests that the most natural ontology for quantum physics centres on processes and their interactions, rather than on states and their dynamics.

Chapter 3 shows how constructor theory is essentially process-theoretic, from both a conceptual and a mathematical point of view. This implies that constructor theories of information [102], thermodynamics [107], probability [106], and life [105] are potential process theories as well. Concrete details of this claim will be the subject of future work. With regards to quantum theory, Chapter 3 argues how the principles of locality and compositionality cannot coexist. Furthermore, discarding the principle of locality, CQM achieves the desiderata of a constructor theory for quantum physics—which is, of course, process-theoretic by construction.

Chapter 4 develops a formalism for wave-based logic circuits in which processes are primitive. Logic variables are initialised as phase-shifting processes. Composing these phase shifting processes gives rise to different logic gates depending on the topological structure of the composite processes.

Chapter 5 describes two-dimensional circuits to model grammar, meaning, and their interaction in natural languages. The key assumption underlying DisCoCirc is that the meaning of a piece of text is determined by how it updates the meanings of the words within it. In DisCoCirc, this assumption is implemented process-theoretically using text circuits. The meanings of nouns (denoted by wires) are updated by processes like verbs and adjectives (represented by gates), the meanings of which are in turn altered by processes like adverbs and adpositions (represented by higher order gates). DisCoCirc offers prospects in quantum natural language processing [193, 194], where the work on quantum processes and computation discussed in Chapters 2 and 3 is of immediate relevance.

The person most notably associated with process philosophy in recent history is Alfred North Whitehead. Whitehead's treatise *Process and Reality* [214] was published in 1929, but was based on his 1927-28 lectures at the University of Edinburgh. Quantum mechanics,

as we know it today, was born in the mid-1920s and undoubtedly influenced Whitehead's work. Whitehead was a mathematician first and a philosopher later; however, he did not have access to the tools of applied category theory—which are arguable particularly well-suited to process ontology, relationalism and compositionality. The absence of an appropriate accompanying mathematical framework may be one reason process philosophy has not yet caught on in scientific research.

Category theory, developed in the mid-1940s [215, 216], was initially of interest in pure mathematics only and was considered very abstract. In the last two decades, it has found application in scientific domains as wide-ranging as quantum physics and quantum computing [20–24, 217], chemistry [218–220], information theory [221], electrical circuit theory [25, 222], photonics [28, 29], linguistics [30–32], game theory [40–42], control theory [33, 34], thermostatics [223], cryptography [224, 225], probability theory [226], functional programming [227], machine learning [38, 228] etc. This recent surge of interest in applied category theory [10] is perhaps due to the development of graphical calculi like string diagrams that marry visual intuition with mathematical rigour.

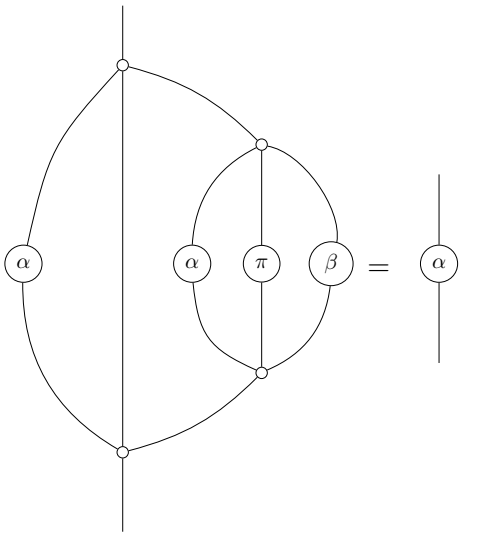
String diagrams are a boon not only to professional scientists but also to high school students. In a recent study [44], students aged 16 to 18 were taught quantum physics and quantum computing using the ZX-calculus [23]. The excellent performance of the students (over 80% pass-rate and around half the number of students earning a distinction) on a postgraduate-level quantum theory exam [45] underscores the potential impact of the string-diagrammatic formalisms on science education.

The author of this thesis invites you to the research area of process-relational philosophy and its application to the sciences. A powerful mathematical toolbox is available to support the accompanying conceptual change of worldview. At best, this research area will help make concrete progress by solving problems or making discoveries. At worst, it will provide a new perspective to understand our existing scientific theories and thereby also learn something about nature. There is nothing to lose.

A

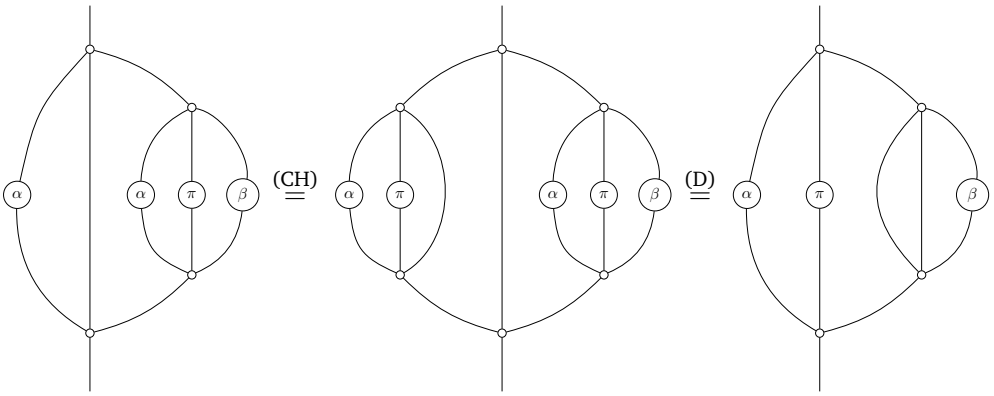
Wave Logic Circuits: Derivation of the Associativity Rule

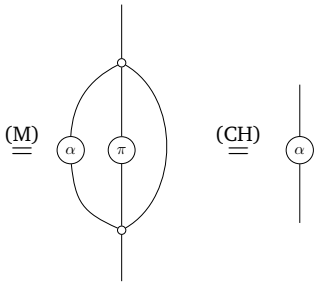
Proposition A.0.1. For all $\alpha, \beta \in \{0, \pi\}$,



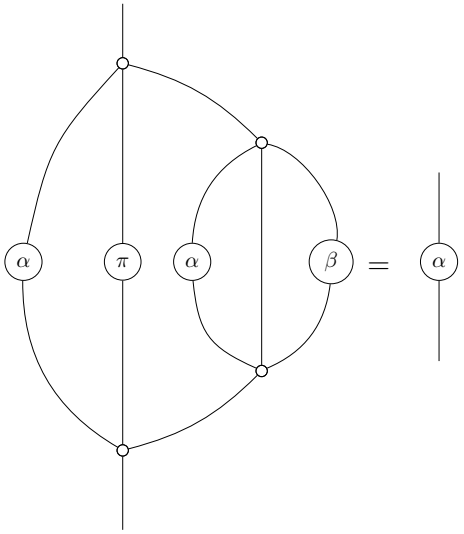
.

Proof.



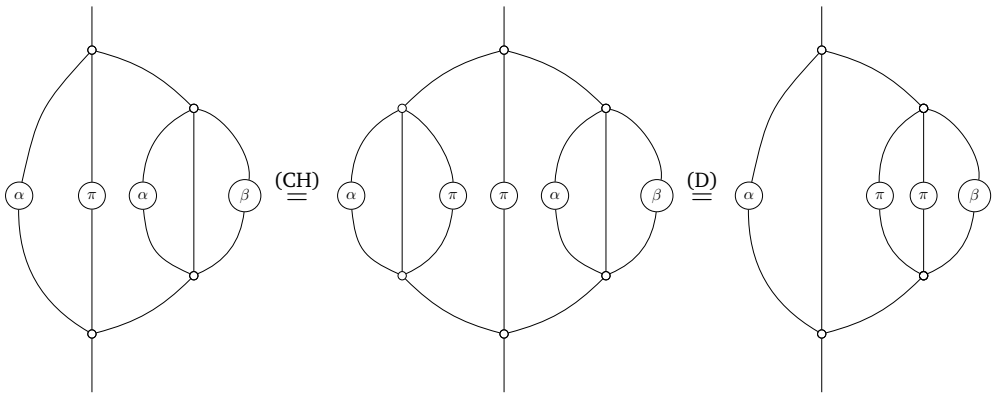


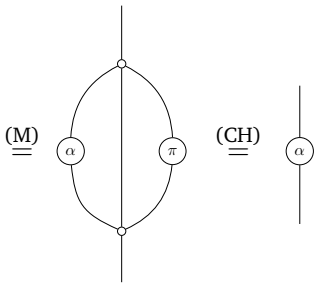
□

Proposition A.0.2. For all $\alpha, \beta \in \{0, \pi\}$, 

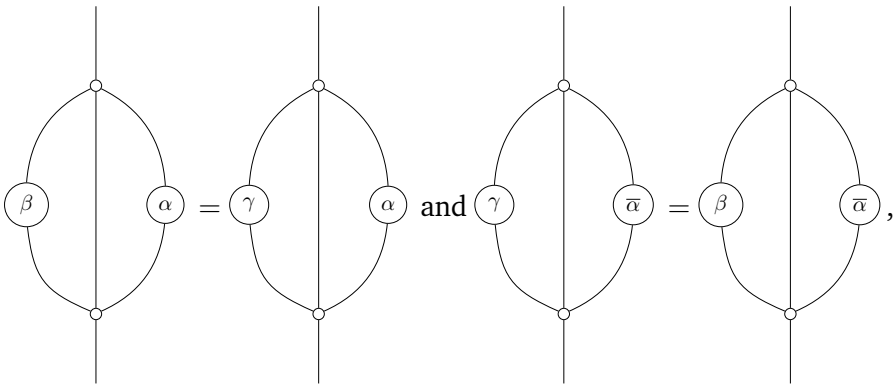
.

Proof.

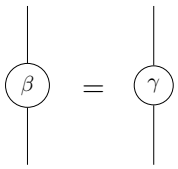




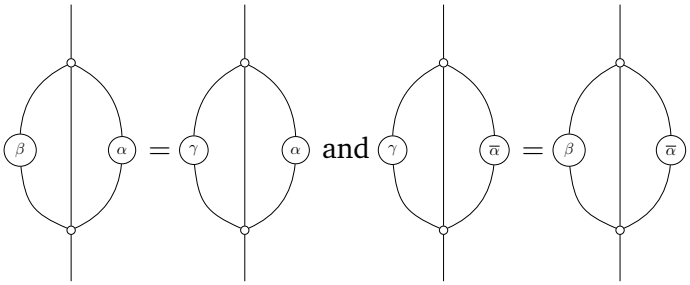
□

Proposition A.0.3. If 

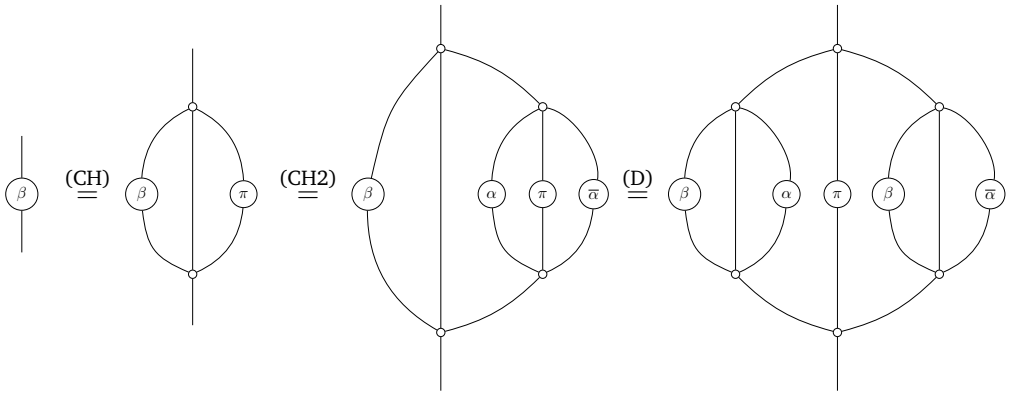
,

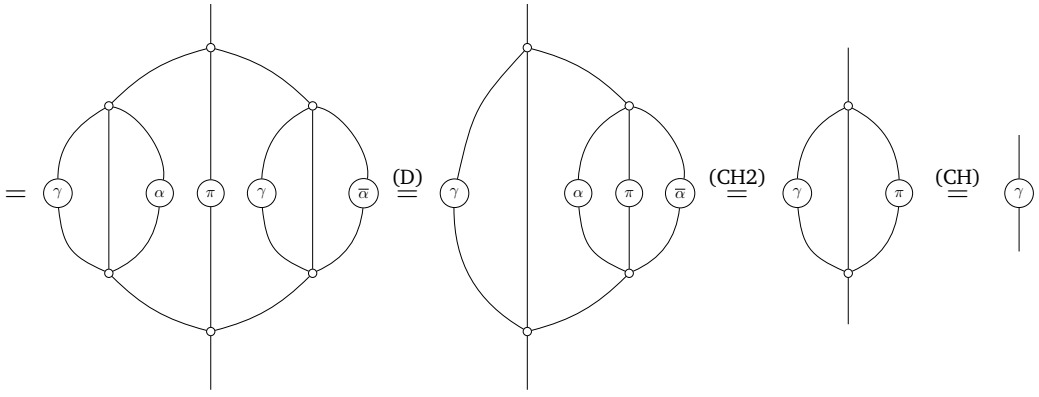
then 

.

Proof. Assume 

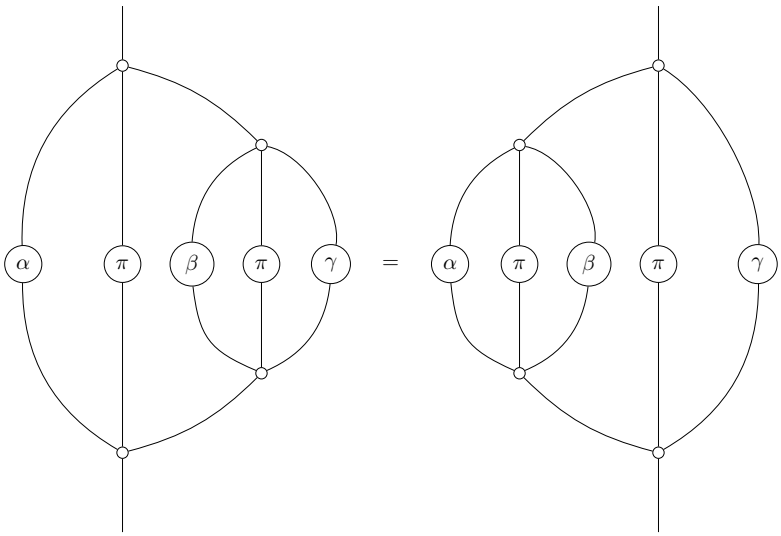
. Then, we have

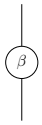
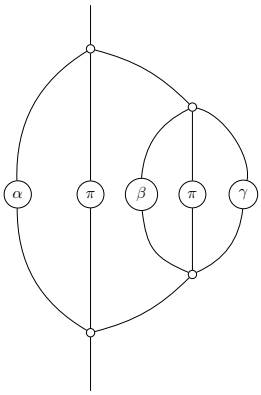
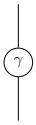
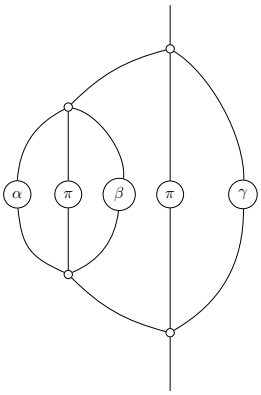




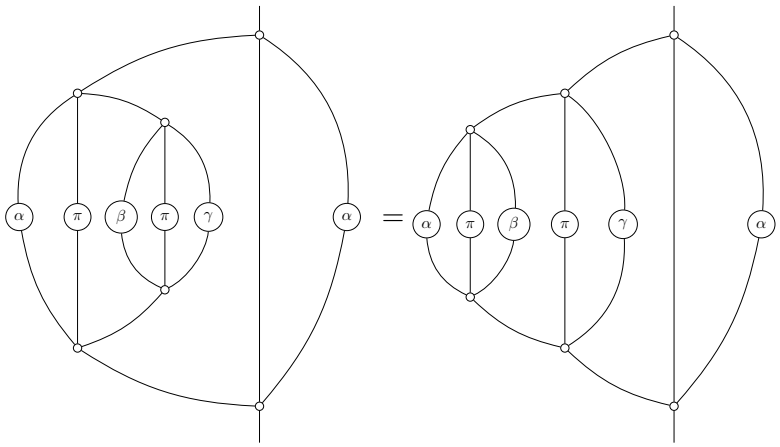
□

Proposition A.0.4. For all $\alpha, \beta, \gamma \in \{0, \pi\}$,

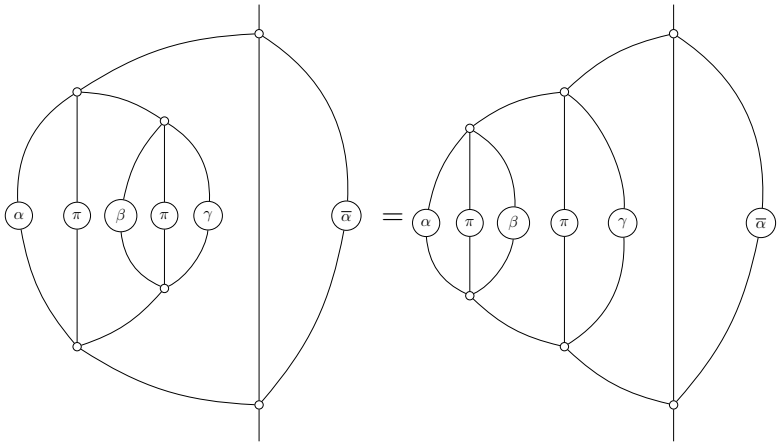


Proof. In Proposition (A.0.3), replace  by  and  by  .

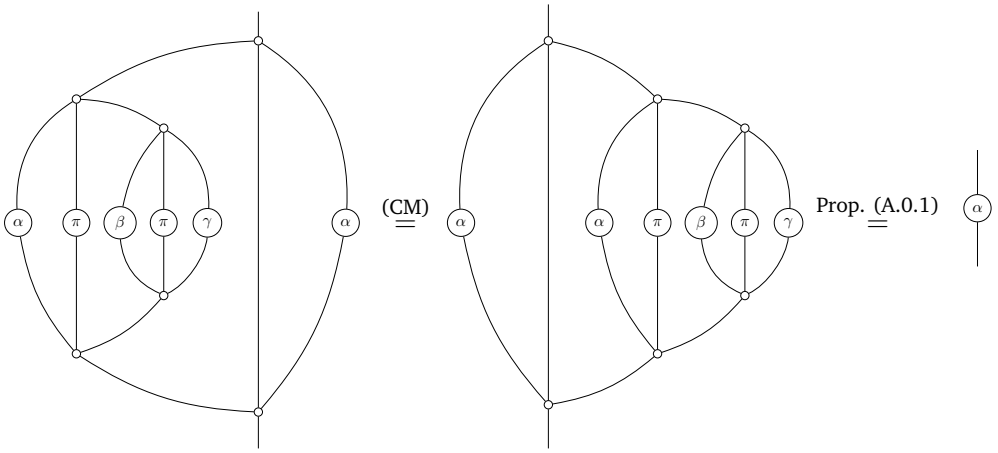
Now, we need to show:

(a) 

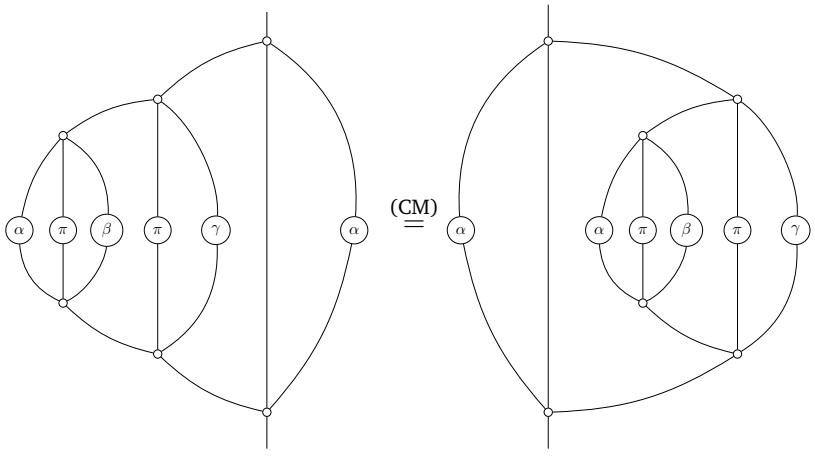
and (b)

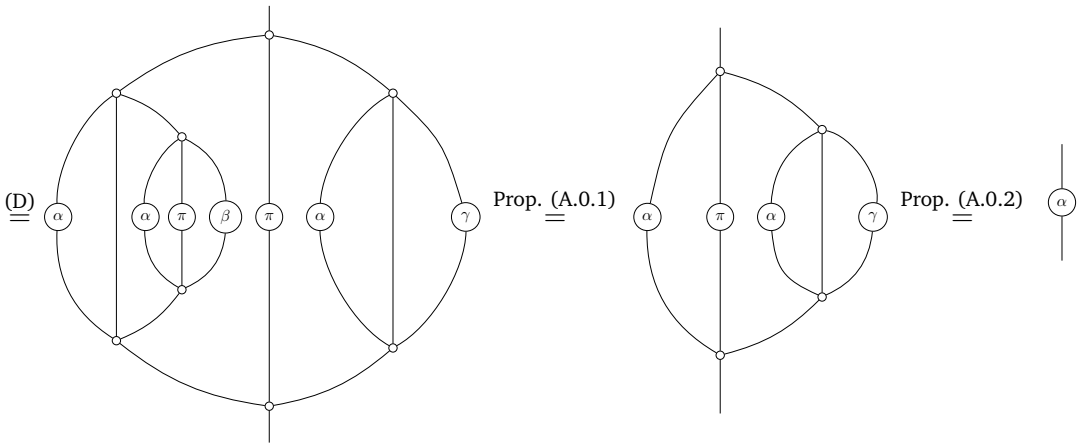


(a) $L.H.S. =$

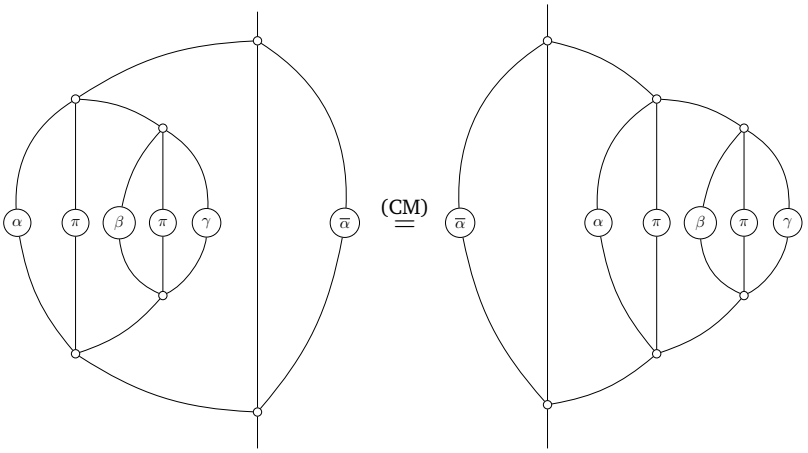


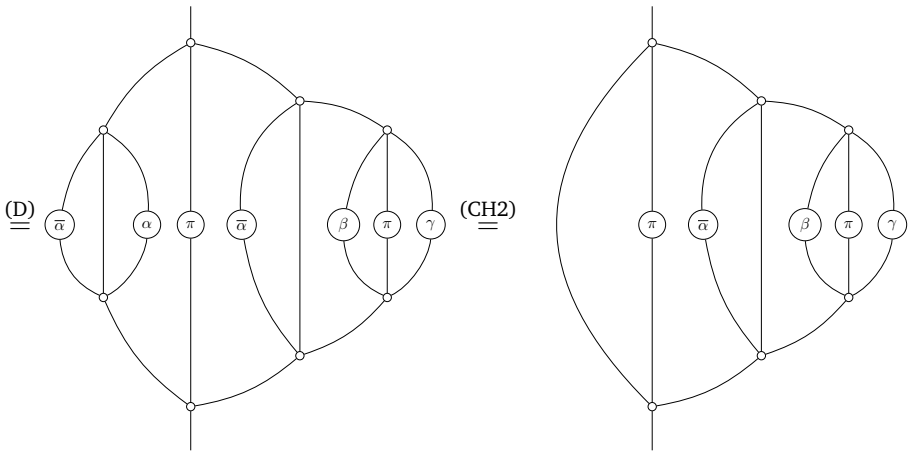
$R.H.S. =$

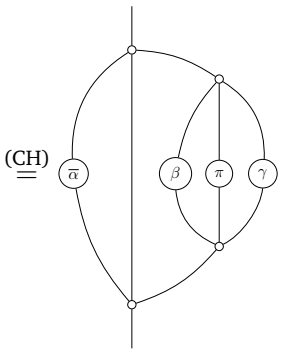


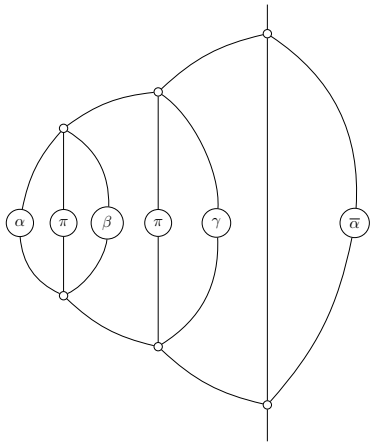


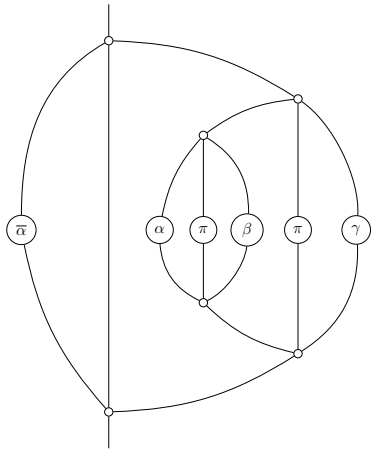
(b) $L.H.S =$

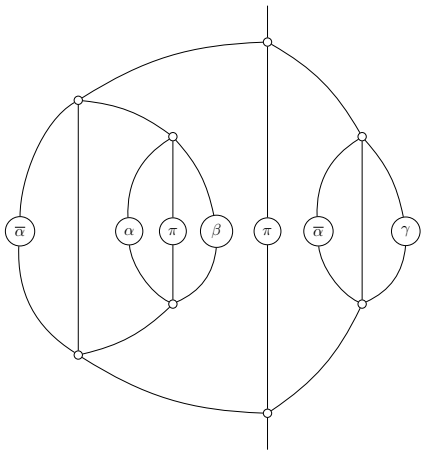


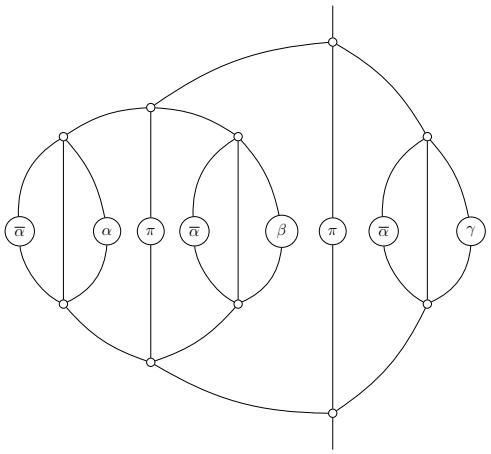


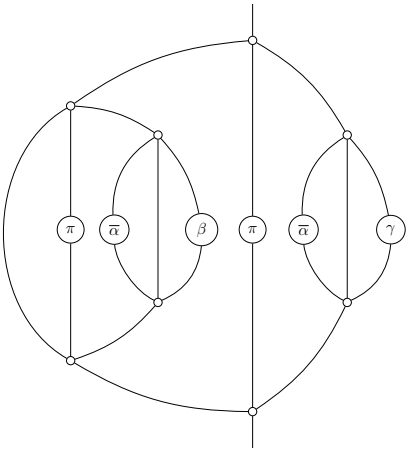


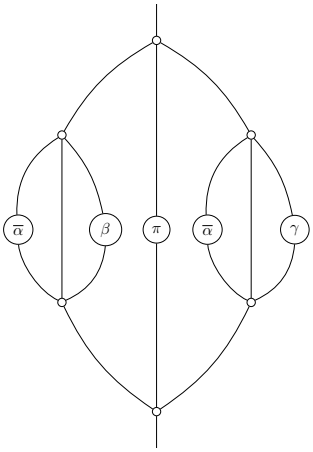
$R.H.S. =$ 

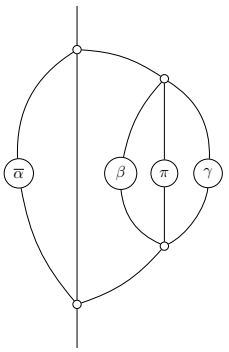
(CM) $=$ 

(D) $=$ 

(D) $=$ 

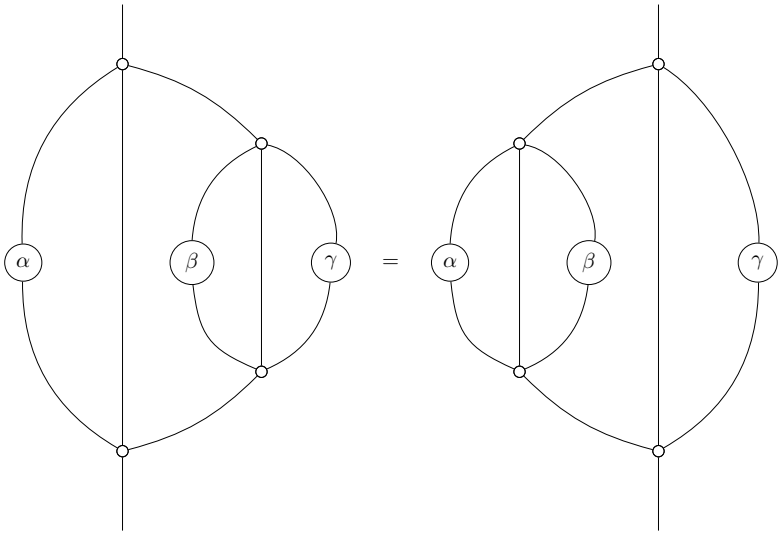
(CH2) $=$ 

(CH) $=$ 

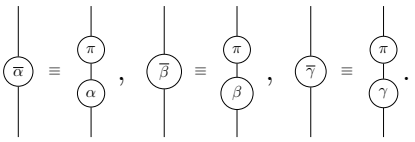
(D) $=$ 

□

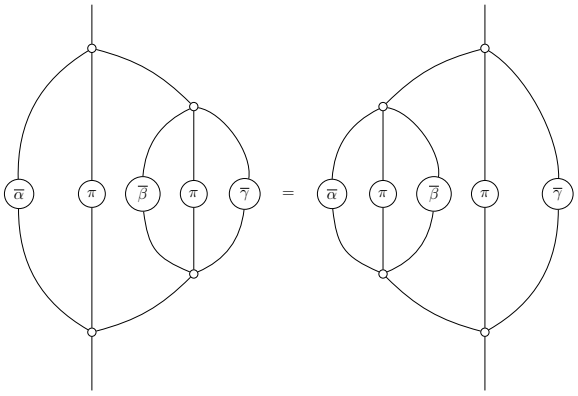
Proposition A.0.5. For all $\alpha, \beta, \gamma \in \{0, \pi\}$,



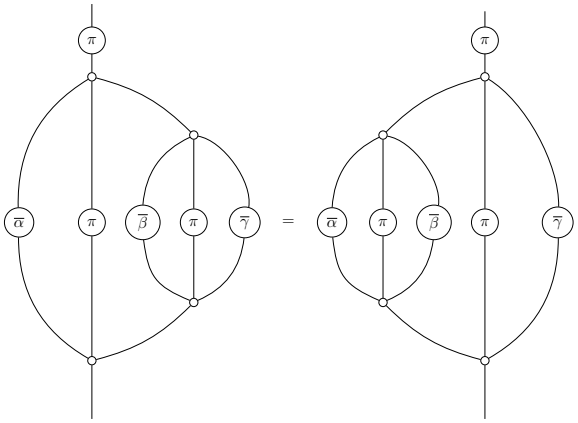
Proof. Define complements of α, β and γ:

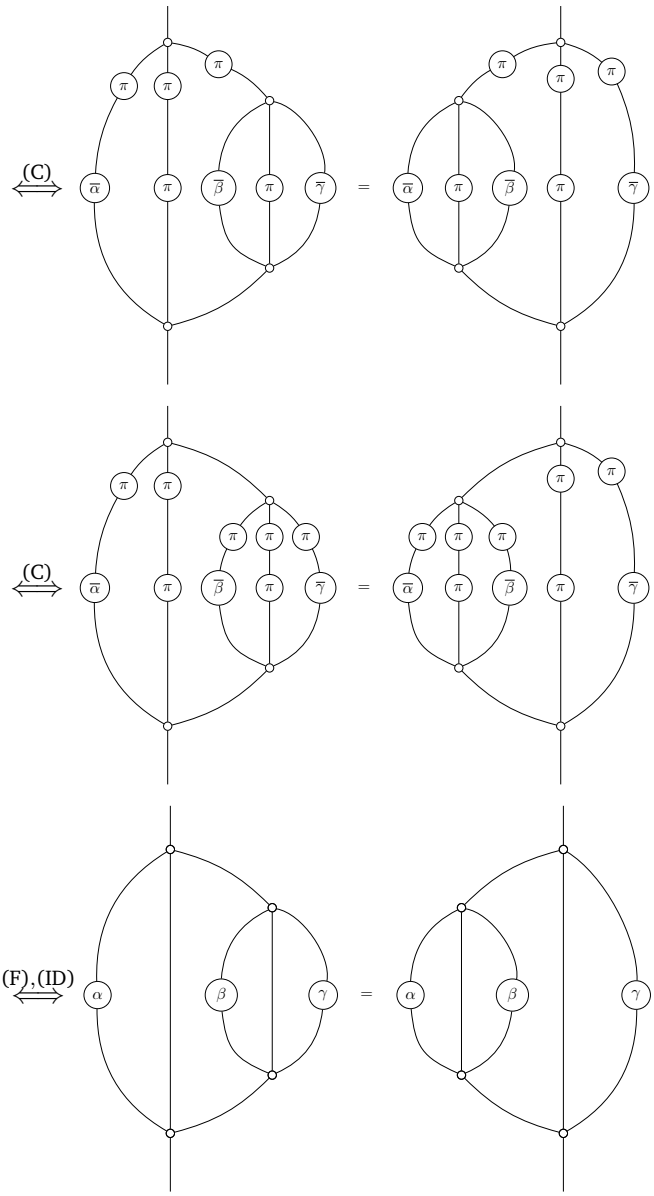


Then, from Proposition (A.0.4), we have



Taking the complement of both sides, we get




(C)
(C)
(F),(ID)

□

References

[1] H. Poincaré. *Science and Hypothesis*. London and Newcastle-on-Tyne: The Walter Scott Publishing Co., Ltd., 1905.

[2] J. Seibt. “Process Philosophy”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Spring 2024. Metaphysics Research Lab, Stanford University, 2024.

[3] S. Berryman. “Democritus”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Spring 2023. Metaphysics Research Lab, Stanford University, 2023.

[4] J. Palmer. “Parmenides”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2020. Metaphysics Research Lab, Stanford University, 2020.

[5] D. W. Graham. “Heraclitus”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Winter 2023. Metaphysics Research Lab, Stanford University, 2023.

[6] J. K. McDonough. “Leibniz’s philosophy of physics”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Fall 2024. Metaphysics Research Lab, Stanford University, 2024.

[7] N. Rescher. *Process Metaphysics: An Introduction to Process Philosophy*. State University of New York Press, 1996.

[8] R. Desmet and A. D. Irvine. “Alfred North Whitehead”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Winter 2022. Metaphysics Research Lab, Stanford University, 2022.

[9] E. Schrödinger. “Discussion of probability relations between separated systems”. In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 31. 4. Cambridge University Press. 1935, pp. 555–563.

[10] B. Fong and D. I. Spivak. *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. Cambridge University Press, 2019.

[11] N. S. Yanofsky. *Monoidal Category Theory: Unifying Concepts in Mathematics, Physics, and Computing*. MIT Press, 2024.

[12] R. Penrose. “Applications of negative dimensional tensors”. In: *Combinatorial mathematics and its applications* 1 (1971), pp. 221–244.

[13] A. Joyal and R. Street. “The geometry of tensor calculus, I”. In: *Advances in mathematics* 88.1 (1991), pp. 55–112.

[14] P. Selinger. “A survey of graphical languages for monoidal categories”. In: *New Structures for Physics*. Ed. by B. Coecke. Lecture Notes in Physics. arXiv:0908.3347. Springer-Verlag, 2011, pp. 275–337.

[15] R. Piedeleu and F. Zanasi. *An Introduction to String Diagrams for Computer Scientists*. Elements in Applied Category Theory. Cambridge University Press, 2025.

[16] R. Feynman. “Space-time approach to quantum electrodynamics”. In: *Physical Review* 76.6 (1949), pp. 769–789.

[17] R. Feynman. “The theory of positrons”. In: *Physical Review* 76.6 (1949), pp. 749–759.

[18] G. Hotz. “Eine algebraisierung des syntheseproblems von schaltkreisen I.” In: *Elektronische Informationsverarbeitung und Kybernetik* 1.3 (1965), pp. 185–205.

[19] D. Pavlovic. *Programs as Diagrams: From Categorical Computability to Computable Categories*. Springer Nature, 2023.

[20] B. Coecke. “Kindergarten quantum mechanics: lecture notes”. In: *AIP Conference Proceedings*. Vol. 810. 1. American Institute of Physics. 2006, pp. 81–98.

[21] B. Coecke, D. Horsman, A. Kissinger, and Q. Wang. “Kindergarden quantum mechanics graduates... or how I learned to stop gluing LEGO together and love the ZX-calculus”. In: *Theoretical Computer Science* 897 (2022), pp. 1–22.

[22] B. Coecke and A. Kissinger. *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[23] B. Coecke and S. Gogioso. *Quantum in Pictures*. Quantinuum, 2022.

[24] B. Coecke. “Basic ZX-calculus for students and professionals”. In: *arXiv preprint arXiv:2303.03163* (2023).

[25] G. Boisseau and P. Sobocinski. “String diagrammatic electrical circuit theory”. In: *Proceedings of the Fourth International Conference on Applied Category Theory, ACT 2021, Cambridge, United Kingdom, 12-16th July 2021*. Ed. by K. Kishida. Vol. 372. EPTCS. 2021, pp. 178–191.

[26] D. R. Ghica and A. Jung. “Categorical semantics of digital circuits”. In: *2016 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE. 2016, pp. 41–48.

[27] D. R. Ghica, G. Kaye, and D. Sprunger. “A complete theory of sequential digital circuits: denotational, operational and algebraic semantics”. In: *arXiv preprint arXiv:2201.10456* (2025).

[28] G. de Felice and B. Coecke. “Quantum linear optics via string diagrams”. In: *Electronic Proceedings in Theoretical Computer Science* 394 (2023), pp. 83–100.

[29] A. Clément, N. Heurtel, S. Mansfield, S. Perdrix, and B. Valiron. “LO_{v}-calculus: a graphical language for linear optical quantum circuits”. In: *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*. Ed. by S. Szeider, R. Ganian, and A. Silva. Vol. 241. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 35:1–35:16.

[30] S. Clark, B. Coecke, and M. Sadrzadeh. “A compositional distributional model of meaning”. In: *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*. Oxford. 2008, pp. 133–140.

[31] B. Coecke. “The mathematics of text structure”. In: *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*. Springer, 2021, pp. 181–217.

[32] V. Wang-Maścianica, J. Liu, and B. Coecke. “Distilling text into circuits”. In: *arXiv preprint arXiv:2301.10595* (2023).

[33] J. C. Baez and J. Erbele. “Categories in control”. In: *Theory and Applications of Categories* 30.24 (2015), pp. 836–881.

[34] J. M. Erbele. “Categories in control: applied PROPs”. In: *arXiv preprint arXiv:1611.07591* (2016).

[35] R. Lorenz and S. Tull. “Causal models in string diagrams”. In: *arXiv preprint arXiv:2304.07638* (2023).

[36] K. Cho and B. Jacobs. “Disintegration and Bayesian inversion via string diagrams”. In: *Mathematical Structures in Computer Science* 29.7 (2019), pp. 938–971.

[37] T. Fritz, T. Gonda, and P. Perrone. “De Finetti’s theorem in categorical probability”. In: *Journal of Stochastic Analysis* 2.4 (2021), p. 6.

[38] D. Shiebler, B. Gavranović, and P. Wilson. “Category theory in machine learning”. In: *arXiv preprint arXiv:2106.07032* (2021).

[39] A. Koziell-Pipe and A. Kissinger. “Hybrid quantum-classical machine learning with string diagrams”. In: *arXiv preprint arXiv:2407.03673* (2024).

[40] J. Hedges. “String diagrams for game theory”. In: *arXiv preprint arXiv:1503.06072* (2015).

[41] J. Hedges, E. Shprits, V. Winschel, and P. Zahn. “Compositionality and string diagrams for game theory”. In: *arXiv preprint arXiv:1604.06061* (2016).

[42] N. Ghani, J. Hedges, V. Winschel, and P. Zahn. “Compositional game theory”. In: *Proceedings of the 33rd annual ACM/IEEE symposium on logic in computer science*. 2018, pp. 472–481.

[43] B. Coecke. “Quantum Picturalism”. In: *Contemporary Physics* 51 (2009). arXiv:0908.1787, pp. 59–83.

[44] S. Dündar-Coecke, L. Yeh, C. Puca, S. M.-L. Pfaendler, M. H. Waseem, T. Cervoni, A. Kissinger, S. Gogioso, and B. Coecke. “Quantum Picturalism: learning quantum theory in high school”. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Vol. 3. IEEE. 2023, pp. 21–32.

[45] S. Dündar-Coecke, C. Puca, L. Yeh, M. H. Waseem, E. M. Pothos, T. Cervoni, S. M. .-. Pfaendler, V. Wang-Maścianica, P. Sigrist, F. Tomassini, V. Anandraj, I. Khan, S. Gogioso, A. Kissinger, and B. Coecke. “Making the quantum world accessible to young learners through Quantum Picturalism: an experimental study”. In: *arXiv preprint arXiv:2504.01013* (2025).

[46] D. Deutsch. “Constructor theory”. In: *Synthese* 190.18 (2013), pp. 4331–4359.

[47] A. Mahmoud, F. Ciubotaru, F. Vanderveken, A. V. Chumak, S. Hamdioui, C. Adelmann, and S. Cotofana. “Introduction to spin wave computing”. In: *Journal of Applied Physics* 128.16 (2020), p. 161101.

[48] S. Gogioso, V. Wang-Maścianica, M. H. Waseem, C. M. Scandolo, and B. Coecke. “Constructor theory as process theory”. In: *Electronic Proceedings in Theoretical Computer Science* 397 (2023), pp. 137–151.

[49] D. Deutsch and P. Hayden. “Information flow in entangled quantum systems”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 456.1999 (2000), pp. 1759–1774.

[50] M. H. Waseem and A. D. Karenowska. “String diagrams for wave-based computation”. In: *Applied Physics Letters* 123.24 (2023).

[51] M. H. Waseem, J. Liu, V. Wang-Maścianica, and B. Coecke. “Language-independence of DisCoCirc’s text circuits: English and Urdu”. In: Proceedings *End-to-End Compositional Models of Vector-Based Semantics,* NUI Galway, 15-16 August 2022. Ed. by M. Moortgat and G. Wijnholds. Vol. 366. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2022, pp. 50–60.

[52] M. H. Waseem, Faizan-e-Ilahi, and M. S. Anwar. *Quantum Mechanics in the Single Photon Laboratory*. IOP Publishing, 2020.

[53] M. S. Anwar, Faizan-e-Ilahi, S. B. Hyder, and M. H. Waseem. *Quantum Mechanics in the Single-Photon Laboratory*. IOP Publishing, 2024.

[54] M. Rédei. "Why John von Neumann did not like the Hilbert space formalism of quantum mechanics (and what he liked instead)". In: *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics* 27.4 (1996), pp. 493–510.

[55] S. Abramsky and B. Coecke. "A categorical semantics of quantum protocols". In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* IEEE. 2004, pp. 415–425.

[56] J. von Neumann. *Mathematische Grundlagen der Quantenmechanik*. [Translation, Mathematical Foundations of Quantum Mechanics, Princeton University Press, 1955.] Springer-Verlag, 1932.

[57] J. H. Selby, C. M. Scandolo, and B. Coecke. "Reconstructing quantum theory from diagrammatic postulates". In: *Quantum* 5 (2021), p. 445.

[58] B. Coecke, D. Moore, and A. Wilce. "Operational quantum logic: an overview". In: *Current Research in Operational Quantum Logic: Algebras, Categories, Languages* (2000), pp. 1–36.

[59] L. Hardy. "On the theory of composition in physics". In: *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky*. arXiv:1303.1537. Springer, 2013, pp. 83–106.

[60] B. Coecke. "Kindergarten quantum mechanics". In: *Quantum Theory: Reconsiderations of the Foundations III*. Ed. by A. Khrennikov. arXiv:quant-ph/0510032. AIP Press, 2005, pp. 81–98.

[61] C. Heunen, M. Sadrzadeh, and E. Grefenstette. *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*. Oxford University Press, 2013.

[62] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi. "Foundations for near-term quantum natural language processing". In: *arXiv preprint arXiv:2012.03755* (2020).

[63] K. Meichanetzidis, S. Gogioso, G. de Felice, N. Chiappori, A. Toumi, and B. Coecke. "Quantum natural language processing on near-term quantum computers". In: *Proceedings 17th International Conference on Quantum Physics and Logic, QPL 2020, Paris, France, June 2 - 6, 2020*. Ed. by B. Valiron, S. Mansfield, P. Arrighi, and P. Panangaden. Vol. 340. EPTCS. 2020, pp. 213–229.

[64] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke. "QNLP in practice: running compositional models of meaning on a quantum computer". In: *Journal of Artificial Intelligence Research* 76 (2023), pp. 1305–1342.

[65] A. Kissinger and J. van de Wetering. "Reducing the number of non-Clifford gates in quantum circuits". In: *Physical Review A* 102.2 (2020), p. 022406.

[66] R. Duncan, A. Kissinger, S. Perdrix, and J. Van De Wetering. "Graph-theoretic simplification of quantum circuits with the ZX-calculus". In: *Quantum* 4 (2020), p. 279.

[67] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah. "Phase gadget synthesis for shallow circuits". In: *16th International Conference on Quantum Physics and Logic 2019*. Open Publishing Association. 2019, pp. 213–228.

[68] A. B. Khesin, J. Z. Lu, and P. W. Shor. "Graphical quantum Clifford-encoder compilers from the ZX calculus". In: *arXiv preprint arXiv:2301.02356* (2023).

[69] N. de Beaudrap and D. Horsman. "The ZX calculus is a language for surface code lattice surgery". In: *Quantum* 4 (2020), p. 218.

[70] C. Gidney and A. G. Fowler. “Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2\,|T\rangle$ transformation”. In: *Quantum* 3 (2019), p. 135.

[71] C. Zhao and X.-S. Gao. “Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus”. In: *Quantum* 5 (2021), p. 466.

[72] Q. Wang, R. Yeung, and M. Koch. “Differentiating and integrating ZX diagrams with applications to quantum machine learning”. In: *Quantum* 8 (2024), p. 1491.

[73] A. Kissinger and J. van de Wetering. “Universal MBQC with generalised parity-phase interactions and Pauli measurements”. In: *Quantum* 3 (2019), p. 134.

[74] T. McElvanney and M. Backens. “Complete flow-preserving rewrite rules for MBQC patterns with Pauli measurements”. In: *19th International Conference on Quantum Physics and Logic*. Open Publishing Association. 2023, pp. 66–82.

[75] B. Coecke and A. Kissinger. “Categorical quantum mechanics II: classical-quantum interaction”. In: *International Journal of Quantum Information* 14.04 (2016), p. 1640020.

[76] J. van de Wetering. “ZX-calculus for the working quantum computer scientist”. In: *arXiv preprint arXiv:2012.13966* (2020).

[77] A. Kissinger and J. van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. Preprint, 2024.

[78] B. Coecke and R. Duncan. “Interacting quantum observables”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2008, pp. 298–310.

[79] B. Coecke and R. Duncan. “Interacting quantum observables: categorical algebra and diagrammatics”. In: *New Journal of Physics* 13.4 (2011), p. 043016.

[80] M. Backens. “The ZX-calculus is complete for stabilizer quantum mechanics”. In: *New Journal of Physics* 16.9 (2014), p. 093021.

[81] M. Backens. “The ZX-calculus is complete for the single-qubit Clifford+T group”. In: *Electronic Proceedings in Theoretical Computer Science* 172 (2014), pp. 293–303.

[82] A. Hadzihasanovic. “A diagrammatic axiomatisation for qubit entanglement”. In: *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE. 2015, pp. 573–584.

[83] E. Jeandel, S. Perdrix, and R. Vilmart. “A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics”. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018, pp. 559–568.

[84] A. Hadzihasanovic, K. F. Ng, and Q. Wang. “Two complete axiomatisations of pure-state qubit quantum computing”. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018, pp. 502–511.

[85] E. Jeandel, S. Perdrix, and R. Vilmart. “Diagrammatic reasoning beyond Clifford+T quantum mechanics”. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018, pp. 569–578.

[86] R. Vilmart. “A near-minimal axiomatisation of ZX-calculus for pure qubit quantum mechanics”. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2019, pp. 1–10.

[87] M. de Visme and R. Vilmart. “Minimality in finite-dimensional ZW-calculi”. In: *33rd EACSL Annual Conference on Computer Science Logic (CSL 2025)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2025, pp. 49–1.

[88] M. Backens and A. Kissinger. "ZH: a complete graphical calculus for quantum computations involving classical non-linearity". In: *15th International Conference on Quantum Physics and Logic (QPL 2018)*. Open Publishing Association. 2019, pp. 23–42.

[89] B. Poór, Q. Wang, R. A. Shaikh, L. Yeh, R. Yeung, and B. Coecke. "Completeness for arbitrary finite dimensions of ZXW-calculus, a unifying calculus". In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2023, pp. 1–14.

[90] Q. Wang and B. Poór. "Completeness of qufinite ZXW calculus, a graphical language for mixed-dimensional quantum computing". In: *arXiv preprint arXiv:2309.13014* (2023).

[91] B. Poór, R. A. Shaikh, and Q. Wang. "ZX-calculus is complete for finite-dimensional Hilbert spaces". In: *arXiv preprint arXiv:2405.10896* (2024).

[92] N. de Beaudrap, X. Bian, and Q. Wang. "Fast and effective techniques for T-count reduction via spider nest identities". In: *15th Conference on the Theory of Quantum Computation, Communication and Cryptography*. 2020.

[93] J. van de Wetering, R. Yeung, T. Laakkonen, and A. Kissinger. "Optimal compilation of parametrised quantum circuits". In: *arXiv preprint arXiv:2401.12877* (2024).

[94] A. Kissinger. "Phase-free ZX diagrams are CSS codes (... or how to graphically grok the surface code)". In: *arXiv preprint arXiv:2204.14038* (2022).

[95] J. Huang, S. M. Li, L. Yeh, A. Kissinger, M. Mosca, and M. Vasmer. "Graphical CSS code transformation using ZX calculus". In: *Electronic Proceedings in Theoretical Computer Science* 384 (2023), pp. 1–19.

[96] A. Cowtan and S. Burton. "CSS code surgery as a universal construction". In: *Quantum* 8 (2024), p. 1344.

[97] R. Duncan and S. Perdrix. "Rewriting measurement-based quantum computations with generalised flow". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2010, pp. 285–296.

[98] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering. "There and back again: a circuit extraction tale". In: *Quantum* 5 (2021), p. 421.

[99] H. Bombin, D. Litinski, N. Nickerson, F. Pastawski, and S. Roberts. "Unifying flavors of fault tolerance with the ZX calculus". In: *Quantum* 8 (2024), p. 1379.

[100] B. Pankovich, A. Neville, A. Kan, S. Omkar, K. H. Wan, and K. Brádler. "Flexible entangled-state generation in linear optics". In: *Physical Review A* 110.3 (2024), p. 032402.

[101] A. C. Doyle. *The Sign of Four*. London: Spencer Blackett, 1890.

[102] D. Deutsch and C. Marletto. "Constructor theory of information". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2174 (2015), p. 20140540.

[103] C. Marletto. *The Science of Can and Can't: A Physicist's Journey through the Land of Counterfactuals*. Penguin, 2022.

[104] A. Gefter. *How to rewrite the laws of physics in the language of impossibility*. `https://www.quantamagazine.org/with-constructor-theory-chiara-marletto-invokes-the-impossible-20210429/`. [Accessed 01-07-2024]. 29 April 2021.

[105] C. Marletto. "Constructor theory of life". In: *Journal of The Royal Society Interface* 12.104 (2015), p. 20141226.

[106] C. Marletto. “Constructor theory of probability”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 472.2192 (2016), p. 20150883.

[107] C. Marletto. “Constructor theory of thermodynamics”. In: *arXiv preprint arXiv:1608.02625* (2017).

[108] B. Coecke. “A universe of processes and some of its guises”. In: *Deep Beauty: Understanding the Quantum World through Mathematical Innovation*. Cambridge University Press Cambridge, 2011, pp. 129–186.

[109] B. Coecke and E. O. Paquette. “Categories for the practising physicist”. In: *New Structures for Physics* (2011), pp. 173–286.

[110] B. Coecke. “An alternative gospel of structure: order, composition, processes”. In: *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*. Oxford University Press, Feb. 2013.

[111] P. A. Schilpp, ed. *Albert Einstein: Philosopher-Scientist*. Vol. 7. The Library of Living Philosophers. The Library of Living Philosophers, 1949.

[112] D. Howard. “Einstein on locality and separability”. In: *Studies in History and Philosophy of Science Part A* 16.3 (1985), pp. 171–201.

[113] N. Tibau Vidal. “On the locality of indistinguishable quantum systems”. PhD thesis. University of Oxford, 2023.

[114] D. Deutsch. “Vindication of quantum locality”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 468.2138 (2012), pp. 531–544.

[115] C. A. Bédard. “The ABC of Deutsch–Hayden descriptors”. In: *Quantum Reports* 3.2 (2021).

[116] S. M. Cohen and C. D. C. Reeve. “Aristotle’s metaphysics”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Spring 2025. Metaphysics Research Lab, Stanford University, 2025.

[117] B. Coecke and D. Pavlovic. “Quantum measurements without sums”. In: *Mathematics of Quantum Computation and Quantum Technology*. Chapman and Hall/CRC, 2007, pp. 577–620.

[118] B. Coecke, E. O. Paquette, and D. Pavlovic. “Classical and quantum structuralism”. In: *Semantic Techniques in Quantum Computation* (2008), pp. 29–69.

[119] L. Hardy. “Foliable operational structures for general probabilistic theories”. In: *Deep Beauty: Understanding the Quantum World through Mathematical Innovation*. Cambridge University Press Cambridge, UK, 2011, pp. 409–442.

[120] M. Violaris and C. Marletto. “Irreversibility beyond Landauer’s principle: is it possible to perform erasure using the quantum homogenizer?” In: *New Journal of Physics* 24.11 (2022), p. 113030.

[121] C. Marletto, V. Vedral, L. T. Knoll, F. Piacentini, E. Bernardi, E. Rebufello, A. Avella, M. Gramegna, I. P. Degiovanni, and M. Genovese. “Emergence of constructor-based irreversibility in quantum systems: theory and experiment”. In: *Physical Review Letters* 128.8 (2022), p. 080401.

[122] F. C. Andrews. “Statistical mechanics and irreversibility”. In: *Proceedings of the National Academy of Sciences* 54.1 (1965), pp. 13–20.

[123] D. Wallace. “The quantitative content of statistical mechanics”. In: *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics* 52 (2015), pp. 285–293.

[124] B. Coecke, T. Fritz, and R. W. Spekkens. “A mathematical theory of resources”. In: *Information and Computation* 250 (2016), pp. 59–86.

[125] A. N. Whitehead. *An Introduction to Mathematics*. New York: Henry Holt, 1911.

[126] R. L. Boylestad and L. Nashelsky. *Electronic Devices and Circuit Theory*. Pearson Education, 2002.

[127] E.-h. Jiang and W.-b. Jiang. “Theory of expansion Boolean algebra and its applications in CMOS VLSI digital systems”. In: *Circuits, Systems, and Signal Processing* 38 (2019), pp. 5817–5838.

[128] J. P. Uyemura. *CMOS Logic Circuit Design*. Springer, 1999.

[129] R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc. “Design of ion-implanted MOSFET’s with very small physical dimensions”. In: *IEEE Journal of Solid-State Circuits* 9.5 (1974), pp. 256–268.

[130] G. E. Moore. “Cramming more components on electronic circuits”. In: *Electronics* 38.8 (1965), p. 114.

[131] G. E. Moore. “Progress in digital integrated electronics”. In: *Electron Devices Meeting*. Vol. 21. Washington, DC. 1975, pp. 11–13.

[132] M. M. Waldrop. “The chips are down for Moore’s law”. In: *Nature News* 530.7589 (2016), p. 144.

[133] T. N. Theis and H.-S. P. Wong. “The end of Moore’s law: A new beginning for information technology”. In: *Computing in Science & Engineering* 19.2 (2017), pp. 41–50.

[134] S. Manipatruni, D. E. Nikonov, and I. A. Young. “Beyond CMOS computing with spin and polarization”. In: *Nature Physics* 14.4 (2018), pp. 338–343.

[135] Y. Imai and Y. Ohtsuka. “Optical computing based on interference fringe shifting”. In: *Optical Engineering* 25.1 (1986), pp. 98–102.

[136] P. Ambs. “Optical computing: a 60-year adventure.” In: *Advances in Optical Technologies* (2010).

[137] N. L. Kazanskiy, M. A. Butt, and S. N. Khonina. “Optical computing: status and perspectives”. In: *Nanomaterials* 12.13 (2022), p. 2171.

[138] M. Rahman, S. Khasanvis, J. Shi, and C. A. Moritz. “Wave interference functions for neuromorphic computing”. In: *IEEE Transactions on Nanotechnology* 14.4 (2015), pp. 742–750.

[139] F. Zangeneh-Nejad, D. L. Sounas, A. Alù, and R. Fleury. “Analogue computing with metamaterials”. In: *Nature Reviews Materials* 6.3 (2021), pp. 207–225.

[140] B. J. MacLennan. “The promise of analog computation”. In: *International Journal of General Systems* 43.7 (2014), pp. 682–696.

[141] C. Ostrove, B. La Cour, A. Lanham, and G. Ott. “Improving performance of an analog electronic device using quantum error correction”. In: *Journal of Physics Communications* 3.8 (2019), p. 085017.

[142] L. Belostotski, A. Uddin, A. Madanayake, and S. Mandal. “A survey of analog computing for domain-specific accelerators”. In: *Electronics* 14.16 (2025), p. 3159.

[143] D. Xu, S. Wen, and H. Luo. “Metasurface-based optical analog computing: from fundamentals to applications”. In: *Advanced Devices & Instrumentation* 2022 (2022), p. 0002.

[144] A. Khitun. “Multi-frequency magnonic logic circuits for parallel data processing”. In: *Journal of Applied Physics* 111.5 (2012), p. 054307.

[145] R. Cuykendall and D. R. Andersen. “Reversible optical computing circuits”. In: *Optics Letters* 12.7 (1987), pp. 542–544.

[146] M. Balynskiy, H. Chiang, D. Gutierrez, A. Kozhevnikov, Y. Filimonov, and A. Khitun. “Reversible magnetic logic gates based on spin wave interference”. In: *Journal of Applied Physics* 123.14 (2018).

[147] J. Atulasimha and S. Bandyopadhyay. *Nanomagnetic and Spintronic Devices for Energy-efficient Memory and Computing*. John Wiley & Sons, 2016.

[148] A. V. Chumak, V. I. Vasyuchka, A. A. Serga, and B. Hillebrands. “Magnon spintronics”. In: *Nature Physics* 11.6 (2015), pp. 453–461.

[149] M. Balynsky, H. Chiang, D. Gutierrez, A. Kozhevnikov, Y. Filimonov, and A. Khitun. “Quantum computing without quantum computers: database search and data processing using classical wave superposition”. In: *Journal of Applied Physics* 130.16 (2021).

[150] A. Imre, G. Csaba, L. Ji, A. Orlov, G. Bernstein, and W. Porod. “Majority logic gate for magnetic quantum-dot cellular automata”. In: *Science* 311.5758 (2006), pp. 205–208.

[151] H. Iwamura, M. Akazawa, and Y. Amemiya. “Single-electron majority logic circuits”. In: *IEICE Transactions on Electronics* 81.1 (1998), pp. 42–48.

[152] A. Vaysset, M. Manfrini, D. E. Nikonov, S. Manipatruni, I. A. Young, G. Pourtois, I. P. Radu, and A. Thean. “Toward error-free scaled spin torque majority gates”. In: *AIP Advances* 6.6 (2016).

[153] W. Li, Y. Yang, H. Yan, and Y. Liu. “Three-input majority logic gate and multiple input logic circuit based on DNA strand displacement”. In: *Nano Letters* 13.6 (2013), pp. 2980–2988.

[154] M. Vacca, M. Graziano, J. Wang, F. Cairo, G. Causapruno, G. Urgese, A. Biroli, and M. Zamboni. “Nanomagnet logic: an architectural level overview”. In: *Field-Coupled Nanocomputing: Paradigms, Progress, and Perspectives* (2014), pp. 223–256.

[155] D. Khokhriakov, S. Sayed, A. M. Hoque, B. Karpiak, B. Zhao, S. Datta, and S. P. Dash. “Multifunctional spin logic operations in graphene spin circuits”. In: *Physical Review Applied* 18.6 (2022), p. 064063.

[156] G. S. Rose and M. R. Stan. “A programmable majority logic array using molecular scale electronics”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.11 (2007), pp. 2380–2390.

[157] S. O. Demokritov and A. N. Slavin. *Magnonics: From Fundamentals to Applications*. Springer, 2012.

[158] S. M. Rezende. *Fundamentals of Magnonics*. Springer, 2020.

[159] A. Chumak and H. Schultheiss. “Magnonics: spin waves connecting charges, spins and photons”. In: *Journal of Physics D: Applied Physics* 50.30 (2017), p. 300201.

[160] D. Lachance-Quirion, Y. Tabuchi, A. Gloppe, K. Usami, and Y. Nakamura. “Hybrid quantum systems based on magnonics”. In: *Applied Physics Express* 12.7 (2019), p. 070101.

[161] B. Flebus, D. Grundler, B. Rana, Y. Otani, I. Barsukov, A. Barman, G. Gubbiotti, P. Landeros, J. Akerman, and U. S. Ebels. “The 2024 magnonics roadmap”. In: *Journal of Physics: Condensed Matter* (2024).

[162] A. Serga, A. Chumak, and B. Hillebrands. “YIG magnonics”. In: *Journal of Physics D: Applied Physics* 43.26 (2010), p. 264002.

[163] A. Khitun, M. Bao, and K. L. Wang. “Magnonic logic circuits”. In: *Journal of Physics D: Applied Physics* 43.26 (2010), p. 264005.

[164] A. Khitun and A. Kozhanov. “Magnonic logic devices”. In: *Nanomagnetic and Spintronic Devices for Energy-Efficient Memory and Computing* (2016), pp. 189–219.

[165] A. V. Chumak. “Magnon spintronics: fundamentals of magnon-based computing”. In: *Spintronics Handbook, Second Edition: Spin Transport and Magnetism*. CRC Press, 2019, pp. 247–302.

[166] M. Kostylev, A. Serga, T. Schneider, B. Leven, and B. Hillebrands. “Spin-wave logical gates”. In: *Applied Physics Letters* 87.15 (2005), p. 153501.

[167] T. Schneider, A. A. Serga, B. Leven, B. Hillebrands, R. L. Stamps, and M. P. Kostylev. “Realization of spin-wave logic gates”. In: *Applied Physics Letters* 92.2 (2008), p. 022505.

[168] Á. Papp, W. Porod, Á. I. Csurgay, and G. Csaba. “Nanoscale spectrum analyzer based on spin-wave interference”. In: *Scientific Reports* 7.1 (2017), pp. 1–9.

[169] A. Khitun. “Magnonic holographic devices for special type data processing”. In: *Journal of Applied Physics* 113.16 (2013), p. 164503.

[170] F. Macià, A. D. Kent, and F. C. Hoppensteadt. “Spin-wave interference patterns created by spin-torque nano-oscillators for memory and computation”. In: *Nanotechnology* 22.9 (2011), p. 095301.

[171] Á. Papp, G. Csaba, G. I. Bourianoff, and W. Porod. “Spin-wave-based computing devices”. In: *14th IEEE International Conference on Nanotechnology*. IEEE. 2014, pp. 96–99.

[172] A. Mahmoud, F. Vanderveken, F. Ciubotaru, C. Adelmann, S. Cotofana, and S. Hamdioui. “Spin wave based 4-2 compressor”. In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE. 2021, pp. 1–4.

[173] A. Mahmoud, F. Vanderveken, C. Adelmann, F. Ciubotaru, S. Cotofana, and S. Hamdioui. “Spin wave normalization toward all magnonic circuits”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.1 (2020), pp. 536–549.

[174] A. Mahmoud, N. Cucu-Laurenciu, F. Vanderveken, F. Ciubotaru, C. Adelmann, S. Cotofana, and S. Hamdioui. “Would magnonic circuits outperform CMOS counterparts?” In: *Proceedings of the Great Lakes Symposium on VLSI 2022*. 2022, pp. 309–313.

[175] Q. Wang, R. Verba, T. Brächer, F. Ciubotaru, C. Adelmann, S. D. Cotofana, P. Pirro, and A. V. Chumak. “Integrated magnonic half-adder”. In: *arXiv preprint arXiv:1902.02855* (2019).

[176] A. Mahmoud, F. Vanderveken, F. Ciubotaru, C. Adelmann, S. Cotofana, and S. Hamdioui. “Spin wave based full adder”. In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2021, pp. 1–5.

[177] U. Garlando, Q. Wang, O. V. Dobrovolskiy, A. V. Chumak, and F. Riente. “Numerical model for 32-bit magnonic ripple carry adder”. In: *IEEE Transactions on Emerging Topics in Computing* 11.3 (2023), pp. 679–688.

[178] T. Fischer, M. Kewenig, D. Bozhko, A. Serga, I. Syvorotka, F. Ciubotaru, C. Adelmann, B. Hillebrands, and A. Chumak. “Experimental prototype of a spin-wave majority gate”. In: *Applied Physics Letters* 110.15 (2017), p. 152401.

[179] O. Zografos, S. Dutta, M. Manfrini, A. Vaysset, B. Sorée, A. Naeemi, P. Raghavan, R. Lauwereins, and I. P. Radu. "Non-volatile spin wave majority gate at the nanoscale". In: *AIP Advances* 7.5 (2017).

[180] J. Reuben. "Rediscovering majority logic in the post-CMOS era: A perspective from in-memory computing". In: *Journal of Low Power Electronics and Applications* 10.3 (2020), p. 28.

[181] R. Verba, M. Carpentieri, Y.-J. Chen, I. N. Krivorotov, G. Finocchio, V. Tiberkevich, and A. Slavin. "Correction of phase errors in a spin-wave transmission line by nonadiabatic parametric pumping". In: *Physical Review Applied* 11.5 (2019), p. 054040.

[182] S. Dutta, D. E. Nikonov, S. Manipatruni, I. A. Young, and A. Naeemi. "Compact physical model for crosstalk in spin-wave interconnects". In: *IEEE Transactions on Electron Devices* 62.11 (2015), pp. 3863–3869.

[183] Y. Li, W. Zhang, V. Tyberkevych, W.-K. Kwok, A. Hoffmann, and V. Novosad. "Hybrid magnonics: physics, circuits and applications for coherent information processing". In: *arXiv preprint arXiv:2006.16158* (2020).

[184] S. Vemuru, P. Wang, and M. Niamat. "Majority logic gate synthesis approaches for post-CMOS logic circuits: a review". In: *IEEE International Conference on Electro/Information Technology*. IEEE. 2014, pp. 284–289.

[185] B. Parhami, D. Abedi, and G. Jaberipur. "Majority logic, its applications, and atomic-scale embodiments". In: *Computers & Electrical Engineering* 83 (2020), p. 106562.

[186] T. Zhang, H. Jiang, W. Liu, F. Lombardi, L. Liu, S.-B. Ko, and J. Han. "A survey of majority logic designs in emerging nanotechnologies for computing". In: *IEEE Transactions on Nanotechnology* (2023).

[187] K. Walus, G. Schulhof, G. Jullien, R. Zhang, and W. Wang. "Circuit design based on majority gates for applications with quantum-dot cellular automata". In: *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.* Vol. 2. IEEE. 2004, pp. 1354–1357.

[188] D. Fan, J. Wang, E. Wang, and S. Dong. "Propelling DNA computing with materials' power: recent advancements in innovative DNA logic computing systems and smart bio-applications". In: *Advanced Science* 7.24 (2020), p. 2001766.

[189] N. Kanazawa, T. Goto, K. Sekiguchi, A. B. Granovsky, C. A. Ross, H. Takagi, Y. Nakamura, H. Uchida, and M. Inoue. "The role of Snell's law for a magnonic majority gate". In: *Scientific Reports* 7.1 (2017), p. 7898.

[190] S. Dutta, O. Zografos, S. Gurunarayanan, I. Radu, B. Soree, F. Catthoor, and A. Naeemi. "Proposal for nanoscale cascaded plasmonic majority gates for non-Boolean computation". In: *Scientific reports* 7.1 (2017), p. 17866.

[191] B. Coecke, M. Sadrzadeh, and S. Clark. "Mathematical foundations for a compositional distributional model of meaning". In: *A Festschrift for Jim Lambek*. Ed. by J. van Benthem, M. Moortgat, and W. Buszkowski. Vol. 36. Linguistic Analysis. arxiv:1003.4394. 2010, pp. 345–384.

[192] M. Sadrzadeh, S. Clark, and B. Coecke. "The Frobenius anatomy of word meanings I: subject and object relative pronouns". In: *Journal of Logic and Computation* 23 (2013). arXiv:1404.5278, pp. 1293–1317.

[193] T. Laakkonen, K. Meichanetzidis, and B. Coecke. "Quantum algorithms for compositional text processing". In: *arXiv preprint arXiv:2408.06061* (2024).

[194] T. Duneau, S. Bruhn, G. Matos, T. Laakkonen, K. Saiti, A. Pearson, K. Meichanetzidis, and B. Coecke. “Scalable and interpretable quantum natural language processing: an implementation on trapped ions”. In: *arXiv preprint arXiv:2409.08777* (2024).

[195] V. Wang-Maścianica and B. Coecke. “Talking space: inference from spatial linguistic meanings”. In: *arXiv preprint arXiv:2109.06554* (2021).

[196] B. Rodatz, R. Shaikh, and L. Yeh. “Conversational negation using worldly context in compositional distributional semantics”. In: *Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace)*. 2021, pp. 53–65.

[197] R. A. Shaikh, L. Yeh, B. Rodatz, and B. Coecke. “Composing conversational negation”. In: *Proceedings of the Fourth International Conference on Applied Category Theory, ACT 2021, Cambridge, United Kingdom, 12-16th July 2021*. Ed. by K. Kishida. Vol. 372. EPTCS. 2021, pp. 352–367.

[198] T. Duneau. “Solving logical puzzles in DisCoCirc”. In: *Proceedings of the 2020 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace)*. 2020, pp. 355–389.

[199] T. Duneau. “Parsing conjunctions in DisCoCirc”. In: *Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace)*. 2021, pp. 66–75.

[200] H. Kamp and U. Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Springer, 2013.

[201] P. S. Peters Jr and R. W. Ritchie. “On the generative power of transformational grammars”. In: *Information sciences* 6 (1973), pp. 49–83.

[202] J. Lambek. *From Word to Sentence: A Computational Algebraic Approach to Grammar*. Polimetrica, 2008.

[203] L. Tesnière. *Elements of Structural Syntax*. John Benjamins Publishing Company, 2015.

[204] B. Coecke and V. Wang. “Grammar equations”. In: *arXiv preprint arXiv:2106.07485* (2021).

[205] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[206] J. C. Baez and M. Stay. “Physics, topology, logic and computation: a Rosetta Stone”. In: *New Structures for Physics*. Ed. by B. Coecke. Lecture Notes in Physics. Springer, 2011, pp. 95–172.

[207] M. Kapović. “Indo-European languages: Introduction”. In: *The Indo-European languages,* (2017), pp. 1–9.

[208] M. Jaeggli and K. Safir. *The Null Subject Parameter*. Springer, 2012.

[209] M. Kenstowicz. “The null subject parameter in modern Arabic dialects”. In: *The Null Subject Parameter*. Springer, 1989, pp. 263–275.

[210] A. S. Eddington. *The Philosophy of Physical Science*. New York: The Macmillan Company, 1939.

[211] S. Plesnik and M. Violaris. “Impossibility of universal work extraction from coherence: reconciling axiomatic and resource-theory approaches”. In: *New Journal of Physics* 26.10 (2024), p. 103019.

[212] S. K. Bhatt. “Teaching Hindi and Urdu as Hindi-Urdu”. In: *Electronic Journal of Foreign Language Teaching* 15.1 (2018), pp. 179–191.

[213] H. Kamp, J. Van Genabith, and U. Reyle. “Discourse representation theory”. In: *Handbook of Philosophical Logic: Volume 15*. Springer, 2010, pp. 125–394.

[214] A. N. Whitehead. *Process and Reality*. New York: Macmillan, 1929.

[215] S. Eilenberg and S. MacLane. “General theory of natural equivalences”. In: *Transactions of the American Mathematical Society* 58.2 (1945), pp. 231–294.

[216] S. Mac Lane. *Categories for the Working Mathematician*. New York: Springer, 1978.

[217] C. Heunen and J. Vicary. *Categories for Quantum Theory: An Introduction*. Oxford University Press, 2019.

[218] J. Baez, S. Cho, D. Cicala, N. Otter, and V. de Paiva. “Applied category theory in chemistry, computing, and social networks”. In: *Notices of the American Mathematical Society* 69.2 (2022).

[219] J. C. Baez and B. S. Pollard. “A compositional framework for reaction networks”. In: *Reviews in Mathematical Physics* 29.09 (2017), p. 1750028.

[220] F. Genovese and D. I. Spivak. “A categorical semantics for guarded Petri nets”. In: *International Conference on Graph Transformation*. Springer. 2020, pp. 57–74.

[221] T.-D. Bradley. “Entropy as a topological operad derivation”. In: *Entropy* 23.9 (2021), p. 1195.

[222] J. C. Baez and B. Fong. “A compositional framework for passive linear networks”. In: *Theory and Applications of Categories* 33.38 (2018), pp. 1158–1222.

[223] J. C. Baez, O. Lynch, and J. Moeller. “Compositional thermostatics”. In: *Journal of Mathematical Physics* 64.2 (2023).

[224] A. Broadbent. “Categorical composable cryptography”. In: *Foundations of Software Science and Computation Structures: 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2–7, 2022, Proceedings*. Vol. 13242. Springer Nature. 2022, p. 161.

[225] D. Pavlovic. “Chasing diagrams in cryptography”. In: *Categories and Types in Logic, Language, and Physics: Essays Dedicated to Jim Lambek on the Occasion of His 90th Birthday*. Springer, 2014, pp. 353–367.

[226] K. Sturtz. “Categorical probability theory”. In: *arXiv preprint arXiv:1406.6030* (2014).

[227] B. Milewski. *Category Theory for Programmers*. Blurb, 2019.

[228] B. Gavranović. “Fundamental components of deep learning: a category-theoretic approach”. In: *arXiv preprint arXiv:2403.13001* (2024).